IoT·인공지능·빅데이터 개론 및 실습

Linear Models

서울대학교 전기·정보공학부
윤성로

Contents

Contents

1 Three Linear Models

(1) 세 가지 중요한 문제

- **Classification**
  - 범주 예측 (예: 남녀 사진 구분)
- **Regression**
  - 연속값 예측 (예: 몸무게 → 키)
- **Logistic regression**
  - 확률값 예측 (예: 혈압 → 심장마비 확률)

(2) Perceptron: linear classification

- **Model**
  - first, compute $s = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^\top \mathbf{x}$
  - then, get $h(\mathbf{x}) = \mathrm{sign}(s)$
- **Training: PLA**

1. start with an arbitrary weight vector $\mathbf{w}(0)$
2. then, at every time step $t \geq 0$

   2a. select *any* misclassified data point $(\mathbf{x}(t), y(t))$

   2b. update $\mathbf{w}$:
   $$\mathbf{w}(t+1) = \mathbf{w}(t) + y(t)\mathbf{x}(t)$$

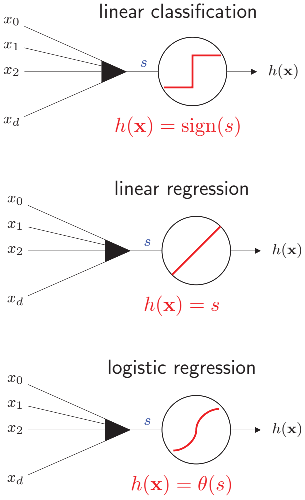
1 Three Linear Models

(3) 비교

- **“signal”**

$$s = \sum_{i=0}^{d} w_i x_i$$

- **“activation”**
  - different



❷ Advantages of Linear Models

(1) Simplicity

- Easy to implement, test, and interpret

(2) Generalization

- Higher chance of $\mathrm{E}_{\mathrm{in}} \approx \mathrm{E}_{\mathrm{out}}$ than complex models

(3) Extension

- Non-linear transform, kernel trick

  (예: support vector machine)
- Basis for artificial neural network

Contents

❶ Introduction

(1) A method to study relationship between $\mathbf{x}$ and y

$\mathbf{x}$ predictor variable/independent variable/feature

y response/dependent variable

(2) Training data $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

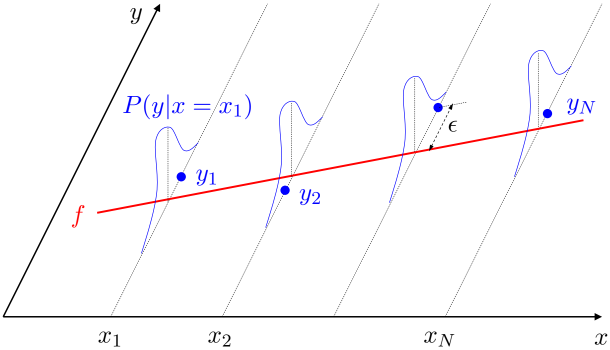
- Noise ϵ is added to target: $y_n = f(\mathbf{x}_n) + \epsilon$

$\Rightarrow$ $y \sim P(y|\mathbf{x})$ instead of $y = f(\mathbf{x})$

(3) Our goal

- Find a model $g(\mathbf{x})$ that approximates y_n

(4) Concept



Contents

❷ Learning Objective

(1) Minimize squared error:

$$\mathrm{E}_{\mathrm{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2 \qquad (1)$$

(2) Hypothesis h

- A linear combination of the components of x :

$$h(\mathbf{x}) = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^\top \mathbf{x} \qquad (2)$$

- $\mathrm{w}^\top \mathrm{x}$: also called **“signal”**

(3) $\mathbf{w}_{\text{lin}}$: the solution to linear regression

- Derived by minimizing $\mathrm{E}_{\text{in}}(\mathbf{w})$ over all $\mathbf{w} \in \mathbb{R}^{d+1}$

$$\mathbf{w}_{\text{lin}} = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} \mathrm{E}_{\text{in}}(\mathbf{w}) \tag{3}$$

$$= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}||X\mathbf{w} - \mathbf{y}||^2 \tag{4}$$

$$= \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^{d+1}} \frac{1}{N}(\mathbf{w}^\top X^\top X\mathbf{w} - 2\mathbf{w}^\top X^\top \mathbf{y} + \mathbf{y}^\top \mathbf{y}) \tag{5}$$

Contents

❸ Linear Regression Algorithm

(1) Minimizing $\mathrm{E}_{\mathrm{in}}(\mathbf{w})$

- $\mathrm{E}_{\mathrm{in}}(\mathrm{w})$: **continuous, differentiable, and convex**
- **Find** w **that minimizes** $\mathrm{E}_{\mathrm{in}}(\mathrm{w})$ **by requiring**

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \mathbf{0} \tag{6}$$

- **From Eq. (5)**

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}) = \frac{2}{N}(X^\top X\mathbf{w} - X^\top \mathbf{y}) \tag{7}$$

③ Linear Regression Algorithm

(2) The solution

- **Solve for w that satisfies the normal equations:**

$$X^\top X\mathbf{w} = X^\top \mathbf{y} \tag{8}$$

(3) Two scenarios

S1: if $X^\top X$ is invertible, $\boxed{\mathrm{w} = X^\dagger \mathbf{y}}$

- $\boxed{X^\dagger = (X^\top X)^{-1} X^\top}$ is *pseudo-inverse* of X
- resulting $\mathbf{w}$ is the unique optimal solution to (3)

S2: if $X^\top X$ is not invertible

- pseudo-inverse defined, but no unique solution
- many solutions for $\mathbf{w}$ that minimizes E_{in}

❸ Linear Regression Algorithm

(4) The linear regression algorithm

- **Construct matrix X and vector $\mathbf{y}$:**

$$\underbrace{X = \begin{bmatrix} — \mathbf{x}_1^\top — \\ — \mathbf{x}_2^\top — \\ \vdots \\ — \mathbf{x}_N^\top — \end{bmatrix}}_{\text{input data matrix}} \qquad \underbrace{\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}$$

- **Compute pseudo-inverse $X^\dagger$; if $X^\top X$ is invertible,**

$$X^\dagger = (X^\top X)^{-1} X^\top$$

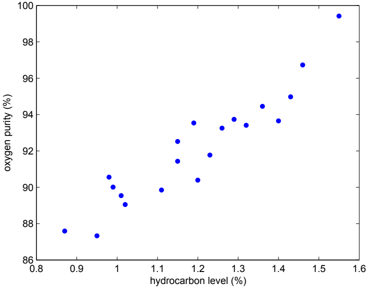
- **Return $\mathbf{w}_{\text{lin}} = X^\dagger \mathbf{y}$**

(5) Example: oxygen and hydrocarbon levels

no.	x	y
1	0.99	90.01
2	1.02	89.05
3	1.15	91.43
4	1.29	93.74
5	1.46	96.73
6	1.36	94.45
7	0.87	87.59
8	1.23	91.77
9	1.55	99.42
10	1.40	93.65
11	1.19	93.54
12	1.15	92.52
13	0.98	90.56
14	1.01	89.54
15	1.11	89.85
16	1.20	90.39
17	1.26	93.25
18	1.32	93.41
19	1.43	94.98
20	0.95	87.33

y: **purity (%) of oxygen**

x: **hydrocarbons (%)**



❸ Linear Regression Algorithm

(5) Example: oxygen and hydrocarbon levels

- **training data:**

$$\text{data matrix } X = \begin{bmatrix} 1 & 0.99 \\ 1 & 1.02 \\ 1 & 1.15 \\ 1 & 1.29 \\ 1 & 1.46 \\ 1 & 1.36 \\ 1 & 0.87 \\ 1 & 1.23 \\ 1 & 1.55 \\ 1 & 1.40 \\ 1 & 1.19 \\ 1 & 1.15 \\ 1 & 0.98 \\ 1 & 1.01 \\ 1 & 1.11 \\ 1 & 1.20 \\ 1 & 1.26 \\ 1 & 1.32 \\ 1 & 1.43 \\ 1 & 0.95 \end{bmatrix} \quad \text{target vector } \mathbf{y} = \begin{bmatrix} 90.01 \\ 89.05 \\ 91.43 \\ 93.74 \\ 96.73 \\ 94.45 \\ 87.59 \\ 91.77 \\ 99.42 \\ 93.65 \\ 93.54 \\ 92.52 \\ 90.56 \\ 89.54 \\ 89.85 \\ 90.39 \\ 93.25 \\ 93.41 \\ 94.98 \\ 87.33 \end{bmatrix}$$

(5) Example: oxygen and hydrocarbon levels

- $X^\top X$ is invertible

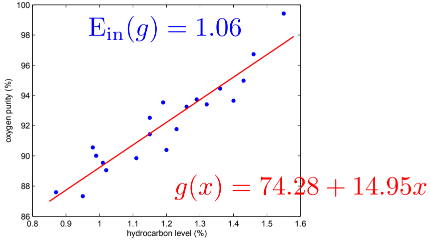
$$X^\top X = \begin{bmatrix} 20.00 & 23.92 \\ 23.92 & 29.29 \end{bmatrix} \Rightarrow (X^\top X)^{-1} = \begin{bmatrix} 2.15 & -1.76 \\ -1.76 & 1.47 \end{bmatrix}$$

- $(X^\top X)^{-1} X^\top \mathbf{y}$ yields

$$\mathbf{w}_{\text{lin}} = \begin{bmatrix} 74.28 \\ 14.95 \end{bmatrix}$$

- the learned model:

$$\begin{aligned} g(x) &= \mathbf{w}_{\text{lin}}^\top \mathbf{x} \\ &= 74.28 + 14.95x \end{aligned}$$



Contents

(1) Motivating example

- **Heart attack prediction:**
  - based on cholesterol level, blood pressure, age, ...
- **Cannot predict a heart attack with any certainty**
- **A more suitable model:**
  - output y that varies continuously between 0 and 1
  - the closer y is to 1, the higher chance of heart attack

(2) Logistic regression: ‘soft’ binary classification

- **Outputs probability of a binary response**
  - *e.g.* heart attack or not, dead or alive
  - returns ‘soft labels’ (probability)
- **Our new model: called logistic regression**
  - output: real (like regression) but bounded (like classification)
- **Linear classification vs logistic regression**
  - both deal with a binary event
  - logistic regression: allowed to be uncertain

(3) Logistic regression model

- **Linear classification: a hard threshold on signal**

$$h(\mathbf{x}) = \mathrm{sign}(\mathbf{w}^\top \mathbf{x})$$

- **Linear regression: no threshold**

$$h(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}$$

- **New model: needs something between these two**
  - smoothly restricts output to probability range [0,1]

$$h(\mathbf{x}) = \theta(\mathbf{w}^\top \mathbf{x})$$

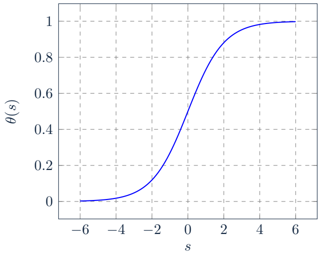
  - θ is so-called *logistic* function

(4) Logistic function θ

- **Definition**
  - for $-\infty < s < \infty$:

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

$$1-\theta(s) = \frac{e^{-s}}{1+e^{-s}} = \theta(-s)$$

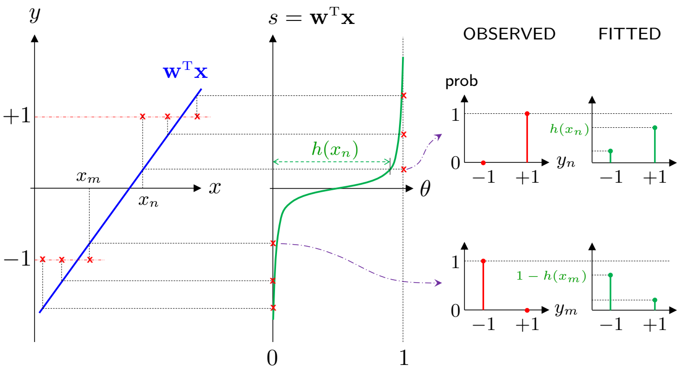


- **Output lies between 0 and 1**
  - can be interpreted as probability for binary events

Contents

2 Logistic Regression Model

(1) Big picture



2 Logistic Regression Model

(2) Learning target

- **Probability of event $y = +1$ given input $\mathbf{x}$**

$$\boxed{f(\mathbf{x}) = \mathbb{P}[y = +1 \mid \mathbf{x}]}$$

  - *e.g.* probability of a patient being at risk for heart attack given the characteristics of the patient

- **We view data as**
  - generated by target distribution $P(y|\mathbf{x})$

$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases} \tag{9}$$

❷ Logistic Regression Model

(3) Defining error measure

▶ Based on likelihood

- how 'likely' is it to get output y from input $\mathbf{x}$,
  if target distribution $P(y|\mathbf{x})$ was captured by $h(\mathbf{x})$?

▶ Based on (9), the likelihood would be

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1 \\ 1 - h(\mathbf{x}) & \text{for } y = -1 \end{cases}$$
$$= \boxed{\theta(y\mathbf{w}^\top \mathbf{x})} \tag{10}$$

- recall: $h(\mathbf{x}) = \theta(\mathbf{w}^\top \mathbf{x})$ and $1 - \theta(s) = \theta(-s)$

(4) Cross entropy

- **Consider two pmfs**
  - $\{p, 1-p\}$ and $\{q, 1-q\}$ with binary outcomes
  - *e.g.* $\mathbb{P}[\text{pass}] = p$ and $\mathbb{P}[\text{fail}] = 1 - p$
- **Cross entropy for these two pmfs:**

$$p \log \frac{1}{q} + (1-p) \log \frac{1}{1-q} \tag{11}$$

- **Cross entropy measures 'error' for approximating**
  - 'observed' pmf $\{p, 1-p\}$ by 'fitted' pmf $\{q, 1-q\}$

❷ Logistic Regression Model

(5) Cross entropy error measure

- **Recall:**

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1 \\ 1 - h(\mathbf{x}) & \text{for } y = -1 \end{cases} \tag{12}$$

$$= \theta(y\mathbf{w}^\top \mathbf{x}) \tag{13}$$

- **(12) can also be written as**

$$P(y|\mathbf{x}) = h(\mathbf{x})^{[\![y=+1]\!]}(1 - h(\mathbf{x}))^{[\![y=-1]\!]} \tag{14}$$

- **Likelihood of data** $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$

$$\prod_{n=1}^{N} P(y_n|\mathbf{x}_n) = \prod_{n=1}^{N} h(\mathbf{x}_n)^{[\![y_n=+1]\!]}(1 - h(\mathbf{x}_n))^{[\![y_n=-1]\!]}$$

(5) Cross entropy error measure

- **Negative log-likelihood (NLL) is given by**

$$
\begin{aligned}
NLL(\mathbf{w}) &\propto -\frac{1}{N}\log\left\{\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right\} \\
&= -\frac{1}{N}\log\left\{\prod_{n=1}^{N} h(\mathbf{x}_n)^{[\![y_n=+1]\!]}(1-h(\mathbf{x}_n))^{[\![y_n=-1]\!]}\right\} \\
&= \frac{1}{N}\sum_{n=1}^{N}\left\{[\![y_n=+1]\!]\log\frac{1}{h(\mathbf{x}_n)} + [\![y_n=-1]\!]\log\frac{1}{1-h(\mathbf{x}_n)}\right\}
\end{aligned}
$$

  - this is also called *cross-entropy* error

Contents

3 Training by Gradient Descent

(1) Training objective

- **Maximizing likelihood = minimizing NLL:**

$$-\frac{1}{N}\log\left\{\prod_{n=1}^{N} P(y_n|\mathbf{x}_n)\right\} = \frac{1}{N}\sum_{n=1}^{N}\log\frac{1}{P(y_n\mid\mathbf{x}_n)}$$

$$= \frac{1}{N}\sum_{n=1}^{N}\log\frac{1}{\theta(y_n\mathbf{w}^\top\mathbf{x}_n)}$$

- **Substituting functional form for θ gives**
  - in-sample error measure for logistic regression

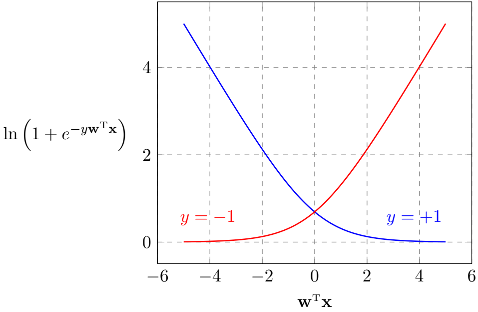
$$\boxed{\mathrm{E}_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\log\left(1+e^{-y_n\mathbf{w}^\top\mathbf{x}_n}\right)} \qquad (15)$$

③ Training by Gradient Descent

(1) Training objective

- **Implied pointwise error**

$$\mathbf{e}(h(\mathbf{x}_n), y_n) = \ln\left(1 + e^{-y_n \mathbf{w}^\top \mathbf{x}_n}\right)$$



③ Training by Gradient Descent

(2) Minimizing $\mathrm{E_{in}}$

- **Set $\nabla \mathrm{E_{in}}(\mathbf{w}) = 0$ and solve for $\mathbf{w}$**
- **Unfortunately, $\nabla \mathrm{E_{in}}$ for logistic regression:**
  - not easy to manipulate analytically

$$\nabla \mathrm{E_{in}}(\mathbf{w}) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n}} \tag{16}$$

$$= \frac{1}{N}\sum_{n=1}^{N} -y_n\mathbf{x}_n\theta(-y_n\mathbf{w}^{\mathrm{T}}\mathbf{x}_n) \tag{17}$$

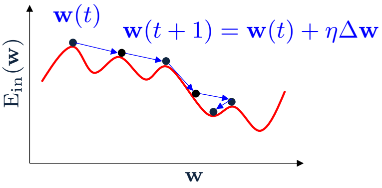
  - we need iterative optimization

3 Training by Gradient Descent

(3) Gradient descent

▶ Basic idea

- $E_{in}(\mathbf{w})$ is a 'surface' in high-dimensional space
- in step 0, we start somewhere on this surface, at $\mathbf{w}(0)$
- try to roll down the surface, thereby decreasing E_{in}



▶ Two things to decide

1. which direction?
2. how much?

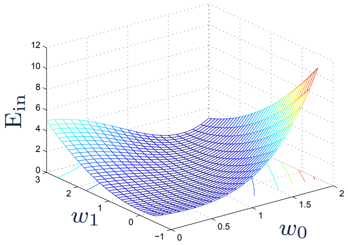
$$\Delta \mathbf{w} = -\nabla E_{in}(\mathbf{w}(t))$$

η : learning rate

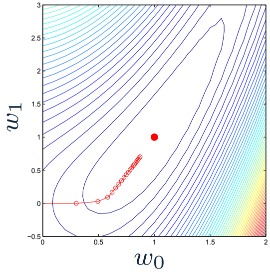
❸ Training by Gradient Descent

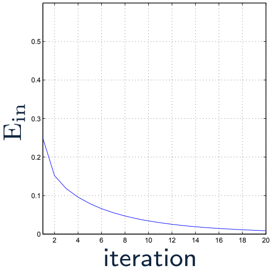
(4) Example

- **Global minimum 0 at (1,1)**



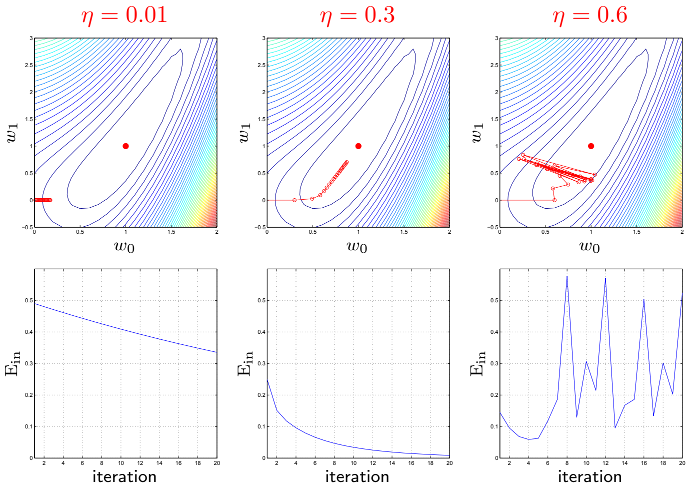
- **Start at (0,0)**
    - $\#$ iterations (steps) $= 20$
    - step size $\eta = 0.3$





❸ Training by Gradient Descent

(4) Example



(5) Logistic regression algorithm

```
initialize weights at time step t = 0 to w(0)
for t = 0, 1, 2, ... do
    compute the gradient

        ∇E_in(w(t)) = -(1/N) Σ_{n=1}^{N} y_n x_n / (1 + e^{y_n w^⊤(t) x_n})

    set the direction to move: v_t = -∇E_in(w(t))
    update weights: w(t+1) = w(t) + η v_t
    iterate to next step until it is time to stop
return final weights w
```

$$\nabla \mathrm{E}_{\mathrm{in}}(\mathbf{w}(t)) = -\frac{1}{N}\sum_{n=1}^{N}\frac{y_n\mathbf{x}_n}{1+e^{y_n\mathbf{w}^\top(t)\mathbf{x}_n}}$$

❸ Training by Gradient Descent

(6) Termination criteria

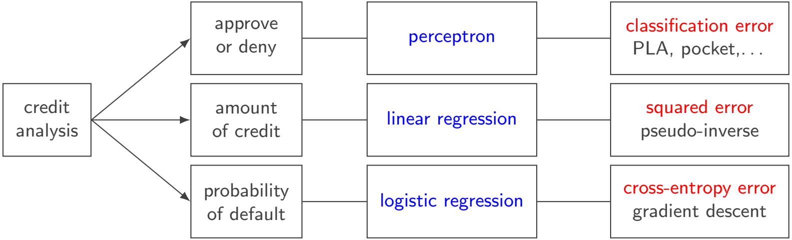
- **Combination works reasonably well**
  - maximum number of iterations
  - marginal error improvement
  - small value for the error itself

(7) Variants of gradient descent

- **Batch gradient descent**
  - use all N examples in each iteration
- **Stochastic gradient descent**
  - use 1 example in each iteration
- **Mini-batch gradient descent**
  - use b examples in each iteration
  - b: mini-batch size (typically $2 \sim 100$)

Summary

- **Three linear models considered**



- **Respective goals, error measures, and algorithms**
  - nonetheless, share similar sets of linear hypotheses
- **You should first try a linear model**
  - simplicity, generalization, extension

Summary

	linear classification	linear regression	logistic regression
$\mathcal{Y}$	$\{-1,+1\}$	$\mathbb{R}$	$\{-1,+1\}$
$\hat{y}=h(\mathbf{x})$	$\mathrm{sign}(\mathbf{w}^\top\mathbf{x})$	$\mathbf{w}^\top\mathbf{x}$	$\theta(\mathbf{w}^\top\mathbf{x})$
$\mathrm{e}(\hat{y},y)$	0-1 loss $[\![\hat{y}\neq y]\!]$	squared error $(\hat{y}-y)^2$	cross-entropy error $[\![y{=}+1]\!]\ln\frac{1}{\hat{y}}$ $+[\![y{=}-1]\!]\ln\frac{1}{1-\hat{y}}$
$\mathrm{E}_{\mathrm{in}}(h)$	$\frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n){\neq}y_n]\!]$	$\frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n){-}y_n)^2$	$\frac{1}{N}\sum_{n=1}^{N}\ln\left(1{+}e^{-y_n\mathbf{w}^\top\mathbf{x}_n}\right)$
opt.	combinatorial optimization (NP-hard)	set $\nabla\mathrm{E}_{\mathrm{in}}(\mathbf{w})=0$ (closed-form solution exists)	set $\nabla\mathrm{E}_{\mathrm{in}}(\mathbf{w})=0$ iterative optimization (*e.g.* gradient descent)

⋆ logistic sigmoid $\theta(s)=1/(1+e^{-s})$