Convex Sets

Pontus Giselsson

Outline

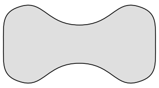
- **Definition and convex hull**
- Examples of convex sets
- Convexity preserving operations
- Concluding convexity – Examples
- Separating and supporting hyperplanes

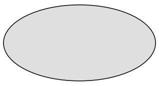
Convex sets – Definition

- A set C is convex if for every $x, y \in C$ and $\theta \in [0, 1]$:

$$\theta x + (1 - \theta)y \in C$$

- "Every line segment that connect any two points in C is in C"


Nonconvex


Convex

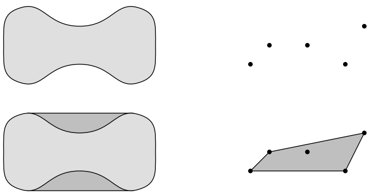

Nonconvex


Nonconvex

- Will assume that all sets are nonempty and closed

Convex combination and convex hull

Convex hull ($\mathrm{conv}S$) of S is smallest convex set that contains S:



Mathematical construction:

- Convex combinations of $x_1, \ldots, x_k$ are all points x of the form

$$x = \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_k x_k$$

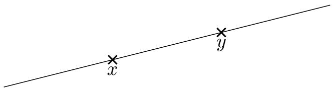
where $\theta_1 + \ldots + \theta_k = 1$ and $\theta_i \geq 0$
- Convex hull: set of all convex combinations of points in S

Outline

- Definition and convex hull
- **Examples of convex sets**
- Convexity preserving operations
- Concluding convexity – Examples
- Separating and supporting hyperplanes

Affine sets

- Take any two points $x, y \in V$: V is affine if full line in V:



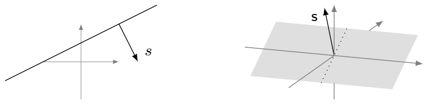
Lines and planes are affine sets
- Definition: A set V is affine if for every $x, y \in V$ and $\alpha \in \mathbb{R}$:

$$\alpha x + (1 - \alpha)y \in V \tag{1}$$

hence convex this holds in particular for $\alpha \in [0, 1]$

Affine hyperplanes

- Affine hyperplanes in $\mathbb{R}^n$ are affine sets that cut $\mathbb{R}^n$ in two halves



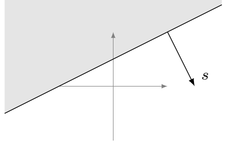
- Dimension of affine hyperplane in $\mathbb{R}^n$ is $n - 1$ (If $s \neq 0$)
- All affine sets in $\mathbb{R}^n$ of dimension $n - 1$ are hyperplanes
- Mathematical definition:

$$h_{s,r} := \{x \in \mathbb{R}^n : s^T x = r\}$$

where $s \in \mathbb{R}^n$ and $r \in \mathbb{R}$, i.e., defined by one *affine function*
- Vector s is called normal to hyperplane

Halfspaces

- A halfspace is one of the halves constructed by a hyperplane



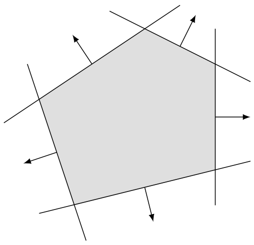
- Mathematical definition:

$$H_{r,s} = \{x \in \mathbb{R}^n : s^T x \leq r\}$$

- Halfspaces are convex, and vector s is called normal to halfspace

Polytopes

- A *polytope* is intersection of halfspaces and hyperplanes



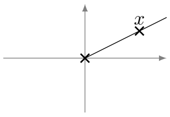
- Mathematical representation:

$$C = \{x \in \mathbb{R}^n : s_i^T x \le r_i \text{ for } i \in \{1,\ldots,m\} \text{ and } s_i^T x = r_i \text{ for } i \in \{m+1,...,p\}\}$$

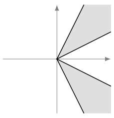
- Polytopes convex since intersection of convex sets

Cones

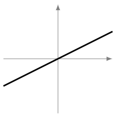
- A set K is a cone if for all $x \in K$ and $\alpha \ge 0$: $\alpha x \in K$
- If x is in cone K, so is entire ray from origin passing through x:



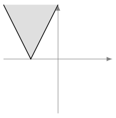
- Examples:



Cone



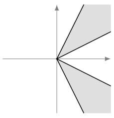
Cone



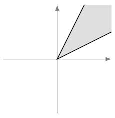
Not cone

Convex cones

- Cones can be convex or nonconvex:



Nonconvex cone



Convex cone

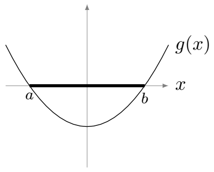
- Convex cone examples:
  - Linear subspaces $\{x \in \mathbb{R}^n : Ax = 0\}$ (but not affine subspaces)
  - Halfspaces based on linear (not affine) hyperplanes $\{x : s^T x \le 0\}$
  - Positive semi-definite matrices $\{X \in \mathbb{R}^{n\times n} : X \text{ symmetric and } z^T X z \ge 0 \text{ for all } z \in \mathbb{R}^n\}$
  - Nonnegative orthant $\{x \in \mathbb{R}^n : x \ge 0\}$
  - Second order cone $\{(x, r) \in \mathbb{R}^n \times \mathbb{R} : \|x\|_2 \le r\}$

Sublevel sets

- Suppose that $g : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function
- The ($0th$) sublevel set of g is defined as

$$S := \{x \in \mathbb{R}^n : g(x) \le 0\}$$

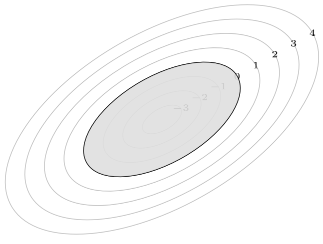
- Example: construction giving 1D interval $S = [a, b]$



- S is a convex set if g is a convex function
- S is not necessarily nonconvex although g is

Sublevel sets – Examples

- Levelset of convex quadratic function



$\{x \in \mathbb{R}^n : \frac{1}{2}x^T P x + q^T x + r \le 0\}$, with P positive definite
- Norm balls $\{x \in \mathbb{R}^n : \|x\| - r \le 0\}$
- Second-order cone $\{(x, r) \in \mathbb{R}^n \times \mathbb{R} : \|x\|_2 - r \le 0\}$
- Halfspaces $\{x \in \mathbb{R}^n : c^T x - r \le 0\}$

Outline

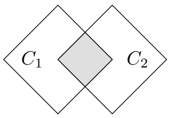
- Definition and convex hull
- Examples of convex sets
- **Convexity preserving operations**
- Concluding convexity – Examples
- Separating and supporting hyperplanes

Convexity preserving operations

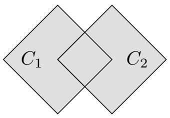
- Intersection (but not union)
- Affine image and inverse affine image of a set

Intersection and union

- Intersection $C = C_1 \cap C_2$ means $x \in C$ if $x \in C_1$ **and** $x \in C_2$
- Union $C = C_1 \cup C_2$ means $x \in C$ if $x \in C_1$ **or** $x \in C_2$



Intersection



Union

- Intersection of any number of, e.g., infinite, convex sets is convex
- Union of convex sets need not be convex

Image sets and inverse image sets

- Let $L(x) = Ax + b$ be an affine mapping defined by
  - matrix $A \in \mathbb{R}^{m\times n}$
  - vector $b \in \mathbb{R}^m$
- Let C be a convex set in $\mathbb{R}^n$ then the *image set of C under L*

$$\{Ax + b : x \in C\}$$

  is convex
- Let D be a convex set in $\mathbb{R}^m$ then the *inverse image of D under L*

$$\{x : Ax + b \in D\}$$

  is convex

Outline

- Definition and convex hull
- Examples of convex sets
- Convexity preserving operations
- **Concluding convexity – Examples**
- Separating and supporting hyperplanes

Ways to conclude convexity

- Use convexity definition
- Show that set is sublevel set of a convex function
- Show that set constructed by convexity preserving operations

Example – Nonnegative orthant

- Nonnegative orthant is set $C = \{x \in \mathbb{R}^n : x \geq 0\}$
- Prove convexity from definition:
  - Let $x \geq 0$ and $y \geq 0$ be arbitrary points in C
  - For all $\theta \in [0, 1]$:

$$\theta x \geq 0 \qquad \text{and} \qquad (1-\theta)y \geq 0$$

  - All convex combinations therefore also satisfy

$$\theta x + (1-\theta)y \geq 0$$

    i.e., they belongs to C and the set is convex

Example – Positive semidefinite cone

- The positive semidefinite (PSD) cone is

  $\{X \in \mathbb{R}^{n\times n} : X \text{ symmetric}\} \bigcap \{X \in \mathbb{R}^{n\times n} : z^TXz \geq 0 \text{ for all } z \in \mathbb{R}^n\}$
- This can be written as the following intersection over all $z \in \mathbb{R}^n$

$$\{X \in \mathbb{R}^{n\times n} : X \text{ symmetric}\} \bigcap_{z\in\mathbb{R}^n} \{X \in \mathbb{R}^{n\times n} : z^TXz \geq 0\}$$

  which, by noting that $z^TXz = \operatorname{tr}(z^TXz) = \operatorname{tr}(zz^TX)$, is equal to

$$\{X \in \mathbb{R}^{n\times n} : X \text{ symmetric}\} \bigcap_{z\in\mathbb{R}^n} \{X \in \mathbb{R}^{n\times n} : \operatorname{tr}(zz^TX) \geq 0\}$$

  where $\operatorname{tr}(zz^TX) \geq 0$ is a halfspace in $\mathbb{R}^{n\times n}$ (except when $z = 0$)
- The PSD cone is convex since it is intersection of
  - symmetry set, which is a finite set of (convex) linear equalities
  - an infinite number of (convex) halfspaces in $\mathbb{R}^{n\times n}$
- Notation: If X belong to the PSD cone, we write $X \succeq 0$

Example – Linear matrix inequality

- Let us consider a linear matrix inequality (LMI) of the form

$$\{x \in \mathbb{R}^k : A + \sum_{i=1}^{k} x_iB_i \succeq 0\}$$

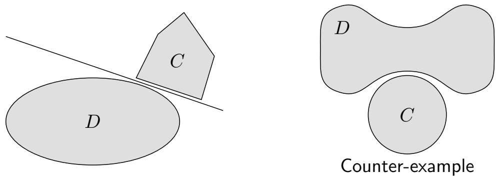
  where A and B_i are fixed matrices in $\mathbb{R}^{n\times n}$
- Convex since inverse image of PSD cone under affine mapping

Outline

- Definition and convex hull
- Examples of convex sets
- Convexity preserving operations
- Concluding convexity – Examples
- **Separating and supporting hyperplanes**

Separating hyperplane theorem

- Suppose that $C, D \subseteq \mathbb{R}^n$ are two non-intersecting convex sets
- Then there exists hyperplane with C and D in opposite halves



Example

Counter-example
D nonconvex

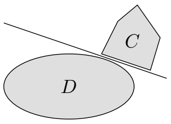
- Mathematical formulation: There exists $s \neq 0$ and r such that

$$\begin{aligned} s^Tx &\leq r \qquad && \text{for all } x \in C \\ s^Tx &\geq r && \text{for all } x \in D \end{aligned}$$

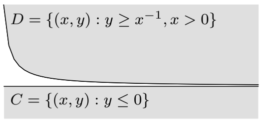
- The hyperplane $\{x : s^Tx = r\}$ is called *separating hyperplane*

A strictly separating hyperplane theorem

- Suppose that $C, D \subseteq \mathbb{R}^n$ are non-intersecting closed and convex sets and that one of them is compact (closed and bounded)
- Then there exists hyperplane with strict separation



Example



Counter example
C, D not compact

- Mathematical formulation: There exists $s \neq 0$ and r such that

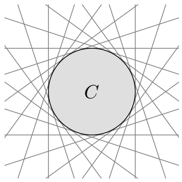
$$s^T x < r \qquad \text{for all } x \in C$$
$$s^T x > r \qquad \text{for all } x \in D$$

Consequence – C is intersection of halfspaces

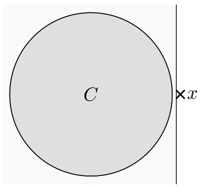
a closed convex set C is the intersection of all halfspaces that contain it

proof:

- let H be the intersection of all halfspaces containing C
- $\Rightarrow$: obviously $x \in C \Rightarrow x \in H$
- $\Leftarrow$: assume $x \notin C$, since C closed and convex and $\{x\}$ compact singleton, there exists a strictly separating hyperplane, i.e., $x \notin H$:



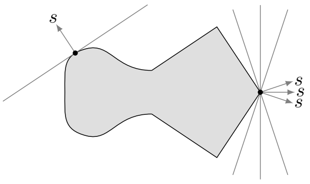
$\Rightarrow$



$\Leftarrow$

Supporting hyperplanes

- Supporting hyperplanes touch set and have full set on one side:



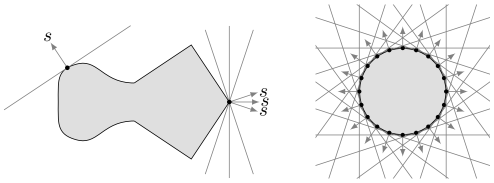
- We call the halfspace that contains the set *supporting halfspace*
- s is called *normal vector* to C at x
- Definition: Hyperplane $\{y : s^T y = r\}$ supports C at $x \in$ bd C if

$$s^T x = r \qquad \text{and} \qquad s^T y \leq r \text{ for all } y \in C$$

Supporting hyperplane theorem

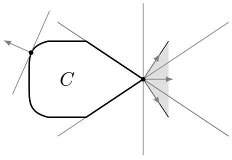
Let C be a nonempty convex set and let $x \in \operatorname{bd}(C)$. Then there exists a supporting hyperplane to C at x.

- Does not exist for all point on boundary for nonconvex sets
- Many supporting hyperplanes exist for points of nonsmoothness



Normal cone operator

- Normal cone to C at $x \in \operatorname{bd}(C)$ is set of normals at x



- Normal cone operator N_C to C takes point input and returns set:
  - $x \in \operatorname{bd}(C) \cap C$: set of normal vectors to supporting halfspaces
  - $x \in \operatorname{int}(C)$: returns zero set $\{0\}$
  - $x \notin C$: returns emptyset $\emptyset$
- Mathematical definition: The normal cone operator to a set C is

$$N_C(x) = \begin{cases} \{s : s^T(y - x) \leq 0 \text{ for all } y \in C\} & \text{if } x \in C \\ \emptyset & \text{else} \end{cases}$$

i.e., vectors that form obtuse angle between s and all $y - x$, $y \in C$

- For all $x \in C$: the N_C outputs a set that contains 0

Convex Functions

Pontus Giselsson

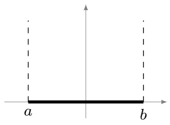
Outline

- **Definition, epigraph, convex envelope**
- First- and second-order conditions for convexity
- Convexity preserving operations
- Concluding convexity – Examples
- Strict and strong convexity
- Smoothness

Extended-valued functions and domain

- We consider extended-valued functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\} =: \overline{\mathbb{R}}$
- Example: Indicator function of interval $[a, b]$

$$\iota_{[a,b]}(x) = \begin{cases} 0 & \text{if } a \leq x \leq b \\ \infty & \text{else} \end{cases}$$



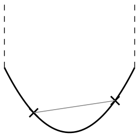
- The (effective) domain of $f\ :\ \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is the set

$$\text{dom } f = \{x \in \mathbb{R}^n : f(x) < \infty\}$$

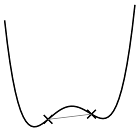
- (Will always assume $\text{dom} f \neq \emptyset$, this is called proper)

Convex functions

- Graph below line connecting any two pairs $(x, f(x))$ and $(y, f(y))$



convex function



nonconvex function

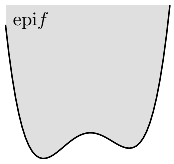
- Function $f\ :\ \mathbb{R}^n \to \overline{\mathbb{R}}$ is *convex* if for all $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y)$$

(in extended valued arithmetics)
- A function f is *concave* if $-f$ is convex

Epigraphs

- The *epigraph* of a function f is the set of points above graph



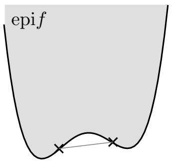
- Mathematical definition:

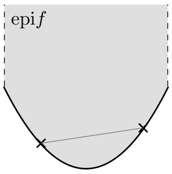
$$\text{epi} f = \{(x, r) \mid f(x) \leq r\}$$

- The epigraph is a set in $\mathbb{R}^n \times \mathbb{R}$

Epigraphs and convexity

- Let $f\ :\ \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$
- Then f is convex if and only if $\text{epi} f$ is a convex set in $\mathbb{R}^n \times \mathbb{R}$

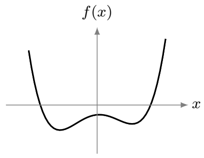


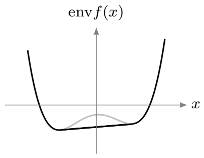


- f is called closed (lower semi-continuous) if $\text{epi} f$ is closed set

Convex envelope

- Convex envelope of f is largest convex minorizer



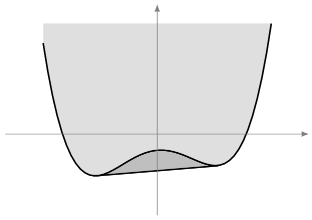


- Definition: The convex envelope $\text{env} f$ satisfies: $\text{env} f$ convex,

$$\text{env} f \leq f \qquad \text{and} \qquad \text{env} f \geq g \text{ for all convex } g \leq f$$

Convex envelope and convex hull

- Assume $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed
- Epigraph of convex envelope of f is closed convex hull of $\text{epi} f$



- $\text{epi} f$ in light gray, $\text{epi}\,\text{env} f$ includes dark gray

Outline

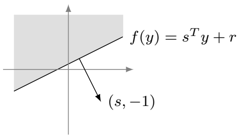
- Definition, epigraph, convex envelope
- **First- and second-order conditions for convexity**
- Convexity preserving operations
- Concluding convexity – Examples
- Strict and strong convexity
- Smoothness

Affine functions

- Affine functions $f : \mathbb{R}^n \to \mathbb{R}$ are of the form

$$f(y) = s^T y + r$$

- Affine functions $f : \mathbb{R}^n \to \mathbb{R}$ cut $\mathbb{R}^n \times \mathbb{R}$ in two halves



- s defines slope of function
- Upper halfspace is epigraph with normal vector $(s, -1)$:

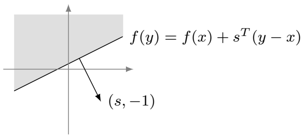
$$\text{epi} f = \{(y, t) : t \geq s^T y + r\} = \{(y, t) : (s, -1)^T (y, t) \leq -r\}$$

Affine functions – Reformulation

- Pick any fixed $x \in \mathbb{R}^n$; affine $f(y) = s^T y + r$ can be written as

$$f(y) = f(x) + s^T (y - x)$$

(since $r = f(x) - s^T x$)



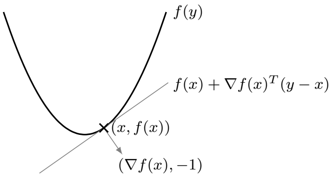
- Affine function of this form is important in convex analysis

First-order condition for convexity

- A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$

for all $x, y \in \mathbb{R}^n$



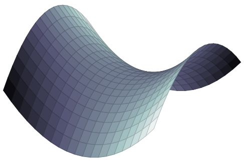
- Function f has for all $x \in \mathbb{R}^n$ an affine minorizer that:
  - coincides with function f at x
  - has slope s defined by ∇f, which coincides the function slope
  - is supporting hyperplane to epigraph of f
  - defines normal $(\nabla f(x), -1)$ to epigraph of f

Second-order condition for convexity

- A twice differentiable function is convex if and only if

$$\nabla^2 f(x) \succeq 0$$

for all $x \in \mathbb{R}^n$ (i.e., the Hessian is positive semi-definite)
- "The function has non-negative curvature"
- Nonconvex example: $f(x) = x^T \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} x$ with $\nabla^2 f(x) \not\succeq 0$



Outline

- Definition, epigraph, convex envelope
- First- and second-order conditions for convexity
- **Convexity preserving operations**
- Concluding convexity – Examples
- Strict and strong convexity
- Smoothness

Operations that preserve convexity

- Positive sum
- Marginal function
- Supremum of family of convex functions
- Composition rules
- Prespective of convex function

Positive sum

- Assume that f_j are convex for all $j \in \{1, \ldots, m\}$
- Assume that there exists x such that $f_j(x) < \infty$ for all j
- Then the positive sum

$$f = \sum_{j=1}^{m} t_j f_j$$

with $t_j > 0$ is convex

Marginal function

- Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ be convex
- Define the marginal function

$$g(x) := \inf_y f(x, y)$$

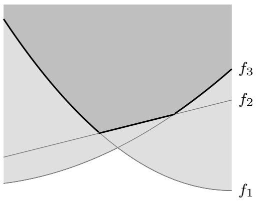
- The marginal function g is convex if f is

Supremum of convex functions

- Point-wise supremum of convex functions from family $\{f_j\}_{j \in J}$:

$$f(x) := \sup\{f_j(x) \ : \ j \in J\}$$

- Supremum is over functions in family for fixed x
- Example:



- Convex since epigraph is intersection of convex epigraphs

Scalar composition rule

- Consider the function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ defined as

$$f(x) = h(g(x))$$

  where $h : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ is convex and $g : \mathbb{R}^n \to \mathbb{R}$
- Suppose that one of the following holds:
  - h is nondecreasing and g is convex
  - h is nonincreasing and g is concave
  - g is affine

  Then f is convex

Vector composition rule

- Consider the function $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ defined as

$$f(x) = h(g_1(x), g_2(x), \ldots, g_k(x))$$

  where $h : \mathbb{R}^k \to \mathbb{R} \cup \{\infty\}$ is convex and $g_i : \mathbb{R}^n \to \mathbb{R}$
- Suppose that for each $i \in \{1, \ldots, k\}$ one of the following holds:
  - h is nondecreasing in the ith argument and g_i is convex
  - h is nonincreasing in the ith argument and g_i is concave
  - g_i is affine

  Then f is convex

Perspective of function

Let

- $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be convex
- t be positive, i.e, $t \in \mathbb{R}_+$

then the perspective function $g : \mathbb{R}^n \times \mathbb{R} \to \overline{\mathbb{R}}$, defined by

$$g(x, t) := \begin{cases} tf(x/t) & \text{if } t > 0 \\ \infty & \text{else} \end{cases}$$

is convex

Outline

- Definition, epigraph, convex envelope
- First- and second-order conditions for convexity
- Convexity preserving operations
- **Concluding convexity – Examples**
- Strict and strong convexity
- Smoothness

Ways to conclude convexity

- Use convexity definition
- Show that epigraph is convex set
- Use first or second order condition for convexity
- Show that function constructed by convexity preserving operations

Conclude convexity – Some examples

- From definition:
  - indicator function of convex set C

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{else} \end{cases}$$

  - norms: $\|x\|$
- From first- or second-order conditions:
  - affine functions: $f(x) = s^T x + r$
  - quadratics: $f(x) = \frac{1}{2} x^T Q x$ with Q positive semi-definite matrix
- From convex epigraph:
  - matrix fractional function: $f(x, Y) = \begin{cases} x^T Y^{-1} x & \text{if } Y \succ 0 \\ \infty & \text{else} \end{cases}$
- From marginal function:
  - (shortest) distance to convex set C: $\text{dist}_C(x) = \inf_{y \in C}(\|y - x\|)$

Example – Convexity of norms

Show that $f(x) := \|x\|$ is convex from convexity definition

- Norms satisfy the triangle inequality

$$\|u + v\| \leq \|u\| + \|v\|$$

- For arbitrary x, y and $\theta \in [0, 1]$:

$$\begin{aligned} f(\theta x + (1-\theta)y) &= \|\theta x + (1-\theta)y\| \\ &\leq \|\theta x\| + \|(1-\theta)y\| \\ &= \theta\|x\| + (1-\theta)\|y\| \\ &= \theta f(x) + (1-\theta)f(y) \end{aligned}$$

  which is definition of convexity
- Proof uses triangle inequality and $\theta \in [0, 1]$

Example – Matrix fractional function

Show that the matrix fractional function is convex via its epigraph

- The matrix fractional function

$$f(x, Y) = \begin{cases} x^T Y^{-1} x & \text{if } Y \succ 0 \\ \infty & \text{else} \end{cases}$$

- The epigraph satisfies

$$\begin{aligned} \text{epi} f(x, Y, t) &= \{(x, Y, t) : f(x, Y) \leq t\} \\ &= \{(x, Y, t) : x^T Y^{-1} x \leq t \text{ and } Y \succ 0\} \end{aligned}$$

- Schur complement condition says for $Y \succ 0$ that

$$x^T Y^{-1} x \leq t \quad \Leftrightarrow \quad \begin{bmatrix} Y & x \\ x^T & t \end{bmatrix} \succeq 0$$

  which is a (convex) linear matrix inequality (LMI) in (x, Y, t)
- Epigraph is intersection between LMI and positive definite cone

Example – Composition with matrix

- Let
  - $f : \mathbb{R}^m \to \overline{\mathbb{R}}$ be convex
  - $L \in \mathbb{R}^{m \times n}$ be a matrix

  then composition with a matrix

$$(f \circ L)(x) := f(Lx)$$

  is convex
- Vector composition with convex function and affine mappings

Example – Image of function under linear mapping

- Let
  - $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be convex
  - $L \in \mathbb{R}^{m \times n}$ be a matrix

  then image function (sometimes called infimal postcomposition)

$$(Lf)(x) := \inf_y \{f(y) \ : \ Ly = x\}$$

  is convex
- Proof: Define

$$h(x, y) = f(y) + \iota_{\{0\}}(Ly - x)$$

  which is convex in (x, y), then

$$(Lf)(x) = \inf_y h(x, y)$$

  which is convex since marginal of convex function

Example – Nested composition

Show that: $f(x) := e^{\|Lx-b\|_2^3}$ is convex where L is matrix b vector:

- Let

$$g_1(u) = \|u\|_2, \qquad g_2(u) = \begin{cases} 0 & \text{if } u < 0 \\ u^3 & \text{if } u \geq 0 \end{cases}, \qquad g_3(u) = e^u$$

  then $f(x) = g_3(g_2(g_1(Lx - b)))$
- $g_1(Lx - b)$ convex: convex g_1 and $Lx - b$ affine
- $g_2(g_1(Lx - b))$ convex: cvx nondecreasing g_2 and cvx $g_1(Lx - b)$
- $f(x)$ convex: convex nondecreasing g_3 and convex $g_2(g_1(Lx - b))$

Example – Conjugate function

Show that the *conjugate* $f^*(s) := \sup_{x \in \mathbb{R}^n} (s^T x - f(x))$ is convex:

- Define (uncountable) index set J and x_j such that $\cup_{j \in J} x_j = \mathbb{R}^n$
- Define $r_j := f(x_j)$ and affine (in s): $a_j(s) := s^T x_j - r_j$
- Therefore $f^*(s) = \sup(a_j(s) : j \in J)$
- Convex since supremum over family of convex (affine) functions
- Note convexity of f^* not dependent on convexity of f

Outline

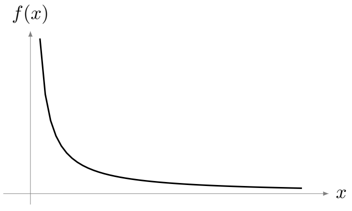
- Definition, epigraph, convex envelope
- First- and second-order conditions for convexity
- Convexity preserving operations
- Concluding convexity – Examples
- **Strict and strong convexity**
- Smoothness

Strict convexity

- A function is strictly convex if

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y)$$

  for all $x \neq y$ and $\theta \in (0, 1)$
- Convexity definition with strict inequality
- No flat (affine) regions
- Example: $f(x) = 1/x$ for $x > 0$

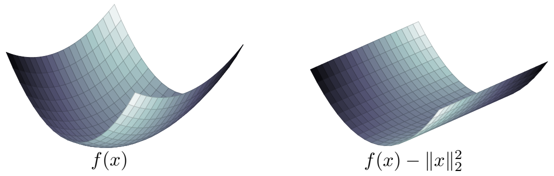


Strong convexity

- Let $\sigma > 0$
- A function f is *σ-strongly convex* if $f - \frac{\sigma}{2}\|\cdot\|_2^2$ is convex
- Alternative equivalent definition of σ-strong convexity:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) - \frac{\sigma}{2}\theta(1-\theta)\|x-y\|^2$$

  holds for every $x, y \in \mathbb{R}^n$ and $\theta \in [0,1]$
- Strongly convex functions are strictly convex and convex
- Example: f 2-strongly convex since $f - \|\cdot\|_2^2$ convex:



Uniqueness of minimizers

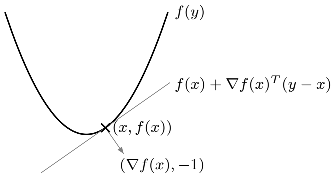
- Strictly (strongly) convex functions have unique minimizers
- Strictly convex functions may not have a minimizing point
- Strongly convex functions always have a unique minimizing point

First-order condition for strict convexity

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable
- f is strictly convex if and only if

$$f(y) > f(x) + \nabla f(x)^T(y-x)$$

  for all $x, y \in \mathbb{R}^n$ where $x \neq y$



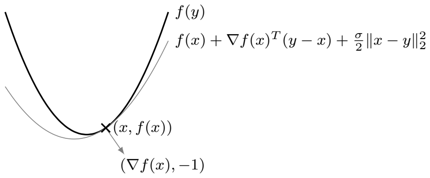
- Function f has for all $x \in \mathbb{R}^n$ an affine minorizer that:
  - has slope s defined by ∇f
  - coincides with function f *only* at x
  - is supporting hyperplane to epigraph of f
  - defines normal $(\nabla f(x), -1)$ to epigraph of f

First-order condition for strong convexity

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable
- f is σ-strongly convex with $\sigma > 0$ if and only if

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \frac{\sigma}{2}\|x-y\|_2^2$$

  for all $x, y \in \mathbb{R}^n$



- Function f has for all $x \in \mathbb{R}^n$ a quadratic minorizer that:
  - has curvature defined by σ
  - coincides with function f at x
  - defines normal $(\nabla f(x), -1)$ to epigraph of f

Second-order condition for strict/strong convexity

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable

- f is strictly convex if

$$\nabla^2 f(x) \succ 0$$

  for all $x \in \mathbb{R}^n$ (i.e., the Hessian is positive definite)
- f is σ-strongly convex if and only if

$$\nabla^2 f(x) \succeq \sigma I$$

  for all $x \in \mathbb{R}^n$

Examples of strictly/strongly convex functions

Strictly convex

- $f(x) = -\log(x) + \iota_{>0}(x)$
- $f(x) = 1/x + \iota_{>0}(x)$
- $f(x) = e^{-x}$

Strongly convex

- $f(x) = \frac{\lambda}{2}\|x\|_2^2$
- $f(x) = \frac{1}{2}x^TQx$ where Q positive definite
- $f(x) = f_1(x) + f_2(x)$ where f_1 strongly convex and f_2 convex
- $f(x) = f_1(x) + f_2(x)$ where f_1, f_2 strongly convex
- $f(x) = \frac{1}{2}x^TQx + \iota_C(x)$ where Q positive definite and C convex

Proofs for two examples

Strict convexity of $f(x) = e^{-x}$:

- $\nabla f(x) = -e^{-x}$, $\nabla^2 f(x) = e^{-x} > 0$ for all $x \in \mathbb{R}$

Strong convexity of $f(x) = \frac{1}{2}x^TQx$ with Q positive definite

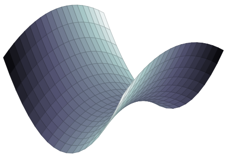
- $\nabla f(x) = Qx$, $\nabla^2 f(x) = Q \succeq \lambda_{\min}(Q)I$ where $\lambda_{\min}(Q) > 0$

Outline

- Definition, epigraph, convex envelope
- First- and second-order conditions for convexity
- Convexity preserving operations
- Concluding convexity – Examples
- Strict and strong convexity
- **Smoothness**

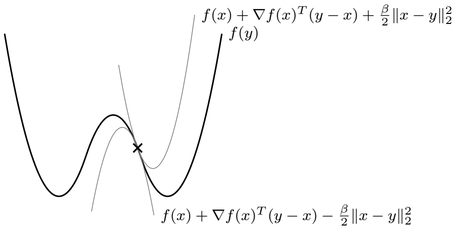
Smoothness

- A function is called β-*smooth* if its gradient is β-Lipschitz:
$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \beta \|x - y\|_2$$
for all $x, y \in \mathbb{R}^n$ (it is not necessarily convex)
- Alternative equivalent definition of β-smoothness
$$f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta) f(y) - \tfrac{\beta}{2}\theta(1-\theta)\|x-y\|^2$$
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y) + \tfrac{\beta}{2}\theta(1-\theta)\|x-y\|^2$$
hold for every $x, y \in \mathbb{R}^n$ and $\theta \in [0,1]$
- Smoothness does not imply convexity
- Example:



First-order condition for smoothness

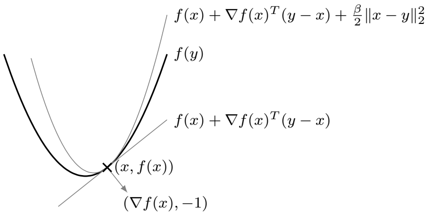
- f is β-smooth with $\beta \geq 0$ if and only if
$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) - \tfrac{\beta}{2}\|x-y\|_2^2$$
for all $x, y \in \mathbb{R}^n$



- Quadratic upper/lower bounds with curvatures defined by β
- Quadratic bounds coincide with function f at x

First-order condition for smooth convex

- f is β-smooth with $\beta \geq 0$ and convex if and only if
$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x)$$
for all $x, y \in \mathbb{R}^n$



- Quadratic upper bounds and affine lower bound
- Bounds coincide with function f at x
- Quadratic upper bound is called *descent lemma*

Second-order condition for smoothness

Let $f : \mathbb{R}^n \to \mathbb{R}$ be twice differentiable

- f is β-smooth if and only if
$$-\beta I \preceq \nabla^2 f(x) \preceq \beta I$$
for all $x \in \mathbb{R}^n$
- f is β-smooth and convex if and only if
$$0 \preceq \nabla^2 f(x) \preceq \beta I$$
for all $x \in \mathbb{R}^n$

Convex Optimization Problems

Composite optimization form

- We will consider optimization problem on composite form
$$\underset{x}{\text{minimize}}\, f(Lx) + g(x)$$
where f and g are convex functions and L is a matrix
- Convex problem due to convexity preserving operations
- Can model constrained problems via indicator function
- This model format is suitable for many algorithms

Subdifferentials and Proximal Operators

Pontus Giselsson

Outline

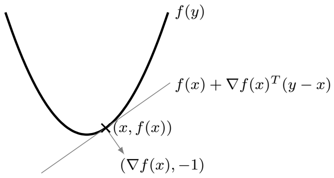
- **Subdifferential and subgradient – Definition and basic properties**
- Monotonicity
- Examples
- Strong monotonicity and cocoercivity
- Fermat's rule
- Subdifferential calculus
- Optimality conditions
- Proximal operators

Gradients of convex functions

- Recall: A *differentiable* function $f : \mathbb{R}^n \to \mathbb{R}$ is convex iff

$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

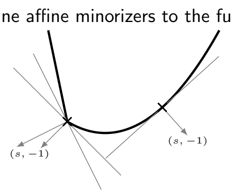
for all $x, y \in \mathbb{R}^n$



- Function f has for all $x \in \mathbb{R}^n$ an affine minorizer that:
  - has slope s defined by ∇f
  - coincides with function f at x
  - defines normal $(\nabla f(x), -1)$ to epigraph of f
- What if function is nondifferentiable?

Subdifferentials and subgradients

- Subgradients s define affine minorizers to the function that:

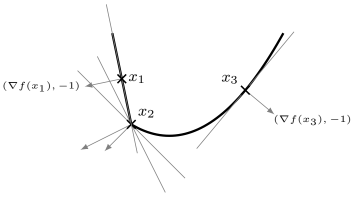


  - coincide with f at x
  - define normal vector $(s, -1)$ to epigraph of f
  - can be one of many affine minorizers at nondifferentiable points x
- Subdifferential of $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ at x is set of vectors s satisfying

$$f(y) \geq f(x) + s^T(y - x) \quad \text{for all } y \in \mathbb{R}^n, \qquad (1)$$

- Notation:
  - subdifferential: $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ (power-set notation $2^{\mathbb{R}^n}$)
  - subdifferential at x: $\partial f(x) = \{s : (1) \text{ holds}\}$
  - elements $s \in \partial f(x)$ are called *subgradients* of f at x

Relation to gradient



- If f differentiable at x and $\partial f(x) \neq \emptyset$ then $\partial f(x) = \{\nabla f(x)\}$:
- If f convex but not differentiable at $x \in \operatorname{int} \operatorname{dom} f$, then

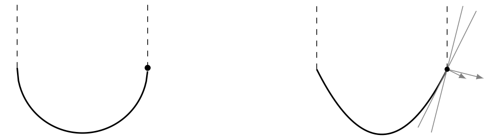
$$\partial f(x) = \operatorname{cl}(\operatorname{conv} S(x))$$

where $S(x)$ is set of all s such that $\nabla f(x_k) \to s$ when $x_k \to x$
- In general for convex f: $\partial f(x) = \operatorname{cl}(\operatorname{conv} S(x)) + N_{\operatorname{dom} f}(x)$

Subgradient existence – Convex setting

For *finite-valued convex* functions, a subgradient exists for every x

- In extended-valued setting, let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be convex:
  - (i) Subgradients exist for all x in relative interior of $\operatorname{dom} f$
  - (ii) Subgradients sometimes exist for x on relative boundary of $\operatorname{dom} f$
  - (iii) No subgradient exists for x outside $\operatorname{dom} f$
- Examples for second case, boundary points of $\operatorname{dom} f$:

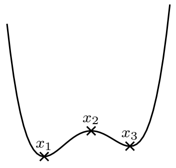


$-\sqrt{1 - x^2} + \iota_{[-1,1]}(x)$ $\qquad$ $x^2 + \iota_{[-2,2]}(x)$

- No subgradient (affine minorizer) exists for left function at $x = 1$

Subgradient existence – Nonconvex setting

- Function can be differentiable at x but $\partial f(x) = \emptyset$



  - x_1: $\partial f(x_1) = \{0\}$, $\nabla f(x_1) = 0$
  - x_2: $\partial f(x_2) = \emptyset$, $\nabla f(x_2) = 0$
  - x_3: $\partial f(x_3) = \emptyset$, $\nabla f(x_3) = 0$
- Gradient is a local concept, subdifferential is a global property

Outline

- Subdifferential and subgradient – Definition and basic properties
- **Monotonicity**
- Examples
- Strong monotonicity and cocoercivity
- Fermat's rule
- Subdifferential calculus
- Optimality conditions
- Proximal operators

Monotonicity of subdifferential

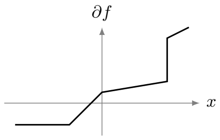
- Subdifferential operator is *monotone*:

$$(s_x - s_y)^T(x-y) \geq 0$$

for all $s_x \in \partial f(x)$ and $s_y \in \partial f(y)$
- Proof: Add two copies of subdifferential definition

$$f(y) \geq f(x) + s_x^T(y-x)$$

with x and y swapped
- $\partial f : \mathbb{R} \to 2^{\mathbb{R}}$: Minimum slope 0 and maximum slope ∞



Monotonicity beyond subdifferentials

- Let $A : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ be monotone, i.e.:

$$(u-v)^T(x-y) \geq 0$$

for all $u \in Ax$ and $v \in Ay$

- If $n = 1$, then $A = \partial f$ for some function $f : \mathbb{R} \to \mathbb{R} \cup \{\infty\}$
- If $n \geq 2$ there exist monotone A that are not subdifferentials

Maximal monotonicity

- Let the set $\mathrm{gph}\,\partial f := \{(x,u) : u \in \partial f(x)\}$ be the graph of ∂f
- ∂f is maximally monotone if no other function g exists with

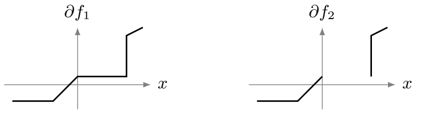
$$\mathrm{gph}\,\partial f \subset \mathrm{gph}\,\partial g,$$

with strict inclusion
- A result (due to Rockafellar):

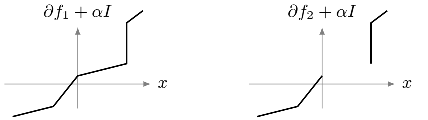
$$\boxed{f \text{ is closed convex if and only if } \partial f \text{ is maximally monotone}}$$

Minty's theorem

- Let $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ and $\alpha > 0$
- ∂f is maximally monotone if and only if $\mathrm{range}(\alpha I + \partial f) = \mathbb{R}^n$



maximally monotone　　not maximally monotone



full range　　not full range

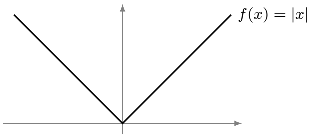
- Interpretation: No "holes" in $\mathrm{gph}\,\partial f$

Outline

- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- **Examples**
- Strong monotonicity and cocoercivity
- Fermat's rule
- Subdifferential calculus
- Optimality conditions
- Proximal operators

Example – Absolute value

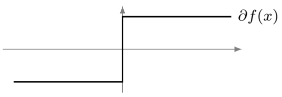
- The absolute value:



- Subdifferential
  - For $x > 0$, f differentiable and $\nabla f(x) = 1$, so $\partial f(x) = \{1\}$
  - For $x < 0$, f differentiable and $\nabla f(x) = -1$, so $\partial f(x) = \{-1\}$
  - For $x = 0$, f not differentiable, but since f convex:

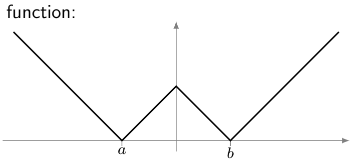
$$\partial f(0) = \mathrm{cl}(\mathrm{conv} S(0)) = \mathrm{cl}(\mathrm{conv}(\{-1,1\}) = [-1,1]$$

- The subdifferential operator:

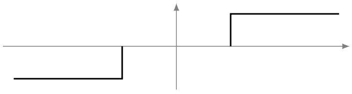


A nonconvex example

- Nonconvex function:



- Subdifferential
  - For $x > b$, f differentiable and $\nabla f(x) = 1$, so $\partial f(x) = \{1\}$
  - For $x < a$, f differentiable and $\nabla f(x) = -1$, so $\partial f(x) = \{-1\}$
  - For $x \in (a,b)$, no affine minorizer, $\partial f(x) = \emptyset$
  - For $x = a$, f not differentiable, $\partial f(x) = [-1,0]$
  - For $x = b$, f not differentiable, $\partial f(x) = [0,1]$
- The subdifferential operator:



Example – Separable functions

- Consider the separable function $f(x) = \sum_{i=1}^n f_i(x_i)$
- Subdifferential

$$\partial f(x) = \{s = (s_1,\ldots,s_n) : s_i \in \partial f_i(x_i)\}$$

- The subgradient $s \in \partial f(x)$ if and only if each $s_i \in \partial f_i(x_i)$
- Proof:
  - Assume all $s_i \in \partial f(x_i)$:

$$f(y) - f(x) = \sum_{i=1}^n f_i(y_i) - f_i(x_i) \geq \sum_{i=1}^n s_i(y_i - x_i) = s^T(y-x)$$

  - Assume $s_j \notin \partial f(x_j)$ and $x_i = y_i$ for all $i \neq j$:

$$f_j(y_j) - f_j(x_j) < s_j(y_j - x_j)$$

which gives

$$f(y) - f(x) = f_j(y_j) - f_j(x_j) < s_j(y_j - x_j) = s^T(y-x)$$

Example – 1-norm

- Consider the 1-norm $f(x) = \|x\|_1 = \sum_{i=1}^n |x_i|$
- It is a separable function of absolute values
- From previous examples, we conclude that the subdifferential is

$$\partial f(x) = \left\{ (s_1, \ldots, s_n) : \begin{cases} s_i = -1 & \text{if } x_i < 0 \\ s_i \in [-1, 1] & \text{if } x_i = 0 \\ s_i = 1 & \text{if } x_i > 0 \end{cases} \right\}$$

Example – 2-norm

- Consider the 2-norm $f(x) = \|x\|_2 = \sqrt{\|x\|_2^2}$
- The function is differentiable everywhere except for when $x = 0$
- Divide into two cases; $x = 0$ and $x \neq 0$
- Subdifferential for $x \neq 0$: $\partial f(x) = \{\nabla f(x)\}$:
  - Let $h(u) = \sqrt{u}$ and $g(x) = \|x\|_2^2$, then $f(x) = (h \circ g)(x)$
  - The gradient for all $x \neq 0$ by chain rule (since $h : \mathbb{R}_+ \to \mathbb{R}$):

$$\nabla f(x) = \nabla h(g(x)) \nabla g(x) = \tfrac{1}{2\sqrt{\|x\|_2^2}} 2x = \frac{x}{\|x\|_2}$$

Example cont'd – 2-norm

Subdifferential of $\|x\|_2$ at $x = 0$

(i) educated guess of subdifferential from $\partial f(0) = \mathrm{cl}(\mathrm{conv} S(0))$
  - recall $S(0)$ is set of all limit points of $(\nabla f(x_k))_{k \in \mathbb{N}}$ when $x_k \to 0$
  - let $x_k = t^k d$ with $t \in (0, 1)$ and $d \in \mathbb{R}^n \backslash 0$, then $\nabla f(x_k) = \frac{d}{\|d\|_2}$
  - since d arbitrary, $(\nabla f(x_k))$ can converge to any unit norm vector
  - so $S(0) = \{s : \|s\|_2 = 1\}$ and $\partial f(0) = \{s : \|s\|_2 \leq 1\}$?

(ii) verify using subgradient definition $f(y) \geq f(0) + s^T(y - 0) = s^T y$
  - Let $\|s\|_2 > 1$, then for, e.g., $y = 2s$

$$s^T y = 2\|s\|_2^2 > 2\|s\|_2 = f(y)$$

so such s are not subgradients
  - Let $\|s\|_2 \leq 1$, then for all y:

$$s^T y \leq \|s\|_2 \|y\|_2 \leq \|y\|_2 = f(y)$$

so such s are subgradients

Outline

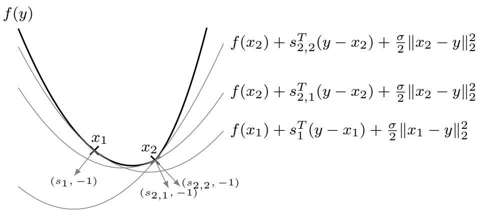
- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- Examples
- **Strong monotonicity and cocoercivity**
- Fermat's rule
- Subdifferential calculus
- Optimality conditions
- Proximal operators

Strong convexity revisited

- Recall that f is σ-strongly convex if $f - \frac{\sigma}{2}\|\cdot\|_2^2$ is convex
- If f is σ-strongly convex then

$$f(y) \geq f(x) + s^T(y - x) + \tfrac{\sigma}{2}\|x - y\|_2^2$$

holds for all $x \in \mathrm{dom}\partial f$, $s \in \partial f(x)$, and $y \in \mathbb{R}^n$
- The function has convex quadratic minorizers instead of affine



- Multiple lower bounds at x_2 with subgradients $s_{2,1}$ and $s_{2,2}$

Strong monotonicity

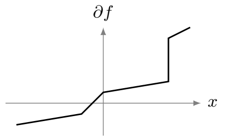
- If f σ-strongly convex function, then ∂f is *σ-strongly monotone*:

$$(s_x - s_y)^T(x - y) \geq \sigma\|x - y\|_2^2$$

for all $s_x \in \partial f(x)$ and $s_y \in \partial f(y)$
- Proof: Add two copies of strong convexity inequality

$$f(y) \geq f(x) + s_x^T(y - x) + \tfrac{\sigma}{2}\|x - y\|_2^2$$

with x and y swapped
- ∂f is σ-strongly monotone if and only if $\partial f - \sigma I$ is monotone
- $\partial f : \mathbb{R} \to 2^{\mathbb{R}}$: Minimum slope σ and maximum slope ∞



Strongly convex functions – An equivalence

The following are equivalent for $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$

(i) f is closed and σ-strongly convex
(ii) ∂f is maximally monotone and σ-strongly monotone

Proof:

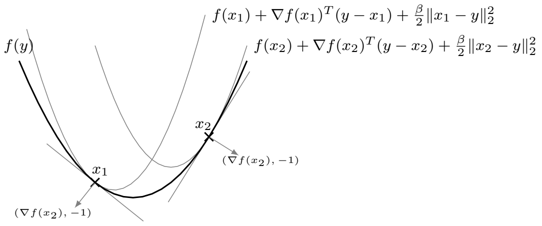
(i)⇒(ii): we know this from before
(ii)⇒(i): (ii) $\Rightarrow \partial f - \sigma I = \partial(f - \frac{\sigma}{2}\|\cdot\|_2^2)$ maximally monotone
$\Rightarrow f - \frac{\sigma}{2}\|\cdot\|_2^2$ closed convex
$\Rightarrow f$ closed and σ-strongly convex

Smooth convex functions

- A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and β-smooth if

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|x - y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

hold for all $x, y \in \mathbb{R}^n$
- f has convex quadratic majorizers and affine minorizers



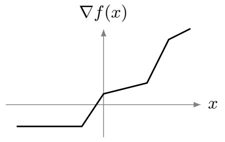
- Quadratic upper bound is called *descent lemma*

Cocoercivity of gradient

- Gradient of smooth convex function is monotone and Lipschitz

$$(\nabla f(x) - \nabla f(y))^T(x-y) \geq 0$$
$$\|\nabla f(y) - \nabla f(x)\|_2 \leq \beta\|x-y\|_2$$

- $\nabla f : \mathbb{R} \to \mathbb{R}$: Minimum slope 0 and maximum slope β



- Actually satisfies the stronger $\frac{1}{\beta}$-*cocoercivity* property:

$$(\nabla f(x) - \nabla f(y))^T(x-y) \geq \tfrac{1}{\beta}\|\nabla f(y) - \nabla f(x)\|_2^2$$

due to the *Baillon-Haddad theorem*

Smooth convex functions – An equivalence

Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable. The following are equivalent:

(i) ∇f is $\frac{1}{\beta}$-cocoercive
(ii) ∇f is maximally monotone and β-Lipschitz continuous
(iii) f is closed convex and satisfies descent lemma (is β-smooth)

Will later connect smooth convexity and strong convexity via conjugates

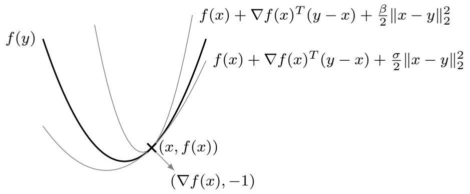
Smooth strongly convex functions

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable
- f is β-smooth and σ-strongly convex with $0 < \sigma \leq \beta$ if

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \tfrac{\sigma}{2}\|x-y\|_2^2$$

hold for all $x, y \in \mathbb{R}^n$

- f has quadratic minorizers and quadratic majorizers



- We say that the ratio $\frac{\beta}{\sigma}$ is the *condition number* for the function

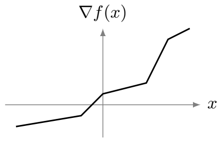
Gradient of smooth strongly convex function

- Gradient of β-smooth σ-strongly convex function f satisfies

$$\|\nabla f(y) - \nabla f(x)\|_2 \leq \beta\|x-y\|_2$$
$$(\nabla f(x) - \nabla f(y))^T(x-y) \geq \sigma\|x-y\|_2^2$$

so is β-Lipschitz continuous and σ-strongly monotone

- $\nabla f : \mathbb{R} \to \mathbb{R}$: Minimum slope σ and maximum slope β



- Actually satisfies this stronger property:

$$(\nabla f(x) - \nabla f(y))^T(x-y) \geq \tfrac{1}{\beta+\sigma}\|\nabla f(y) - \nabla f(x)\|_2^2 + \tfrac{\sigma\beta}{\beta+\sigma}\|x-y\|_2^2$$

for all $x, y \in \mathbb{R}^n$

Proof of stronger property

- f is σ-strongly convex if and only if $g := f - \frac{\sigma}{2}\|\cdot\|_2^2$ is convex
- Since f is β-smooth g is $(\beta-\sigma)$-smooth
- Since g convex and $(\beta-\sigma)$-smooth, ∇g is $\frac{1}{\beta-\sigma}$-cocoercive:

$$(\nabla g(x) - \nabla g(y))^T(x-y) \geq \tfrac{1}{\beta-\sigma}\|\nabla g(x) - \nabla g(y)\|_2^2$$

which by using $\nabla g = \nabla f - \sigma I$ gives

$$(\nabla f(x) - \nabla f(y))^T(x-y) - \sigma\|x-y\|_2^2 \geq \tfrac{1}{\beta-\sigma}\|\nabla f(x) - \nabla f(y) - \sigma(x-y)\|_2^2$$

which by expanding the square and rearranging is equivalent to

$$(\nabla f(x) - \nabla f(y))^T(x-y) \geq \tfrac{1}{\beta+\sigma}\|\nabla f(x) - \nabla f(y)\|_2^2 + \tfrac{\sigma\beta}{\beta+\sigma}\|x-y\|_2^2$$

Outline

- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- Examples
- Strong monotonicity and cocoercivity
- **Fermat's rule**
- Subdifferential calculus
- Optimality conditions
- Proximal operators

Fermat's rule

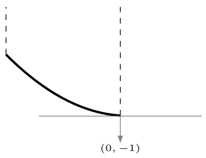
Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, then x minimizes f if and only if
$0 \in \partial f(x)$

- Proof: x minimizes f if and only if

$$f(y) \geq f(x) = f(x) + 0^T(y-x) \quad \text{for all } y \in \mathbb{R}^n$$

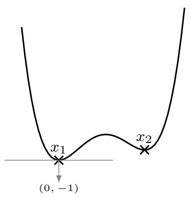
which by definition of subdifferential is equivalent to $0 \in \partial f(x)$

- Example: several subgradients at solution, including 0



Fermat's rule – Nonconvex example

- Fermat's rule holds also for nonconvex functions
- Example:



  - $\partial f(x_1) = 0$ and $\nabla f(x_1) = 0$ (global minimum)
  - $\partial f(x_2) = \emptyset$ and $\nabla f(x_2) = 0$ (local minimum)
- For nonconvex f, we can typically only hope to find local minima

Outline

- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- Examples
- Strong monotonicity and cocoercivity
- Fermat's rule
- **Subdifferential calculus**
- Optimality conditions
- Proximal operators

Subdifferential calculus rules

- Subdifferential of sum $\partial(f_1 + f_2)$
- Subdifferential of composition with matrix $\partial(g \circ L)$

Subdifferential of sum

If f_1, f_2 closed convex and $\operatorname{relint} \operatorname{dom} f_1 \cap \operatorname{relint} \operatorname{dom} f_2 \neq \emptyset$:
$\partial(f_1 + f_2) = \partial f_1 + \partial f_2$

- One direction always holds: if $x \in \operatorname{dom}\partial f_1 \cap \operatorname{dom}\partial f_2$:

$$\partial(f_1 + f_2)(x) \supseteq \partial f_1(x) + \partial f_2(x)$$

  Proof: let $s_i \in \partial f_i(x)$, add subdifferential definitions:

$$f_1(y) + f_2(y) \geq f_1(x) + f_2(x) + (s_1 + s_2)^T(y - x)$$

  i.e. $s_1 + s_2 \in \partial(f_1 + f_2)(x)$
- If f_1 and f_2 differentiable, we have (without convexity of f)

$$\nabla(f_1 + f_2) = \nabla f_1 + \nabla f_2$$

Subdifferential of composition

If f closed convex and $\operatorname{relint} \operatorname{dom}(f \circ L) \neq \emptyset$:
$\partial(f \circ L)(x) = L^T \partial f(Lx)$

- One direction always holds: If $Lx \in \operatorname{dom} f$, then

$$\partial(f \circ L)(x) \supseteq L^T \partial f(Lx)$$

  Proof: let $s \in \partial f(Lx)$, then by definition of subgradient of f:

$$(f \circ L)(y) \geq (f \circ L)(x) + s^T(Ly - Lx) = (f \circ L)(x) + (L^T s)^T(y - x)$$

  i.e., $L^T s \in \partial(f \circ L)(x)$
- If f differentiable, we have chain rule (without convexity of f)

$$\nabla(f \circ L)(x) = L^T \nabla f(Lx)$$

Outline

- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- Examples
- Strong monotonicity and cocoercivity
- Fermat's rule
- Subdifferential calculus
- **Optimality conditions**
- Proximal operators

Composite optimization problems

- We consider optimization problems on *composite form*

$$\underset{x}{\operatorname{minimize}}\, f(Lx) + g(x)$$

  where $f : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$, $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, and $L \in \mathbb{R}^{m \times n}$
- Can model constrained problems via indicator function
- This model format is suitable for many algorithms

A sufficient optimality condition

Let $f : \mathbb{R}^m \to \overline{\mathbb{R}}$, $g : \mathbb{R}^n \to \overline{\mathbb{R}}$, and $L \in \mathbb{R}^{m \times n}$ then:

$$\operatorname{minimize} f(Lx) + g(x) \tag{1}$$

is solved by every $x \in \mathbb{R}^n$ that satisfies

$$0 \in L^T \partial f(Lx) + \partial g(x) \tag{2}$$

- Subdifferential calculus inclusions say:

$$0 \in L^T \partial f(Lx) + \partial g(x) \subseteq \partial((f \circ L)(x) + g(x))$$

  which by Fermat's rule is equivalent to x solution to (1)
- Note: (1) can have solution but no x exists that satisfies (2)

A necessary and sufficient optimality condition

Let $f : \mathbb{R}^m \to \overline{\mathbb{R}}$, $g : \mathbb{R}^n \to \overline{\mathbb{R}}$, $L \in \mathbb{R}^{m \times n}$ with f, g closed convex and assume $\operatorname{relint} \operatorname{dom}(f \circ L) \cap \operatorname{relint} \operatorname{dom} g \neq \emptyset$ then:

$$\operatorname{minimize} f(Lx) + g(x) \tag{1}$$

is solved by $x \in \mathbb{R}^n$ if and only if x satisfies

$$0 \in L^T \partial f(Lx) + \partial g(x) \tag{2}$$

- Subdifferential calculus equality rules say:

$$0 \in L^T \partial f(Lx) + \partial g(x) = \partial((f \circ L)(x) + g(x))$$

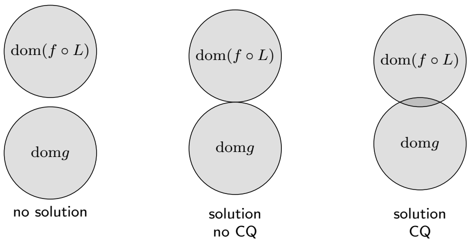
  which by Fermat's rule is equivalent to x solution to (1)
- Algorithms search for x that satisfy $0 \in L^T \partial f(Lx) + \partial g(x)$

A comment on constraint qualification

- The condition

$$\operatorname{relint}\operatorname{dom}(f\circ L)\cap\operatorname{relint}\operatorname{dom}g\neq\emptyset$$

  is called *constraint qualification* and referred to as CQ
- It is a mild condition that rarely is not satisfied



Evaluating subgradients of convex functions

- Obviously need to evaluate subdifferentials to solve

$$0\in L^T\partial f(Lx)+\partial g(x)$$

- Explicit evaluation:
  - If function is differentiable: ∇f (unique)
  - If function is nondifferentiable: compute element in ∂f
- Implicit evaluation:
  - Proximal operator (specific element of subdifferential)

Outline

- Subdifferential and subgradient – Definition and basic properties
- Monotonicity
- Examples
- Strong monotonicity and cocoercivity
- Fermat's rule
- Subdifferential calculus
- Optimality conditions
- **Proximal operators**

Proximal operators

Proximal operator – Definition

- Proximal operator of g defined as:

$$\mathrm{prox}_{\gamma g}(z)=\underset{x}{\mathrm{argmin}}(g(x)+\tfrac{1}{2\gamma}\|x-z\|_2^2)$$

  where $\gamma>0$ is a parameter
- Evaluating *prox* requires solving optimization problem
- For convex g, prox is well-defined and single-valued
  - Why? Objective is strongly convex $\Rightarrow$ argmin exists and is unique

Prox is generalization of projection

- Recall the indicator function of a set C

$$\iota_C(x):=\begin{cases}0 & \text{if } x\in C\\ \infty & \text{otherwise}\end{cases}$$

- Then

$$\begin{aligned}\mathrm{prox}_{\iota_C}(z)&=\underset{x}{\mathrm{argmin}}(\tfrac{1}{2}\|x-z\|_2^2+\iota_C(x))\\&=\underset{x}{\mathrm{argmin}}(\tfrac{1}{2}\|x-z\|_2^2:x\in C)\\&=\underset{x}{\mathrm{argmin}}(\|x-z\|_2:x\in C)\\&=\Pi_C(z)\end{aligned}$$

- Projection onto C equals prox of indicator function of C

Prox computes a subgradient

- Fermat's rule on prox definition: $x=\mathrm{prox}_{\gamma g}(z)$ if and only if

$$0\in\partial g(x)+\gamma^{-1}(x-z)\quad\Leftrightarrow\quad\gamma^{-1}(z-x)\in\partial g(x)$$

  Hence, $\gamma^{-1}(z-x)$ is element in $\partial g(x)$
- A subgradient $\partial g(x)$ where $x=\mathrm{prox}_{\gamma g}(z)$ is computed

Prox is 1-cocoercive

- For convex g, the proximal operator is 1-cocoercive:

$$(x-y)^T(\mathrm{prox}_{\gamma g}(x)-\mathrm{prox}_{\gamma f}(y))\geq\|\mathrm{prox}_{\gamma g}(x)-\mathrm{prox}_{\gamma f}(y)\|_2^2$$

- Proof
  - Combine monotonicity of ∂g, that for all $z_u\in\partial g(u)$, $z_v\in\partial g(v)$:

$$(z_u-z_v)^T(u-v)\geq 0$$

  - with Fermat's rule on prox that evalutes subgradients of g:

$$\begin{aligned}u&=\mathrm{prox}_{\gamma g}(x)\quad\text{if and only if}\quad\gamma^{-1}(x-u)\in\partial g(u)\\v&=\mathrm{prox}_{\gamma g}(y)\quad\text{if and only if}\quad\gamma^{-1}(y-v)\in\partial g(v)\end{aligned}$$

  - which gives, by letting $z_u=\gamma^{-1}(x-u)$ and $z_v=\gamma^{-1}(y-v)$:

$$\begin{aligned}&\gamma^{-1}((x-u)-(y-v))^T(u-v)\geq 0\\\Leftrightarrow\quad&(x-\mathrm{prox}_{\gamma g}(x)-(y-\mathrm{prox}_{\gamma g}(y)))^T(\mathrm{prox}_{\gamma g}(x)-\mathrm{prox}_{\gamma g}(y))\geq 0\\\Leftrightarrow\quad&(x-y)^T(\mathrm{prox}_{\gamma g}(x)-\mathrm{prox}_{\gamma g}(y))\geq\|\mathrm{prox}_{\gamma g}(x)-\mathrm{prox}_{\gamma g}(y)\|_2^2\end{aligned}$$

Prox is (firmly) nonexpansive

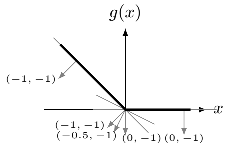
- We know 1-cocoercivity implies nonexpansiveness (1-Lipschitz)
$$\|\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma g}(y)\|_2 \leq \|x-y\|_2$$
which was shown using Cauchy-Schwarz inequality
- Actually the stronger *firm* nonexpansive inequality holds
$$\|\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma g}(y)\|_2^2 \leq \|x-y\|_2^2 - \|x - \mathrm{prox}_{\gamma g}(x) - (y - \mathrm{prox}_{\gamma g}(y))\|_2^2$$
which implies nonexpansiveness
- Proof:
  - take 1-cocoercivity and multiply both sides by 2:
  $$2(x-y)^T(\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma f}(y)) \geq 2\|\mathrm{prox}_{\gamma g}(x) - \mathrm{prox}_{\gamma f}(y)\|_2^2$$
  - use the following equality with $u = \mathrm{prox}_{\gamma g}(x)$ and $v = \mathrm{prox}_{\gamma g}(y)$:
  $$(x-y)^T(u-v) = \tfrac{1}{2}\left(\|x-y\|_2^2 + \|u-v\|_2^2 - \|x-y-(u-v)\|_2^2\right)$$

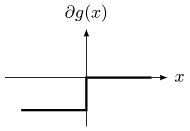
Proximal operator – Separable functions

- Let $x = (x_1, \ldots, x_n)$ and $g(x) = \sum_{i=1}^n g_i(x_i)$ be separable, then
$$\mathrm{prox}_{\gamma g}(z) = (\mathrm{prox}_{\gamma g_1}(z_1), \ldots, \mathrm{prox}_{\gamma g_n}(z_n))$$
decomposes into n individual proxes
- Why? Since also $\|\cdot\|_2^2$ is separable:
$$\begin{aligned}\mathrm{prox}_{\gamma g}(z) &= \underset{x}{\mathrm{argmin}}(g(x) + \tfrac{1}{2\gamma}\|x-z\|_2^2)\\ &= \underset{x}{\mathrm{argmin}}\left(\sum_{i=1}^n (g_i(x_i) + \tfrac{1}{2\gamma}(x_i - z_i)^2)\right)\end{aligned}$$
which gives n independent optimization problems
$$\underset{x_i}{\mathrm{argmin}}(g_i(x_i) + \tfrac{1}{2\gamma}(x_i - z_i)^2) = \mathrm{prox}_{\gamma g_i}(z_i)$$

Proximal operator – Example 1

- Consider the function g with subdifferential ∂g:
$$g(x) = \begin{cases} -x & \text{if } x \leq 0 \\ 0 & \text{if } x \geq 0 \end{cases} \qquad \partial g(x) = \begin{cases} -1 & \text{if } x < 0 \\ [-1,0] & \text{if } x = 0 \\ 0 & \text{if } x > 0 \end{cases}$$
- Graphical representations





- Fermat's rule for $x = \mathrm{prox}_{\gamma g}(z)$:
$$0 \in \partial g(x) + \gamma^{-1}(x-z)$$

Proximal operator – Example 1 cont'd

- Let $x < 0$, then Fermat's rule reads
$$0 = -1 + \gamma^{-1}(x-z) \quad \Leftrightarrow \quad x = z + \gamma$$
which is valid ($x < 0$) if $z < -\gamma$
- Let $x = 0$, then Fermat's rule reads
$$0 \in [-1,0] + \gamma^{-1}(0-z)$$
which is valid ($x = 0$) if $z \in [-\gamma, 0]$
- Let $x > 0$, then Fermat's rule reads
$$0 = 0 + \gamma^{-1}(x-z) \quad \Leftrightarrow \quad x = z$$
which is valid ($x > 0$) if $z > 0$
- The prox satisfies
$$\mathrm{prox}_{\gamma g}(z) = \begin{cases} z + \gamma & \text{if } z < -\gamma \\ 0 & \text{if } z \in [-\gamma, 0] \\ z & \text{if } z > 0 \end{cases}$$

Proximal operator – Example 2

Let $g(x) = \frac{1}{2}x^TPx + q^Tx$ with P positive semidefinite

- Gradient satisfies $\nabla g(x) = Px + q$
- Fermat's rule for $x = \mathrm{prox}_{\gamma g}(z)$:
$$\begin{aligned}0 = \nabla g(x) + \gamma^{-1}(x-z) \quad &\Leftrightarrow \quad 0 = Px + q + \gamma^{-1}(x-z)\\ &\Leftrightarrow \quad (I + \gamma P)x = z - \gamma q\\ &\Leftrightarrow \quad x = (I + \gamma P)^{-1}(z - \gamma q)\end{aligned}$$
- So $\mathrm{prox}_{\gamma g}(z) = (I + \gamma P)^{-1}(z - \gamma q)$

Computational cost

- Evaluating prox requires solving optimization problem
$$\mathrm{prox}_{\gamma g}(z) = \underset{x}{\mathrm{argmin}}(g(x) + \tfrac{1}{2\gamma}\|x-z\|_2^2)$$
- Prox often more expensive to evaluate than gradient
  - Example: Quadratic $g(x) = \frac{1}{2}x^TPx + q^Tx$:
  $$\mathrm{prox}_{\gamma g}(z) = (I + \gamma P)^{-1}(z - \gamma q), \qquad \nabla g(z) = Pz + q$$
- But typically cheap to evaluate for separable functions
- Prox often used for nondifferentiable and separable functions

Conjugate Functions, Optimality Conditions, and Duality

Pontus Giselsson

Outline

- **Conjugate function – Definition and basic properties**
- Examples
- Biconjugate
- Fenchel-Young's inequality
- Duality correspondence
- Moreau decomposition
- Duality and optimality conditions
- Weak and strong duality

Conjugate Functions

Conjugate function – Definition

- The conjugate function of $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined as

$$f^*(s) := \sup_x \left(s^T x - f(x)\right)$$

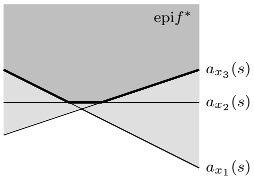
- Implicit definition via optimization problem

Conjugate function properties

- Let $a_x(s) := s^T x - f(x)$ be affine function parameterized by x:

$$f^*(s) = \sup_x a_x(s)$$

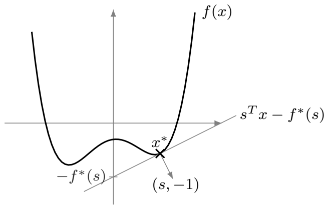
  is supremum of family of affine functions
- Epigraph of f^* is intersection of epigraphs of (below three) a_x



- f^* convex: epigraph intersection of convex halfspaces $\mathrm{epi}\, a_x$
- f^* closed: epigraph intersection of closed halfspaces $\mathrm{epi}\, a_x$

Conjugate interpretation

- Conjugate $f^*(s)$ defines affine minorizer to f with slope s:



  where $-f^*(s)$ decides constant offset to get support
- Why?

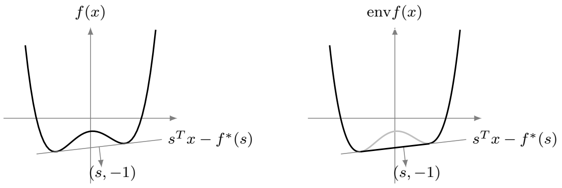
$$f^*(s) = \sup_x \left(s^T x - f(x)\right) \quad \Leftrightarrow \quad f^*(s) \geq s^T x - f(x) \text{ for all } x$$

$$\Leftrightarrow \quad f(x) \geq s^T x - f^*(s) \text{ for all } x$$

- Maximizing argument x^* gives support: $f(x^*) = s^T x^* - f^*(s)$
- We have $f(x^*) = s^T x^* - f^*(s)$ if and only if $s \in \partial f(x^*)$

Consequence

- Conjugate of f and $\mathrm{env} f$ are the same, i.e., $f^* = (\mathrm{env} f)^*$



- Functions have same supporting affine functions
- Epigraphs have same supporting hyperplanes

Outline

- Conjugate function – Definition and basic properties
- **Examples**
- Biconjugate
- Fenchel-Young's inequality
- Duality correspondence
- Moreau decomposition
- Duality and optimality conditions
- Weak and strong duality

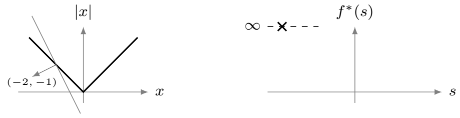
Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s = -2$ $f^*(s)$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis



Slope, $s = -2$ $f^*(s) \to \infty$

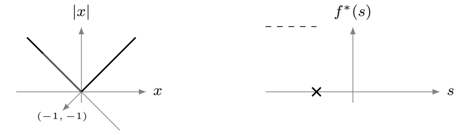
Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s = -1$ $f^*(s)$

Example – Absolute value

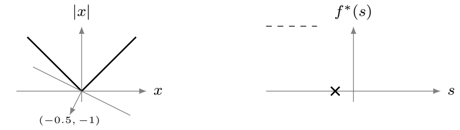
- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis



Slope, $s = -1$ $f^*(s) = 0$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis



Slope, $s = -0.5$ $f^*(s)$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s = -0.5$ $f^*(s) = 0$

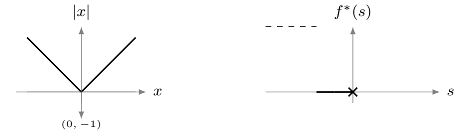
Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s = 0$ $f^*(s)$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis



Slope, $s = 0$ $f^*(s) = 0$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$0.5 $f^*(s)$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$0.5 $f^*(s) = 0$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$1 $f^*(s)$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$1 $f^*(s) = 0$

Example – Absolute value

- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$2 $f^*(s)$

Example – Absolute value

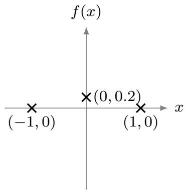
- Compute conjugate of $f(x) = |x|$
- For given slope s: $-f^*(s)$ is point that crosses $|x|$-axis

Slope, $s =$2 $f^*(s) \to \infty$

- Conjugate is $f^*(s) = \iota_{[-1,1]}(s)$

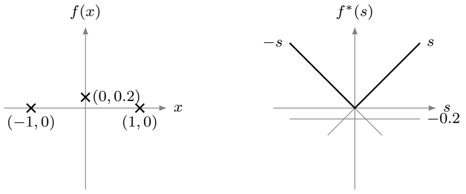
A nonconvex example

- Draw conjugate of f ($f(x) = \infty$ outside points)



A nonconvex example

- Draw conjugate of f ($f(x) = \infty$ outside points)



- Draw all affine $a_x(s)$ and select for each s the max to get $f^*(s)$

$$f^*(s) = \sup_x (sx - f(x)) = \max(-s - 0, 0s - 0.2, s - 0)$$

$$= \max(-s, -0.2, s) = |s|$$

Example – Quadratic functions

Let $g(x) = \frac{1}{2}x^TQx + p^Tx$ with Q positive definite (invertible)

- Gradient satisfies $\nabla g(x) = Qx + p$
- Fermat's rule for $g^*(s) = \sup_x(s^Tx - \frac{1}{2}x^TQx - p^Tx)$:

$$0 = s - Qx - p \quad \Leftrightarrow \quad x = Q^{-1}(s-p)$$

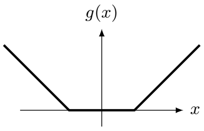
- So

$$\begin{aligned} g^*(s) &= s^TQ^{-1}(s-p) - \tfrac{1}{2}(s-p)^TQ^{-1}QQ^{-1}(s-p) + p^TQ^{-1}(s-p) \\ &= \tfrac{1}{2}(s-p)^TQ^{-1}(s-p) \end{aligned}$$

Example – A piece-wise linear function

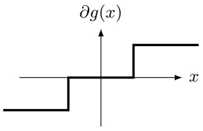
- Consider

$$g(x) = \begin{cases} -x-1 & \text{if } x \leq -1 \\ 0 & \text{if } x \in [-1,1] \\ x-1 & \text{if } x \geq 1 \end{cases}$$



- Subdifferential satisfies

$$\partial g(x) = \begin{cases} -1 & \text{if } x < -1 \\ [-1,0] & \text{if } x = -1 \\ 0 & \text{if } x \in (-1,1) \\ [0,1] & \text{if } x = 1 \\ 1 & \text{if } x > 1 \end{cases}$$



Example cont'd

- We use $g^*(s) = sx - g(x)$ if $s \in \partial g(x)$:
  - $x < -1$: $s = -1$, hence $g^*(-1) = -1x - (-x-1) = 1$
  - $x = -1$: $s \in [-1,0]$ hence $g^*(s) = -s - 0 = -s$
  - $x \in (-1,1)$: $s = 0$ hence $g^*(0) = 0x - 0 = 0$
  - $x = 1$: $s \in [0,1]$ hence $g^*(s) = s - 0 = s$
  - $x > 1$: $s = 1$ hence $g^*(1) = x - (x-1) = 1$
- That is

$$g^*(s) = \begin{cases} -s & \text{if } s \in [-1,0] \\ s & \text{if } s \in [0,1] \end{cases}$$

- For $s < -1$ and $s > 1$, $g^*(s) = \infty$:
  - $s < -1$: let $x = t \to -\infty$ and $g^*(s) \geq ((s+1)t+1) \to \infty$
  - $s > 1$: let $x = t \to \infty$ and $g^*(s) \geq ((s-1)t+1) \to \infty$

Example – Separable functions

- Let $f(x) = \sum_{i=1}^n f_i(x_i)$ be a separable function, then

$$f^*(s) = \sum_{i=1}^n f_i^*(s_i)$$

  is also separable
- Proof:

$$\begin{aligned} f^*(s) &= \sup_x(s^Tx - \sum_{i=1}^n f_i(x_i)) \\ &= \sup_x(\sum_{i=1}^n(s_ix_i - f_i(x_i))) \\ &= \sum_{i=1}^n \sup_{x_i}(s_ix_i - f_i(x_i)) \\ &= \sum_{i=1}^n f_i^*(s_i) \end{aligned}$$

Example – 1-norm

- Let $f(x) = \|x\|_1 = \sum_{i=1}^n |x_i|$ be the 1-norm
- It is a separable sum of absolute values
- Use separable sum formula and that $|\cdot|^* = \iota_{[-1,1]}$:

$$f^*(s) = \sum_{i=1}^n f_i^*(s_i) = \sum_{i=1}^n \iota_{[-1,1]}(s_i) = \begin{cases} 0 & \text{if } \max_i(|s_i|) \leq 1 \\ \infty & \text{else} \end{cases}$$

- We have $\max_i(|s_i|) = \|s\|_\infty$, let

$$B_\infty(r) = \{s : \|s\|_\infty \leq r\}$$

  be the infinity norm ball of radius r, then

$$f^*(s) = \iota_{B_\infty(1)}(s)$$

  is the indicator function for the unit infinity norm ball

Outline

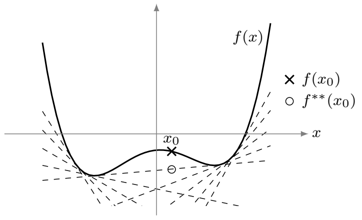
- Conjugate function – Definition and basic properties
- Examples
- **Biconjugate**
- Fenchel-Young's inequality
- Duality correspondence
- Moreau decomposition
- Duality and optimality conditions
- Weak and strong duality

Biconjugate

- Biconjuate $f^{**} := (f^*)^*$ is conjugate of conjugate

$$f^{**}(x) = \sup_s(x^Ts - f^*(s))$$

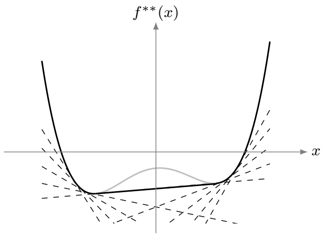
- For every x, it is largest value of all affine minorizers



- Why?:
  - $x^Ts - f^*(s)$: supporting affine minorizer to f with slope s
  - $f^{**}(x)$ picks largest over all these affine minorizers evaluated at x

Biconjugate and convex envelope

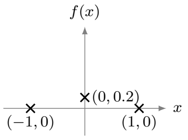
- Biconjugate is closed convex envelope of f



- $f^{**} \leq f$ and $f^{**} = f$ if and only if f (closed and) convex

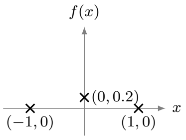
Biconjugate – Example

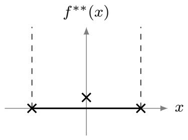
- Draw the biconjugate of f ($f(x) = \infty$ outside points)



Biconjugate – Example

- Draw the biconjugate of f ($f(x) = \infty$ outside points)





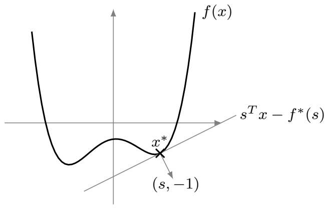
- Biconjugate is convex envelope of f
- We found before $f^*(s) = |s|$, and now $(f^*)^*(x) = \iota_{[-1,1]}(x)$
- Therefore also $\iota^*_{[-1,1]}(s) = |s|$
  (since $f^* = (\mathrm{env} f)^* = (f^{**})^* =: f^{***}$)

Outline

- Conjugate function – Definition and basic properties
- Examples
- Biconjugate
- **Fenchel-Young's inequality**
- Duality correspondence
- Moreau decomposition
- Duality and optimality conditions
- Weak and strong duality

Fenchel-Young's inequality

- Going back to conjugate interpretation:

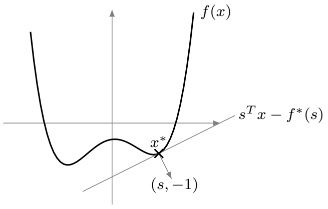


- Fenchel-Youngs's inequality: $f(x) \geq s^Tx - f^*(s)$ for all x, s
- Follows immediately from definition: $f^*(s) = \sup_x(s^Tx - f(x))$

Fenchel-Young's equality

- When is do we have equality in Fenchel-Young?

$$f(x) = s^Tx - f^*(s)$$



- Fenchel-Young's equality and equivalence:

$$\boxed{f(x^*) = s^Tx^* - f^*(s) \text{ holds if and only if } s \in \partial f(x^*)}$$

Proof – Fenchel-Young's equality

$$\boxed{f(x) = s^Tx - f^*(s) \text{ holds if and only if } s \in \partial f(x)}$$

- $s \in \partial f(x)$ if and only if (by defintion of subgradient)

$$\begin{aligned} & f(y) \geq f(x) + s^T(y - x) \text{ for all } y \\ \Leftrightarrow \quad & s^Tx - f(x) \geq s^Ty - f(y) \text{ for all } y \\ \Leftrightarrow \quad & s^Tx - f(x) \geq \sup_y \left(s^Ty - f(y)\right) \\ \Leftrightarrow \quad & s^Tx - f(x) \geq f^*(s) \end{aligned}$$

  which is Fenchel-Young's inequality with inequality reversed
- Fenchel-Young's inequality always holds:

$$f^*(s) \geq s^Tx - f(x)$$

  so we have equality if and only if $s \in \partial f(x)$

A subdifferential formula for convex f

$$\boxed{\text{Assume } f \text{ closed convex, then } \partial f(x) = \mathrm{Argmax}_s(s^Tx - f^*(s))}$$

- Since $f^{**} = f$, we have $f(x) = \sup_s(x^Ts - f^*(s))$ and

$$\begin{aligned} s^* \in \underset{s}{\mathrm{Argmax}}(x^Ts - f^*(s)) \quad & \Longleftrightarrow \quad f(x) = x^Ts^* - f^*(s^*) \\ & \Longleftrightarrow \quad s^* \in \partial f(x) \end{aligned}$$

- The last equivalence is from previous slide

Subdifferential formulas for f^*

- For general f, we have that

$$\partial f^*(s) = \underset{x}{\mathrm{Argmax}}(s^Tx - f^{**}(x))$$

  by previous formula and since f^* closed and convex
- For closed convex f, we have, since $f = f^{**}$, that

$$\partial f^*(s) = \underset{x}{\mathrm{Argmax}}(s^Tx - f(x))$$

Relation between ∂f and ∂f^* – General case

$\boxed{s \in \partial f(x) \text{ implies that } x \in \partial f^*(s)}$

- Since $f^{**} \leq f$ and $s \in \partial f(x)$, Fenchel-Young's equality gives:

$$0 = f^*(s) + f(x) - s^T x \geq f^*(s) + f^{**}(x) - s^T x \geq 0$$

  where last step is Fenchel-Young's inequality
- Hence $f^*(s) + f^{**}(x) - s^T x = 0$ and FY $\Rightarrow x \in \partial f^*(s)$

Inverse relation between ∂f and ∂f^* – Convex case

$\boxed{\text{Suppose } f \text{ closed convex, then } s \in \partial f(x) \Longleftrightarrow x \in \partial f^*(s)}$

- Using implication on previous slide twice and $f^{**} = f$:

$$s \in \partial f(x) \Rightarrow x \in \partial f^*(s) \Rightarrow s \in \partial f^{**}(x) \Rightarrow s \in \partial f(x)$$

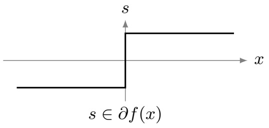
- Another way to write the result is that for closed convex f:

$$\partial f^* = (\partial f)^{-1}$$

  (Definition of inverse of set-valued A: $x \in A^{-1}u \Longleftrightarrow u \in Ax$)

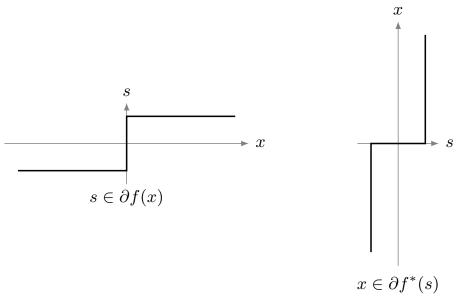
Example 1 – Relation between ∂f and ∂f^*

- What is ∂f^* for below ∂f?



Example 1 – Relation between ∂f and ∂f^*

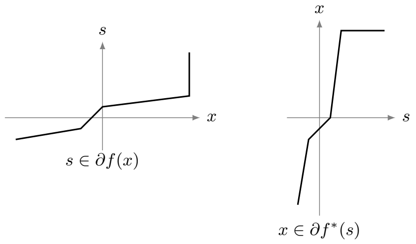
- What is ∂f^* for below ∂f?



- Since $\partial f^* = (\partial f)^{-1}$, we flip the figure

---

Example 2 – Relation between ∂f and ∂f^*



- region with slope σ in $\partial f(x) \Leftrightarrow$ region with slope $\frac{1}{\sigma}$ in $\partial f^*(s)$
- Implication: ∂f σ-strong monotone $\Leftrightarrow$ $\partial f^*(s)$ σ-cocoercive?
  (Recall: σ-cocoercivity $\Leftrightarrow$ $\frac{1}{\sigma}$-Lipschitz and monotone)

Outline

- Conjugate function – Definition and basic properties
- Examples
- Biconjugate
- Fenchel-Young's inequality
- **Duality correspondence**
- Moreau decomposition
- Duality and optimality conditions
- Weak and strong duality

Cocoercivity and strong monotonicity

$$\boxed{\begin{array}{c}\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n} \text{ maximal monotone and } \sigma\text{-strongly monotone} \\ \Longleftrightarrow \\ \partial f^* = \nabla f^* : \mathbb{R}^n \to \mathbb{R}^n \text{ single-valued and } \sigma\text{-cocoercive}\end{array}}$$

- σ-strong monotonicity: for all $u \in \partial f(x)$ and $v \in \partial f(y)$

$$(u - v)^T (x - y) \geq \sigma \|x - y\|_2^2 \qquad (1)$$

  or equivalently for all $x \in \partial f^*(u)$ and $y \in \partial f^*(v)$
- ∂f^* is single-valued:
  - Assume $x \in \partial f^*(u)$ and $y \in \partial f^*(u)$, then lhs of (1) 0 and $x = y$
- ∇f^* is σ-cocoercive: plug $x = \nabla f^*(u)$ and $y = \nabla f^*(v)$ into (1)
- That ∂f^* has full domain follows from Minty's theorem

Duality correspondance

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$. Then the following are equivalent:

(i) f is closed and σ-strongly convex
(ii) ∂f is maximally monotone and σ-strongly monotone
(iii) ∇f^* is σ-cocoercive
(iv) ∇f^* is maximally monotone and $\frac{1}{\sigma}$-Lipschitz continuous
(v) f^* is closed convex and satisfies descent lemma (is $\frac{1}{\sigma}$-smooth)

where $\nabla f^* : \mathbb{R}^n \to \mathbb{R}^n$ and $f^* : \mathbb{R}^n \to \mathbb{R}$

Comments:

- (i) $\Leftrightarrow$ (ii) and (iii) $\Leftrightarrow$ (iv) $\Leftrightarrow$ (v): Previous lecture
- (ii) $\Leftrightarrow$ (iii): This lecture
- Since $f = f^{**}$ the result holds with f and f^* interchanged
- Full proof available on course webpage

Example – Proximal operator is 1-cocoercive

Assume g closed convex, then $\mathrm{prox}_{\gamma g}$ is 1-cocoercive

- Prox definition $\mathrm{prox}_{\gamma g}(z) = \mathrm{argmin}_x(g(x) + \frac{1}{2\gamma}\|x-z\|_2^2)$
- Let $r = \gamma g + \frac{1}{2}\|\cdot\|_2^2$, then

$$\begin{aligned} \mathrm{prox}_{\gamma g}(z) &= \underset{x}{\mathrm{argmin}}(g(x) + \tfrac{1}{2\gamma}\|x-z\|_2^2) \\ &= \underset{x}{\mathrm{argmax}}(-\gamma g(x) - \tfrac{1}{2}\|x-z\|_2^2) \\ &= \underset{x}{\mathrm{argmax}}(z^T x - (\tfrac{1}{2}\|x\|_2^2 + \gamma g(x))) \\ &= \underset{x}{\mathrm{argmax}}(z^T x - r(x)) \\ &= \nabla r^*(z) \end{aligned}$$

  where last step is subdifferential formula for r^* for convex r
- Now, r is 1-strongly convex and $\nabla r^* = \mathrm{prox}_{\gamma g}$ is 1-cocoercive

Example – Proximal operator for strongly convex g

Assume g is σ-strongly convex, then $\mathrm{prox}_{\gamma g}$ is $(1+\gamma\sigma)$-cocoercive

- Let $r = \gamma g + \frac{1}{2}\|\cdot\|_2^2$, and use $\mathrm{prox}_{\gamma g}(z) = \nabla r^*(z)$
- r is $(1+\gamma\sigma)$-strongly convex and ∇r^* is $(1+\gamma\sigma)$-cocoercive

Outline

- Conjugate function – Definition and basic properties
- Examples
- Biconjugate
- Fenchel-Young's inequality
- Duality correspondence
- **Moreau decomposition**
- Duality and optimality conditions
- Weak and strong duality

Moreau decomposition – Statement

Assume g closed convex, then $\mathrm{prox}_g(z) + \mathrm{prox}_{g^*}(z) = z$

- When g scaled by $\gamma > 0$, Moreau decomposition is

$$z = \mathrm{prox}_{\gamma g}(z) + \mathrm{prox}_{(\gamma g)^*}(z) = \mathrm{prox}_{\gamma g}(z) + \gamma \mathrm{prox}_{\gamma^{-1} g^*}(\gamma^{-1} z)$$

  (since $\mathrm{prox}_{(\gamma g)^*} = \gamma \mathrm{prox}_{\gamma^{-1} g^*} \circ \gamma^{-1}\mathrm{Id}$)
- Don't need to know g^* to compute $\mathrm{prox}_{\gamma g^*}$

Moreau decomposition – Proof

- Let $u = z - x$
- Fermat's rule: $x = \mathrm{prox}_g(z)$ if and only if

$$\begin{aligned} 0 \in \partial g(x) + x - z \quad &\Leftrightarrow \quad z - x \in \partial g(x) \\ &\Leftrightarrow \quad u \in \partial g(x) \\ &\Leftrightarrow \quad x \in \partial g^*(u) \\ &\Leftrightarrow \quad z - u \in \partial g^*(u) \\ &\Leftrightarrow \quad 0 \in \partial g^*(u) + u - z \end{aligned}$$

  if and only if $u = \mathrm{prox}_{g^*}(z)$ by Fermat's rule
- Using $z = x + u$, we get

$$z = x + u = \mathrm{prox}_g(z) + \mathrm{prox}_{g^*}(z)$$

Optimality Conditions and Duality

Outline

- Conjugate function – Definition and basic properties
- Examples
- Biconjugate
- Fenchel-Young's inequality
- Duality correspondence
- Moreau decomposition
- **Duality and optimality conditions**
- Weak and strong duality

Composite optimization problem

- Consider *primal* composite optimization problem

$$\mathrm{minimize}\, f(Lx) + g(x)$$

  where f, g closed convex and L is a matrix
- We will derive primal-dual optimality conditions and dual problem

Primal optimality condition

Let $f:\mathbb{R}^m \to \overline{\mathbb{R}}$, $g:\mathbb{R}^n \to \overline{\mathbb{R}}$, $L \in \mathbb{R}^{m\times n}$ with f, g closed convex and assume CQ, then:

$$\text{minimize}\, f(Lx) + g(x)$$

is solved by $x^\star \in \mathbb{R}^n$ if and only if $x^\star$ satisfies

$$0 \in L^T \partial f(Lx^\star) + \partial g(x^\star)$$

- Optimality condition implies that vector s exists such that

$$s \in L^T \partial f(Lx^\star) \qquad \text{and} \qquad -s \in \partial g(x^\star)$$

- So CQ implies a subgradient exists for both functions at solution

Primal-dual optimality condition 1

- Introduce *dual* variable $\mu \in \partial f(Lx)$, then optimality condition

$$0 \in L^T \underbrace{\partial f(Lx)}_{\mu} + \partial g(x)$$

is equivalent to

$$\begin{aligned} \mu &\in \partial f(Lx) \\ -L^T\mu &\in \partial g(x) \end{aligned}$$

- This is a necessary and sufficient primal-dual optimality condition
- (*Primal-dual* since involves primal x and dual μ variables)

Primal-dual optimality condition 2

- Primal-dual optimality condition

$$\begin{aligned} \mu &\in \partial f(Lx) \\ -L^T\mu &\in \partial g(x) \end{aligned}$$

- Using subdifferential inverse:

$$\mu \in \partial f(Lx) \qquad \Longleftrightarrow \qquad Lx \in \partial f^*(\mu)$$

gives equivalent primal dual optimality condition

$$\begin{aligned} Lx &\in \partial f^*(\mu) \\ -L^T\mu &\in \partial g(x) \end{aligned}$$

Dual optimality condition

- Using subdifferential inverse on other condition

$$-L^T\mu \in \partial g(x) \qquad \Longleftrightarrow \qquad x \in \partial g^*(-L^T\mu)$$

gives equivalent primal dual optimality condition

$$\begin{aligned} Lx &\in \partial f^*(\mu) \\ x &\in \partial g^*(-L^T\mu) \end{aligned}$$

- This is equivalent to that:

$$0 \in \partial f^*(\mu) - L\underbrace{\partial g^*(-L^T\mu)}_{x}$$

which is a dual optimality condition since it involves only μ

Dual problem

- The dual optimality condition

$$0 \in \partial f^*(\mu) - L\partial g^*(-L^T\mu)$$

is a sufficient condition for solving the *dual problem*

$$\text{minimize}\, f^*(\mu) + g^*(-L^T\mu)$$

- Have also necessity under CQ on dual, which is mild

Why dual problem?

- Sometimes easier to solve than primal
- Only useful if primal solution can be obtained from dual

Solving primal from dual

- Assume f, g closed convex and CQ holds
- Assume optimal dual μ known: $0 \in \partial f^*(\mu) - L\partial g^*(-L^T\mu)$
- Optimal primal x must satisfy any and all primal-dual conditions:

$$\begin{cases} \mu \in \partial f(Lx) \\ -L^T\mu \in \partial g(x) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ -L^*\mu \in \partial g(x) \end{cases}$$

$$\begin{cases} \mu \in \partial f(Lx) \\ x \in \partial g^*(-L^T\mu) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ x \in \partial g^*(-L^T\mu) \end{cases}$$

- If one of these uniquely characterizes x, then must be solution:
  - g^* is differentiable at $-L^T\mu$ for dual solution μ
  - f^* is differentiable at dual solution μ and L invertible
  - $\ldots$

Optimality conditions – Summary

- Assume f, g closed convex and that CQ holds
- Problem $\min_x f(Lx) + g(x)$ is solved by x if and only if

$$0 \in L^T\partial f(Lx) + \partial g(x)$$

- Primal dual necessary and sufficient optimality conditions:

$$\begin{cases} \mu \in \partial f(Lx) \\ -L^T\mu \in \partial g(x) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ -L^T\mu \in \partial g(x) \end{cases}$$

$$\begin{cases} \mu \in \partial f(Lx) \\ x \in \partial g^*(-L^T\mu) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ x \in \partial g^*(-L^T\mu) \end{cases}$$

- Dual optimality condition

$$0 \in \partial f^*(\mu) - L\partial g^*(-L^T\mu)$$

solves dual problem $\min_\mu f^*(\mu) + g^*(-L^T\mu)$

Outline

- Conjugate function – Definition and basic properties
- Examples
- Biconjugate
- Fenchel-Young's inequality
- Duality correspondence
- Moreau decomposition
- Duality and optimality conditions
- **Weak and strong duality**

Concave dual problem

- We have defined dual as convex minimization problem

$$\underset{\mu}{\text{minimize}}\, f^*(\mu) + g^*(-L^T\mu)$$

- Dual problem can be written as concave maximization problem:

$$\underset{\mu}{\text{maximize}} -f^*(\mu) - g^*(-L^T\mu)$$

- Same solutions but optimal values minus of each other
- Concave formulation gives nicer optimal value comparisons
- To compare, we let the primal and dual optimal values be

$$p^\star = \inf_x (f(Lx) + g(x)) \qquad \text{and} \qquad d^\star = \sup_\mu (-f^*(\mu) - g^*(-L^T\mu))$$

Weak duality

Weak duality always holds meaning $p^\star \geq d^\star$

- We have by Fenchel-Young's inequality for all μ and x:

$$\begin{aligned} f^*(\mu) + g^*(-L^T\mu) &\geq \mu^T Lx - f(Lx) + (-L^T\mu)^T x - g(x) \\ &= -f(Lx) - g(x) \end{aligned}$$

- Negate, maximize lhs over μ, minimize rhs over x, to get

$$d^\star = \sup_\mu (-f^*(\mu) - g^*(-L^T\mu)) \leq \inf_x (f(Lx) + g(x)) = p^\star$$

Strong duality

Assume f, g closed convex, solution $x^\star$ exists, and CQ then *strong duality* holds meaning $p^\star = d^\star$

- Dual $\mu^\star$ and primal $x^\star$ solutions exist such that

$$\mu^\star \in \partial f(Lx^\star) \qquad \text{and} \qquad -L^T\mu^\star \in \partial g(x^\star)$$

- We have by Fenchel-Young's equality:

$$\begin{aligned} p^\star &= f(Lx^\star) + g(x^\star) \\ &= (\mu^\star)^T Lx^\star - f^*(\mu^\star) + (-L^T\mu^\star)^T x^\star - g^*(-L^T\mu^\star) \\ &= -f^*(\mu^\star) - g^*(-L^T\mu^\star) = d^\star \end{aligned}$$

Dual problem gives lower bound

- Consider again concave dual problem with optimal value

$$d^\star = \sup_\mu (-f^*(\mu) - g^*(-L^T\mu))$$

- We know that for all dual variables μ

$$p^\star \geq d^\star \geq -f^*(\mu) - g^*(-L^T\mu)$$

- So can find lower bound to $p^\star$ by evaluating dual objective

Proximal Gradient Method

Pontus Giselsson

Outline

- **Introducing proximal gradient method and examples**
- Solving composite problem – Fixed-points and convergence
- Application to primal and dual problems

Composite optimization problems

- We have introduced the composite optimization problem
$$\underset{x}{\text{minimize}}\, f(Lx) + g(x)$$
- Need an algorithm that solves it - proximal gradient method
- We will consider the simpler composite optimization problem
$$\underset{x}{\text{minimize}}\, f(x) + g(x)$$
that gives the former by letting $f \to f \circ L$

Problem assumptions

- Proximal gradient method works, e.g., for problems that satisfy
  - f is β-smooth $f : \mathbb{R}^n \to \mathbb{R}$ (not necessarily convex)
  - g is closed convex
- Recall that if β-smoothness implies that f satisfies
$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y - x) - \tfrac{\beta}{2}\|y - x\|_2^2$$
it has convex quadratic upper and concave quadratic lower bounds
- If f in addition is convex, we instead have
$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$
where the concave quadratic lower bound is replaced by affine

Minimizing upper bound

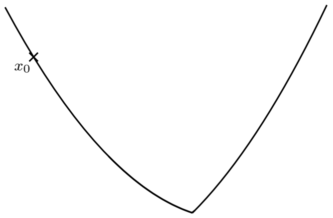
- Due to β-smoothness of f, we have
$$f(y) + g(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2 + g(y)$$
for all $x, y \in \mathbb{R}^n$, i.e., r.h.s. is upper bound to l.h.s.
- Minimizing in every iteration the r.h.s. w.r.t. y for given x gives
$$\begin{aligned} v &= \underset{y}{\operatorname{argmin}} \left(f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2 + g(y)\right) \\ &= \underset{y}{\operatorname{argmin}} \left(g(y) + \tfrac{\beta}{2}\|y - (x - \beta^{-1}\nabla f(x))\|_2^2\right) \\ &= \operatorname{prox}_{\beta^{-1}g}(x - \beta^{-1}\nabla f(x)) \end{aligned}$$

Proximal gradient method

- Let us replace β by γ_k^{-1}, x by x_k, and v by x_{k+1} to get:
$$\begin{aligned} x_{k+1} &= \underset{y}{\operatorname{argmin}} \left(f(x_k) + \nabla f(x_k)^T(y - x_k) + \tfrac{1}{2\gamma_k}\|y - x_k\|_2^2 + g(y)\right) \\ &= \underset{y}{\operatorname{argmin}} \left(g(y) + \tfrac{1}{2\gamma_k}\|y - (x_k - \gamma_k\nabla f(x_k))\|_2^2\right) \\ &= \operatorname{prox}_{\gamma_k g}(x_k - \gamma_k\nabla f(x_k)) \end{aligned}$$
- This is exactly the proximal gradient method
- The method replaces f by quadratic approximation and minimizes
- (Note that we need an initial guess x_0 to start the iteration)

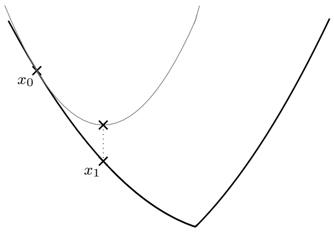
Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\text{minimize}}\, \frac{1}{2}(x - a)^2 + |x|$
- $f(x) = \frac{1}{2}(x - a)^2$ is smooth term and $g(x) = |x|$ is nonsmooth
- Iteration: $x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



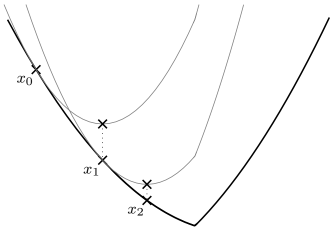
Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\text{minimize}}\, \frac{1}{2}(x - a)^2 + |x|$
- $f(x) = \frac{1}{2}(x - a)^2$ is smooth term and $g(x) = |x|$ is nonsmooth
- Iteration: $x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



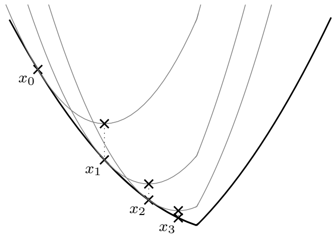
Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\mathrm{minimize}}\,\frac{1}{2}(x-a)^2+|x|$
- $f(x)=\frac{1}{2}(x-a)^2$ is smooth term and $g(x)=|x|$ is nonsmooth
- Iteration: $x_{k+1}=\mathrm{prox}_{\gamma g}(x_k-\gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



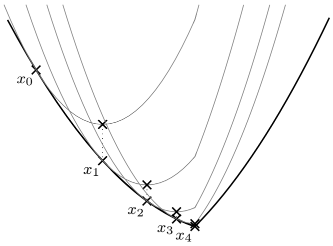
Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\mathrm{minimize}}\,\frac{1}{2}(x-a)^2+|x|$
- $f(x)=\frac{1}{2}(x-a)^2$ is smooth term and $g(x)=|x|$ is nonsmooth
- Iteration: $x_{k+1}=\mathrm{prox}_{\gamma g}(x_k-\gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



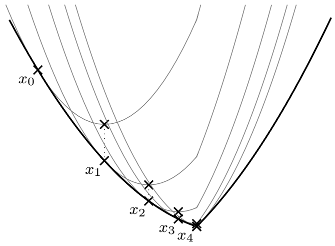
Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\mathrm{minimize}}\,\frac{1}{2}(x-a)^2+|x|$
- $f(x)=\frac{1}{2}(x-a)^2$ is smooth term and $g(x)=|x|$ is nonsmooth
- Iteration: $x_{k+1}=\mathrm{prox}_{\gamma g}(x_k-\gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



Proximal gradient – Example

- Proximal gradient iterations for problem $\underset{x}{\mathrm{minimize}}\,\frac{1}{2}(x-a)^2+|x|$
- $f(x)=\frac{1}{2}(x-a)^2$ is smooth term and $g(x)=|x|$ is nonsmooth
- Iteration: $x_{k+1}=\mathrm{prox}_{\gamma g}(x_k-\gamma\nabla f(x_k))$
- Note: convergence in finite number of iterations (not always)



Proximal gradient – Special cases

- Proximal gradient method:
  - solves $\underset{x}{\mathrm{minimize}}(f(x)+g(x))$
  - iteration: $x_{k+1}=\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))$
- Proximal gradient method with $g=0$:
  - solves $\underset{x}{\mathrm{minimize}}(f(x))$
  - $\mathrm{prox}_{\gamma_k g}(z)=\mathrm{argmin}_x(0+\frac{1}{2\gamma}\|x-z\|_2^2)=z$
  - iteration: $x_{k+1}=\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))=x_k-\gamma_k\nabla f(x_k)$
  - reduces to gradient method
- Proximal gradient method with $f=0$:
  - solves $\underset{x}{\mathrm{minimize}}(g(x))$
  - $\nabla f(x)=0$
  - iteration: $x_{k+1}=\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))=\mathrm{prox}_{\gamma_k g}(x_k)$
  - reduces to *proximal point method* (which is not very useful)

Outline

- Introducing proximal gradient method and examples
- **Solving composite problem – Fixed-points and convergence**
- Application to primal and dual problems

Proximal gradient method – Fixed-point set

- Proximal gradient step

$$x_{k+1}=\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))$$

- If $x_{k+1}=x_k$, they are in *proximal gradient fixed-point set*

$$\{x : x=\mathrm{prox}_{\gamma g}(x-\gamma\nabla f(x))\}$$

- Under some assumptions, algorithm will satisfy $x_{k+1}-x_k\to 0$
  - this means that fixed-point equation will be satisfied in limit
  - what does it mean for x to be a fixed-point?

Proximal gradient – Optimality condition

- Proximal gradient step:

$$v=\mathrm{prox}_{\gamma g}(x-\gamma\nabla f(x))=\underset{y}{\mathrm{argmin}}(g(y)+\underbrace{\tfrac{1}{2\gamma}\|y-(x-\gamma\nabla f(x))\|_2^2}_{h(y)})$$

  where v is unique due to strong convexity of h
- Fermat's rule (since CQ holds) gives $v=\mathrm{prox}_{\gamma g}(x-\gamma\nabla f(x))$ iff:

$$\begin{aligned}0&\in\partial g(v)+\partial h(v)\\&=\partial g(v)+\gamma^{-1}(v-(x-\gamma\nabla f(x)))\\&=\partial g(v)+\nabla f(x)+\gamma^{-1}(v-x)\end{aligned}$$

  since h differentiable

Proximal gradient – Fixed-point characterization

For $\gamma > 0$, we have that

$$\bar{x} = \mathrm{prox}_{\gamma g}(\bar{x} - \gamma\nabla f(\bar{x})) \quad \text{if and only if} \quad 0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$$

- Proof: the proximal step equivalence

  $$v = \mathrm{prox}_{\gamma g}(x - \gamma\nabla f(x)) \quad \Leftrightarrow \quad 0 \in \partial g(v) + \nabla f(x) + \gamma^{-1}(v - x)$$

  evaluated at a fixed-point $x = v = \bar{x}$ reads

  $$\bar{x} = \mathrm{prox}_{\gamma g}(\bar{x} - \gamma\nabla f(\bar{x})) \quad \Leftrightarrow \quad 0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$$
- We call inclusion $0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$ *fixed-point characterization*

Meaning of fixed-point characterization

- What does fixed-point characterization $0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$ mean?
- For convex differentiable f, subdifferential $\partial f(x) = \{\nabla f(x)\}$ and

  $$0 \in \partial f(\bar{x}) + \partial g(\bar{x}) = \partial(f + g)(\bar{x})$$

  (subdifferential sum rule holds), i.e., fixed-points solve problem
- For nonconvex differentiable f, we might have $\partial f(\bar{x}) = \emptyset$
  - Fixed-point are not in general global solutions
  - Points $\bar{x}$ that satisfy $0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$ are called *critical points*
  - If $g = 0$, the condition is $\nabla f(\bar{x}) = 0$, i.e., a *stationary point*
- Quality of fixed-points differs between convex and nonconvex f

Conditions on γ_k for convergence

- We replace in proximal gradient method $f(y)$ by

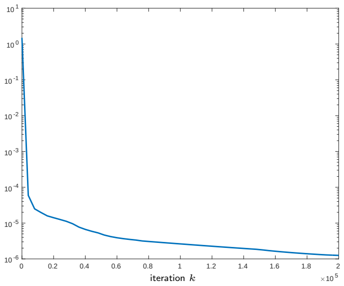
  $$f(x_k) + \nabla f(x_k)^T(y - x_k) + \tfrac{1}{2\gamma_k}\|y - x_k\|_2^2$$

  and minimize this plus $g(y)$ over y to get the next iterate
- We know from β-smoothness of f that for all x, y

  $$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2$$
- If $\gamma_k \in [\epsilon, \frac{1}{\beta}]$ with $\epsilon > 0$, an upper bound is minimized
- Can use $\gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$ and show convergence of some quantity

Practical convergence – Example

- Logarithmic y axis of quantity that should go to 0 for convergence
- Linear x axis with iteration number



- Fast convergence to medium accuracy, slow from medium to high
- Many iterations may be required

Stopping conditions

- For β-smooth $f : \mathbb{R}^n \to \mathbb{R}$, we can stop algorithm when

  $$\tfrac{1}{\beta}u_k := \tfrac{1}{\beta}(\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k))$$

  is small (notation and reason will be motivated in future lecture)
- This is the plotted quantity on the previous slide
- We can use absolute or relative stopping conditions:
  - absolute stopping conditions with small $\epsilon_{\rm abs} > 0$

    $$\tfrac{1}{\beta}\|u_k\|_2 \leq \epsilon_{\rm abs} \quad \text{or} \quad \tfrac{1}{\beta}\|u_k\|_2 \leq \epsilon_{\rm abs}\sqrt{n}$$
  - relative stopping condition with small $\epsilon_{\rm rel}, \epsilon > 0$:

    $$\tfrac{1}{\beta}\tfrac{\|u_k\|_2}{\|x_k\|_2 + \beta^{-1}\|\nabla f(x_k)\|_2 + \epsilon} \leq \epsilon_{\rm rel}$$
- Problem considered solved to optimality if, say, $\frac{1}{\beta}\|u_k\|_2 \leq 10^{-6}$
- Often lower accuracy of 10^{-3} or 10^{-4} is enough

Outline

- Introducing proximal gradient method and examples
- Solving composite problem – Fixed-points and convergence
- **Application to primal and dual problems**

Applying proximal gradient to primal problems

Problem $\underset{x}{\mathrm{minimize}}\, f(x) + g(x)$:

- Assumptions:
  - f smooth
  - g closed convex and prox friendly[1]
- Algorithm: $x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k\nabla f(x_k))$

Problem $\underset{x}{\mathrm{minimize}}\, f(Lx) + g(x)$:

- Assumptions:
  - f smooth (implies $f \circ L$ smooth)
  - g closed convex and prox friendly[1]
- Gradient $\nabla(f \circ L)(x) = L^T\nabla f(Lx)$
- Algorithm: $x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k L^T\nabla f(Lx_k))$

---

[1] Prox friendly: proximal operator cheap to evaluate, e.g., g separable

Applying proximal gradient to dual problem

- Let us apply the proximal gradient method to the dual problem

  $$\underset{\mu}{\mathrm{minimize}}\, f^*(\mu) + g^*(-L^T\mu)$$
- Assumptions:
  - f: closed convex and prox friendly
  - g: σ-strongly convex
- Why these assumptions?
  - f^*: closed convex and prox friendly
  - $g^* \circ -L^T$: $\frac{\|L\|_2^2}{\sigma}$-smooth and convex
- Algorithm:

  $$\mu_{k+1} = \mathrm{prox}_{\gamma_k f^*}(\mu_k - \gamma_k\nabla(g^* \circ -L^T)(\mu_k))$$

Dual proximal gradient method – Explicit version 1

- We will make the dual proximal gradient method more explicit

$$\mu_{k+1} = \mathrm{prox}_{\gamma_k f^*}(\mu_k - \gamma_k \nabla(g^* \circ -L^T)(\mu_k))$$

- Use $\nabla(g^* \circ -L^T)(\mu) = -L\nabla g^*(-L^T\mu)$ to get

$$\begin{aligned} x_k &= \nabla g^*(-L^T\mu_k) \\ \mu_{k+1} &= \mathrm{prox}_{\gamma_k f^*}(\mu_k + \gamma_k L x_k) \end{aligned}$$

Dual proximal gradient method – Explicit version 2

- Restating the previous formulation

$$\begin{aligned} x_k &= \nabla g^*(-L^T\mu_k) \\ \mu_{k+1} &= \mathrm{prox}_{\gamma_k f^*}(\mu_k + \gamma_k L x_k) \end{aligned}$$

- Use Moreau decomposition for prox:

$$\mathrm{prox}_{\gamma f^*}(v) = v - \gamma \mathrm{prox}_{\gamma^{-1} f}(\gamma^{-1} v)$$

  to get

$$\begin{aligned} x_k &= \nabla g^*(-L^T\mu_k) \\ v_k &= \mu_k + \gamma_k L x_k \\ \mu_{k+1} &= v_k - \gamma_k \mathrm{prox}_{\gamma_k^{-1} f}(\gamma_k^{-1} v_k) \end{aligned}$$

Dual proximal gradient method – Explicit version 3

- Restating the previous formulation

$$\begin{aligned} x_k &= \nabla g^*(-L^T\mu_k) \\ v_k &= \mu_k + \gamma_k L x_k \\ \mu_{k+1} &= v_k - \gamma_k \mathrm{prox}_{\gamma_k^{-1} f}(\gamma_k^{-1} v_k) \end{aligned}$$

- Use subdifferential formula, since g^* differentiable:

$$\nabla g^*(\nu) = \underset{x}{\mathrm{argmax}}(\nu^T x - g(x)) = \underset{x}{\mathrm{argmin}}(g(x) - \nu^T x)$$

  with $\nu = -L^T\mu_k$ to get

$$\begin{aligned} x_k &= \underset{x}{\mathrm{argmin}}(g(x) + (\mu_k)^T L x) \\ v_k &= \mu_k + \gamma_k L x_k \\ \mu_{k+1} &= v_k - \gamma_k \mathrm{prox}_{\gamma_k^{-1} f}(\gamma_k^{-1} v_k) \end{aligned}$$

- Can implement method without computing conjugate functions

Dual proximal gradient method – Primal recovery

- Can we recover a primal solution from dual prox grad method?
- Let us use explicit version 1

$$\begin{aligned} x_k &= \nabla g^*(-L^T\mu_k) \\ \mu_{k+1} &= \mathrm{prox}_{\gamma_k f^*}(\mu_k + \gamma_k L x_k) \end{aligned}$$

  and assume we have found fixed-point $(\bar{x}, \bar{\mu})$: for some $\bar{\gamma} > 0$,

$$\begin{aligned} \bar{x} &= \nabla g^*(-L^T\bar{\mu}) \\ \bar{\mu} &= \mathrm{prox}_{\bar{\gamma} f^*}(\bar{\mu} + \bar{\gamma} L \bar{x}) \end{aligned}$$

- Fermat's rule for proximal step

$$0 \in \partial f^*(\bar{\mu}) + \bar{\gamma}^{-1}(\bar{\mu} - (\bar{\mu} + \bar{\gamma} L \bar{x})) = \partial f^*(\bar{\mu}) - L\bar{x}$$

  is with $\bar{x} = \nabla g^*(-L^T\bar{\mu})$ a primal-dual optimality condition
- So x_k will solve primal problem if algorithm converges

Problems that prox-grad cannot solve

- Problem $\underset{x}{\mathrm{minimize}}\, f(x) + g(x)$
- Assumptions: f and g convex but nondifferentiable
- No term differentiable, another method must be used:
  - Subgradient method
  - Douglas-Rachford splitting
  - Primal-dual methods

Problems that prox-grad cannot solve efficiently

- Problem $\underset{x}{\mathrm{minimize}}\, f(x) + g(Lx)$
- Assumptions:
  - f smooth
  - g nonsmooth convex
  - L arbitrary structured matrix
- Can apply proximal gradient method

$$x_{k+1} = \underset{y}{\mathrm{argmin}}(g(Ly) + \tfrac{1}{2\gamma_k}\|y - (x_k - \gamma_k \nabla f(x_k))\|_2^2)$$

  but proximal operator of $g \circ L$

$$\mathrm{prox}_{\gamma(g \circ L)}(z) = \underset{x}{\mathrm{argmin}}(g(Lx) + \tfrac{1}{2\gamma}\|x - z\|_2^2)$$

  often not "prox friendly", i.e., it is expensive to evaluate

Least Squares

Pontus Giselsson

Outline

- **Supervised learning – Overview**
- Least squares – Basics
- Nonlinear features
- Generalization, overfitting, and regularization
- Cross validation
- Feature selection
- Training problem properties

Machine learning

- Machine learning can very roughly be divided into:
  - Supervised learning
  - Unsupervised learning
  - Semisupervised learning (between supervised and unsupervised)
  - Reinforcement learning
- We will focus on supervised learning

Supervised learning

- Let (x, y) represent object and label pairs
  - Object $x \in \mathcal{X} \subseteq \mathbb{R}^n$
  - Label $y \in \mathcal{Y} \subseteq \mathbb{R}^K$
- Available: Labeled training data (training set) $\{(x_i, y_i)\}_{i=1}^N$
  - Data $x_i \in \mathbb{R}^n$, or *examples* (often n large)
  - Labels $y_i \in \mathbb{R}^K$, or *response variables* (often $K = 1$)

Objective: Find a model (function) $m(x)$:

- that takes data (example, object) x as input
- and predicts corresponding label (response variable) y

How?:

- learn m from training data, but should *generalize* to all (x, y)

Relation to optimization

Training the “machine” m consists in solving optimization problem

Regression vs Classification

There are two main types of supervised learning tasks:

- Regression:
  - Predicts quantities
  - Real-valued labels $y \in \mathcal{Y} = \mathbb{R}^K$ (will mainly consider $K = 1$)
- Classification:
  - Predicts class belonging
  - Finite number of class labels, e.g., $y \in \mathcal{Y} = \{1, 2, \ldots, k\}$

Examples of data and label pairs

Data	Label	R/C
text in email	spam?	C
dna	blood cell concentration	R
dna	cancer?	C
image	cat or dog	C
advertisement display	click?	C
image of handwritten digit	digit	C
house address	selling cost	R
stock	price	R
sport analytics	winner	C
speech representation	spoken word	C

R/C is for regression or classification

In this course

Lectures will cover different supervised learning methods:

- Classical methods with convex training problems
  - Least squares (this lecture)
  - Logistic regression
  - Support vector machines
- Deep learning methods with nonconvex training problem

Highlight difference:

- Deep learning (specific) nonlinear model instead of linear

Notation

- (Primal) Optimization variable notation:
  - Optimization literature: x, y, z (as in first part of course)
  - Statistics literature: β
  - Machine learning literature: θ, w, b
- Reason: data, labels in statistics and machine learning are x, y
- Will use machine learning notation in these lectures
- We collect training data in matrices (one example per row)

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \qquad\qquad Y = \begin{bmatrix} y_1^T \\ \vdots \\ y_N^T \end{bmatrix}$$

- Columns X_j of data matrix $X = [X_1, \ldots, X_n]$ are called *features*

Outline

- Supervised learning – Overview
- **Least squares – Basics**
- Nonlinear features
- Generalization, overfitting, and regularization
- Cross validation
- Feature selection
- Training problem properties

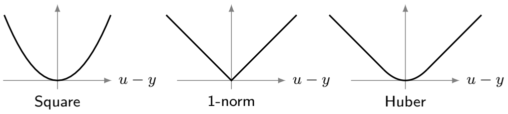
Regression training problem

- Objective: Find data model m such that for all (x, y):

$$m(x) - y \approx 0$$

- Let model output $u = m(x)$; Examples of data misfit losses

$$L(u, y) = \tfrac{1}{2}(u - y)^2$$
$$L(u, y) = |u - y|$$
$$L(u, y) = \begin{cases} \tfrac{1}{2}(u - y)^2 & \text{if } |u - v| \leq c \\ c(|u - y| - c/2) & \text{else} \end{cases}$$



Square 1-norm Huber

- Training: find model m that minimizes sum of training set losses

$$\underset{m}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i), y_i)$$

Supervised learning – Least squares

- Parameterize model m and set a linear (affine) structure

$$m(x; \theta) = w^T x + b$$

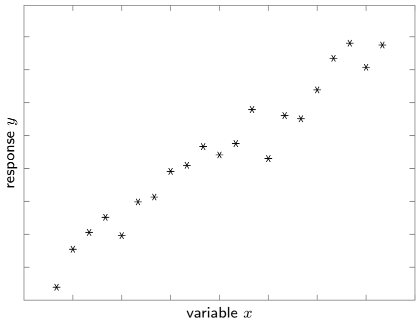
  where $\theta = (w, b)$ are *parameters* (also called *weights*)
- Training: find model parameters that minimize training cost

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i; \theta), y_i) = \tfrac{1}{2} \sum_{i=1}^{N} (w^T x_i + b - y_i)^2$$

  (note: optimization over model *parameters* θ)
- Once trained, predict response of new input x as $\hat{y} = w^T x + b$

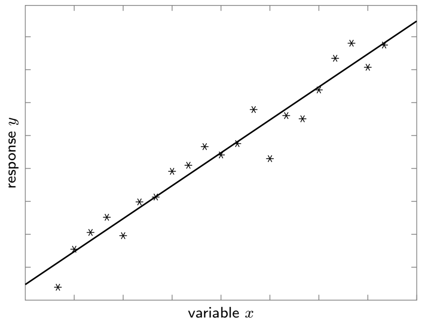
Example – Least squares

- Find affine function parameters that fit data:



Example – Least squares

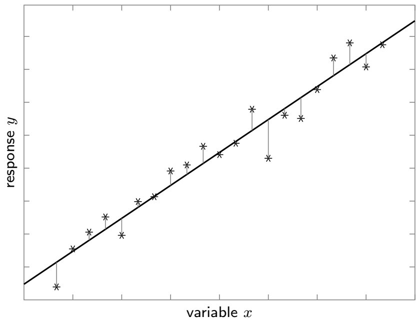
- Find affine function parameters that fit data:



- Data points (x, y) marked with ($*$), LS model $wx + b$ (—–)

Example – Least squares

- Find affine function parameters that fit data:



- Data points (x, y) marked with ($*$), LS model $wx + b$ (—–)
- Least squares finds affine function that minimizes squared distance

Solving for constant term

- Constant term b also called *bias term* or *intercept*
- What is optimal b?

$$\underset{w,b}{\text{minimize}} \tfrac{1}{2} \sum_{i=1}^{N} (w^T x_i + b - y_i)^2$$

- Optimality condition w.r.t. b (gradient w.r.t. b is 0):

$$0 = Nb + \sum_{i=1}^{N} (w^T x_i - y_i) \quad \Leftrightarrow \quad b = \bar{y} - w^T \bar{x}$$

  where $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$ and $\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i$ are mean values

Equivalent problem

- Plugging in optimal $b = \bar{y} - w^T\bar{x}$ in least squares estimate gives

$$\underset{w,b}{\text{minimize}}\ \tfrac{1}{2}\sum_{i=1}^{N}(w^Tx_i + b - y_i)^2 = \tfrac{1}{2}\sum_{i=1}^{N}(w^T(x_i - \bar{x}) - (y_i - \bar{y}))^2$$

- Let $\tilde{x}_i = x_i - \bar{x}$ and $\tilde{y}_i = y_i - \bar{y}$, then it is equivalent to solve

$$\underset{w}{\text{minimize}}\ \tfrac{1}{2}\sum_{i=1}^{N}(w^T\tilde{x}_i - \tilde{y}_i)^2 = \tfrac{1}{2}\|Xw - Y\|_2^2$$

  where X and Y now contain all $\tilde{x}_i$ and $\tilde{y}_i$ respectively
- Obviously $\tilde{x}_i$ and $\tilde{y}_i$ have zero averages (by construction)
- Will often assume averages subtracted from data and responses

Least squares – Solution

- Training problem

$$\underset{w}{\text{minimize}}\ \tfrac{1}{2}\|Xw - Y\|_2^2$$

- Strongly convex if X full column rank
  - Features linearly independent and more examples than features
  - Consequences: X^TX is invertible and solution exists and is unique
- Optimal w satisfies (set gradient to zero)

$$0 = X^TXw - X^TY$$

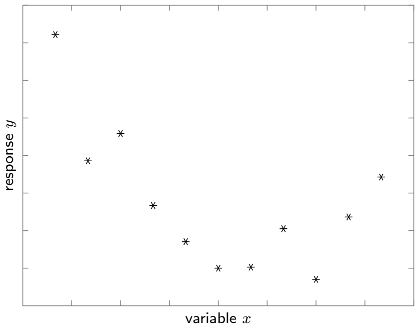
  if X full column rank, then unique solution $w = (X^TX)^{-1}X^TY$

Outline

- Supervised learning – Overview
- Least squares – Basics
- **Nonlinear features**
- Generalization, overfitting, and regularization
- Cross validation
- Feature selection
- Training problem properties

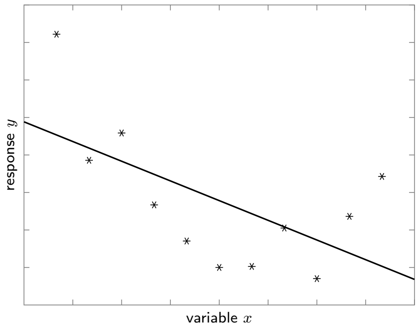
Nonaffine example

- What if data that cannot be well approximated by affine mapping?



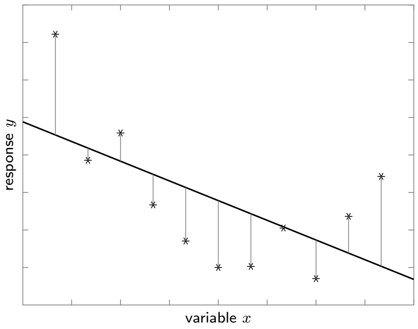
Nonaffine example

- What if data that cannot be well approximated by affine mapping?



Nonaffine example

- What if data that cannot be well approximated by affine mapping?



Adding nonlinear features

- A linear model is not rich enough to model relationship
- Try, e.g., a quadratic model

$$m(x;\theta) = b + \sum_{i=1}^{n} w_ix_i + \sum_{i=1}^{n}\sum_{j=1}^{i} q_{ij}x_ix_j$$

  where $x = (x_1, \ldots, x_n)$ and parameters $\theta = (b, w, q)$
- For $x \in \mathbb{R}^2$, the model is

$$m(x;\theta) = b + w_1x_1 + w_2x_2 + q_{11}x_1^2 + q_{12}x_1x_2 + q_{22}x_2^2 = \theta^T\phi(x)$$

  where $x = (x_1, x_2)$ and

$$\theta = (b, w_1, w_2, q_{11}, q_{12}, q_{22})$$
$$\phi(x) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$$

- Add *nonlinear features* $\phi(x)$, but model still *linear in parameter* θ

Least squares with nonlinear features

- Can, of course, use other nonlinear feature maps ϕ
- Gives models $m(x;\theta) = \theta^T\phi(x)$ with increased fitting capacity
- Use least squares estimate with new model

$$\underset{\theta}{\text{minimize}}\ \tfrac{1}{2}\sum_{i=1}^{N}(m(x_i;\theta) - y_i)^2 = \tfrac{1}{2}\sum_{i=1}^{N}(\theta^T\phi(x_i) - y_i)^2$$

  which is still convex since ϕ does not depend on θ!
- Build new data matrix (with one column per feature in ϕ)

$$X = \begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}$$

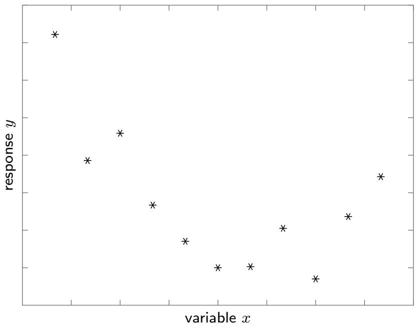
  to arrive at least squares formulation

$$\underset{\theta}{\text{minimize}}\ \tfrac{1}{2}\|X\theta - Y\|_2^2$$

- The more features, the more parameters θ to optimize (lifting)

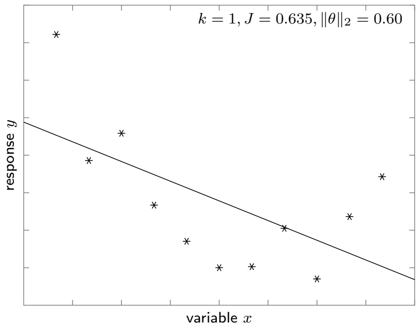
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



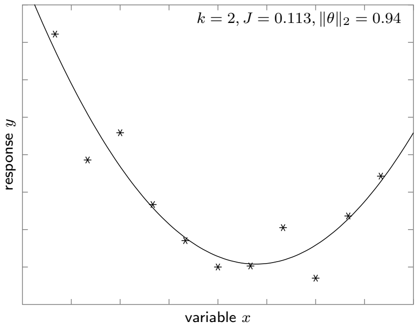
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



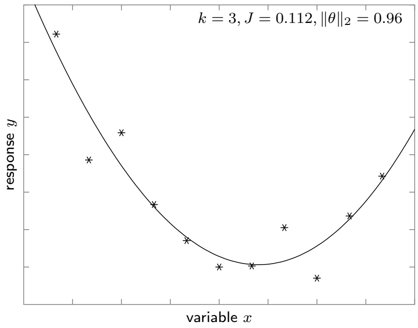
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



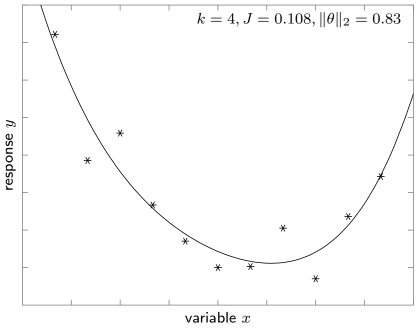
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



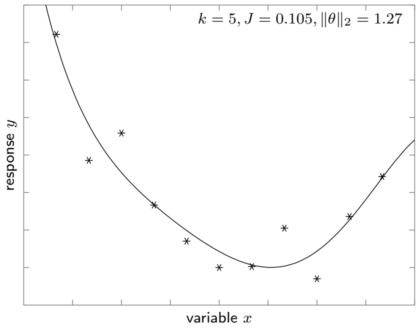
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



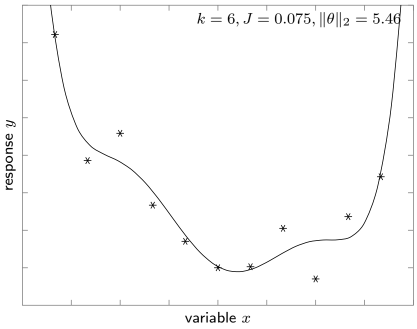
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



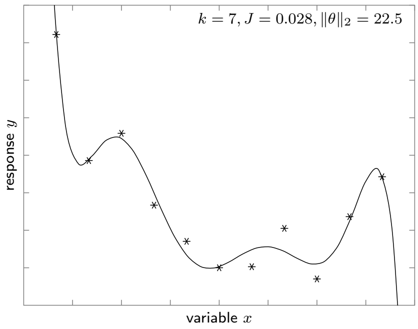
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



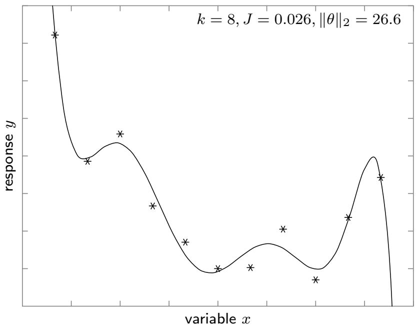
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



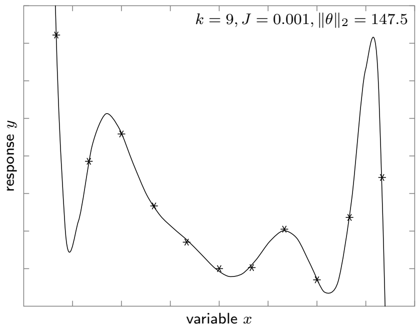
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



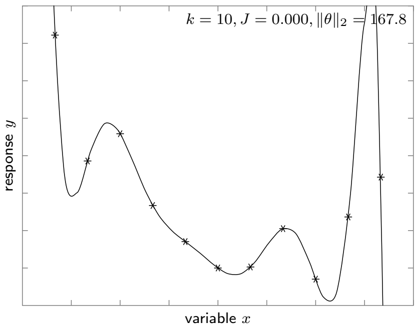
Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



Nonaffine example

- Fit polynomial of degree k to data using LS (J is cost):



Outline

- Supervised learning – Overview
- Least squares – Basics
- Nonlinear features
- **Generalization, overfitting, and regularization**
- Cross validation
- Feature selection
- Training problem properties

Generalization and overfitting

- *Generalization*: How well does model perform on unseen data
- *Overfitting*: Model explains training data, but not unseen data
- How to reduce overfitting/improve generalization?

Tikhonov Regularization

- Example indicates: Reducing $\|\theta\|_2$ seems to reduce overfitting
- Least squares with *Tikhonov regularization*:

$$\underset{\theta}{\text{minimize}}\ \tfrac{1}{2}\|X\theta - Y\|_2^2 + \tfrac{\lambda}{2}\|\theta\|_2^2$$

- Regularization parameter $\lambda \geq 0$ controls fit vs model expressivity
- Optimization problem called ridge regression in statistics
- (Could regularize with $\|\theta\|_2$, but square easier to solve)
- (Don't regularize b – constant data offset gives different solution)

Ridge Regression – Solution

- Recall ridge regression problem for given λ:

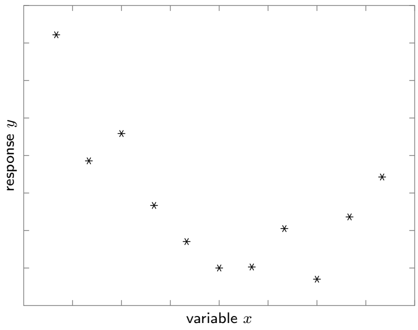
$$\underset{\theta}{\text{minimize}}\ \tfrac{1}{2}\|X\theta - Y\|_2^2 + \tfrac{\lambda}{2}\|\theta\|_2^2$$

- Objective λ-strongly convex for all $\lambda > 0$, hence unique solution
- Objective is differentiable, Fermat's rule:

$$\begin{aligned} 0 = X^T(X\theta - Y) + \lambda\theta \quad &\iff \quad (X^TX + \lambda I)\theta = X^TY \\ &\iff \quad \theta = (X^TX + \lambda I)^{-1}X^TY \end{aligned}$$

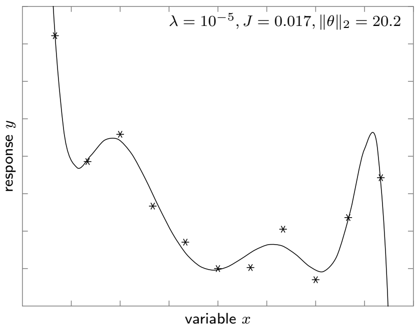
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



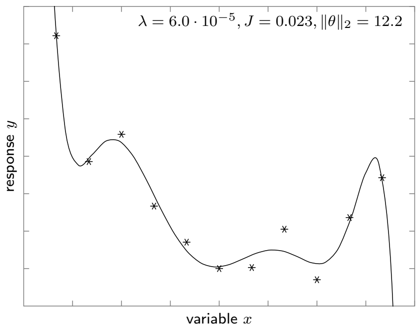
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



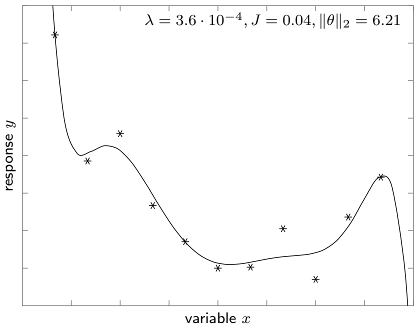
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



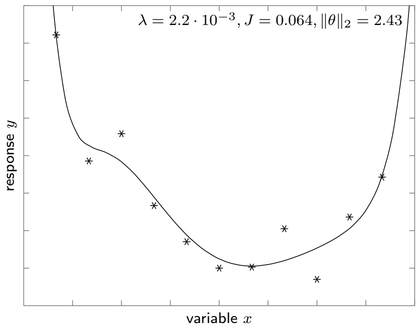
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



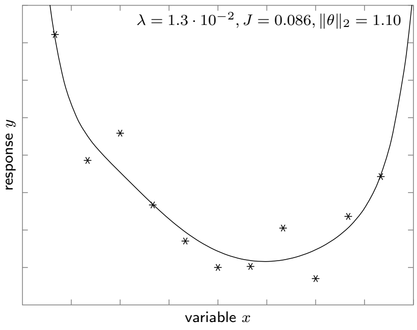
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



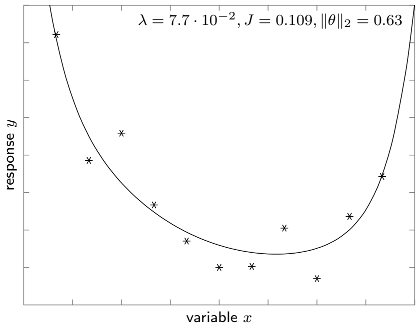
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



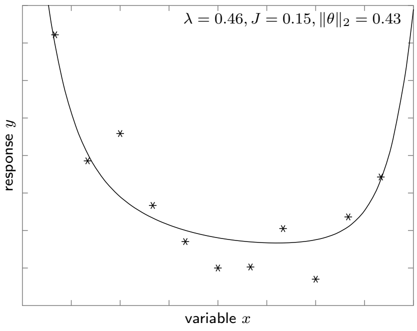
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



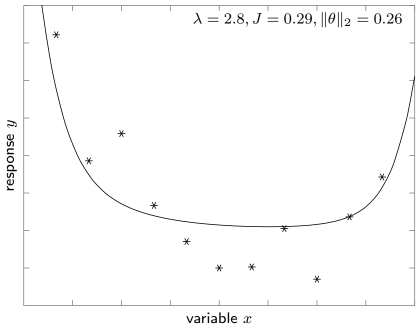
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



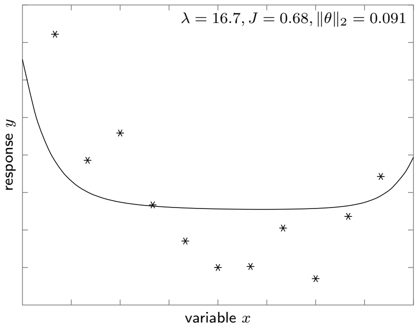
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



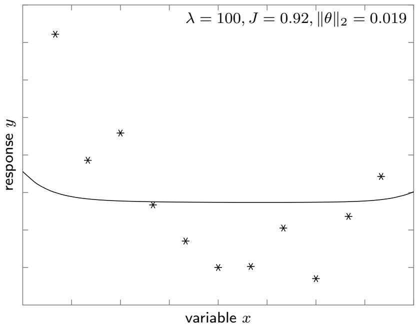
Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



Ridge Regression – Example

- Same problem data as before
- Fit 10-degree polynomial with Tikhonov regularization
- λ: regularization parameter, J LS cost, $\|\theta\|_2$ norm of weights



Outline

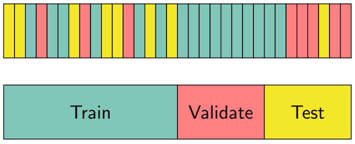
- Supervised learning – Overview
- Least squares – Basics
- Nonlinear features
- Generalization, overfitting, and regularization
- **Cross validation**
- Feature selection
- Training problem properties

Selecting model hyperparameters

- Parameters in machine learning models are called *hyperparameters*
- Ridge model has polynomial order and λ as hyperparameters
- How to select hyperparameters?

Holdout

- Randomize data and assign to train, validate, or test set



Training set:

- Solve training problems with different hyperparameters

Validation set:

- Estimate generalization performance of all trained models
- Use this to select model that seems to generalize best

Test set:

- Final assessment on how chosen model generalizes to unseen data
- *Not* for model selection, then final assessment too optimistic

---

Holdout – Comments

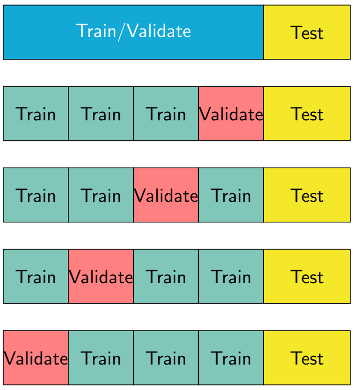
- Typical division between sets 50/25/25 (or 70/20/10)
- Sometimes no test set (then no assessment of final model)
- If no test set, then validation set often called test set
- Can work well if lots of data, if less, use *(k-fold) cross validation*

---

k-fold cross validation

- Similar to hold out – divide first into training/validate and test set
- Divide training/validate set into k data chunks
- Train k models with $k-1$ chunks, use k:th chunk for validation
- Loop
    1. Set hyperparameters and train all k models
    2. Evaluate generalization score on its validation data
    3. Sum scores to get model performance
- Select final model hyperparameters based on best score
- Simpler model with slightly worse score may generalize better
- Estimate generalization performance via test set

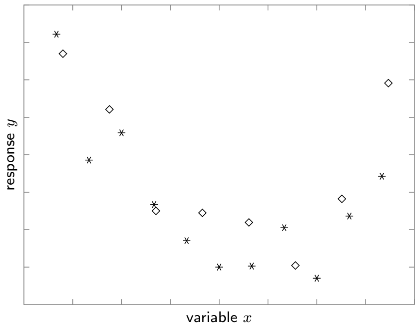
4-fold cross validation – Graphics



Train/Validate				Test
Train	Train	Train	Validate	Test
Train	Train	Validate	Train	Test
Train	Validate	Train	Train	Test
Validate	Train	Train	Train	Test

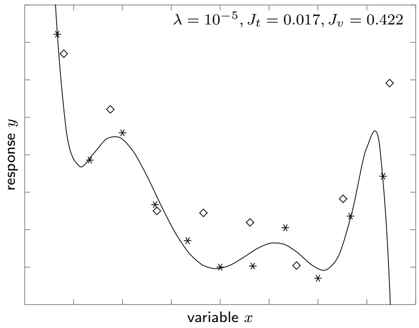
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



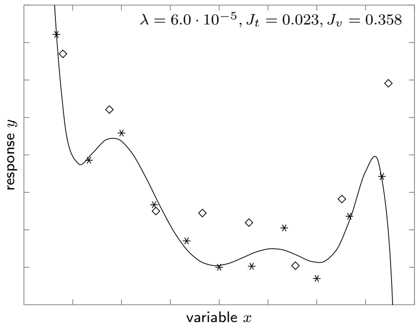
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



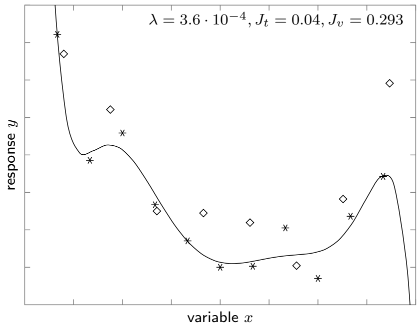
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



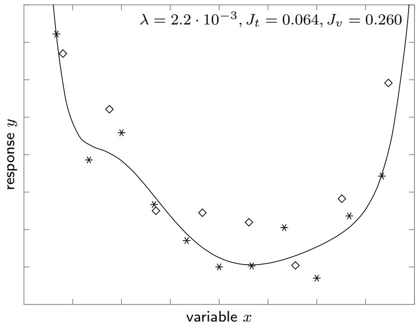
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



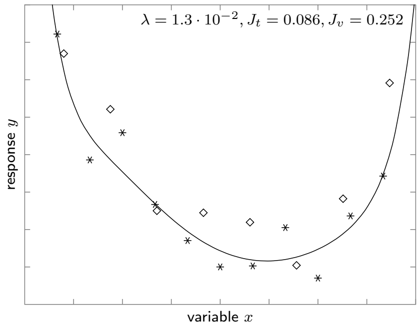
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



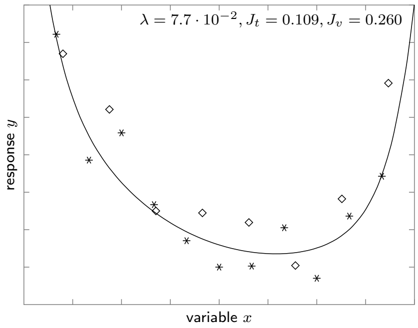
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



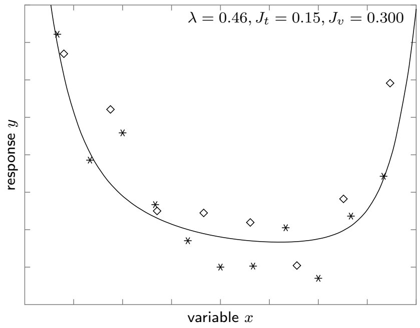
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



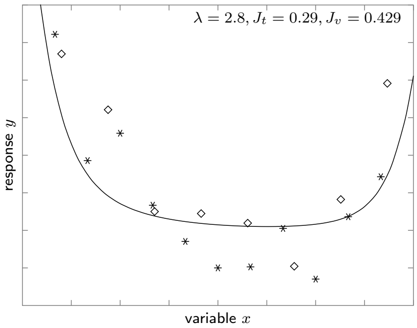
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



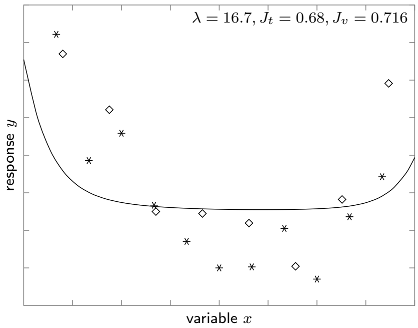
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



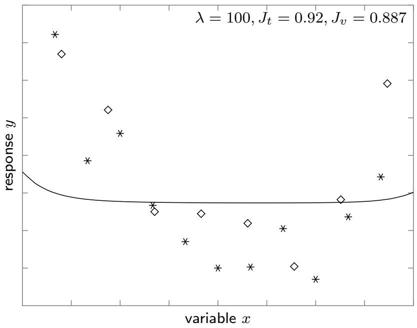
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



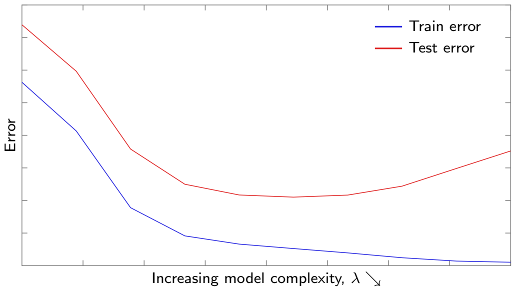
Evaluate generalization score/performance

- Ridge regression example generalization, validation data (◇)
- λ: regularization parameter, J_t train cost, J_v validation cost



Selecting model

- Average training and test error vs model complexity
- Average training error smaller than average test error
- Large λ (left) model not rich enough
- Small λ (right) model too rich (overfitting)



Outline

- Supervised learning – Overview
- Least squares – Basics
- Nonlinear features
- Generalization, overfitting, and regularization
- Cross validation
- **Feature selection**
- Training problem properties

Feature selection

- Assume $X \in \mathbb{R}^{m \times n}$ with $m < n$ (fewer examples than features)
- Want to find a subset of features that explains data well
- Example: Which genes in genome control eyecolor

Lasso

- Feature selection by regularizing least squares with 1-norm:

$$\underset{w}{\text{minimize}}\ \tfrac{1}{2}\|Xw - Y\|_2^2 + \lambda\|w\|_1$$

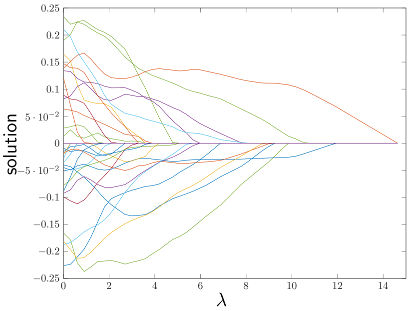
- Problem can be written as

$$\underset{w}{\text{minimize}}\ \tfrac{1}{2}\left\|\sum_{i=1}^{n} w_i X_i - Y\right\|_2^2 + \lambda\|w\|_1$$

  if $w_i = 0$, then feature X_i not important
- The 1-norm promotes sparsity (many 0 variables) in solution
- It also reduces size (shrinks) w (like $\|\cdot\|_2^2$ regularization)
- Problem is called the *Lasso* problem

Example – Lasso

- Data $X \in \mathbb{R}^{30 \times 200}$, Lasso solution for different λ



- For large enough λ solution $w = 0$
- More nonzero elements in solution as λ decreases
- For small λ, 30 (nbr examples) nonzero w_i (i.e., 170 $w_i = 0$)

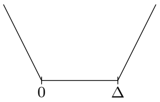
Lasso and correlated features

- Assume two equal features exist, e.g., $X_1 = X_2$, lasso problem is

$$\text{minimize} \, \tfrac{1}{2} \left\| (w_1 + w_2) X_1 + \sum_{i=3}^{n} w_i X_i - Y \right\|_2^2 + \lambda(|w_1| + |w_2| + \|w_{3:n}\|_1)$$

- Assume w^* solves the problem and let $\Delta := w_1^* + w_2^* > 0$ (wlog)
- Then all $w_1 \in [0, \Delta]$ with $w_2 = \Delta - w_1$ solves problem:
  - quadratic cost unchanged since sum $w_1 + w_2$ still Δ
  - the remainder of the regularization part reduces to

$$\min_{w_1} \lambda(|w_1| + |\Delta - w_1|)$$



- For almost correlated features:
  - often only w_1 or w_2 nonzero (the one with slightly better fit)
  - however, features highly correlated, if X_1 explains data so does X_2

Elastic net

- Add Tikhonov regularization to the Lasso

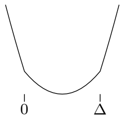
$$\text{minimize} \, \tfrac{1}{2}\|Xw - Y\|^2 + \lambda_1 \|w\|_1 + \tfrac{\lambda_2}{2}\|w\|_2^2$$

- This problem is called *elastic net* in statistics
- Can perform better with correlated features

Elastic net and correlated features

- Assume equal features $X_1 = X_2$ and that w^* solves the elastic net
- Let $\Delta := w_1^* + w_2^* > 0$ (wlog), then $w_1^* = w_2^* = \frac{\Delta}{2}$
  - Data fit cost still unchanged for $w_2 = \Delta - w_1$ with $w_1 \in [0, \Delta]$
  - Remaining (regularization) part is

$$\min_{w_1} \lambda_1(|w_1| + |\Delta - w_1|) + \lambda_2(w_1^2 + (\Delta - w_1)^2)$$



which is minimized in the middle at $w_1 = w_2 = \frac{\Delta}{2}$

- For highly correlated features, both (or none) probably selected

Group lasso

- Sometimes want groups of variables to be 0 or nonzero
- Introduce blocks $w = (w_1, \ldots, w_p)$ where $w_i \in \mathbb{R}^{n_i}$
- The group Lasso problem is

$$\text{minimize} \, \tfrac{1}{2}\|Xw - Y\|_2^2 + \lambda \sum_{i=1}^{p} \|w_i\|_2$$

(note $\|\cdot\|_2$-norm without square)

- With all $n_i = 1$, it reduces to the Lasso
- Promotes block sparsity, meaning full block $w_i \in \mathbb{R}^{n_i}$ would be 0

Outline

- Supervised learning – Overview
- Least squares – Basics
- Nonlinear features
- Generalization, overfitting, and regularization
- Cross validation
- Feature selection
- **Training problem properties**

Composite optimization

- Least squares problems are convex problems of the form

$$\underset{\theta}{\text{minimize}} \, f(X\theta) + g(\theta),$$

where

- $f = \frac{1}{2}\|\cdot - Y\|_2^2$ is data misfit term
- X is training data matrix (potentially extended with features)
- g is regularization term (1-norm, squared 2-norm, group lasso)

- Function properties
  - f is 1-strongly convex and 1-smooth and $f \circ X$ is $\|X\|_2^2$-smooth
  - g is convex and possibly nondifferentiable
- Gradient $\nabla(f \circ X)(\theta) = X^T(X\theta - Y)$

Logistic Regression

Pontus Giselsson

Outline

- **Classification**
- Logistic regression
- Nonlinear features
- Overfitting and regularization
- Multiclass logistic regression
- Training problem properties

Classification

- Let (x, y) represent object and label pairs
  - Object $x \in \mathcal{X} \subseteq \mathbb{R}^n$
  - Label $y \in \mathcal{Y} = \{1, \ldots, K\}$ that corresponds to K different classes
- Available: Labeled training data (training set) $\{(x_i, y_i)\}_{i=1}^N$

Objective: Find parameterized model (function) $m(x; \theta)$:

- that takes data (example, object) x as input
- and predicts corresponding label (class) $y \in \{1, \ldots, K\}$

How?:

- learn parameters θ by solving training problem with training data

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i; \theta), y_i)$$

with some loss function L

Binary classification

- Labels $y = 0$ or $y = 1$ (alternatively $y = -1$ or $y = 1$)
- Training problem

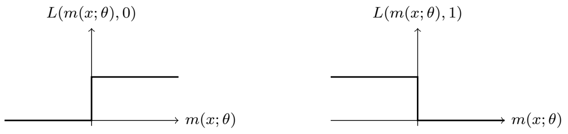
$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i; \theta), y_i)$$

- Design loss L to train model parameters θ such that:
  - $m(x_i; \theta) < 0$ for pairs (x_i, y_i) where $y_i = 0$
  - $m(x_i; \theta) > 0$ for pairs (x_i, y_i) where $y_i = 1$
- Predict class belonging for new data points x with trained θ^*:
  - $m(x; \theta^*) < 0$ predict class $y = 0$
  - $m(x; \theta^*) > 0$ predict class $y = 1$

  objective is that this prediction is accurate on unseen data

Binary classification – Cost functions

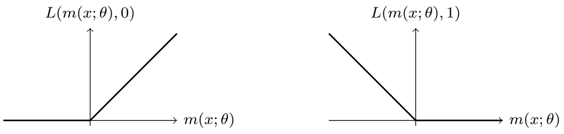
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x; \theta) \ll 0$ large for $m(x; \theta) \gg 0$
  - $y = 1$: Small cost for $m(x; \theta) \gg 0$ large for $m(x; \theta) \ll 0$



nonconvex (Neyman Pearson loss)

Binary classification – Cost functions

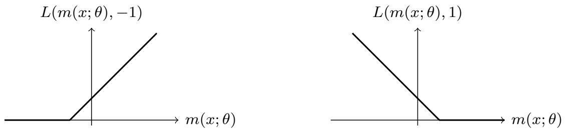
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x; \theta) \ll 0$ large for $m(x; \theta) \gg 0$
  - $y = 1$: Small cost for $m(x; \theta) \gg 0$ large for $m(x; \theta) \ll 0$



$L(u, y) = \max(0, u) - yu$

Binary classification – Cost functions

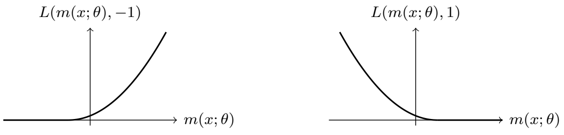
- Different cost functions L can be used:
  - $y = -1$: Small cost for $m(x; \theta) \ll 0$ large for $m(x; \theta) \gg 0$
  - $y = 1$: Small cost for $m(x; \theta) \gg 0$ large for $m(x; \theta) \ll 0$



$L(u, y) = \max(0, 1 - yu)$ (hinge loss used in SVM)

Binary classification – Cost functions

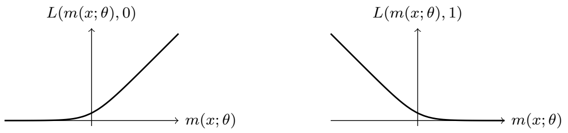
- Different cost functions L can be used:
  - $y = -1$: Small cost for $m(x; \theta) \ll 0$ large for $m(x; \theta) \gg 0$
  - $y = 1$: Small cost for $m(x; \theta) \gg 0$ large for $m(x; \theta) \ll 0$



$L(u, y) = \max(0, 1 - yu)^2$ (squared hinge loss)

Binary classification – Cost functions

- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



$L(u,y) = \log(1+e^u) - yu$ (logistic loss)

Outline

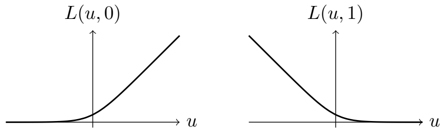
- Classification
- **Logistic regression**
- Nonlinear features
- Overfitting and regularization
- Multiclass logistic regression
- Training problem properties

Logistic regression

- Logistic regression uses:
  - affine parameterized model $m(x;\theta) = w^T x + b$ (where $\theta = (w,b)$)
  - loss function $L(u,y) = \log(1+e^u) - yu$ (if labels $y=0$, $y=1$)
- Training problem, find model parameters by solving:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) = \sum_{i=1}^{N} \left(\log(1+e^{x_i^T w+b}) - y_i(x_i^T w + b)\right)$$

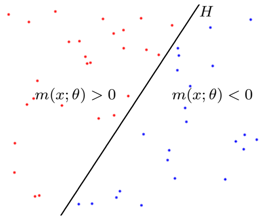
- Training problem convex in $\theta = (w,b)$ since:
  - model $m(x;\theta)$ is affine in θ
  - loss function $L(u,y)$ is convex in u



Prediction

- Use trained model m to predict label y for unseen data point x
- Since affine model $m(x;\theta) = w^T x + b$, prediction for x becomes:
  - If $w^T x + b < 0$, predict corresponding label $y = 0$
  - If $w^T x + b > 0$, predict corresponding label $y = 1$
  - If $w^T x + b = 0$, predict either $y = 0$ or $y = 1$
- A hyperplane (decision boundary) separates class predictions:

$$H := \{x : w^T x + b = 0\}$$

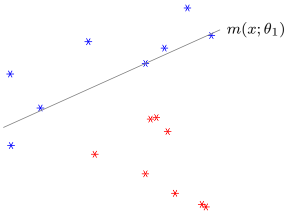


Training problem interpretation

- Every parameter choice $\theta = (w,b)$ gives hyperplane in data space:

$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to "best" separates classes
- Example – models with different parameters θ:

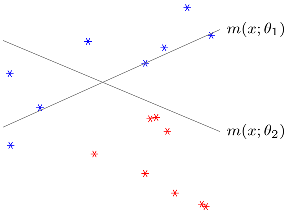


Training problem interpretation

- Every parameter choice $\theta = (w,b)$ gives hyperplane in data space:

$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to "best" separates classes
- Example – models with different parameters θ:

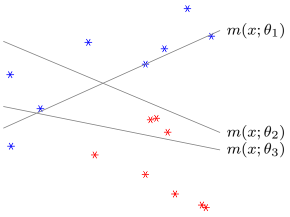


Training problem interpretation

- Every parameter choice $\theta = (w,b)$ gives hyperplane in data space:

$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to "best" separates classes
- Example – models with different parameters θ:

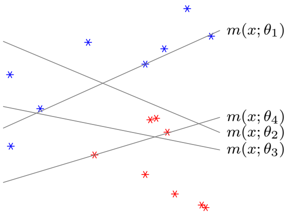


Training problem interpretation

- Every parameter choice $\theta = (w,b)$ gives hyperplane in data space:

$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to "best" separates classes
- Example – models with different parameters θ:

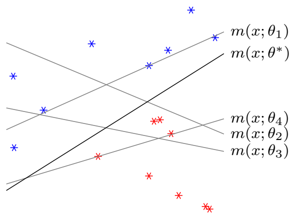


Training problem interpretation

- Every parameter choice $\theta = (w, b)$ gives hyperplane in data space:

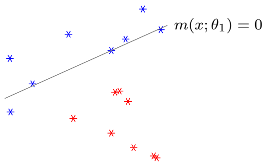
$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to "best" separates classes
- Example – models with different parameters θ:

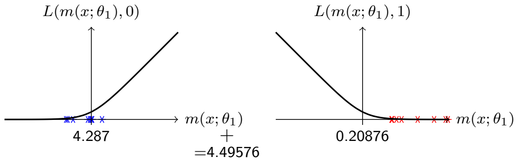


What is "best" separation?

- The "best" separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_1$:

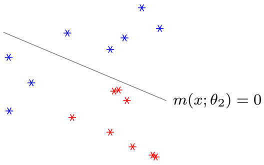


- Training loss:

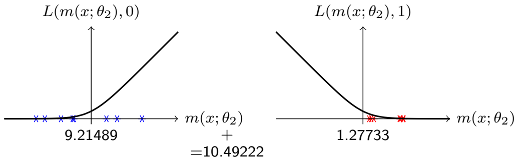


What is "best" separation?

- The "best" separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_2$:

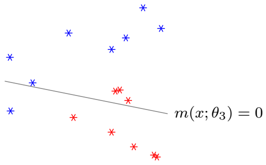


- Training loss:

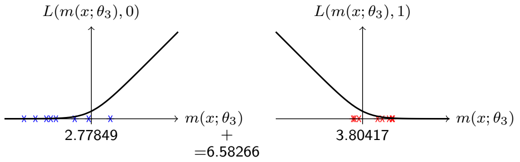


What is "best" separation?

- The "best" separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_3$:

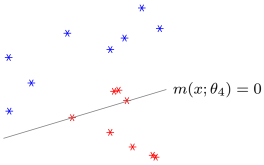


- Training loss:

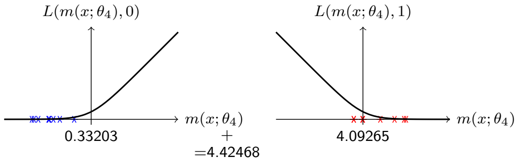


What is "best" separation?

- The "best" separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_4$:

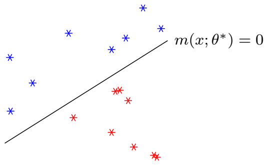


- Training loss:

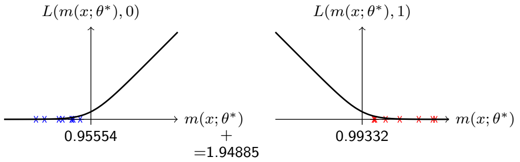


What is "best" separation?

- The "best" separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta^*$:

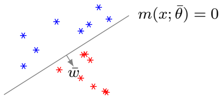


- Training loss:

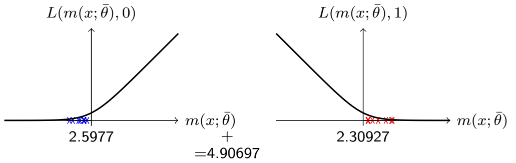


Fully separable data – Solution

- Let $\bar{\theta} = (\bar{w}, \bar{b})$ give model that separates data:

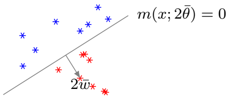


- Let $H_{\bar{\theta}} := \{x : m(x;\bar{\theta}) = \bar{w}^T x + \bar{b} = 0\}$ be hyperplane separates
- Training loss:

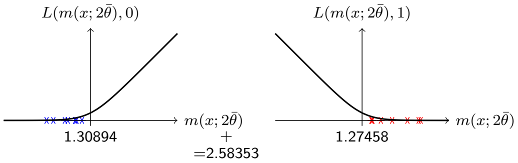


Fully separable data – Solution

- Also $2\bar{\theta} = (2\bar{w}, 2\bar{b})$ separates data:

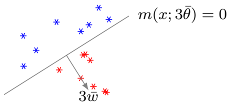


- Hyperplane $H_{2\bar{\theta}} := \{x : m(x;2\bar{\theta}) = 2(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss reduced since input $m(x;2\bar{\theta}) = 2m(x;\bar{\theta})$ further out:

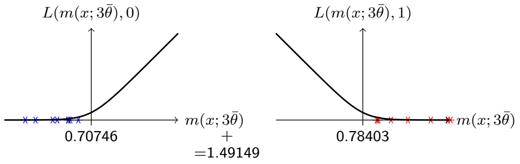


Fully separable data – Solution

- And $3\bar{\theta} = (3\bar{w}, 3\bar{b})$ also separates data:

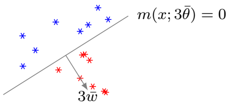


- Hyperplane $H_{3\bar{\theta}} := \{x : m(x; 3\bar{\theta}) = 3(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss further reduced since input $m(x; 3\bar{\theta}) = 3m(x; \bar{\theta})$:

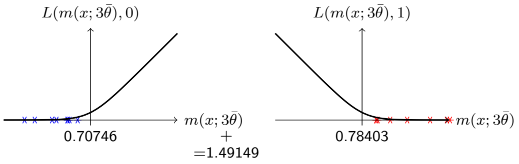


Fully separable data – Solution

- And $3\bar{\theta} = (3\bar{w}, 3\bar{b})$ also separates data:



- Hyperplane $H_{3\bar{\theta}} := \{x : m(x; 3\bar{\theta}) = 3(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss



- Let $\theta = t\bar{\theta}$ and $t \to \infty$, then loss $\to 0 \Rightarrow$ no optimal point

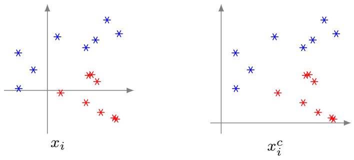
---

The bias term

- The model $m(x; \theta) = w^T x + b$ bias term is b
- Least squares: optimal b has simple formula
- No simple formula to remove bias term here!

Bias term gives shift invariance

- Assume all data points shifted $x_i^c := x_i + c$
- We want same hyperplane to separate data, but shifted



- Assume $\theta = (w, b)$ is optimal for $\{(x_i, y_i)\}_{i=1}^N$
- Then $\theta_c = (w, b_c)$ with $b_c = b - w^T c$ optimal for $\{(x_i^c, y_i)\}_{i=1}^N$
- Why? Model outputs the same for all x_i:
  - $m(x_i; \theta) = w^T x_i + b$
  - $m(x_i^c; \theta_c) = w^T x_i^c + b_c = w^T x_i + b + w^T(c - c) = w^T x_i + b$

Another derivation of logistic loss

- Assume model is instead $\sigma(w^T x + b)$, with $\sigma(u) = \frac{1}{1+e^{-u}}$
- *Binary cross entropy* applied to model with sigmoid output:

$$\begin{aligned} & -y\log(\sigma(u)) - (1-y)\log(1-\sigma(u)) \\ = & -y\log(\frac{1}{1+e^{-u}}) - (1-y)\log(1 - \frac{1}{1+e^{-u}}) \\ = & -y\log(\frac{e^u}{1+e^u}) - (1-y)\log(\frac{e^{-u}}{1+e^{-u}}) \\ = & -y(u - \log(1+e^u)) + (1-y)\log(1+e^u) \\ = & \log(1+e^u) - yu \ (= \text{logistic loss}) \end{aligned}$$

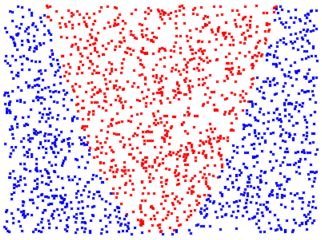
- Two equivalent formulations to arrive at same problem:
  - Real-valued model $m(x; \theta)$ and logistic loss $\log(1 + e^u) - yu$
  - $(0, 1)$-valued model $\sigma(m(x; \theta))$ and binary cross entropy
- Prefer previous formulation
  - easier to see how deviations penalized
  - easier to conclude convexity of training problem

Outline

- Classification
- Logistic regression
- **Nonlinear features**
- Overfitting and regularization
- Multiclass logistic regression
- Training problem properties

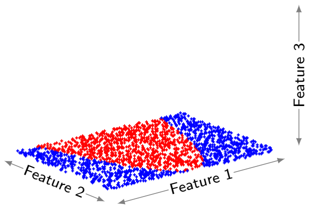
Logistic regression – Nonlinear example

- Logistic regression tries to affinely separate data
- Can nonlinear boundary be approximated by logistic regression?
- Introduce features (perform lifting)



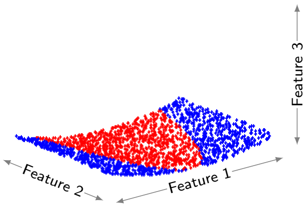
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



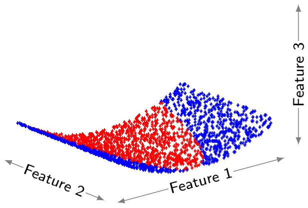
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



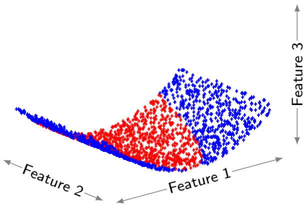
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



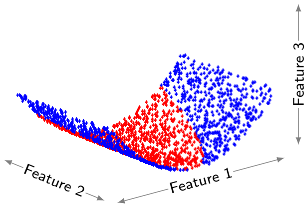
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



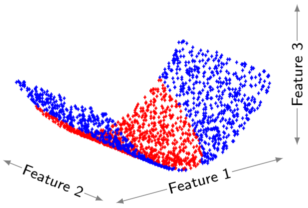
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



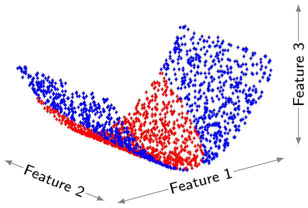
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



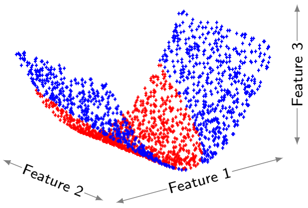
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



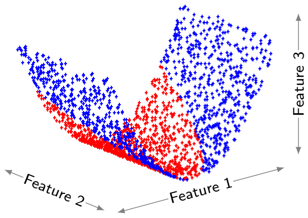
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



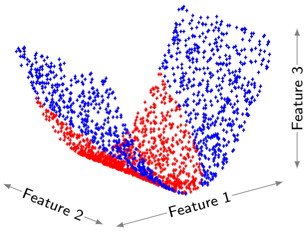
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



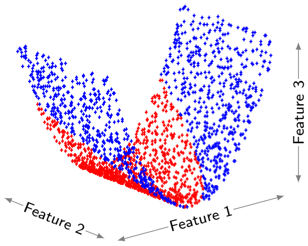
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



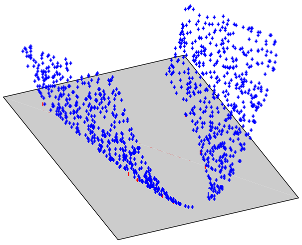
Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



Logistic regression – Example

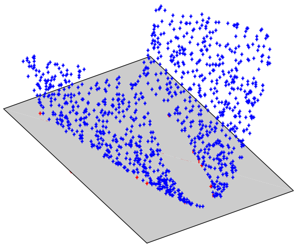
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

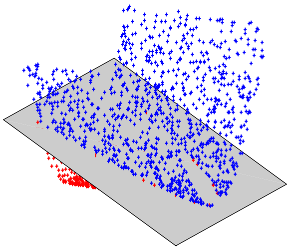
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

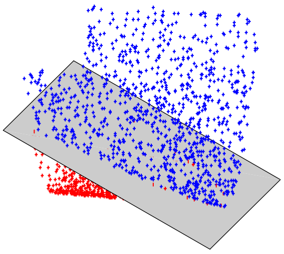
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

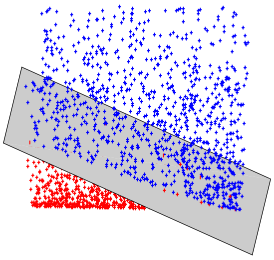
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

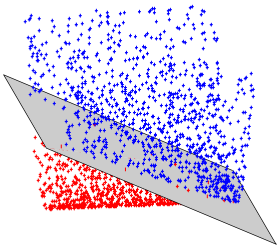
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

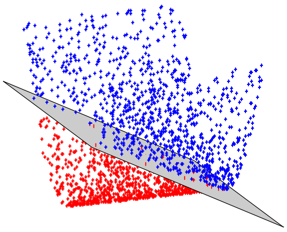
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

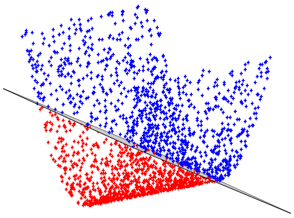
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

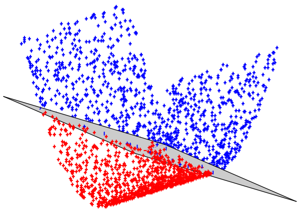
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

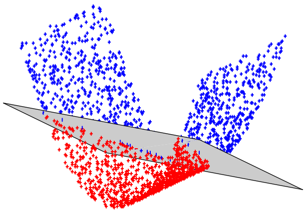
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

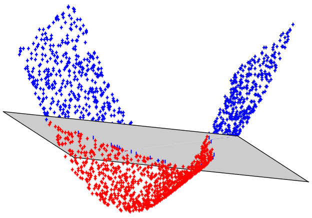
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Nonlinear models – Features

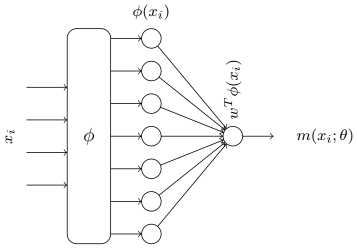
- Create feature map $\phi : \mathbb{R}^n \to \mathbb{R}^p$ of training data
- Data points $x_i \in \mathbb{R}^n$ replaced by featured data points $\phi(x_i) \in \mathbb{R}^p$
- New model: $m(x;\theta) = w^T\phi(x) + b$, still linear in parameters
- Feature can include original data x
- We can add feature 1 and remove bias term b
- Logistic regression training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \left(\log(1 + e^{\phi(x_i)^T w + b}) - y_i(\phi(x_i)^T w + b) \right)$$

same as before, but with features as inputs

Graphical model representation

- A graphical view of model $m(x;\theta) = w^T\phi(x)$:



- The input x_i is transformed by *fixed* nonlinear features ϕ
- Feature-transformed input is multiplied by model parameters θ
- Model output is then fed into cost $L(m(x_i;\theta), y)$
- Problem convex since L convex and model affine in θ

Polynomial features

- Polynomial feature map for $\mathbb{R}^n$ with $n = 2$ and degree $d = 3$

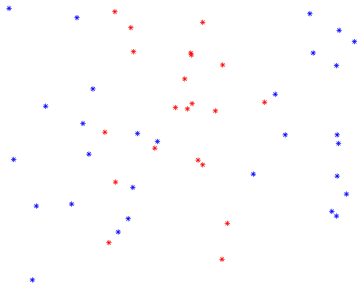
$$\phi(x) = (x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$$

(note that original data is also there)

- New model: $m(x;\theta) = w^T\phi(x) + b$, still linear in parameters
- Number of features $p + 1 = \binom{n+d}{d} = \frac{(n+d)!}{d!n!}$ grows fast!
- Training problem has $p + 1$ instead of $n + 1$ decision variables

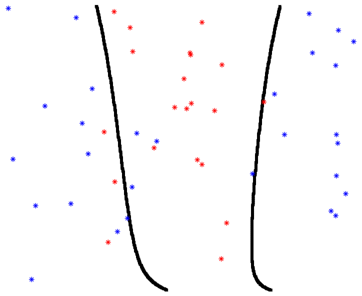
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree:



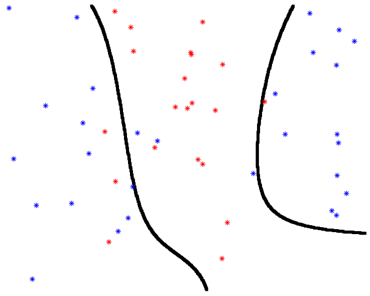
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 2



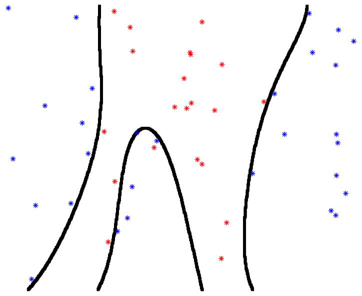
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 3



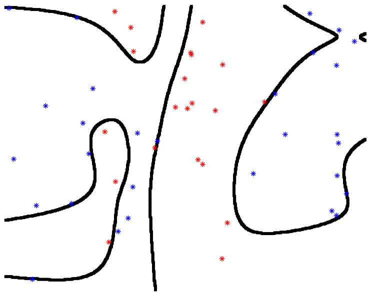
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 4



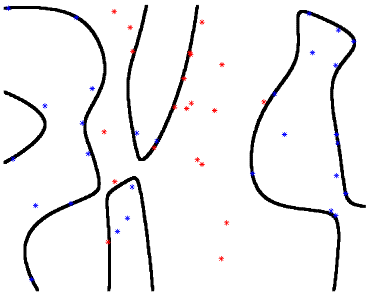
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 5



Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 6

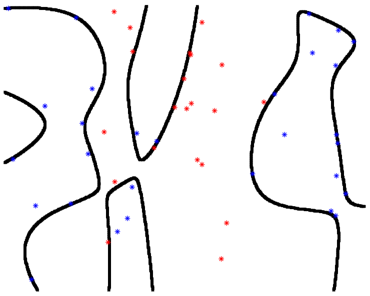


Outline

- Classification
- Logistic regression
- Nonlinear features
- **Overfitting and regularization**
- Multiclass logistic regression
- Training problem properties

Overfitting

- Models with higher order polynomials overfit
- Logistic regression (no regularization) polynomial features of degree 6



- Tikhonov regularization can reduce overfitting

Tikhonov regularization

Regularized problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \Big(\log(1+e^{x_i^T w+b}) - y_i(x_i^T w+b)\Big) + \lambda\|w\|_2^2$$

Regularization:

- Regularize only w and not the bias term b
- Why? Model looses shift invariance if also b regularized

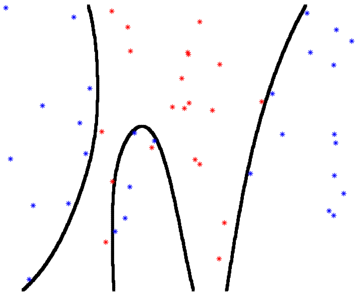
Problem properties:

- Problem is strongly convex in $w \Rightarrow$ optimal w exists and is unique
- Optimal b is bounded if examples from both classes exist

Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

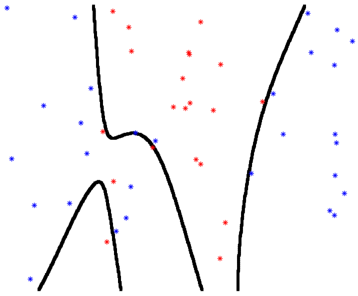
λ	J	# mislabels
0.00001	4.60	1



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

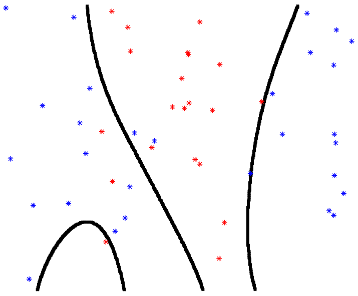
λ	J	# mislabels
0.00006	6.83	5



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

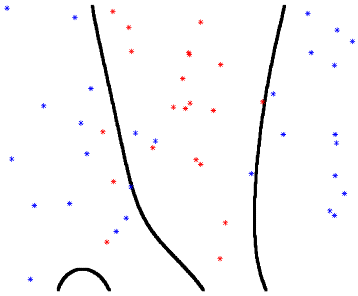
λ	J	# mislabels
0.00036	9.94	5



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

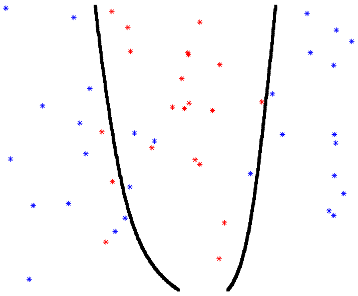
λ	J	# mislabels
0.0021	12.1	6



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

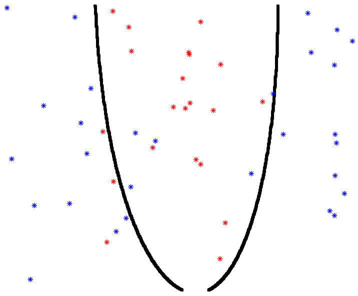
λ	J	# mislabels
0.013	13.6	7



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

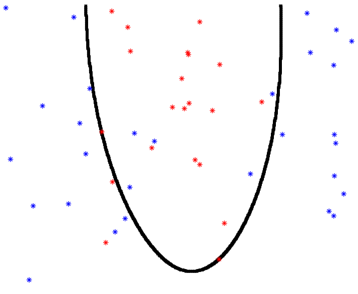
λ	J	# mislabels
0.077	15.4	8



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

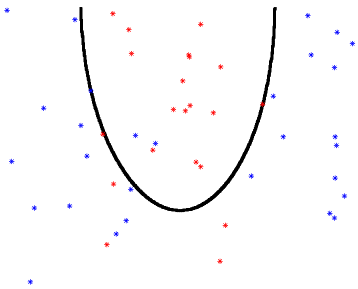
λ	J	# mislabels
0.46	19.2	7



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

λ	J	# mislabels
2.78	25.2	8



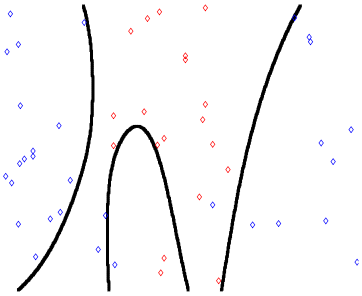
Generalization

- Interested in models that *generalize* well to unseen data
- Assess generalization using holdout or k-fold cross validation

Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

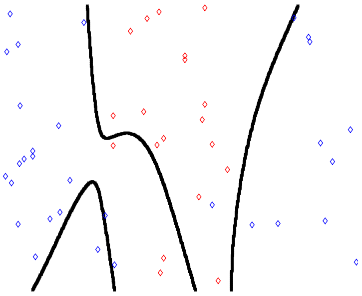
λ	J	# mislabels
0.00001	4.60/38.5	1/7



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

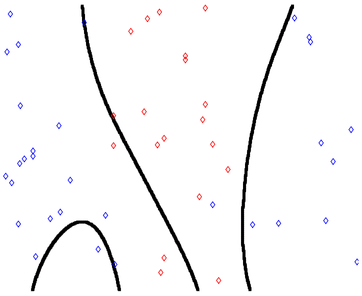
λ	J	# mislabels
0.00006	6.83/25.7	5/7



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

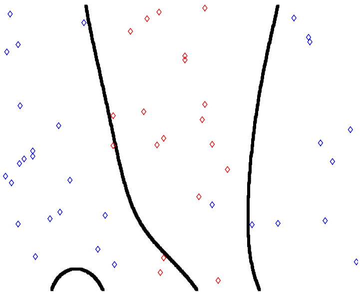
λ	J	# mislabels
0.00036	9.94/13.4	5/8



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

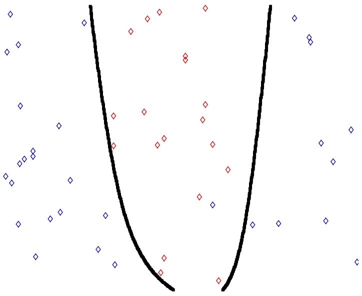
λ	J	# mislabels
0.0021	12.1/8.70	6/5



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

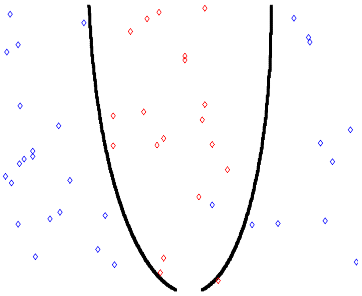
λ	J	# mislabels
0.013	13.6/8.12	7/2



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

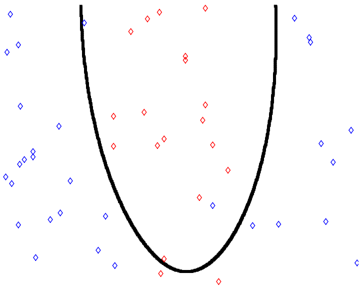
λ	J	# mislabels
0.077	15.4/10.2	8/3



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

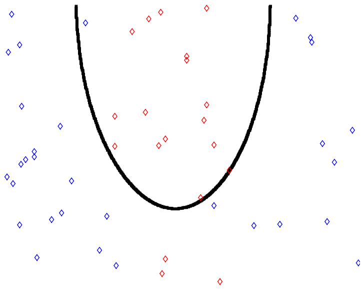
λ	J	# mislabels
0.46	19.2/15.2	7/4



Example – Validation data

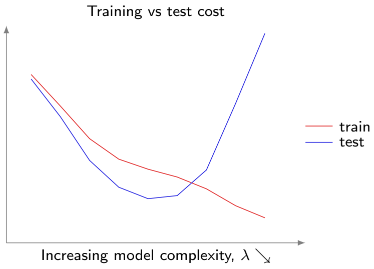
- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

λ	J	# mislabels
2.78	25.2/23.2	8/4



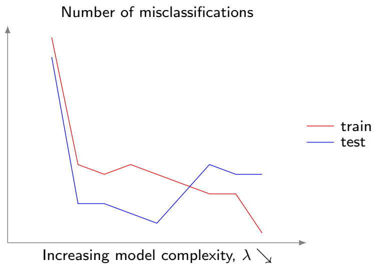
Test vs training error – Cost

- Decreasing λ gives higher complexity model
- Overfitting to the right, underfitting to the left
- Select lowest complexity model that gives good generalization



Test vs training error – Classification accuracy

- Decreasing λ gives higher complexity model
- Overfitting to the right, underfitting to the left
- Cost often better measure of over/underfitting



Outline

- Classification
- Logistic regression
- Nonlinear features
- Overfitting and regularization
- **Multiclass logistic regression**
- Training problem properties

What is multiclass classification?

- We have previously seen binary classification
  - Two classes (cats and dogs)
  - Each sample belongs to one class (has one label)
- Multiclass classification
  - K classes with $K \geq 3$ (cats, dogs, rabbits, horses)
  - Each sample belongs to one class (has one label)
  - (Not to confuse with multilabel classification with ≥ 2 labels)

Multiclass classification from binary classification

- 1-vs-1: Train binary classifiers between all classes
  - Example:
    - cat-vs-dog,
    - cat-vs-rabbit
    - cat-vs-horse
    - dog-vs-rabbit
    - dog-vs-horse
    - rabbit-vs-horse
  - Prediction: Pick, e.g., the one that wins the most classifications
  - Number of classifiers: $\frac{K(K-1)}{2}$
- 1-vs-all: Train each class against the rest
  - Example
    - cat-vs-(dog,rabbit,horse)
    - dog-vs-(cat,rabbit,horse)
    - rabbit-vs-(cat,dog,horse)
    - horse-vs-(cat,dog,rabbit)
  - Prediction: Pick, e.g., the one that wins with highest margin
  - Number of classifiers: K
  - Always skewed number of samples in the two classes

Multiclass logistic regression

- K classes in $\{1, \ldots, K\}$ and data/labels $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- Labels: $y \in \mathcal{Y} = \{e_1, \ldots, e_K\}$ where $\{e_j\}$ coordinate basis
  - Example, $K = 5$ class 2: $y = e_2 = [0, 1, 0, 0, 0]^T$
- Use one model per class $m_j(x; \theta_j)$ for $j \in \{1, \ldots, K\}$
- Objective: Find $\theta = (\theta_1, \ldots, \theta_K)$ such that for all models j:
  - $m_j(x; \theta_j) \gg 0$, if label $y = e_j$ and $m_j(x; \theta_j) \ll 0$ if $y \neq e_j$
- Training problem loss function:

$$L(u, y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - u^T y$$

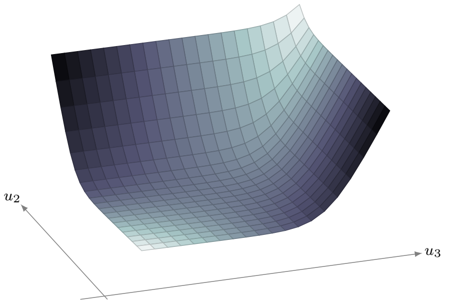
where label y is a "one-hot" basis vector, is convex in u

Multiclass logistic loss function – Example

- Multiclass logistic loss for $K = 3$, $u_1 = 1$, $y = e_1$

$$L((1, u_2, u_3), 1) = \log(e^1 + e^{u_2} + e^{u_3}) - 1$$

- Model outputs $u_2 \ll 0$, $u_3 \ll 0$ give smaller cost for label $y = e_1$

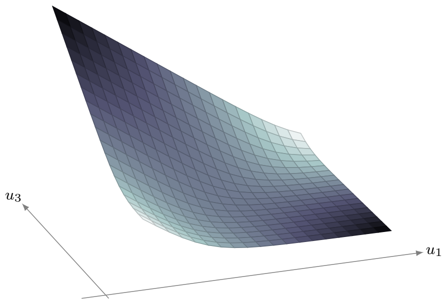


Multiclass logistic loss function – Example

- Multiclass logistic loss for $K = 3$, $u_2 = -1$, $y = e_1$

$$L((u_1, -1, u_3), 1) = \log(e^{u_1} + e^{-1} + e^{u_3}) - u_1$$

- Model outputs $u_1 \gg 0$ and $u_3 \ll 0$ give smaller cost for $y = e_1$



Multiclass logistic regression – Training problem

- Affine data model $m(x; \theta) = w^T x + b$ with

$$w = [w_1, \ldots, w_K] \in \mathbb{R}^{n \times K}, \qquad b = [b_1, \ldots, b_K]^T \in \mathbb{R}^K$$

- One data model per class

$$m(x; \theta) = \begin{bmatrix} m_1(x; \theta_1) \\ \vdots \\ m_K(x; \theta_K) \end{bmatrix} = \begin{bmatrix} w_1^T x + b_1 \\ \vdots \\ w_K^T x + b_K \end{bmatrix}$$

- Training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \log\left(\sum_{j=1}^{K} e^{w_j^T x_i + b_j}\right) - y_i^T (w^T x_i + b)$$

  where y_i is "one-hot" encoding of label
- Problem is convex since affine model is used
- (Alt.: model $\sigma(w^T x + b)$ with σ softmax and cross entropy loss)

Multiclass logistic regression – Prediction

- Assume model is trained and want to predict label for new data x
- Predict class with parameter θ for x according to:

$$\underset{j \in \{1, \ldots, K\}}{\text{argmax}} \; m_j(x; \theta)$$

  i.e., class with largest model value (since trained to achieve this)

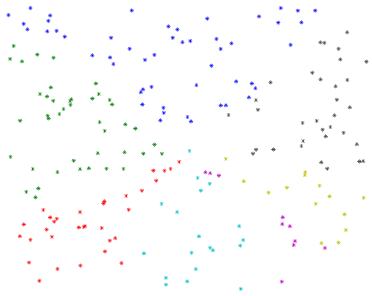
Special case – Binary logistic regression

- Consider two-class version and let
  - $\Delta u = u_1 - u_2$, $\Delta w = w_1 - w_2$, and $\Delta b = b_1 - b_2$
  - $\Delta u = m_{\text{bin}}(x; \theta) = m_1(x; \theta_1) - m_2(x; \theta_2) = \Delta w^T x + \Delta b$
  - $y_{\text{bin}} = 1$ if $y = (1, 0)$ and $y_{\text{bin}} = 0$ if $y = (0, 1)$
- Loss L is equivalent to binary, but with different variables:

$$\begin{aligned} L(u, y) &= \log(e^{u_1} + e^{u_2}) - y_1 u_1 - y_2 u_2 \\ &= \log\left(1 + e^{u_1 - u_2}\right) + \log(e^{u_2}) - y_1 u_1 - y_2 u_2 \\ &= \log\left(1 + e^{\Delta u}\right) - y_1 u_1 - (y_2 - 1) u_2 \\ &= \log\left(1 + e^{\Delta u}\right) - y_{\text{bin}} \Delta u \end{aligned}$$

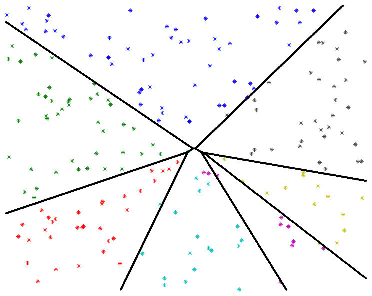
Example – Linearly separable data

- Problem with 7 classes



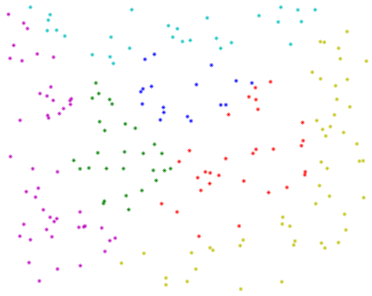
Example – Linearly separable data

- Problem with 7 classes and affine multiclass model



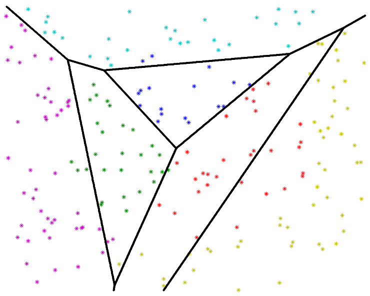
Example – Quadratically separable data

- Same data, new labels in 6 classes



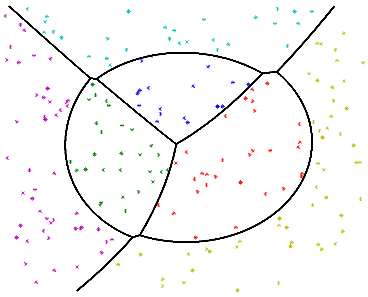
Example – Quadratically separable data

- Same data, new labels in 6 classes, affine model



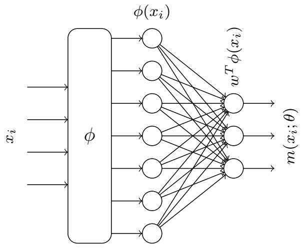
Example – Quadratically separable data

- Same data, new labels in 6 classes, quadratic model



Features

- Used quadratic features in last example
- Same procedure as before:
  - replace data vector x_i with feature vector $\phi(x_i)$
  - run classification method with feature vectors as inputs



Outline

- Classification
- Logistic regression
- Nonlinear features
- Overfitting and regularization
- Multiclass logistic regression
- **Training problem properties**

Composite optimization – Binary logistic regression

Regularized (with g) logistic regression training problem (no features)

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \left(\log\left(1+e^{w^Tx_i+b}\right) - y_i(w^Tx_i+b)\right) + g(\theta)$$

can be written on the form

$$\underset{\theta}{\text{minimize}}\, f(L\theta) + g(\theta),$$

where

- $f(u) = \sum_{i=1}^{N}(\log(1+e^{u_i}) - y_iu_i)$ is data misfit term
- $L = [X, \mathbf{1}]$ where training data matrix X and $\mathbf{1}$ satisfy

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix} \qquad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

- g is regularization term

Gradient and function properties

- Gradient of $h_i(u_i) = \log(1+e^{u_i}) - y_iu_i$ is:

$$\nabla h_i(u_i) = \frac{e^{u_i}}{1+e^{u_i}} - y_i = \frac{1}{1+e^{-u_i}} - y_i =: \sigma(u_i) - y_i$$

  where $\sigma(u_i) = (1+e^{-u_i})^{-1}$ is called a *sigmoid* function
- Gradient of $(f \circ L)(\theta)$ satisfies:

$$\begin{aligned} \nabla(f \circ L)(\theta) = \nabla \sum_{i=1}^{N} h_i(L_i\theta) &= \sum_{i=1}^{N} L_i^T \nabla h_i(L_i\theta) \\ &= \sum_{i=1}^{N} \begin{bmatrix} x_i \\ 1 \end{bmatrix} (\sigma(x_i^Tw+b) - y_i) \\ &= \begin{bmatrix} X^T \\ \mathbf{1}^T \end{bmatrix} (\sigma(Xw+b\mathbf{1}) - Y) \end{aligned}$$

  where last $\sigma : \mathbb{R}^N \to \mathbb{R}^N$ applies $\frac{1}{1+e^{-u_i}}$ to all $[Xw+b\mathbf{1}]_i$
- Function and sigmoid properties:
  - sigmoid σ is 0.25-Lipschitz continuous:
  - f is convex and 0.25-smooth and $f \circ L$ is $0.25\|L\|_2^2$-smooth

Support Vector Machines

Pontus Giselsson

Outline

- **Classification**
- Support vector machines
- Nonlinear features
- Overfitting and regularization
- Dual problem
- Kernel SVM
- Training problem properties

Binary classification

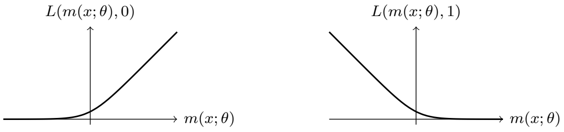
- Labels $y = 0$ or $y = 1$ (alternatively $y = -1$ or $y = 1$)
- Training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

- Design loss L to train model parameters θ such that:
  - $m(x_i;\theta) < 0$ for pairs (x_i, y_i) where $y_i = 0$
  - $m(x_i;\theta) > 0$ for pairs (x_i, y_i) where $y_i = 1$
- Predict class belonging for new data points x with trained $\bar{\theta}$:
  - $m(x;\bar{\theta}) < 0$ predict class $y = 0$
  - $m(x;\bar{\theta}) > 0$ predict class $y = 1$

Binary classification – Cost functions

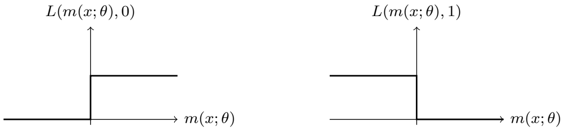
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



$L(u,y) = \log(1 + e^u) - yu$ (logistic loss)

Binary classification – Cost functions

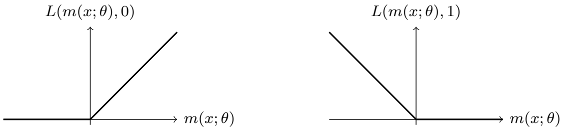
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



nonconvex (Neyman Pearson loss)

Binary classification – Cost functions

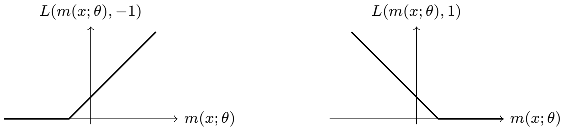
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



$L(u,y) = \max(0,u) - yu$

Binary classification – Cost functions

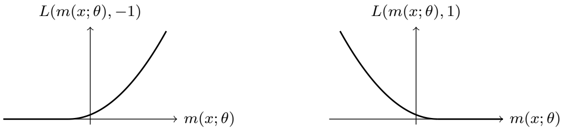
- Different cost functions L can be used:
  - $y = -1$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



$L(u,y) = \max(0, 1 - yu)$ (hinge loss used in SVM)

Binary classification – Cost functions

- Different cost functions L can be used:
  - $y = -1$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



$L(u,y) = \max(0, 1 - yu)^2$ (squared hinge loss)

Outline

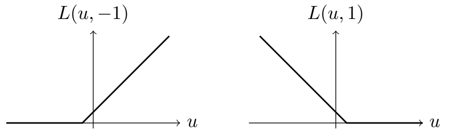
- Classification
- **Support vector machines**
- Nonlinear features
- Overfitting and regularization
- Dual problem
- Kernel SVM
- Training problem properties

Support vector machine

- SVM uses:
  - affine parameterized model $m(x;\theta) = w^T x + b$ (where $\theta = (w, b)$)
  - loss function $L(u, y) = \max(0, 1 - yu)$ (if labels $y = -1$, $y = 1$)
- Training problem, find model parameters by solving:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) = \sum_{i=1}^{N} \max(0, 1 - y_i(w^T x_i + b))$$

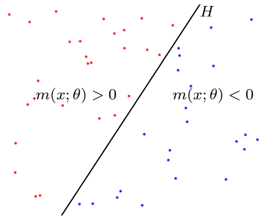
- Training problem convex in $\theta = (w, b)$ since:
  - model $m(x;\theta)$ is affine in θ
  - loss function $L(u, y)$ is convex in u



Prediction

- Use trained model m to predict label y for unseen data point x
- Since affine model $m(x;\theta) = w^T x + b$, prediction for x becomes:
  - If $w^T x + b < 0$, predict corresponding label $y = -1$
  - If $w^T x + b > 0$, predict corresponding label $y = 1$
  - If $w^T x + b = 0$, predict either $y = -1$ or $y = 1$
- A hyperplane (decision boundary) separates class predictions:

$$H := \{x : w^T x + b = 0\}$$

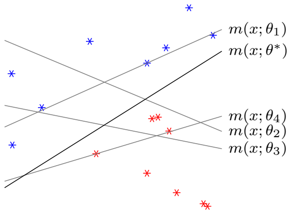


Training problem interpretation

- Every parameter choice $\theta = (w, b)$ gives hyperplane in data space:

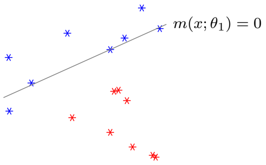
$$H := \{x : w^T x + b = 0\} = \{x : m(x;\theta) = 0\}$$

- Training problem searches hyperplane to “best” separates classes
- Example – models with different parameters θ:

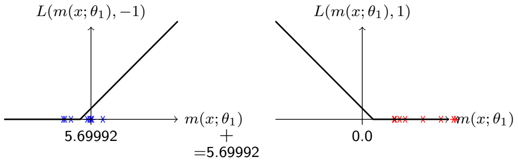


What is “best” separation?

- The “best” separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_1$:

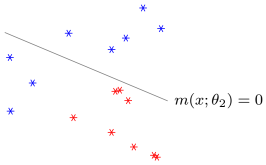


- Training loss:

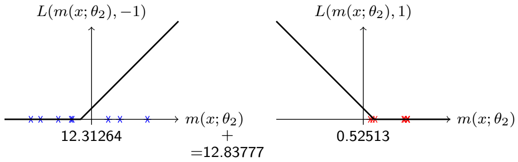


What is “best” separation?

- The “best” separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_2$:

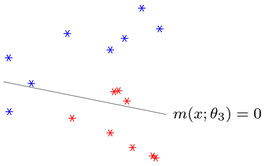


- Training loss:

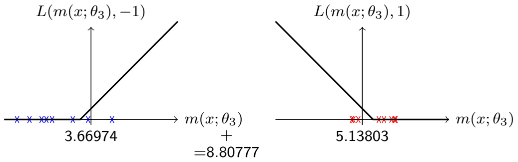


What is “best” separation?

- The “best” separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_3$:

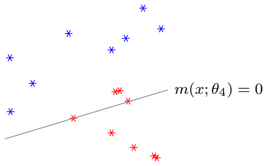


- Training loss:

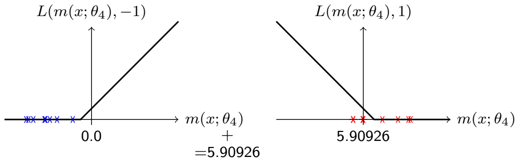


What is “best” separation?

- The “best” separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta_4$:

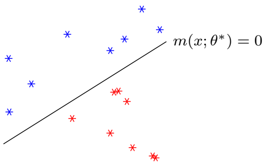


- Training loss:

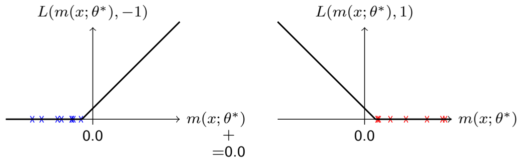


What is “best” separation?

- The “best” separation is the one that minimizes the loss function
- Hyperplane for model $m(\cdot;\theta)$ with parameter $\theta = \theta^*$:

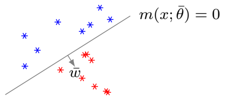


- Training loss:

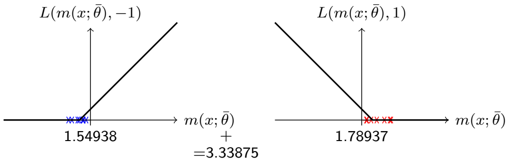


Fully separable data – Solution

- Let $\bar{\theta} = (\bar{w}, \bar{b})$ give model that separates data:

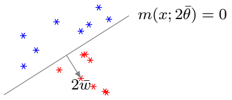


- Let $H_{\bar{\theta}} := \{x : m(x;\bar{\theta}) = \bar{w}^T x + \bar{b} = 0\}$ be hyperplane separates
- Training loss:

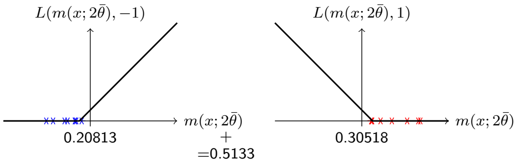


Fully separable data – Solution

- Also $2\bar{\theta} = (2\bar{w}, 2\bar{b})$ separates data:

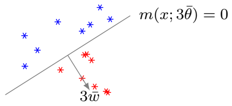


- Hyperplane $H_{2\bar{\theta}} := \{x : m(x;2\bar{\theta}) = 2(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss reduced since input $m(x;2\bar{\theta}) = 2m(x;\bar{\theta})$ further out:

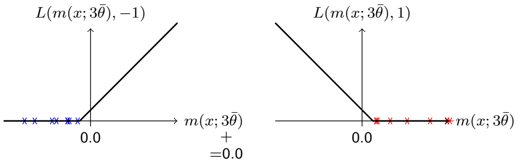


Fully separable data – Solution

- And $3\bar{\theta} = (3\bar{w}, 3\bar{b})$ also separates data:

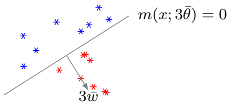


- Hyperplane $H_{3\bar{\theta}} := \{x : m(x;3\bar{\theta}) = 3(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss further reduced since input $m(x;3\bar{\theta}) = 3m(x;\bar{\theta})$:

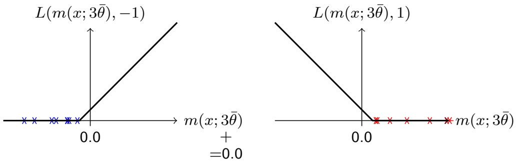


Fully separable data – Solution

- And $3\bar{\theta} = (3\bar{w}, 3\bar{b})$ also separates data:



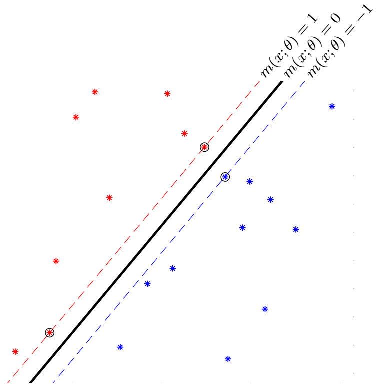
- Hyperplane $H_{3\bar{\theta}} := \{x : m(x;3\bar{\theta}) = 3(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss



- As soon as $|m(x_i;\theta)| \geq 1$ (with correct sign) for all x_i, cost is 0

Margin classification and support vectors

- Support vector machine classifiers for separable data
- Classes separated with margin, ○ marks *support vectors*

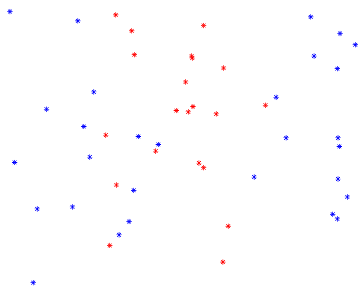


Outline

- Classification
- Support vector machines
- **Nonlinear features**
- Overfitting and regularization
- Dual problem
- Kernel SVM
- Training problem properties

Nonlinear example

- Can classify nonlinearly separable data using lifting



Adding features

- Create feature map $\phi : \mathbb{R}^n \to \mathbb{R}^p$ of training data
- Data points $x_i \in \mathbb{R}^n$ replaced by featured data points $\phi(x_i) \in \mathbb{R}^p$
- Example: Polynomial feature map with $n = 2$ and degree $d = 3$

$$\phi(x) = (x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$$

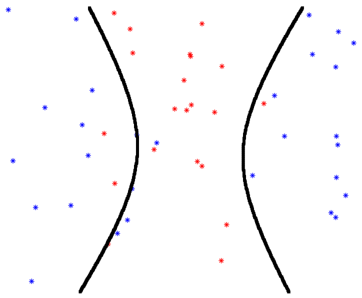
- Number of features $p + 1 = \binom{n+d}{d} = \frac{(n+d)!}{d!n!}$ grows fast!
- SVM training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \max(0, 1 - y_i(w^T\phi(x_i) + b))$$

still convex since features fixed

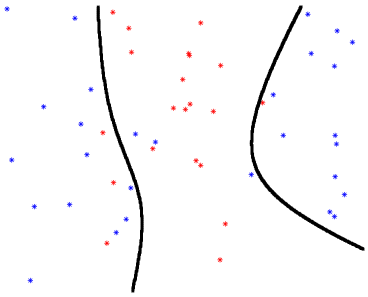
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 2



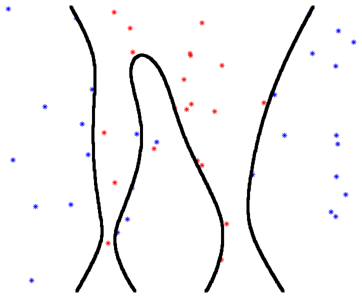
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 3



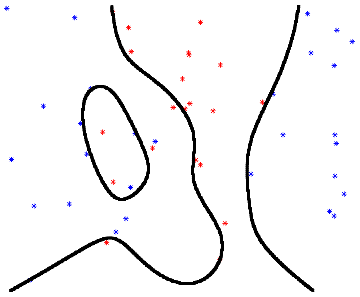
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 4



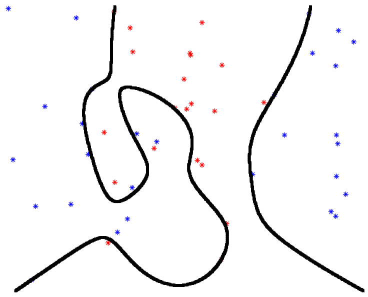
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 5



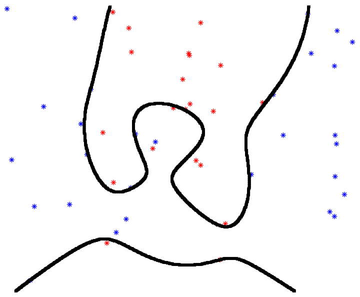
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 6



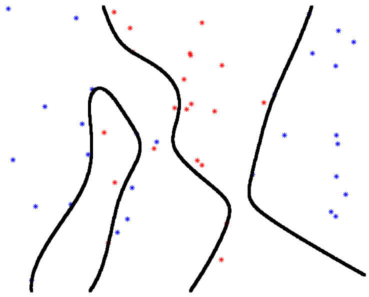
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 7



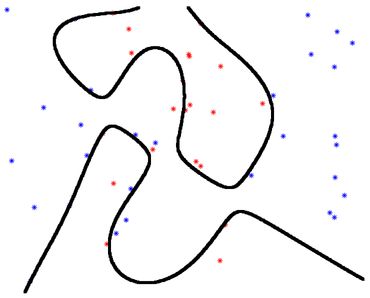
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 8



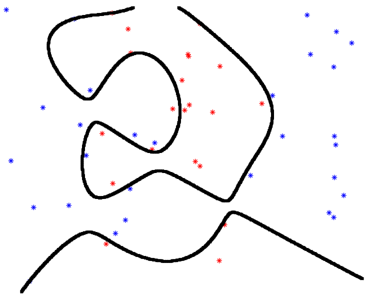
Nonlinear example – Polynomial features

- SVM and polynomial features of degree 9



Nonlinear example – Polynomial features

- SVM and polynomial features of degree 10



Outline

- Classification
- Support vector machines
- Nonlinear features
- **Overfitting and regularization**
- Dual problem
- Kernel SVM
- Training problem properties

Overfitting and regularization

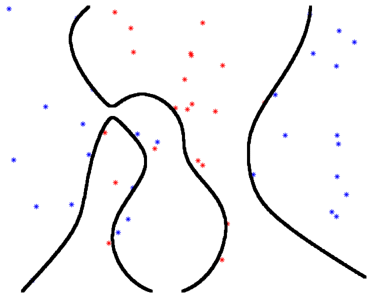
- SVM is prone to overfitting if model too expressive
- Regularization using $\|\cdot\|_1$ (for sparsity) or $\|\cdot\|_2^2$
- Tikhonov regularization with $\|\cdot\|_2^2$ especially important for SVM
- Regularize only linear terms w, not bias b
- Training problem with Tikhonov regularization of w

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \max(0, 1 - y_i(w^T\phi(x_i) + b)) + \tfrac{\lambda}{2}\|w\|_2^2$$

(note that features are used $\phi(x_i)$)

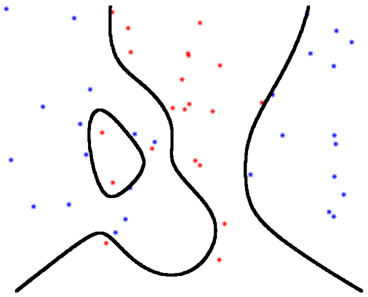
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.00001$



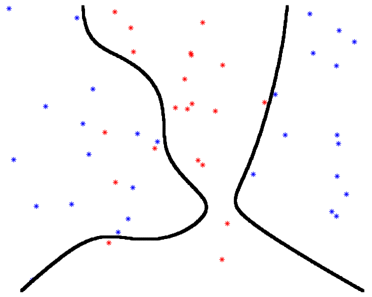
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.00006$



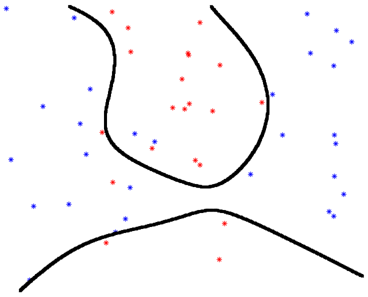
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.00036$



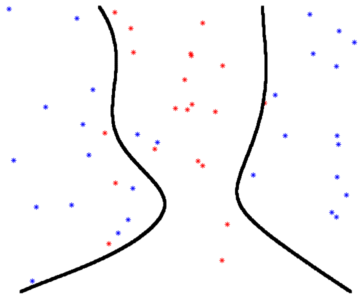
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.0021$



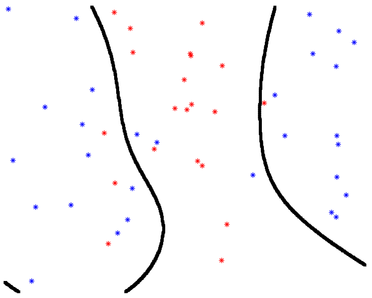
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.013$



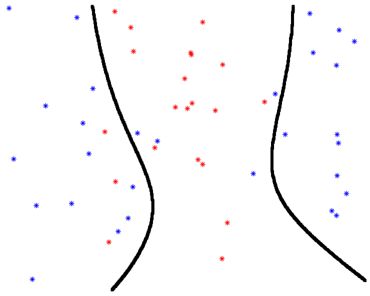
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.077$



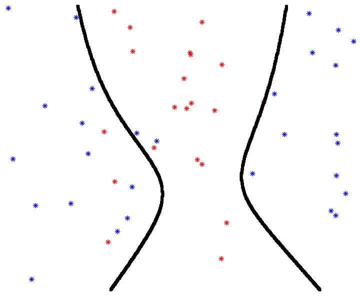
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 0.46$



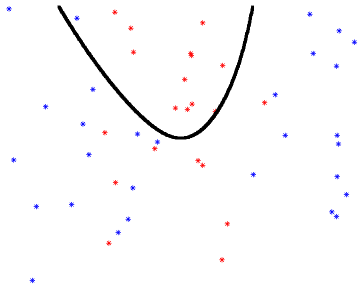
Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 2.78$



Nonlinear example revisited

- Regularized SVM and polynomial features of degree 6
- Regularization parameter: $\lambda = 16.7$



- λ and polynomial degree chosen using cross validation/holdout

Outline

- Classification
- Support vector machines
- Nonlinear features
- Overfitting and regularization
- **Dual problem**
- Kernel SVM
- Training problem properties

SVM problem reformulation

- Consider Tikhonov regularized SVM:

$$\underset{w,b}{\text{minimize}} \sum_{i=1}^{N} \max(0, 1 - y_i(w^T\phi(x_i) + b)) + \tfrac{\lambda}{2}\|w\|_2^2$$

- Derive dual from reformulation of SVM:

$$\underset{w,b}{\text{minimize}}\, \mathbf{1}^T \max(\mathbf{0}, \mathbf{1} - (X_{\phi,Y}w + Yb)) + \tfrac{\lambda}{2}\|w\|_2^2$$

where max is vector valued and

$$X_{\phi,Y} = \begin{bmatrix} y_1\phi(x_1)^T \\ \vdots \\ y_N\phi(x_N)^T \end{bmatrix}, \qquad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}$$

Dual problem

- Let $L = [X_{\phi,Y}, Y]$ and write problem as

$$\underset{w,b}{\text{minimize}}\, \underbrace{\mathbf{1}^T \max(\mathbf{0}, \mathbf{1} - (X_{\phi,Y}w + Yb))}_{f(L(w,b))} + \underbrace{\tfrac{\lambda}{2}\|w\|_2^2}_{g(w,b)}$$

  where
  - $f(\psi) = \sum_{i=1}^{N} f_i(\psi_i)$ and $f_i(\psi_i) = \max(0, 1 - \psi_i)$ (hinge loss)
  - $g(w,b) = \frac{\lambda}{2}\|w\|_2^2$, i.e., does not depend on b
- Dual problem

$$\underset{\nu}{\text{minimize}}\, f^*(\nu) + g^*(-L^T\nu)$$

Conjugate of g

- Conjugate of $g(w,b) = \frac{\lambda}{2}\|w\|_2^2 =: g_1(w) + g_2(b)$ is

$$g^*(\mu_w, \mu_b) = g_1^*(\mu_w) + g_2^*(\mu_b) = \tfrac{1}{2\lambda}\|\mu_w\|_2^2 + \iota_{\{0\}}(\mu_b)$$

- Evaluated at $-L^T\nu = -[X_{\phi,Y}, Y]^T\nu$:

$$\begin{aligned} g^*(-L^T\nu) &= g^*\left(-\begin{bmatrix} X_{\phi,Y}^T \\ Y^T \end{bmatrix}\nu\right) = \tfrac{1}{2\lambda}\| - X_{\phi,Y}^T\nu\|_2^2 + \iota_{\{0\}}(-Y^T\nu) \\ &= \tfrac{1}{2\lambda}\nu^T X_{\phi,Y}X_{\phi,Y}^T\nu + \iota_{\{0\}}(Y^T\nu) \end{aligned}$$

Conjugate of f

- Conjugate of $f_i(\psi_i) = \max(0, 1 - \psi_i)$ (hinge-loss):

$$f_i^*(\nu_i) = \begin{cases} \nu_i & \text{if } -1 \leq \nu_i \leq 0 \\ \infty & \text{else} \end{cases}$$

- Conjugate of $f(\psi) = \sum_{i=1}^N f_i(\psi_i)$ is sum of individual conjugates:

$$f^*(\nu) = \sum_{i=1}^N f_i^*(\nu_i) = \mathbf{1}^T\nu + \iota_{[-\mathbf{1},\mathbf{0}]}(\nu)$$

SVM dual

- The SVM dual is

$$\underset{\nu}{\text{minimize}}\, f^*(\nu) + g^*(-L^T\nu)$$

- Inserting the above computed conjugates gives dual problem

$$\begin{array}{ll} \underset{\nu}{\text{minimize}} & \sum_{i=1}^N \nu_i + \frac{1}{2\lambda}\nu^T X_{\phi,Y}X_{\phi,Y}^T\nu \\ \text{subject to} & -\mathbf{1} \leq \nu \leq \mathbf{0} \\ & Y^T\nu = 0 \end{array}$$

- Since $Y \in \mathbb{R}^N$, $Y^T\nu = 0$ is a hyperplane constraint
- If no bias term b; dual same but without hyperplane constraint

Primal solution recovery

- Meaningless to solve dual if we cannot recover primal
- Necessary and sufficient primal-dual optimality conditions

$$0 \in \begin{cases} \partial f^*(\nu) - L(w,b) \\ \partial g^*(-L^T\nu) - (w,b) \end{cases}$$

- From dual solution ν, find (w,b) that satisfies both of the above
- For SVM, second condition is

$$\partial g^*(-L^T\nu) = \begin{bmatrix} \frac{1}{\lambda}(-X_{\phi,Y}^T\nu) \\ \partial\iota_{\{0\}}(-Y^T\nu) \end{bmatrix} \ni \begin{bmatrix} w \\ b \end{bmatrix}$$

  which gives optimal $w = -\frac{1}{\lambda}X_{\Phi,Y}^T\nu$ (since unique)
- Cannot recover b from this condition

Primal solution recovery – Bias term

- Necessary and sufficient primal-dual optimality conditions

$$0 \in \begin{cases} \partial f^*(\nu) - L(w,b) \\ \partial g^*(-L^T\nu) - (w,b) \end{cases}$$

- For SVM, row i of first condition is $0 \in \partial f_i^*(\nu_i) - L_i(w,b)$ where

$$\partial f_i^*(\nu_i) = \begin{cases} [-\infty, 1] & \text{if } \nu_i = -1 \\ \{1\} & \text{if } -1 < \nu_i < 0 \\ [1, \infty] & \text{if } \nu_i = 0 \\ \emptyset & \text{else} \end{cases}, \quad L_i = y_i[\phi(x_i)^T\ 1]$$

- Pick i with $\nu_i \in (-1, 0)$, then unique subgradient $\partial f_i(\nu_i)$ is 1 and

$$0 = 1 - y_i(w^T\phi(x_i) + b)$$

  and optimal b must satisfy $b = y_i - w^T\phi(x_i)$ for such i

Outline

- Classification
- Support vector machines
- Nonlinear features
- Overfitting and regularization
- Dual problem
- **Kernel SVM**
- Training problem properties

SVM dual – A reformulation

- Dual problem

$$\begin{array}{ll} \underset{\nu}{\text{minimize}} & \sum_{i=1}^N \nu_i + \frac{1}{2\lambda}\nu^T X_{\phi,Y}X_{\phi,Y}^T\nu \\ \text{subject to} & -\mathbf{1} \leq \nu \leq \mathbf{0} \\ & Y^T\nu = 0 \end{array}$$

- Let $\kappa_{ij} := \phi(x_i)^T\phi(x_j)$ and rewrite quadratic term:

$$\begin{aligned} \nu^T X_{\phi,Y}X_{\phi,Y}^T\nu &= \nu\,\mathbf{diag}(Y)\begin{bmatrix} \phi(x_1)^T \\ \vdots \\ \phi(x_N)^T \end{bmatrix}\begin{bmatrix} \phi(x_1) & \cdots & \phi(x_N) \end{bmatrix}\mathbf{diag}(Y)\nu \\ &= \nu\,\mathbf{diag}(Y)\underbrace{\begin{bmatrix} \kappa_{11} & \cdots & \kappa_{1N} \\ \vdots & \ddots & \vdots \\ \kappa_{N1} & \cdots & \kappa_{NN} \end{bmatrix}}_{K}\mathbf{diag}(Y)\nu \end{aligned}$$

  where K is called *Kernel matrix*

SVM dual – Kernel formulation

- Dual problem with Kernel matrix

$$\begin{array}{ll} \underset{\nu}{\text{minimize}} & \sum_{i=1}^N \nu_i + \frac{1}{2\lambda}\nu^T\,\mathbf{diag}(Y)K\,\mathbf{diag}(Y)\nu \\ \text{subject to} & -\mathbf{1} \leq \nu \leq \mathbf{0} \\ & Y^T\nu = 0 \end{array}$$

- Solved without evaluating features, only scalar products:

$$\kappa_{ij} := \phi(x_i)^T\phi(x_j)$$

Kernel methods

- We explicitly defined features and created Kernel matrix
- We can instead create Kernel that implicitly defines features

Kernel operators

- Define:
  - Kernel operator $\kappa(x,y) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$
  - Kernel shortcut $\kappa_{ij} = \kappa(x_i, x_j)$
  - A Kernel matrix

$$K = \begin{bmatrix} \kappa_{11} & \cdots & \kappa_{1N} \\ \vdots & \ddots & \vdots \\ \kappa_{N1} & \cdots & \kappa_{NN} \end{bmatrix}$$

- A Kernel operator $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is:
  - *symmetric* if $\kappa(x,y) = \kappa(y,x)$
  - *positive semidefinite* (PSD) if symmetric and

$$\sum_{i,j}^{m} a_i a_j \kappa(x_i, x_j) \geq 0$$

for all $m \in \mathbb{N}$, $\alpha_i, \alpha_j \in \mathbb{R}$, and $x_i, x_j \in \mathbb{R}^n$

- All Kernel matrices PSD if Kernel operator PSD

Mercer's theorem

- Assume κ is a positive semidefinite Kernel operator
- Mercer's theorem:

> There exists continuous functions $\{e_j\}_{j=1}^{\infty}$ and nonnegative $\{\lambda_j\}_{j=1}^{\infty}$ such that
>
> $$\kappa(x,y) = \sum_{j=1}^{\infty} \lambda_j e_j(x) e_j(y)$$

- Let $\phi(x) = (\sqrt{\lambda_1} e_1(x), \sqrt{\lambda_2} e_2(x), \ldots)$ be a feature map, then

$$\kappa(x,y) = \langle \phi(x), \phi(y) \rangle$$

where scalar product in ℓ_2 (space of square summable sequences)

- A PSD kernel operator implicitly defines features

Kernel SVM dual and corresponding primal

- SVM dual from Kernel κ with Kernel matrix $K_{ij} = \kappa(x_i, x_j)$

$$\begin{array}{ll} \underset{\nu}{\text{minimize}} & \sum_{i=1}^{N} \nu_i + \frac{1}{2\lambda} \nu \, \mathbf{diag}(Y) K \, \mathbf{diag}(Y) \nu \\ \text{subject to} & -\mathbf{1} \leq \nu \leq \mathbf{0} \\ & Y^T \nu = 0 \end{array}$$

- Due to Mercer's theorem, this is dual to primal problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \max(0, 1 - y_i(\langle w, \phi(x_i) \rangle + b)) + \tfrac{\lambda}{2} \|w\|^2$$

with potentially an infinite number of features ϕ and variables w

Primal recovery and class prediction

- Assume we know Kernel operator, dual solution, but not features
  - Can recover: Label prediction and primal solution b
  - Cannot recover: Primal solution w (might be infinite dimensional)
- Primal solution $b = y_i - w^T \phi(x_i)$:

$$w^T \phi(x_i) = -\tfrac{1}{\lambda} \nu^T X_{\phi,Y} \phi(x_i) = -\tfrac{1}{\lambda} \nu^T \begin{bmatrix} y_1 \phi(x_1)^T \\ \vdots \\ y_N \phi(x_N)^T \end{bmatrix} \phi(x_i) = -\tfrac{1}{\lambda} \nu^T \begin{bmatrix} y_1 \kappa_{1i} \\ \vdots \\ y_N \kappa_{Ni} \end{bmatrix}$$

- Label prediction for new data x (sign of $w^T \phi(x) + b$):

$$w^T \phi(x) + b = -\tfrac{1}{\lambda} \nu^T \begin{bmatrix} y_1 \phi(x_1)^T \phi(x) \\ \vdots \\ y_N \phi(x_N)^T \phi(x) \end{bmatrix} + b = -\tfrac{1}{\lambda} \nu^T \begin{bmatrix} y_1 \kappa(x_1, x) \\ \vdots \\ y_N \kappa(x_N, x) \end{bmatrix} + b$$

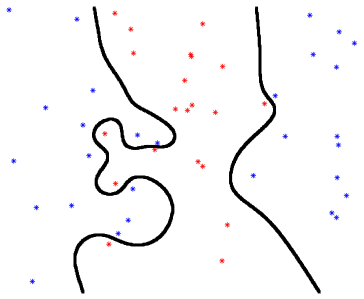
- We are really interested in label prediction, not primal solution

Valid kernels

- Polynomial kernel of degree d: $\kappa(x,y) = (1 + x^T y)^d$
- Radial basis function kernels:
  - Gaussian kernel: $\kappa(x,y) = e^{-\frac{\|x-y\|_2^2}{2\sigma^2}}$
  - Laplacian kernel: $\kappa(x,y) = e^{-\frac{\|x-y\|_2}{\sigma}}$
- Bias term b often not needed with Kernel methods

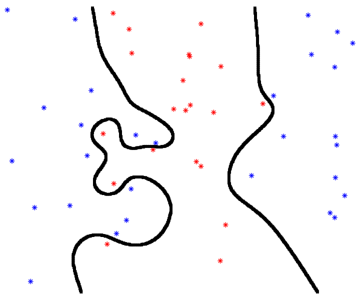
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 0.01$



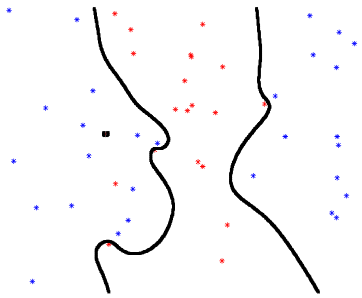
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 0.035938$



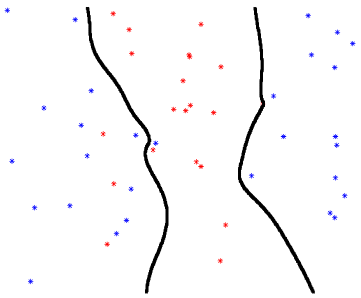
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 0.12915$



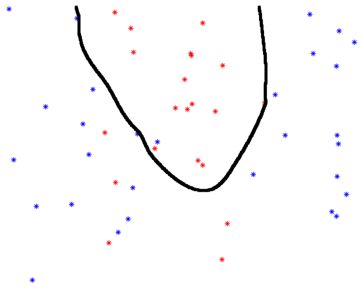
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 0.46416$



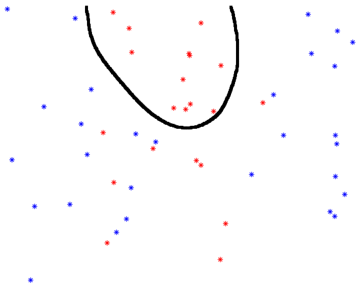
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 1.6681$



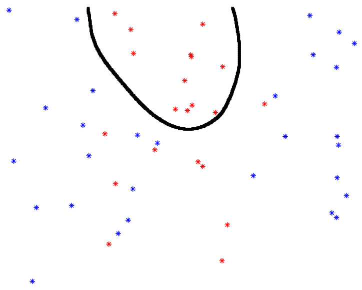
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 5.9948$



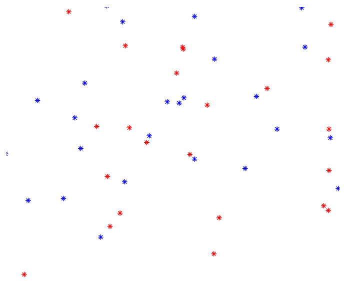
Example – Laplacian Kernel

- Regularized SVM with Laplacian Kernel with $\sigma = 1$
- Regularization parameter: $\lambda = 21.5443$



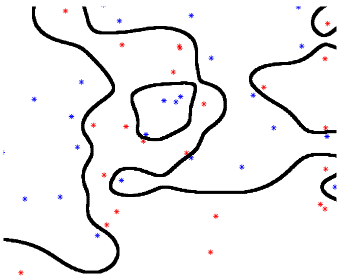
Example – Laplacian Kernel

- What if there is no structure in data? (Labels are randomly set)



Example – Laplacian Kernel

- What if there is no structure in data? (Labels are randomly set)
- Regularized SVM Laplacian Kernel, regularization parameter: $\lambda = 0.01$



- Linearly separable in high dimensional feature space
- Can be prone to overfitting ⇒ Regularize and use cross validation

---

Outline

- Classification
- Support vector machines
- Nonlinear features
- Overfitting and regularization
- Dual problem
- Kernel SVM
- **Training problem properties**

Composite optimization – Dual SVM

Dual SVM problems

$$\begin{array}{ll}\underset{\nu}{\text{minimize}} & \sum_{i=1}^N \nu_i + \frac{1}{2\lambda}\nu^T X_{\phi,Y} X_{\phi,Y}^T \nu \\ \text{subject to} & -\mathbf{1} \leq \nu \leq \mathbf{0} \\ & Y^T\nu = 0\end{array}$$

can be written on the form

$$\underset{\nu}{\text{minimize}}\, h_1(\nu) + h_2(-X_{\phi,Y}^T\nu),$$

where

- $h_1(\nu) = \mathbf{1}^T\nu + \iota_{[-\mathbf{1},\mathbf{0}]}(\nu) + \iota_{\{0\}}(Y^T\nu)$
  - First part $\mathbf{1}^T\nu + \iota_{[-\mathbf{1},\mathbf{0}]}(\nu)$ is conjugate of sum of hinge losses
  - Second part $\iota_{\{0\}}(Y^T\nu)$ comes from that bias b not regularized
- $h_2(\mu) = \frac{1}{2\lambda}\|\mu\|_2^2$ is conjugate to Tikhonov regularization $\frac{\lambda}{2}\|w\|_2^2$

Gradient and function properties

- Gradient of $(h_2 \circ -X_{\phi,Y}^T)$ satisfies:

$$\begin{aligned}\nabla(h_2 \circ -X_{\phi,Y}^T)(\nu) &= \nabla\left(\tfrac{1}{2\lambda}\nu^T X_{\phi,Y} X_{\phi,Y}^T \nu\right) = \tfrac{1}{\lambda} X_{\phi,Y} X_{\phi,Y}^T \nu \\ &= \tfrac{1}{\lambda}\,\mathbf{diag}(Y) K\,\mathbf{diag}(Y)\nu\end{aligned}$$

  where K is Kernel matrix
- Function properties
  - h_2 is convex and λ^{-1}-smooth, $h_2 \circ -X_{\phi,Y}^T$ is $\frac{\|X_{\phi,Y}\|_2^2}{\lambda}$-smooth
  - h_1 is convex and nondifferentiable, use prox in algorithms

Deep Learning

Pontus Giselsson

Outline

- **Deep learning**
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Deep learning

- Can be used both for classification and regression
- Deep learning training problem is of the form

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

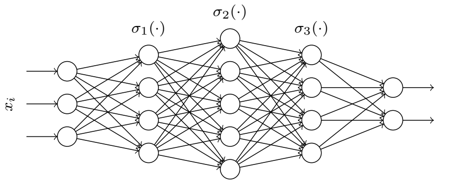
  where typically
  - $L(u,y) = \frac{1}{2}\|u-y\|_2^2$ is used for regression
  - $L(u,y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - y^T u$ is used for K-class classification
- Difference to previous convex methods: *Nonlinear model* $m(x;\theta)$
  - Deep learning regression generalizes least squares
  - DL classification generalizes multiclass logistic regression
  - Nonlinear model makes training problem nonconvex

Deep learning – Model

- Nonlinear model of the following form is often used:

  $m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$

  where θ contains all W_i and b_i
- Each activation σ_j constitutes a hidden layer in the model network
- We have no final layer activation (is instead part of loss)
- Graphical representation with three hidden layers



- Some reasons for using this structure:
  - (Assumed) universal function approximators
  - Efficient gradient computation using backpropagation

No final layer activation in classification

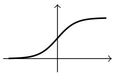
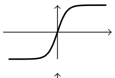
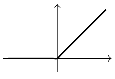
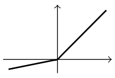
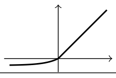
- In classification, it is common to use
  - Softmax final layer activation
  - Cross entropy loss function
- Equivalent to
  - no (identity) final layer activation
  - multiclass logistic loss

  which is what we use

Activation functions

- Activation function σ_j takes as input the output of $W_j(\cdot)+b_j$
- Often a function $\bar{\sigma}_j : \mathbb{R} \to \mathbb{R}$ is applied to each element
  - Example: $\sigma_j : \mathbb{R}^3 \to \mathbb{R}^3$ is $\sigma_j(u) = \begin{bmatrix}\bar{\sigma}_j(u_1)\\ \bar{\sigma}_j(u_2)\\ \bar{\sigma}_j(u_3)\end{bmatrix}$
- We will use notation over-loading and call both functions σ_j

Examples of activation functions

Name	$\sigma(u)$	Graph
Sigmoid	$\frac{1}{1+e^{-u}}$	
Tanh	$\frac{e^u-e^{-u}}{e^{-u}+e^u}$	
ReLU	$\max(u,0)$	
LeakyReLU	$\max(u,\alpha u)$	
ELU	$\begin{cases} u & \text{if } u \geq 0 \\ \alpha(e^u-1) & \text{else} \end{cases}$	

Examples of affine transformations

- Dense (fully connected): Dense W_j
- Sparse: Sparse W_j
  - Convolutional layer (convolution with small pictures)
  - Fixed (random) sparsity pattern
- Subsampling: reduce size, W_j fat (smaller output than input)
  - average pooling

Prediction

- Prediction as in least squares and multiclass logistic regression
- Assume model $m(x;\theta)$ trained and "optimal" $\theta^\star$ found
- Regression:
  - Predict response for new data x using $\hat{y} = m(x;\theta^\star)$
- Classification (with no final layer activation):
  - We have one model $m_j(x;\theta^\star)$ output for each class
  - Predict class belonging for new data x according to

$$\underset{j\in\{1,\ldots,K\}}{\operatorname{argmax}}\ m_j(x;\theta^\star)$$

i.e., class with largest model value (since loss designed this way)

Outline

- Deep learning
- **Learning features**
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Learning features

- Convex methods use *prespecified* feature maps (or kernels)
- Deep learning instead *learns* feature map during training
  - Define parameter dependent feature vector:

$$\phi(x;\theta) := \sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})$$

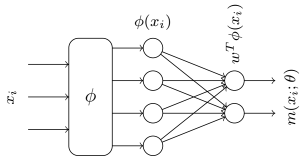
  - Model becomes $m(x;\theta) = W_n\phi(x;\theta) + b_n$
  - Inserted into training problem:

$$\underset{\theta}{\operatorname{minimize}} \sum_{i=1}^{N} L(W_n\phi(x_i;\theta) + b_n, y_i)$$

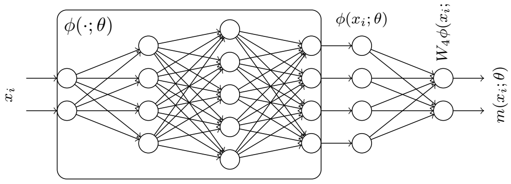
same as before, but with learned (parameter-dependent) features
- Learning features at training makes training nonconvex

Learning features – Graphical representation

- Fixed features gives convex training problems



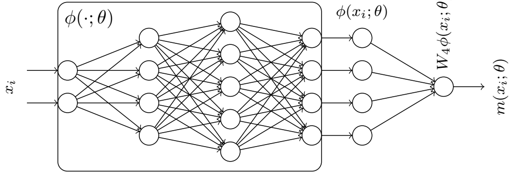
- Learning features gives nonconvex training problems



- Output of last activation function is feature vector

Optimizing only final layer

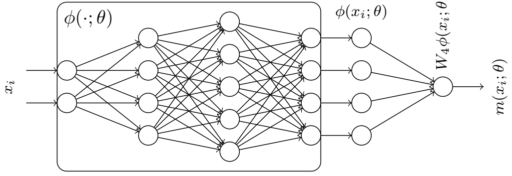
- Assume:
  - that parameters $\bar{\theta}_f$ in the layers in the square are fixed
  - that we optimize only the final layer parameters
  - that the loss is a (binary) logistic loss



- What can you say about the training problem?

Optimizing only final layer

- Assume:
  - that parameters $\bar{\theta}_f$ in the layers in the square are fixed
  - that we optimize only the final layer parameters
  - that the loss is a (binary) logistic loss



- What can you say about the training problem?
  - It reduces to logistic regression with fixed features $\phi(x_i;\bar{\theta}_f)$

$$\underset{\theta=(W_n,b_n)}{\operatorname{minimize}} \sum_{i=1}^{N} L(W_n\phi(x_i;\bar{\theta}_f) + b_n, y_i)$$

  - The training problem is convex

Design choices

Many design choices in building model to create good features

- Number of layers
- Width of layers
- Types of layers
- Types of activation functions
- Different model structures (e.g., residual network)

Outline

- Deep learning
- Learning features
- **Model properties and activation functions**
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Model properties – ReLU networks

- Recall model

  $m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$

  where θ contains all W_i and b_i
- Assume that all activation functions are (Leaky)ReLU
- What can you say about the properties of $m(\cdot;\theta)$ for fixed θ?

Model properties – ReLU networks

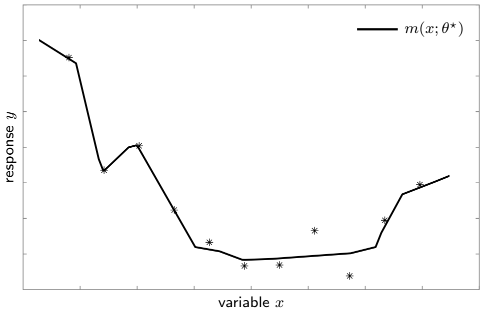
- Recall model

  $m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$

  where θ contains all W_i and b_i
- Assume that all activation functions are (Leaky)ReLU
- What can you say about the properties of $m(\cdot;\theta)$ for fixed θ?
  - It is continuous piece-wise affine

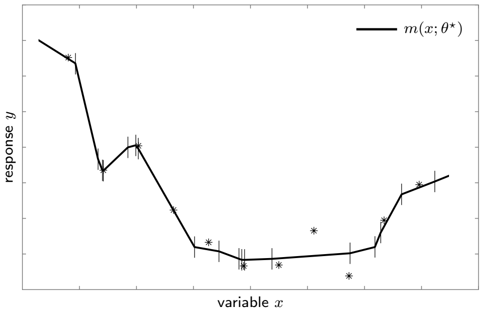
1D Regression – Model properties

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyReLU



1D Regression – Model properties

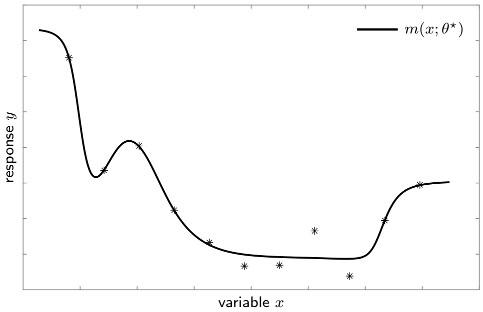
- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyReLU



- Vertical lines show kinks

1D Regression – Model properties

- Fully connected, layers widths: 5,5,5,1,1 (78 params), Tanh



- No kinks for Tanh

Identity activation

- Do we need nonlinear activation functions?
- What can you say about model if all $\sigma_j = \mathrm{Id}$ in

  $m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$

  where θ contains all W_j and b_j

Identity activation

- Do we need nonlinear activation functions?
- What can you say about model if all $\sigma_j = \mathrm{Id}$ in

  $m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$

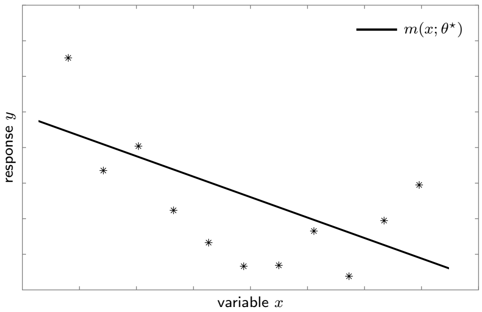
  where θ contains all W_j and b_j
- We then get

$$\begin{aligned} m(x;\theta) &:= W_n(W_{n-1}(\cdots(W_2(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n \\ &= \underbrace{W_nW_{n-1}\cdots W_2W_1}_{W}x + \underbrace{b_n + \sum_{l=2}^{n-1} W_n\cdots W_l b_{l-1}}_{b} \\ &= Wx+b \end{aligned}$$

  which is linear in x (but training problem nonconvex)

Network with identity activations – Example

- Fully connected, layers widths: 5,5,5,1,1 (78 params), Identity



Outline

- Deep learning
- Learning features
- Model properties and activation functions
- **Loss landscape**
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Training problem properties

- Recall model

$$m(x;\theta) := W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x+b_1)+b_2)\cdots)+b_{n-1})+b_n$$

where θ includes all W_j and b_j and training problem

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta),y_i)$$

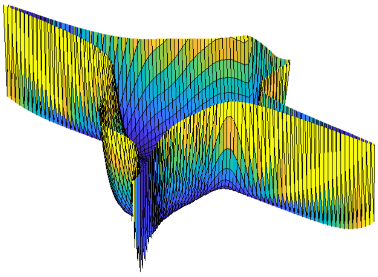
- If all σ_j LeakyReLU and $L(u,y)=\frac{1}{2}\|u-y\|_2^2$, then for fixed x,y
  - $m(x;\cdot)$ is continuous piece-wise polynomial (cpp) of degree n in θ
  - $L(m(x;\theta),y)$ is cpp of degree $2n$ in θ

where both model output and loss can grow fast

- If σ_j is instead Tanh
  - model no longer piece-wise polynomial (but "more" nonlinear)
  - model output grows slower since $\sigma_j:\mathbb{R}\to(-1,1)$

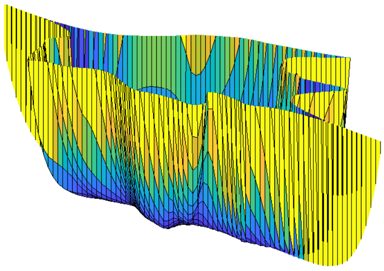
Loss landscape – Leaky ReLU

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- First choice of θ_1 and θ_2:



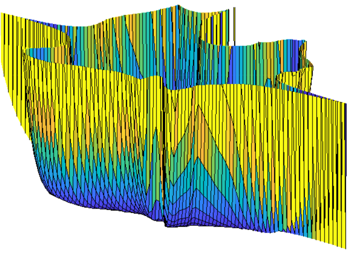
Loss landscape – Leaky ReLU

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- Second choice of θ_1 and θ_2:



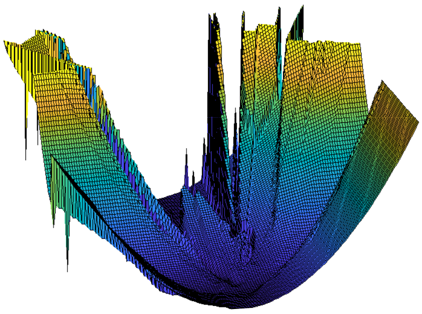
Loss landscape – Leaky ReLU

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- Third choice of θ_1 and θ_2:



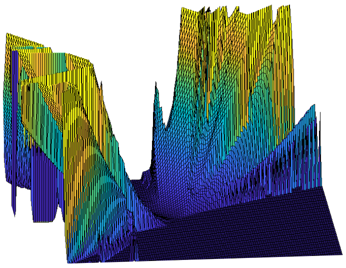
Loss landscape – Tanh

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- First choice of θ_1 and θ_2:



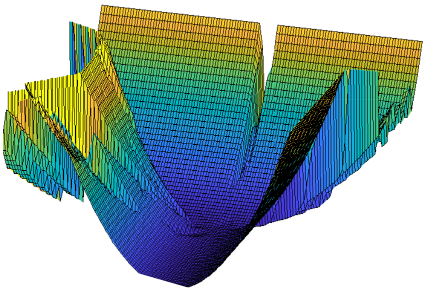
Loss landscape – Tanh

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- Second choice of θ_1 and θ_2:



Loss landscape – Tanh

- Fully connected, layers widths: 5,5,5,1,1 (78 params), LeakyRelu
- Regression problem, least squares loss
- Plot: $\sum_{i=1}^{N} L(m(x_i;\theta^\star+t_1\theta_1+t_2\theta_2),y_i)$ vs scalars t_1, t_2, where
  - $\theta^\star$ is numerically found solution to training problem
  - θ_1 and θ_2 are random directions in parameter space
- Third choice of θ_1 and θ_2:



ReLU vs Tanh

Previous figures suggest:

- ReLU: more regular and similar loss landscape?
- Tanh: less steep (on macro scale)?
- Tanh: Minima extend over larger regions?

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- **Residual networks**
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Performance with increasing depth

- Increasing depth can deteriorate performance
- Deep networks may even have worse training errors than shallow
- Intuition: deeper layers bad at approximating identity mapping

Residual networks

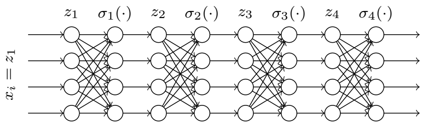
- Add skip connections between layers
- Instead of network architecture with $z_1 = x_i$ (see figure):

$$z_{j+1} = \sigma_j(W_j z_j + b_j) \text{ for } j \in \{1, \dots, n-1\}$$

use residual architecture

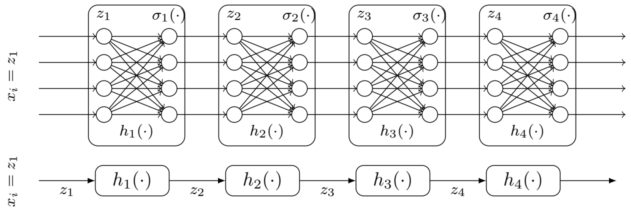
$$z_{j+1} = z_j + \sigma_j(W_j z_j + b_j) \text{ for } j \in \{1, \dots, n-1\}$$

- Assume $\sigma(0) = 0$, $W_j = 0$, $b_j = 0$ for $j = 1, \dots, m$ $(m < n-1)$
  $\Rightarrow$ deeper part of network is identity mapping and does no harm
- Learns variation from identity mapping (residual)



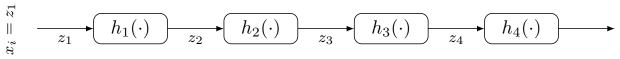
Graphical representation

For graphical representation, first collapse nodes into single node

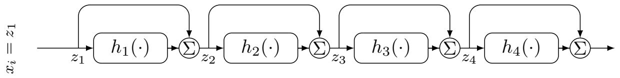


Graphical representation

- Collapsed network representation



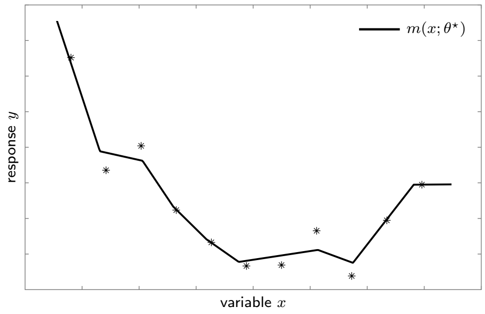
- Residual network



- If some $h_j = 0$ gives same performance as shallower network

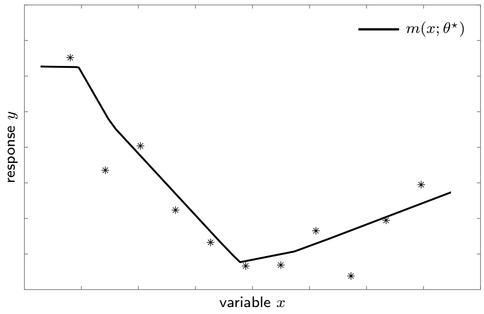
Residual network – Example

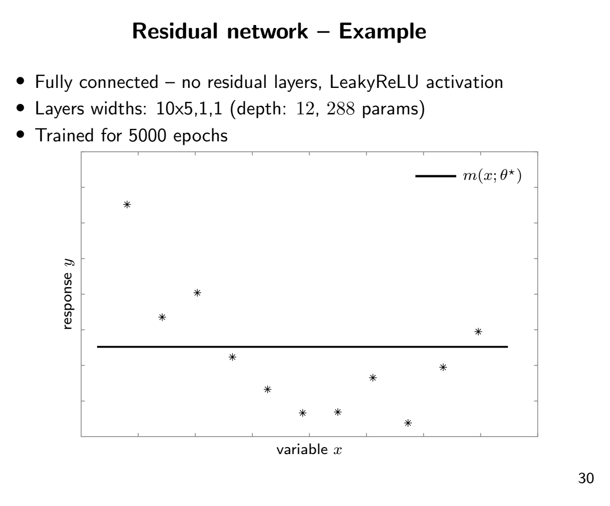
- Fully connected – no residual layers, LeakyReLU activation
- Layers widths: 3x5,1,1 (depth: 5, 78 params)
- Trained for 5000 epochs



Residual network – Example

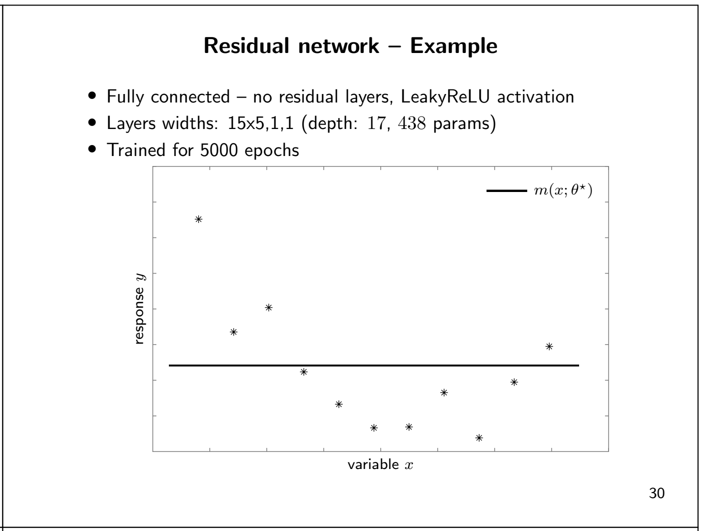
- Fully connected – no residual layers, LeakyReLU activation
- Layers widths: 5x5,1,1 (depth: 7, 138 params)
- Trained for 5000 epochs





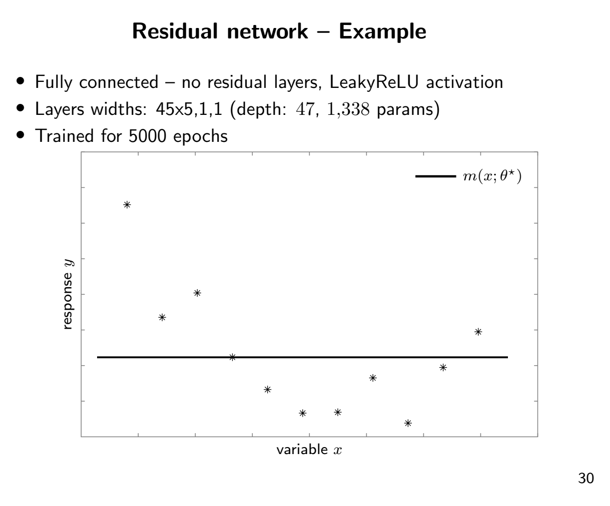
Residual network – Example

- Fully connected – no residual layers, LeakyReLU activation
- Layers widths: 10x5,1,1 (depth: 12, 288 params)
- Trained for 5000 epochs



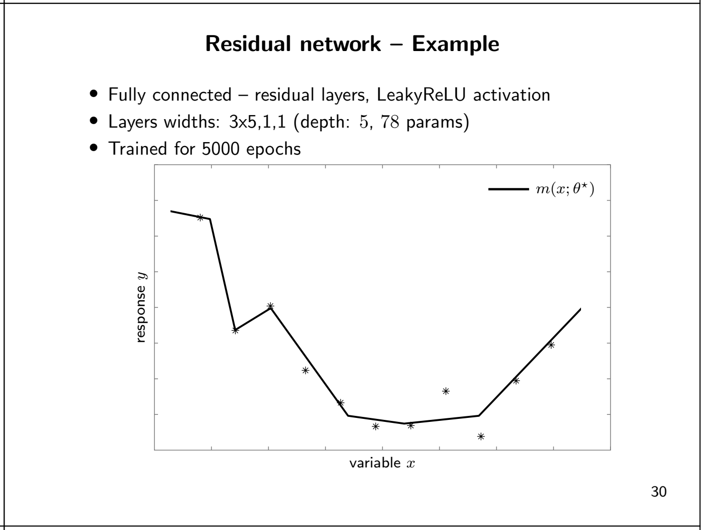
Residual network – Example

- Fully connected – no residual layers, LeakyReLU activation
- Layers widths: 15x5,1,1 (depth: 17, 438 params)
- Trained for 5000 epochs



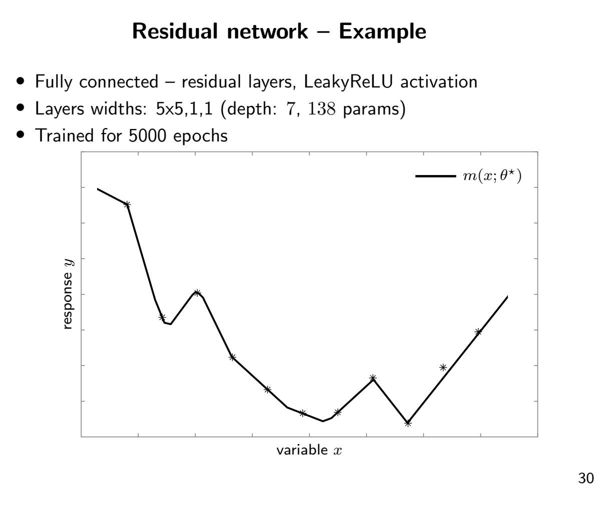
Residual network – Example

- Fully connected – no residual layers, LeakyReLU activation
- Layers widths: 45x5,1,1 (depth: 47, $1{,}338$ params)
- Trained for 5000 epochs



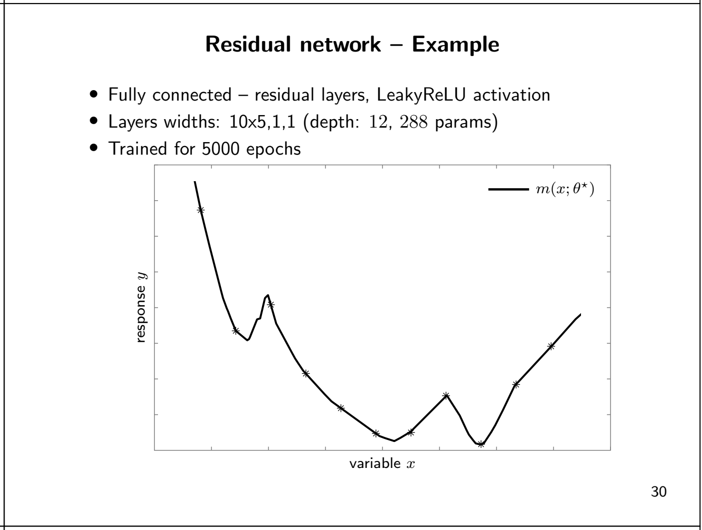
Residual network – Example

- Fully connected – residual layers, LeakyReLU activation
- Layers widths: 3x5,1,1 (depth: 5, 78 params)
- Trained for 5000 epochs



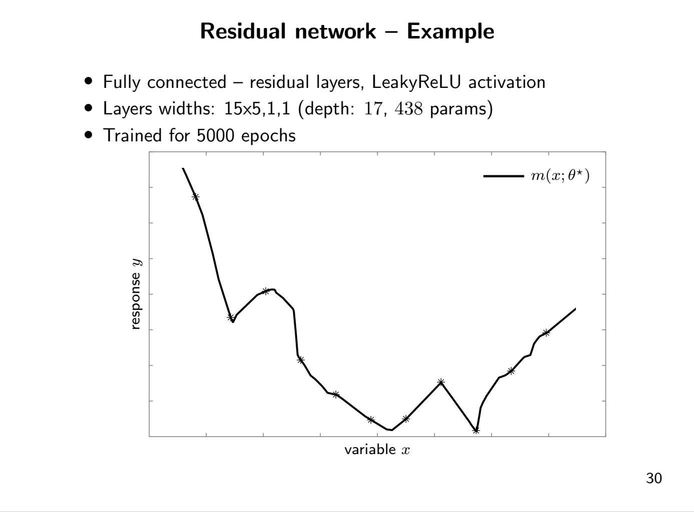
Residual network – Example

- Fully connected – residual layers, LeakyReLU activation
- Layers widths: 5x5,1,1 (depth: 7, 138 params)
- Trained for 5000 epochs



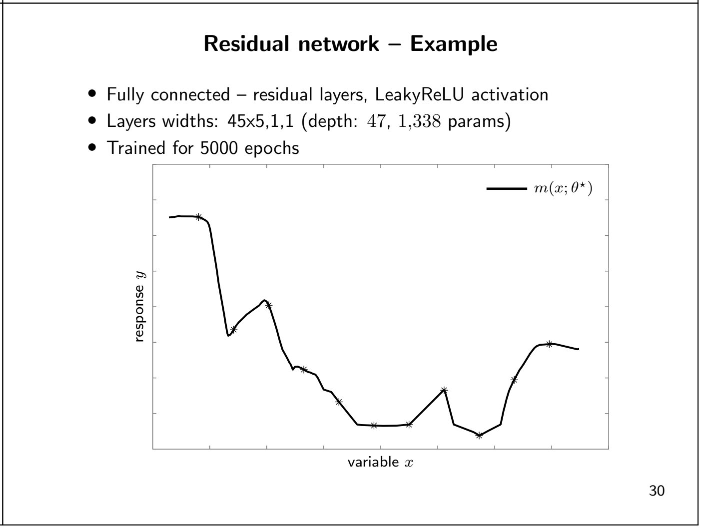
Residual network – Example

- Fully connected – residual layers, LeakyReLU activation
- Layers widths: 10x5,1,1 (depth: 12, 288 params)
- Trained for 5000 epochs



Residual network – Example

- Fully connected – residual layers, LeakyReLU activation
- Layers widths: 15x5,1,1 (depth: 17, 438 params)
- Trained for 5000 epochs



Residual network – Example

- Fully connected – residual layers, LeakyReLU activation
- Layers widths: 45x5,1,1 (depth: 47, $1{,}338$ params)
- Trained for 5000 epochs

Outline

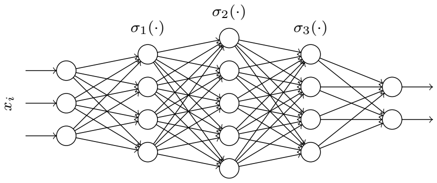
- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- **Overparameterized networks**
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Why overparameterization?

- Neural networks are often overparameterized in practice
- Why? They often perform better than underparameterized

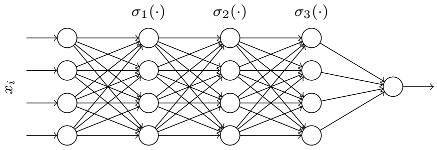
What is overparameterization?

- We mean that many solutions exist that can:
  - fit all data points (0 training loss) in regression
  - correctly classify all training examples in classification
- This requires (many) more parameters than training examples
  - Need wide and deep enough networks
  - Can result in overfitting
- Questions:
  - Which of all solutions give best generalization?
  - (How) can network design affect generalization?



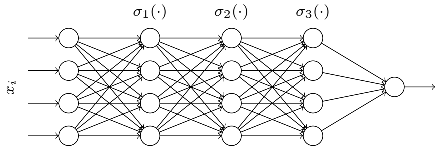
Overparameterization – An example

- Assume fully connected network with
  - input data $x_i \in \mathbb{R}^p$
  - n layers and $N \approx p^2$ samples
  - same width throughout (except last layer, which can be neglected)
- What is the relation between number of weights and samples?



Overparameterization – An example

- Assume fully connected network with
  - input data $x_i \in \mathbb{R}^p$
  - n layers and $N \approx p^2$ samples
  - same width throughout (except last layer, which can be neglected)
- What is the relation between number of weights and samples?



- We have:
  - Number of parameters approximately: $(W_j)_{lk}$: p^2n and $(b_j)_l$: pn
  - Then $\frac{\#\text{weights}}{\#\text{samples}} \approx \frac{p^2n}{p^2} = n$ more weights than samples

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- **Generalization and regularization**
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Generalization

- Most important for model to generalize well to unseen data
- General approach in training
  - Train a model that is too expressive for the underlying data
    - Overparameterization in deep learning
  - Use regularization to
    - find model of appropriate (lower) complexity
    - favor models with desired properties

Regularization

What regularization techniques in DL are you familiar with?

Regularization techniques

- Reduce number of parameters
  - Sparse weight tensors (e.g., convolutional layers)
  - Subsampling (gives fewer parameters deeper in network)
- Explicit regularization term in cost function, e.g., Tikhonov
- Data augmentation – more samples, artificial often OK
- Early stopping – stop algorithm before convergence
- Dropouts
- ...

Implicit vs explicit regularization

- Regularization can be explicit or implicit
- Explicit – Introduce something with intent to regularize:
  - Add cost function to favor desirable properties
  - Design (adapt) network to have regularizing properties
- Implicit – Use something with regularization as byproduct:
  - Use algorithm that finds favorable solution among many
  - Will look at implicit regularization via SGD

Generalization – Our focus

Will here discuss generalization via:

- Norm of parameters – leads to implicit regularization via SGD
- Flatness of minima – leads to implicit regularization via SGD

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- **Generalization – Norm of weights**
- Generalization – Flatness of minima
- Backpropagation
- Vanishing and exploding gradients

Lipschitz continuity of ReLU networks

- Assume that all activation functions 1-Lipschitz continuous
- The neural network model $m(\cdot;\theta)$ is Lipschitz continuous in x,

$$\|m(x_1;\theta)-m(x_2;\theta)\|_2 \leq L\|x_1-x_2\|_2$$

  for fixed θ, e.g., the θ obtained after training
- This means output differerences are bounded by input differences
- A Lipschitz constant L is given by

$$L=\|W_n\|_2\cdot\|W_{n-1}\|_2\cdots\|W_1\|_2$$

  since activation functions are 1-Lipschitz continuous
- For residual layers each $\|W_j\|_2$ replaced by $(1+\|W_j\|_2)$

Desired Lipschitz constant

- Overparameterization gives many solutions that perfectly fit data
- Would you favor one with high or low Lipschitz constant L?

Small norm likely to generalize better

- Smaller Lipschitz constant probably generalizes better if perfect fit
- “Similar inputs give similar outputs”, recall

$$\|m(x_1;\theta)-m(x_2;\theta)\|_2 \leq L\|x_1-x_2\|_2$$

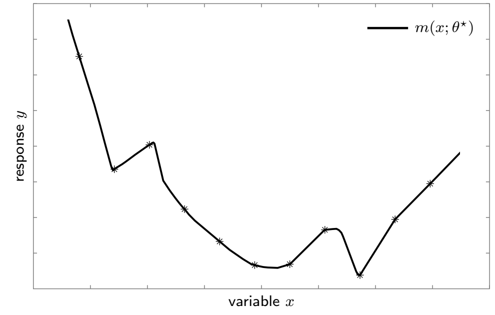
  with a Lipschitz constant is given by

$$L=\|W_n\|_2\cdot\|W_{n-1}\|_2\cdots\|W_1\|_2$$

  or with $\|W_j\|_2$ replaced by $(1+\|W_j\|_2)$ for residual layers
- Smaller weight norms give better generalization if perfect fit

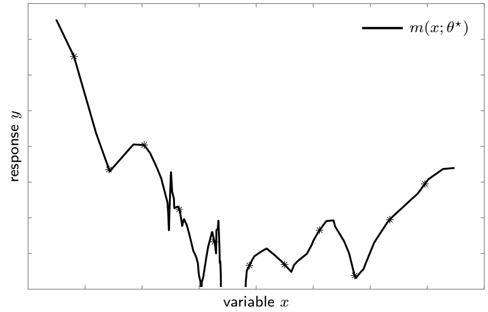
Generalization – Norm of weights

- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 72



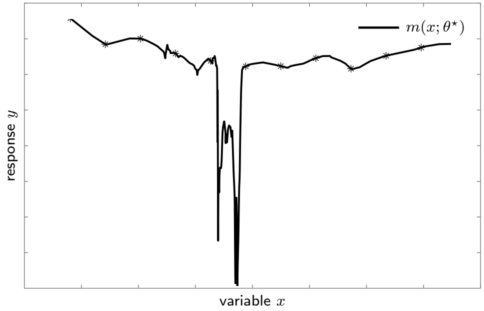
Generalization – Norm of weights

- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 540



Generalization – Norm of weights

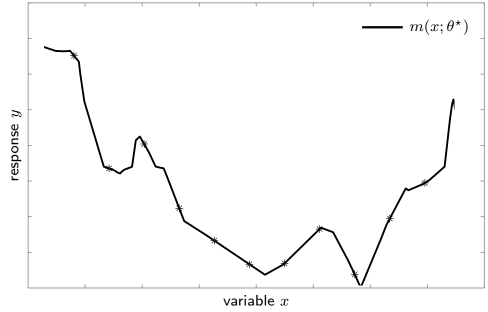
- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 540



- Same as previous, new scaling

Generalization – Norm of weights

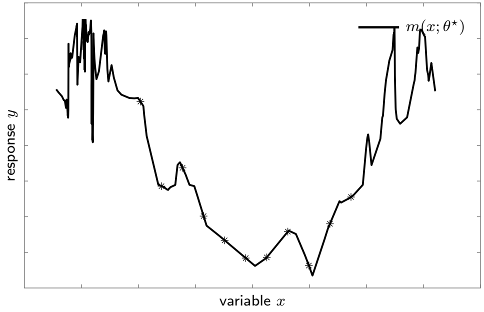
- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 595



- Large norm, but seemingly fair generalization

Generalization – Norm of weights

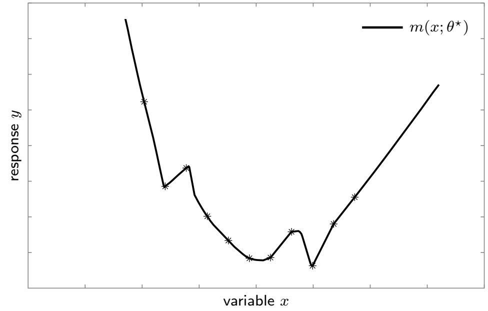
- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 595



- Same as previous, new scaling

Generalization – Norm of weights

- Fully connected – residual layers, LeakyReLU
- Layers widths: 30x5,1,1 (888 params)
- Norm of weights (with perfect fit): 72



- Same as first, new scaling – overfits less than large norm solutions

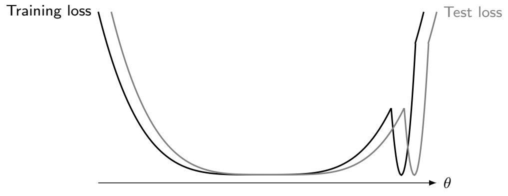
---

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- **Generalization – Flatness of minima**
- Backpropagation
- Vanishing and exploding gradients

Flatness of minima

- Consider the following illustration of *average* loss:

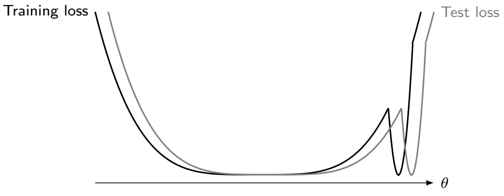


- Depicts test loss as shifted training loss
- Motivation to that flat minima generalize better than sharp

---

Flatness of minima

- Consider the following illustration of *average* loss:



- Depicts test loss as shifted training loss
- Motivation to that flat minima generalize better than sharp
- Is there a limitation in considering the average loss only?

Generalization from loss landscape

- Training set $\{(x_i, y_i)\}_{i=1}^{N}$ and training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

- Test set $\{(\hat{x}_i, \hat{y}_i)\}_{i=1}^{\hat{N}}$, θ generalizes well if test loss small

$$\sum_{i=1}^{\hat{N}} L(m(\hat{x}_i;\theta), \hat{y}_i)$$

- By overparameterization, we can for each $(\hat{x}_i, \hat{y}_i)$ find $\hat{\theta}_i$ so that

$$L(m(\hat{x}_i;\theta), \hat{y}_i) = L(m(x_{j_i};\theta + \hat{\theta}_i), y_{j_i})$$

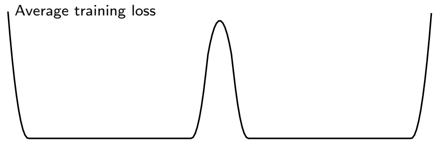
  for all θ given a (similar) (x_{j_i}, y_{j_i}) pair in training set
  - Evaluate test loss by training loss at shifted points $\theta + \hat{\theta}_i$ 1)
  - Test loss small if original individual loss small at all $\theta + \hat{\theta}_i$
  - Previous figure used same $\hat{\theta}_i = \hat{\theta}$ for all i

---

1) Don't compute in practice, just thought experiment to connect generalization to training loss

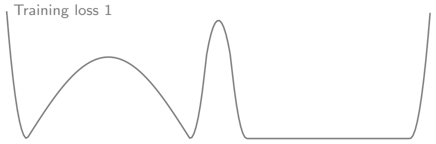
Example

- Can flat (local) minima be different?
- Does one of the following minima generalize better?



Example

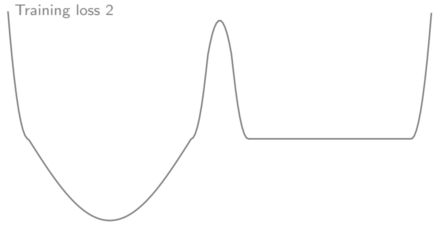
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses

Example

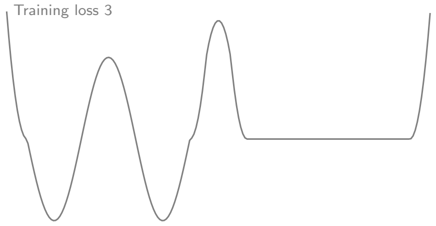
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses

Example

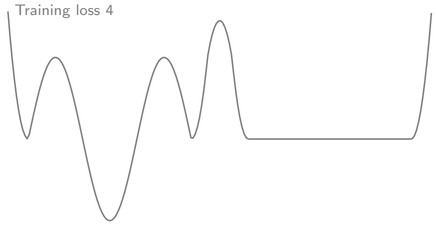
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses

Example

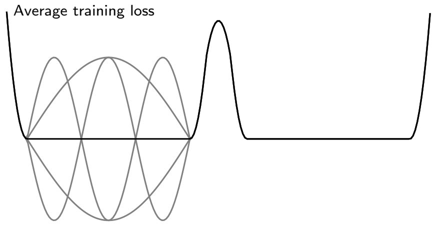
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses

Example

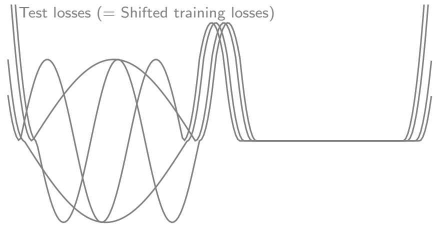
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses

Example

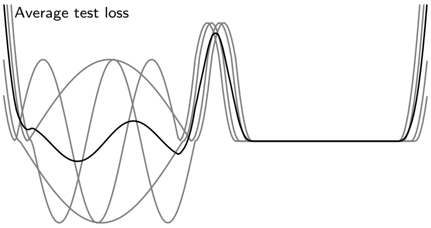
- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses
- Let us evaluate test loss by shifting individual training losses

Example

- Can flat (local) minima be different?
- Does one of the following minima generalize better?



- It depends on individual losses
- Let us evaluate test loss by shifting individual training losses
- Do not only want flat minima, want individual losses flat at minima

Individually flat minima

- Both flat minima have $\nabla f(\theta) = 0$, but
  - One minima has large individual gradients $\|\nabla f_i(\theta)\|$
  - Other minima has small individual gradients $\|\nabla f_i(\theta)\|$
  - The latter (individually flat minima) seems to generalize better
- Want individually flat minima (with small $\|\nabla f_i(\theta)\|$)
  - This implies average flat minima
  - The reverse implication may not hold
  - Overparameterized networks:
    - The reverse implication may often hold at global minima
    - Why? $f(\theta) = 0$ and $\nabla f(\theta) = 0$ implies $f_i(\theta) = 0$ and $\nabla f_i(\theta) = 0$

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- **Backpropagation**
- Vanishing and exploding gradients

Training algorithm

- Neural networks often trained using stochastic gradient descent
- DNN weights are updated via gradients in training
- Gradient of cost is sum of gradients of summands (samples)
- Gradient of each summand computed using backpropagation

Backpropagation

- Backpropagation is reverse mode automatic differentiation
- Based on chain-rule in differentiation
- Backpropagation must be performed per sample
- Our derivation assumes:
  - Fully connected layers (W full, if not, set elements in W to 0)
  - Activation functions $\sigma_j(v) = (\sigma_j(v_1), \ldots, \sigma_j(v_p))$ element-wise (overloading of σ_j notation)
  - Weights W_j are matrices, samples x_i and responses y_i are vectors
  - No residual connections

Jacobians

- The Jacobian of a function $f : \mathbb{R}^n \to \mathbb{R}^m$ is given by

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

- The Jacobian of a function $f : \mathbb{R}^{p \times n} \to \mathbb{R}$ is given by

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_{11}} & \cdots & \frac{\partial f}{\partial x_{1n}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f}{\partial x_{p1}} & \cdots & \frac{\partial f}{\partial x_{pn}} \end{bmatrix} \in \mathbb{R}^{p \times n}$$

- The Jacobian of a function $f : \mathbb{R}^{p \times n} \to \mathbb{R}^m$ is at layer j given by

$$\left[\frac{\partial f}{\partial x}\right]_{:,j,:} = \begin{bmatrix} \frac{\partial f_1}{\partial x_{j1}} & \cdots & \frac{\partial f_1}{\partial x_{jn}} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_m}{\partial x_{j1}} & \cdots & \frac{\partial f_m}{\partial x_{jn}} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

  the full Jacobian is a 3D tensor in $\mathbb{R}^{m \times p \times n}$

Jacobian vs gradient

- The Jacobian of a function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \cdots & \frac{\partial f}{\partial x_n} \end{bmatrix}$$

- The gradient of a function $f : \mathbb{R}^n \to \mathbb{R}$ is given by

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

  i.e., transpose of Jacobian for $f : \mathbb{R}^n \to \mathbb{R}$
- Chain rule holds for Jacobians:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z}\frac{\partial z}{\partial x}$$

Jacobian vs gradient – Example

- Consider differentiable $f : \mathbb{R}^m \to \mathbb{R}$ and $M \in \mathbb{R}^{m \times n}$
- Compute Jacobian of $g = (f \circ M)$ using chain rule:
  - Rewrite as $g(x) = f(z)$ where $z = Mx$
  - Compute Jacobian by partial Jacobians $\frac{\partial f}{\partial z}$ and $\frac{\partial z}{\partial x}$:

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial f}{\partial z}\frac{\partial z}{\partial x} = \nabla f(z)^T M = \nabla f(Mx)^T M \in \mathbb{R}^{1 \times n}$$

- Know gradient of $(f \circ M)(x)$ satisfies

$$\nabla (f \circ M)(x) = M^T \nabla f(Mx) \in \mathbb{R}^n$$

  which is transpose of Jacobian

Backpropagation – Introduce states

- Compute gradient/Jacobian of

$$L(m(x_i;\theta), y_i)$$

w.r.t. $\theta = \{(W_j, b_j)\}_{j=1}^n$, where

$$m(x_i;\theta) = W_n\sigma_{n-1}(W_{n-1}\sigma_{n-2}(\cdots(W_2\sigma_1(W_1x_i+b_1)+b_2)\cdots)+b_{n-1})+b_n$$

- Rewrite as function with states z_j

$$\begin{aligned} & L(z_{n+1}, y_i) \\ \text{where} \quad & z_{j+1} = \sigma_j(W_jz_j+b_j) \text{ for } j \in \{1,\ldots,n\} \\ \text{and} \quad & z_1 = x_i \end{aligned}$$

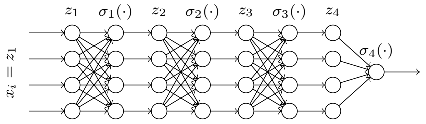
where $\sigma_n(u) \equiv u$

Graphical representation

- Per sample loss function

$$\begin{aligned} & L(z_{n+1}, y_i) \\ \text{where} \quad & z_{j+1} = \sigma_j(W_jz_j+b_j) \text{ for } j \in \{1,\ldots,n\} \\ \text{and} \quad & z_1 = x_i \end{aligned}$$

where $\sigma_n(u) \equiv u$
- Graphical representation



Backpropagation – Chain rule

- Jacobian of L w.r.t. W_j and b_j can be computed as

$$\begin{aligned} \frac{\partial L}{\partial W_j} &= \frac{\partial L}{\partial z_{n+1}}\frac{\partial z_{n+1}}{\partial z_n}\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial W_j} \\ \frac{\partial L}{\partial b_j} &= \frac{\partial L}{\partial z_{n+1}}\frac{\partial z_{n+1}}{\partial z_n}\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial b_j} \end{aligned}$$

where we mean derivative w.r.t. first argument in L
- Backpropagation evaluates partial Jacobians as follows

$$\begin{aligned} \frac{\partial L}{\partial W_j} &= \left(\left(\frac{\partial L}{\partial z_{n+1}}\frac{\partial z_{n+1}}{\partial z_n}\right)\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\right)\frac{\partial z_{j+1}}{\partial W_j} \\ \frac{\partial L}{\partial b_j} &= \left(\left(\frac{\partial L}{\partial z_{n+1}}\frac{\partial z_{n+1}}{\partial z_n}\right)\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\right)\frac{\partial z_{j+1}}{\partial b_j} \end{aligned}$$

Backpropagation – Forward and backward pass

- Jacobian of $L(z_{n+1}, y_i)$ w.r.t. z_{n+1} (transpose of gradient)
- Computing Jacobian of $L(z_{n+1}, y_i)$ requires z_{n+1}
$\Rightarrow$ forward pass: $z_1 = x_i$, $z_{j+1} = \sigma_j(W_jz_j+b_j)$
- Backward pass, store δ_j:

$$\frac{\partial L}{\partial z_{j+1}} = \underbrace{\left(\underbrace{\left(\underbrace{\frac{\partial L}{\partial z_{n+1}}}_{\delta_{n+1}^T}\frac{\partial z_{n+1}}{\partial z_n}\right)}_{\delta_n^T}\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\right)}_{\delta_{j+1}^T}$$

- Compute

$$\begin{aligned} \frac{\partial L}{\partial W_j} &= \frac{\partial L}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial W_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial W_j} \\ \frac{\partial L}{\partial b_j} &= \frac{\partial L}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial b_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial b_j} \end{aligned}$$

Dimensions

- Let $z_j \in \mathbb{R}^{n_j}$, consequently $W_j \in \mathbb{R}^{n_{j+1}\times n_j}$, $b_j \in \mathbb{R}^{n_{j+1}}$
- Dimensions

$$\frac{\partial L}{\partial W_j} = \underbrace{\underbrace{\left(\underbrace{\left(\underbrace{\frac{\partial L}{\partial z_{n+1}}}_{1\times n_{n+1}}\underbrace{\frac{\partial z_{n+1}}{\partial z_n}}_{n_{n+1}\times n_n}\right)}_{1\times n_n}\cdots\underbrace{\frac{\partial z_{j+2}}{\partial z_{j+1}}}_{n_{j+2}\times n_{j+1}}\right)}_{1\times n_{j+1}}\underbrace{\frac{\partial z_{j+1}}{\partial W_j}}_{n_{j+1}\times n_{j+1}\times n_j}}_{n_{j+1}\times n_j}$$

$$\frac{\partial L}{\partial b_j} = \underbrace{\underbrace{\left(\left(\frac{\partial L}{\partial z_{n+1}}\frac{\partial z_{n+1}}{\partial z_n}\right)\cdots\frac{\partial z_{j+2}}{\partial z_{j+1}}\right)}_{1\times n_{j+1}}\underbrace{\frac{\partial z_{j+1}}{\partial b_j}}_{n_{j+1}\times n_{j+1}}}_{1\times n_{j+1}}$$

- Vector matrix multiplies except for in last step
- Multiplication with tensor $\frac{\partial z_{j+1}}{\partial W_j}$ can be simplified
- Backpropagation variables $\delta_j \in \mathbb{R}^{n_j}$ are vectors (not matrices)

Partial Jacobian $\frac{\partial z_{j+1}}{\partial z_j}$

- Recall relation $z_{j+1} = \sigma_j(W_jz_j+b_j)$ and let $v_j = W_jz_j + b_j$
- Chain rule gives

$$\begin{aligned} \frac{\partial z_{j+1}}{\partial z_j} &= \frac{\partial z_{j+1}}{\partial v_j}\frac{\partial v_j}{\partial z_j} = \mathbf{diag}(\sigma_j'(v_j))\frac{\partial v_j}{\partial z_j} \\ &= \mathbf{diag}(\sigma_j'(W_jz_j+b_j))W_j \end{aligned}$$

where, with abuse of notation (notation overloading)

$$\sigma_j'(u) = \begin{bmatrix} \sigma_j'(u_1) \\ \vdots \\ \sigma_j'(u_{n_{j+1}}) \end{bmatrix}$$

- Reason: $\sigma_j(u) = [\sigma_j(u_1), \ldots, \sigma_j(u_{n_{j+1}})]^T$ with $\sigma_j : \mathbb{R}^{n_{j+1}} \to \mathbb{R}^{n_{j+1}}$, gives

$$\frac{d\sigma_j}{du} = \begin{bmatrix} \sigma_j'(u_1) & & \\ & \ddots & \\ & & \sigma_j'(u_{n_{j+1}}) \end{bmatrix} = \mathbf{diag}(\sigma_j'(u))$$

Partial Jacobian $\delta_j^T = \frac{\partial L}{\partial z_j}$

- For any vector $\delta_{j+1} \in \mathbb{R}^{n_{j+1}\times 1}$, we have

$$\begin{aligned} \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial z_j} &= \delta_{j+1}^T\mathbf{diag}(\sigma_j'(W_jz_j+b_j))W_j \\ &= (W_j^T(\delta_{j+1}^T\mathbf{diag}(\sigma_j'(W_jz_j+b_j)))^T)^T \\ &= (W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j)))^T \end{aligned}$$

where $\odot$ is element-wise (Hadamard) product
- We have defined $\delta_{n+1}^T = \frac{\partial L}{\partial z_{n+1}}$, then

$$\delta_n^T = \frac{\partial L}{\partial z_n} = \delta_{n+1}^T\frac{\partial z_{n+1}}{\partial z_n} = \underbrace{(W_n^T(\delta_{n+1}\odot\sigma_n'(W_nz_n+b_n)))}_{\delta_n}^T$$

- Consequently, using induction:

$$\delta_j^T = \frac{\partial L}{\partial z_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial z_j} = \underbrace{(W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j)))}_{\delta_j}^T$$

Information needed to compute $\frac{\partial L}{\partial z_j}$

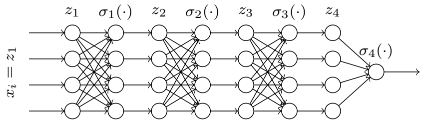
- To compute first Jacobian $\frac{\partial L}{\partial z_n}$, we need $z_n \Rightarrow$ forward pass
- Computing

$$\frac{\partial L}{\partial z_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial z_j} = (W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j)))^T = \delta_j^T$$

is done using a backward pass

$$\delta_j = W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))$$

- All z_j (or $v_j = W_jz_j + b_j$) need to be stored for backward pass



Partial Jacobian $\frac{\partial L}{\partial W_j}$

- Computed by

$$\frac{\partial L}{\partial W_j} = \frac{\partial L}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial W_j} = \delta_{j+1}^T \frac{\partial z_{j+1}}{\partial W_j}$$

where $z_{j+1} = \sigma_j(v_j)$ and $v_j = W_j z_j + b_j$
- Recall $\frac{\partial z_{j+1}}{\partial W_l}$ is 3D tensor, compute Jacobian w.r.t. row l $(W_j)_l$

$$\delta_{j+1}^T \frac{\partial z_{j+1}}{\partial (W_j)_l} = \delta_{j+1}^T \frac{\partial z_{j+1}}{\partial v_j}\frac{\partial v_j}{\partial (W_j)_l} = \delta_{j+1}^T\,\mathbf{diag}(\sigma_j'(v_j))\begin{bmatrix}0\\ \vdots\\ z_j^T\\ \vdots\\ 0\end{bmatrix}$$

$$= (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))^T\begin{bmatrix}0\\ \vdots\\ z_j^T\\ \vdots\\ 0\end{bmatrix} = (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))_l z_j^T$$

Partial Jacobian $\frac{\partial L}{\partial W_j}$ cont'd

- Stack Jacobians w.r.t. rows to get full Jacobian:

$$\frac{\partial L}{\partial W_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial W_j} = \begin{bmatrix}\delta_{j+1}^T\frac{\partial z_{j+1}}{\partial (W_j)_1}\\ \vdots\\ \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial (W_j)_{n_{j+1}}}\end{bmatrix} = \begin{bmatrix}(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))_1 z_j^T\\ \vdots\\ (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))_{n_{j+1}} z_j^T\end{bmatrix}$$
$$= (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))z_j^T$$

for all $j \in \{1,\ldots,n-1\}$

- Dimension of result is $n_{j+1}\times n_j$, which matches W_j
- This is used to update W_j weights in algorithm

Partial Jacobian $\frac{\partial L}{\partial b_j}$

- Recall $z_{j+1} = \sigma_j(v_j)$ where $v_j = W_jz_j + b_j$
- Computed by

$$\frac{\partial L}{\partial b_j} = \frac{\partial L}{\partial z_{j+1}}\frac{\partial z_{j+1}}{\partial v_j}\frac{\partial v_j}{\partial b_j} = \delta_{j+1}^T\frac{\partial z_{j+1}}{\partial v_j}\frac{\partial v_j}{\partial b_j} = \delta_{j+1}^T\,\mathbf{diag}(\sigma_j'(v_j))$$
$$= (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))^T$$

Backpropagation summarized

1. Forward pass: Compute and store z_j (or $v_j = W_jz_j + b_j$):
$$z_{j+1} = \sigma_j(W_jz_j + b_j)$$
where $z_1 = x_i$ and $\sigma_n = \mathrm{Id}$
2. Backward pass:
$$\delta_j = W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))$$
with $\delta_{n+1} = \frac{\partial L}{\partial z_{n+1}}$
3. Weight update Jacobians (used in SGD)
$$\frac{\partial L}{\partial W_j} = (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))z_j^T$$
$$\frac{\partial L}{\partial b_j} = (\delta_{j+1}\odot\sigma_j'(W_jx_j+b_j))^T$$

Backpropagation – Residual networks

1. Forward pass: Compute and store z_j (or $v_j = W_jz_j + b_j$):
$$z_{j+1} = \sigma_j(W_jz_j + b_j) + z_j$$
where $z_1 = x_i$ and $\sigma_n = \mathrm{Id}$
2. Backward pass:
$$\delta_j = W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j)) + \delta_{j+1}$$
with $\delta_{n+1} = \frac{\partial L}{\partial z_{n+1}}$
3. Weight update Jacobians (used in SGD)
$$\frac{\partial L}{\partial W_j} = (\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))z_j^T$$
$$\frac{\partial L}{\partial b_j} = (\delta_{j+1}\odot\sigma_j'(W_jx_j+b_j))^T$$

Outline

- Deep learning
- Learning features
- Model properties and activation functions
- Loss landscape
- Residual networks
- Overparameterized networks
- Generalization and regularization
- Generalization – Norm of weights
- Generalization – Flatness of minima
- Backpropagation
- **Vanishing and exploding gradients**

Vanishing and exploding gradient problem

- For some activation functions, gradients can vanish
- For other activation functions, gradients can explode

Vanishing gradient example: Sigmoid

- Assume $\|W_j\| \leq 1$ for all j and $\|\delta_{n+1}\| \leq C$
- Maximal derivative of sigmoid (σ) is 0.25
- Then

$$\left\|\frac{\partial L}{\partial z_j}\right\| = \|\delta_j\| = \|W_j^T(\delta_{j+1}\odot\sigma_j'(W_jz_j+b_j))\| \leq 0.25\|\delta_{j+1}\|$$
$$\leq 0.25^{n-j+1}\|\delta_{n+1}\| \leq 0.25^{n-j+1}C$$

- Hence, as n grows, gradients can become very small for small i
- In general, vanishing gradient if $\sigma' < 1$ everywhere
- Similar reasoning: exploding gradient if $\sigma' > 1$ everywhere
- Hence, need $\sigma' = 1$ in important regions

Vanishing gradients – Residual networks

- Residual networks with forward pass

$$z_{j+1} = \sigma_j(W_j z_j + b_j) + z_j$$

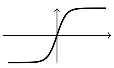
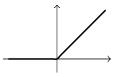
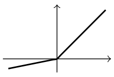
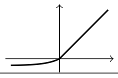
  and backward pass

$$\delta_j = W_j^T(\delta_{j+1} \odot \sigma_j'(W_j z_j + b_j)) + \delta_{j+1}$$

- Gradients do not vanish in passes despite small σ gain

Examples of activation functions

Activation functions that (partly) avoid vanishing gradients

Name	$\sigma(u)$	Graph
Tanh	$\frac{e^u - e^{-u}}{e^{-u} + e^u}$	
ReLU	$\max(u, 0)$	
LeakyReLU	$\max(u, \alpha u)$	
ELU	$\begin{cases} u & \text{if } u \geq 0 \\ \alpha(e^u - 1) & \text{else} \end{cases}$	

Exploding gradient – Example

- Assume L-Lipschitz activation (ReLU, Tanh etc have $L = 1$)
  - Forward pass estimation:

$$\begin{aligned}\|z_{j+1}\|_2 &= \|\sigma_j(W_j z_j + b_j)\|_2 \leq L\|W_j z_j + b_j\|_2 \leq L(\|W_j z_j\|_2 + \|b_j\|_2) \\ &\leq L\|W_j\|_2\|z_j\|_2 + L\|b_j\|_2\end{aligned}$$

  - Backward pass estimation:

$$\begin{aligned}\|\delta_j\|_2 &= \|W_j^T(\delta_{j+1} \odot \sigma_j'(W_j z_j + b_j))\|_2 \\ &\leq \|W_j^T\|_2\|\delta_{j+1} \odot \sigma_j'(W_j z_j + b_j)\|_2 \\ &\leq L\|W_j\|_2\|\delta_{j+1}\|_2\end{aligned}$$

  - If $L \leq 1$, $\|W_j\|_2 \leq 1$ and $\|b_j\|_2$ small, gradients do not explode
- ReLU "average" $L = 0.5$ reduces "average estimate"
- Tanh reduces "average estimates" more since
  - σ_j-outputs are constrained to $(-1, 1)$
  - "average Lipschitz constant" is smaller

Exploding gradient – Residual network

- Assume L-Lipschitz activation (ReLU, Tanh have $L = 1$)
  - Forward pass estimation:

$$\|z_{j+1}\|_2 = \|\sigma_j(W_j z_j + b_j)\|_2 + \|z_j\|_2 \leq (1 + L\|W_j\|_2)\|z_j\|_2 + L\|b_j\|_2$$

  - Backward pass estimation:

$$\begin{aligned}\|\delta_j\|_2 &= \|W_j^T(\delta_{j+1} \odot \sigma_j'(W_j z_j + b_j))\|_2 + \delta_{j+1} \\ &\leq (1 + L\|W_j\|_2)\|\delta_{j+1}\|_2\end{aligned}$$

- Larger estimates than for non-residual networks
- Activations with $L \leq 1$ to avoid exploding and vanishing gradients:
  - $\alpha\times$ReLU with $\alpha \in (0, 1)$
  - $\alpha\times$Tanh with $\alpha \in (0, 1)$

Algorithms and Convergence

Pontus Giselsson

Outline

- **Algorithm overview**
- Convergence and convergence rates
- Proving convergence rates

What is an algorithm?

- We are interested in algorithms that solve composite problems

$$\underset{x}{\text{minimize}}\, f(x) + g(x)$$

- An algorithm:
  - generates a sequence $(x_k)_{k\in\mathbb{N}}$ that hopefully converges to solution
  - often creates next point in sequence according to

$$x_{k+1} = \mathcal{A}_k x_k$$

  where
    - $\mathcal{A}_k$ is a mapping that gives the next point from the current
    - $\mathcal{A}_k = \text{prox}_{\gamma_k g}(I - \gamma_k \nabla f)$ for proximal gradient method

Deterministic and stochastic algorithms

- We have deterministic algorithms

$$x_{k+1} = \mathcal{A}_k x_k$$

  that given initial x_0 will give the same sequence $(x_k)_{k\in\mathbb{N}}$
- We will also see stochastic algorithms that iterate

$$x_{k+1} = \mathcal{A}_k(\xi_k) x_k$$

  where ξ_k is a random variable that also decides the mapping
  - $(x_k)_{k\in\mathbb{N}}$ is a stochastic process, i.e., collection of random variables
  - when running the algorithm, we evaluate ξ_k and get a realization
  - different realization $(x_k)_{k\in\mathbb{N}}$ every time even if started at same x_0
- Stochastic algorithms useful although problem is deterministic

Optimization algorithm overview

- Algorithms can roughly be divided into the following classes:
  - Second-order methods
  - Quasi second-order methods
  - First-order methods
  - Stochastic and coordinate-wise first-order methods
- The first three are typically deterministic and the last stochastic
- Cost of computing one iteration decreases down the list

Second-order methods

- Solves problems using second-order (Hessian) information
- Requires smooth (twice continuously differentiable) functions
- Example: Newton's method to minimize smooth function f:

$$x_{k+1} = x_k - \gamma_k(\nabla^2 f(x_k))^{-1}\nabla f(x_k)$$

- Constraints can be incorporated via barrier functions:
  - Use sequence of smooth constraint barrier functions
  - Make barriers increasingly well approximate constraint set
  - For each barrier, solve smooth problem using Newton's method
  - Resulting scheme called interior point method
  - (Can be applied to directly solve primal-dual optimality condition)
- Computational backbone: solving linear systems $O(n^3)$
- Often restricted to small to medium scale problems
- We will cover Newton's method

Quasi second-order methods

- Estimates second-order information from first-order
- Solves problems using estimated second-order information
- Requires smooth (twice continuously differentiable) functions
- Quasi-Newton method for smooth f

$$x_{k+1} = x_k - \gamma_k B_k \nabla f(x_k)$$

  where B_k is:
  - estimate of Hessian inverse (not Hessian to avoid inverse)
  - cheaply computed from gradient information
- Computational backbone: forming B_k and matrix multiplication
- Limited memory versions exist with cheaper iterations
- Can solve large-scale smooth problems
- Will briefly look into most common method (BFGS)

First-order methods

- Solves problems using first-order (sub-gradient) information
- Computational primitives: (sub)gradients and proximal operators
- Use gradient if function differentiable, prox if nondifferentiable
- Examples for solving $\underset{x}{\text{minimize}}\, f(x) + g(x)$
  - Proximal gradient method (requires smooth f since gradient used)

$$x_{k+1} = \text{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))$$

  - Douglas-Rachford splitting (no smoothness requirement)

$$z_{k+1} = \tfrac{1}{2}z_k + \tfrac{1}{2}(2\text{prox}_{\gamma g} - I)(2\text{prox}_{\gamma f} - I)z_k$$

    and $x_k = \text{prox}_{\gamma f}(z_k)$ converges to solution
- Iteration often cheaper than second-order if function split wisely
- Can solve large-scale problems
- Will look at proximal gradient method and accelerated version

Stochastic and coordinate-wise first-order methods

- Sometimes first-order methods computationally too expensive
- Stochastic gradient methods:
  - Use stochastic approximation of gradient
  - For finite sum problems, cheaply computed approximation exists
- Coordinate-wise updates:
  - Update only one (or block of) coordinates in every iteration:
    - via direct minimization
    - via proximal gradient step
  - Can update coordinates in cyclic fashion
  - Stronger convergence results if random selection of block
  - Efficient if cost of updating one coordinate is $1/n$ of full update
- Can solve huge scale problems
- Will cover randomized coordinate and stochastic methods

Outline

- Algorithm overview
- **Convergence and convergence rates**
- Proving convergence rates

Types of convergence

- Let $x^\star$ be solution to composite problem and $p^\star = f(x^\star) + g(x^\star)$
- We will see convergence of different quantities in different settings
- For deterministic algorithms that generate $(x_k)_{k\in\mathbb{N}}$, we will see
  - Sequence convergence: $x_k \to x^\star$
  - Function value convergence: $f(x_k) + g(x_k) \to p^\star$
  - If $g = 0$, gradient norm convergence: $\|\nabla f(x_k)\|_2 \to 0$
- Convergence is stronger as we go up the list
- First two common in convex setting, last in nonconvex

Convergence for stochastic algorithms

- Stochastic algorithms described by stochastic process $(x_k)_{k\in\mathbb{N}}$
- When algorithm is run, we get realization of stochastic process
- We analyze stochastic process and will see summability, e.g., of:
  - Expected distance to solution: $\sum_{k=0}^{\infty} \mathbb{E}[\|x_k - x^\star\|_2] < \infty$
  - Expected function value: $\sum_{k=0}^{\infty} \mathbb{E}[f(x_k) + g(x_k) - p^\star] < \infty$
  - If $g = 0$, expected gradient norm: $\sum_{k=0}^{\infty} \mathbb{E}[\|\nabla f(x_k)\|_2^2] < \infty$
- Sometimes arrive at weaker conclusion, when $g = 0$, that, e.g.,:
  - Expected smallest function value: $\mathbb{E}[\min_{l\in\{0,\ldots,k\}} f(x_l) - p^\star] \to 0$
  - Expected smallest gradient norm: $\mathbb{E}[\min_{l\in\{0,\ldots,k\}} \|\nabla f(x_l)\|_2] \to 0$
- Says what happens with expected value of different quantities

---

Algorithm realizations – Summable case

- Will conclude that sequence of expected values containing, e.g.,:

  $$\mathbb{E}[\|x_k - x^\star\|_2] \quad \text{or} \quad \mathbb{E}[f(x_k) + g(x_k) - p^\star] \quad \text{or} \quad \mathbb{E}[\|\nabla f(x_k)\|_2]$$

  is summable, where all quantities are nonnegative
- What happens with the actual algorithm realizations?
- We can make conclusions by the following result: If
  - $(Z_k)_{k\in\mathbb{N}}$ is a stochastic process with $Z_k \geq 0$
  - the sequence $\{\mathbb{E}[Z_k]\}_{k\in\mathbb{N}}$ is summable: $\sum_{k=0}^{\infty} \mathbb{E}[Z_k] < \infty$

  then almost sure convergence to 0:

  $$P(\lim_{k\to\infty} Z_k = 0) = 1$$

  i.e., convergence to 0 with probability 1

Algorithm realizations – Convergent case

- Will conclude that sequence of expected values containing, e.g.,:

  $$\mathbb{E}[\min_{l\in\{0,\ldots,k\}} f(x_l) - p^\star] \quad \text{or} \quad \mathbb{E}[\min_{l\in\{0,\ldots,k\}} \|\nabla f(x_l)\|_2]$$

  converges to 0, where all quantities are nonnegative
- What happens with the actual algorithm realizations?
- We can make conclusions by the following result: If
  - $(Z_k)_{k\in\mathbb{N}}$ is a stochastic process with $Z_k \geq 0$
  - the expected value $\mathbb{E}[Z_k] \to 0$ as $k \to \infty$

  then convergence to 0 in probability; for all $\epsilon > 0$

  $$\lim_{k\to\infty} P(Z_k > \epsilon) = 0$$

  which is weaker than almost sure convergence to 0

Convergence rates

- We have only talked about convergence, not convergence *rate*
- Rates indicate how fast (in iterations) algorithm reaches solution
- Typically divided into:
  - Sublinear rates
  - Linear rates (also called geometric rates)
  - Quadratic rates (or more generally superlinear rates)
- Sublinear rates slowest, quadratic rates fastest
- Linear rates further divided into Q-linear and R-linear
- Quadratic rates further divided into Q-quadratic and R-quadratic

---

Linear rates

- A Q-linear rate with factor $\rho \in [0, 1)$ can be:

  $$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \rho(f(x_k) + g(x_k) - p^\star)$$
  $$\mathbb{E}[\|x_{k+1} - x^\star\|_2] \leq \rho\mathbb{E}[\|x_k - x^\star\|_2]$$

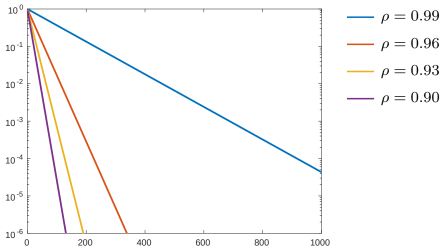
- An R-linear rate with factor $\rho \in [0, 1)$ and some $C > 0$ can be:

  $$\|x_k - x^\star\|_2 \leq \rho^k C$$

  this is implied by Q-linear rate and has *exponential decrease*
- Linear rate is superlinear if $\rho = \rho_k$ and $\rho_k \to 0$ as $k \to \infty$
- Examples:
  - (Accelerated) proximal gradient with strongly convex cost
  - Randomized coordinate descent with strongly convex cost
  - BFGS has *local* superlinear with strongly convex cost
  - but SGD with strongly convex cost gives sublinear rate

Linear rates – Comparison

- Different rates in log-lin plot



- Called linear rate since linear in log-lin plot

Quadratic rates

- Q-quadratic rate with factor $\rho \in [0, 1)$ can be:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \le \rho(f(x_k) + g(x_k) - p^\star)^2$$
$$\|x_{k+1} - x^\star\|_2 \le \rho\|x - x^\star\|_2^2$$

- R-quadratic rate with factor $\rho \in [0, 1)$ and some $C > 0$ can be:

$$\|x_k - x^\star\|_2 \le \rho^{2^k} C$$

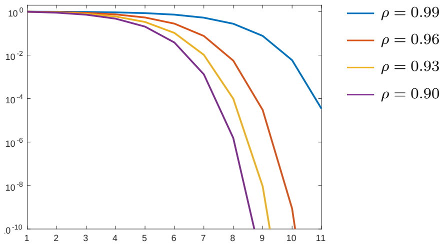
- Quadratic (ρ^{2^k}) vs linear (ρ^k) rate with factor $\rho = 0.9$:

Quadratic	Linear
1.000000000000	1.000000000000
0.810000000000	0.900000000000
0.656099945000	0.810000000000
0.430467133000	0.729000000000
0.185302002000	0.656099945000
0.034336821000	0.590490005000
0.001179017030	0.531440964000
0.000001390081	0.478296936000
0.000000000002	0.430467270000

- Example: *Locally* for Newton's method with strongly convex cost

Quadratic rates – Comparison

- Different rates in log-lin scale



- Quadratic convergence is superlinear

Sublinear rates

- A rate is sublinear if it is slower than linear
- A sublinear rate can, for instance, be of the form

$$f(x_k) + g(x_k) - p^\star \le \frac{C}{\psi(k)}$$
$$\|x_{k+1} - x_k\|_2^2 \le \frac{C}{\psi(k)}$$
$$\min_{l=0,\ldots,k} \mathbb{E}[\|\nabla f(x_l)\|_2^2] \le \frac{C}{\psi(k)}$$

where $C > 0$ and ψ decides how fast it decreases, e.g.,

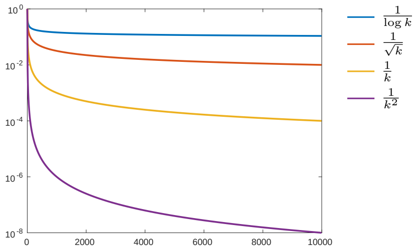
  - $\psi(k) = \log k$: Stochastic gradient descent $\gamma_k = c/k$
  - $\psi(k) = \sqrt{k}$: Stochastic gradient descent: optimal γ_k
  - $\psi(k) = k$: Proximal gradient, coordinate proximal gradient
  - $\psi(k) = k^2$: Accelerated proximal gradient method

with improved rate further down the list

- We say that the rate is $O(\frac{1}{\psi(k)})$ for the different ψ
- To be sublinear ψ has slower than exponential growth

Sublinear rates – Comparison

- Different rates on log-lin scale



- Many iterations may be needed for high accuracy

Rate vs iteration cost

- Consider these classes of algorithms
  - Second-order methods
  - Quasi second-order methods
  - First-order methods
  - Stochastic and coordinate-wise first-order methods
- Rate deteriorates and iterations increase as we go down the list $\Downarrow$
- Iteration cost increases as we go up the list $\Uparrow$
- Performance is roughly (# iterations)×(iteration cost)
- This gives a tradeoff when selecting algorithm
- Rough advise for problem size: small ($\Uparrow$) medium ($\Uparrow\Downarrow$) large ($\Downarrow$)

Outline

- Algorithm overview
- Convergence and convergence rates
- **Proving convergence rates**

Proving convergence rates

- To prove a convergence rate typically requires
  - Using inequalities that describe problem class
  - Using algorithm definition equalities (or inclusions)
  - Combine these to a form so that convergence can be concluded
- Linear and quadratic rates proofs conceptually straightforward
- Sublinear rates implicit via a *Lyapunov inequality*

Proving linear or quadratic rates

- If we suspect linear or quadratic convergence for $V_k \geq 0$:

$$V_{k+1} \leq \rho V_k^p$$

  where $\rho \in [0,1)$ and $p = 1$ or $p = 2$ and V_k can, e.g., be

$$V_k = \|x_k - x^\star\|_2 \quad \text{or} \quad V_k = f(x_k) + g(x_k) - p^\star \quad \text{or} \quad V_k = \|\nabla f(x_k)\|_2$$

- Can prove by starting with V_{k+1} (or V_{k+1}^2) and continue using
  - function class inequalities
  - algorithm equalities
  - propeties of norms
  - ...

Sublinear convergence – Lyapunov inequality

- Assume we want to show sublinear convergence of some $R_k \geq 0$
- This typically requires finding a *Lyapunov inequality*:

$$V_{k+1} \leq V_k + W_k - R_k$$

  where
  - $(V_k)_{k\in\mathbb{N}}$, $(W_k)_{k\in\mathbb{N}}$, and $(R_k)_{k\in\mathbb{N}}$ are nonnegative real numbers
  - $(W_k)_{k\in\mathbb{N}}$ is summable, i.e., $\overline{W} := \sum_{k=0}^{\infty} W_k < \infty$
- Such a Lyapunov inequality can be found by using
  - function class inequalities
  - algorithm equalities
  - propeties of norms
  - ...

Lyapunov inequality consequences

- From the Lyapunov inequality:

$$V_{k+1} \leq V_k + W_k - R_k$$

  we can conclude that
  - V_k is nonincreasing if all $W_k = 0$
  - V_k converges as $k \to \infty$ (will not prove)
- Recursively applying the inequality for $l \in \{k, \ldots, 0\}$ gives

$$V_{k+1} \leq V_0 + \sum_{l=0}^{k} W_l - \sum_{l=0}^{k} R_l \leq V_0 + \overline{W} - \sum_{l=0}^{k} R_l$$

  where $\overline{W}$ is infinite sum of W_k, this implies

$$\sum_{l=0}^{k} R_l \leq V_0 - V_{k+1} + \sum_{l=0}^{k} W_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

  from which we can
  - conclude that $R_k \to 0$ as $k \to \infty$ since $R_k \geq 0$
  - derive sublinear rates of convergence for R_k towards 0

Concluding sublinear convergence

- Lyapunov inequality consequence restated

$$\sum_{l=0}^{k} R_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

- We can derive sublinear convergence for
  - Best R_k: $(k+1)\min_{l\in\{0,\ldots,k\}} R_l \leq \sum_{l=0}^{k} R_l$
  - Last R_k (if R_k decreasing): $(k+1)R_k \leq \sum_{l=0}^{k} R_l$
  - Average R_k: $\bar{R}_k = \frac{1}{k+1}\sum_{l=0}^{k} R_l$
- Let $\hat{R}_k$ be any of these quantities, and we have

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{k+1} \leq \frac{V_0 + \overline{W}}{k+1}$$

  which shows a $O(1/k)$ sublinear convergence

Deriving other than $O(1/k)$ convergence (1/3)

- Other rates can be derived from a modified Lyapunov inequality:

$$V_{k+1} \leq V_k + W_k - \lambda_k R_k$$

  with $\lambda_k > 0$ when we are interested in convergence of R_k, then

$$\sum_{l=0}^{k} \lambda_l R_l \leq V_0 + \sum_{l=0}^{k} W_l \leq V_0 + \overline{W}$$

- We have $R_k \to 0$ as $k \to \infty$ if, e.g., $\sum_{l=0}^{\infty} \lambda_l = \infty$

Deriving other than $O(1/k)$ convergence (2/3)

- Restating the consequence: $\sum_{l=0}^{k} \lambda_l R_l \leq V_0 + \overline{W}$
- We can derive sublinear convergence for
  - Best R_k: $\min_{l\in\{0,\ldots,k\}} R_l \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l R_l$
  - Last R_k (if R_k decreasing): $R_k \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l R_l$
  - Weighted average R_k: $\bar{R}_k = \frac{1}{\sum_{l=0}^{k} \lambda_l} \sum_{l=0}^{k} \lambda_l R_l$
- Let $\hat{R}_k$ be any of these quantities, and we have

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \lambda_l} \leq \frac{V_0 + \overline{W}}{\sum_{l=0}^{k} \lambda_l}$$

Deriving other than $O(1/k)$ convergence (3/3)

- How to get a rate out of:

$$\hat{R}_k \leq \frac{V_0 + \overline{W}}{\sum_{l=0}^{k} \lambda_l}$$

- Assume $\psi(k) \leq \sum_{l=0}^{k} \lambda_l$, then $\psi(k)$ decides rate:

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \lambda_l} \leq \frac{V_0 + \overline{W}}{\psi(k)}$$

  which gives a $O(\frac{1}{\psi(k)})$ rate
  - If $\lambda_k = c$ is constant: $\psi(k) = c(k+1)$ and we have $O(1/k)$ rate
  - If λ_k is decreasing: slower rate than $O(1/k)$
  - If λ_k is increasing: faster rate than $O(1/k)$

Estimating ψ via integrals

- Assume that $\lambda_k = \phi(k)$, then $\psi(k) \leq \sum_{l=0}^{k} \phi(l)$ and

$$\hat{R}_k \leq \frac{\sum_{l=0}^{k} R_l}{\sum_{l=0}^{k} \phi(l)} \leq \frac{V_0 + \overline{W}}{\psi(k)}$$

- To estimate ψ, we use the integral inequalities
  - for decreasing nonnegative ϕ:

$$\int_{t=0}^{k} \phi(t)dt + \phi(k) \leq \sum_{l=0}^{k} \phi(l) \leq \int_{t=0}^{k} \phi(t)dt + \phi(0)$$

  - for increasing nonnegative ϕ:

$$\int_{t=0}^{k} \phi(t)dt + \phi(0) \leq \sum_{l=0}^{k} \phi(l) \leq \int_{t=0}^{k} \phi(t)dt + \phi(k)$$

- Remove $\phi(k), \phi(0) \geq 0$ from the lower bounds and use estimate:

$$\psi(k) = \int_{t=0}^{k} \phi(t)dt \leq \sum_{l=0}^{k} \phi(l)$$

Sublinear rate examples

- For Lyapunov inequality $V_{k+1} \leq V_k + W_k - \lambda_k R_k$, we get:

$$\hat{R}_k \leq \frac{V_0 + \overline{W}}{\psi(k)} \qquad \text{where} \qquad \lambda_k = \phi(k) \text{ and } \psi(k) = \int_{t=0}^{k} \phi(t)dt$$

- Let us quantify the rate ψ in a few examples:
  - Two examples that are slower than $O(1/k)$:
    - $\lambda_k = \phi(k) = c/(k+1)$ gives slow $O(\frac{1}{\log k})$ rate:

$$\psi(k) = \int_{t=0}^{k} \frac{c}{t+1} dt = c[\log(t+1)]_{t=0}^{k} = c\log(k+1)$$

    - $\lambda_k = \phi(k) = c/(k+1)^\alpha$ for $\alpha \in (0,1)$, gives faster $O(\frac{1}{k^{1-\alpha}})$ rate:

$$\psi(k) = \int_{t=0}^{k} \frac{c}{(t+1)^\alpha} dt = c[\tfrac{(t+1)^{1-\alpha}}{(1-\alpha)}]_{t=0}^{k} = \tfrac{c}{1-\alpha}((k+1)^{1-\alpha} - 1)$$

  - An example that is faster than $O(1/k)$
    - $\lambda_k = \phi(k) = c(k+1)$ gives $O(\frac{1}{k^2})$ rate:

$$\psi(k) = \int_{t=0}^{k} c(t+1)dt = c[\tfrac{1}{2}(t+1)^2]_{t=0}^{k} = \tfrac{c}{2}((k+1)^2 - 1)$$

Stochastic setting and law of total expectation

- In the stochastic setting, we analyze the stochastic process

$$x_{k+1} = \mathcal{A}_k(\xi_k)x_k$$

- We will look for inequalities of the form

$$\mathbb{E}[V_{k+1}|x_k] \leq \mathbb{E}[V_k|x_k] + \mathbb{E}[W_k|x_k] - \lambda_k \mathbb{E}[R_k|x_k]$$

  to see what happens in one step given x_k (but not given ξ_k)
- We use *law of total expectation* $\mathbb{E}[\mathbb{E}[X|Y]] = \mathbb{E}[X]$ to get

$$\mathbb{E}[V_{k+1}] \leq \mathbb{E}[V_k] + \mathbb{E}[W_k] - \lambda_k \mathbb{E}[R_k]$$

  which is a Lyapunov inequality
- We can draw rate conclusions, as we did before, now for $\mathbb{E}[R_k]$
- For realizations we can say:
  - If $\mathbb{E}[R_k]$ is summable, then $R_k \to 0$ almost surely
  - If $\mathbb{E}[R_k] \to 0$, then $R_k \to 0$ in probability

Rates in stochastic setting

- Lyapunov inequality $\mathbb{E}[V_{k+1}] \leq \mathbb{E}[V_k] + \mathbb{E}[W_k] - \lambda_k \mathbb{E}[R_k]$ implies:

$$\sum_{l=0}^{k} \lambda_l \mathbb{E}[R_l] \leq V_0 + \sum_{l=0}^{\infty} \mathbb{E}[W_l] \leq V_0 + \bar{W}$$

- Same procedure as before gives sublinear rates for
  - Best $\mathbb{E}[R_k]$: $\min_{l\in\{0,\ldots,k\}} \mathbb{E}[R_l] \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l \mathbb{E}[R_l]$
  - Last $\mathbb{E}[R_k]$ (if $\mathbb{E}[R_k]$ decreasing): $\mathbb{E}[R_k] \sum_{l=0}^{k} \lambda_l \leq \sum_{l=0}^{k} \lambda_l \mathbb{E}[R_l]$
  - Weighted average: $\mathbb{E}[\bar{R}_k] = \frac{1}{\sum_{l=0}^{k} \lambda_l} \sum_{l=0}^{k} \lambda_l \mathbb{E}[R_l]$
- Jensen's inequality for concave $\min_l$ in best residual reads

$$\mathbb{E}[\min_{l\in\{0,\ldots,k\}} R_l] \leq \min_{l\in\{0,\ldots,k\}} \mathbb{E}[R_l]$$

- Let $\hat{R}_k$ be any of the above quantities, and we have

$$\mathbb{E}[\hat{R}_k] \leq \frac{V_0 + \bar{W}}{\sum_{l=0}^{k} \lambda_l}$$

Proximal Gradient Method

Pontus Giselsson

Outline

- **A fundamental inequality**
- Nonconvex setting
- Convex setting
- Strongly convex setting
- Backtracking
- Stopping conditions
- Accelerated gradient method
- Scaling

Proximal gradient method

- We consider composite optimization problems of the form

$$\underset{x}{\text{minimize}}\, f(x) + g(x)$$

- The proximal gradient method is

$$\begin{aligned} x_{k+1} &= \underset{y}{\text{argmin}} \Big(f(x_k) + \nabla f(x_k)^T(y - x_k) + \tfrac{1}{2\gamma_k}\|y - x_k\|_2^2 + g(y) \Big) \\ &= \underset{y}{\text{argmin}} \Big(g(y) + \tfrac{1}{2\gamma_k}\|y - (x_k - \gamma_k \nabla f(x_k))\|_2^2 \Big) \\ &= \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k)) \end{aligned}$$

Proximal gradient – Optimality condition

- Proximal gradient iteration is:

$$\begin{aligned} x_{k+1} &= \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k)) \\ &= \underset{y}{\text{argmin}}(g(y) + \underbrace{\tfrac{1}{2\gamma_k}\|y - (x_k - \gamma_k \nabla f(x_k))\|_2^2)}_{h(y)} \end{aligned}$$

  where x_{k+1} is unique due to strong convexity of h
- Fermat's rule gives, since g convex, optimality condition:

$$\begin{aligned} 0 &\in \partial g(x_{k+1}) + \partial h(x_{k+1}) \\ &= \partial g(x_{k+1}) + \gamma_k^{-1}(x_{k+1} - (x_k - \gamma_k \nabla f(x_k))) \end{aligned}$$

  since h differentiable
- A consequence is that $\partial g(x_{k+1})$ is nonempty

Proximal gradient method – Convergence rates

- We will analyze proximal gradient method in different settings:
  - Nonconvex
    - $O(1/k)$ convergence for squared residual
  - Convex
    - $O(1/k)$ convergence for function values
  - Strongly convex
    - Linear convergence in distance to solution
- First two rates based on a *fundamental inequality* for the method

Assumptions for fundamental inequality

(i) $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable (not necessarily convex)

(ii) For every x_k and x_{k+1} there exists $\beta_k \in [\eta, \eta^{-1}]$, $\eta \in (0, 1]$:

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_k - x_{k+1}\|_2^2$$

where β_k is a sort of local Lipschitz constant

(iii) $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed convex

(iv) A minimizer $x^\star$ exists and $p^\star = f(x^\star) + g(x^\star)$ is optimal value

(v) Proximal gradient method parameters $\gamma_k > 0$

- Assumption (ii) satisfied with $\beta_k \geq \beta$ if f is β-smooth
- Assumptions will be strengthened later

A fundamental inequality

For all $z \in \mathbb{R}^n$, the proximal gradient method satisfies

$$\begin{aligned} f(x_{k+1}) + g(x_{k+1}) \leq f(x_k) &+ \nabla f(x_k)^T(z - x_k) - \tfrac{\gamma_k^{-1} - \beta_k}{2}\|x_{k+1} - x_k\|_2^2 \\ &+ g(z) + \tfrac{1}{2\gamma_k}(\|x_k - z\|_2^2 - \|x_{k+1} - z\|_2^2) \end{aligned}$$

where $x_{k+1} = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$

A fundamental inequality – Proof (1/2)

Using

(a) Upper bound assumption on f, i.e., Assumption (ii)

(b) Prox optimality condition: There exists $s_{k+1} \in \partial g(x_{k+1})$

$$0 = s_{k+1} + \gamma_k^{-1}(x_{k+1} - (x_k - \gamma_k \nabla f(x_k)))$$

(c) Subgradient definition: $\forall z, g(z) \geq g(x_{k+1}) + s_{k+1}^T(z - x_{k+1})$

$$\begin{aligned} &f(x_{k+1}) + g(x_{k+1}) \\ &\quad \overset{(a)}{\leq} f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(x_{k+1}) \\ &\quad \overset{(c)}{\leq} f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(z) \\ &\quad\quad - s_{k+1}^T(z - x_{k+1}) \\ &\quad \overset{(b)}{=} f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(z) \\ &\quad\quad + \gamma_k^{-1}(x_{k+1} - (x_k - \gamma_k \nabla f(x_k)))^T(z - x_{k+1}) \\ &\quad = f(x_k) + \nabla f(x_k)^T(z - x_k) + \tfrac{\beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(z) \\ &\quad\quad + \gamma_k^{-1}(x_{k+1} - x_k)^T(z - x_{k+1}) \end{aligned}$$

A fundamental inequality – Proof (2/2)

- The proof continues by using the equality

$$
\begin{aligned}
&(x_{k+1}-x_k)^T(z-x_{k+1})\\
&\qquad= \tfrac{1}{2}(\|x_k-z\|_2^2-\|x_{k+1}-z\|_2^2-\|x_{k+1}-x_k\|_2^2)
\end{aligned}
$$

- Applying to previous inequality gives

$$
\begin{aligned}
&f(x_{k+1})+g(x_{k+1})\\
&\qquad\le f(x_k)+\nabla f(x_k)^T(z-x_k)+\tfrac{\beta_k}{2}\|x_{k+1}-x_k\|_2^2+g(z)\\
&\qquad\quad+\gamma_k^{-1}(x_{k+1}-x_k)^T(z-x_{k+1})\\
&\qquad= f(x_k)+\nabla f(x_k)^T(z-x_k)+\tfrac{\beta_k}{2}\|x_{k+1}-x_k\|_2^2+g(z)\\
&\qquad\quad+\tfrac{1}{2\gamma_k}(\|x_k-z\|_2^2-\|x_{k+1}-z\|_2^2-\|x_k-x_{k+1}\|_2^2)
\end{aligned}
$$

which after rearrangement gives the fundamental inequality

Outline

- A fundamental inequality
- **Nonconvex setting**
- Convex setting
- Strongly convex setting
- Backtracking
- Stopping conditions
- Accelerated gradient method
- Scaling

Nonconvex setting

- We will analyze the proximal gradient method

$$x_{k+1}=\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))$$

in a nonconvex setting for solving

$$\mathrm{minimize}\, f(x)+g(x)$$

- Will show sublinear $O(1/k)$ convergence
- Analysis based on *A fundamental inequality*

Nonconvex setting – Assumptions

(i) $f:\mathbb{R}^n\to\mathbb{R}$ is continuously differentiable (not necessarily convex)

(ii) For every x_k and x_{k+1} there exists $\beta_k\in[\eta,\eta^{-1}]$, $\eta\in(0,1]$:

$$f(x_{k+1})\le f(x_k)+\nabla f(x_k)^T(x_{k+1}-x_k)+\tfrac{\beta_k}{2}\|x_k-x_{k+1}\|_2^2$$

where β_k is a sort of local Lipschitz constant

(iii) $g:\mathbb{R}^n\to\mathbb{R}\cup\{\infty\}$ is closed convex

(iv) A minimizer $x^\star$ exists and $p^\star=f(x^\star)+g(x^\star)$ is optimal value

(v) Algorithm parameters $\gamma_k\in[\epsilon,\frac{2}{\beta_k}-\epsilon]$, where $\epsilon>0$

- Differs from assumptions for fundamental inequality only in (v)
- Assumption (ii) satisfied with $\beta_k\ge\beta$ if f is β-smooth

Nonconvex setting – Analysis

- Use fundamental inequality

$$
\begin{aligned}
f(x_{k+1})+g(x_{k+1})\le f(x_k)+\nabla f(x_k)^T(z-x_k)-\tfrac{\gamma_k^{-1}-\beta_k}{2}\|x_{k+1}-x_k\|_2^2\\
+g(z)+\tfrac{1}{2\gamma_k}(\|x_k-z\|_2^2-\|x_{k+1}-z\|_2^2)
\end{aligned}
$$

- Set $z=x_k$ to get

$$f(x_{k+1})+g(x_{k+1})\le f(x_k)+g(x_k)-(\gamma_k^{-1}-\tfrac{\beta_k}{2})\|x_{k+1}-x_k\|_2^2$$

Step-size requirements

- Step-sizes γ_k should be restricted for inequality to be useful:

$$f(x_{k+1})+g(x_{k+1})\le f(x_k)+g(x_k)-(\gamma_k^{-1}-\tfrac{\beta_k}{2})\|x_{k+1}-x_k\|_2^2$$

- Requirements $\beta_k\in[\eta,\eta^{-1}]$ and $\gamma_k\in[\epsilon,\frac{2}{\beta_k}-\epsilon]$:
  - upper bound $\gamma_k\le\frac{2}{\beta_k}-\epsilon$ can be written as

$$\gamma_k\le\tfrac{2}{\beta_k+2\delta_k}\qquad\text{where}\qquad\delta_k=\tfrac{\beta_k\epsilon}{2\left(\frac{2}{\beta_k}-\epsilon\right)}\ge\tfrac{\beta_k^2\epsilon}{4}\ge\tfrac{\eta^2\epsilon}{4}>0$$

    since upper bound $\beta_k\le\eta^{-1}$ gives $\frac{2}{\beta_k}-\epsilon\ge2\eta-\epsilon>0$ and $\epsilon>0$

  - Inverting upper step-size bound and letting $\delta:=\frac{\eta^2\epsilon}{4}\le\delta_k$:

$$\gamma_k^{-1}\ge\tfrac{\beta_k+2\delta_k}{2}\ge\tfrac{\beta_k}{2}+\delta\qquad\Rightarrow\qquad\gamma_k^{-1}-\tfrac{\beta_k}{2}\ge\delta>0$$

- This implies, by subtracting $p^\star$ from both sides to have $V_k\ge0$,

$$\underbrace{f(x_{k+1})+g(x_{k+1})-p^\star}_{V_{k+1}}\le\underbrace{f(x_k)+g(x_k)-p^\star}_{V_k}-\underbrace{\delta\|x_{k+1}-x_k\|_2^2}_{R_k}$$

where bounds on γ_k imply that all R_k are nonnegative

Lyapunov inequality consequences

- Restating Lyapunov inequality

$$\underbrace{f(x_{k+1})+g(x_{k+1})-p^\star}_{V_{k+1}}\le\underbrace{f(x_k)+g(x_k)-p^\star}_{V_k}-\underbrace{\delta\|x_{k+1}-x_k\|_2^2}_{R_k}$$

- Consequences:
  - Function value is decreasing sequence (may not converge to $p^\star$)
  - Fixed-point residual converges to 0 as $k\to\infty$:

$$\|x_{k+1}-x_k\|_2=\|\mathrm{prox}_{\gamma_k g}(x_k-\gamma_k\nabla f(x_k))-x_k\|_2\to0$$

  - Best fixed-point residual norm square converges as $O(1/k)$:

$$\min_{i\in\{0,\ldots,k\}}\|x_{i+1}-x_i\|_2^2\le\frac{f(x_0)+g(x_0)-p^\star}{\delta(k+1)}$$

Lyapunov inequality consequences – $g=0$

- For $g=0$, then $x_{k+1}=x_k-\gamma_k\nabla f(x_k)$ and

$$\|x_{k+1}-x_k\|_2=\gamma_k\|\nabla f(x_k)\|_2\qquad\text{and}\qquad R_k=\delta\gamma_k^2\|\nabla f(x_k)\|_2^2$$

- Lyapunov inequality consequences in this setting:
  - Gradient converges to 0 (since $\gamma_k\ge\epsilon$): $\|\nabla f(x_k)\|_2\to0$
  - Smallest gradient norm square converges as:

$$\min_{i\in\{0,\ldots,k\}}\|\nabla f(x_i)\|_2^2\le\frac{f(x_0)-p^\star}{\delta\sum_{i=0}^k\gamma_i^2}$$

  - If, in addition, f is β-smooth and $\gamma_k=\frac{1}{\beta}$:

$$\min_{i\in\{0,\ldots,k\}}\|\nabla f(x_i)\|_2^2\le\frac{2\beta(f(x_0)-p^\star)}{k+1}$$

    since then $\beta_k=\beta$ and $\gamma_k^{-1}-\frac{\beta_k}{2}=\frac{\beta}{2}=\delta>0$

- So, will approach local maximum, minimum, or saddle-point

Fixed-point residual convergence – Implication

What does $\|\mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k)) - x_k\|_2 \to 0$ imply?

- By prox-grad optimality condition and $\|x_{k+1} - x_k\|_2 \to 0$:

$$\partial g(x_{k+1}) + \nabla f(x_k) \ni \gamma_k^{-1}(x_k - x_{k+1}) \to 0$$

  as $k \to \infty$ (since $\gamma_k \geq \epsilon$, i.e., $0 < \gamma_k^{-1} \leq \epsilon^{-1}$) or equivalently

$$\partial g(x_{k+1}) + \nabla f(x_{k+1}) \ni \underbrace{\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k)}_{u_k} \to 0$$

  where $u_k \to 0$ is concluded by continuity of ∇f
- Critical point definition for nonconvex f satisfied in the limit

Outline

- A fundamental inequality
- Nonconvex setting
- **Convex setting**
- Strongly convex setting
- Backtracking
- Stopping conditions
- Accelerated gradient method
- Scaling

Convex setting

- We will analyze the proximal gradient method

$$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$$

  in the convex setting for solving

$$\mathrm{minimize}\, f(x) + g(x)$$

- Will show sublinear $O(1/k)$ convergence for function values
- Analysis based on *A fundamental inequality*

Convex setting – Assumptions

(i) $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and convex

(ii) For every x_k and x_{k+1} there exists $\beta_k \in [\eta, \eta^{-1}]$, $\eta \in (0, 1]$:

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_k - x_{k+1}\|_2^2$$

where β_k is a sort of local Lipschitz constant

(iii) $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed convex

(iv) A minimizer $x^\star$ exists and $p^\star = f(x^\star) + g(x^\star)$ is optimal value

(v) Algorithm parameters $\gamma_k \in [\epsilon, \frac{2}{\beta_k} - \epsilon]$, where $\epsilon > 0$

- Assumptions as for fundamental inequality plus
  - convexity of f
  - restricted step-size parameters γ_k (as in nonconvex setting)
- Assumption (ii) satisfied with $\beta_k \geq \beta$ if f is β-smooth

Convex setting – Analysis

- Use fundamental inequality with $z = x^\star$, where $x^\star$ is solution

$$\begin{aligned} f(x_{k+1}) + g(x_{k+1}) \leq{}& f(x_k) + \nabla f(x_k)^T(x^\star - x_k) \\ &- \tfrac{\gamma_k^{-1} - \beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(x^\star) \\ &+ \tfrac{1}{2\gamma_k}(\|x_k - x^\star\|_2^2 - \|x_{k+1} - x^\star\|_2^2) \end{aligned}$$

- and convexity of f

$$f(x^\star) \geq f(x_k) + \nabla f(x_k)^T(x^\star - x_k)$$

- This gives

$$\begin{aligned} f(x_{k+1}) + g(x_{k+1}) \leq{}& f(x^\star) - \tfrac{\gamma_k^{-1} - \beta_k}{2}\|x_{k+1} - x_k\|_2^2 + g(x^\star) \\ &+ \tfrac{1}{2\gamma_k}(\|x_k - x^\star\|_2^2 - \|x_{k+1} - x^\star\|_2^2) \end{aligned}$$

  which, by multiplying by $2\gamma_k$ and using $p^\star = f(x^\star) + g(x^\star)$, gives

$$\begin{aligned} \|x_{k+1} - x^\star\|_2^2 \leq{}& \|x_k - x^\star\|_2^2 + (\beta_k\gamma_k - 1)\|x_{k+1} - x_k\|_2^2 \\ &- 2\gamma_k(f(x_{k+1}) + g(x_{k+1}) - p^\star) \end{aligned}$$

Lyapunov inequality – Convex setting

- The last inequality on previous slide is Lyapunov inequality

$$\begin{aligned} \underbrace{\|x_{k+1} - x^\star\|_2^2}_{V_{k+1}} \leq{}& \underbrace{\|x_k - x^\star\|_2^2}_{V_k} + \underbrace{(\beta_k\gamma_k - 1)\|x_{k+1} - x_k\|_2^2}_{W_k} \\ &- 2\gamma_k \underbrace{(f(x_{k+1}) + g(x_{k+1}) - p^\star)}_{R_k} \end{aligned}$$

- Will divide analysis two cases: Short and long step-sizes
  - Step-sizes $\gamma_k \in [\epsilon, \frac{1}{\beta_k}]$: gives $\beta_k\gamma_k \leq 1$ and $W_k \leq 0$
  - Step-sizes $\gamma_k \in [\frac{1}{\beta_k}, \frac{2}{\beta_k} - \epsilon]$: gives $\beta_k\gamma_k \geq 1$ and $W_k \geq 0$

  since W_k contribute differently

Short step-sizes

- For step-sizes $\gamma_k \in [\epsilon, \frac{1}{\beta_k}]$, the Lyapunov inequality implies:

$$\underbrace{\|x_{k+1} - x^\star\|_2^2}_{V_{k+1}} \leq \underbrace{\|x_k - x^\star\|_2^2}_{V_k} - 2\gamma_k \underbrace{(f(x_{k+1}) + g(x_{k+1}) - p^\star)}_{R_k}$$

  where we have used $W_k = 0$ (which is OK since $W_k \leq 0$)
- Nonconvex analysis says function value decreases in every iteration
- Consequences:
  - Distance to solution $\|x_k - x^\star\|_2$ converges as $k \to \infty$
  - Function value decreases to optimal function value as:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \frac{\|x_0 - x^\star\|_2^2}{2\sum_{i=0}^k \gamma_i}$$

    if f is β-smooth and $\gamma_k = \frac{1}{\beta}$, then converges as $O(1/k)$:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \frac{\beta\|x_0 - x^\star\|_2^2}{2(k+1)}$$

Long step-sizes

- For step-sizes $\gamma_k \in [\frac{1}{\beta_k}, \frac{2}{\beta_k} - \epsilon]$, the Lyapunov inequality is:

$$\begin{aligned} \underbrace{\|x_{k+1} - x^\star\|_2^2}_{V_{k+1}} \leq{}& \underbrace{\|x_k - x^\star\|_2^2}_{V_k} + \underbrace{(\beta_k\gamma_k - 1)\|x_{k+1} - x_k\|_2^2}_{W_k} \\ &- 2\gamma_k \underbrace{(f(x_{k+1}) + g(x_{k+1}) - p^\star)}_{R_k} \end{aligned}$$

- From nonconvex analysis can conclude that W_k is summable
  - We showed for $\gamma_k \in [\epsilon, \frac{2}{\beta_k} - \epsilon]$, $(\|x_{k+1} - x_k\|_2^2)_{k \in \mathbb{N}}$ is summable
  - Since $\beta_k\gamma_k$ bounded, also $(W_k)_{k \in \mathbb{N}}$ is summable
  - Let us define $\overline{W} = \sum_{k=0}^\infty W_k$
- Consequences:
  - Distance to solution $\|x_k - x^\star\|_2$ converges as $k \to \infty$
  - Function value decreases to optimal function value as:

$$f(x_{k+1}) + g(x_{k+1}) - p^\star \leq \frac{\|x_0 - x^\star\|_2^2 + \overline{W}}{2\sum_{i=0}^k \gamma_i}$$

    for β-smooth f with $\gamma_k = \frac{1}{\beta}$, denominator replaced by $\frac{2(k+1)}{\beta}$

Outline

- A fundamental inequality
- Nonconvex setting
- Convex setting
- **Strongly convex setting**
- Backtracking
- Stopping conditions
- Accelerated gradient method
- Scaling

Strongly convex setting

- We will analyze the proximal gradient method

$$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$$

in a strongly convex setting for solving

$$\mathrm{minimize}\, f(x) + g(x)$$

- Will show linear convergence for distance to solution $\|x_k - x^\star\|_2$
- Two ways to show linear convergence, we can:
  - (i) Base analysis on *A fundamental inequality*
  - (ii) Start by $\|x_{k+1} - x^\star\|_2^2$ and expand (which is what we will do)

Strongly convex setting – Assumptions

(i) $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and σ-strongly convex
(ii) f is β-smooth
(iii) $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed convex
(iv) A minimizer $x^\star$ exists and $p^\star = f(x^\star) + g(x^\star)$ is optimal value
(v) Algorithm parameters $\gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$, where $\epsilon > 0$

- Assumptions as for fundamental inequality plus
  - σ-strong convexity of f
  - β-smoothness of f instead of upper bound for x_{k+1} and x_k
  - restricted step-size parameters γ_k (as in (non)convex setting)
- But will not use fundamental inequality in analysis

Strongly convex setting – Analysis

Use that

(a) $x^\star = \mathrm{prox}_{\gamma g}(x^\star - \gamma \nabla f(x^\star))$ for all $\gamma > 0$
(b) the proximal operator is nonexpansive
(c) gradients of β-smooth σ-strongly convex functions f satisfy

$$(\nabla f(x) - \nabla f(y))^T(x - y) \geq \tfrac{1}{\beta+\sigma}\|\nabla f(x) - \nabla f(y)\|_2^2 + \tfrac{\sigma\beta}{\beta+\sigma}\|x - y\|_2^2$$

to get

$$\begin{aligned}
&\|x_{k+1} - x^\star\|_2^2 \\
&\overset{(a)}{=} \|\mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k)) - \mathrm{prox}_{\gamma_k g}(x^\star - \gamma_k \nabla f(x^\star))\|_2^2 \\
&\overset{(b)}{\leq} \|(x_k - \gamma_k \nabla f(x_k)) - (x^\star - \gamma_k \nabla f(x^\star))\|_2^2 \\
&= \|x_k - x^\star\|_2^2 - 2\gamma_k(\nabla f(x_k) - \nabla f(x^\star))^T(x_k - x^\star) \\
&\quad + \gamma_k^2\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2 \\
&\overset{(c)}{\leq} \|x_k - x^\star\|_2^2 - \tfrac{2\gamma_k}{\beta+\sigma}(\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2 + \sigma\beta\|x_k - x^\star\|_2^2) \\
&\quad + \gamma_k^2\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2 \\
&= (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma})\|x_k - x^\star\|_2^2 - \gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k)\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2
\end{aligned}$$

Lyapunov inequality – Strongly convex setting

- Lyapunov inequality from previous slide is

$$\begin{aligned}\|x_{k+1} - x^\star\|_2^2 \leq{}& (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma})\|x_k - x^\star\|_2^2 \\ &- \underbrace{\gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k)\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2}_{W_k}\end{aligned}$$

- Will divide analysis into two cases: Short and long step-sizes
  - Step-sizes $\gamma_k \in [\epsilon, \frac{2}{\beta+\sigma}]$: gives $W_k \geq 0$
  - Step-sizes $\gamma_k \in [\frac{2}{\beta+\sigma}, \frac{2}{\beta} - \epsilon]$: gives $W_k \leq 0$

Short step-sizes

- Lyapunov inequality

$$\begin{aligned}\|x_{k+1} - x^\star\|_2^2 \leq{}& (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma})\|x_k - x^\star\|_2^2 \\ &- \underbrace{\gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k)\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2}_{W_k}\end{aligned}$$

for $\gamma_k \in [\epsilon, \frac{2}{\beta+\sigma}]$ implies $W_k \geq 0$

- Strong monotonicity with modulus σ of ∇f implies

$$\|\nabla f(x_k) - \nabla f(x^\star)\|_2 \geq \sigma\|x_k - x^\star\|_2$$

- So we have linear convergence since

$$\begin{aligned}\|x_{k+1} - x^\star\|_2^2 &\leq (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma} - \sigma^2\gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k))\|x_k - x^\star\|_2^2 \\ &= (1 - \tfrac{2\gamma_k\sigma(\beta+\sigma)}{\beta+\sigma} + \sigma^2\gamma_k^2)\|x_k - x^\star\|_2^2 \\ &= (1 - \sigma\gamma_k)^2\|x_k - x^\star\|_2^2\end{aligned}$$

where $(1 - \sigma\gamma_k)^2 \in [0, 1)$ for full range of γ_k

Long step-sizes

- Lyapunov inequality

$$\begin{aligned}\|x_{k+1} - x^\star\|_2^2 \leq{}& (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma})\|x_k - x^\star\|_2^2 \\ &- \underbrace{\gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k)\|\nabla f(x_k) - \nabla f(x^\star)\|_2^2}_{W_k}\end{aligned}$$

for $\gamma_k \in [\frac{2}{\beta+\sigma}, \frac{2}{\beta} - \epsilon]$ implies $W_k \leq 0$

- That f is β-smooth implies ∇f is β-Lipschitz continuous:

$$\|\nabla f(x_k) - \nabla f(x^\star)\|_2 \leq \beta\|x_k - x^\star\|_2$$

- So we have linear convergence since

$$\begin{aligned}\|x_{k+1} - x^\star\|_2^2 &\leq (1 - \tfrac{2\gamma_k\sigma\beta}{\beta+\sigma} - \beta^2\gamma_k(\tfrac{2}{\beta+\sigma} - \gamma_k))\|x_k - x^\star\|_2^2 \\ &= (1 - \tfrac{2\gamma_k\beta(\sigma+\beta)}{\beta+\sigma} + \beta^2\gamma_k^2)\|x_k - x^\star\|_2^2 \\ &= (1 - \beta\gamma_k)^2\|x_k - x^\star\|_2^2\end{aligned}$$

where $(1 - \beta\gamma_k)^2 \in [0, 1)$ for full range of γ_k

Unified rate

- By removing the square and checking sign, we have
  - for step-sizes $\gamma_k \in [\epsilon, \frac{2}{\beta+\sigma}]$:

$$\|x_{k+1} - x^\star\|_2 \leq (1 - \sigma\gamma_k)\|x_k - x^\star\|_2$$

  - for step-sizes $\gamma_k \in [\frac{2}{\beta+\sigma}, \frac{2}{\beta} - \epsilon]$:

$$\|x_{k+1} - x^\star\|_2 \leq (\beta\gamma_k - 1)\|x_k - x^\star\|_2$$

- The linear convergence result can be summarized as

$$\|x_{k+1} - x^\star\|_2 \leq \max(1 - \sigma\gamma_k, \beta\gamma_k - 1)\|x_k - x^\star\|_2$$

Optimal step-size

- For fixed-step-sizes $\gamma_k = \gamma$, the rate result is

$$\|x_{k+1} - x^\star\|_2 \leq \underbrace{\max(1 - \sigma\gamma, \beta\gamma - 1)}_{\rho} \|x_k - x^\star\|_2$$

- Optimal γ that gives smallest contraction is $\gamma = \frac{2}{\beta+\sigma}$:
  - $(1 - \sigma\gamma)$ decreasing in γ, optimal at upper bound $\gamma = \frac{2}{\beta+\sigma}$
  - $(\beta\gamma - 1)$ increasing in γ, optimal at lower bound $\gamma = \frac{2}{\beta+\sigma}$
  - Bounds coincide at $\gamma = \frac{2}{\beta+\sigma}$ to give rate factor $\rho = \frac{\beta-\sigma}{\beta+\sigma}$

Outline

- A fundamental inequality
- Nonconvex setting
- Convex setting
- Strongly convex setting
- **Backtracking**
- Stopping conditions
- Accelerated gradient method
- Scaling

Choose β_k and γ_k

- In nonconvex and convex analysis, we assume β_k known such that

$$f(x_{k+1}) \leq f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta_k}{2}\|x_k - x_{k+1}\|_2^2$$

  for consecutive iterates x_k and x_{k+1}
- This is an assumption on the function f
- We call it *descent condition* (DC)
- If f is β-smooth, then $\beta_k = \beta$ is valid choice since

$$f(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|x - y\|_2^2$$

  for all x, y, then we can select $\gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$

Choose β_k and γ_k – Backtracking

- Backtracking: choose $\kappa > 1$, $\beta_{k,0} \in [\eta, \eta^{-1}]$, let $l_k = 0$, and loop

```
1. choose $\gamma_k \in [\epsilon, \frac{2}{\beta_{k,l_k}} - \epsilon]$
2. compute $x_{k+1} = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$
3. if descent condition (DC) satisfied
        set $k \leftarrow k + 1$      // increment algorithm counter
        set $\bar{l}_k \leftarrow l_k$        // store final backtrack counter
        set $\beta_k \leftarrow \beta_{k,l_k}$      // store final $\beta$ variable
        break backtrack loop
   else
        set $\beta_{k,l_k+1} \leftarrow \kappa\beta_{k,l_k}$  // increase backtrack parameter
        set $l_k \leftarrow l_k + 1$       // increment backtrack counter
   end
```

- Larger β_{k,l_k} gives smaller upper bound for step-size γ_k
- Forwardtracking on β_{k,l_k}, backtracking for γ_k upper bound

When to use backtracking

- f is β-smooth but constant β unknown:
  - initialize $\beta_{k,0} = \beta_{k-1,\bar{l}_{k-1}}$ to previously used value
  - then $(\beta_k)_{k\in\mathbb{N}}$ nondecreasing
  - finally $\beta_k \geq \beta$ (if needed), then
    - step-size bound $\gamma_k \in [\epsilon, \frac{2}{\beta_{k,\bar{l}_k}} - \epsilon]$ makes (DC) hold directly
    - so will have constant β_k after finite number of algoritm iterations
- ∇f locally Lipschitz and sequence bounded (as in convex case):
  - initialize $\beta_{k,0} = \bar{\beta}$, for some pre-chosen $\bar{\beta} > 0$
  - reset to same value $\bar{\beta}$ in every algorithm iteration
  - will find a local Lipschitz constant

Outline

- A fundamental inequality
- Nonconvex setting
- Convex setting
- Strongly convex setting
- Backtracking
- **Stopping conditions**
- Accelerated gradient method
- Scaling

When to stop algorithm?

- Consider $\underset{x}{\text{minimize}}\, f(x) + g(x)$
- Apply proximal gradient method $x_{k+1} = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$
- Algorithm sequence satisfies

$$\partial g(x_{k+1}) + \nabla f(x_{k+1}) \ni \underbrace{\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k)}_{u_k} \to 0$$

  is $\|u_k\|_2$ small a good measure of being close to fixed-point?

When to stop algorithm – Scaled problem

Let $a > 0$ and solve equivalent problem $\underset{x}{\text{minimize}}\, af(x) + ag(x)$:

- Denote algorithm parameter $\gamma_{a,k} = \frac{\gamma_k}{a}$
- Algorithm satisfies:

$$x_{k+1} = \text{prox}_{\gamma_{a,k} ag}(x_k - \gamma_{a,k} \nabla af(x_k)) = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$$

  i.e., the same algorithm as before
- However, $u_{a,k}$ in this setting satisfies

$$\begin{aligned} u_{a,k} &= \gamma_{a,k}^{-1}(x_k - x_{k+1}) + \nabla af(x_{k+1}) - \nabla af(x_k) \\ &= a(\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k)) \\ &= au_k \end{aligned}$$

  i.e., same algorithm but different optimality measure
- Optimality measure should be scaling invariant

Scaling invariant stopping condition

- For β-smooth f, use scaled condition $\frac{1}{\beta}u_k$

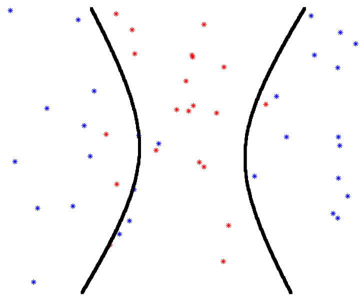
$$\tfrac{1}{\beta}u_k := \tfrac{1}{\beta}(\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k))$$

  that we have seen before
- Let us scale problem by a to get $\operatorname{minimize} af(x) + ag(x)$, then
  - smoothness constant $\beta_a = a\beta$ scaled by $a \Rightarrow$ use $\gamma_{a,k} = \frac{\gamma_k}{a}$
  - optimality measure $\frac{1}{\beta_a}u_{a,k} = \frac{1}{a\beta}au_k = \frac{1}{\beta}u_k$ remains the same

  so it is scaling invariant
- Problem considered solved to optimality if, say, $\frac{1}{\beta}\|u_k\|_2 \leq 10^{-6}$
- Often lower accuracy 10^{-3} to 10^{-4} is enough

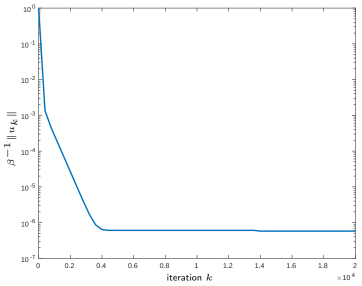
Example – SVM

- Classification problem from SVM lecture, SVM with
  - polynomial features of degree 2
  - regularization parameter $\lambda = 0.00001$



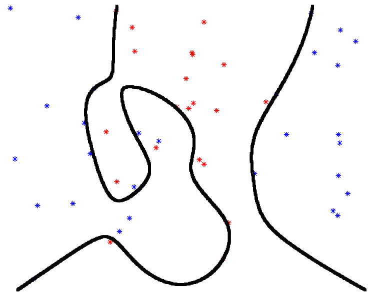
Example – Optimality measure

- Plots $\beta^{-1}\|u_k\|_2 = \beta^{-1}\|\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k)\|_2$
- Shows $\beta^{-1}\|u_k\|_2$ up to 20'000 iterations
- Quite many iterations needed to converge



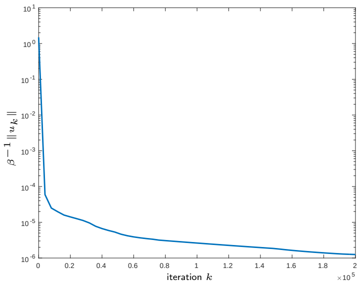
Example – SVM higher degree polynomial

- Classification problem from SVM lecture, SVM with
  - polynomial features of degree 6
  - regularization parameter $\lambda = 0.00001$



Example – Optimality measure

- Plots $\beta^{-1}\|u_k\|_2 = \beta^{-1}\|\gamma_k^{-1}(x_k - x_{k+1}) + \nabla f(x_{k+1}) - \nabla f(x_k)\|_2$
- Shows $\beta^{-1}\|u_k\|_2$ up to 200'000 iterations (10x more than before)
- Many iterations needed for high accuracy



Outline

- A fundamental inequality
- Nonconvex setting
- Convex setting
- Strongly convex setting
- Backtracking
- Stopping conditions
- **Accelerated gradient method**
- Scaling

Accelerated proximal gradient method

- Consider *convex* composite problem

$$\underset{x}{\operatorname{minimize}}\, f(x) + g(x)$$

  where
  - $f : \mathbb{R}^n \to \mathbb{R}$ is β-smooth and convex
  - $g : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is closed and convex
- Proximal gradient descent

$$x_{k+1} = \operatorname{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))$$

  achieves $O(1/k)$ convergence rate in function value
- *Accelerated* proximal gradient method

$$\begin{aligned} y_k &= x_k + \theta_k(x_k - x_{k-1}) \\ x_{k+1} &= \operatorname{prox}_{\gamma g}(y_k - \gamma \nabla f(y_k)) \end{aligned}$$

  (with specific θ_k) achieves faster $O(1/k^2)$ convergence rate

Accelerated proximal gradient method – Parameters

- *Accelerated* proximal gradient method

$$\begin{aligned} y_k &= x_k + \theta_k(x_k - x_{k-1}) \\ x_{k+1} &= \operatorname{prox}_{\gamma g}(y_k - \gamma \nabla f(y_k)) \end{aligned}$$

- Step-sizes are restricted $\gamma \in (0, \frac{1}{\beta}]$
- The θ_k parameters can be chosen either as

$$\theta_k = \tfrac{k-1}{k+2}$$

  or $\theta_k = \frac{t_{k-1}-1}{t_k}$ where

$$t_k = \tfrac{1+\sqrt{1+4t_{k-1}^2}}{2}$$

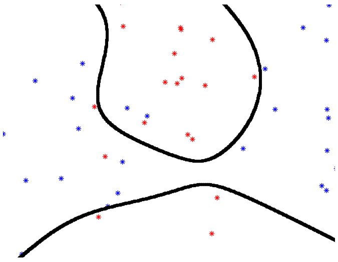
  these choices are very similar
- Algorithm behavior in nonconvex setting not well understood

Not a descent method

- Descent method means function value is decreasing every iteration
- We know that proximal gradient method is a descent method
- However, accelerated proximal gradient method is not

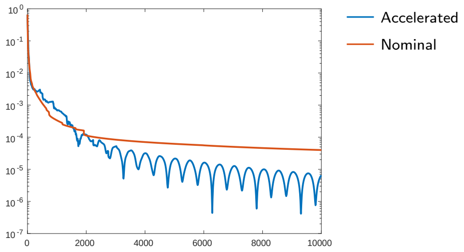
Accelerated gradient method – Example

- Accelerated vs nominal proximal gradient method
- Problem from SVM lecture, polynomial deg 6 and $\lambda = 0.0215$



Accelerated gradient method – Example

- Accelerated vs nominal proximal gradient method
- Problem from SVM lecture, polynomial deg 6 and $\lambda = 0.0215$



Outline

- A fundamental inequality
- Nonconvex setting
- Convex setting
- Strongly convex setting
- Backtracking
- Stopping conditions
- Accelerated gradient method
- **Scaling**

Scaled proximal gradient method

- Proximal gradient method:

$$x_{k+1} = \underset{y}{\operatorname{argmin}} \left(\underbrace{f(x_k) + \nabla f(x_k)^T(y-x) + \tfrac{1}{2\gamma_k}\|y-x_k\|_2^2}_{\hat{f}_{x_k}(y)} + g(y) \right)$$

  approximates function $f(y)$ around x_k by $\hat{f}_{x_k}(y)$
- The better the approximation, the faster the convergence
- By scaling: we mean to use an approximation of the form

$$\hat{f}_{x_k}(y) = f(x_k) + \nabla f(x_k)^T(y-x_k) + \tfrac{1}{2\gamma_k}\|y-x_k\|_H^2$$

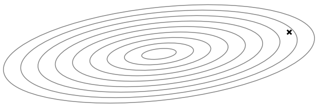
  where $H \in \mathbb{R}^{n\times n}$ is a positive definite matrix and $\|x\|_H^2 = x^T H x$

Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}} \ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T \begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

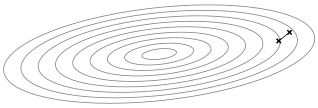


Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}} \ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T \begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

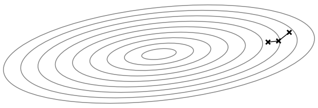


Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}} \ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T \begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

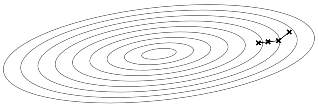


Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

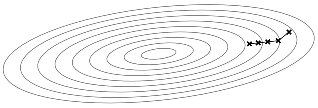


Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

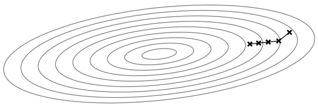


Gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Step-size $\gamma = \frac{1}{\beta}$ and norm $\|\cdot\|_2$ in model

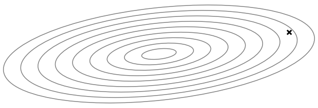


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$

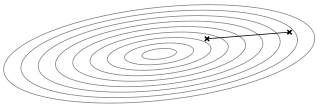


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$

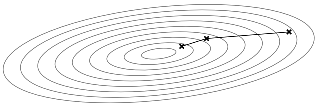


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$

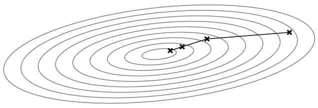


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$

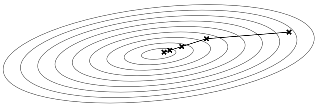


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$

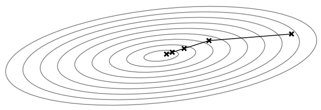


Scaled gradient descent – Example

- Gradient descent on β-smooth quadratic problem

$$\underset{x}{\text{minimize}} \ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f)$, γ is inverse smoothness w.r.t. $\|\cdot\|_H$



Smoothness w.r.t. $\|\cdot\|_H$

What is $\|\cdot\|_H$?

- Requirement: $H \in \mathbb{R}^{n\times n}$ is symmetric positive definite ($H \succ 0$)
- The norm $\|x\|_H^2 := x^T Hx$, for $H = I$, we get $\|x\|_I^2 = \|x\|_2^2$

Smoothness

- Function $f : \mathbb{R}^n \to \mathbb{R}$ is β-smooth if for all $x, y \in \mathbb{R}^n$:

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) - \tfrac{\beta}{2}\|x-y\|_2^2$$

- We say f β_H-smoothness w.r.t. scaled norm $\|\cdot\|_H$ if

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta_H}{2}\|x-y\|_H^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) - \tfrac{\beta_H}{2}\|x-y\|_H^2$$

  for all $x, y \in \mathbb{R}^n$
- If f is smooth (w.r.t. $\|\cdot\|_2$) it is also smooth w.r.t. $\|\cdot\|_H$

Example – A quadratic

- Let $f(x) = \frac{1}{2}x^T Hx = \frac{1}{2}\|x\|_H^2$ with $H \succ 0$
- f is 1-smooth w.r.t $\|\cdot\|_H$ (with equality):

$$\begin{aligned} &f(x) + \nabla f(x)^T(y-x) + \tfrac{1}{2}\|x-y\|_H^2\\ &\quad= \tfrac{1}{2}x^T Hx + (Hx)^T(y-x) + \tfrac{1}{2}\|x-y\|_H^2\\ &\quad= \tfrac{1}{2}x^T Hx + (Hx)^T(y-x) + \tfrac{1}{2}(\|x\|_H^2 - 2(Hx)^T y + \|y\|_H^2)\\ &\quad= \tfrac{1}{2}\|y\|_H^2 = f(y)\end{aligned}$$

  which holds also if adding linear term $q^T x$ to f
- f is $\lambda_{\max}(H)$-smooth (w.r.t. $\|\cdot\|_2$), continue equality:

$$\begin{aligned} f(y) &= f(x) + \nabla f(x)^T(y-x) + \tfrac{1}{2}\|x-y\|_H^2\\ &\leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\lambda_{\max}(H)}{2}\|x-y\|_2^2\end{aligned}$$

  much more conservative estimate of function!

Scaled proximal gradient for quadratics

- Let $f(x) = \frac{1}{2}x^T Hx$ with $H \succ 0$, which is 1-smooth w.r.t. $\|\cdot\|_H$
- Approximation with scaled norm $\|\cdot\|_H$ and $\gamma_k = 1$ satisfies $\forall x_k$:

$$\hat{f}_{x_k}(y) = f(x_k) + \nabla f(x_k)^T(y-x_k) + \tfrac{1}{2}\|x_k-y\|_H^2 = f(y)$$

  since f is 1-smooth w.r.t. $\|\cdot\|_H$ with equality
- An iteration then reduces to solving problem itself:

$$x_{k+1} = \underset{y}{\operatorname{argmin}}(\hat{f}_{x_k}(y) + g(y)) = \underset{y}{\operatorname{argmin}}(f(y) + g(y))$$

- Model very accurate, but very expensive iterations

Scaled proximal gradient method reformulation

- Proximal gradient method with scaled norm $\|\cdot\|_H$:

$$\begin{aligned} x_{k+1} &= \underset{y}{\operatorname{argmin}}\left(f(x_k) + \nabla f(x_k)^T(y-x) + \tfrac{1}{2\gamma_k}\|y-x_k\|_H^2 + g(y)\right)\\ &= \underset{y}{\operatorname{argmin}}\left(g(y) + \tfrac{1}{2\gamma_k}\|y-(x_k-\gamma_k H^{-1}\nabla f(x_k))\|_H^2\right)\\ &=: \operatorname{prox}_{\gamma_k g}^H(x_k - \gamma_k H^{-1}\nabla f(x_k))\end{aligned}$$

  where $H = I$ gives nominal method
- Computational difference per iteration:
  1. Need to invert H^{-1} (or solve $Hd_k = \nabla f(x_k)$)
  2. Need to compute prox with new metric

$$\operatorname{prox}_{\gamma_k g}^H(z) := \underset{x}{\operatorname{argmin}}(g(x) + \tfrac{1}{2\gamma_k}\|x-z\|_H^2)$$

  that may be very costly

Computational cost

- Assume that H is dense or general sparse
  - H^{-1} dense: cubic complexity (vs maybe quadratic for gradient)
  - H^{-1} sparse: lower than cubic complexity
  - $\operatorname{prox}_{\gamma_k g}^H$: difficult optimization problem
- Assume that H is diagonal
  - H^{-1}: invert diagonal elements – linear complexity
  - $\operatorname{prox}_{\gamma_k g}^H$: often as cheap as nominal prox (e.g., for separable g)
  - this gives individual step-sizes for each coordinate
- Assume that H is block-diagonal with small blocks
  - H^{-1}: invert individual blocks – also cheap
  - $\operatorname{prox}_{\gamma_k g}^H$: often quite cheap (e.g., for block-separable g)
- If $H = I$, method is nominal method

Convergence

- We get similar results as in the nominal $H = I$ case
- We assume β_H smoothness w.r.t. $\|\cdot\|_H$
- We can replace all $\|\cdot\|_2$ with $\|\cdot\|_H$ and ∇f with $H^{-1}\nabla f$:
  - Nonconvex setting with $\gamma_k = \frac{1}{\beta_H}$

$$\min_{l\in\{0,\ldots,k\}}\|\nabla f(x_l)\|_{H^{-1}}^2 \leq \frac{2\beta_H(f(x_0)+g(x_0)-p^\star)}{k+1}$$

  - Convex setting with $\gamma_k = \frac{1}{\beta_H}$

$$f(x_k) + g(x_k) - p^\star \leq \frac{\beta_H\|x_0-x^\star\|_H^2}{2(k+1)}$$

  - Strongly convex setting with f σ_H-strongly convex w.r.t. $\|\cdot\|_H$

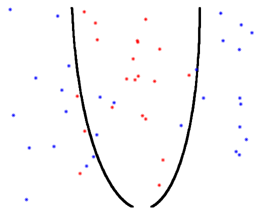
$$\|x_{k+1}-x^\star\|_H \leq \max(\beta_H\gamma - 1, 1-\sigma_H\gamma)\|x_k-x^\star\|_H$$

Example – Logistic regression

- Logistic regression with $\theta = (w, b)$:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \log(1 + e^{w^T\phi(x_i)+b}) - y_i(w^T\phi(x_i)+b) + \tfrac{\lambda}{2}\|w\|_2^2$$

  on the following data set (from logistic regression lecture)
- Polynomial features of degree 6, Tikhonov regularization $\lambda = 0.01$
- Number of decision variables: 28



Algorithms

Compare the following algorithms, all with backtracking:

1. Gradient method
2. Gradient method with fixed diagonal scaling
3. Gradient method with fixed full scaling

Fixed scalings

- Logistic regression gradient and Hessian satisfy with $L = [X, \mathbf{1}]$

$$\nabla f(\theta) = L^T(\sigma(L\theta) - Y) + \lambda I_w\theta \quad \nabla^2 f(\theta) = L^T\sigma'(L\theta)L + \lambda I_w$$

where σ is the (vector-version of) sigmoid, and $I_w(w, b) = (w, 0)$

- The sigmoid function σ is 0.25-Lipschitz continuous
- Gradient method with fixed full scaling (3.) uses

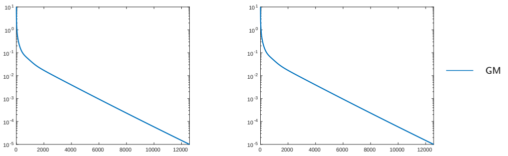
$$H = 0.25L^TL + \lambda I_w$$

- Gradient method with fixed diagonal scaling (2.) uses

$$H = \mathbf{diag}(0.25L^TL + \lambda I_w)$$

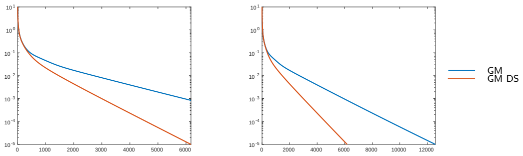
Example – Numerics

- Logistic regression polynomial features of degree 6, $\lambda = 0.01$
- Standard gradient method with backtracking (GM)



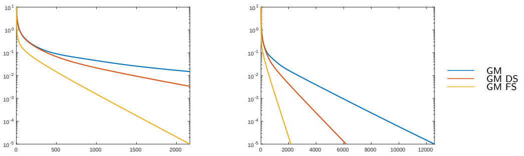
Example – Numerics

- Logistic regression polynomial features of degree 6, $\lambda = 0.01$
- Gradient method with diagonal scaling (GM DS)



Example – Numerics

- Logistic regression polynomial features of degree 6, $\lambda = 0.01$
- Gradient method with full matrix scaling (GM FS)



Comments

- Smaller number of iterations with better scaling
- Performance is roughly (iteration cost)×(number of iterations)
  - We have only compared number of iterations
  - Iteration cost for (GM) and (GM DS) are the same
  - Iteration cost for (GM FS) higher
  - Need to quantify iteration cost to assess which is best
- In general, can be difficult to find H that performs better

Stochastic Gradient Descent

Qualitative Convergence Behavior

Pontus Giselsson

Outline

- **Stochastic gradient descent**
- Convergence and distance to solution
- Convergence and solution norms
- Overparameterized vs underparameterized setting
- Escaping not individually flat minima
- SGD step-sizes
- SGD convergence

Notation

- Optimization (decision) variable notation:
  - Optimization literature: x, y, z
  - Statistics literature: β
  - Machine learning literature: θ, w, b
- Data and labels in statistics and machine learning are x, y
- Training problems in supervised learning

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

optimizes over decision variable θ for fixed data $\{(x_i, y_i)\}_{i=1}^{N}$
- Optimization problem in standard optimization notation

$$\underset{x}{\text{minimize}}\, f(x)$$

optimizes over decision variable x
- Will use optimization notation when algorithms not applied in ML

Gradient method

- Gradient method is applied problems of the form

$$\underset{x}{\text{minimize}}\, f(x)$$

where f is differentiable and gradient method is

$$x_{k+1} = x_k - \gamma_k \nabla f(x_k)$$

where $\gamma_k > 0$ is a step-size
- f not differentiable in DL with ReLU but still say gradient method
- For large problems, gradient can be expensive to compute
⇒ replace by unbiased stochastic approximation of gradient

Unbiased stochastic gradient approximation

- Stochastic gradient *estimator*:
  - notation: $\widehat{\nabla} f(x)$
  - outputs random vector in $\mathbb{R}^n$ for each $x \in \mathbb{R}^n$
- Stochastic gradient *realization*:
  - notation: $\widetilde{\nabla} f(x) : \mathbb{R}^n \to \mathbb{R}^n$
  - outputs, $\forall x \in \mathbb{R}^n$, vector in $\mathbb{R}^n$ drawn from distribution of $\widehat{\nabla} f(x)$
- An unbiased stochastic gradient estimator $\widehat{\nabla} f$ satisfies $\forall x \in \mathbb{R}^n$:

$$\mathbb{E}\widehat{\nabla} f(x) = \nabla f(x)$$

- If x is random vector in $\mathbb{R}^n$, unbiased estimator satisfies

$$\mathbb{E}[\widehat{\nabla} f(x)|x] = \nabla f(x)$$

(both are random vectors in $\mathbb{R}^n$)

Stochastic gradient descent (SGD)

- The following iteration generates $(x_k)_{k\in\mathbb{N}}$ of *random* variables:

$$x_{k+1} = x_k - \gamma_k \widehat{\nabla} f(x_k)$$

since $\widehat{\nabla} f$ outputs random vectors in $\mathbb{R}^n$
- Stochastic gradient descent finds a *realization* of this sequence:

$$x_{k+1} = x_k - \gamma_k \widetilde{\nabla} f(x_k)$$

where $(x_k)_{k\in\mathbb{N}}$ here is a realization with values in $\mathbb{R}^n$
- Sloppy in notation for when x_k is *random variable* vs *realization*
- Can be efficient if evaluating $\widetilde{\nabla} f$ much cheaper than ∇f

Stochastic gradients – Finite sum problems

- Consider *finite sum problems* of the form

$$\underset{x}{\text{minimize}}\, \underbrace{\tfrac{1}{N}\left(\sum_{i=1}^{N} f_i(x)\right)}_{f(x)}$$

where $\frac{1}{N}$ is for convenience and gives average loss
- Training problems of this form, where sum over training data
- Stochastic gradient: select f_i at random and take gradient step

Single function stochastic gradient

- Let I be a $\{1, \ldots, N\}$-valued random variable
- Let, as before, $\widehat{\nabla} f$ denote the stochastic gradient estimator
- Realization: let i be drawn from probability distribution of I

$$\widetilde{\nabla} f(x) = \nabla f_i(x)$$

where we will use uniform probability distribution

$$p_i = p(I = i) = \tfrac{1}{N}$$

- Stochastic gradient is unbiased:

$$\mathbb{E}[\widehat{\nabla} f(x)] = \sum_{i=1}^{N} p_i \nabla f_i(x) = \tfrac{1}{N}\sum_{i=1}^{N} \nabla f_i(x) = \nabla f(x)$$

Mini-batch stochastic gradient

- Let $\mathcal{B}$ be set of K-sample mini-batches to choose from:
  - Example: 2-sample mini-batches and $N = 4$:

$$\mathcal{B} = \{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$$

  - Number of mini batches $\binom{N}{K}$, each item in $\binom{N-1}{K-1}$ batches
- Let $\mathbb{B}$ be $\mathcal{B}$-valued random variable
- Let, as before, $\widehat{\nabla} f$ denote stochastic gradient estimator
- Realization: let B be drawn from probability distribution of $\mathbb{B}$

$$\widetilde{\nabla} f(x) = \tfrac{1}{K}\sum_{i\in B}\nabla f_i(x)$$

where we will use uniform probability distribution

$$p_B = p(\mathbb{B} = B) = \tfrac{1}{\binom{N}{K}}$$

- Stochastic gradient is unbiased:

$$\mathbb{E}\widehat{\nabla} f(x) = \tfrac{1}{\binom{N}{K}}\sum_{B\in\mathcal{B}}\tfrac{1}{K}\sum_{i\in B}\nabla f_i(x) = \tfrac{\binom{N-1}{K-1}}{\binom{N}{K}K}\sum_{i=1}^{N}\nabla f_i(x) = \tfrac{1}{N}\sum_{i=1}^{N}\nabla f_i(x) = \nabla f(x)$$

Stochastic gradient descent for finite sum problems

- The algorithm, choose $x_0 \in \mathbb{R}^n$ and iterate:
  1. Sample a mini-batch $B_k \in \mathcal{B}$ of K indices uniformly
  2. Update

$$x_{k+1} = x_k - \tfrac{\gamma_k}{K}\sum_{j\in B_k}\nabla f_j(x_k)$$

- Can have $\mathcal{B} = \{\{1\},\ldots,\{N\}\}$ and sample only one function
- Gives realization of underlying stochastic process

Outline

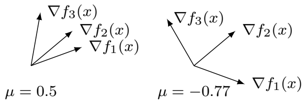
- Stochastic gradient descent
- **Convergence and distance to solution**
- Convergence and solution norms
- Overparameterized vs underparameterized setting
- Escaping not individually flat minima
- SGD step-sizes
- SGD convergence

Qualitative convergence behavior

- Consider single-function batch setting
- Assume that the individual gradients satisfy

$$(\nabla f_i(x))^T(\nabla f_j(x)) \geq \mu$$

for all i, j and for some $\mu \in \mathbb{R}$ (i.e., can be positive or negative)



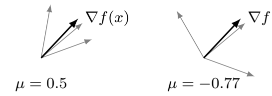
Will larger or smaller μ likely give better SGD convergence? Why?

Qualitative convergence behavior

- Consider single-function batch setting
- Assume that the individual gradients satisfy

$$(\nabla f_i(x))^T(\nabla f_j(x)) \geq \mu$$

for all i, j and for some $\mu \in \mathbb{R}$ (i.e., can be positive or negative)



Will larger or smaller μ likely give better SGD convergence? Why?

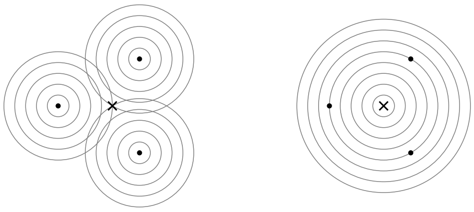
- Larger μ gives more similar to full gradient and faster convergence

Minibatch setting

- Larger minibatch gives larger μ and faster convergence
- Comes at the cost of higher per iteration count
- Limiting minibatch case is the gradient method
- Tradeoff in how large minibatches to use to optimize convergence
- Other reasons exist that favor small batches (later)

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\text{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look for SGD with $\gamma_k = 1/3$?

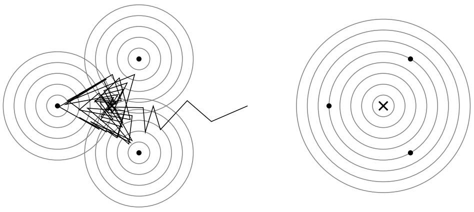


Levelsets of summands

Levelset of sum

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\text{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look for SGD with $\gamma_k = 1/3$?

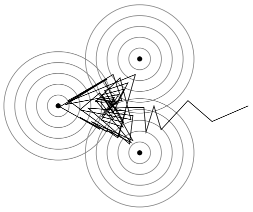


Levelsets of summands

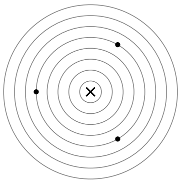
Levelset of sum

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\mathrm{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look for SGD with $\gamma_k = 1/3$?



Levelsets of summands

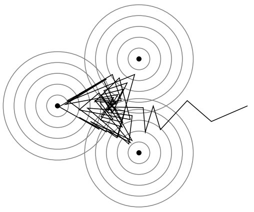


Levelset of sum

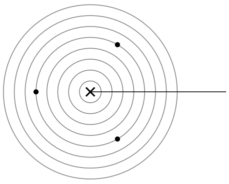
- Fast convergence outside “triangle” where gradients similar, slow inside
- Constant step SGD converges to noise ball

SGD – Example

- Let $c_1 + c_2 + c_3 = 0$
- Solve $\mathrm{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look for SGD with $\gamma_k = 1/3$?



Levelsets of summands

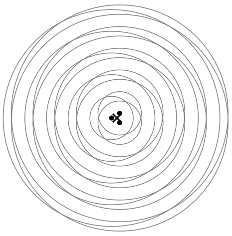


Levelset of sum

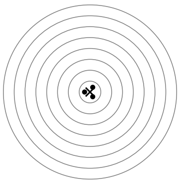
- Constant step GD converges (in this case straight to) solution (right)
- Difference is noise in stochastic gradient that can be measured by μ

SGD – Example zoomed out

- Same example but zoomed out
- Solve $\mathrm{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look with $\gamma_k = 1/3$ from more global view?



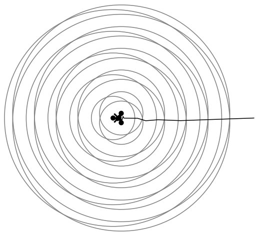
Levelsets of summands



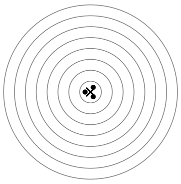
Levelset of sum

SGD – Example zoomed out

- Same example but zoomed out
- Solve $\mathrm{minimize}_x(\frac{1}{2}(\|x-c_1\|_2^2 + \|x-c_2\|_2^2 + \|x-c_3\|_2^2)) = \frac{3}{2}\|x\|_2^2 + c$
- How will trajectory look with $\gamma_k = 1/3$ from more global view?



Levelsets of summands



Levelset of sum

- Far form solution ∇f_i more similar to ∇f, larger $\mu \Rightarrow$ faster convergence

---

Qualitative convergence behavior

- Often fast convergence far from solution, slow close to solution
- Fixed-step size converges to noise ball in general
- Need diminishing step-size to converge to solution in general

---

Drawback of diminishing step-size

- Diminishing step-size typically gives slow convergence
- Often better convergence with constant step (if it works)
- Is there a setting in which constant step-size works?

Outline

- Stochastic gradient descent
- Convergence and distance to solution
- **Convergence and solution norms**
- Overparameterized vs underparameterized setting
- Escaping not individually flat minima
- SGD step-sizes
- SGD convergence

Fixed step-size SGD does not converge to solution

- We can at most hope for finding point $\bar{x}$ such that

$$\nabla f(\bar{x}) = 0$$

- Let $x_k = \bar{x}$, and assume $\nabla f_i(x_k) \neq 0$, then

$$x_{k+1} = x_k - \gamma_k \nabla f_i(x_k) \neq x_k$$

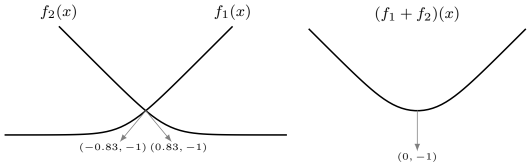
  i.e., moves away from solution $\bar{x}$
- Only hope with fixed step-size if all $\nabla f_i(\bar{x}) = 0$, since for $x_k = \bar{x}$

$$x_{k+1} = x_k - \gamma_k \nabla f_i(x_k) = x_k$$

  independent on γ_k and algorithm stays at solution
- How does norm of individual gradients affect local convergence?

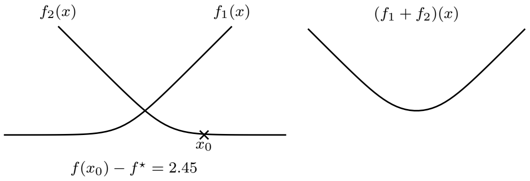
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



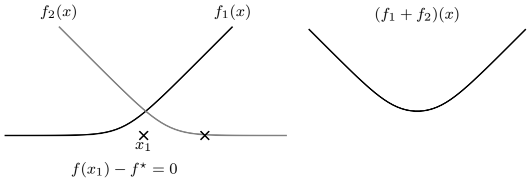
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



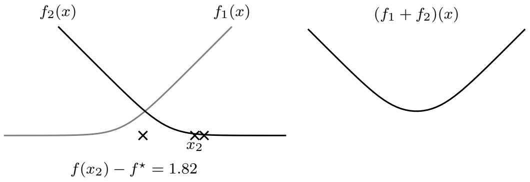
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



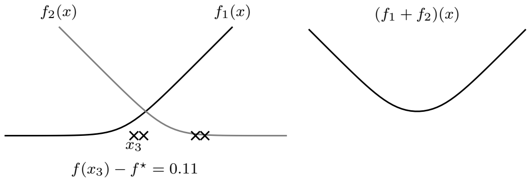
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



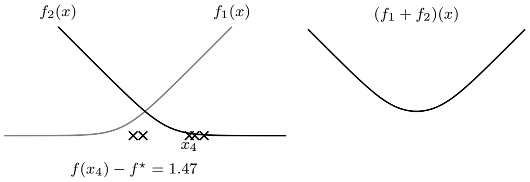
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



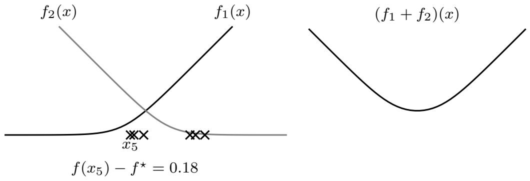
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



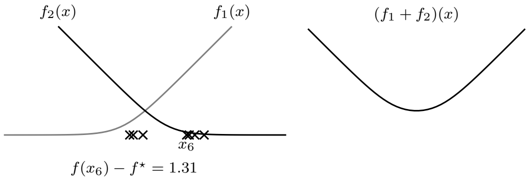
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



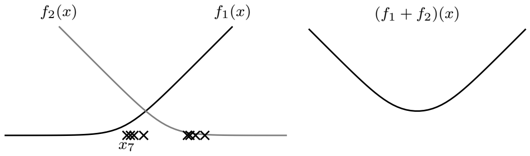
Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



Example – Large gradients at solution

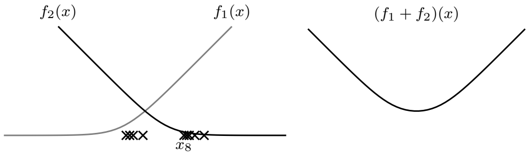
- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_7) - f^\star = 0.28$

Example – Large gradients at solution

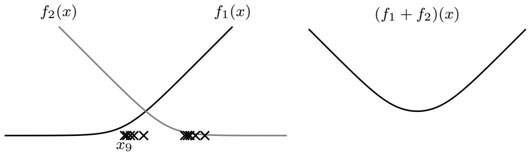
- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_8) - f^\star = 1.16$

Example – Large gradients at solution

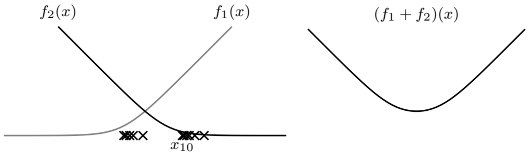
- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_9) - f^\star = 0.35$

Example – Large gradients at solution

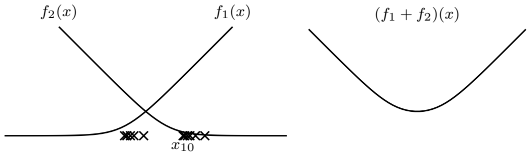
- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_{10}) - f^\star = 1.07$

Example – Large gradients at solution

- Individal gradients at solution 0: $\nabla f_1(0) = 0.83$, $\nabla f_2(0) = -0.83$
- SGD with $\gamma = 0.07$ and cyclic update order:

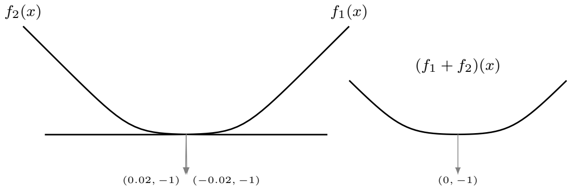


$f(x_{10}) - f^\star = 1.07$

- Will not converge to solution with constant step-size

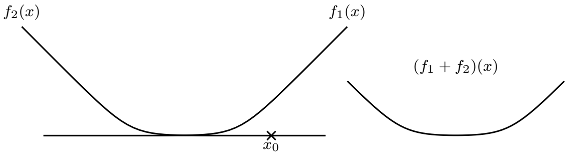
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



Example – Small gradients at solution

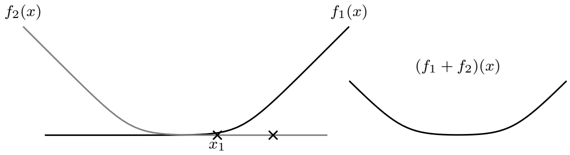
- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_0) - f^\star = 2.45$

Example – Small gradients at solution

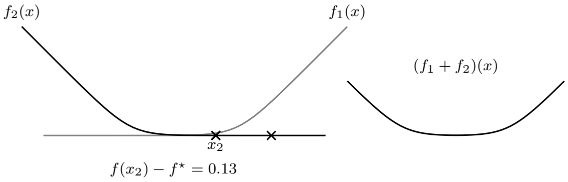
- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



$f(x_1) - f^\star = 0.13$

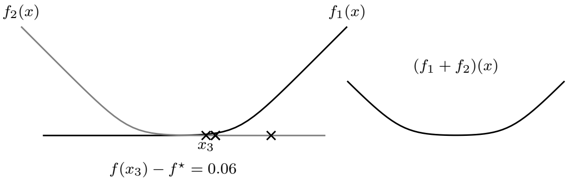
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



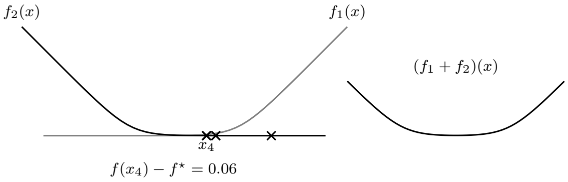
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



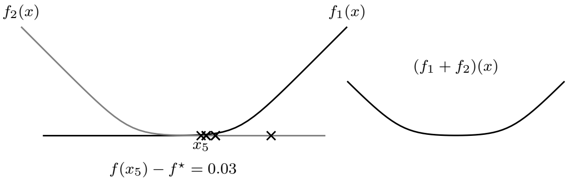
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



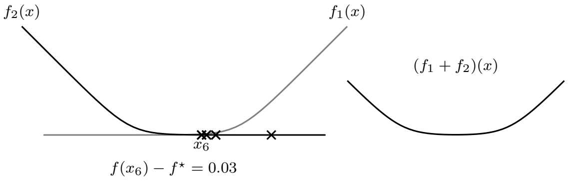
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



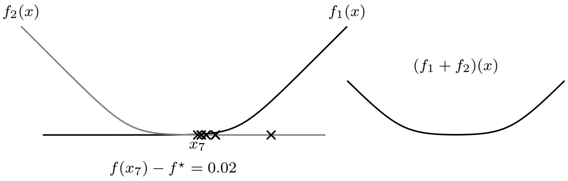
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



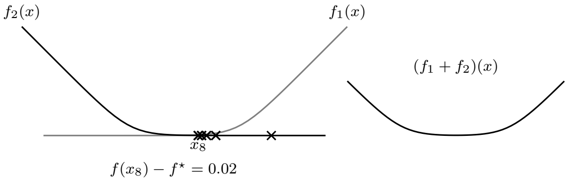
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



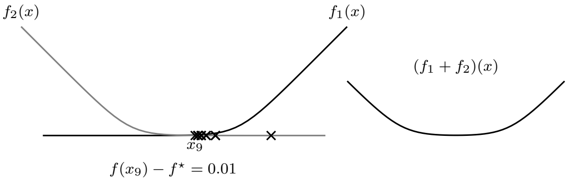
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



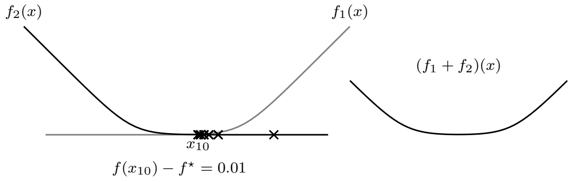
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



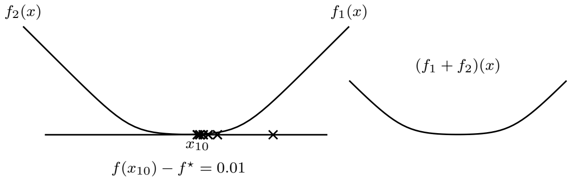
Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



Example – Small gradients at solution

- Shift f_1 and f_2 "outwards" to get new problem
- Individal gradients at solution 0: $\nabla f_1(0) = 0.02$, $\nabla f_2(0) = -0.02$
- SGD with $\gamma = 0.07$ and cyclic update order:



- Much faster to reach small loss

Convergence and individual gradient norm

Local convergence of stochastic gradient descent is:

- slow if individual functions do not agree on minima
  - individual norms "large" at and around minima
- faster if individual functions do agree on minima
  - individual norms "small" at and around minima

Outline

- Stochastic gradient descent
- Convergence and distance to solution
- Convergence and solution norms
- **Overparameterized vs underparameterized setting**
- Escaping not individually flat minima
- SGD step-sizes
- SGD convergence

Over- vs under-parameterized models

- Model overparameterized if:
  - in regression, zero loss is possible
  - in classification, correct classification with margin possible
    - logistic loss gives close to 0 loss
    - hinge loss gives 0 loss
- Model underparameterized if the above does not hold

Overparameterization – LS example

- Data $A \in \mathbb{R}^{N \times n}$, $b \in \mathbb{R}^N$, and $x \in \mathbb{R}^n$
- Consider least squares problem

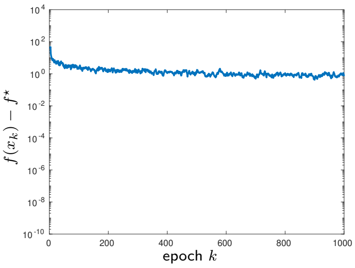
$$\underset{x}{\text{minimize}}\ \underbrace{\tfrac{1}{2}\|Ax - b\|_2^2}_{f(x)} = \sum_{i=1}^{N} \underbrace{\tfrac{1}{2}(a_i x - b_i)^2}_{f_i(x)}$$

where $a_i \in \mathbb{R}^{1 \times n}$ are rows in A and problem is

- overparameterized if $n > N$ (infinitely many 0-loss solutions)
- underparameterized if $n \leq N$ (unique solution if A full rank)

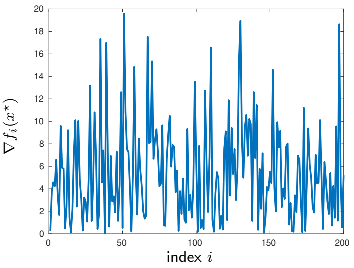
Convergence – LS example

- Random problem data: $A \in \mathbb{R}^{200 \times 100}$, $b \in \mathbb{R}^{200}$ from Gaussian
- Underparameterized setting and unique solution
- Local convergence of SGD quite slow:



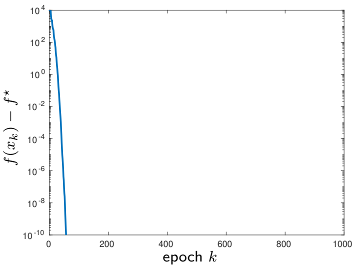
Convergence – LS example

- Random problem data: $A \in \mathbb{R}^{200 \times 100}$, $b \in \mathbb{R}^{200}$ from Gaussian
- Underparameterized setting and unique solution
- Norms of $\nabla f_i(x^\star) = \frac{1}{2}(a_i x^\star - b_i)$ quite large:



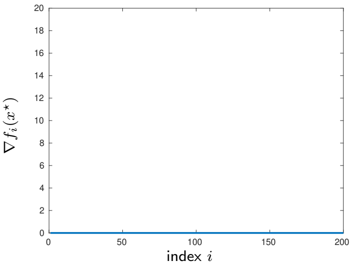
Convergence – LS example

- Random problem data: $A \in \mathbb{R}^{200\times1000}$, $b \in \mathbb{R}^{200}$ from Gaussian
- Overparameterized, many 0-loss solutions, larger problem
- Convergence of SGD much faster:



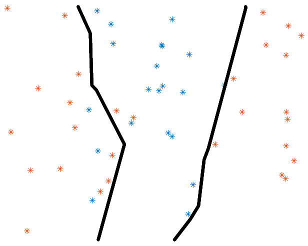
Convergence – LS example

- Random problem data: $A \in \mathbb{R}^{200\times1000}$, $b \in \mathbb{R}^{200}$ from Gaussian
- Overparameterized, many 0-loss solutions, larger problem
- Individual norms $\nabla f_i(x^\star) = \frac{1}{2}(a_i x^\star - b_i) = 0$:



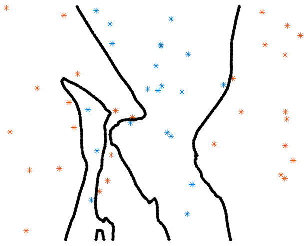
Convergence – DL example

- Classification problem: logistic loss
- Network: Residual, ReLU, 3x5,2,1 widths (5 layers)
- Underparameterized:



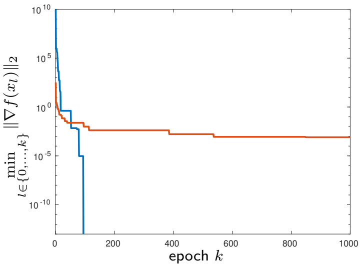
Convergence – DL example

- Classification problem: logistic loss
- Network: Residual, ReLU, 15x25,2,1 widths (17 layers)
- Overparameterized:



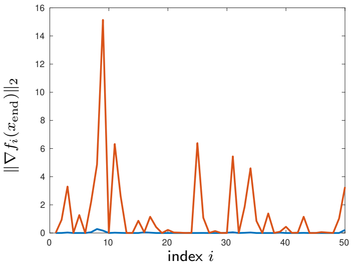
Convergence – DL example

- Classification problem: logistic loss
- Network: Residual, ReLU, 3x5,2,1 vs 15x25,2,1
- Convergence of "best gradient" (final loss: 0.17 vs 0.00018):



Convergence – DL example

- Classification problem: logistic loss
- Network: Residual, ReLU, 3x5,2,1 vs 15x25,2,1
- Final norm of individual gradients (final loss: 0.17 vs 0.00018):



Overparameterized networks and convergence

- Overparameterized models seems to give faster SGD convergence
- Reason: individual gradients agree better!

Outline

- Stochastic gradient descent
- Convergence and distance to solution
- Convergence and solution norms
- Overparameterized vs underparameterized setting
- **Escaping not individually flat minima**
- SGD step-sizes
- SGD convergence

Step-length

- The step-length in constant step SGD is given by

$$\|x_{k+1} - x_k\|_2 = \gamma \|\nabla f_i(x_k)\|_2$$

i.e., proportional to individual gradient norm

- The step-length in constant step GD is given by

$$\|x_{k+1} - x_k\|_2 = \gamma \|\nabla f(x_k)\|_2$$

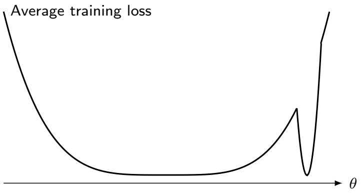
i.e., proportional to full (average) gradient norm

Flatness of minima

- Is SGD or GD more likely to escape the sharp minima?

Flatness of minima

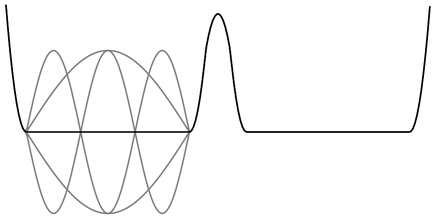
- Is SGD or GD more likely to escape the sharp minima?



- Impossible to say only from average training loss

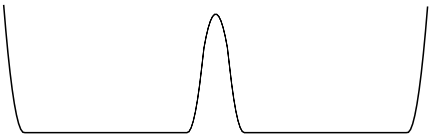
Example

- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



Example

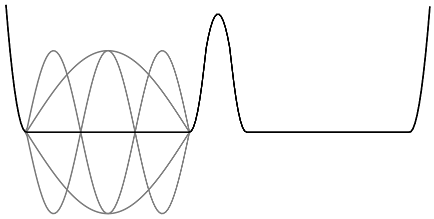
- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



- GD will stay in both minima ($\nabla f(x_k) = 0 \Rightarrow x_{k+1} = x_k$)

Example

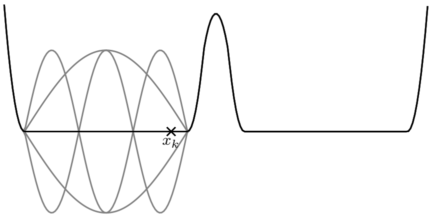
- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



- GD will stay in both minima ($\nabla f(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD will stay in right minima ($\nabla f_i(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD may escape left minima ($\|\nabla f_i(x_k)\|_2 \neq 0 \Rightarrow x_{k+1} \neq x_k$)

Example

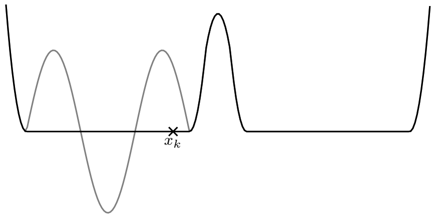
- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



- GD will stay in both minima ($\nabla f(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD will stay in right minima ($\nabla f_i(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD may escape left minima ($\|\nabla f_i(x_k)\|_2 \neq 0 \Rightarrow x_{k+1} \neq x_k$)
- $x_k = 0.8$ and $\gamma = 0.5$

Example

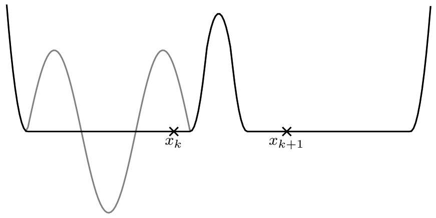
- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



- GD will stay in both minima ($\nabla f(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD will stay in right minima ($\nabla f_i(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD may escape left minima ($\|\nabla f_i(x_k)\|_2 \neq 0 \Rightarrow x_{k+1} \neq x_k$)
- $x_k = 0.8$ and $\gamma = 0.5$, $i = 4$ and $\nabla f_i(x_k) = -2.77$

Example

- Flat (local) minima can be different
- Is SGD or GD more likely to escape right/left minima?



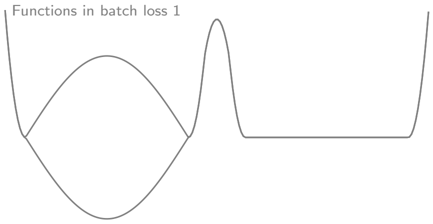
- GD will stay in both minima ($\nabla f(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD will stay in right minima ($\nabla f_i(x_k) = 0 \Rightarrow x_{k+1} = x_k$)
- SGD may escape left minima ($\|\nabla f_i(x_k)\|_2 \neq 0 \Rightarrow x_{k+1} \neq x_k$)
- $x_k = 0.8$ and $\gamma = 0.5$, $i = 4$ and $\nabla f_i(x_k) = -2.77$, $x_{k+1} = 2.18$

Mini-batch vs single-batch

- Is escape property effected by mini-batch size?
- How large mini-batch size is best for escaping?

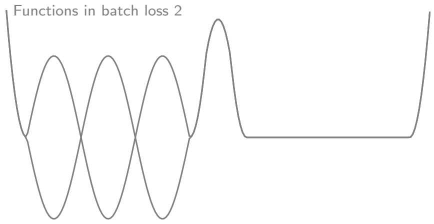
Mini-batch setting

- Use mini-batches of size 2:



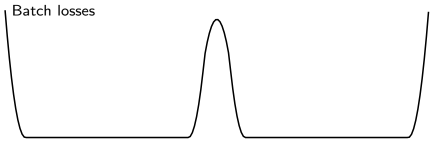
Mini-batch setting

- Use mini-batches of size 2:



Mini-batch setting

- Use mini-batches of size 2:



- Larger mini-batch $\Rightarrow$ smaller gradients $\Rightarrow$ worse at escaping
- Single-batch better at escaping

Connection to generalization

- Argued that individually flat minima generalize better, i.e.,

all $\|\nabla f_i(x)\|_2$ small in region around minima

- SGD more likely to escape if individual gradients not small
- Smaller batch size increases chances of escaping "bad" minima

Have also argued for:

- Good convergence properties towards individually flat minima

In summary:

- Single-batch SGD well suited for overparameterized training

Outline

- Stochastic gradient descent
- Convergence and distance to solution
- Convergence and solution norms
- Overparameterized vs underparameterized setting
- Escaping not individually flat minima
- **SGD step-sizes**
- SGD convergence

Step-sizes

- Diminising step-sizes are needed for convergence in general
- Common static step-size rules
  - redude step-size every K epochs:

$$\gamma_k = \frac{\gamma_0}{1 + \lceil k/K \rceil} \qquad \gamma_k = \frac{\gamma_0}{1 + \sqrt{\lceil k/K \rceil}}$$

  where $\lceil k/K \rceil$ increases by 1 every K epochs
  - Convergence analysis under smoothness or convexity requires

$$\sum_{k=0}^{\infty} \gamma_k = \infty \qquad \text{and} \qquad \sum_{k=0}^{\infty} \gamma_k^2 < \infty$$

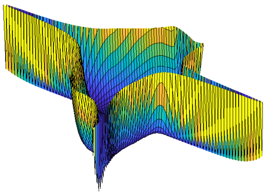
  which is satisfied by first but not second above
  - Refined analysis gives requirements

$$\sum_{k=0}^{\infty} \gamma_k = \infty \qquad \text{and} \qquad \frac{\sum_{k=0}^{\infty} \gamma_k}{\sum_{k=0}^{\infty} \gamma_k^2} = \infty$$

  which is satisfied by all the above

Large gradients

- Fixed step-size rules does not take gradient size into account
- Gradients can be very large:



- Step-size rule

$$\gamma_k = \frac{\gamma_0}{\alpha\|\widetilde{\nabla} f(x_k)\|_2 + 1}$$

with $\gamma_0, \alpha > 0$ gives
  - small steps if $\|\widetilde{\nabla} f(x_k)\|_2$ large
  - approximately γ_0 steps if $\|\widetilde{\nabla} f(x_k)\|_2$ small

Combined step-size rule

- Combination the two previous rules

$$\gamma_k = \frac{\gamma_0}{(1+\psi(\lceil k/K\rceil))(\alpha\|\widetilde{\nabla} f(x_k)\|_2 + 1)}$$

where, e.g., $\psi(x) = \frac{1}{x}$ or $\psi(x) = \frac{1}{\sqrt{x}}$ (as before)
- Properties
  - $\|\widetilde{\nabla} f(x_k)\|_2$ large: small step-sizes
  - $\|\widetilde{\nabla} f(x_k)\|_2$ small: diminshing step-sizes according to $\frac{\gamma_0}{1+\psi(\lceil k/K\rceil)}$

Step-size rules and convergence

- Classification, Residual layers, ReLU, 15x25,2,1 widths (17 layers)
- Step-size parameters: $\psi(x) = 0.5\sqrt{x}$, $K = 50$, $\alpha = \gamma_0 = 0.1$
- Iteration data:

# epoch	step-size	batch norm	full norm
0	$4.8 \cdot 10^{-8}$	$2.1 \cdot 10^{7}$	$6.8 \cdot 10^{5}$
10	$1.4 \cdot 10^{-5}$	$7.2 \cdot 10^{4}$	$1.4 \cdot 10^{4}$
50	0.097	0.31	1.4
100	0.016	0.28	3.2
200	0.012	$6.8 \cdot 10^{-5}$	0.72
300	0.01	0.33	11.8
500	0.008	0	0.529
700	0.007	$1.2 \cdot 10^{-6}$	0.0008
1000	0.006	$3.1 \cdot 10^{-6}$	0.0003

- Large initial gradients dampened
- Diminishing step-size gives local convergence

Step-size rules and convergence

- Classification, Residual layers, ReLU, 15x25,2,1 widths (17 layers)
- Step-size parameters: $\psi(x) = 0.5\sqrt{x}$, $K = 50$, $\alpha = 0$, $\gamma_0 = 0.1$
- Iteration data:

# epoch	step-size	batch norm	full norm
1	0.1	$1.2 \cdot 10^{6}$	$6.8 \cdot 10^{5}$
2	-	NaN	NaN
50	-	NaN	NaN
100	-	NaN	NaN
200	-	NaN	NaN
300	-	NaN	NaN
500	-	NaN	NaN
700	-	NaN	NaN
1000	-	NaN	NaN

- No adaptation to large gradients – Gradient explodes
- Diminishing step-size does of course not help

Step-size rules and convergence

- Classification, Residual layers, ReLU, 15x25,2,1 widths (17 layers)
- Step-size parameters: $\psi \equiv 0$, $\alpha = \gamma_0 = 0.1$
- Iteration data:

# epoch	step-size	batch norm	full norm
0	$1.4 \cdot 10^{-7}$	$7.0 \cdot 10^{6}$	$4.7 \cdot 10^{5}$
10	0.004	257	39.4
50	0.10	$6.2 \cdot 10^{-10}$	4.1
100	0.087	1.5	1.3
200	0.089	1.2	0.26
300	0.1	$2.0 \cdot 10^{-12}$	1.3
500	0.1	$5.1 \cdot 10^{-12}$	0.198
700	0.1	$2.4 \cdot 10^{-13}$	0.16
1000	0.087	1.5	0.013

- Large initial gradients dampened
- Larger final full norm than first choice since not diminishing γ_k

Outline

- Stochastic gradient descent
- Convergence and distance to solution
- Convergence and solution norms
- Overparameterized vs underparameterized setting
- Escaping not individually flat minima
- SGD step-sizes
- **SGD convergence**

Convergence analysis

- Need some inequality that function satisfies to analyze SGD
- Convexity inequality not applicable in deep learning
- Smoothness inequality not applicable in deep learning in general
  - ReLU networks are not differentiable and therefore not smooth
  - Tanh networks with smooth loss are cont. diff. ⇒ locally smooth
- We have seen that training problem is piece-wise polynomial if
  - L2 loss and piece-wise linear activation functions
  - hinge loss and piece-wise linear activation functions

  but does not provide an inequality for proving convergence

Error bound

- In absence of convexity, an *error bound* is useful in analysis:

$$\delta(f(x) - f(x^\star)) \leq \|\nabla f(x)\|_2^2$$

that holds locally around solution $x^\star$ with $\delta > 0$
- Gradient in error bound can be replaced by
  - sub-gradient for convex nondifferentiable f
  - limiting sub-gradient for nonconvex nondifferentiable f

Kurdyka-Lojasiewicz

- Error bound is instance of the Kurdyka-Lojasiewicz (KL) property
- KL property has exponent $\alpha \in [0, 1)$, $\alpha = \frac{1}{2}$ gives error bound
- Examples of KL functions:
  - Continuous (on closed domain) semialgebraic functions are KL:
    $$\mathrm{graph} f = \cup_{i=1}^{r} \left(\cap_{j=1}^{q} \{x : h_{ij}(x) = 0\} \cap_{l=1}^{p} \{x : g_{il}(x) < 0\} \right)$$
    graph is union of intersection, where h_{ij} and g_{il} polynomials
  - Continuous piece-wise polynomials (some DL training problems)
  - Strongly convex functions
- Often difficult to decide KL-exponent
- Result: descent methods on KL functions converge
  - sublinearly if $\alpha \in (\frac{1}{2}, 1)$
  - linearly if $\alpha \in (0, \frac{1}{2}]$ (the error bound regime)

Strongly convex functions satisfy error bound

- $s + \sigma x \in \partial f(x)$ with $s \in \partial g(x)$ for convex $g = f - \frac{\sigma}{2}\| \cdot \|_2^2$
- Therefore
  $$\begin{aligned} \|s + \sigma x\|_2^2 &= \|s\|_2^2 + 2\sigma s^T x + \sigma^2 \|x\|_2^2 \\ &\geq \|s\|_2^2 + 2\sigma s^T x^\star + 2\sigma(g(x) - g(x^\star)) + \sigma^2 \|x\|_2^2 \\ &= \|s\|_2^2 + 2\sigma s^T x^\star + \sigma \|x^\star\|_2^2 + 2\sigma(f(x) - f(x^\star)) \\ &= \|s + \sigma x^\star\|_2^2 + 2\sigma(f(x) - f(x^\star)) \\ &\geq 2\sigma(f(x) - f(x^\star)) \end{aligned}$$
  where we used
  - subgradient definition $g(x^\star) \geq g(x) + s^T(x^\star - x)$ in first inequality
  - nonnegativity of norms in the second inequality

Implications of error bound

- Restating error bound for differentiable case
  $$\delta(f(x) - f(x^\star)) \leq \|\nabla f(x)\|_2^2$$
- Assume it holds for all x in some ball X around solution $x^\star$
- What can you say about local minima and saddle-points in X?

Implications of error bound

- Restating error bound for differentiable case
  $$\delta(f(x) - f(x^\star)) \leq \|\nabla f(x)\|_2^2$$
- Assume it holds for all x in some ball X around solution $x^\star$
- What can you say about local minima and saddle-points in X?
- There are none! Proof by contradiction:
  - Assume local minima or saddle-point $\bar{x}$
  - Then $\nabla f(\bar{x}) = 0 \Rightarrow f(\bar{x}) = f(x^\star)$ and $\bar{x}$ is global minima

---

Convergence analysis – Smoothness and error bound

- Convergence analysis of gradient method
- β-smoothness and error bound assumptions ($f^\star = f(x^\star)$):
  $$\begin{aligned} f(x_{k+1}) - f^\star &\leq f(x_k) - f^\star + \nabla f(x_k)^T(x_{k+1} - x_k) + \tfrac{\beta}{2}\|x_k - x_{k+1}\|_2^2 \\ &= f(x_k) - f^\star - \gamma_k \|\nabla f(x_k)\|_2^2 + \tfrac{\beta \gamma_k^2}{2} \|\nabla f(x_k)\|_2^2 \\ &= f(x_k) - f^\star - \gamma_k (1 - \tfrac{\beta \gamma_k}{2}) \|\nabla f(x_k)\|_2^2 \\ &\leq (1 - \gamma_k \delta (1 - \tfrac{\beta \gamma_k}{2}))(f(x_k) - f^\star) \end{aligned}$$
  where
  - β-smoothness of f is used in first inequality
  - gradient update $x_{k+1} = x_k - \gamma_k \nabla f(x_k)$ in first equality
  - error bound is used in the final inequality
- Linear convergence in function values if $\gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$, $\epsilon > 0$

Semi-smoothness

- Typical DL training problems are not smooth
  - E.g.: overparameterized ReLU networks with smooth loss
- But semi-smooth[1] in neighborhood around random initialization[2]:
  $$f(x) \leq f(y) + \nabla f(y)^T (x - y) + c\|x - y\|_2 \sqrt{f(y)} + \tfrac{\beta}{2}\|x - y\|_2^2$$
  for some constants c and β
  - Holds locally for large enough c, β if cont. piece-wise polynomial
  - Constants and neighborhood quantified in [1][2]
- $c = 0$ gives smoothness
- c small gives close to smoothness but allows nondifferentiable

---

[1] Semismoothness definition not a standard semismoothness definition

[2] [1] A Convergence Theory for Deep Learning via Over-Parameterization. Z. Allen-Zhu et al.

Convergence – Error bound and semi-smoothness

- Convergence analysis of gradient descent method
- Assumptions: (c,β)-semi-smooth, δ-error bound, $f^\star = 0$ (w.l.o.g.)
- Parameters $c \leq \frac{\sqrt{\delta}\gamma\beta}{2}$ and $\gamma \in (0, \frac{1}{\beta})$:
  $$\begin{aligned} &f(x_{k+1}) \\ &\leq f(x_k) + \nabla f(x_k)^T(x_{k+1} - x_k) + c\|x_{k+1} - x_k\|\sqrt{f(x_k)} + \tfrac{\beta}{2}\|x_{k+1} - x_k\|_2^2 \\ &= f(x_k) - \gamma\|\nabla f(x_k)\|_2^2 + c\gamma\|\nabla f(x_k)\|\sqrt{f(x_k)} + \tfrac{\beta\gamma^2}{2}\|\nabla f(x_k)\|_2^2 \\ &\leq f(x_k) - \gamma\|\nabla f(x_k)\|_2^2 + \tfrac{c\gamma}{\sqrt{\delta}}\|\nabla f(x_k)\|^2 + \tfrac{\beta\gamma^2}{2}\|\nabla f(x_k)\|_2^2 \\ &\leq f(x_k) - \gamma\|\nabla f(x_k)\|_2^2 + \beta\gamma^2\|\nabla f(x_k)\|^2 \\ &\leq f(x_k) - \gamma(1 - \beta\gamma)\|\nabla f(x_k)\|_2^2 \\ &\leq (1 - c\gamma(1 - \beta\gamma))f(x_k) \end{aligned}$$
  which shows linear convergence to 0 loss
- Need the nonsmooth part of upper bound c to be small enough
- Can analyze SGD in similar manner

Convergence in deep learning

- Setting: ReLU network, fully connected, smooth loss
- c is small enough when model overparameterized enough [1][1]
- Linear convergence (with high prob.) for random initialization [1]
- In practice:
  - β will be big – relies on small enough ($\leq \frac{1}{\beta}$) constant step-size
  - need to find "correct" step-size by diminishing rule
  - need to control steps to not depart from linear convergence region
  - hopefully achieved by previous step-size rule

---

[1] [1] A Convergence Theory for Deep Learning via Over-Parameterization. Z. Allen-Zhu et al.

Stochastic Gradient Descent

Implicit Regularization

Pontus Giselsson

Outline

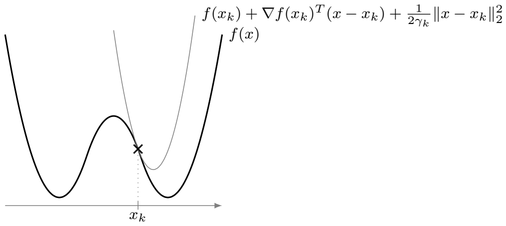
- **Variable metric methods**
- Convergence to projection point
- Convergence to sharp or flat minima

Gradient method interpretation

- Gradient method minimizes quadratic approximation of function

$$\begin{aligned} x_{k+1} &= \underset{x}{\operatorname{argmin}} \left(f(x_k) + \nabla f(x_k)^T(x - x_k) + \tfrac{1}{2\gamma_k}\|x - x_k\|_2^2 \right) \\ &= \underset{x}{\operatorname{argmin}} \left(\tfrac{1}{2\gamma_k}\|x - (x_k - \gamma_k \nabla f(x_k))\|_2^2 \right) \\ &= x_k - \gamma_k \nabla f(x_k) \end{aligned}$$

- Graphical illustration of one step

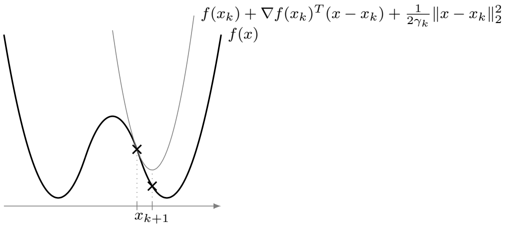


Gradient method interpretation

- Gradient method minimizes quadratic approximation of function

$$\begin{aligned} x_{k+1} &= \underset{x}{\operatorname{argmin}} \left(f(x_k) + \nabla f(x_k)^T(x - x_k) + \tfrac{1}{2\gamma_k}\|x - x_k\|_2^2 \right) \\ &= \underset{x}{\operatorname{argmin}} \left(\tfrac{1}{2\gamma_k}\|x - (x_k - \gamma_k \nabla f(x_k))\|_2^2 \right) \\ &= x_k - \gamma_k \nabla f(x_k) \end{aligned}$$

- Graphical illustration of one step



Scaled gradient method

- Quadratic approximation same in all directions due to $\|\cdot\|_2^2$

$$x_{k+1} = \underset{x}{\operatorname{argmin}} \left(f(x_k) + \nabla f(x_k)^T(x - x_k) + \tfrac{1}{2\gamma_k}\|x - x_k\|_2^2 \right)$$

- Scaled gradient method minimizes scaled quadratic approximation

$$\begin{aligned} x_{k+1} &= \underset{x}{\operatorname{argmin}} \left(f(x_k) + \nabla f(x_k)^T(x - x_k) + \tfrac{1}{2\gamma_k}\|x - x_k\|_H^2 \right) \\ &= \underset{x}{\operatorname{argmin}} \left(\tfrac{1}{2\gamma_k}\|x - (x_k - \gamma_k H^{-1} \nabla f(x_k))\|_H^2 \right) \\ &= x_k - \gamma_k H^{-1} \nabla f(x_k) \end{aligned}$$

  where H is a positive definite matrix and $\|x\|_H^2 = x^T H x$
- Nominal gradient method obtained by $H = I$
- Better quadratic approximation (good H) ⇒ faster convergence

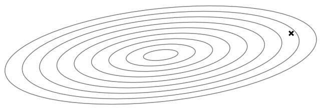
---

Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}} \; \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 0.1 & -0.1 \\ -0.1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Graphical illustration:

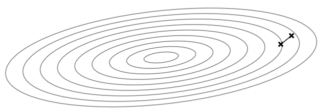


Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}} \; \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 0.1 & -0.1 \\ -0.1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Graphical illustration:

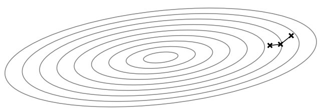


Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}} \; \frac{1}{2} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \begin{bmatrix} 0.1 & -0.1 \\ -0.1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Graphical illustration:

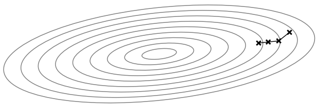


Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Graphical illustration:

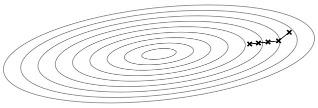


Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Graphical illustration:

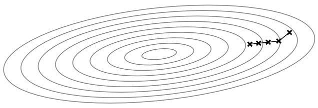


Gradient descent – Example

- (Unscaled) Gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Graphical illustration:

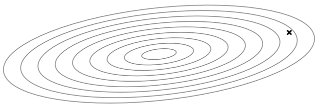


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:

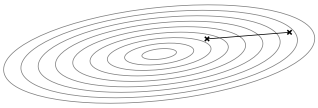


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:

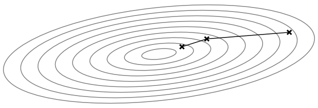


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:

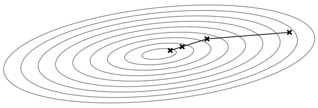


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:

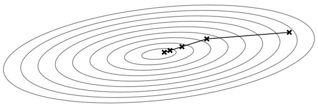


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:

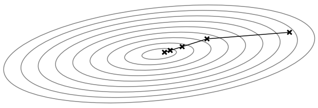


Scaled gradient descent – Example

- Scaled gradient descent on convex quadratic problem

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\begin{bmatrix}x_1\\x_2\end{bmatrix}^T\begin{bmatrix}0.1 & -0.1\\-0.1 & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}$$

- Scaling $H = \mathbf{diag}(\nabla^2 f) := P$:



How to select metric H?

- A priori: Use a fixed H thoughout iterations
  - can be difficult to find a good performing H
  - does not adapt to local geometry
- Adaptively: Iteration-dependent H_k that adapts to local geometry

---

Adaptive metric methods

- Algorithms with full H_k:
  - (Regularized) Newton methods
  - Quasi-Newton methods
- Algorithms with diagonal H_k (in stochastic setting):
  - Adagrad
  - RMSProp
  - Adam
  - Adamax/Adadelta
  - ...

SGD variations with adaptive diagonal scaling

- Diagonal scaling gives one step-size (learning rate) per variable
- SGD type methods with diagonal $H_k = \mathbf{diag}(h_{1,k}, \ldots, h_{N,k})$:

$$x_{k+1} = x_k - \gamma_k H_k^{-1} \widehat{\nabla} f(x_k)$$

  where
  - the inverse is $H_k^{-1} = \mathbf{diag}(\frac{1}{h_{1,k}}, \ldots, \frac{1}{h_{N,k}})$
  - $\widehat{\nabla} f(x_k)$ is a stochastic gradient approximation
- Methods called variable metric methods since H_k defines a metric
- Introduced to improve convergence compared to SGD
- Can have worse generalization properties?

Metrics – RMSprop and Adam

- Estimate coordinate-wise variance:

$$\hat{v}_k = b_v \hat{v}_{k-1} + (1 - b_v)(\widetilde{\nabla} f(x_{k-1}))^2$$

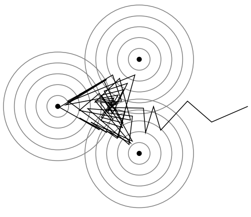
  where $\hat{v}_0 = 0$, $b_v \in (0, 1)$
- Metric H_k is chosen (approximately) as standard deviation:
  - RMSprop: biased estimate $H_k = \mathbf{diag}(\sqrt{\hat{v}_k} + \epsilon)$
  - Adam: unbiased estimate $H_k = \mathbf{diag}(\sqrt{\frac{\hat{v}_k}{1-b_v^k}} + \epsilon)$
- Intuition:
  - Reduce step size for high variance coordinates
  - Increase step size for low variance coordinates
- Alternative intuition:
  - Reduce step size for "steep" coordinate directions
  - Increase step size for "flat" coordinate directions

Filtered stochastic gradients

- Adam also filters stochastic gradients for smoother updates
- Let $\hat{m}_0 = 0$ and $b_m \in (0, 1)$, and update

$$\hat{m}_k = b_m \hat{m}_{k-1} + (1 - b_m)\widetilde{\nabla} f(x_{k-1})$$

- Adam uses unbiased estimate: $\frac{\hat{m}_k}{1-b_m^k}$
- Fixed step-size without filtered gradient



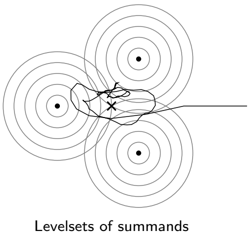
Levelsets of summands

Filtered stochastic gradients

- Adam also filters stochastic gradients for smoother updates
- Let $\hat{m}_0 = 0$ and $b_m \in (0, 1)$, and update

$$\hat{m}_k = b_m \hat{m}_{k-1} + (1 - b_m)\widetilde{\nabla} f(x_{k-1})$$

- Adam uses unbiased estimate: $\frac{\hat{m}_k}{1-b_m^k}$
- Fixed step-size with filtered gradient



Levelsets of summands

Adam – Summary

- Initialize $\hat{m}_0 = \hat{v}_0 = 0$, $b_m, b_v \in (0, 1)$, and select $\gamma > 0$
  1. $g_k = \widetilde{\nabla} f(x_{k-1})$ (stochastic gradient)
  2. $\hat{m}_k = b_m \hat{m}_{k-1} + (1 - b_m) g_k$
  3. $\hat{v}_k = b_v \hat{v}_{k-1} + (1 - b_v) g_k^2$
  4. $m_k = \hat{m}_k / (1 - b_m^k)$
  5. $v_k = \hat{v}_k / (1 - b_v^k)$
  6. $x_{k+1} = x_k - \gamma m_k./(\sqrt{v_k} + \epsilon\mathbf{1})$
- Suggested choices: $b_m = 0.9$, $b_v = 0.999$, $\epsilon = 10^{-8}$, $\gamma = 0.001$
- More succinctly

$$x_{k+1} = x_k - \gamma H_k^{-1} m_k$$

  where metric $H_k = \mathbf{diag}(\sqrt{v_{k,1}} + \epsilon, \ldots, \sqrt{v_{k,n}} + \epsilon)$

Adam vs SGD

- Adam designed to converge faster than SGD by adaptive scaling
- Often observed to give worse generalization than SGD
- Two possible reasons for worse generalization:
  - Convergence to larger norm solutions?
  - Convergence to sharper minima?

Outline

- Variable metric methods
- **Convergence to projection point**
- Convergence to sharp or flat minima

Generalization in neural networks

- Recall: Lipschitz constant L of neural network

$$L = \|W_n\|_2 \cdot \|W_{n-1}\|_2 \cdots \|W_1\|_2$$

  or with $\|W_j\|_2$ replaced by $(1 + \|W_j\|_2)$ for residual layers
- Can use $\|\theta\|_2$ where $\theta = \{(W_i, b_i)\}_{i=1}^n$ as proxy
- Overparameterized networks
  - Infinitely many solutions exist
  - Want a solution with small $\|\theta\|_2$ for good generalization

Explicit vs implicit regularization

- Tikhonov adds $\|\cdot\|_2^2$ norm penalty for better generalization

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i) + \tfrac{\lambda}{2}\|\theta\|_2^2$$

  which gives a smaller θ and is a form of explicit regularization
- Deep learning has no explicit regularization $\Rightarrow$ training problem:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

  with many 0-loss solutions in overparameterized setting
- Implicit regularization if algorithm finds small norm solution

(S)GD limit points

- Assume overparameterized convex least squares problem
- Gradient descent converges to projection point of initial point
- If SGD converges, it converges to same projection point

Least squares

- Consider least squares problem of the form

$$\underset{x}{\text{minimize}}\, \tfrac{1}{2}\|Ax - b\|_2^2$$

  where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $m < n$, and $\exists \bar{x}$ such that $A\bar{x} = b$
- Problem is overparameterized and has many solutions
- Since $m < n$, solution set is

$$X := \{x : Ax = b\}$$

  which is (at least) $n - m$-dimensional affine set

Gradient method convergence to projection point

- Will show that scaled gradient method

$$x_{k+1} = x_k - \gamma_k H^{-1}\nabla f(x_k)$$

  converges to $\|\cdot\|_H$-norm projection onto solution set from x_0
- Means that scaled gradient method converges to solution of

$$\begin{array}{ll}\text{minimize}_x & \|x - x_0\|_H^2 \\ \text{subject to} & Ax = b\end{array}$$

  where H decides metric in which to measure distance from x_0
- If $x_0 = 0$, we get minimum $\|\cdot\|_H$-norm solution in $\{x : Ax = b\}$

Characterizing projection point

- The unique projection point $\hat{x} = \underset{x\in X}{\text{argmin}}(\|x - x_0\|_H^2)$ if and only if

$$H\hat{x} - Hx_0 \in \mathcal{R}(A^T) \qquad \text{and} \qquad A\hat{x} = b$$

  where $\mathcal{R}(A^T)$ is the range space of A^T
- The range space is $\mathcal{R}(A^T) = \{v \in \mathbb{R}^n : v = A^T\lambda \text{ and } \lambda \in \mathbb{R}^m\}$

Convergence to projection point

- The scaled gradient method can be written as

$$Hx_{k+1} = Hx_k - \gamma_k A^T(Ax_k - b),$$

if all $\gamma_k > \epsilon > 0$ are small enough, it converges to a solution $\bar{x}$:

$$x_k \to \bar{x} \qquad \text{and} \qquad A\bar{x} = b$$

- Letting $\lambda_k = -\sum_{l=0}^{k} \gamma_l(Ax_l - b) \in \mathbb{R}^m$ and unfolding iteration:

$$Hx_{k+1} - Hx_0 = -\sum_{l=0}^{k} \gamma_l A^T(Ax_l - b) = A^T\lambda_k \in \mathcal{R}(A^T)$$

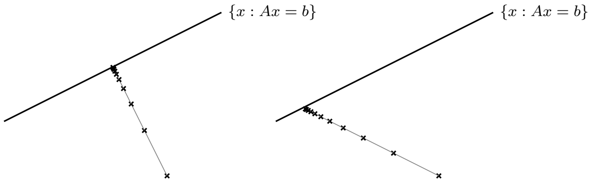
- In the limit $x_k \to \bar{x}$, we get

$$H\bar{x} - Hx_0 \in \mathcal{R}(A^T)$$

which with $A\bar{x} = b$ gives optimality conditions for projection
- If $x_0 = 0$, the algorithm converges to $\underset{x \in X}{\operatorname{argmin}}(\|x\|_H)$

Graphical interpretation

- What happens with scaled gradient method?
- Solution set X extends infinitely
  - sequence is perpendicular to X in scalar product $(Hx)^T y$
  - algorithm converges to projection point $\operatorname{argmin}_{x \in X}(\|x - x_0\|_H)$



Gradient method
A scaled gradient method

SGD – Convergence to projection point

- Least squares problem on finite sum form

$$\underset{x}{\text{minimize}}\ \tfrac{1}{2}\|Ax - b\|_2^2 = \tfrac{1}{2}\sum_{i=1}^{m}(a_i^T x - b_i)^2$$

where $A = [a_1, \ldots, a_m]^T$
- Applying single-batch scaled SGD:

$$x_{k+1} = x_k - \gamma_k H^{-1} a_{i_k}(a_{i_k}^T x_k - b_{i_k})$$

- The iteration can be unfolded as

$$Hx_{k+1} - Hx_0 = -\sum_{l=0}^{k} a_{i_l}\gamma_l(a_{i_l}^T x_l - b_{i_l}) = A^T \begin{bmatrix} -\sum_{l=0}^{k} \underset{i_l=1}{\chi}(\gamma_l(a_1^T x_l - b_1)) \\ \vdots \\ -\sum_{l=0}^{k} \underset{i_l=m}{\chi}(\gamma_l(a_m^T x_l - b_m)) \end{bmatrix}$$

where $\underset{i_l=j}{\chi}(v) = v$ if $i_l = j$, else 0, so $Hx_{k+1} - Hx_0 \in \mathcal{R}(A^T)$
- Assume $x_k \to \bar{x}$ with $A\bar{x} = b \Rightarrow$ convergence to projection point

SGD vs Adam

This analysis hints towards that SGD gives smaller norm solutions and better generalization than variable metric Adam. Is this true?

How about deep learning?

- The analysis does not carry over to nonconvex DL settings
- However, often convergence to similar norm as initial point

How to select initial point?

- For standard networks:
  - To avoid vanishing and exploding gradient, we want:

$$L\|W_j\|_2 \approx 1 \qquad \text{and} \qquad \|b_j\|_2 \text{ small}$$

where L is average activation Lipschitz constant ($L = 0.5$ for ReLU)
  - Initialization for ReLU:
    - $(W_j)_{il} \sim \mathcal{N}(0, \frac{2}{\sqrt{m_j n_j}})$ gives average $\|W_j\|_2 = 2$
    - $(b_j)_i$ small or 0
- For residual networks:
  - To avoid vanishing and exploding gradient, we want

$$L(1 + \|W_j\|_2) \approx 1 \qquad \text{and} \qquad \|b_j\|_2 \text{ small}$$

where L is average activation Lipschitz constant
  - Use smaller initilization than for standard networks

Initialization in next example

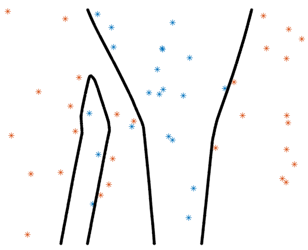
- Set scaling of weights by σ
- For the residual layers (all square layers)
  - $(W_j)_{ij} \sim \mathcal{N}(0, 1)$, normalize W_j, scale by σ
  - $(b_j)_i \sim \mathcal{N}(0, 1)$, normalize b_j, scale by σ
- For the non-residual layers (non-square layers)
  - $(W_j)_{ij} \sim \mathcal{N}(0, 1)$, normalize W_j, scale by $\max(1, \sigma)$
  - $(b_j)_i \sim \mathcal{N}(0, 1)$, normalize b_j, scale by $\max(1, \sigma)$
  - use $\max(1, \sigma)$ for gradient to not vanish in non-residual layers

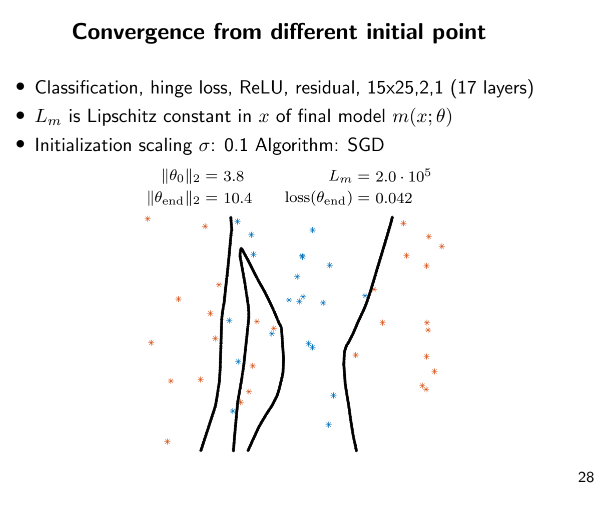
Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x; \theta)$
- Initialization scaling σ: 0.01 Algorithm: SGD

$\|\theta_0\|_2 = 3.57$ $\quad L_m = 8.4 \cdot 10^4$

$\|\theta_{\text{end}}\|_2 = 9.9$ $\quad \text{loss}(\theta_{\text{end}}) = 0.051$

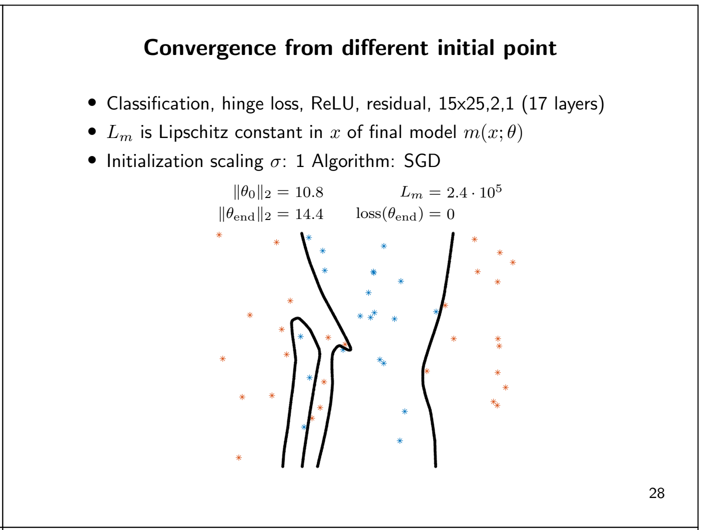




Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 0.1 Algorithm: SGD

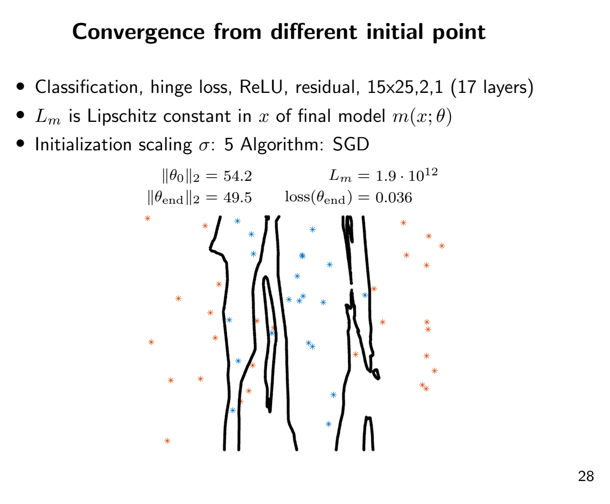
$\|\theta_0\|_2 = 3.8$ $L_m = 2.0 \cdot 10^5$
$\|\theta_{\text{end}}\|_2 = 10.4$ $\text{loss}(\theta_{\text{end}}) = 0.042$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 1 Algorithm: SGD

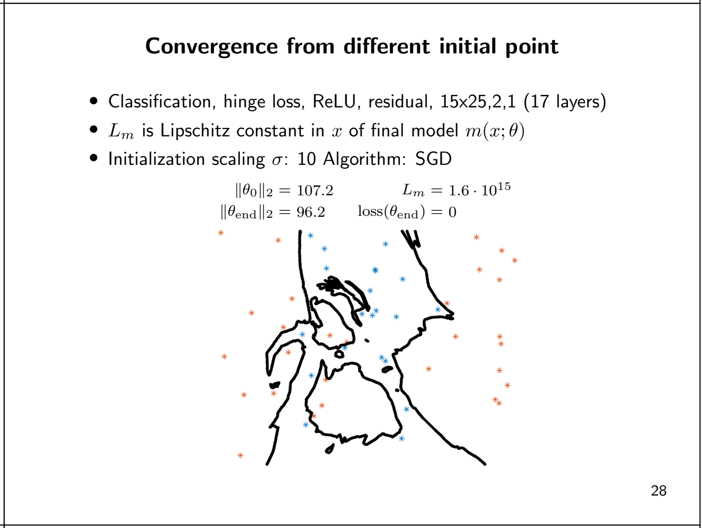
$\|\theta_0\|_2 = 10.8$ $L_m = 2.4 \cdot 10^5$
$\|\theta_{\text{end}}\|_2 = 14.4$ $\text{loss}(\theta_{\text{end}}) = 0$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 5 Algorithm: SGD

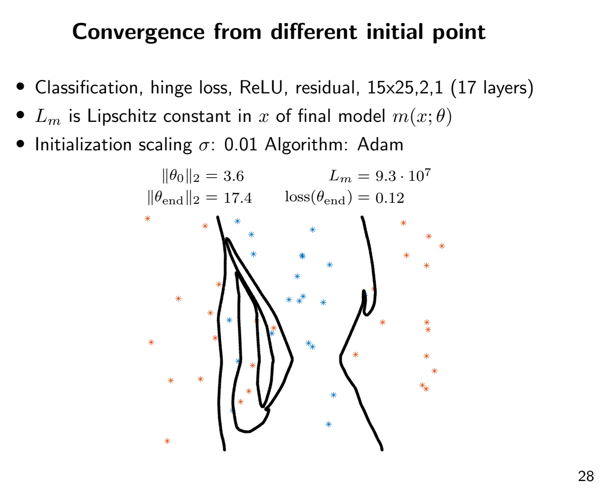
$\|\theta_0\|_2 = 54.2$ $L_m = 1.9 \cdot 10^{12}$
$\|\theta_{\text{end}}\|_2 = 49.5$ $\text{loss}(\theta_{\text{end}}) = 0.036$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 10 Algorithm: SGD

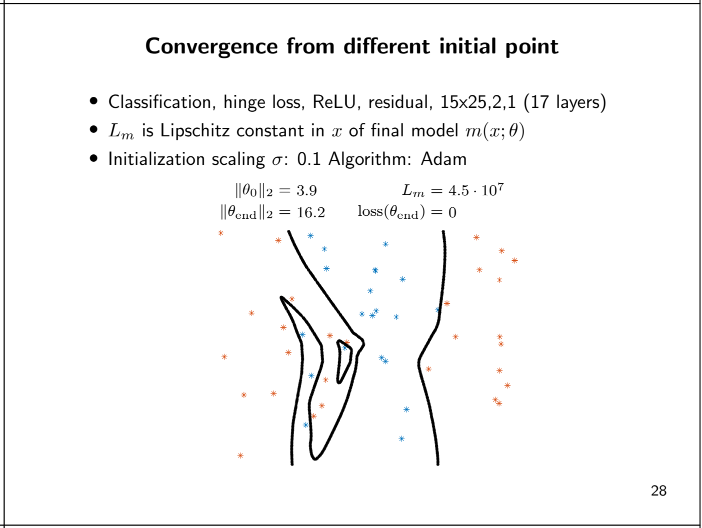
$\|\theta_0\|_2 = 107.2$ $L_m = 1.6 \cdot 10^{15}$
$\|\theta_{\text{end}}\|_2 = 96.2$ $\text{loss}(\theta_{\text{end}}) = 0$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 0.01 Algorithm: Adam

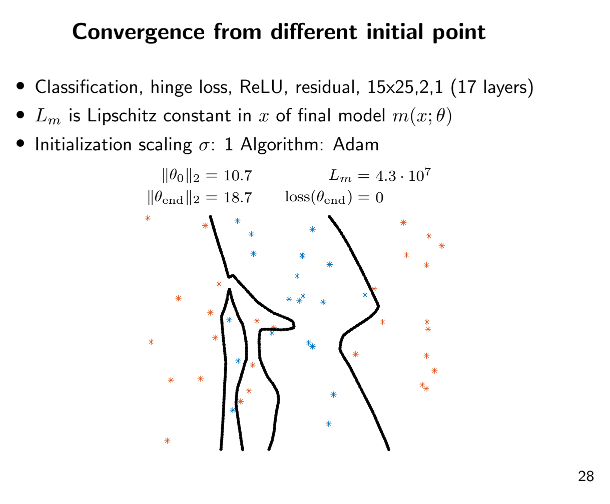
$\|\theta_0\|_2 = 3.6$ $L_m = 9.3 \cdot 10^7$
$\|\theta_{\text{end}}\|_2 = 17.4$ $\text{loss}(\theta_{\text{end}}) = 0.12$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 0.1 Algorithm: Adam

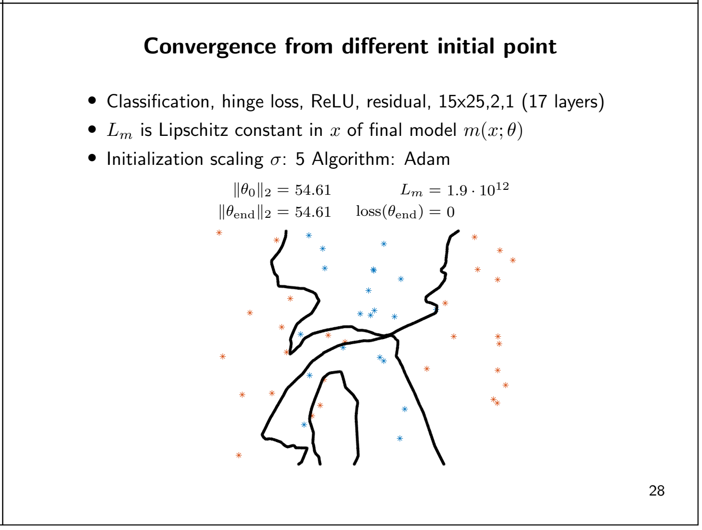
$\|\theta_0\|_2 = 3.9$ $L_m = 4.5 \cdot 10^7$
$\|\theta_{\text{end}}\|_2 = 16.2$ $\text{loss}(\theta_{\text{end}}) = 0$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 1 Algorithm: Adam

$\|\theta_0\|_2 = 10.7$ $L_m = 4.3 \cdot 10^7$
$\|\theta_{\text{end}}\|_2 = 18.7$ $\text{loss}(\theta_{\text{end}}) = 0$



Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 5 Algorithm: Adam

$\|\theta_0\|_2 = 54.61$ $L_m = 1.9 \cdot 10^{12}$
$\|\theta_{\text{end}}\|_2 = 54.61$ $\text{loss}(\theta_{\text{end}}) = 0$

Convergence from different initial point

- Classification, hinge loss, ReLU, residual, 15x25,2,1 (17 layers)
- L_m is Lipschitz constant in x of final model $m(x;\theta)$
- Initialization scaling σ: 10 Algorithm: Adam

$\|\theta_0\|_2 = 109.278$ $\quad L_m = 3.8 \cdot 10^{16}$

$\|\theta_{\text{end}}\|_2 = 109.282$ $\quad \text{loss}(\theta_{\text{end}}) = 0$

Conclusions

- Choice of initial point is significant for generalization
- Here, Adam gives models with larger Lipschitz constant L_m

	Adam			SGD		
scaling σ	$\|\theta_0\|_2$	$\|\theta_{\text{end}}\|_2$	L_m	$\|\theta_0\|_2$	$\|\theta_{\text{end}}\|_2$	L_m
0.01	3.6	17.4	$9.3 \cdot 10^7$	3.57	9.9	$8.4 \cdot 10^4$
0.1	3.9	16.2	$4.5 \cdot 10^7$	3.8	10.4	$2.0 \cdot 10^5$
1	10.7	18.7	$4.3 \cdot 10^7$	10.8	14.4	$2.4 \cdot 10^5$
5	54.61	54.61	$1.9 \cdot 10^{12}$	54.2	49.5	$1.9 \cdot 10^{12}$
10	109.278	109.282	$3.8 \cdot 10^{16}$	107.2	96.2	$1.6 \cdot 10^{15}$

Outline

- Variable metric methods
- Convergence to projection point
- **Convergence to sharp or flat minima**

Convergence to sharp or flat minima

- Have argued flat minima generalize well, sharp minima poorly
- Is Adam or SGD most likely to converge to sharp minimum?

Variable metric methods – Interpretation

- Variable metric methods

$$x_{k+1} = x_k - \gamma_k H_k^{-1} \nabla f(x_k) \qquad (1)$$

can be interpreted as taking pure (stochastic) gradient step on

$$f_{H_k} = (f \circ H_k^{-1/2})(x)$$

- Why? Gradient method on f_{H_k} is

$$v_{k+1} = v_k - \gamma_k \nabla f_{H_k}(v_k) = v_k - \gamma_k H_k^{-1/2} f(H_k^{-1/2} v_k)$$

which after
  - multiplication with $H^{-1/2}$
  - and change of variables according to $x_k = H_k^{-1/2} v_k$

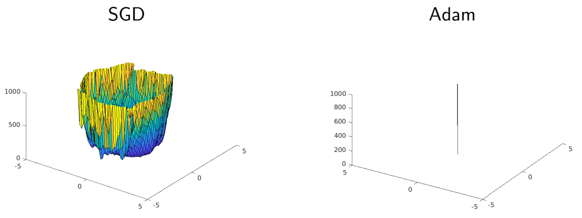
gives (1)

Interpretation consequence

- Variable metric methods choose H_k to make f_{H_k} well conditioned
- Consequences:
  - Sharp minima in f become less sharp in f_{H_k}
  - (Flat minima in f become less flat in f_{H_k})
- Adam maybe more likely to converge to sharp minima than SGD
- This can be a reason for worse generalization in Adam than SGD

Adam vs SGD – Flat or sharp minima

- Data from previous classification example with $\sigma = 10$
- Loss landscape around final point θ_{end} for SGD and Adam
- SGD and Adam reach 0 loss but Adam minimum much sharper
- Same θ_1, θ_2 directions, same axes, $z_{\max} = 1000$

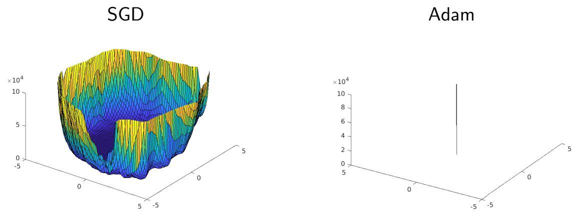


SGD

Adam

Adam vs SGD – Flat or sharp minima

- Data from previous classification example with $\sigma = 10$
- Loss landscape around final point θ_{end} for SGD and Adam
- SGD and Adam reach 0 loss but Adam minimum much sharper
- Same θ_1, θ_2 directions, same axes, $z_{\max} = 100000$



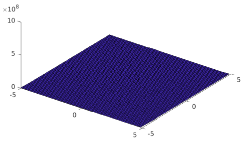
SGD

Adam

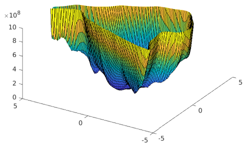
Adam vs SGD – Flat or sharp minima

- Data from previous classification example with $\sigma = 10$
- Loss landscape around final point θ_{end} for SGD and Adam
- SGD and Adam reach 0 loss but Adam minimum much sharper
- Same θ_1, θ_2 directions, same axes, $z_{\max} = 10^9$

SGD



Adam



Recap

Pontus Giselsson

Outline

- Convex analysis
- Composite optimization and duality
- Solving composite optimization problems – Algorithms

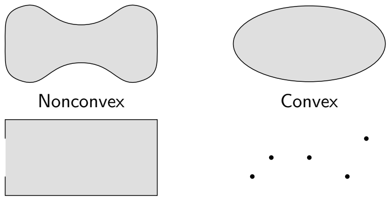
Convex Analysis

Convex sets

- A set C is convex if for every $x, y \in C$ and $\theta \in [0, 1]$:

$$\theta x + (1 - \theta)y \in C$$

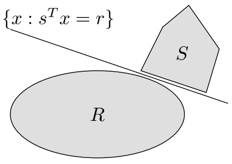
- "Every line segment that connect any two points in C is in C"



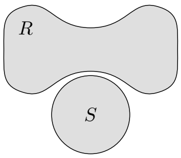
- Will assume that all sets are nonempty and closed

Separating hyperplane theorem

- Suppose that $R, S \subseteq \mathbb{R}^n$ are two non-intersecting convex sets
- Then there exists hyperplane with S and R in opposite halves



Example



Counter-example
R nonconvex

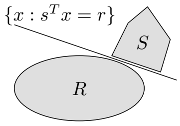
- Mathematical formulation: There exists $s \neq 0$ and r such that

$$\begin{aligned} s^T x &\le r && \text{for all } x \in R \\ s^T x &\ge r && \text{for all } x \in S \end{aligned}$$

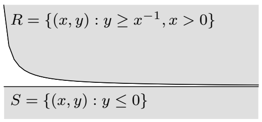
- The hyperplane $\{x : s^T x = r\}$ is called *separating hyperplane*

A strictly separating hyperplane theorem

- Suppose that $R, S \subseteq \mathbb{R}^n$ are non-intersecting closed and convex sets and that one of them is compact (closed and bounded)
- Then there exists hyperplane with strict separation



Example



Counter example
R, S not compact

- Mathematical formulation: There exists $s \neq 0$ and r such that

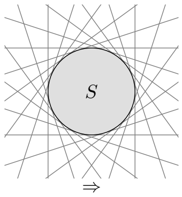
$$\begin{aligned} s^T x &< r && \text{for all } x \in R \\ s^T x &> r && \text{for all } x \in S \end{aligned}$$

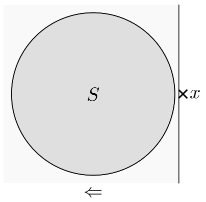
Consequence – S is intersection of halfspaces

$\boxed{\text{a closed convex set } S \text{ is the intersection of all halfspaces that contain it}}$

proof:

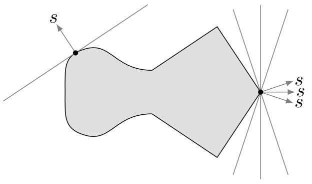
- let H be the intersection of all halfspaces containing S
- $\Rightarrow$: obviously $x \in S \Rightarrow x \in H$
- $\Leftarrow$: assume $x \notin S$, since S closed and convex and x compact (a point), there exists a strictly separating hyperplane, i.e., $x \notin H$:





Supporting hyperplanes

- Supporting hyperplanes touch set and have full set on one side:



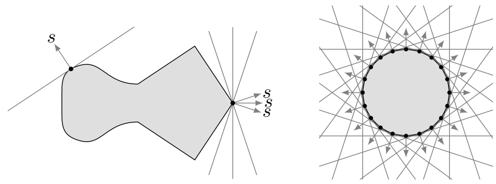
- We call the halfspace that contains the set *supporting halfspace*
- s is called *normal vector* to S at x
- Definition: Hyperplane $\{y : s^T y = r\}$ supports S at $x \in$ bd S if

$$s^T y \le r \text{ for all } y \in S \qquad \text{and} \qquad s^T x = r$$

Supporting hyperplane theorem

Let S be a nonempty convex set and let $x \in \mathrm{bd}(S)$. Then there exists a supporting hyperplane to S at x.

- Does not exist for all point on boundary for nonconvex sets
- Many supporting hyperplanes exist for points of nonsmoothness



Connection to duality and subgradients

Supporting hyperplanes are at the core of convex analysis:

- Subgradients define supporting hyperplanes to $\mathrm{epi}f$
- Conjugate functions define supporting hyperplanes to $\mathrm{epi}f$
- Duality is based on subgradients, hence supporting hyperplanes:
  - Consider $\mathrm{minimize}_x(f(x) + g(x))$ and primal solution $x^\star$
  - Dual problem $\mathrm{minimize}_\mu(f^*(\mu) + g^*(-\mu))$ solution $\mu^\star$ satisfies

$$\mu^\star \in \partial f(x^\star) \qquad\qquad -\mu^\star \in \partial g(x^\star)$$

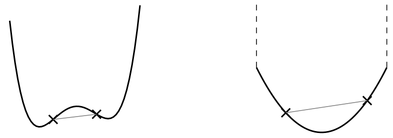
i..e, dual problem finds subgradients at optimal point[1]

---

[1] When solving $\min_x(f(Lx) + g(x))$ dual problem finds μ such that $L^T\mu \in \partial(f \circ L)(x)$ and $-L^T\mu \in \partial g(x)$.

Convex functions

- Graph below line connecting any two pairs $(x, f(x))$ and $(y, f(y))$



nonconvex function convex function

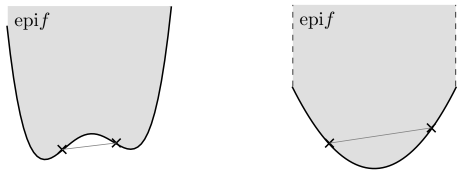
- Function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is *convex* if for all $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

  (in extended valued arithmetics)
- A function f is *concave* if $-f$ is convex

Epigraphs and convexity

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$
- Then f is convex if and only $\mathrm{epi}f$ is a convex set in $\mathbb{R}^n \times \mathbb{R}$



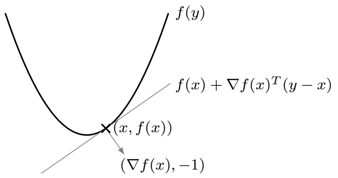
- f is called closed (lower semi-continuous) if $\mathrm{epi}f$ is closed set

First-order condition for convexity

- A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if

$$f(y) \geq f(x) + \nabla f(x)^T(y - x)$$

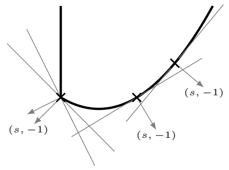
  for all $x, y \in \mathbb{R}^n$



- Function f has for all $x \in \mathbb{R}^n$ an affine minorizer that:
  - has slope s defined by ∇f
  - coincides with function f at x
  - is supporting hyperplane to epigraph of f
  - defines normal $(\nabla f(x), -1)$ to epigraph of f

Subdifferentials and subgradients

- Subgradients s define affine minorizers to the function that:



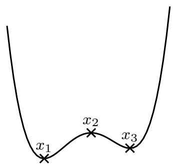
  - coincide with f at x
  - define normal vector $(s, -1)$ to epigraph of f
  - can be one of many affine minorizers at nondifferentiable points x
- Subdifferential of $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ at x is set of vectors s satisfying

$$f(y) \geq f(x) + s^T(y - x) \quad \text{for all } y \in \mathbb{R}^n, \qquad (1)$$

- Notation:
  - subdifferential: $\partial f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ (power-set notation $2^{\mathbb{R}^n}$)
  - subdifferential at x: $\partial f(x) = \{s : (1) \text{ holds}\}$
  - elements $s \in \partial f(x)$ are called *subgradients* of f at x

Subgradient existence – Nonconvex example

- Function can be differentiable at x but $\partial f(x) = \emptyset$



  - x_1: $\partial f(x_1) = \{0\}$, $\nabla f(x_1) = 0$
  - x_2: $\partial f(x_2) = \emptyset$, $\nabla f(x_2) = 0$
  - x_3: $\partial f(x_3) = \emptyset$, $\nabla f(x_3) = 0$
- Gradient is a local concept, subdifferential is a global property

Existence for extended-valued convex functions

- Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ be convex, then:
  1. Subgradients exist for all x in relative interior of $\mathrm{dom}f$
  2. Subgradients sometimes exist for x on boundary of $\mathrm{dom}f$
  3. No subgradient exists for x outside $\mathrm{dom}f$
- Examples for second case, boundary points of $\mathrm{dom}f$:

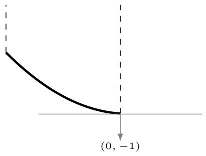
$-\sqrt{1-x^2} + \iota_{[-1,1]}(x)$ $x^2 + \iota_{[-2,2]}(x)$

- No subgradient (affine minorizer) exists for left function at $x = 1$

Fermat's rule

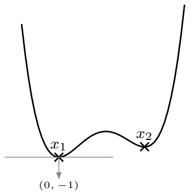
Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, then x minimizes f if and only if
$$0 \in \partial f(x)$$

- Proof: x minimizes f if and only if
$$f(y) \geq f(x) + 0^T(y - x) \quad \text{for all } y \in \mathbb{R}^n$$
which by definition of subdifferential is equivalent to $0 \in \partial f(x)$
- Example: several subgradients at solution, including 0



Fermat's rule – Nonconvex example

- Fermat's rule holds also for nonconvex functions
- Example:



  - $\partial f(x_1) = 0$ and $\nabla f(x_1) = 0$ (global minimum)
  - $\partial f(x_2) = \emptyset$ and $\nabla f(x_2) = 0$ (local minimum)
- For nonconvex f, we can typically only hope to find local minima

Subdifferential calculus rules

- Subdifferential of sum $\partial(f_1 + f_2)$
- Subdifferential of composition with matrix $\partial(g \circ L)$

Subdifferential of sum

If f_1, f_2 closed convex and $\operatorname{relint} \operatorname{dom} f_1 \cap \operatorname{relint} \operatorname{dom} f_2 \neq \emptyset$:
$$\partial(f_1 + f_2) = \partial f_1 + \partial f_2$$

- One direction always holds: if $x \in \text{dom}\partial f_1 \cap \text{dom}\partial f_2$:
$$\partial(f_1 + f_2)(x) \supseteq \partial f_1(x) + \partial f_2(x)$$
Proof: let $s_i \in \partial f_i(x)$, add subdifferential definitions:
$$f_1(y) + f_2(y) \geq f_1(x) + f_2(x) + (s_1 + s_2)^T(y - x)$$
i.e. $s_1 + s_2 \in \partial(f_1 + f_2)(x)$
- If f_1 and f_2 differentiable, we have (without convexity of f)
$$\nabla(f_1 + f_2) = \nabla f_1 + \nabla f_2$$

Subdifferential of composition

If f closed convex and $\operatorname{relint} \operatorname{dom}(f \circ L) \neq \emptyset$:
$$\partial(f \circ L)(x) = L^T \partial f(Lx)$$

- One direction always holds: If $Lx \in \text{dom} f$, then
$$\partial(f \circ L)(x) \supseteq L^T \partial f(Lx)$$
Proof: let $s \in \partial f(Lx)$, then by definition of subgradient of f:
$$(f \circ L)(y) \geq (f \circ L)(x) + s^T(Ly - Lx) = (f \circ L)(x) + (L^T s)^T(y - x)$$
i.e., $L^T s \in \partial(f \circ L)(x)$
- If f differentiable, we have chain rule (without convexity of f)
$$\nabla(f \circ L)(x) = L^T \nabla f(Lx)$$

A sufficient optimality condition

Let $f : \mathbb{R}^m \to \overline{\mathbb{R}}$, $g : \mathbb{R}^n \to \overline{\mathbb{R}}$, and $L \in \mathbb{R}^{m \times n}$ then:
$$\operatorname{minimize} f(Lx) + g(x) \tag{1}$$
is solved by every $x \in \mathbb{R}^n$ that satisfies
$$0 \in L^T \partial f(Lx) + \partial g(x) \tag{2}$$

- Subdifferential calculus inclusions say:
$$0 \in L^T \partial f(Lx) + \partial g(x) \subseteq \partial((f \circ L)(x) + g(x))$$
which by Fermat's rule is equivalent to x solution to (1)
- Note: (1) can have solution but no x exists that satisfies (2)

A necessary and sufficient optimality condition

Let $f : \mathbb{R}^m \to \overline{\mathbb{R}}$, $g : \mathbb{R}^n \to \overline{\mathbb{R}}$, $L \in \mathbb{R}^{m \times n}$ with f, g closed convex and assume $\operatorname{relint} \operatorname{dom}(f \circ L) \cap \operatorname{relint} \operatorname{dom} g \neq \emptyset$ then:
$$\operatorname{minimize} f(Lx) + g(x) \tag{1}$$
is solved by $x \in \mathbb{R}^n$ if and only if x satisfies
$$0 \in L^T \partial f(Lx) + \partial g(x) \tag{2}$$

- Subdifferential calculus equality rules say:
$$0 \in L^T \partial f(Lx) + \partial g(x) = \partial((f \circ L)(x) + g(x))$$
which by Fermat's rule is equivalent to x solution to (1)
- Algorithms search for x that satisfy $0 \in L^T \partial f(Lx) + \partial g(x)$

Evaluating subgradients of convex functions

- Obviously need to evaluate subdifferentials to solve
$$0 \in L^T \partial f(Lx) + \partial g(x)$$
- Explicit evaluation:
  - If function is differentiable: ∇f (unique)
  - If function is nondifferentiable: compute element in ∂f
- Implicit evaluation:
  - Proximal operator (specific element of subdifferential)

Proximal operator

- Proximal operator of (convex) g defined as:

$$\operatorname{prox}_{\gamma g}(z) = \underset{x}{\operatorname{argmin}}(g(x) + \tfrac{1}{2\gamma}\|x - z\|_2^2)$$

where $\gamma > 0$ is a parameter
- Evaluating prox requires solving optimization problem
- Objective is strongly convex $\Rightarrow$ solution exists and is unique

Prox evaluates the subdifferential

- Fermat's rule on prox definition: $x = \operatorname{prox}_{\gamma g}(z)$ if and only if

$$0 \in \partial g(x) + \gamma^{-1}(x - z) \quad \Leftrightarrow \quad \gamma^{-1}(z - x) \in \partial g(x)$$

Hence, $\gamma^{-1}(z - x)$ is element in $\partial g(x)$
- A subgradient in $\partial g(x)$ where $x = \operatorname{prox}_{\gamma g}(z)$ is computed
- Often used in algorithms when g nonsmooth (no gradient exists)

Conjugate functions

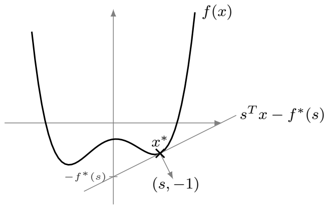
- The conjugate function of $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is defined as

$$f^*(s) := \sup_x \left(s^T x - f(x)\right)$$

- Implicit definition via optimization problem

Conjugate interpretation

- Conjugate $f^*(s)$ defines affine minorizer to f with slope s:



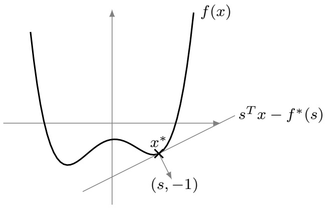
where $f^*(s)$ decides the constant offset to have support at x^*
- "Affine minorizor generator: Pick slope s, get offset for support"
- Why? Consider $f^*(s) = \sup_x \left(s^T x - f(x)\right)$ with maximizer x^*:

$$\begin{aligned} f^*(s) = s^T x^* - f(x^*) \quad &\Leftrightarrow \quad f^*(s) \geq s^T x - f(x) \text{ for all } x \\ &\Leftrightarrow \quad f(x) \geq s^T x - f^*(s) \text{ for all } x \end{aligned}$$

- Support at x^* since $f(x^*) = s^T x^* - f^*(s)$

Fenchel Young's equality

- Going back to conjugate interpretation:



- Fenchel's inequality: $f(x) \geq s^T x - f^*(s)$ for all x, s
- Fenchel-Young's equality and equivalence:

$$\boxed{f(x^*) = s^T x^* - f^*(s) \text{ holds if and only if } s \in \partial f(x^*)}$$

A subdifferential formula

$$\boxed{\text{Assume } f \text{ closed convex, then } \partial f(x) = \operatorname{Argmax}_s(s^T x - f^*(s))}$$

- Since $f^{**} = f$, we have $f(x) = \sup_s(x^T s - f^*(s))$ and

$$\begin{aligned} s^* \in \underset{s}{\operatorname{Argmax}}(x^T s - f^*(s)) \quad &\Longleftrightarrow \quad f(x) = x^T s^* - f^*(s^*) \\ &\Longleftrightarrow \quad s^* \in \partial f(x) \end{aligned}$$

- The last equivalence is Fenchel-Young

Subdifferential of conjugate – Inversion formula

$$\boxed{\text{Suppose } f \text{ closed convex, then } s \in \partial f(x) \Longleftrightarrow x \in \partial f^*(s)}$$

- Consequence of Fenchel-Young
- Another way to write the result is that for closed convex f:

$$\partial f^* = (\partial f)^{-1}$$

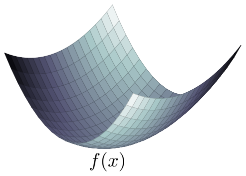
(Definition of inverse of set-valued A: $x \in A^{-1}u \Longleftrightarrow u \in Ax$)

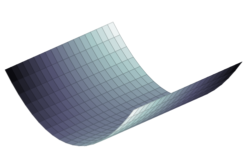
Strong convexity

- Let $\sigma > 0$
- A function f is *σ-strongly convex* if $f - \frac{\sigma}{2}\|\cdot\|_2^2$ is convex
- Alternative equivalent definition of σ-strong convexity:

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) - \tfrac{\sigma}{2}\theta(1-\theta)\|x - y\|^2$$

holds for every $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$
- Strongly convex functions are strictly convex and convex
- Example: f 2-strongly convex since $f - \|\cdot\|_2^2$ convex:


$f(x)$

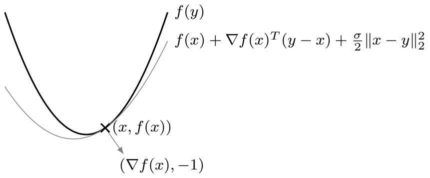

$f(x) - \|x\|_2^2$

First-order condition for strong convexity

- Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable
- f is σ-strongly convex with $\sigma > 0$ if and only if

$$f(y) \geq f(x) + \nabla f(x)^T(y-x) + \tfrac{\sigma}{2}\|x-y\|_2^2$$

for all $x, y \in \mathbb{R}^n$



- Function f has for all $x \in \mathbb{R}^n$ a quadratic minorizer that:
  - has curvature defined by σ
  - coincides with function f at x
  - defines normal $(\nabla f(x), -1)$ to epigraph of f

Smoothness

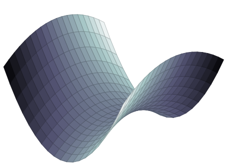
- A function is called β-*smooth* if its gradient is β-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq \beta\|x-y\|_2$$

for all $x, y \in \mathbb{R}^n$ (it is not necessarily convex)
- Alternative equivalent definition of β-smoothness

$$f(\theta x + (1-\theta)y) \geq \theta f(x) + (1-\theta)f(y) - \tfrac{\beta}{2}\theta(1-\theta)\|x-y\|^2$$
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) + \tfrac{\beta}{2}\theta(1-\theta)\|x-y\|^2$$

hold for every $x, y \in \mathbb{R}^n$ and $\theta \in [0,1]$
- Smoothness does not imply convexity
- Example:

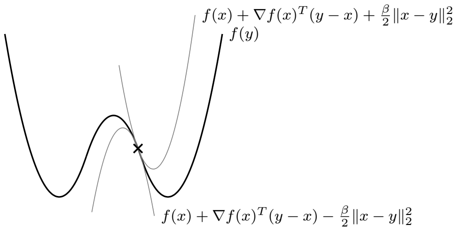


First-order condition for smoothness

- f is β-smooth with $\beta \geq 0$ if and only if

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x) - \tfrac{\beta}{2}\|x-y\|_2^2$$

for all $x, y \in \mathbb{R}^n$



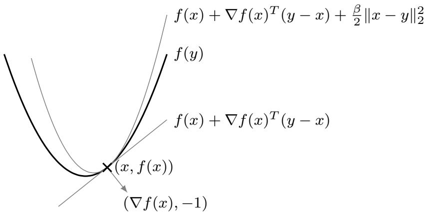
- Quadratic upper/lower bounds with curvatures defined by β
- Quadratic bounds coincide with function f at x

First-order condition for smooth convex

- f is β-smooth with $\beta \geq 0$ and convex if and only if

$$f(y) \leq f(x) + \nabla f(x)^T(y-x) + \tfrac{\beta}{2}\|x-y\|_2^2$$
$$f(y) \geq f(x) + \nabla f(x)^T(y-x)$$

for all $x, y \in \mathbb{R}^n$



- Quadratic upper bound and affine lower bound
- Bounds coincide with function f at x
- Quadratic upper bound is called *descent lemma*

Duality correspondance

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$. Then the following are equivalent:

(i) f is closed and σ-strongly convex
(ii) ∂f is maximally monotone and σ-strongly monotone
(iii) ∇f^* is σ-cocoercive
(iv) ∇f^* is maximally monotone and $\frac{1}{\sigma}$-Lipschitz continuous
(v) f^* is closed convex and satisfies descent lemma (is $\frac{1}{\sigma}$-smooth)

where $\nabla f^* : \mathbb{R}^n \to \mathbb{R}^n$ and $f^* : \mathbb{R}^n \to \mathbb{R}$

Comments:

- Relation (i) ⇔ (v) most important for us
- Since $f = f^{**}$ the result holds with f and f^* interchanged
- Full proof available on course webpage

Composite Optimization

Composite optimization

We consider composite optimization problems of the form

$$\underset{x}{\text{minimize}}\, f(Lx) + g(x)$$

Optimality conditions and dual problem

- Assume f, g closed convex and that CQ holds
- Problem $\text{minimize}_x(f(Lx) + g(x))$ is solved by x iff

$$0 \in L^T \underbrace{\partial f(Lx)}_{\mu} + \partial g(x)$$

where dual variable μ has been defined
- Primal dual necessary and sufficient optimality conditions:

$$\begin{cases} \mu \in \partial f(Lx) \\ -L^T\mu \in \partial g(x) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ -L^*\mu \in \partial g(x) \end{cases}$$

$$\begin{cases} \mu \in \partial f(Lx) \\ x \in \partial g^*(-L^T\mu) \end{cases} \qquad \begin{cases} Lx \in \partial f^*(\mu) \\ x \in \partial g^*(-L^T\mu) \end{cases}$$

- Dual optimality condition

$$0 \in \partial f^*(\mu) + \partial(g^* \circ -L^T)(\mu) \qquad (1)$$

solves dual problem $\text{minimize}_\mu\, f^*(\mu) + g^*(-L^T\mu)$
- If CQ-D holds, all dual problem solutions satisfy (1)
- Dual searches for μ such that $L^T\mu \in \partial f(x)$ and $-L^T\mu \in \partial g(x)$

Solving the primal via the dual

- Why solve dual? Sometimes easier to solve than primal
- Only interesting if primal solution can be recovered
- Assume f, g closed convex and CQ
- Assume optimal dual μ known: $0 \in \partial f^*(\mu) + \partial(g^* \circ -L^T)(\mu)$
- Optimal primal x must satisfy any and all primal-dual conditions:

$$\begin{cases}\mu \in \partial f(Lx) \\ -L^T\mu \in \partial g(x)\end{cases} \qquad \begin{cases}Lx \in \partial f^*(\mu) \\ -L^T\mu \in \partial g(x)\end{cases}$$

$$\begin{cases}\mu \in \partial f(Lx) \\ x \in \partial g^*(-L^T\mu)\end{cases} \qquad \begin{cases}Lx \in \partial f^*(\mu) \\ x \in \partial g^*(-L^T\mu)\end{cases}$$

- If one of these uniquely characterizes x, then must be solution:
  - ∂g^* is differentiable at $-L^T\mu$ for dual solution μ
  - ∂f^* is differentiable at dual solution μ and L invertible
  - ...

Algorithms

Proximal gradient method

- Consider $\underset{x}{\text{minimize}}\, f(x) + g(x)$ where
  - f is β-smooth $f : \mathbb{R}^n \to \mathbb{R}$ (not necessarily convex)
  - g is closed convex
- Due to β-smoothness of f, we have

$$f(y) + g(y) \leq f(x) + \nabla f(x)^T(y - x) + \tfrac{\beta}{2}\|y - x\|_2^2 + g(y)$$

for all $x, y \in \mathbb{R}^n$, i.e., r.h.s. is majorizing function for fixed x

- Majorization minimization with majorizer if $\gamma_k \in [\epsilon, \beta^{-1}]$, $\epsilon > 0$:

$$\begin{aligned} x_{k+1} &= \underset{y}{\operatorname{argmin}} \Big(f(x_k) + \nabla f(x_k)^T(y - x) + \tfrac{1}{2\gamma_k}\|y - x_k\|_2^2 + g(y)\Big) \\ &= \underset{y}{\operatorname{argmin}} \Big(g(y) + \tfrac{1}{2\gamma_k}\|y - (x_k - \gamma_k \nabla f(x_k))\|_2^2\Big) \\ &= \operatorname{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k)) \end{aligned}$$

gives proximal gradient method

Proximal gradient – Fixed-points

- Denote $T_{\mathrm{PG}}^{\gamma} := \operatorname{prox}_{\gamma g}(I - \gamma\nabla f)$, gives algorithm $x_{k+1} = T_{\mathrm{PG}}^{\gamma} x_k$
- Proximal gradient fixed-point set definition

$$\operatorname{fix} T_{\mathrm{PG}}^{\gamma} = \{x : x = T_{\mathrm{PG}}^{\gamma} x\} = \{x : x = \operatorname{prox}_{\gamma g}(x - \gamma \nabla f(x))\}$$

i.e., set of points for which $x_{k+1} = x_k$

Let $\gamma > 0$. Then $\bar{x} \in \operatorname{fix} T_{\mathrm{PG}}^{\gamma}$ if and only if $0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$.

- Consequence: fixed-point set same for all $\gamma > 0$
- We call inclusion $0 \in \partial g(\bar{x}) + \nabla f(\bar{x})$ *fixed-point characterization*
  - For convex problems: global solutions
  - For nonconvex problems: critical points

Applying proximal gradient to primal problems

Problem $\underset{x}{\text{minimize}}\, f(x) + g(x)$:

- Assumptions:
  - f β-smooth
  - g closed convex and prox friendly[1]
  - $\gamma_k \in [\epsilon, \frac{2}{\beta} - \epsilon]$
- Algorithm: $x_{k+1} = \operatorname{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$

Problem $\underset{x}{\text{minimize}}\, f(Lx) + g(x)$:

- Assumptions:
  - f β-smooth (implies $f \circ L$ $\beta\|L\|_2^2$-smooth)
  - g closed convex and prox friendly[1]
  - $\gamma_k \in [\epsilon, \frac{2}{\beta\|L\|_2^2} - \epsilon]$
- Gradient $\nabla(f \circ L)(x) = L^T \nabla f(Lx)$
- Algorithm: $x_{k+1} = \operatorname{prox}_{\gamma_k g}(x_k - \gamma_k L^T \nabla f(Lx_k))$

---

[1] Prox friendly: proximal operator cheap to evaluate, e.g., g separable

Applying proximal gradient to dual problem

Dual problem $\underset{\nu}{\text{minimize}}\, f^*(\nu) + g^*(-L^T\nu)$:

- Assumptions:
  - f closed convex and prox friendly
  - g σ-strongly convex (which implies $g^* \circ -L^T$ $\frac{\|L\|_2^2}{\sigma}$-smooth)
  - $\gamma_k \in [\epsilon, \frac{2\sigma}{\|L\|_2^2} - \epsilon]$
- Gradient: $\nabla(g^* \circ -L^T)(\nu) = -L\nabla g^*(-L^T\nu)$
- Prox (Moreau): $\operatorname{prox}_{\gamma_k f^*}(\nu) = \nu - \gamma_k \operatorname{prox}_{\gamma_k^{-1} f}(\gamma_k^{-1}\nu)$
- Algorithm:

$$\begin{aligned} \nu_{k+1} &= \operatorname{prox}_{\gamma_k f^*}(\nu_k - \gamma_k \nabla(g^* \circ -L^T)(\nu_k)) \\ &= (I - \gamma_k \operatorname{prox}_{\gamma_k^{-1} f}(\gamma_k^{-1} \circ I))(\nu_k + \gamma_k L \nabla g^*(-L^T\nu_k)) \end{aligned}$$

- Problem must be convex to have dual!
- Enough to know prox of f

What problems cannot be solved (efficiently)?

Problem $\underset{x}{\text{minimize}}\, f(x) + g(x)$

- Assumptions: f and g convex and nonsmooth
- No term differentiable, another method must be used:
  - Subgradient method
  - Douglas-Rachford splitting
  - Primal-dual methods

Problem $\underset{x}{\text{minimize}}\, f(x) + g(Lx)$

- Assumptions:
  - f smooth
  - g nonsmooth convex
  - L arbitrary structured matrix
- Can apply proximal gradient method, but

$$\operatorname{prox}_{\gamma_k(g \circ L)}(z) = \underset{x}{\operatorname{argmin}}\, g(Lx) + \tfrac{1}{2\gamma}\|x - z\|_2^2)$$

often not "prox friendly", i.e., it is expensive to evaluate

Training problems

- Training problem format

$$\underset{\theta}{\text{minimize}} \underbrace{\sum_{i=1}^{N} L(m(x_i;\theta), y_i)}_{f(X\theta)} + \underbrace{\sum_{j=1}^{n} g_j(\theta_j)}_{g(\theta)}$$

where f is data misfit term and g is regularizer

- Regularizers ($\theta = (w, b)$)
  - Tikhonov $g(\theta) = \|w\|_2^2$ is prox-friendly
  - Sparsity inducing 1-norm $g(\theta) = \|w\|_1$ is prox-friendly
- Data misfit terms (with $m(x;\theta) = \phi(x)^T\theta$ for convex problems)
  - Least squares $L(u, y) = \|u - y\|_2^2$ smooth, hence f smooth
  - Logistic $L(u, y) = \log(1 + e^u) - yu$ smooth, hence f smooth
  - SVM $L(u, y) = \max(0, 1 - yu)$ not smooth, hence f not smooth
- Proximal gradient method
  - Least squares: can efficiently solve primal
  - Logistic regression: can solve primal
  - SVM: add strongly convex regularization and solve dual
    - Strongly convex regulariztion to have one conjugate smooth
    - If bias term not regularized, only strongly convex in w
    - SVM with $\|\cdot\|_1$-regularization not solvable with prox-grad

Dual training problem

- Convex training problem

$$\underset{\theta}{\text{minimize}} \underbrace{\sum_{i=1}^{N} L(\phi(x_i)^T\theta, y_i)}_{f(X\theta)} + \underbrace{\sum_{j=1}^{n} g_j(\theta_j)}_{g(\theta)}$$

has dual

$$\underset{\theta}{\text{minimize}} \underbrace{\sum_{i=1}^{N} L^*(\mu_i)}_{f^*(\mu)} + \underbrace{\sum_{j=1}^{n} g_j^*((-X^T\mu)_j)}_{g^*(-X^T\mu)}$$

where the conjugate of L is w.r.t. first argument

- Dual has same structure as primal, finite-sum plus separable

Training problem structure

- Primal training problem

$$\underset{\theta}{\text{minimize}} \underbrace{\sum_{i=1}^{N} L(m(x_i;\theta), y_i)}_{f(X\theta)} + \underbrace{\sum_{j=1}^{n} g_j(\theta_j)}_{g(\theta)}$$

- Dual training problem

$$\underset{\theta}{\text{minimize}} \underbrace{\sum_{i=1}^{N} L^*(\mu_i)}_{f^*(\mu)} + \underbrace{\sum_{j=1}^{n} g_j^*((-X^T\mu)_j)}_{g^*(-X^T\mu)}$$

- Common structure, finite sum plus separable:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} f_i((X\theta)_i) + \sum_{j=1}^{n} \psi_j(\theta_j)$$

  - Primal: $f_i = L(m(x_i;\cdot), y_i)$ (one summand per training example)
  - Dual: $f_i = g_j^*((-X^T\cdot)_j)$, $\psi_j = L^*$

Exploiting structure

- Common structure, finite sum plus separable:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} f_i((X\theta)_i) + \sum_{j=1}^{n} \psi_j(\theta_j)$$

- Stochastic gradient descent exploits finite-sum structure:
  - Computes stochastic gradient of *smooth* part f
  - Pick summand f_i at random and perform gradient step
  - Primal formulations: Pick training example and compute gradient
  - Deep learning: evaluted via backpropagation
- Coordinate gradient descent exploits separable structure:
  - Coordinate-wise updates if *nonsmooth* ϕ_j separable
  - Requires efficient coordinate-wise evaluations of ∇f