Empirical Risk Minimization and Maximum Likelihood Estimation

CPT_S 434/534 Neural network design and application

Empirical risk minimization

$$\boxed{h_S^{\mathrm{ERM}}} = \operatorname*{argmin}_{h\in\mathcal{H}} \boxed{\widehat{R}_S(h)}.$$

Empirical risk

Plug in

$$R(h) - R^* = \boxed{\underbrace{\left(R(h) - \inf_{h\in\mathcal{H}} R(h)\right)}_{\text{estimation}}} + \boxed{\underbrace{\left(\inf_{h\in\mathcal{H}} R(h) - R^*\right)}_{\text{approximation}}}.$$

Controlled by constraints/regularization

Q: How to control?

Proposition 4.1 *For any sample S, the following inequality holds for the hypothesis returned by ERM:*

$$\mathbb{P}\left[\boxed{R(h_S^{\mathrm{ERM}}) - \inf_{h\in\mathcal{H}} R(h)} > \epsilon\right] \le \mathbb{P}\left[\sup_{h\in\mathcal{H}} \boxed{|R(h) - \widehat{R}_S(h)|} > \frac{\epsilon}{2}\right]. \qquad (4.3)$$

Corollary 3.19 (VC-dimension generalization bounds) *Let $\mathcal{H}$ be a family of functions taking values in $\{-1,+1\}$ with VC-dimension d. Then, for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $h \in \mathcal{H}$:*

$$\boxed{R(h) \le \widehat{R}_S(h)} + \sqrt{\frac{2d\log\frac{em}{d}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. = \boxed{O(\sqrt{1/m})} \qquad (3.29)$$

Empirical risk minimization

Definition 2.2 (Empirical error) *Given a hypothesis $h \in \mathcal{H}$, a target concept $c \in \mathcal{C}$, and a sample $S = (x_1, \ldots, x_m)$, the* empirical error *or* empirical risk *of h is defined by*

$$\boxed{\min_h} \widehat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \boxed{1_{h(x_i) \neq c(x_i)}}. \tag{2.2}$$

Too hard: need a surrogate

Maximum likelihood principle

Bernoulli distribution:

If X is a random variable with this distribution, then:

$$\Pr(\boxed{X = 1}) = p = 1 - \Pr(\boxed{X = 0}) = 1 - q.$$

The probability mass function f of this distribution, over possible outcomes *k*, is

$$f(k;p) = \begin{cases} p & \text{if } k = 1, \text{[2]} \\ q = 1 - p & \text{if } k = 0. \end{cases}$$

This can also be expressed as

$$f(k;p) = \boxed{p^k (1-p)^{1-k}} \quad \text{for } k \in \{0,1\}$$

(x_i, y_i) → p(x_i) is the **underlying** probability of x_i belonging to class 1 (y_i = 1)

Observe n samples (simultaneously)

$$\boxed{\prod_{i=1}^{n} p(x_i)^{y_i} (1 - p(x_i))^{1-y_i}}$$ Likelihood function

Screenshot from https://en.wikipedia.org/wiki/Bernoulli_distribution

Maximum likelihood estimation (MLE)

- Likelihood function

$$L(w) = P_w(X_1 = x_1, \ldots, X_n = x_n) = f(w; x_1) \times \cdots \times f(w; x_n) = \boxed{\prod_{i=1}^{n} f(w; x_i)}$$

approximate

$$\boxed{\prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i}}$$

Maximum likelihood estimation (MLE)

- Maximizing likelihood function

$$\max_{w} L(w) = \prod_{i=1}^{n} f(w; x_i)$$

Difficult to optimize

Maximum likelihood estimation (MLE)

- Maximizing likelihood function

Difficult to optimize

$$\max_w L(w) = \prod_{i=1}^{n} f(w; x_i)$$

- Maximizing log-likelihood function

Q: how to approximate the probability?

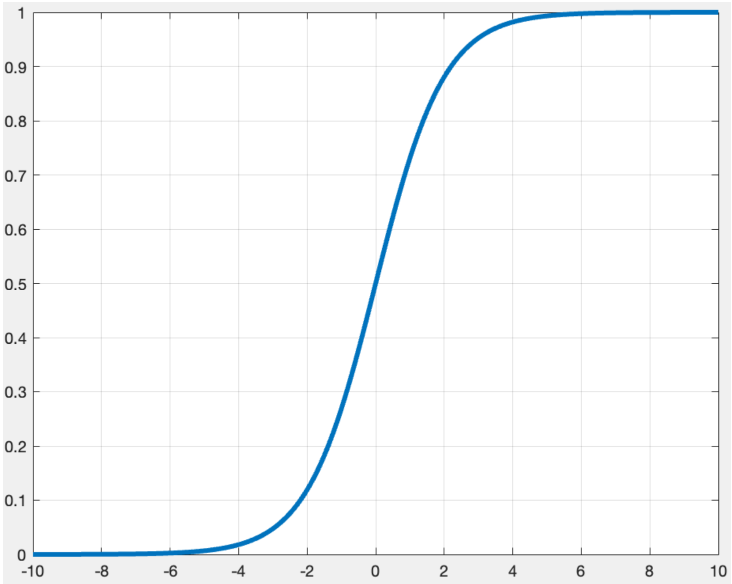
$$\max_w \log L(w) = \log \prod_{i=1}^{n} f(w; x_i) = \sum_{i=1}^{n} \log(\boxed{f(w; x_i)})$$

Easy to optimize

Logistic function

- Logistic regression

$$\min_{w} \frac{1}{n}\sum_{i=1}^{n} \log\left(1+e^{-(2y_i-1)f(w;x_i)}\right) = -\max_{w} \frac{1}{\mathrm{n}}\sum_{i=1}^{n} \log \boxed{\left(\frac{1}{1+e^{-(2y_i-1)f(w;x_i)}}\right)}$$



Probability=1

$$f_{sigmoid}(z) = \frac{1}{1+e^{-z}}$$

Probability=0

Softmax function

- Softmax function: generalization of logistic function to multiclass

Prediction: probability-like output

$$\left\{\begin{array}{l} f_{sigmoid}(z) = \dfrac{1}{1+e^{-z}} \\[2ex] f_{softmax}(z_k) = \dfrac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} \end{array}\right.$$

$z \in \mathbb{R}$: prediction to class 1

$z \in \mathbb{R}^K$: prediction to K classes

Normalization: summation of all elements=1

Softmax function

- Logistic model (binary classification)

$$\min_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(1 + e^{-(2y_i-1)f(w;x_i)}) = \underset{w}{min} -\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log\left(f_{sigmoid}((2y_i-1)f(w;x_i))\right)$$

Softmax function

- Logistic model (binary classification)

$$\min_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(1 + e^{-(2y_i - 1) f(w; x_i)}) = \min_{w} -\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log\left(f_{sigmoid}((2y_i - 1) f(w; x_i))\right)$$

- Softmax classifier (multiclass classification)

$$\min_{w} \boxed{-\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{i,k} \log\left(f_{softmax}(f(w; x_{i,k}))\right)}$$

Cross-entropy loss

One-hot
00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000

8 bits

Softmax function

- Logistic model (binary classification)

$$\min_w \frac{1}{\mathrm{n}}\sum_{i=1}^{n} \log\left(1+e^{-(2y_i-1)f(w;x_i)}\right) = \min_w -\frac{1}{\mathrm{n}}\sum_{i=1}^{n} \log\left(f_{sigmoid}((2y_i-1)f(w;x_i))\right)$$

- Softmax classifier (multiclass classification)

$$\min_w \boxed{-\frac{1}{\mathrm{n}}\sum_{i=1}^{n}\sum_{k=1}^{K} y_{i,k} \log\left(f_{softmax}(f(w;x_{i,k}))\right)}$$

Cross-entropy loss

One-hot
00000001
00000010
00000100
00001000
00010000
00100000
01000000
10000000

8 bits

$$\sum_{k=1}^{K} y_{i,k} \log\left(\frac{e^{f(w;x_{i,k})}}{\sum_{j=1}^{K} e^{f(w;x_{i,j})}}\right) = \sum_{k=1}^{K} y_{i,k} f(w;x_{i,k}) - \underbrace{\boxed{\sum_{k=1}^{K} y_{i,k}}}_{=1} \log\left(\sum_{j=1}^{K} e^{f(w;x_{i,j})}\right)$$

$$= \sum_{k=1}^{K} y_{i,k} f(w;x_{i,k}) - \log\left(\sum_{j=1}^{K} e^{f(w;x_{i,j})}\right)$$

(Usually used in deep learning)

Why is ERM general?

- Including many objective functions used in machine learning
- MLE: a special case of ERM
  - E.g., logistic regression

$$\min_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(1 + e^{-(2y_i - 1) f(w; x_i)})$$

Why is ERM general?

- Including many objective functions used in machine learning
- MLE: a special case of ERM
  - E.g., logistic regression

$$\min_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(1 + e^{-(2y_i - 1) f(w; x_i)})$$

$$= -\max_{w} -\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log\left(1 + e^{-(2y_i - 1) f(w; x_i)}\right)$$

Why is ERM general?

- Including many objective functions used in machine learning
- MLE: a special case of ERM
  - E.g., logistic regression

$$\min_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log(1 + e^{-(2y_i - 1) f(w; x_i)})$$

$$= -\max_{w} -\frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log\left(1 + e^{-(2y_i - 1) f(w; x_i)}\right)$$

$$= -\max_{w} \frac{1}{\mathrm{n}} \sum_{i=1}^{n} \log\left(\frac{1}{1 + e^{-(2y_i - 1) f(w; x_i)}}\right)$$

Determining model parameters

- An optimization problem (on training set)

Objective function

n training data

$$min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} l(f(w;x_i),y_i) = \frac{1}{2n}\sum_{i=1}^{n}(x_i{}'w - y_i)^2$$

- Analytical solution?

Determining model parameters

- An optimization problem (on training set)

Objective function — n training data

$$min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} l(f(w;x_i),y_i) = \frac{1}{2n}\sum_{i=1}^{n}(x_i'w - y_i)^2$$

- Analytical solution?

$$X = [x_1, x_2, \dots, x_n] \in \mathbb{R}^{d\times n}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

$$min_w \frac{1}{2n}\sum_{i=1}^{n}(x_i'w - y_i)^2$$

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i'wx_i - y_ix_i \;\rightarrow 0 \quad \Rightarrow \quad XX'w^* - XY = 0$$

Determining model parameters

- An optimization problem (on training set)

Objective function

n training data

$$min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} l(f(w; x_i), y_i) = \frac{1}{2n}\sum_{i=1}^{n}(x_i{}'w - y_i)^2$$

- Analytical solution?

$$X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d \times n}$$

$$min_w \frac{1}{2n}\sum_{i=1}^{n}(x_i{}'w - y_i)^2$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}'w x_i - y_i x_i \to 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

Determining model parameters

- An optimization problem (on training set)

Objective function

n training data

$$min_w f(w) = \frac{1}{n}\sum_{i=1}^{n} l(f(w;x_i), y_i) = \frac{1}{2n}\sum_{i=1}^{n}(x_i{}'w - y_i)^2$$

- Analytical solution?

$$X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d\times n}$$

$$min_w \frac{1}{2n}\sum_{i=1}^{n}(x_i{}'w - y_i)^2$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_n \end{bmatrix} \in \mathbb{R}^n$$

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}'wx_i - y_i x_i \to 0 \Rightarrow XX'w^* - XY = 0 \Rightarrow \boxed{w^* = (XX')^{-1}XY}$$

Q: is this closed form solution a good way in practice? Why?

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}'wx_i - y_i x_i \;\to\; 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Inverse of a scalar?

$$x\, x^{-1} = 1 \;\to x^{-1} = 1/x$$

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \ \rightarrow \ 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Inverse of a scalar?

$$x\, x^{-1} = 1 \ \rightarrow x^{-1} = 1/x$$

- Inverse of a matrix?

$$XX^{-1} = I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \to 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

$\boxed{\text{Matrix multiplication}}$	One $n \times m$ matrix & one $m \times p$ matrix	One $n \times p$ matrix	Schoolbook matrix multiplication	$\boxed{O(nmp)}$

$\boxed{\text{Matrix inversion}^*}$	One $n \times n$ matrix	One $n \times n$ matrix	Gauss–Jordan elimination	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms	$\boxed{O(n^{2.373})}$

Screenshot from https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#Matrix_algebra

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \rightarrow 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

Matrix multiplication	One $n \times m$ matrix & one $m \times p$ matrix	One $n \times p$ matrix	Schoolbook matrix multiplication	$O(nmp)$

Not every matrix has inversion

Matrix inversion*	One $n \times n$ matrix	One $n \times n$ matrix	Gauss–Jordan elimination	$O(n^3)$
			Strassen algorithm	$O(n^{2.807})$
			Coppersmith–Winograd algorithm	$O(n^{2.376})$
			Optimized CW-like algorithms	$O(n^{2.373})$

Screenshot from https://en.wikipedia.org/wiki/Computational_complexity_of_mathematical_operations#Matrix_algebra

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}'wx_i - y_i x_i \rightarrow 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Matrix multiplication:

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}' w x_i - y_i x_i \;\rightarrow\; 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Matrix multiplication:

$$\boxed{XX'\!: \mathrm{d} \times \mathrm{n} \times \mathrm{d}} \quad \boxed{XY\!: \mathrm{d} \times \mathrm{n}} \quad \boxed{(XX')^{-1}XY\!: d \times d \times n} \;\rightarrow O(d^2 n)$$

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \ \rightarrow \ 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Matrix multiplication:

$$\boxed{XX'\colon \mathrm{d} \times \mathrm{n} \times \mathrm{d}} \quad \boxed{XY\colon \mathrm{d} \times \mathrm{n}} \quad \boxed{(XX')^{-1}XY\colon d \times d \times n} \ \rightarrow O(d^2 n)$$

- Inverse of a matrix:

$$(XX')^{-1}\colon \ O(d^{2.373})$$

Determining model parameters

- Computational complexity for the analytical solution?

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i{}'wx_i - y_i x_i \ \rightarrow 0 \quad \Rightarrow \quad XX'w^* - XY = 0 \quad \Rightarrow \quad w^* = (XX')^{-1}XY$$

- Matrix multiplication:

$$\boxed{XX'\colon \mathrm{d} \times \mathrm{n} \times \mathrm{d}} \quad \boxed{XY\colon \mathrm{d} \times \mathrm{n}} \quad \boxed{(XX')^{-1}XY\colon d \times d \times n} \ \rightarrow O(d^2 n)$$

- Inverse of a matrix:

$$(XX')^{-1}\colon\ O(d^{2.373})$$

- Total complexity

$$O(d^2 n + d^{2.373})$$

Determining model parameters

- Gradient descent (GD)

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \;\rightarrow\; O(dn)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \;\rightarrow\; O(d)$$ → An iterative algorithm

Determining model parameters

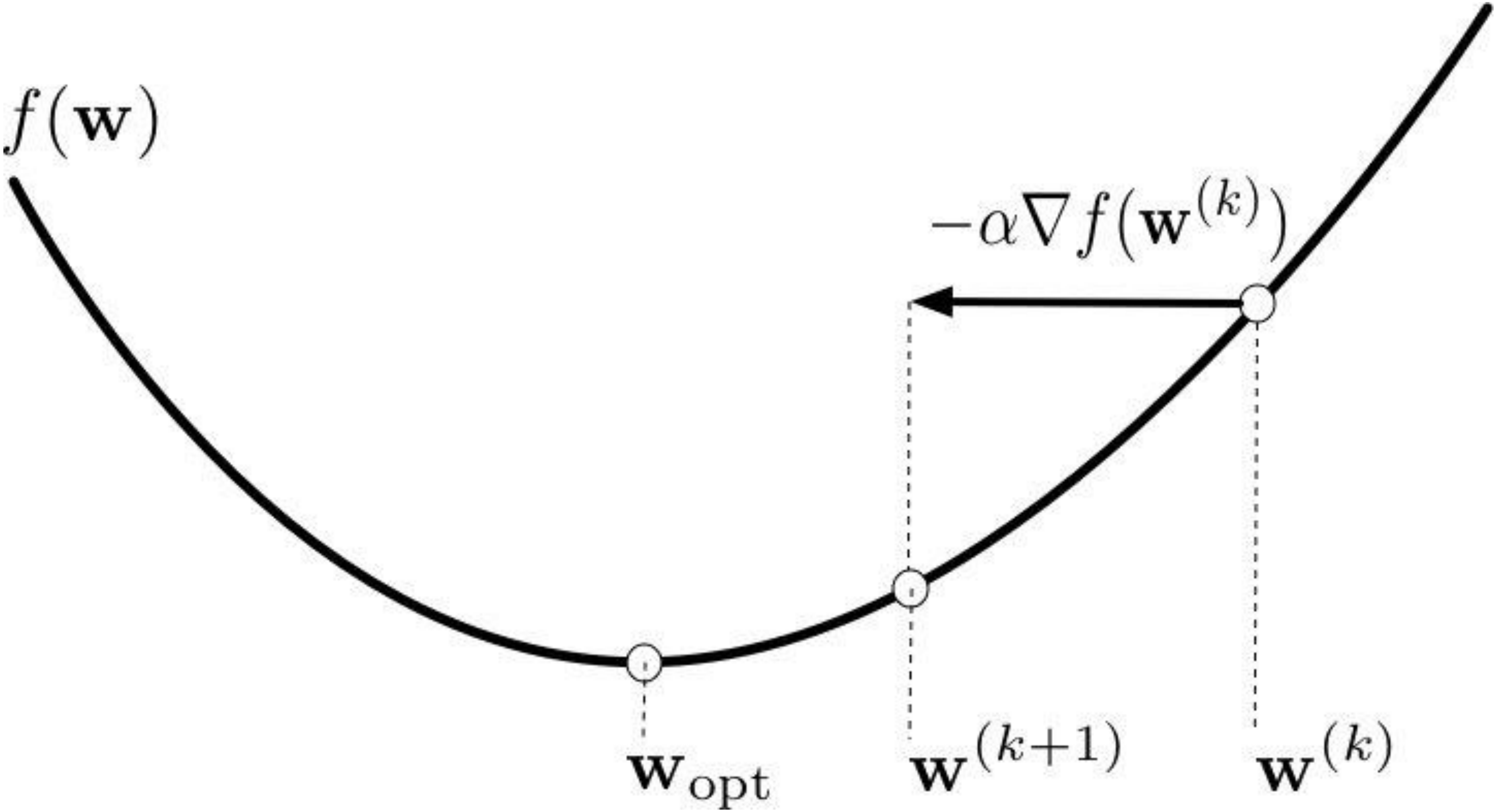
- Gradient descent (GD)

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \ \rightarrow \ O(dn)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \ \rightarrow \ O(d)$$

An iterative algorithm

Step size (learning rate):
usually pre-defined


$f(\mathbf{w})$
$-\alpha\nabla f(\mathbf{w}^{(k)})$
$\mathbf{w}_{\mathrm{opt}}$
$\mathbf{w}^{(k+1)}$
$\mathbf{w}^{(k)}$

[Jung2018]

Determining model parameters

- Gradient descent (GD)

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \ \rightarrow \ O(dn)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \ \rightarrow O(d)$$ → An iterative algorithm

- Suppose run GD for T iterations
- Total complexity

$$O(dnT)$$

Determining model parameters

- Gradient descent (GD)

$$\nabla_w f(w) = \frac{1}{n}\sum_{i=1}^{n} x_i' w x_i - y_i x_i \ \rightarrow O(dn)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \ \rightarrow O(d)$$ $\longrightarrow$ An iterative algorithm

- Suppose run GD for T iterations
- Total complexity

$O(dnT)$ VS. $O(d^2 n + d^{2.373})$ for the closed form solution

Determining model parameters

- When to terminate GD (determining T)?
  - Convergence rate for GD?

Theorem 2.1.14 *If $f \in \mathcal{S}^{1,1}_{\mu,L}(R^n)$ and $0 < h \leq \frac{2}{\mu+L}$ then the gradient method generates a sequence $\{x_k\}$ such that*

$$\| x_k - x^* \|^2 \leq \left(1 - \frac{2h\mu L}{\mu + L}\right)^k \| x_0 - x^* \|^2 .$$

If $h = \frac{2}{\mu+L}$ then

$$\| x_k - x^* \| \leq \left(\frac{Q_f - 1}{Q_f + 1}\right)^k \| x_0 - x^* \|,$$

$$f(x_k) - f^* \leq \frac{L}{2}\left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} \| x_0 - x^* \|^2,$$

where $Q_f = L/\mu$.

[Nesterov 2003]

Determining model parameters

- When to terminate GD (determining T)?
  - Convergence rate for GD?

Theorem 2.1.14 *If $f \in \mathcal{S}^{1,1}_{\mu,L}(R^n)$ and $0 < h \leq \frac{2}{\mu+L}$ then the gradient method generates a sequence $\{x_k\}$ such that*

$$\| x_k - x^* \|^2 \leq \left(1 - \frac{2h\mu L}{\mu + L}\right)^k \| x_0 - x^* \|^2 .$$

If $h = \frac{2}{\mu+L}$ then

$$\| x_k - x^* \| \leq \left(\frac{Q_f-1}{Q_f+1}\right)^k \| x_0 - x^* \|,$$

$$f(x_k) - f^* \leq \boxed{\frac{L}{2}\left(\frac{Q_f-1}{Q_f+1}\right)^{2k} \| x_0 - x^* \|^2,}$$

Approximated solution (not exact)

$$\boxed{= \epsilon(\mathrm{k}) = O(a^k)}$$

where $Q_f = L/\mu$.

[Nesterov 2003]

Determining model parameters

- When to terminate GD (determining T)?
  - Convergence rate for GD?

Theorem 2.1.14 *If $f \in \mathcal{S}^{1,1}_{\mu,L}(R^n)$ and $0 < h \leq \frac{2}{\mu+L}$ then the gradient method generates a sequence $\{x_k\}$ such that*

$$\| x_k - x^* \|^2 \leq \left(1 - \frac{2h\mu L}{\mu + L}\right)^k \| x_0 - x^* \|^2 .$$

If $h = \frac{2}{\mu+L}$ then

$$\| x_k - x^* \| \leq \left(\frac{Q_f - 1}{Q_f + 1}\right)^k \| x_0 - x^* \|,$$

$$f(x_k) - f^* \leq \boxed{\frac{L}{2}\left(\frac{Q_f - 1}{Q_f + 1}\right)^{2k} \| x_0 - x^* \|^2,}$$

where $Q_f = L/\mu$.

Approximated solution (not exact)

$\boxed{= \epsilon(\mathrm{k}) = O(a^k)}$ $\boxed{0 < a < 1}$

$\boxed{k = O(\,log_{1/a}(1/\epsilon))}$

[Nesterov 2003]

Determining model parameters

- Now we can answer the question:

  Can Gradient Descent (GD) do better?

GD	VS	Analytical solution
$\boxed{O(dnT)}$		$\boxed{O(d^2 n + d^{2.373})}$

Determining model parameters

- Now we can answer the question:

  Can Gradient Descent (GD) do better?

GD

$O(dnT)$

$O(dn \times log_{1/a}(1/\epsilon))$

VS

Analytical solution

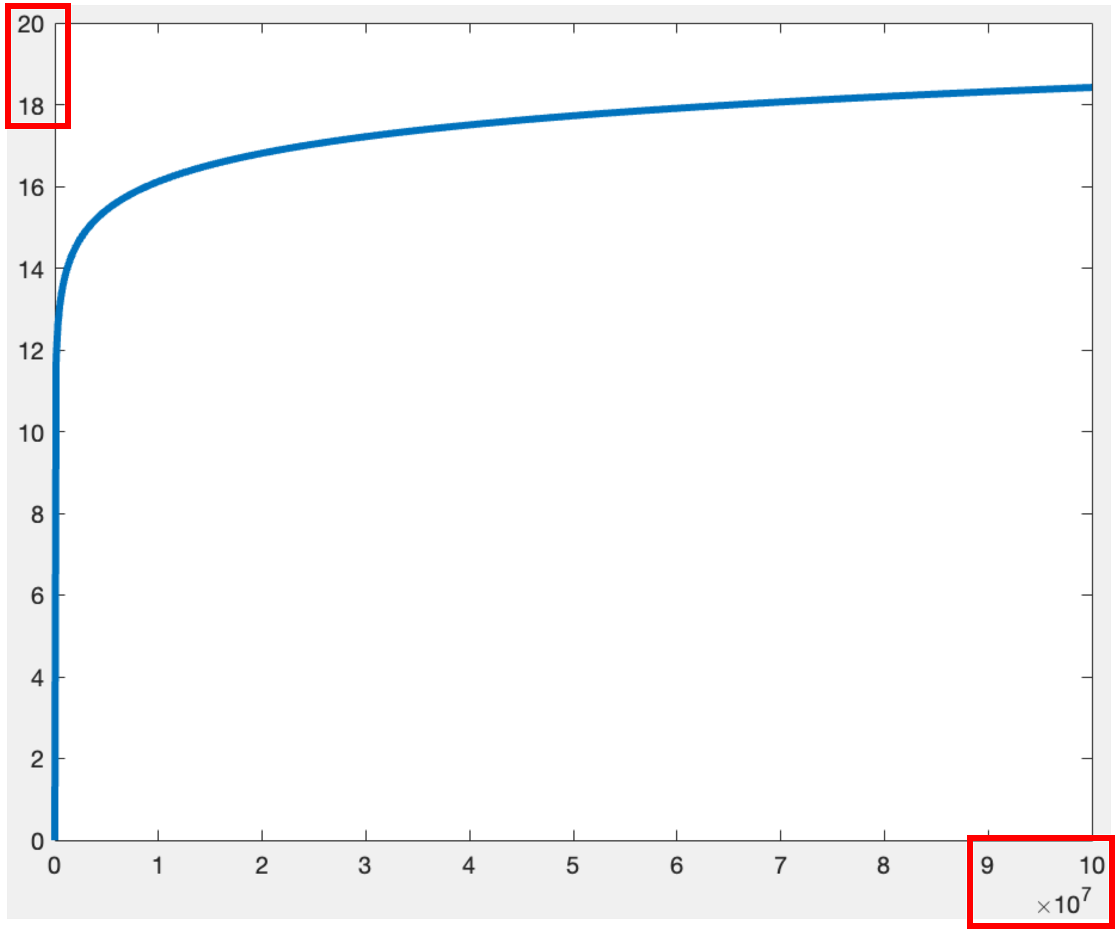
$O(d^2 n + d^{2.373})$

Determining model parameters

- Is log term $log_{1/a}(1/\epsilon)$ large?

Determining model parameters

- Is log term $log_{1/a}(1/\epsilon)$ large?



Determining model parameters

- Is d large?

name	source	type	class	training size	testing size	feature
a1a	UCI	classification	2	1,605	30,956	123
a2a	UCI	classification	2	2,265	30,296	123
a3a	UCI	classification	2	3,185	29,376	123
a4a	UCI	classification	2	4,781	27,780	123
a5a	UCI	classification	2	6,414	26,147	123
a6a	UCI	classification	2	11,220	21,341	123
a7a	UCI	classification	2	16,100	16,461	123
a8a	UCI	classification	2	22,696	9,865	123
a9a	UCI	classification	2	32,561	16,281	123
australian	Statlog	classification	2	690		14
avazu	Avazu's Click-through Prediction	classification	2	40,428,967	4,577,464	1,000,000
breast-cancer	UCI	classification	2	683		10
cod-rna	[AVU06a]	classification	2	59,535		8
colon-cancer	[AU99a]	classification	2	62		2,000
covtype.binary	UCI	classification	2	581,012		54
criteo	Criteo's Display Advertising Challenge	classification	2	45,840,617	6,042,135	1,000,000
criteo_tb	Criteo's Terabyte Click Logs	classification	2	4,195,197,692	178,274,637	1,000,000
diabetes	UCI	classification	2	768		8
duke breast-cancer	[MW01a]	classification	2	44		7,129
epsilon	PASCAL Challenge 2008	classification	2	400,000	100,000	2,000
fourclass	[TKH96a]	classification	2	862		2
german.numer	Statlog	classification	2	1,000		24
gisette	NIPS 2003 Feature Selection Challenge [IG05a]	classification	2	6,000	1,000	5,000
heart	Statlog	classification	2	270		13
HIGGS	UCI	classification	2	11,000,000		28
ijcnn1	[DP01a]	classification	2	49,990	91,701	22
ionosphere	UCI	classification	2	351		34
kdd2010 (algebra)	KDD CUP 2010	classification	2	8,407,752	510,302	20,216,830
kdd2010 (bridge to algebra)	KDD CUP 2010	classification	2	19,264,097	748,401	29,890,095
kdd2010 raw version (bridge to algebra)	KDD CUP 2010	classification	2	19,264,097	748,401	1,163,024
kdd2012	KDD CUP 2012	classification	2	149,639,105		54,686,452

1,000,000

1,000,000
1,000,000

20,216,830
29,890,095
1,164,024
54,686,452

Screenshot from https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Determining model parameters

- Is d large?

  Yes!

GD
$O(dnT)$

vs

Analytical solution
$O(d^2n + d^{2.373})$

$O(dn \times log_{1/a}(1/\epsilon))$

Determining model parameters

- Is d large?

  Yes!

GD

$O(dnT)$

vs

Analytical solution

$O(d^2 n + d^{2.373})$

$O(dn \times log_{1/a}(1/\epsilon))$

If:

$n = 10^8$

$d = 10^6$

$$(10^6)^2 \times 10^8 - 10^6 \times 10^8 = 10^{20} - 10^{14} \approx 10^{20}$$

$$O\left(dn \times log_a\left(\frac{1}{\epsilon}\right)\right) \ll O(d^2 n + n^{2.373})$$

Determining model parameters

- Stochastic gradient descent (SGD)

Randomly sample b data

b>=1

$$\nabla_w f(w) = \boxed{\frac{1}{b}\sum_{i=1}^{b}} x_i' w x_i - y_i x_i \;\rightarrow\; O(db)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \;\rightarrow\; O(d)$$

An iterative algorithm

Determining model parameters

- Stochastic gradient descent (SGD)

Randomly sample b data

b>=1

$$\nabla_w f(w) = \boxed{\frac{1}{b}\sum_{i=1}^{b}} x_i{}' w x_i - y_i x_i \;\rightarrow\; O(db)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \;\rightarrow\; O(d)$$

An iterative algorithm

Theorem 5 *Set the parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ in the* Epoch-GD *algorithm. The final point $\mathbf{x}_1^k$ returned by the algorithm has the property that*

$$\mathbb{E}[F(\mathbf{x}_1^k)] - F(\mathbf{x}^\star) \;\leq\; \frac{16G^2}{\lambda T}. \quad \boxed{= \epsilon(\mathrm{T}) \rightarrow T = O(\tfrac{1}{\epsilon})}$$

The total number of gradient updates is at most T.

Hazan, Elad, and Satyen Kale. "Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization." *The Journal of Machine Learning Research* 15, no. 1 (2014): 2489-2512.

Determining model parameters

Analytical solution
$O(d^2 n + d^{2.373})$

$dn \gg b/\epsilon$

$\epsilon = O(1/\sqrt{n})$

- Stochastic gradient descent (SGD)

Randomly sample b data

$b>=1$

$$\nabla_w f(w) = \boxed{\frac{1}{b}\sum_{i=1}^{b}} x_i' w x_i - y_i x_i \;\rightarrow O(db)$$

$$w_{t+1} = w_t - \alpha_t \nabla_w f(w_t) \;\rightarrow O(d)$$

An iterative algorithm

$O(db\frac{1}{\epsilon})$

Theorem 5 *Set the parameters $T_1 = 4$ and $\eta_1 = \frac{1}{\lambda}$ in the* Epoch-GD *algorithm. The final point $\mathbf{x}_1^k$ returned by the algorithm has the property that*

$$\mathbb{E}[F(\mathbf{x}_1^k)] - F(\mathbf{x}^\star) \leq \frac{16G^2}{\lambda T}.$$

$\boxed{= \epsilon(\mathrm{T}) \rightarrow T = O(\frac{1}{\epsilon})}$

The total number of gradient updates is at most T.

$O(dbT)$

Hazan, Elad, and Satyen Kale. "Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization." *The Journal of Machine Learning Research* 15, no. 1 (2014): 2489-2512.

References

- Jung, Alexander. (2018). Machine Learning: Basic Principles.
- Nesterov, Yurii. *Introductory lectures on convex optimization: A basic course*. Vol. 87. Springer Science & Business Media, 2003.