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

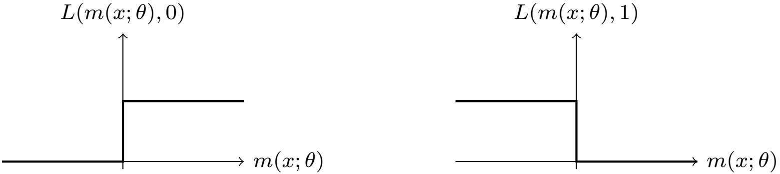
$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} L(m(x_i;\theta), y_i)$$

- 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

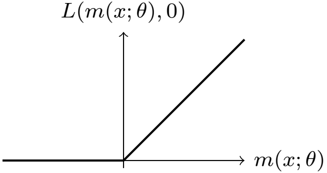
- Different cost functions L can be used:
  - $y = 0$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$

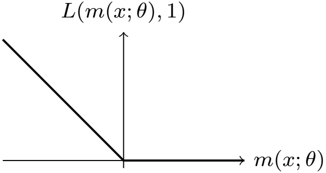


nonconvex (Neyman Pearson loss)

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 $m(x;\theta) \ll 0$

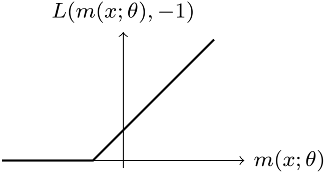


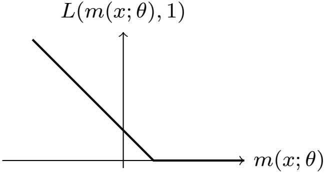


$$L(u, y) = \max(0, u) - yu$$

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 $m(x;\theta) \ll 0$

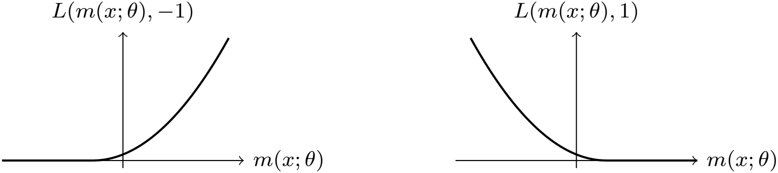




$L(u, y) = \max(0, 1 - yu)$ (hinge loss used in SVM)

Binary classification – Cost functions

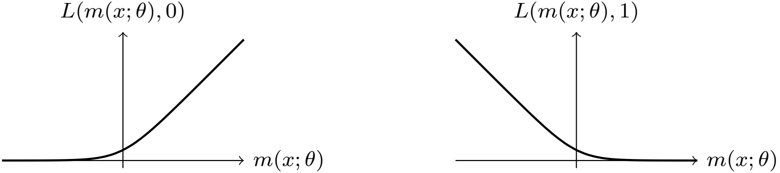
- Different cost functions L can be used:
  - $y = -1$: Small cost for $m(x;\theta) \ll 0$ large for $m(x;\theta) \gg 0$
  - $y = 1$: Small cost for $m(x;\theta) \gg 0$ large for $m(x;\theta) \ll 0$



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

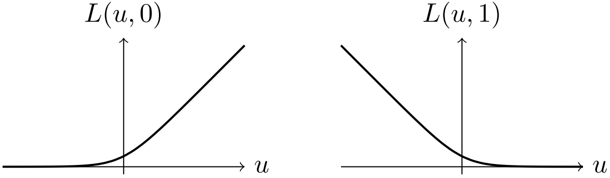
- Classification
- **Logistic regression**
- Nonlinear features
- Overfitting and regularization
- Multiclass logistic regression
- Training problem properties

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:

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

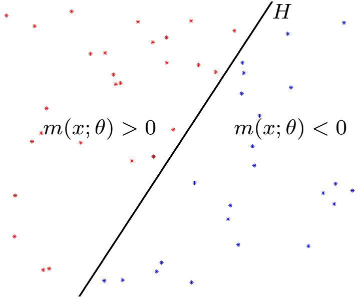
- Training problem convex in $\theta = (w, b)$ since:
  - model $m(x;\theta)$ is affine in θ
  - loss function $L(u, y)$ is convex in u



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:

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

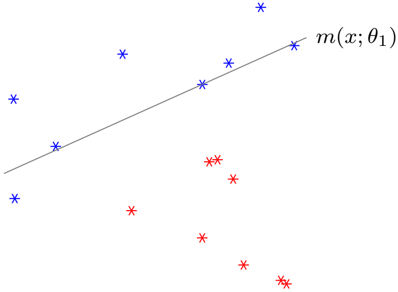


Training problem interpretation

- 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:

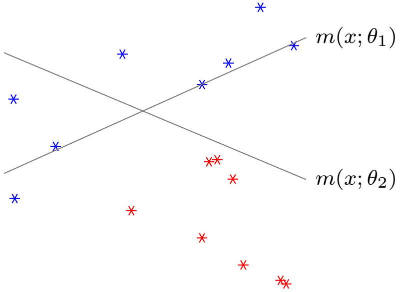


Training problem interpretation

- 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:

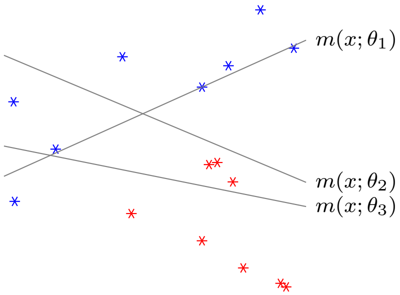


Training problem interpretation

- 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:

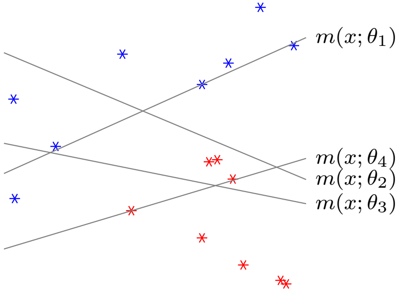


Training problem interpretation

- 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:

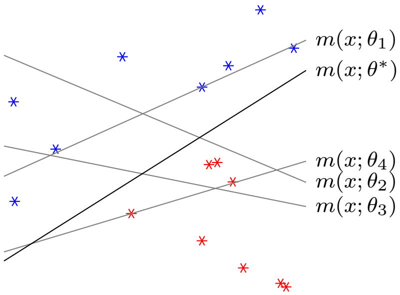


Training problem interpretation

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

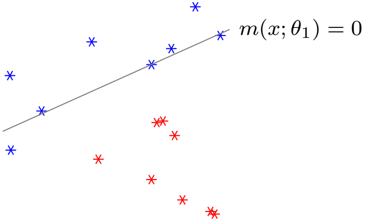
$$H := \{x : w^T x + b = 0\} = \{x : m(x; \theta) = 0\}$$

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

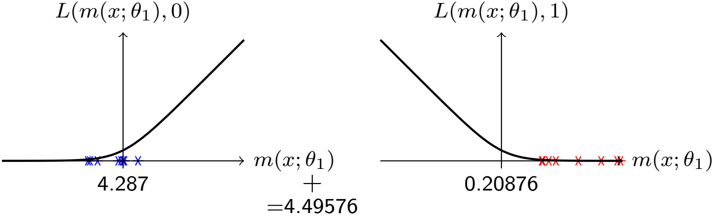


What is "best" separation?

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

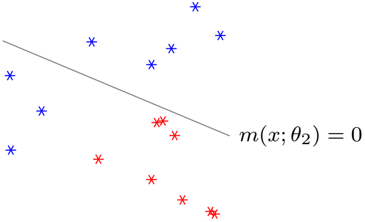


- Training loss:

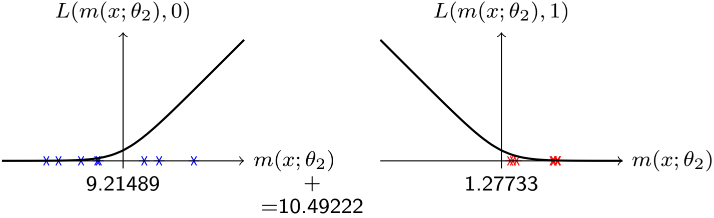


What is “best” separation?

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

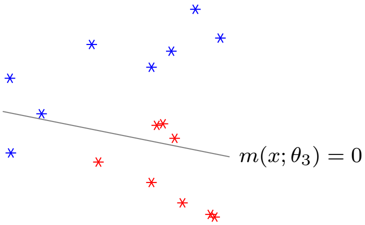


- Training loss:

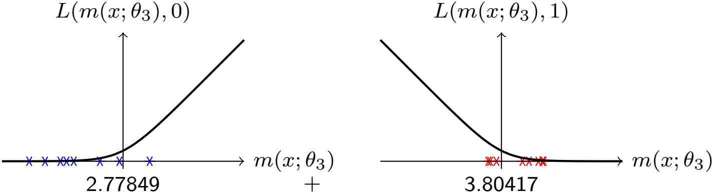


What is "best" separation?

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



- Training loss:

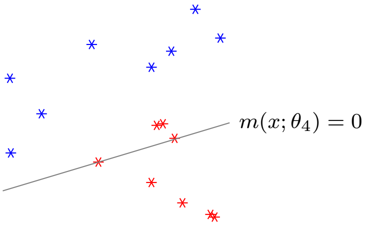


$+$

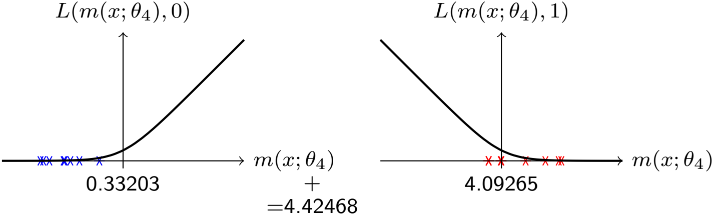
$=6.58266$

What is "best" separation?

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

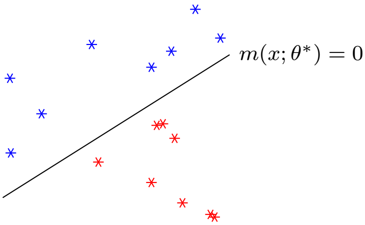


- Training loss:

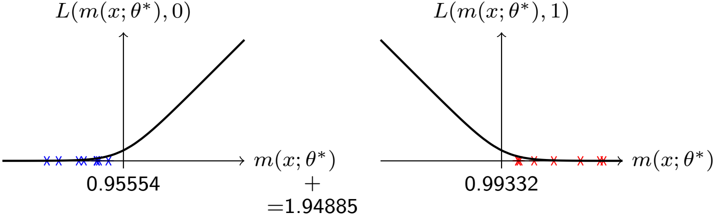


What is "best" separation?

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

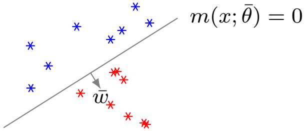


- Training loss:

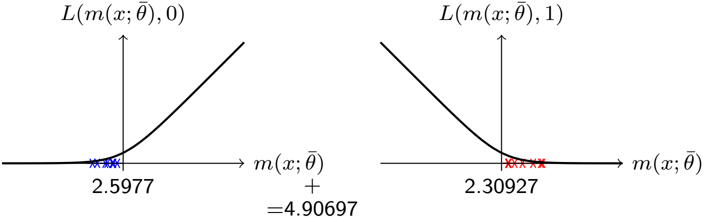


Fully separable data – Solution

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

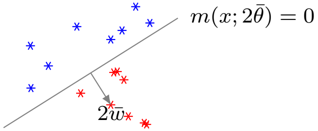


- Let $H_{\bar{\theta}} := \{x : m(x;\bar{\theta}) = \bar{w}^T x + \bar{b} = 0\}$ be hyperplane separates
- Training loss:

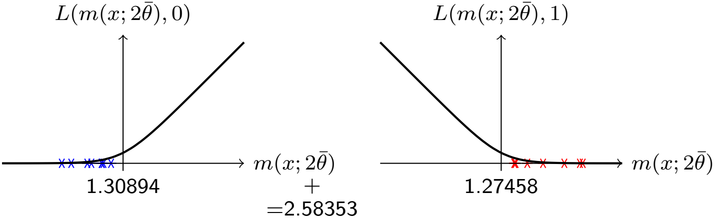


Fully separable data – Solution

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

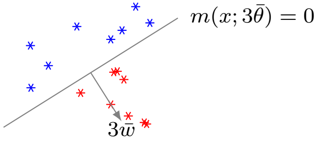


- Hyperplane $H_{2\bar{\theta}} := \{x : m(x; 2\bar{\theta}) = 2(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss reduced since input $m(x; 2\bar{\theta}) = 2m(x; \bar{\theta})$ further out:

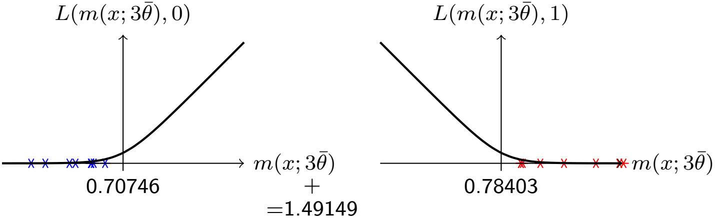


Fully separable data – Solution

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

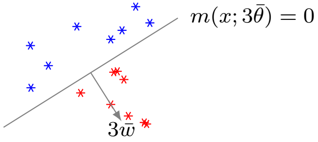


- Hyperplane $H_{3\bar{\theta}} := \{x : m(x; 3\bar{\theta}) = 3(\bar{w}^T x + \bar{b}) = 0\} = H_{\bar{\theta}}$ same
- Training loss further reduced since input $m(x; 3\bar{\theta}) = 3m(x; \bar{\theta})$:

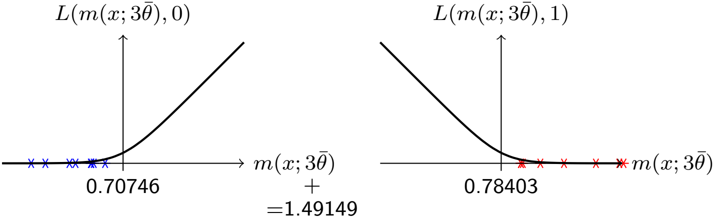


Fully separable data – Solution

- 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



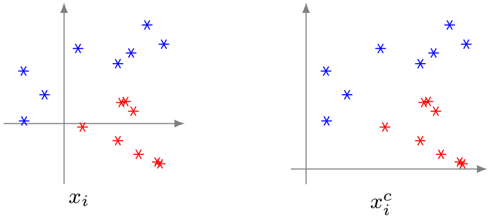
- Let $\theta = t\bar{\theta}$ and $t \to \infty$, then loss $\to 0 \Rightarrow$ no optimal point

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:

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

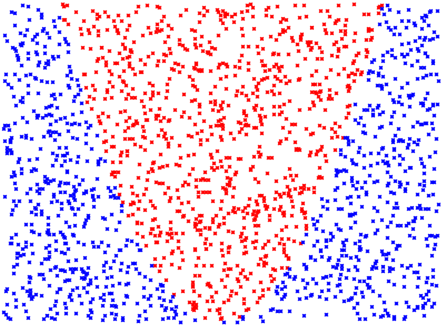
- Two equivalent formulations to arrive at same problem:
  - Real-valued model $m(x;\theta)$ and logistic loss $\log(1+e^{u}) - yu$
  - $(0,1)$-valued model $\sigma(m(x;\theta))$ and binary cross entropy
- Prefer previous formulation
  - easier to see how deviations penalized
  - easier to conclude convexity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

Outline

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

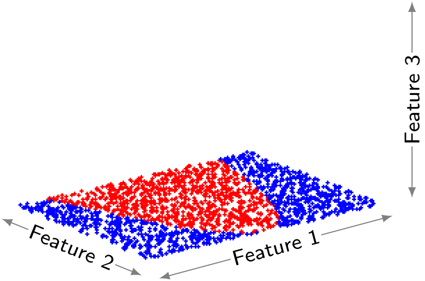
Logistic regression – Nonlinear example

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



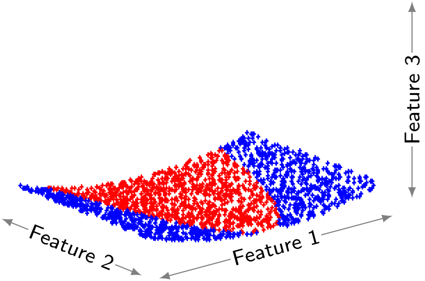
Logistic regression – Example

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



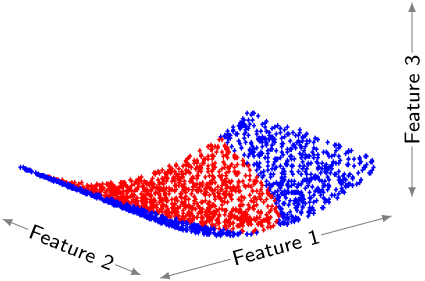
Logistic regression – Example

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



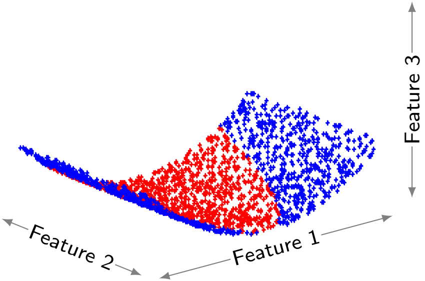
Logistic regression – Example

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



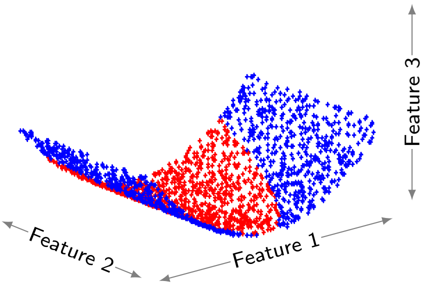
Logistic regression – Example

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



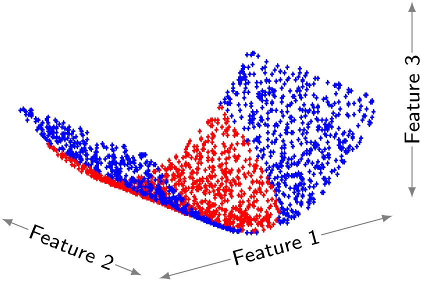
Logistic regression – Example

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



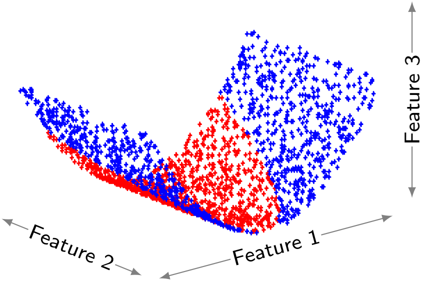
Logistic regression – Example

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



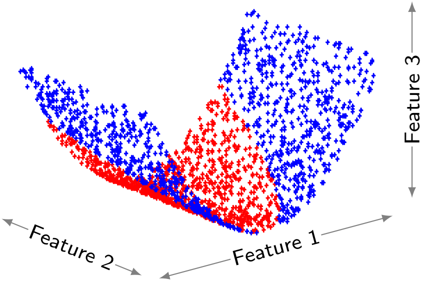
Logistic regression – Example

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



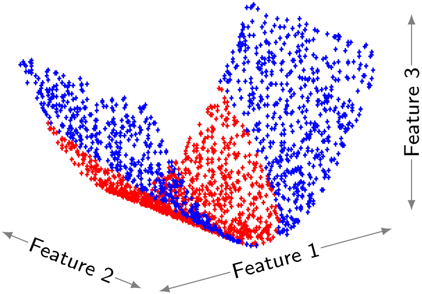
Logistic regression – Example

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



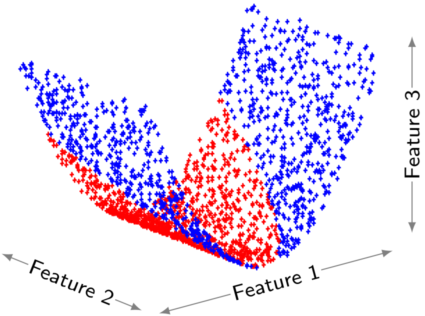
Logistic regression – Example

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



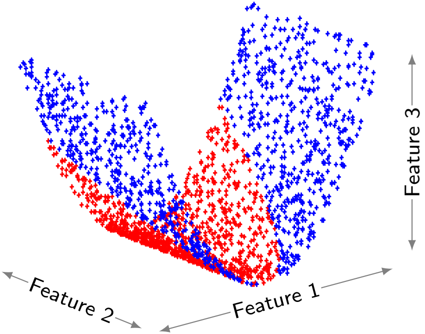
Logistic regression – Example

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



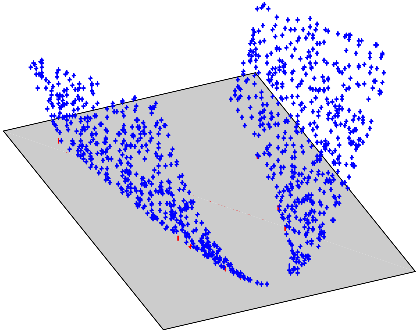
Logistic regression – Example

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



Logistic regression – Example

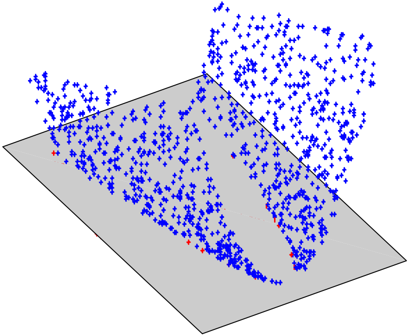
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

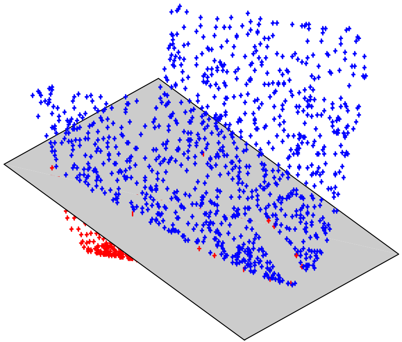
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

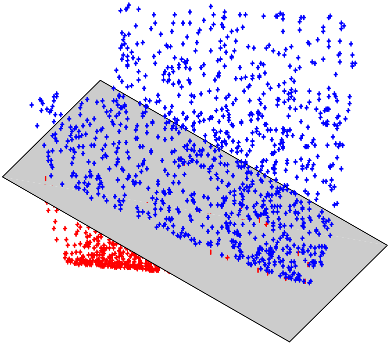
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

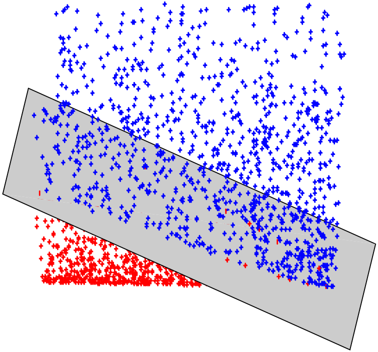
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

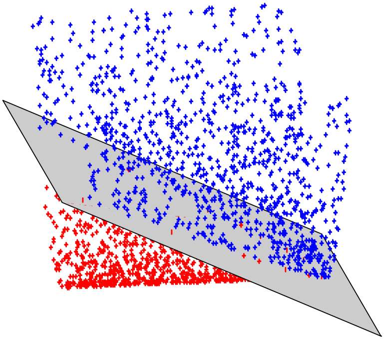
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

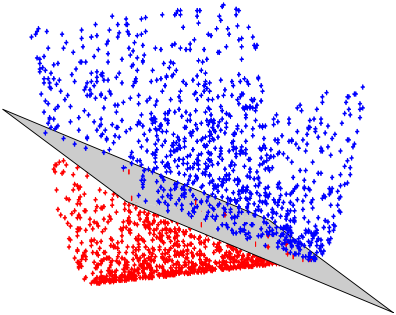
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

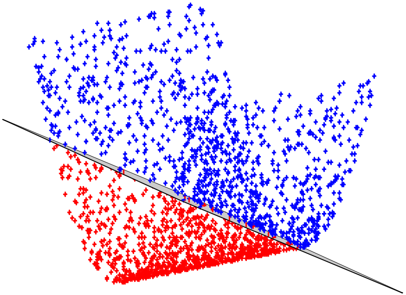
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

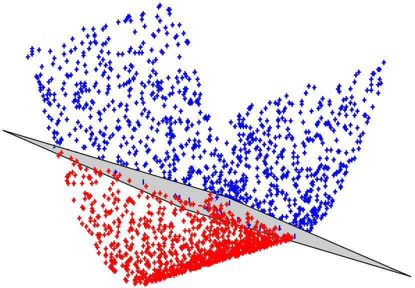
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

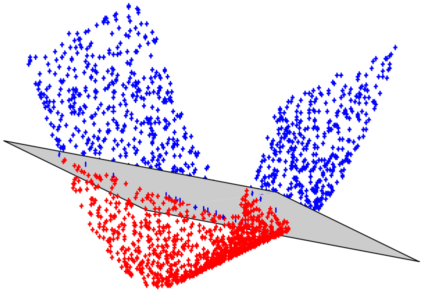
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- Data linearly separable in lifted (feature) space

Logistic regression – Example

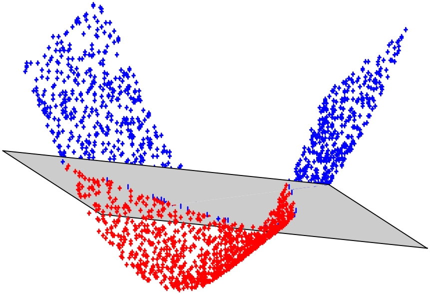
- Seems linear in feature 2 and quadratic in feature 1
- Add a third feature which is feature 1 squared



- 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

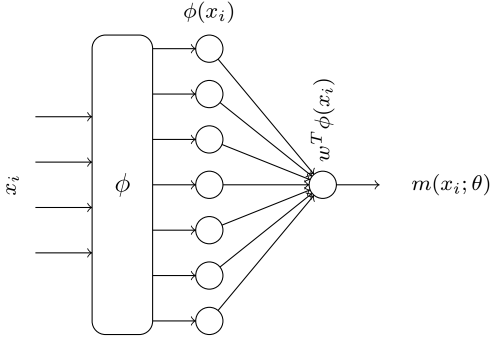
- Create feature map $\phi : \mathbb{R}^n \to \mathbb{R}^p$ of training data
- Data points $x_i \in \mathbb{R}^n$ replaced by featured data points $\phi(x_i) \in \mathbb{R}^p$
- New model: $m(x;\theta) = w^T \phi(x) + b$, still linear in parameters
- Feature can include original data x
- We can add feature 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

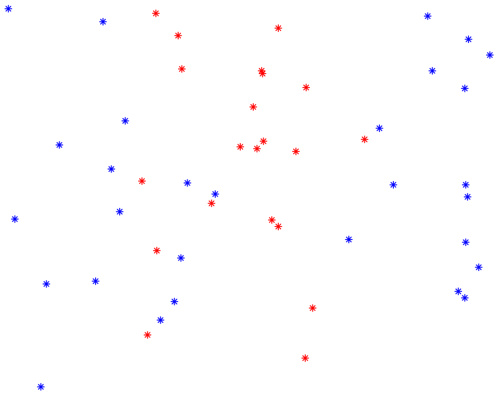
- Polynomial feature map for $\mathbb{R}^n$ with $n = 2$ and degree $d = 3$

$$\phi(x) = (x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3)$$

  (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

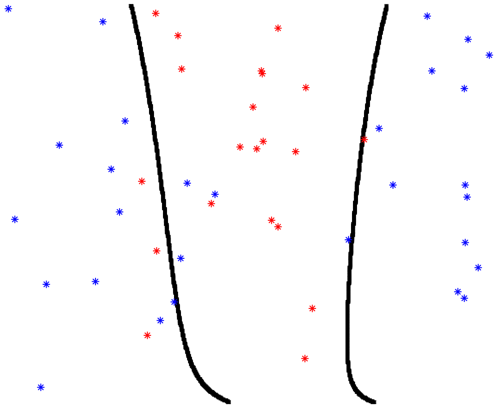
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree:



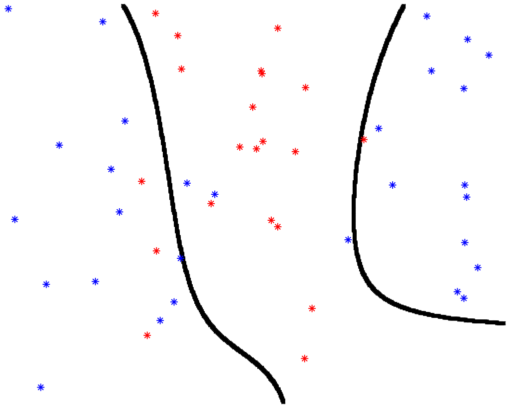
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 2



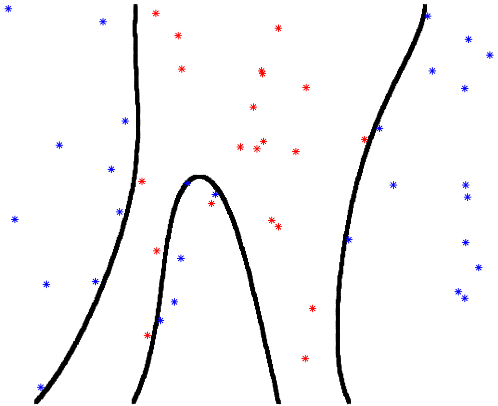
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 3



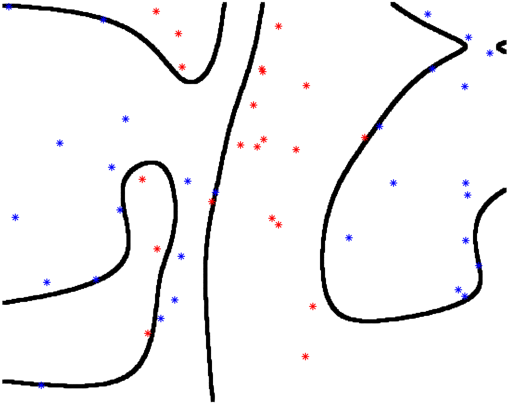
Example – Different polynomial model orders

- "Lifting" example with fewer samples and some mislabels
- Logistic regression (no regularization) polynomial features of degree: 4



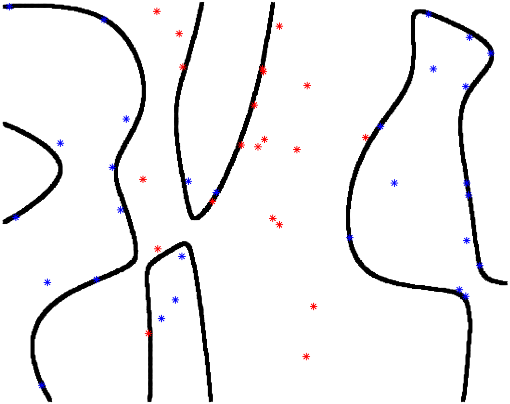
Example – Different polynomial model orders

- "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

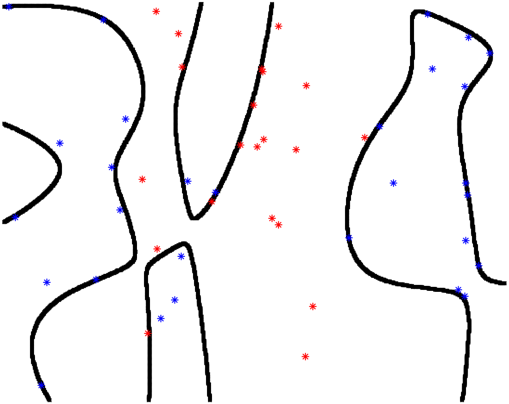


Outline

- 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:

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} \left(\log(1 + e^{x_i^T w + b}) - y_i(x_i^T w + b) \right) + \lambda \|w\|_2^2$$

Regularization:

- Regularize only w and not the bias term b
- Why? Model looses shift invariance if also b regularized

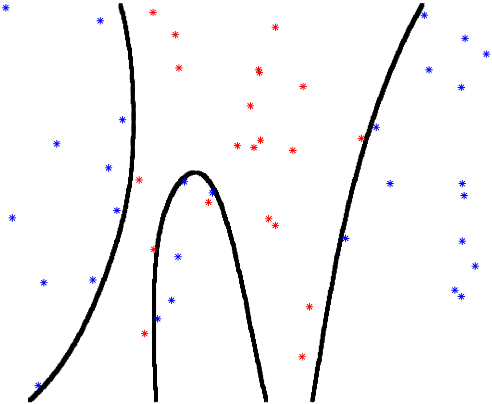
Problem properties:

- 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

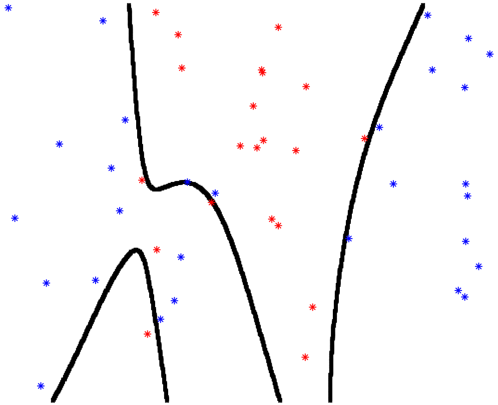
λ	J	# mislabels
0.00001	4.60	1



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

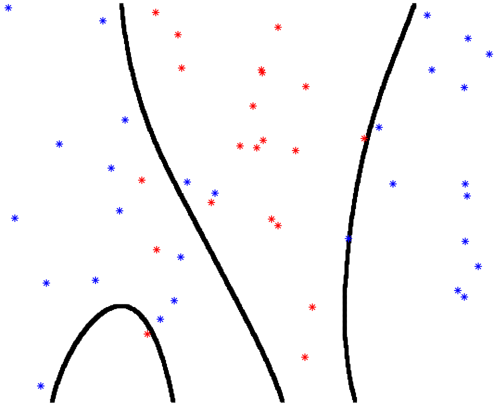
λ	J	# mislabels
0.00006	6.83	5



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

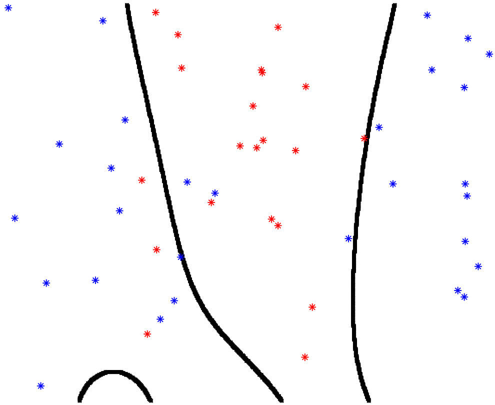
λ	J	# mislabels
0.00036	9.94	5



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

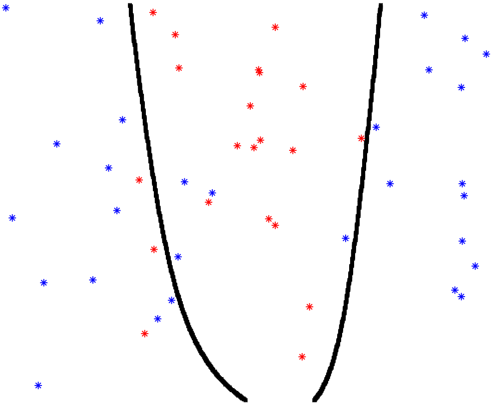
λ	J	# mislabels
0.0021	12.1	6



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

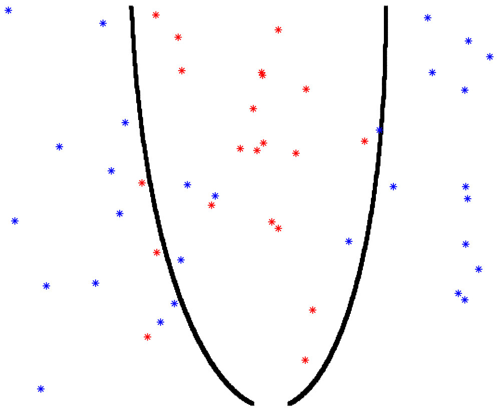
λ	J	# mislabels
0.013	13.6	7



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

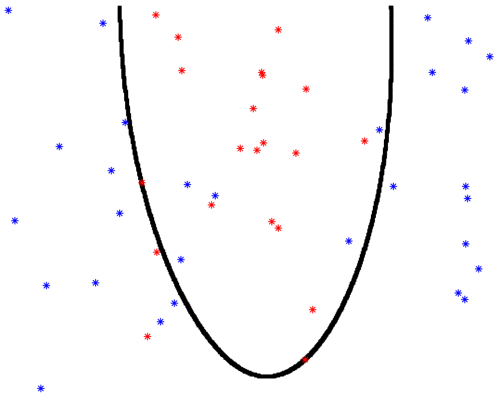
λ	J	# mislabels
0.077	15.4	8



Example – Different regularization

- Regularized logistic regression and polynomial features of degree 6
- Regularization parameter λ, training cost J, # mislabels in training

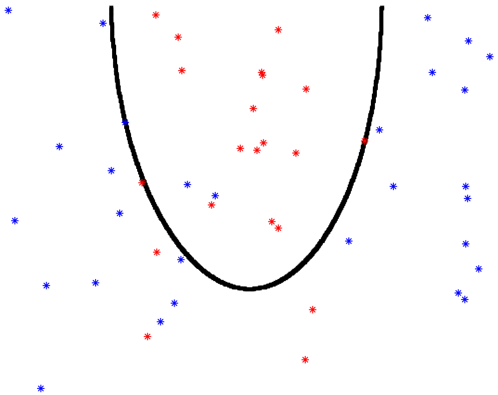
λ	J	# mislabels
0.46	19.2	7



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



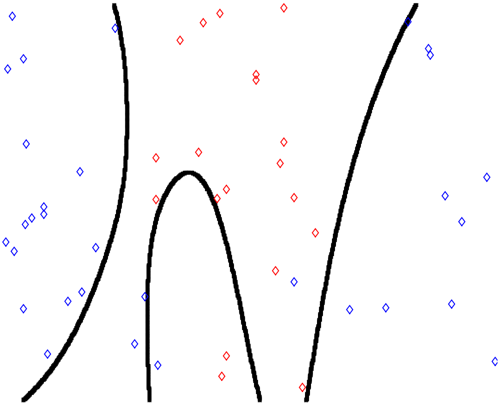
Generalization

- 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

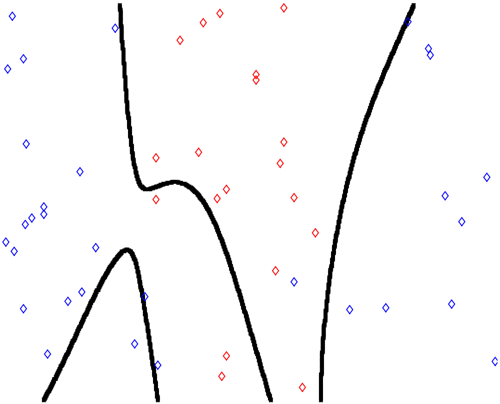
λ	J	# mislabels
0.00001	4.60/38.5	1/7



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

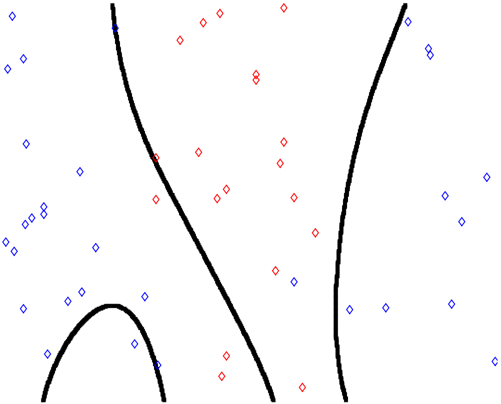
λ	J	# mislabels
0.00006	6.83/25.7	5/7



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

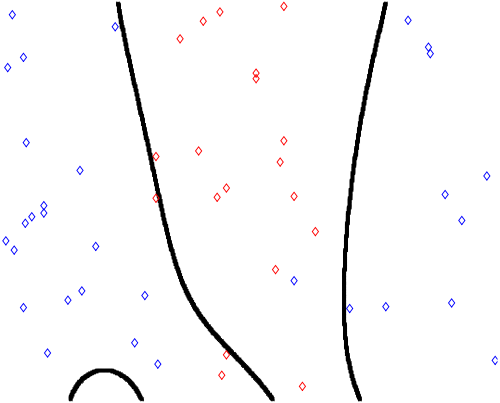
λ	J	# mislabels
0.00036	9.94/13.4	5/8



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

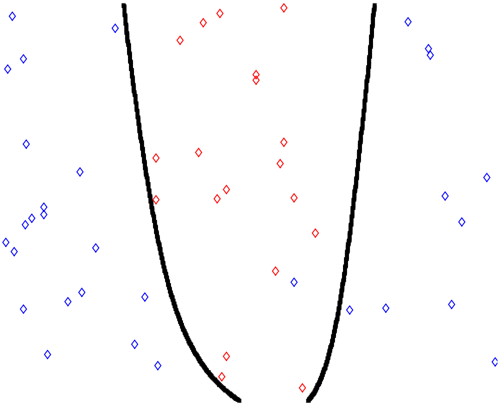
λ	J	# mislabels
0.0021	12.1/8.70	6/5



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and $\#$ mislabels specify training/test values

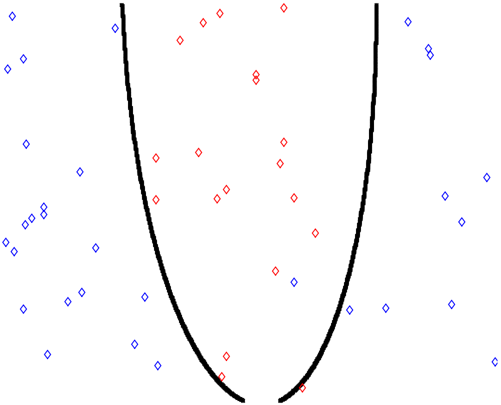
λ	J	$\#$ mislabels
0.013	13.6/8.12	7/2



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

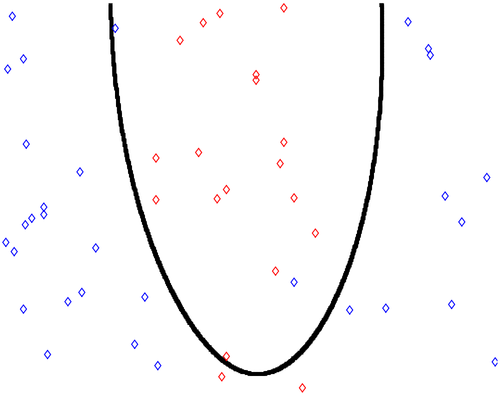
λ	J	# mislabels
0.077	15.4/10.2	8/3



Example – Validation data

- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

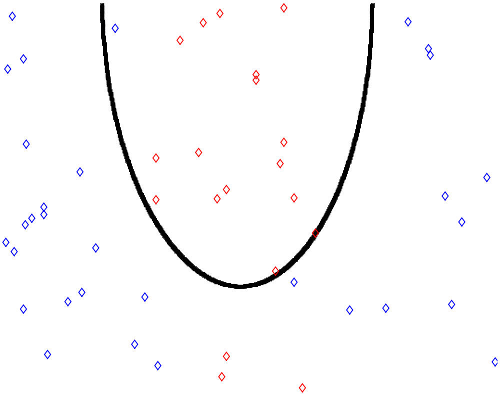
λ	J	# mislabels
0.46	19.2/15.2	7/4



Example – Validation data

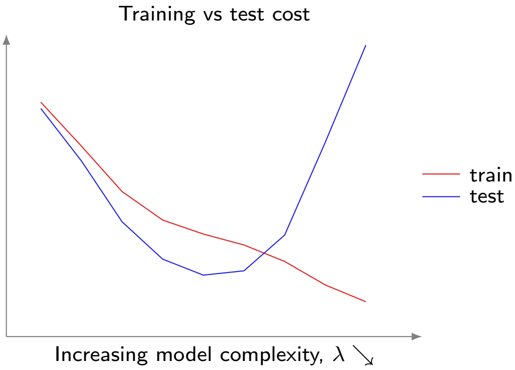
- Regularized logistic regression and polynomial features of degree 6
- J and # mislabels specify training/test values

λ	J	# mislabels
2.78	25.2/23.2	8/4



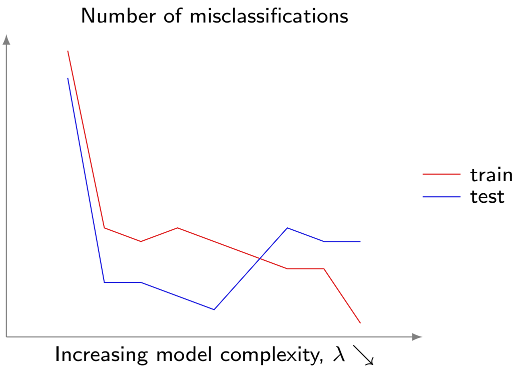
Test vs training error – Cost

- 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:

$$L(u, y) = \log\left(\sum_{j=1}^{K} e^{u_j}\right) - u^T y$$

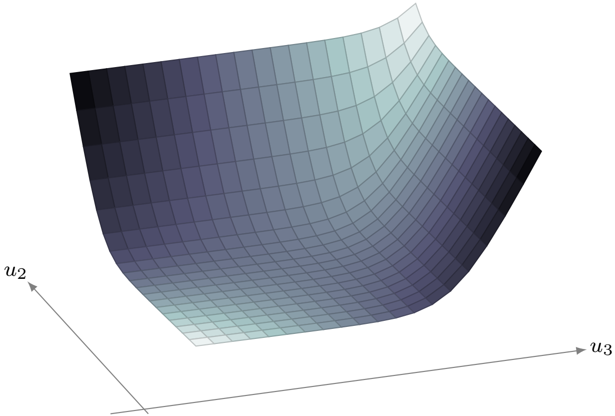
where label y is a "one-hot" basis vector, is convex in u

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 $y = e_1$

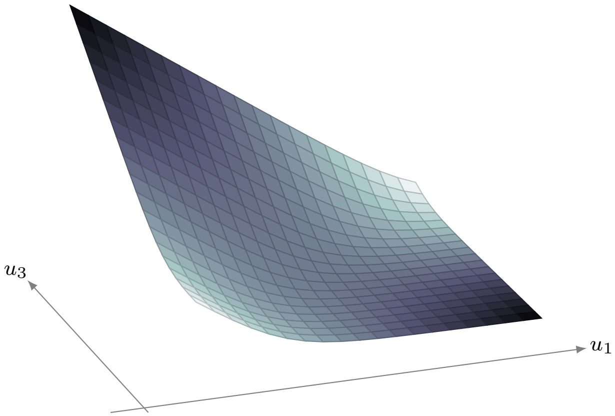


Multiclass logistic loss function – Example

- 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:

$$\underset{j \in \{1,\ldots,K\}}{\operatorname{argmax}} \ m_j(x;\theta)$$

i.e., class with largest model value (since trained to achieve this)

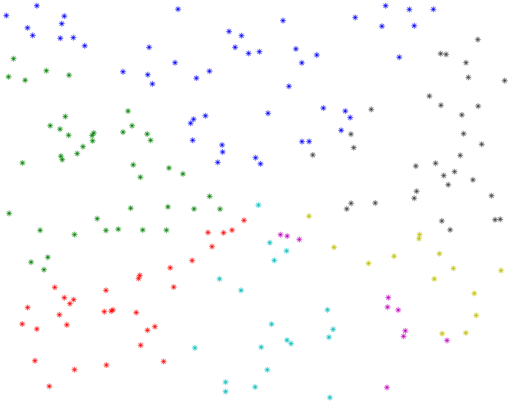
Special case – Binary logistic regression

- 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:

$$
\begin{aligned}
L(u,y) &= \log(e^{u_1} + e^{u_2}) - y_1 u_1 - y_2 u_2 \\
&= \log\left(1 + e^{u_1 - u_2}\right) + \log(e^{u_2}) - y_1 u_1 - y_2 u_2 \\
&= \log\left(1 + e^{\Delta u}\right) - y_1 u_1 - (y_2 - 1) u_2 \\
&= \log\left(1 + e^{\Delta u}\right) - y_{\text{bin}} \Delta u
\end{aligned}
$$

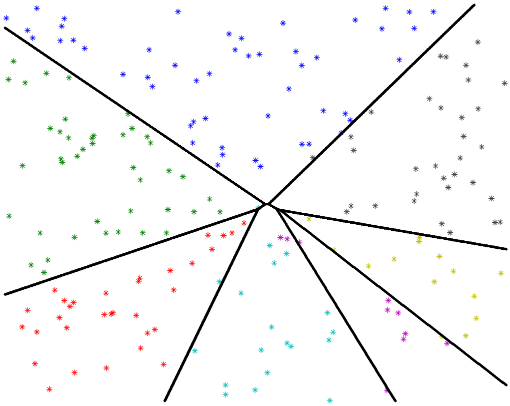
Example – Linearly separable data

- Problem with 7 classes



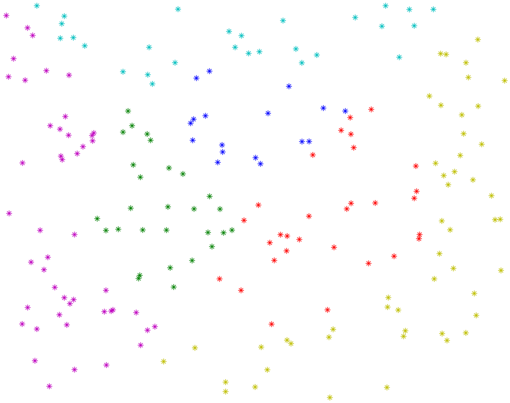
Example – Linearly separable data

- Problem with 7 classes and affine multiclass model



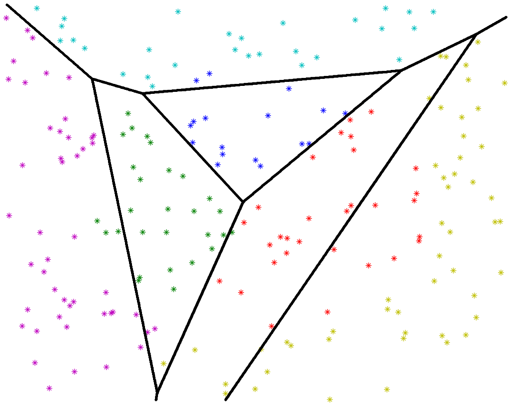
Example – Quadratically separable data

- Same data, new labels in 6 classes



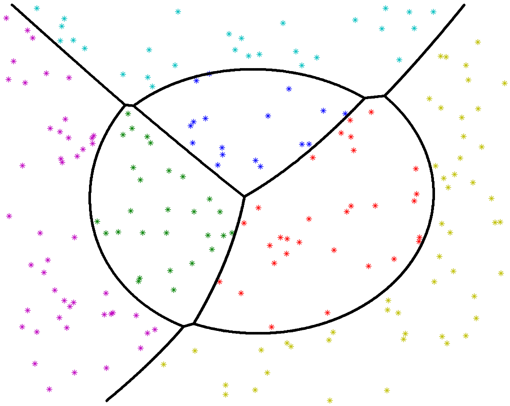
Example – Quadratically separable data

- Same data, new labels in 6 classes, affine model



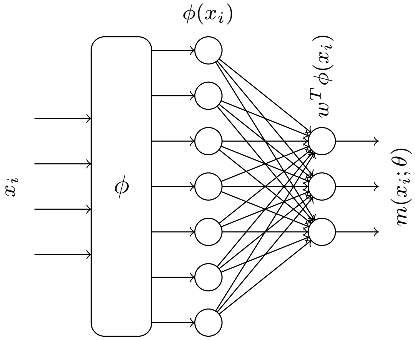
Example – Quadratically separable data

- 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)

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

can be written on the form

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

where

- $f(u) = \sum_{i=1}^{N} (\log(1 + e^{u_i}) - y_i u_i)$ 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 $h_i(u_i) = \log(1 + e^{u_i}) - y_i u_i$ 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:

$$\begin{aligned}\nabla(f \circ L)(\theta) &= \nabla \sum_{i=1}^{N} h_i(L_i\theta) = \sum_{i=1}^{N} L_i^T \nabla h_i(L_i\theta) \\ &= \sum_{i=1}^{N} \begin{bmatrix} x_i \\ 1 \end{bmatrix} (\sigma(x_i^T w + b) - y_i) \\ &= \begin{bmatrix} X^T \\ \mathbf{1}^T \end{bmatrix} (\sigma(Xw + b\mathbf{1}) - Y)\end{aligned}$$

  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