# Machine Learning: Optimization 3

Hiroshi Shimodaira and Hao Tang

2025

*Ver. 1.0*

# Topics

- Gradient descent
- Step size / learning rate
- Approximate solution
- Convergence rate
- Stochastic gradient descent (SGD)
- Mini-batch and epoch

# Convexity on more points

If a function $f$ is convex,

$$f(\alpha_1 \boldsymbol{x}_1 + \alpha_2 \boldsymbol{x}_2 + \alpha_3 \boldsymbol{x}_3) \leq \alpha_1 f(\boldsymbol{x}_1) + \alpha_2 f(\boldsymbol{x}_2) + \alpha_3 f(\boldsymbol{x}_3) \tag{1}$$

for $\alpha_1, \alpha_2, \alpha_3 \geq 0$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

# Convexity on more points (*cont.*)

If a function $f$ is convex,

$$f\left(\sum_{i=1}^{N} \alpha_i \boldsymbol{x}_i\right) \leq \sum_{i=1}^{N} \alpha_i f(\boldsymbol{x}_i) \tag{2}$$

for $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i = 1$.

# Jensen's inequality

If a function $f$ is convex,

$$f(\mathbb{E}[\boldsymbol{x}]) \leq \mathbb{E}[f(\boldsymbol{x})]. \tag{3}$$

$$f(\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[\boldsymbol{x}]) \leq \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[f(\boldsymbol{x})].$$

- For log loss

$$\mathrm{NLL} = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i \boldsymbol{w}^\top \phi(\boldsymbol{x}_i))\right) \tag{4}$$

  we cannot even solve $\nabla_{\boldsymbol{w}} \mathrm{NLL} = \mathbf{0}$.
  (i.e. no closed-form solution)

- How do we find the optimal solution?

# Gradient descent

- Gradient descent is an iterative algorithm, consisting of the steps

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \text{NLL}(\boldsymbol{w}_t). \tag{5}$$

- The variable $\eta_t > 0$ is called the **step size** (or learning rate), and can depend on $t$.

# Gradient descent on log loss

- The log loss in the binary case

$$\text{NLL} = \sum_{i=1}^{N} \log\left(1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)\right). \tag{6}$$

- We have shown that NLL is convex in $\boldsymbol{w}$.

# Gradient descent on log loss (*cont.*)

$$\frac{\partial \text{NLL}}{\partial \boldsymbol{w}} = \sum_{i=1}^{N} \frac{\exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)}{1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)} (-y_i \boldsymbol{x}_i) \tag{7}$$

$$= \sum_{i=1}^{N} \left(1 - \frac{1}{1 + \exp(-y_i \boldsymbol{w}^\top \boldsymbol{x}_i)}\right) (-y_i \boldsymbol{x}_i) \tag{8}$$

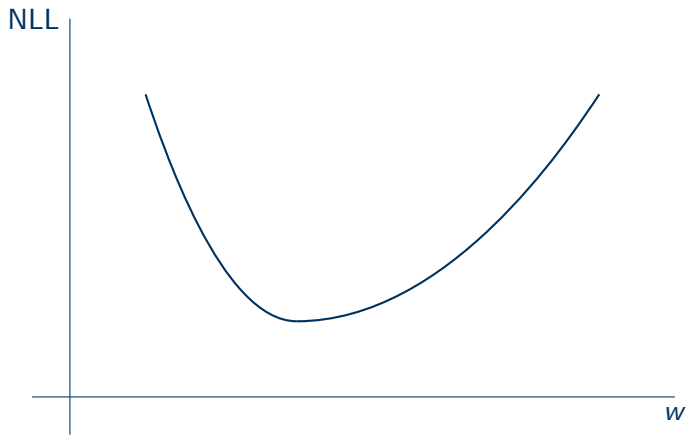
$$= \sum_{i=1}^{N} (1 - p(y_i|\boldsymbol{x}_i)) (-y_i \boldsymbol{x}_i) \tag{9}$$

# Gradient descent on log loss (*cont.*)

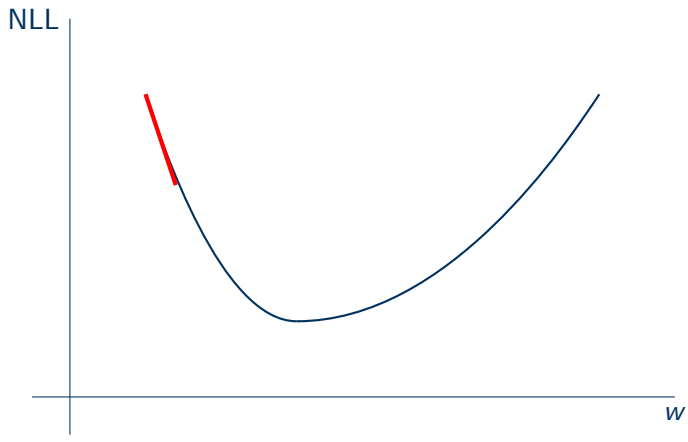
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \text{NLL}(\boldsymbol{w}_t) \tag{10}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \sum_{i=1}^{N} \left(1 - \frac{1}{1 + \exp(-y_i \boldsymbol{w}_t^\top \boldsymbol{x}_i)}\right)(-y_i \boldsymbol{x}_i) \tag{11}$$

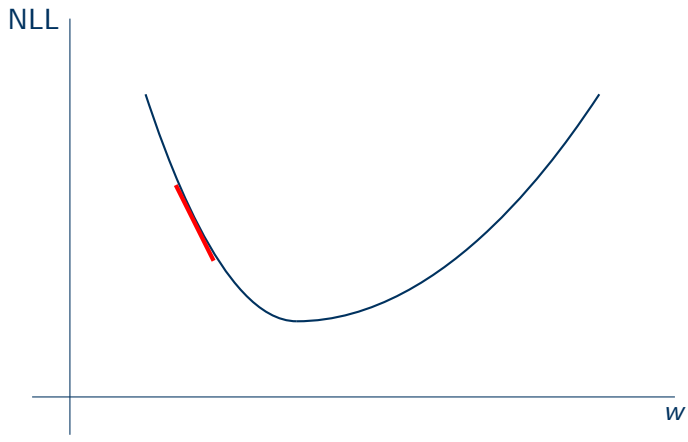
# Gradient descent



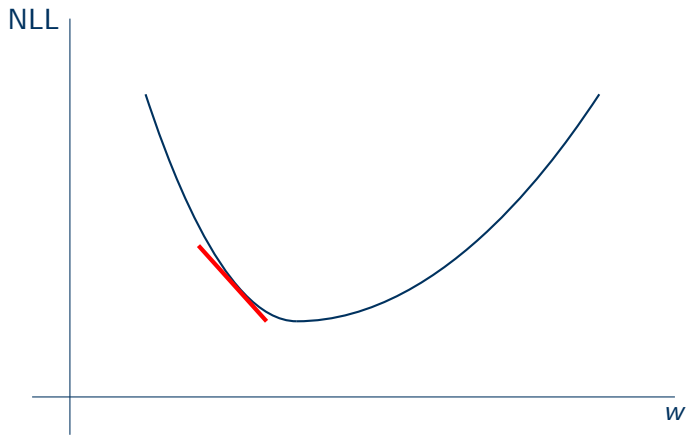
# Gradient descent



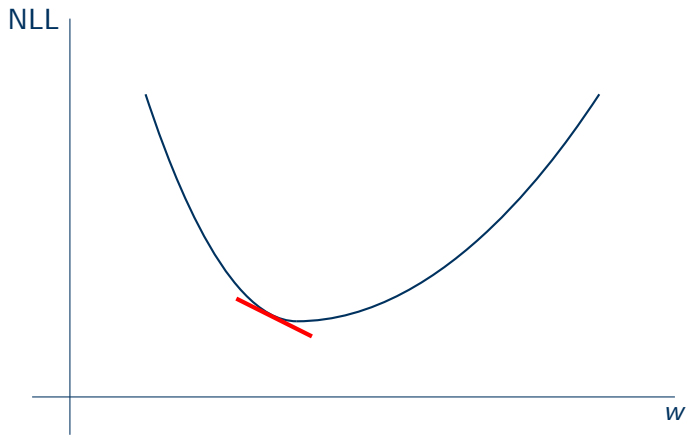
# Gradient descent



# Gradient descent



# Gradient descent



# Approximate solutions in optimisation

- We say that $\hat{\boldsymbol{x}}$ is an approximate solution of the minimiser $\boldsymbol{x}^*$ if, for a given $\epsilon > 0$,

$$f(\hat{\boldsymbol{x}}) - f(\boldsymbol{x}^*) < \epsilon. \tag{12}$$

- Note that it is close in function value, not close in the input.

# Approximate solutions for iterative algorithms

- An iterative algorithm creates a sequence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t$.
- How many updates do we need to achieve an approximate solution?
- Given $\epsilon > 0$, how large does $t$ needs to be to achieve

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) < \epsilon? \tag{13}$$

- We want to express $\epsilon$ as a function of $t$.

# Rate of convergence

$$\lim_{t \to \infty} \frac{|f(\boldsymbol{x}_{t+1}) - f(\boldsymbol{x}^*)|}{|f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*)|^q} = \mu \tag{14}$$

The sequence $\{f(\boldsymbol{x}_t)\}$ is said to *converge* with order $q$ to $f(\boldsymbol{x}^*)$ and with a *rate of convergence* $\mu$

- For $q = 1$
  - If $\mu = 1$, *sublinear rate*
  - If $0 < \mu < 1$, *linear rate*
  - if $\mu = 0$, *superlinear rate*

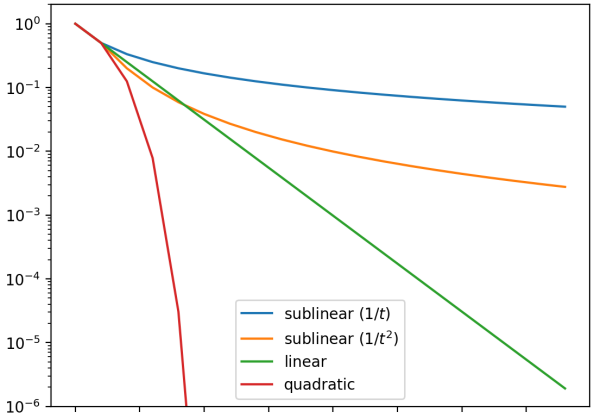
# Potential results

- Sublinear
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq \frac{c}{t^2}$
- Linear
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq cr^t$ for $0 < r < 1$
- Quadratic
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq cr^{2^t}$ for $0 < r < 1$

# Potential results

- Sublinear
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq \frac{c}{t^2}$
  - $\epsilon = O\left(\frac{1}{t^2}\right)$ or $t = O(\frac{1}{\sqrt{\epsilon}})$
- Linear
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq cr^t$ for $0 < r < 1$
  - $\epsilon = O\left(2^{-t}\right)$ or $t = O(\log \frac{1}{\epsilon})$
- Quadratic
  - $f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq cr^{2^t}$ for $0 < r < 1$
  - $\epsilon = O\left(2^{-2^t}\right)$ or $t = O(\log \log \frac{1}{\epsilon})$

# Convergence rates



# Gradient descent on mean-squared error

- The mean-squared error is

$$L = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|_2^2. \tag{15}$$

- We have shown that $L$ is convex in $\boldsymbol{w}$.
- We have shown that the optimal solution is $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$.

# Gradient descent on mean-squared error (*cont.*)

$$\nabla L = 2(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\top \boldsymbol{y}) \tag{16}$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla L(\boldsymbol{w}_t) = \boldsymbol{w}_t - \eta_t 2(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w}_t - \boldsymbol{X}^\top \boldsymbol{y}) \tag{17}$$

# Runtime comparison for solving mean-squared error

- The runtime of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ is $O(nd^2)$ or $O(d^3)$ (whichever dominates).

# Runtime comparison for solving mean-squared error

- The runtime of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ is $O(nd^2)$ or $O(d^3)$ (whichever dominates).

# Runtime comparison for solving mean-squared error

- The runtime of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$ is $O(nd^2)$ or $O(d^3)$ (whichever dominates).
- The runtime of a single gradient step $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t 2(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{w}_t - \boldsymbol{X}^\top \boldsymbol{y})$ is $O(nd)$.

# Convergence rates of gradient descent on mean-squared error

- If we run gradient descent on mean-squared error, we have

$$L(\boldsymbol{w}_t) - L(\boldsymbol{w}^*) = \frac{1}{2}(\boldsymbol{w}_0 - \boldsymbol{w}^*)^\top (\mathbf{I} - \eta \boldsymbol{H})^{2t} \boldsymbol{H} (\boldsymbol{w}_0 - \boldsymbol{w}^*) \tag{18}$$

  where $\boldsymbol{H} = \boldsymbol{X}^\top \boldsymbol{X}$ is the Hessian of $L$.

- If we choose $\eta = \frac{1}{\lambda_{\max}}$, where $\lambda_{\max}$ is the largest eigenvalue of $H$, the convergence is linear.

# Stochastic gradient descent (SGD)

1. Sample $\boldsymbol{x}_t, \boldsymbol{y}_t$ from a data set $S$.
2. $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \ell(\boldsymbol{w}_t; \boldsymbol{x}_t, \boldsymbol{y}_t)$
   - Per sample $L_2$ loss $\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{w}^\top \boldsymbol{x}_t - y_t)^2$
   - Per sample log loss $\ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) = \log(1 + \exp(-y_t w^\top \boldsymbol{x}_t))$
3. Go to 1 until the solution is satisfactory.

# Stochastic gradient descent (SGD) (*cont.*)

- $\nabla \ell(\boldsymbol{w}_t; \boldsymbol{x}_t, y_t)$ is now random, because $\boldsymbol{x}_t$ and $y_t$ is random.
- The expectation

$$\mathbb{E}_{x,y \sim U(S)}[\nabla \ell(w; x, y)] = \nabla L(w) \tag{19}$$

where $U(S)$ is the uniform distribution over the samples in $S$.

# Guarantee for stochastic gradient descent

- If we do SGD on an convex function,

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim U(S)}[L(\bar{\boldsymbol{w}}_t)] - L(\boldsymbol{w}^*) \leq \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2^2}{2\eta t} + \frac{\eta B^2}{2}. \tag{20}$$

- $\|\nabla \ell(\boldsymbol{w}_t; \boldsymbol{x}, \boldsymbol{y})\|_2 \leq B$ for any $t$, $x$, and $y$
- $\bar{\boldsymbol{w}}_t = \frac{\boldsymbol{w}_1 + \cdots + \boldsymbol{w}_t}{t}$
- The runtime is $O(1/\sqrt{t})$ if we choose $\eta = \frac{\|\boldsymbol{w}_0 - \boldsymbol{w}^*\|_2}{B\sqrt{T}}$, independent of the data set size $n$!

# Mini-batch stochastic gradient descent

1. Sample a subset $S_t$ from a data set $S$.
2. $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta_t \nabla \frac{1}{|S_t|} \sum_{\boldsymbol{x},\boldsymbol{y} \in S_t} \ell(\boldsymbol{w}_t; \boldsymbol{x}, \boldsymbol{y})$
   - The random sampling maintains $\mathbb{E}_{S_t \sim U(S)^t} \left[ \nabla \frac{1}{|S_t|} \sum_{\boldsymbol{x},\boldsymbol{y} \in S_t} \ell(\boldsymbol{w}; \boldsymbol{x}, \boldsymbol{y}) \right] = \nabla L(\boldsymbol{w})$.
3. Go to 1 until the solution is satisfactory.

# Common practice and terminology

# Common practice and terminology

- When doing SGD, permute the data and then go in serial.

# Common practice and terminology

- When doing SGD, permute the data and then go in serial.
- A pass over the data set is called an **epoch**.

# Common practice and terminology

- When doing SGD, permute the data and then go in serial.
- A pass over the data set is called an **epoch**.
- When doing mini-batch SGD, remember to normalise by the batch size.

# Common practice and terminology

- When doing SGD, permute the data and then go in serial.
- A pass over the data set is called an **epoch**.
- When doing mini-batch SGD, remember to normalise by the batch size.
- With a larger batch size, we go over an epoch faster.

# Common practice and terminology

- When doing SGD, permute the data and then go in serial.
- A pass over the data set is called an **epoch**.
- When doing mini-batch SGD, remember to normalise by the batch size.
- With a larger batch size, we go over an epoch faster.
- Use the largest batch size you can afford.