Stochastic Gradient Descent

Implicit Regularization

Pontus Giselsson

Outline

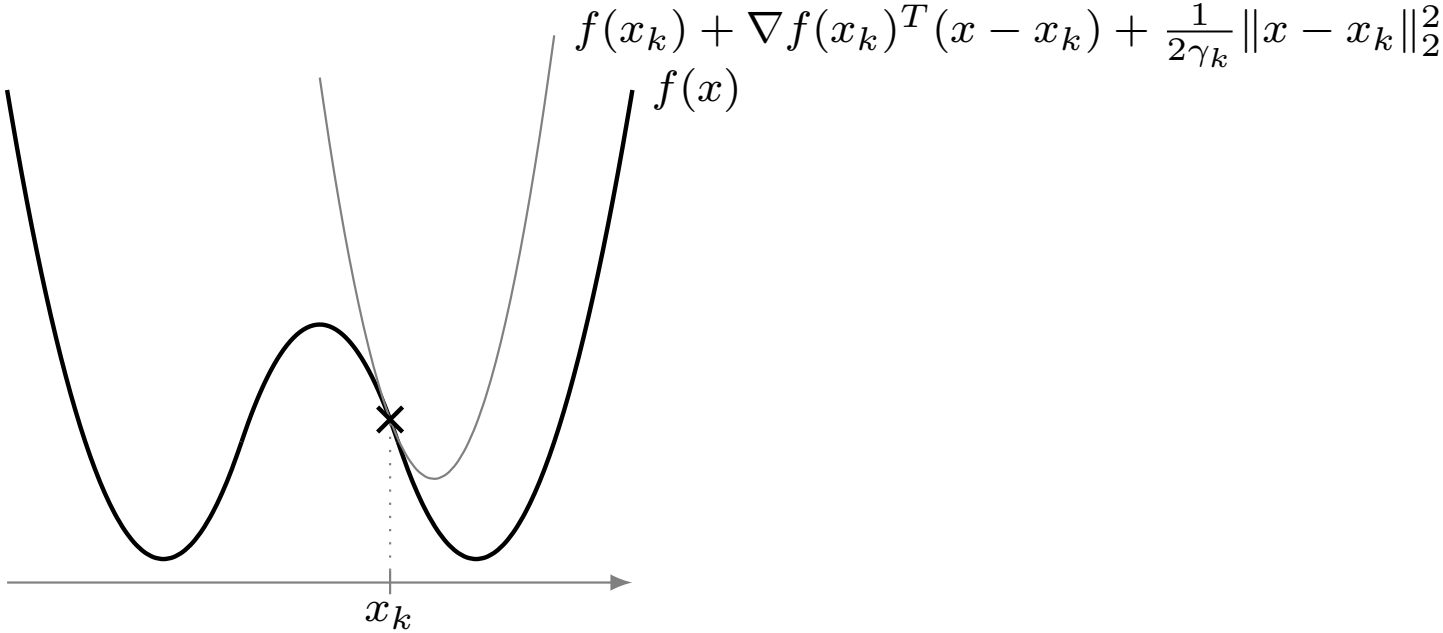
- **Variable metric methods**
- Convergence to projection point
- Convergence to sharp or flat minima

Gradient method interpretation

- Gradient method minimizes quadratic approximation of function

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \left(f(x_k) + \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2\gamma_k} \|x - x_k\|_2^2 \right) \\ &= \operatorname*{argmin}_x \left(\tfrac{1}{2\gamma_k} \|x - (x_k - \gamma_k \nabla f(x_k))\|_2^2 \right) \\ &= x_k - \gamma_k \nabla f(x_k) \end{aligned}$$

- Graphical illustration of one step

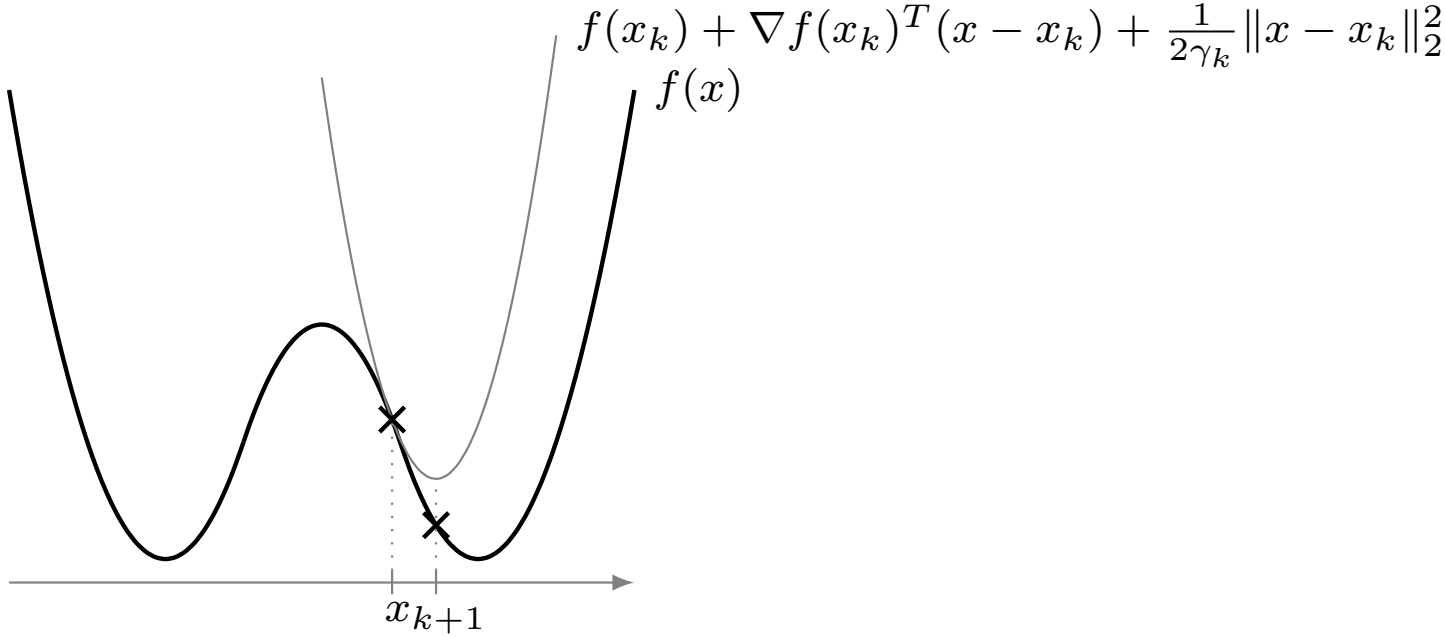


Gradient method interpretation

- Gradient method minimizes quadratic approximation of function

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \left(f(x_k) + \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2\gamma_k} \|x - x_k\|_2^2 \right) \\ &= \operatorname*{argmin}_x \left(\tfrac{1}{2\gamma_k} \|x - (x_k - \gamma_k \nabla f(x_k))\|_2^2 \right) \\ &= x_k - \gamma_k \nabla f(x_k) \end{aligned}$$

- Graphical illustration of one step



Scaled gradient method

- Quadratic approximation same in all directions due to $\|\cdot\|_2^2$

$$x_{k+1} = \operatorname*{argmin}_x \left(f(x_k) + \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2\gamma_k} \|x - x_k\|_2^2 \right)$$

- Scaled gradient method minimizes scaled quadratic approximation

$$\begin{aligned} x_{k+1} &= \operatorname*{argmin}_x \left(f(x_k) + \nabla f(x_k)^T (x - x_k) + \tfrac{1}{2\gamma_k} \|x - x_k\|_H^2 \right) \\ &= \operatorname*{argmin}_x \left(\tfrac{1}{2\gamma_k} \|x - (x_k - \gamma_k H^{-1} \nabla f(x_k))\|_H^2 \right) \\ &= x_k - \gamma_k H^{-1} \nabla f(x_k) \end{aligned}$$

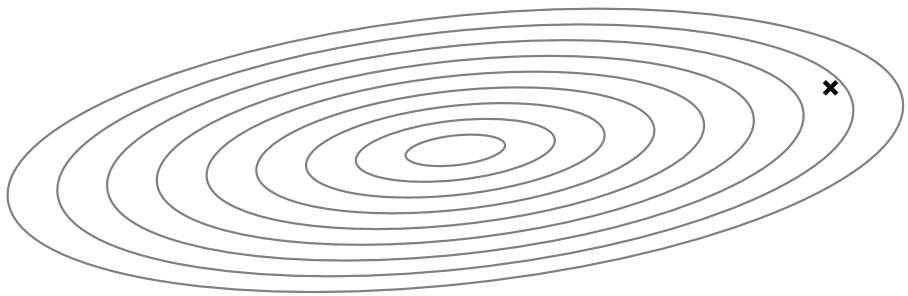
  where H is a positive definite matrix and $\|x\|_H^2 = x^T H x$
- Nominal gradient method obtained by $H = I$
- Better quadratic approximation (good H) $\Rightarrow$ faster convergence

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:

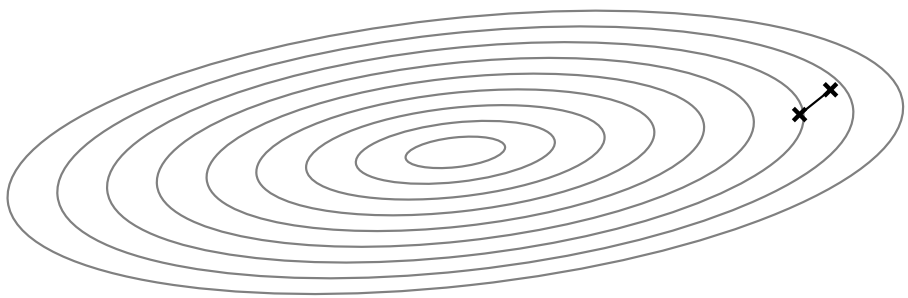


Gradient descent – Example

- (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:

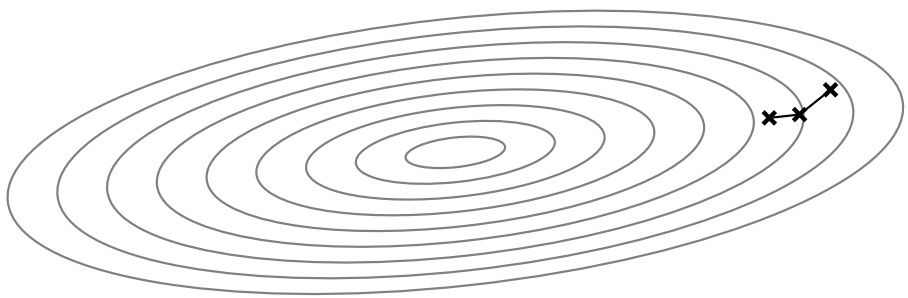


Gradient descent – Example

- (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:

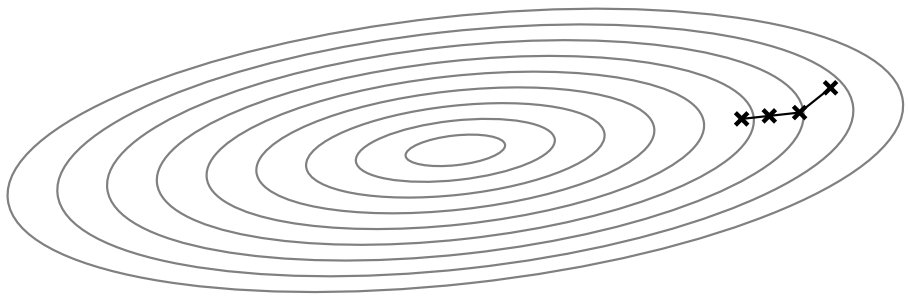


Gradient descent – Example

- (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:

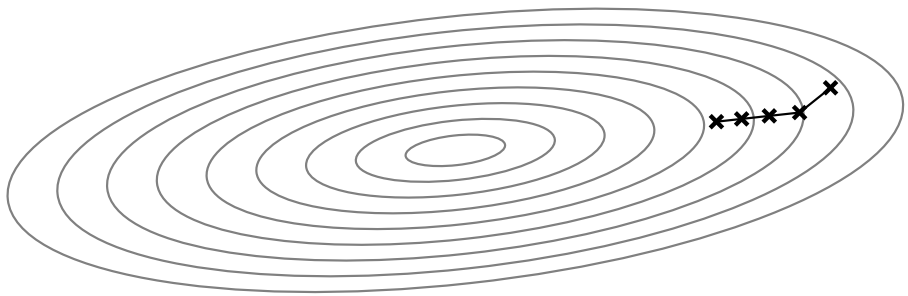


Gradient descent – Example

- (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:

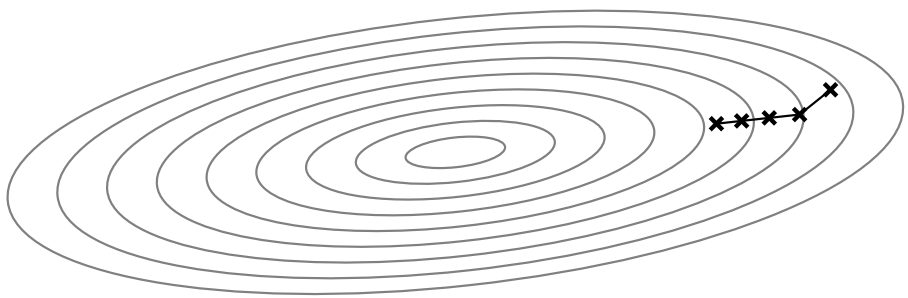


Gradient descent – Example

- (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:

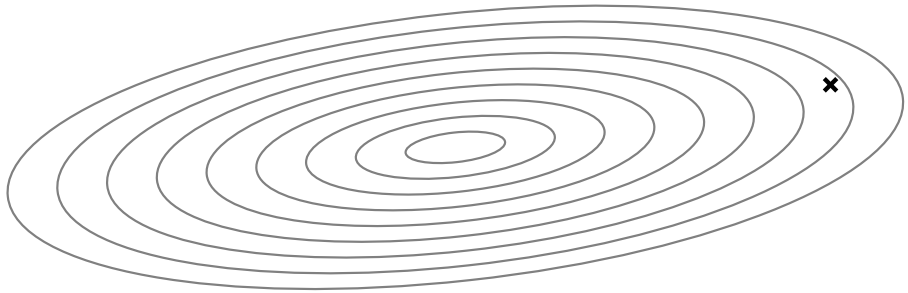


Scaled gradient descent – Example

- 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 $H = \mathbf{diag}(\nabla^2 f) := P$:

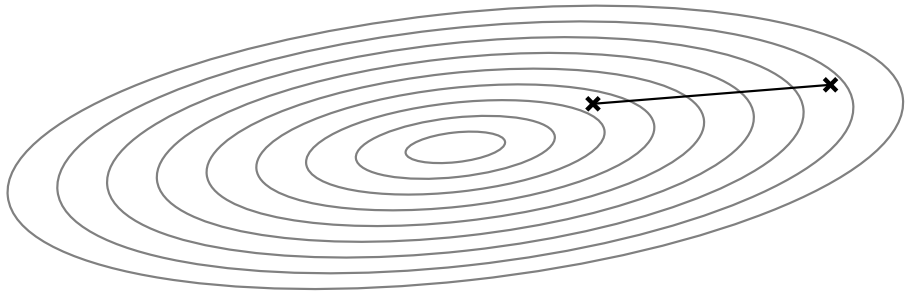


Scaled gradient descent – Example

- 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 $H = \mathbf{diag}(\nabla^2 f) := P$:

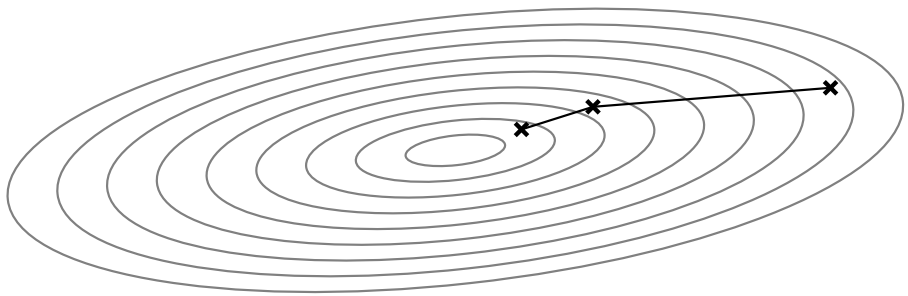


Scaled gradient descent – Example

- 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 $H = \mathbf{diag}(\nabla^2 f) := P$:

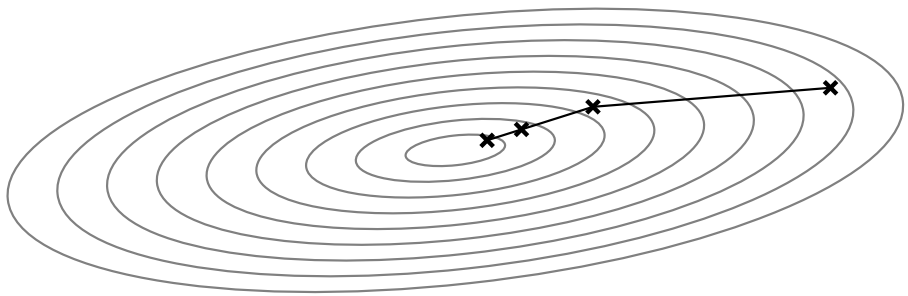


Scaled gradient descent – Example

- 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 $H = \mathbf{diag}(\nabla^2 f) := P$:

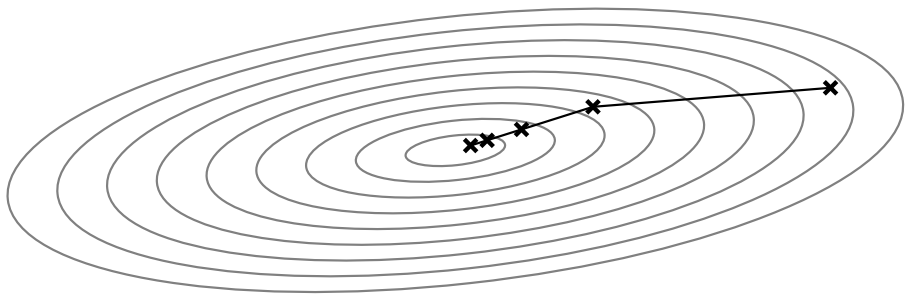


Scaled gradient descent – Example

- 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 $H = \mathbf{diag}(\nabla^2 f) := P$:

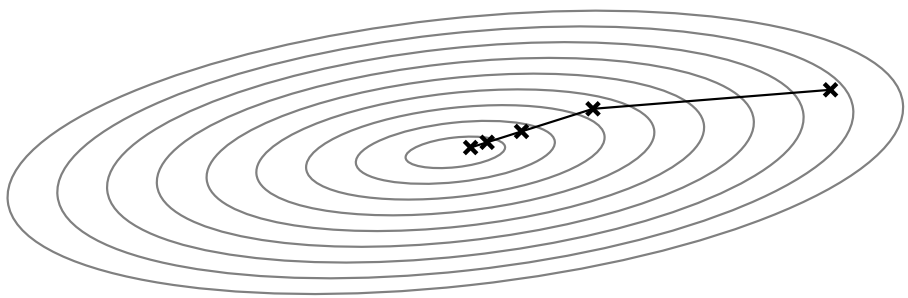


Scaled gradient descent – Example

- 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:

$$\hat{v}_k = b_v \hat{v}_{k-1} + (1 - b_v)(\widetilde{\nabla} f(x_{k-1}))^2$$

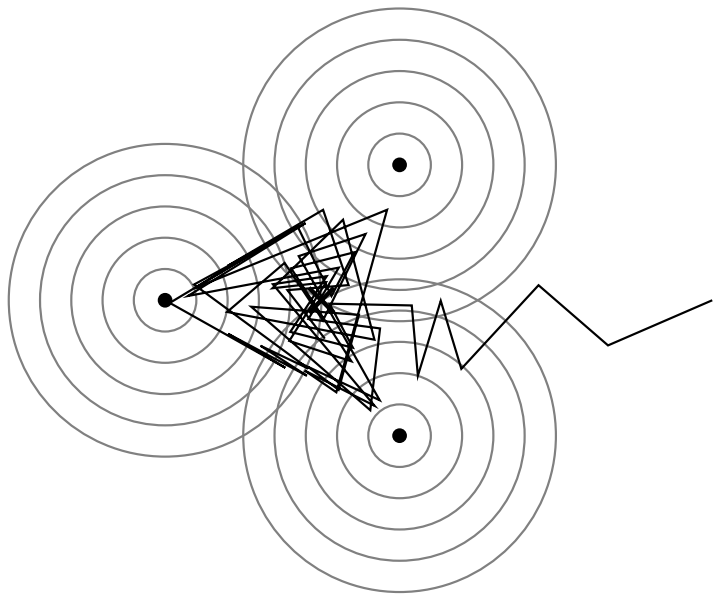
  where $\hat{v}_0 = 0$, $b_v \in (0, 1)$
- Metric H_k is chosen (approximately) as standard deviation:
  - RMSprop: biased estimate $H_k = \mathbf{diag}(\sqrt{\hat{v}_k} + \epsilon)$
  - Adam: unbiased estimate $H_k = \mathbf{diag}(\sqrt{\frac{\hat{v}_k}{1 - b_v^k}} + \epsilon)$
- Intuition:
  - Reduce 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



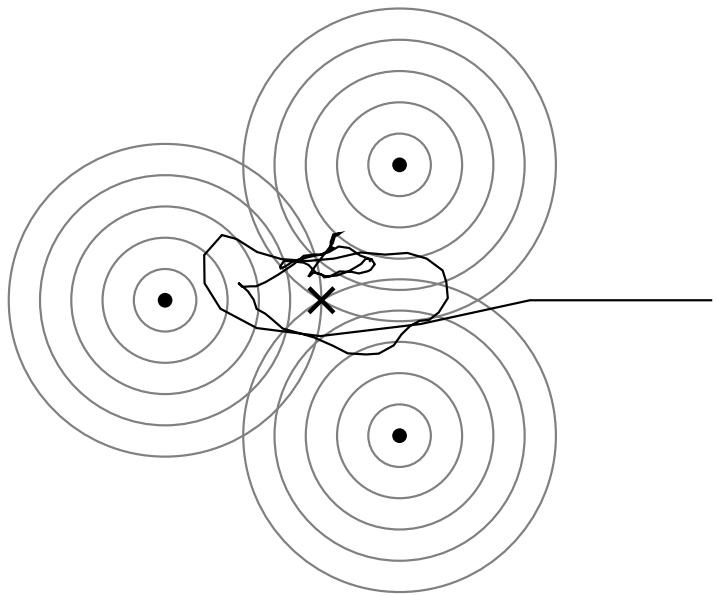
Levelsets of summands

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 $\hat{x} = \underset{x \in X}{\operatorname{argmin}}(\|x - x_0\|_H^2)$ 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:

$$Hx_{k+1} - Hx_0 = -\sum_{l=0}^{k} \gamma_l A^T(Ax_l - b) = A^T\lambda_k \in \mathcal{R}(A^T)$$

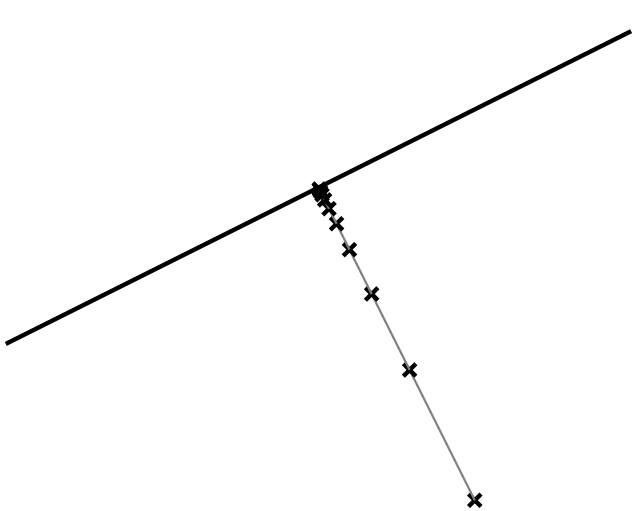
- In the limit $x_k \to \bar{x}$, we get

$$H\bar{x} - Hx_0 \in \mathcal{R}(A^T)$$

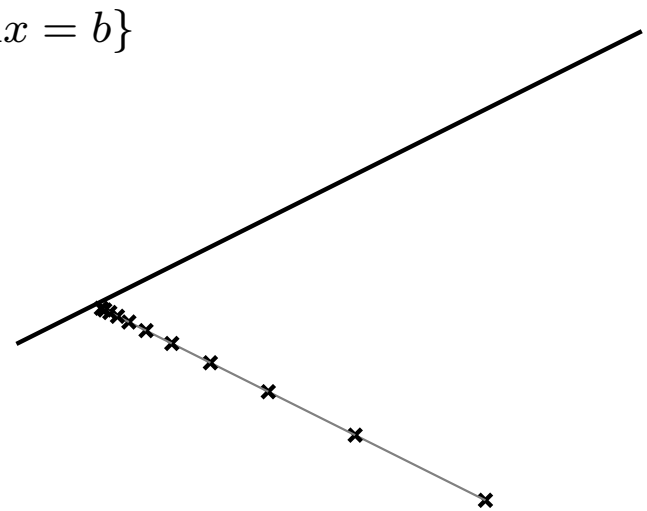
  which with $A\bar{x} = b$ gives optimality conditions for projection
- If $x_0 = 0$, the algorithm converges to $\operatorname*{argmin}_{x \in X}(\|x\|_H)$

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

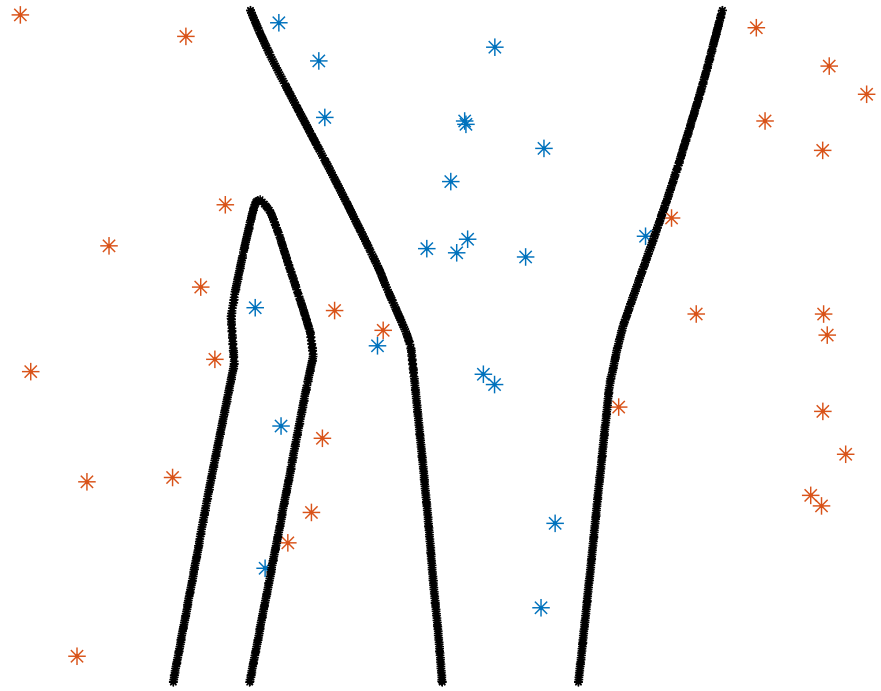
- Set scaling of weights by σ
- For the residual layers (all square layers)
  - $(W_j)_{ij} \sim \mathcal{N}(0,1)$, normalize W_j, scale by σ
  - $(b_j)_i \sim \mathcal{N}(0,1)$, normalize b_j, scale by σ
- For the non-residual layers (non-square layers)
  - $(W_j)_{ij} \sim \mathcal{N}(0,1)$, normalize 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

$\|\theta_0\|_2 = 3.57$ $\quad L_m = 8.4 \cdot 10^4$

$\|\theta_{\text{end}}\|_2 = 9.9$ $\quad \text{loss}(\theta_{\text{end}}) = 0.051$

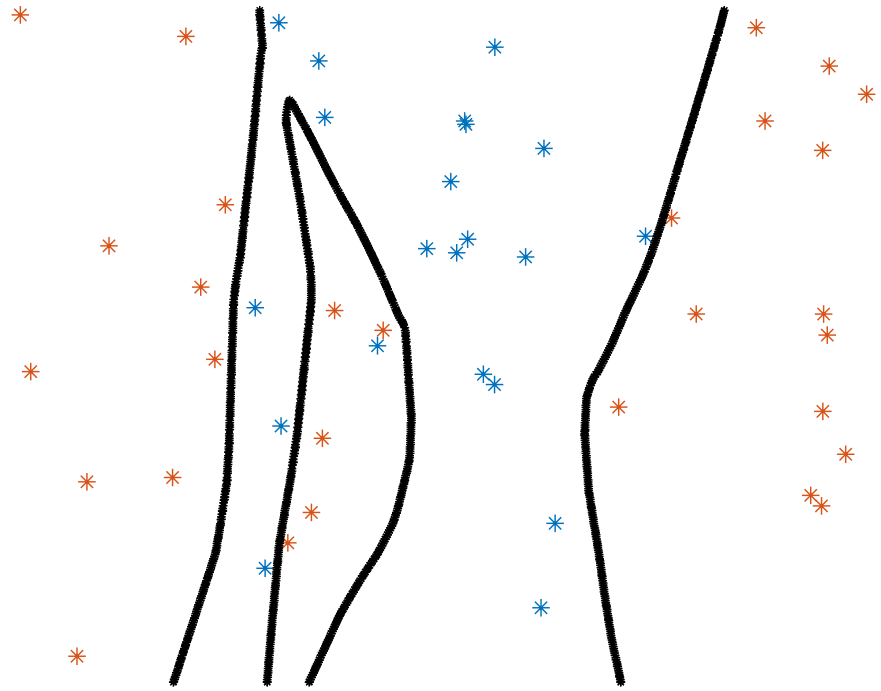


Convergence from different initial point

- 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

$\|\theta_0\|_2 = 3.8$ $\quad L_m = 2.0 \cdot 10^5$

$\|\theta_{\text{end}}\|_2 = 10.4$ $\quad \text{loss}(\theta_{\text{end}}) = 0.042$

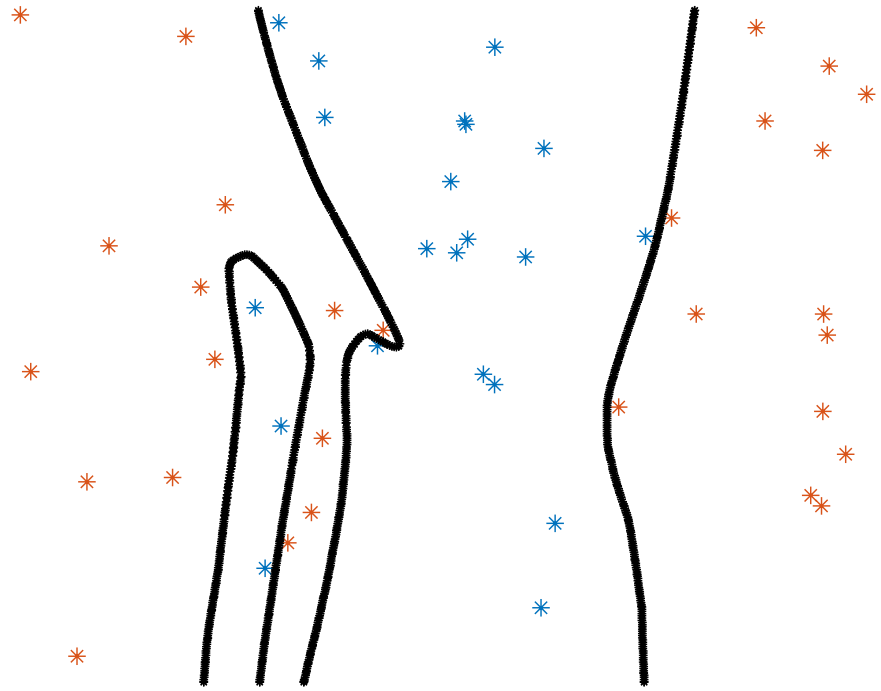


Convergence from different initial point

- 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

$\|\theta_0\|_2 = 10.8$ $\quad L_m = 2.4 \cdot 10^5$

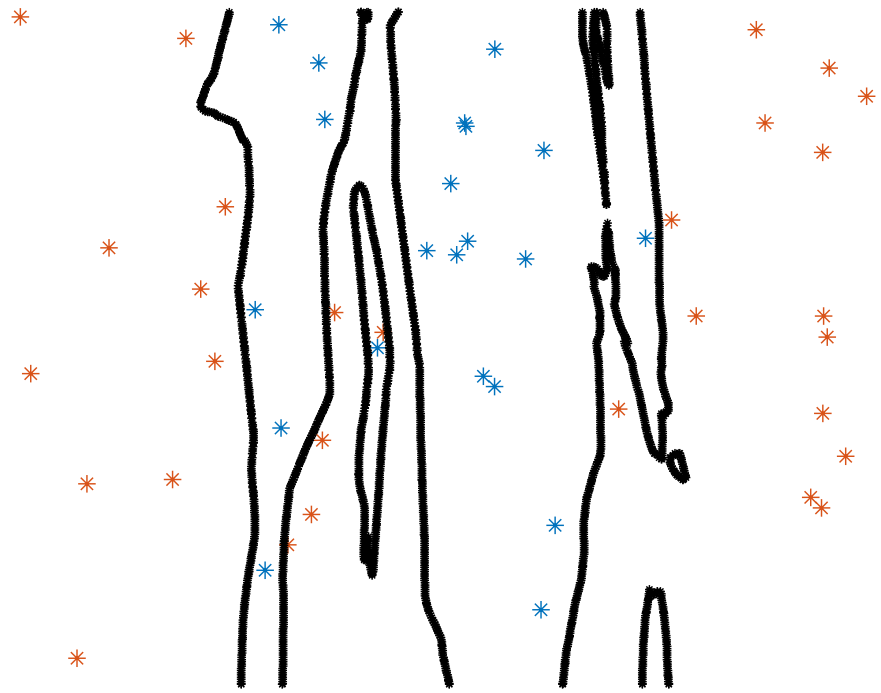
$\|\theta_{\text{end}}\|_2 = 14.4$ $\quad \text{loss}(\theta_{\text{end}}) = 0$



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

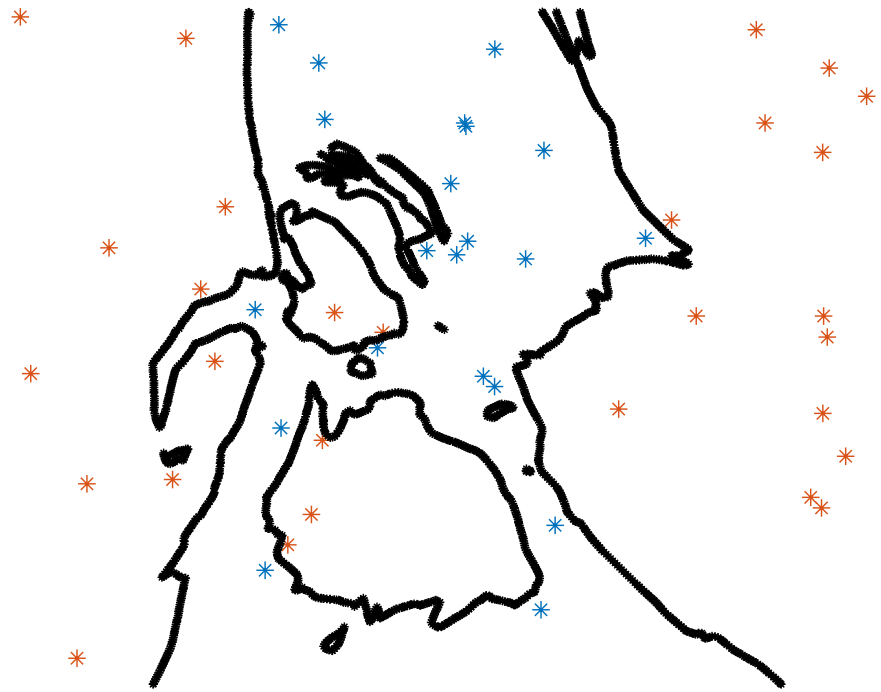
$$\|\theta_0\|_2 = 54.2 \qquad L_m = 1.9 \cdot 10^{12}$$
$$\|\theta_{\text{end}}\|_2 = 49.5 \qquad \text{loss}(\theta_{\text{end}}) = 0.036$$



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

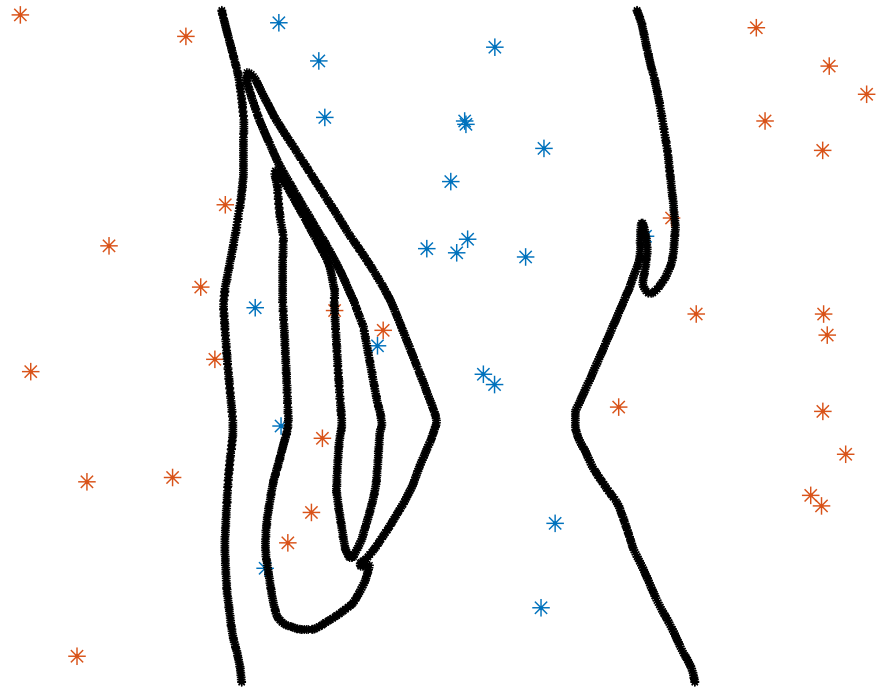
$$\|\theta_0\|_2 = 107.2 \qquad L_m = 1.6 \cdot 10^{15}$$
$$\|\theta_{\text{end}}\|_2 = 96.2 \qquad \text{loss}(\theta_{\text{end}}) = 0$$



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

$$\|\theta_0\|_2 = 3.6 \qquad L_m = 9.3 \cdot 10^7$$
$$\|\theta_{\text{end}}\|_2 = 17.4 \qquad \text{loss}(\theta_{\text{end}}) = 0.12$$

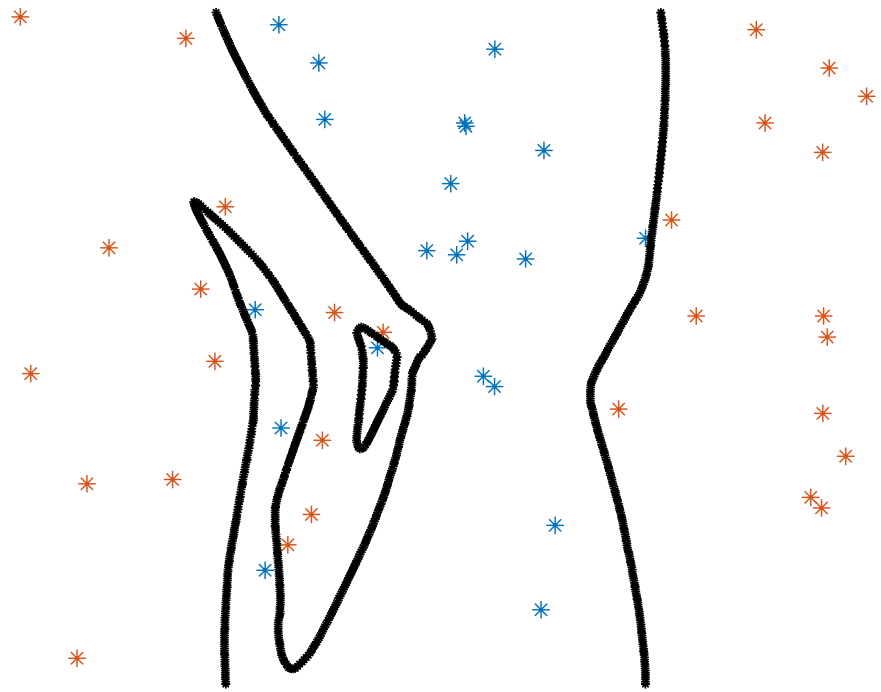


Convergence from different initial point

- 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

$\|\theta_0\|_2 = 3.9$ $\quad L_m = 4.5 \cdot 10^7$

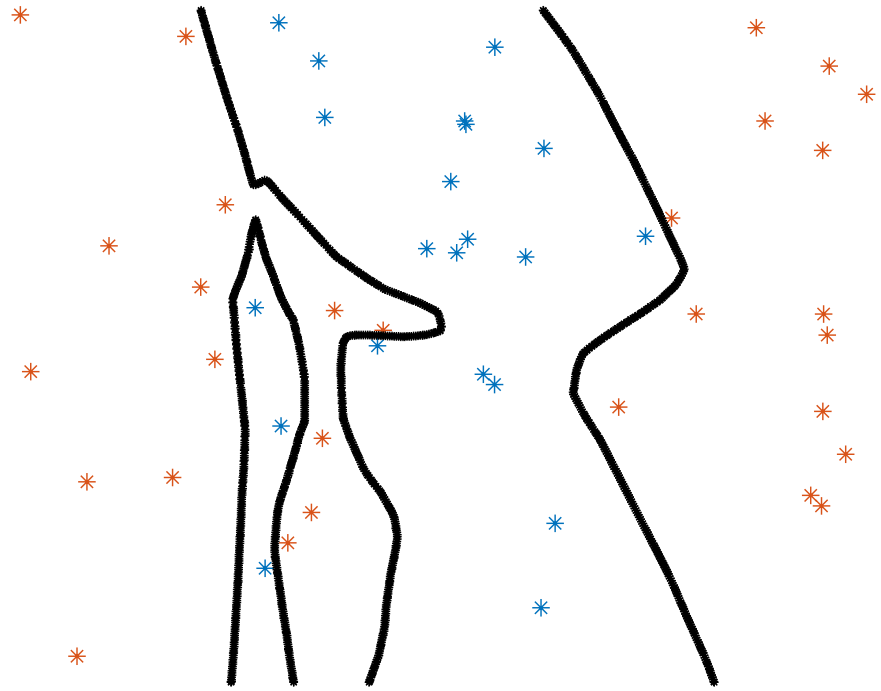
$\|\theta_{\text{end}}\|_2 = 16.2$ $\quad \text{loss}(\theta_{\text{end}}) = 0$



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

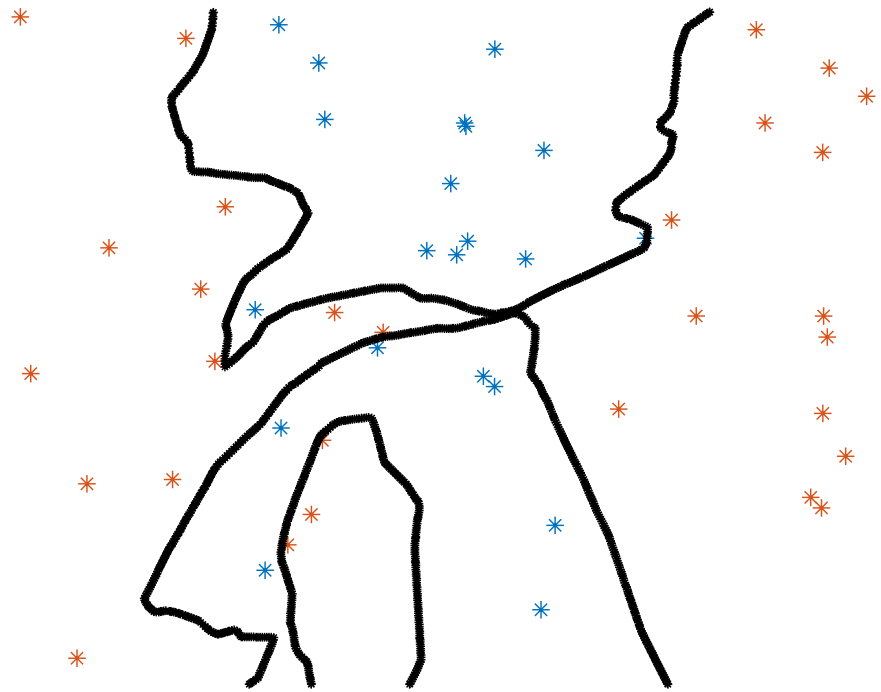
$$\|\theta_0\|_2 = 10.7 \qquad L_m = 4.3 \cdot 10^7$$
$$\|\theta_{\text{end}}\|_2 = 18.7 \qquad \text{loss}(\theta_{\text{end}}) = 0$$



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

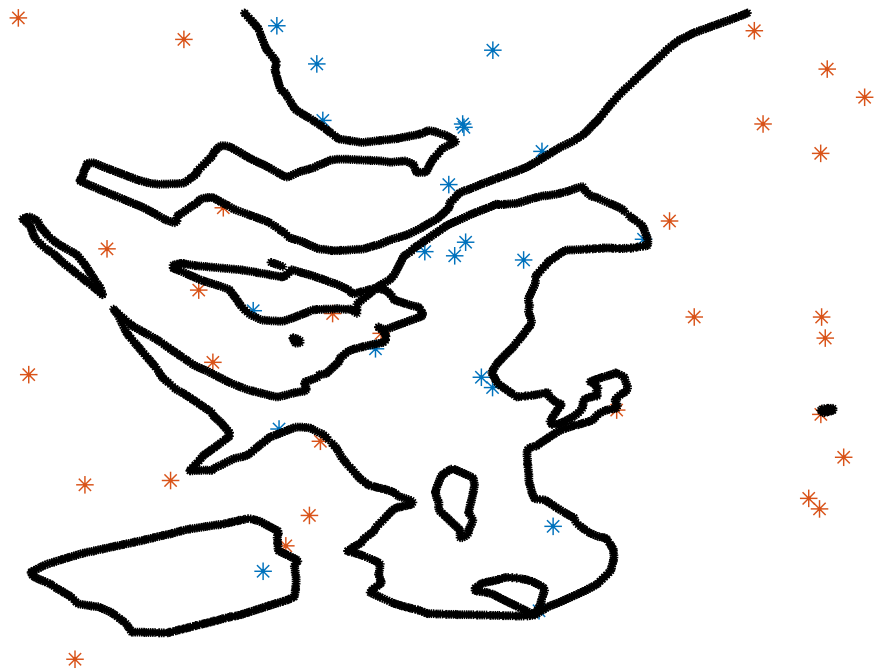
$$\|\theta_0\|_2 = 54.61 \qquad L_m = 1.9 \cdot 10^{12}$$
$$\|\theta_{\text{end}}\|_2 = 54.61 \qquad \text{loss}(\theta_{\text{end}}) = 0$$



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

$$\|\theta_0\|_2 = 109.278 \qquad L_m = 3.8 \cdot 10^{16}$$
$$\|\theta_{\text{end}}\|_2 = 109.282 \quad \text{loss}(\theta_{\text{end}}) = 0$$



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

$$x_{k+1} = x_k - \gamma_k H_k^{-1} \nabla f(x_k) \tag{1}$$

  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)

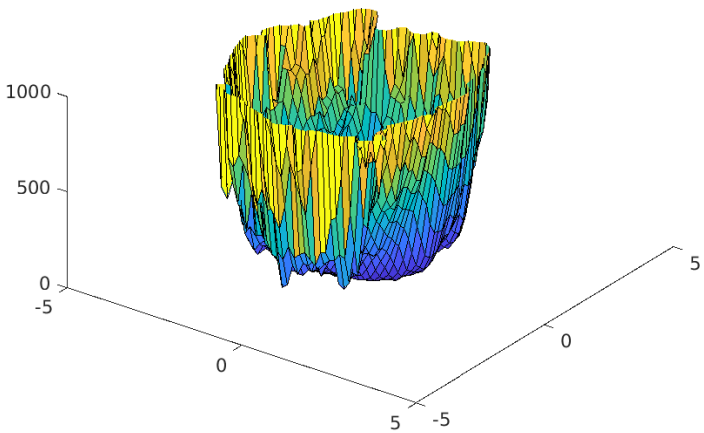
Interpretation consequence

- 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

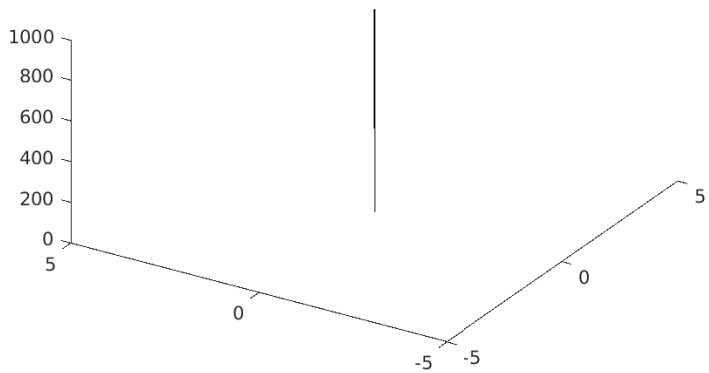
Adam vs SGD – Flat or sharp minima

- 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



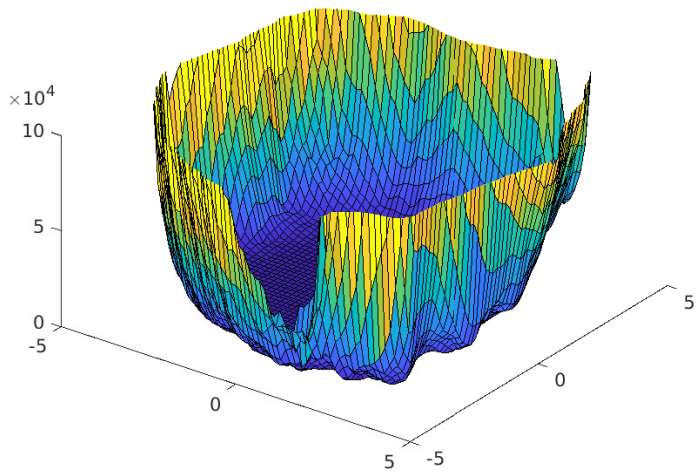
Adam



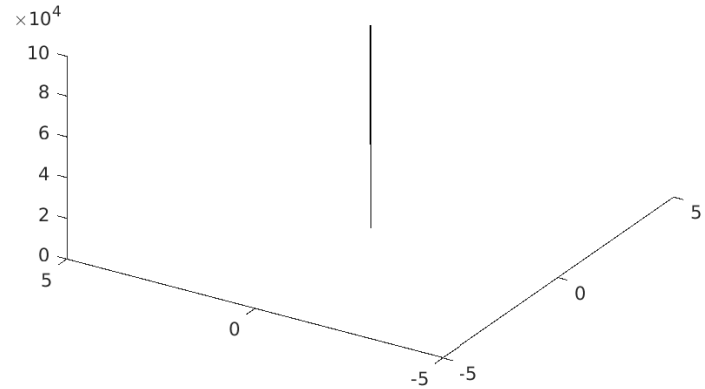
Adam vs SGD – Flat or sharp minima

- 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



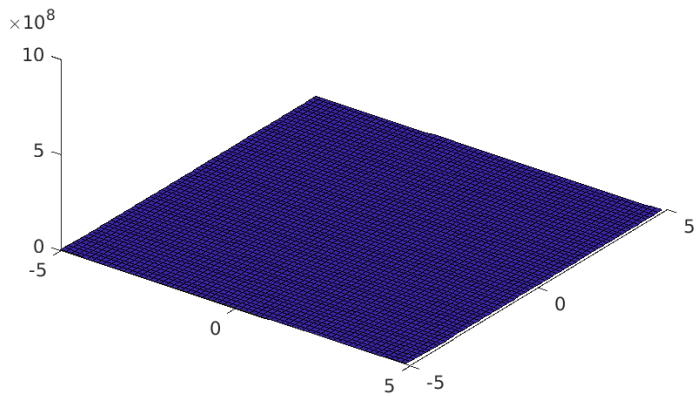
Adam



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