Machine Learning
(機器學習)

Lecture 5: Linear Models

Hsuan-Tien Lin (林軒田)
htlin@csie.ntu.edu.tw

Department of Computer Science
& Information Engineering

National Taiwan University
(國立台灣大學資訊工程系)

Roadmap

1. When Can Machines Learn?
2. Why Can Machines Learn?
3. **How** Can Machines Learn?

Lecture 5: Linear Models

- Linear Regression Problem
- Linear Regression Algorithm

excel 的直线迴歸

- Logistic Regression Problem
- Logistic Regression Error
- Gradient of Logistic Regression Error
- Gradient Descent
- Stochastic Gradient Descent

Credit **Limit** Problem

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

credit limit? **100,000**

unknown target function
$f : \mathcal{X} \to \mathcal{Y}$

*(ideal credit **limit** formula)*

training examples
$\mathcal{D} : (\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$

(historical records in bank)

learning algorithm
$\mathcal{A}$

final hypothesis
$g \approx f$

('learned' formula to be used)

hypothesis set
$\mathcal{H}$

(set of candidate formula)

不再是 Binary

$\mathcal{Y} = \mathbb{R}$: **regression**

Linear Regression Hypothesis

age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

custom's data

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of customer', approximate the desired credit limit with a weighted sum:

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

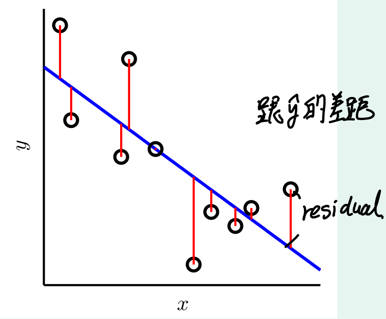
也是加權總分

- linear regression hypothesis: $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ 得到 regression

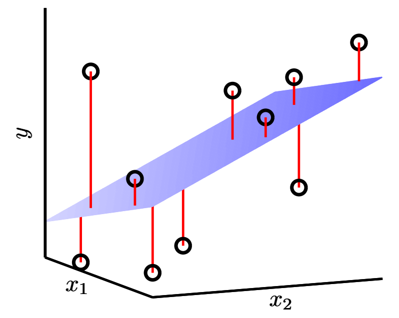
$h(\mathbf{x})$: like **perceptron**, but without the sign

Illustration of Linear Regression

$\mathbf{x} = (x) \in \mathbb{R}$ 2D



$\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ 3D



linear regression:
find lines/hyperplanes with small residuals

Pointwise Error Measure for 'Small Residuals'

final hypothesis
$g \approx f$

how well? often use averaged $\mathrm{err}(g(\mathbf{x}), f(\mathbf{x}))$, like

$$E_{\text{out}}(g) = \underset{\mathbf{x}\sim P}{\mathcal{E}} \underbrace{[\![g(\mathbf{x}) \neq f(\mathbf{x})]\!]}_{\mathrm{err}(g(\mathbf{x}), f(\mathbf{x}))}$$

← classification

← generalization

—err: called **pointwise error measure**

in-sample

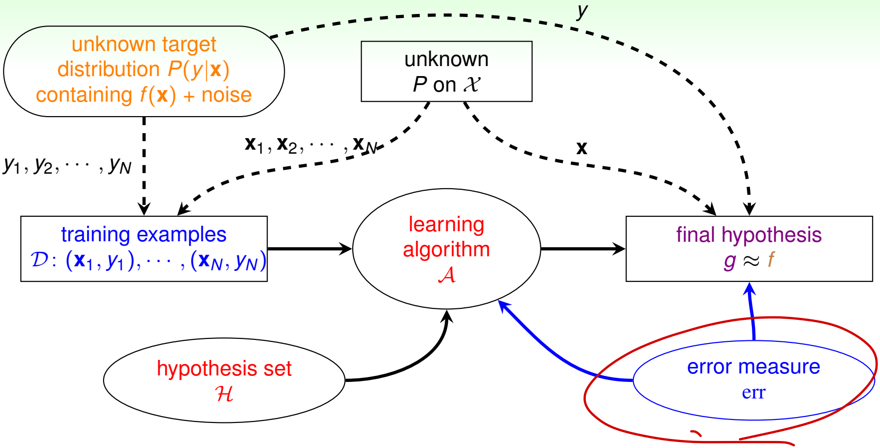
$$E_{\text{in}}(g) = \frac{1}{N}\sum_{n=1}^{N} \mathrm{err}(g(\mathbf{x}_n), f(\mathbf{x}_n))$$

out-of-sample

$$E_{\text{out}}(g) = \underset{\mathbf{x}\sim P}{\mathcal{E}} \mathrm{err}(g(\mathbf{x}), f(\mathbf{x}))$$

will mainly consider pointwise err for simplicity

Learning Flow with Pointwise Error Measure



extended VC theory/'philosophy'

works for most $\mathcal{H}$ and err

Two Important Pointwise Error Measures

$$\text{err}\left(\underbrace{g(\mathbf{x})}_{\tilde{y}}, \underbrace{f(\mathbf{x})}_{y}\right)$$

MSE

0/1 error

$$\text{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$$

- correct or incorrect?
- often for classification

squared error

$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

- how far is $\tilde{y}$ from y?
- often for regression

squared error: quantify '**small residual**'

Squared Error Measure for Regression

popular/historical error measure for linear regression:

squared error $\mathrm{err}(\hat{y}, y) = (\hat{y} - y)^2$

in-sample

prediction label

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\boxed{\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n}} - \boxed{y_n})^2$$

MSE

out-of-sample

$$E_{\text{out}}(\mathbf{w}) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}}(\mathbf{w}^T\mathbf{x} - y)^2$$

next: how to minimize $E_{\text{in}}(\mathbf{w})$?

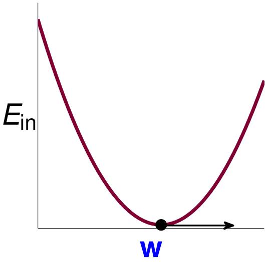
Questions?

Matrix Form of $E_{\text{in}}(\mathbf{w})$

$$
\begin{aligned}
E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - y_n)^2 \\
&= \frac{1}{N}\left\| \begin{matrix} \mathbf{x}_1^T\mathbf{w} - y_1 \\ \mathbf{x}_2^T\mathbf{w} - y_2 \\ \dots \\ \mathbf{x}_N^T\mathbf{w} - y_N \end{matrix} \right\|^2 \\
&= \frac{1}{N}\left\| \overbrace{\begin{bmatrix} -\,-\,\mathbf{x}_1^T\,-\,- \\ -\,-\,\mathbf{x}_2^T\,-\,- \\ \dots \\ -\,-\,\mathbf{x}_N^T\,-\,- \end{bmatrix}}^{d+1} \mathbf{w} - \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} \right\|^2 \\
&= \frac{1}{N}\| \underbrace{\mathrm{X}}_{N\times d+1} \underbrace{\mathbf{w}}_{d+1\times 1} - \underbrace{\mathbf{y}}_{N\times 1} \|^2
\end{aligned}
$$

內積互換

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$



凸函數

- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of 'best' $\mathbf{w}$

各w方向上

$$\nabla E_{\text{in}}(\mathbf{w}) \equiv \begin{bmatrix} \frac{\partial E_{\text{in}}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E_{\text{in}}}{\partial w_1}(\mathbf{w}) \\ \dots \\ \frac{\partial E_{\text{in}}}{\partial w_d}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

—not possible to 'roll down' 最低點

無法再往下

task: find $\mathbf{w}_{\text{LIN}}$ such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \mathbf{0}$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

Loss function

$$E_{\text{in}}(\mathbf{w}) = \underbrace{\frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2}_{} = \frac{1}{N}\left(\mathbf{w}^T \underbrace{\mathrm{X}^T\mathrm{X}}_{\mathrm{A}} \mathbf{w} - 2\mathbf{w}^T \underbrace{\mathrm{X}^T\mathbf{y}}_{\mathbf{b}} + \underbrace{\mathbf{y}^T\mathbf{y}}_{c}\right)$$

simple example

one w only

$$E_{\text{in}}(w) = \frac{1}{N}\left(aw^2 - 2bw + c\right)$$

$$\nabla E_{\text{in}}(w) = \frac{1}{N}\left(2aw - 2b\right)$$

simple! :-)

vector $\mathbf{w}$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(\mathbf{w}^T\mathrm{A}\mathbf{w} - 2\mathbf{w}^T\mathbf{b} + c\right)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(2\mathrm{A}\mathbf{w} - 2\mathbf{b}\right)$$

similar (**derived by definition**)

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\left(\mathrm{X}^T\mathrm{X}\mathbf{w} - \mathrm{X}^T\mathbf{y}\right) = 0$$

Optimal Linear Regression Weights

task: find $\mathbf{w}_{\text{LIN}}$ such that $\frac{2}{N}\left(\mathrm{X}^T\mathrm{X}\mathbf{w} - \mathrm{X}^T\mathbf{y}\right) = \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

invertible $\mathrm{X}^T\mathrm{X}$

- **easy!** unique solution

$$\boxed{\mathbf{w}_{\text{LIN}} = \underbrace{\left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T}_{\text{pseudo-inverse } \mathrm{X}^\dagger}\mathbf{y}}$$

- often the case because $N \gg d+1$

singular $\mathrm{X}^T\mathrm{X}$

- **many** optimal solutions
- one of the solutions

$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^\dagger\mathbf{y}$$

by defining $\mathrm{X}^\dagger$ in other ways

practical suggestion: 直接使用library的pesudo inverse

use **well-implemented $\dagger$ routine**

instead of $\left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T$

for numerical stability when **almost-singular**

Linear Regression Algorithm

1. from $\mathcal{D}$, construct input matrix X and output vector $\mathbf{y}$ by

$$\mathrm{X} = \underbrace{\begin{bmatrix} -\,-\,\mathbf{x}_1^T\,-\,- \\ -\,-\,\mathbf{x}_2^T\,-\,- \\ \cdots \\ -\,-\,\mathbf{x}_N^T\,-\,- \end{bmatrix}}_{N\times(d+1)} \quad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}}_{N\times 1}$$

2. calculate pseudo-inverse $\underbrace{\mathrm{X}^\dagger}_{(d+1)\times N}$

update weight

3. return $\underbrace{\mathbf{w}_{\mathrm{LIN}}}_{(d+1)\times 1} = \mathrm{X}^\dagger \mathbf{y}$

一步到位

simple and efficient
with **good † routine**

Is Linear Regression a 'Learning Algorithm'?

$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^{\dagger}\mathbf{y}$$

No!

- analytic (**closed-form**) solution, 'instantaneous'
- not improving E_{in} nor E_{out} iteratively

Yes!

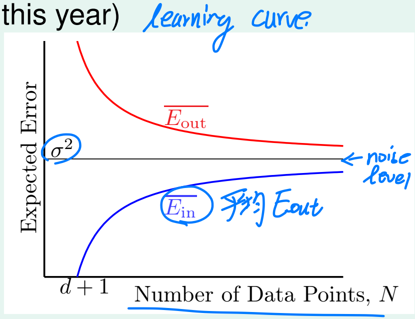
- good E_{in}? **yes, optimal!**
- good E_{out}? **yes, finite d_{VC} like perceptrons**
- improving iteratively? **somewhat, within an iterative pseudo-inverse routine**

if $E_{\text{out}}(\mathbf{w}_{\text{LIN}})$ is good, **learning 'happened'**!

The Learning Curves of Linear Regression

(proof skipped this year)

$$\overline{E_{\text{out}}} = \textbf{noise level} \cdot \left(1 + \frac{d+1}{N}\right)$$
$$\overline{E_{\text{in}}} = \textbf{noise level} \cdot \left(1 - \frac{d+1}{N}\right)$$



- both converge to σ^2 (**noise** level) for $N \to \infty$
- expected generalization error: $\frac{2(d+1)}{N}$
  —**similar to worst-case guarantee from VC**

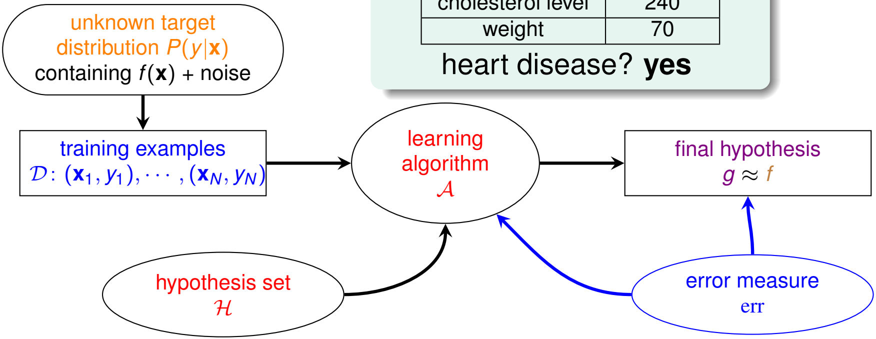
linear regression (LinReg):
learning 'happened'!

Questions?

Heart Attack Prediction Problem (1/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart disease? **yes**



binary classification:

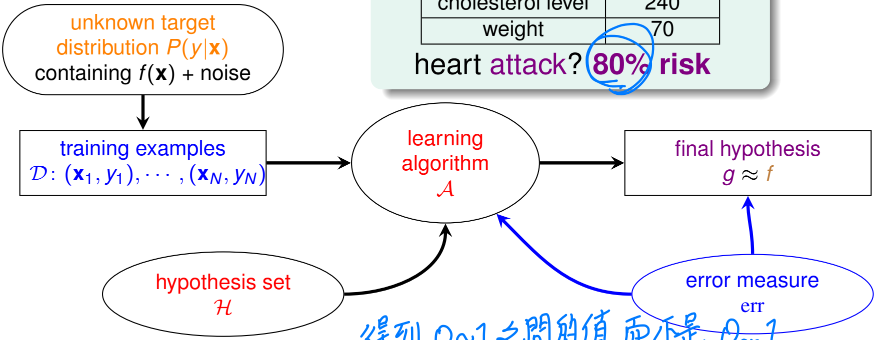
$$\text{ideal } f(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right) \in \{-1, +1\}$$

because of classification err

Heart Attack Prediction Problem (2/2)

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240
weight	70

heart attack? **80% risk**



得到 0~1 之間的值，而不是 0 or 1

'soft' binary classification:

$$f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0, 1]$$

probability

Soft Binary Classification

target function $f(\mathbf{x}) = \underline{P(+1|\mathbf{x})} \in [0, 1]$

又為真的机率

ideal (noiseless) data

$$\begin{pmatrix}\mathbf{x}_1, y'_1 & = 0.9 & = P(+1|\mathbf{x}_1)\end{pmatrix}$$
$$\begin{pmatrix}\mathbf{x}_2, y'_2 & = 0.2 & = P(+1|\mathbf{x}_2)\end{pmatrix}$$
$$\vdots$$
$$\begin{pmatrix}\mathbf{x}_N, y'_N & = 0.6 & = P(+1|\mathbf{x}_N)\end{pmatrix}$$

actual (noisy) data

$$\begin{pmatrix}\mathbf{x}_1, y_1 & = \circ & \sim P(y|\mathbf{x}_1)\end{pmatrix}$$
$$\begin{pmatrix}\mathbf{x}_2, y_2 & = \times & \sim P(y|\mathbf{x}_2)\end{pmatrix}$$
$$\vdots$$
$$\begin{pmatrix}\mathbf{x}_N, y_N & = \times & \sim P(y|\mathbf{x}_N)\end{pmatrix}$$

same data as hard binary classification,
different **target function**

Soft Binary Classification

target function $f(\mathbf{x}) = P(+1|\mathbf{x}) \in [0,1]$

ideal (noiseless) data

$$\begin{aligned}&\left(\mathbf{x}_1, y'_1 = 0.9 = P(+1|\mathbf{x}_1)\right)\\&\left(\mathbf{x}_2, y'_2 = 0.2 = P(+1|\mathbf{x}_2)\right)\\&\quad\vdots\\&\left(\mathbf{x}_N, y'_N = 0.6 = P(+1|\mathbf{x}_N)\right)\end{aligned}$$

抽樣

這是我們最想要的版本

actual (noisy) data

$$\begin{aligned}&\left(\mathbf{x}_1, y'_1 = 1 = \left[\!\left[\circ \overset{?}{\sim} P(y|\mathbf{x}_1)\right]\!\right]\right)\\&\left(\mathbf{x}_2, y'_2 = 0 = \left[\!\left[\circ \overset{?}{\sim} P(y|\mathbf{x}_2)\right]\!\right]\right)\\&\quad\vdots\\&\left(\mathbf{x}_N, y'_N = 0 = \left[\!\left[\circ \overset{?}{\sim} P(y|\mathbf{x}_N)\right]\!\right]\right)\end{aligned}$$

抽到0

手上有的data.

same data as hard binary classification,
different **target function**

Logistic Hypothesis

age	40 years
gender	male
blood pressure	130/85
cholesterol level	240

常數

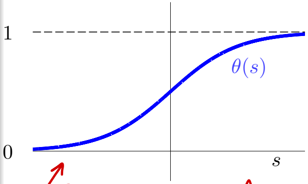
- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of patient', calculate a weighted 'risk score':

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

跟 linear regression 一樣

- convert the score to estimated probability by logistic function $\theta(s)$ ←得到 0~1 之間的值

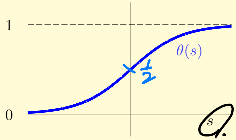
Logistic function



透過 activation function

logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

Logistic Function



$\theta(-\infty) = 0$; $\theta(0) = \frac{1}{2}$; $\theta(\infty) = 1$

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

$\alpha(S)$ $w^T x - w^T x$

$f(x) = \frac{1}{a\sqrt{2\pi}}$

—smooth, monotonic, $\boxed{\text{sigmoid}}$ function of s

S型 function

logistic regression: use

use logistic function

$S = w^T x$

$$h(\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

to approximate target function $f(\mathbf{x}) = P(+1|\mathbf{x})$

Questions?

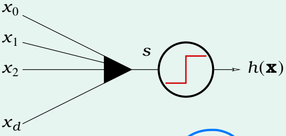
Three Linear Models

linear scoring function: $s = \mathbf{w}^T\mathbf{x}$

activation function 的不同.

linear classification

$h(\mathbf{x}) = \text{sign}(s)$

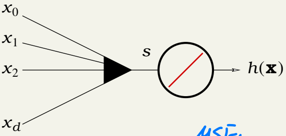


plausible err = 0/1

(small flipping noise)

linear regression

$h(\mathbf{x}) = s$



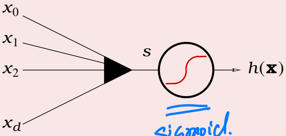
MSE.

friendly err = squared

(easy to minimize)

logistic regression

$h(\mathbf{x}) = \theta(s)$



sigmoid.

err = ?

how to define

$E_{\text{in}}(\mathbf{w})$ for logistic regression?

Likelihood

$P(+1|x)$

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

$P(-1|x)$

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$ ← 這個資料產生的機率

probability that f generates $\mathcal{D}$

有 x_1 然後 x_1是o

$P(\mathbf{x}_1)P(\circ|\mathbf{x}_1) \times$

$P(\mathbf{x}_2)P(\times|\mathbf{x}_2) \times$

$\ldots$

$P(\mathbf{x}_N)P(\times|\mathbf{x}_N)$

$\Longrightarrow$

likelihood that h generates $\mathcal{D}$

$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$

$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$

$\ldots$

$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$

- if $h \approx f$,
  then likelihood(h) $\approx$ probability using f
- probability using f usually **large**

Likelihood

實數域

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

consider $\mathcal{D} = \{(\mathbf{x}_1, \circ), (\mathbf{x}_2, \times), \ldots, (\mathbf{x}_N, \times)\}$

h也產生一樣的data.

probability that f generates $\mathcal{D}$

$P(\mathbf{x}_1) f(\mathbf{x}_1) \times$ 代入

$P(\mathbf{x}_2)(1 - f(\mathbf{x}_2)) \times$

$\ldots$

$P(\mathbf{x}_N)(1 - f(\mathbf{x}_N))$

likelihood that h generates $\mathcal{D}$

用h假佯f

$P(\mathbf{x}_1) h(\mathbf{x}_1) \times$

$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$

$\ldots$

$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$

估計f產生該筆資料的可能性

- if $h \approx f$, then likelihood(h) $\approx$ probability using f

  h.f兩個函數產生資料的機率都很接近

- probability using f usually **large**

因為已經是用f抽樣出來的資料

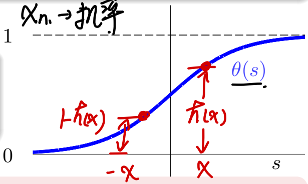
Likelihood of Logistic Hypothesis

$$\text{likelihood}(h) \approx (\text{probability using } f) \approx \textbf{large}$$

$$g = \underset{h}{\operatorname{argmax}} \ \text{likelihood}(h)$$ 所有h之中,找可能性最高的h

$P(+1|x) = ?$ $P(x_n)$

$x_n \rightarrow$ 机率



when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$ $P(y_n|x_n) = ?$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$ 对称性

$1-h(x) = h(-x)$

对所有h,皆相同

$$\text{likelihood}(h) = P(\mathbf{x}_1)h(\mathbf{x}_1) \times P(\mathbf{x}_2)(1 - \overbrace{h(\mathbf{x}_2)}^{h(x_2)})) \times \ldots P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

$y = 0.38$

$\Downarrow h(-x_2)$

都相同,可省

$\text{likelihood}(\hat{\mu}) = \mu^{\alpha} \cdot (1-\mu)^{(N-\alpha)}$ $\alpha = \#\text{head}$

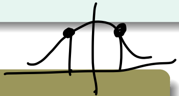
$\Rightarrow$

$$\text{likelihood}(\text{logistic } h) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

Likelihood of Logistic Hypothesis

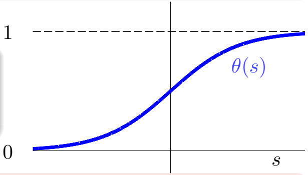
likelihood(h) $\approx$ (probability using f) $\approx$ **large**

$$g = \operatorname*{argmax}_{h} \ \text{likelihood}(h)$$



when logistic: $h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$

$$1 - h(\mathbf{x}) = h(-\mathbf{x})$$



$h(x) = h(-x)$

$$\text{likelihood}(h) = P(\mathbf{x}_1)h(+\mathbf{x}_1) \times P(\mathbf{x}_2)h(-\mathbf{x}_2) \times \ldots P(\mathbf{x}_N)h(-\mathbf{x}_N)$$

likelihood 全部相乘

$$\text{likelihood(logistic } h) \propto \prod_{n=1}^{N} h(y_n\mathbf{x}_n)$$

0..X

Cross-Entropy Error

$$\max_{\cancel{h}\; w} \quad \text{likelihood}(\text{logistic } \cancel{h}\; \underline{\underline{w}}) \propto \prod_{n=1}^{N} \cancel{h}\; w(\underbrace{y_n}_{=\,(2y_n'-1)}\mathbf{x}_n) \qquad (2y_n'-1)\vec{x}_n$$

找一組weight，使得 likelihood 最大

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \text{likelihood}(\mathbf{w}) \propto \prod_{n=1}^{N} \theta\left(\underbrace{y_n}_{2y_n'-1} \mathbf{w}^T \mathbf{x}_n\right)$$

取ln.

Cross-Entropy Error

$$\max_{\mathbf{w}} \ \underbrace{\ln \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)}_{\text{變相加}}$$

作min. instead of max (loss)

$$\max_{w} \ln \prod_{n=1} \theta\big(\underline{(2y'-1)w^T x_n}\big)$$

$2y'w^T x_n - w^T x_n$

假設 $\theta(a+b) = \theta(a) + \theta(b)$

$-\sum \ln\left[\theta(2y'w^T x_n) - \theta(w^T x_n)\right]$

Cross-Entropy Error

$e^1 \cdot e^2$

$$\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} - \ln \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

$e^{a+b} = e^a \cdot e^b$

$\theta(a) = \frac{1}{1+\frac{1}{e^a}} = \frac{e^a}{e^a+1} \cdot \frac{e^b}{e^b+1} \qquad \frac{e^{a+b}}{e^{a+b}+e^a+e^b+1}$

$$\theta(s) = \frac{1}{1+\exp(-s)} \quad : \quad \min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

$\theta(a+b) = \frac{1}{1+\exp(-a-b)}$

$$\Longrightarrow \min_{\mathbf{w}} \underbrace{\frac{1}{N} \sum_{n=1}^{N} \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

$= \frac{1}{1+e^{-(a+b)}} = \frac{1}{1+\frac{1}{e^{a+b}}}$

$\frac{e^{a+b}}{e^{a+b}+1} \qquad \frac{e^a \cdot e^b}{e^a \cdot e^b + 1}$

$$\mathrm{err}(\mathbf{w}, \mathbf{x}, y) = \ln\left(1+\exp(-y\mathbf{w}^T\mathbf{x})\right):$$

cross-entropy error

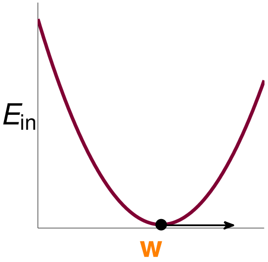
Questions?

Minimizing $E_{\text{in}}(\mathbf{w})$

cross-entropy 的平均

$$\min_{\mathbf{w}} \quad E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \boxed{\ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)}$$

还是 convex



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, twice-differentiable, **convex**
- how to minimize? locate **valley**

want $\nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

first: derive $\nabla E_{\text{in}}(\mathbf{w})$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\underbrace{1+\exp(\overbrace{-y_n\mathbf{w}^T\mathbf{x}_n}^{\bigcirc})}_{\square}\right)$$

找微分為零的方向

$$\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial - y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\qquad\right)\left(\qquad\right)\left(\qquad\right)$$

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)\left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(\bigcirc)\left(-y_n x_{n,i}\right)$$

$$\frac{1}{1+}$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)$$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(\underbrace{1+\exp(\overbrace{-y_n\mathbf{w}^T\mathbf{x}_n}^{\bigcirc})}_{\square}\right)$$

$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial -y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right)\\
&= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{\square}\right)\left(\exp(\bigcirc)\right)\left(-y_n x_{n,i}\right)\\
&= \frac{1}{N}\sum_{n=1}^{N}\boxed{\left(\frac{\exp(\bigcirc)}{1+\exp(\bigcirc)}\right)}\left(-y_n x_{n,i}\right) = \frac{1}{N}\sum_{n=1}^{N}\theta(\bigcirc)\left(-y_n x_{n,i}\right)
\end{aligned}$$

sigmoid

sigmoid

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\theta\left(-y_n\mathbf{w}^T\mathbf{x}_n\right)\left(-y_n\mathbf{x}_n\right)$$

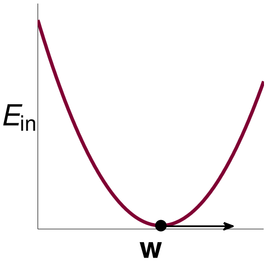
梯度

Minimizing $E_{in}(\mathbf{w})$

$$\min_{\mathbf{w}} \quad E_{in}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1 + \exp(-y_n \mathbf{w}^T \mathbf{x}_n)\right)$$

$$\text{want } \nabla E_{in}(\mathbf{w}) = \boxed{\frac{1}{N}\sum_{n=1}^{N} \theta\left(\underline{-y_n \mathbf{w}^T \mathbf{x}_n}\right)\left(-y_n \mathbf{x}_n\right)} = \mathbf{0}$$

使之為零!?



scaled θ-weighted sum of $-y_n\mathbf{x}_n$

$\theta(-\infty)=0$

- all $\underline{\theta(\cdot)}$ = 0: only if $\underline{y_n \mathbf{w}^T \mathbf{x}_n} \gg 0$ —linear separable $\mathcal{D}$ ← 每個 prediction都正確.
- weighted sum = $\mathbf{0}$: non-linear equation of $\mathbf{w}$

closed-form solution? no :-(

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\underbrace{\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}}_{\text{mistake}}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)} \mathbf{x}_{n(t)}$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. find a mistake of $\mathbf{w}_t$ called $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

2. (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

update.

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{\left[\!\left[\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n\right]\!\right]}_{\text{prediction 正確 or 錯誤}} y_n \mathbf{x}_n$$

when stop, return last $\mathbf{w}$ as g

PLA Revisited: Iterative Optimization

PLA: start from some $\mathbf{w}_0$ (say, $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

For $t = 0, 1, \ldots$

1. (equivalently) pick some n, and update $\mathbf{w}_t$ by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \underbrace{1}_{\eta\ 更新率} \cdot \underbrace{\left([\![\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_n\right) \neq y_n]\!] \cdot y_n \mathbf{x}_n \right)}_{\mathbf{v}\ 更新方向}$$

when stop, return last $\mathbf{w}$ as g

choice of $(\eta, \mathbf{v})$ and stopping condition defines **iterative optimization approach**

Questions?

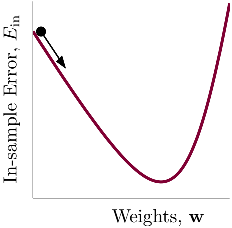
Iterative Optimization

For $t = 0, 1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

- PLA: $\mathbf{v}$ comes from mistake correction
- smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll '**downhill**'?
    - direction $\mathbf{v}$: (assumed) of unit length
    - step size η: (assumed) positive



a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}})$$

Linear Approximation

a greedy approach for some given $\eta > 0$:

$$\min_{\|\mathbf{v}\|=1} \quad E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v})$$

V is unit vector

- still non-linear optimization, now **with constraints**
  —not any easier than $\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w})$
- local approximation by linear formula makes problem easier

  linearlization. 在w_t上微分

$$E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx E_{\text{in}}(\mathbf{w}_t) + \eta\mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)$$

  if η really small (Taylor expansion) ↓線性化

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \quad \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

Gradient Descent

an **approximate** greedy approach for some given **small** η:

$$\min_{\|\mathbf{v}\|=1} \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

V^T与梯度完全反向，最负.

- optimal $\mathbf{v}$: opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

单位向量，且是负的

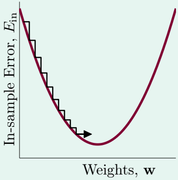
- gradient descent: for **small** η, $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$

往梯度的反方向走

gradient descent:
a simple & popular optimization tool

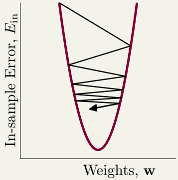
Choice of η learning rate

too small



too slow :-(

too large



too unstable :-(

a naive yet effective heuristic

- if red $\eta \propto \|\nabla E_{\text{in}}(\mathbf{w}_t)\|$ by ratio purple η (the fixed learning rate)

坡度大,就走遠一点

不做 normalize

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|} = \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

fixed learning rate gradient descent:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

Putting Everything Together

Logistic Regression Algorithm

initialize $\mathbf{w}_0$

For $t = 0, 1, \cdots$

1. compute

cross entropy 的推導

sigmoid

$$\nabla E_{\text{in}}(\mathbf{w}_t) = \frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(-y_n \mathbf{x}_n\right)$$

2. update by

update.

$$\underline{\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)}$$

...until $\boxed{\nabla E_{\text{in}}(\mathbf{w}_{t+1}) \cong 0}$ or <u>enough iterations</u> maximum Iteration

return last $\mathbf{w}_{t+1}$ as g

$O(N)$ time complexity in step 1 per iteration

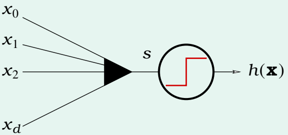
Questions?

Linear Models Revisited

linear scoring function: $s = \mathbf{w}^T\mathbf{x}$

linear classification

$h(\mathbf{x}) = \text{sign}(s)$

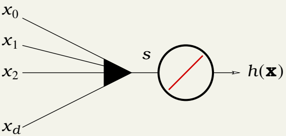


plausible err = 0/1

discrete $E_{\text{in}}(\mathbf{w})$:
NP-hard to solve in general

linear regression

$h(\mathbf{x}) = s$

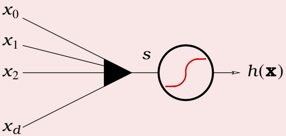


friendly err = squared

quadratic convex $E_{\text{in}}(\mathbf{w})$:
closed-form solution

logistic regression

$h(\mathbf{x}) = \theta(s)$



plausible err = cross-entropy

smooth convex $E_{\text{in}}(\mathbf{w})$:
gradient descent

can linear regression or logistic regression
help linear classification?

Error Functions Revisited

linear scoring function: $s = \mathbf{w}^T\mathbf{x}$ 打分數

for binary classification $y \in \{-1, +1\}$

linear classification

$$h(\mathbf{x}) = \text{sign}(s)$$
$$\text{err}(h, \mathbf{x}, y) = [\![h(\mathbf{x}) \neq y]\!]$$

score label

error $= \text{err}_{0/1}(s, y)$

$$= [\![\text{sign}(s) \neq y]\!]$$ 1
$$= [\![\text{sign}(ys) \neq 1]\!]$$

linear regression

$$h(\mathbf{x}) = s$$
$$\text{err}(h, \mathbf{x}, y) = (h(\mathbf{x}) - y)^2$$

$$\text{err}_{\text{SQR}}(s, y)$$
$$= (s - y)^2$$ (因為 $y = \pm 1$)

$y^2 = 1$

$$= (ys - 1)^2$$

logistic regression

$$h(\mathbf{x}) = \theta(s)$$
$$\text{err}(h, \mathbf{x}, y) = -\ln h(y\mathbf{x})$$

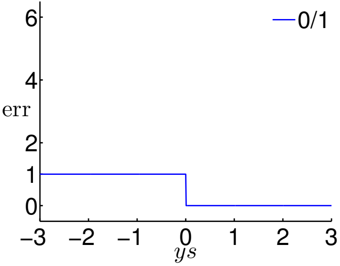
$$\text{err}_{\text{CE}}(s, y)$$
$$= \ln(1 + \exp(-ys))$$

正確度分數

(ys): classification correctness score 希望是正

Visualizing Error Functions

$$
\begin{aligned}
\text{0/1} \quad & \mathrm{err}_{0/1}(s,y) &=& \; [\![\operatorname{sign}(ys) \neq 1]\!] \\
\text{sqr} \quad & \mathrm{err}_{\text{SQR}}(s,y) &=& \; (ys-1)^2 \\
\text{ce} \quad & \mathrm{err}_{\text{CE}}(s,y) &=& \; \ln(1+\exp(-ys)) \\
\text{scaled ce} \quad & \mathrm{err}_{\text{SCE}}(s,y) &=& \; \log_2(1+\exp(-ys))
\end{aligned}
$$

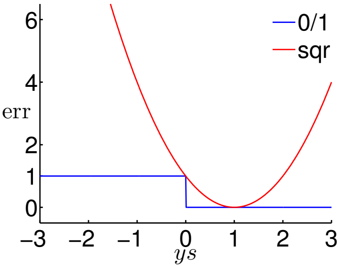


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\text{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\text{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\mathrm{err}_{\text{SCE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$

upper bound:
useful for designing algorithm

Visualizing Error Functions

$$
\begin{aligned}
\text{0/1} \quad & \mathrm{err}_{0/1}(s, y) &=& [\![\mathrm{sign}(ys) \neq 1]\!] \\
\text{sqr} \quad & \mathrm{err}_{\mathrm{SQR}}(s, y) &=& (ys - 1)^2 \\
\text{ce} \quad & \mathrm{err}_{\mathrm{CE}}(s, y) &=& \ln(1 + \exp(-ys)) \\
\text{scaled ce} \quad & \mathrm{err}_{\mathrm{SCE}}(s, y) &=& \log_2(1 + \exp(-ys))
\end{aligned}
$$



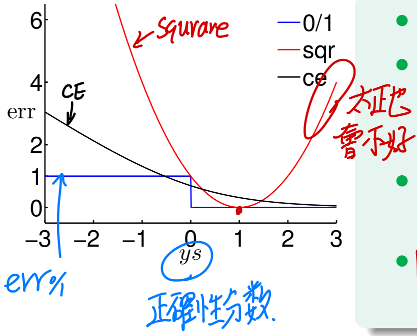
- 0/1: 1 iff $ys \le 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\mathrm{err}_{\mathrm{SCE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$

upper bound:
useful for designing algorithm

Visualizing Error Functions

0/1	$\text{err}_{0/1}(s,y)$	$=$	$[\![\text{sign}(ys) \neq 1]\!]$
sqr	$\text{err}_{\text{SQR}}(s,y)$	$=$	$(ys-1)^2$
ce	$\text{err}_{\text{CE}}(s,y)$	$=$	$\ln(1+\exp(-ys))$
scaled ce	$\text{err}_{\text{SCE}}(s,y)$	$=$	$\log_2(1+\exp(-ys))$

只是乘上一个倍数

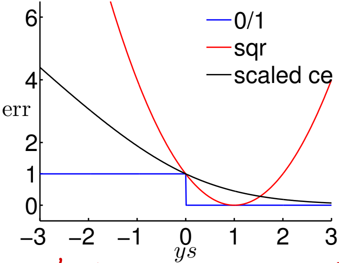


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
  small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- $\boxed{\text{scaled ce}}$: a proper upper bound of 0/1
  small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

upper bound:
useful for designing algorithm

Visualizing Error Functions

0/1	$\mathrm{err}_{0/1}(s, y)$	$=$	$[\![\mathrm{sign}(ys) \neq 1]\!]$
sqr	$\mathrm{err}_{\mathrm{SQR}}(s, y)$	$=$	$(ys - 1)^2$
ce	$\mathrm{err}_{\mathrm{CE}}(s, y)$	$=$	$\ln(1 + \exp(-ys))$
scaled ce	$\mathrm{err}_{\mathrm{SCE}}(s, y)$	$=$	$\log_2(1 + \exp(-ys))$

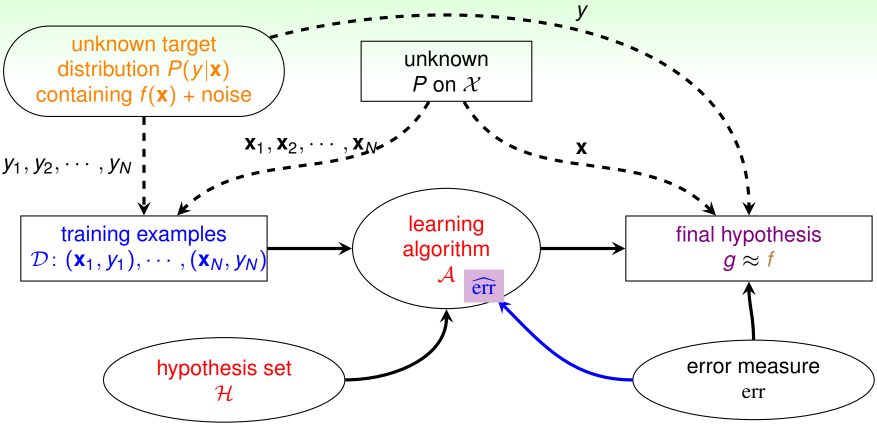


刻意讓CE切在(0,1)的
位置.

- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\mathrm{err}_{\mathrm{SCE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$

upper bound:
useful for designing algorithm

Learning Flow with Algorithmic Error Measure



err: goal, not always easy to optimize;
$\widehat{\text{err}}$: something 'similar' to facilitate $\mathcal{A}$, e.g. upper bound

Theoretical Implication of Upper Bound

For any ys where $s = \mathbf{w}^T\mathbf{x}$

scale.

$$\text{err}_{0/1}(s, y) \le \text{err}_{\text{SCE}}(s, y) = \frac{1}{\ln 2}\text{err}_{\text{CE}}(s, y).$$

$$\Longrightarrow \quad E_{\text{in}}^{0/1}(\mathbf{w}) \le E_{\text{in}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})$$

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le \boxed{E_{\text{out}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})}$$

VC on $0/1$:

mode. complexity

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le E_{\text{in}}^{0/1}(\mathbf{w}) + \Omega^{0/1}$$
$$\le \boxed{\frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})} + \Omega^{0/1}$$

VC-Reg on CE :

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})$$
$$\le \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \frac{1}{\ln 2}\Omega^{\text{CE}}$$

代進 upper bound.

small $E_{\text{in}}^{\text{CE}}(\mathbf{w}) \Longrightarrow$ small $E_{\text{out}}^{0/1}(\mathbf{w})$:
logistic/linear reg. for linear classification

做好 CE 就是做好 $0/1$

Regression for Classification

1. run **logistic**/**linear** **reg.** on $\mathcal{D}$ with $y_n \in \{-1, +1\}$ to get $\mathbf{w}_{\text{REG}}$
2. return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{REG}}^T \mathbf{x})$ 算完分数再取sign.

PLA

- pros: **efficient + strong guarantee if lin. separable**
- cons: works only if lin. separable

导需线性可分

linear regression

- pros: 最easy **'easiest' optimization**
- cons: loose bound of $\text{err}_{0/1}$ for large $|ys|$

ys大的时候,与err0/1差比较多

logistic regression

- pros: **'easy' optimization**
- cons: loose bound of $\text{err}_{0/1}$ for very negative ys

ys小时,

用linear regression算出W_0

- **linear regression** sometimes used to set $\mathbf{w}_0$ for **PLA**/**logistic regression**
- **logistic regression** often preferred in practice

Questions?

Two Iterative Optimization Schemes

For $t = 0, 1, \ldots$

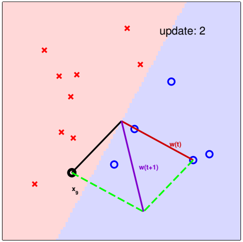
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

PLA

pick $(\mathbf{x}_n, y_n)$ and decide $\mathbf{w}_{t+1}$ by **the one example**

$O(1)$ time per iteration :-)



logistic regression

check $\mathcal{D}$ and decide $\mathbf{w}_{t+1}$ (or new $\hat{\mathbf{w}}$) by **all examples**

$\boxed{O(N) \text{ time per iteration :-(}}$

每次更新都要看过所有data.

logistic regression with **$O(1)$ time per iteration?**

Logistic Regression Revisited

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla E_{\text{in}}(\mathbf{w}_t)}$$

- want: update direction $\mathbf{v} \approx -\nabla E_{\text{in}}(\mathbf{w}_t)$
  while computing $\mathbf{v}$ by one single $(\mathbf{x}_n, y_n)$
- technique on removing $\frac{1}{N}\sum_{n=1}^{N}$:   隨机抽一个点
  view as expectation $\mathcal{E}$ over uniform choice of n!

stochastic gradient: 隨機梯度
$\nabla_{\mathbf{w}}\ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$ with random n
true gradient: 就是Ein
$$\nabla_{\mathbf{w}}\ E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}}\ \nabla_{\mathbf{w}}\ \text{err}(\mathbf{w}, \mathbf{x}_n, y_n)$$

Stochastic Gradient Descent (SGD)

stochastic gradient = true gradient + zero-mean 'noise' directions

隨機性

亂走的部份

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps, ←跑夠多步的話,結果接近.

  average true gradient ≈ average stochastic gradient
- pros: **simple & cheaper computation :-)**

  —useful for **big data** or **online learning** .
- cons: less stable in nature like drunk walk

跟 PLA 有點像.

SGD logistic regression, **looks familiar? :-)**:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla \mathrm{err}(\mathbf{w}_t, \mathbf{x}_n, y_n)}$$

PLA Revisited

SGD logistic regression:

"軟"一案

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta\left(\underline{-y_n \mathbf{w}_t^T \mathbf{x}_n}\right)(y_n \mathbf{x}_n)$$

similar

有錯就更新

PLA:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + 1 \cdot \underline{[\![y_n \neq \mathrm{sign}(\mathbf{w}_t^T \mathbf{x}_n)]\!]}(y_n \mathbf{x}_n)$$

sigmoid function

- SGD logistic regression ≈ 'soft' PLA
- PLA ≈ SGD logistic regression with $\eta = 1$ when $\underline{\mathbf{w}_t^T \mathbf{x}_n \text{ large}}$

two practical rule-of-thumb:

- stopping condition? t **large enough**
- η? 0.1 when $\mathbf{x}$ in proper range

停止條件!?

跑夠久就好.

learning rate 設啥?

Questions?

Summary

1. Why Can Machines Learn?

   Lecture 4: Theory of Generalization

2. **How** Can Machines Learn?

   Lecture 5: Linear Models

   - Linear Regression Problem
   - Linear Regression Algorithm
   - Logistic Regression Problem
   - Logistic Regression Error
   - Gradient of Logistic Regression Error
   - Gradient Descent
   - Stochastic Gradient Descent

- **next: beyond simple linear models**