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
- 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)

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

Linear Regression Hypothesis

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

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

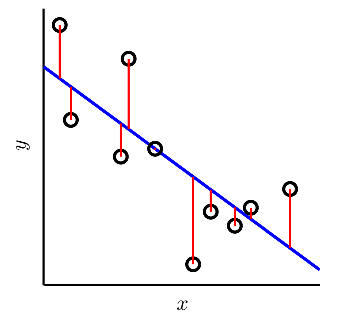
$$y \approx \sum_{i=0}^{d} w_i x_i$$

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

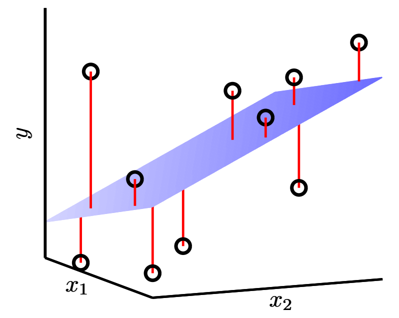
$h(\mathbf{x})$: like **perceptron**, but without the sign

Illustration of Linear Regression

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



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



linear regression:
find lines/hyperplanes with small residuals

Pointwise Error Measure for 'Small Residuals'

final hypothesis
$g \approx f$

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

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

—err: called **pointwise error measure**

in-sample

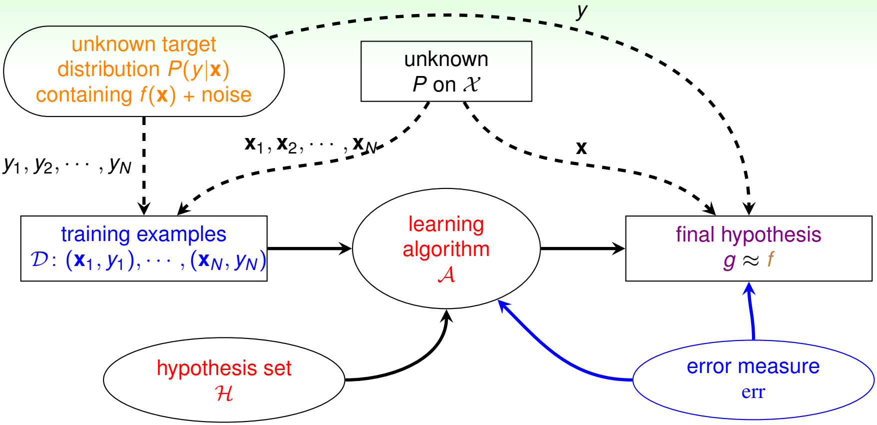
$$E_{\text{in}}(g) = \frac{1}{N} \sum_{n=1}^{N} \text{err}(g(\mathbf{x}_n), f(\mathbf{x}_n))$$

out-of-sample

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

will mainly consider pointwise err for simplicity

Learning Flow with Pointwise Error Measure



extended VC theory/'philosophy'
works for most $\mathcal{H}$ and $\mathbf{err}$

Two Important Pointwise Error Measures

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

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

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

out-of-sample

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

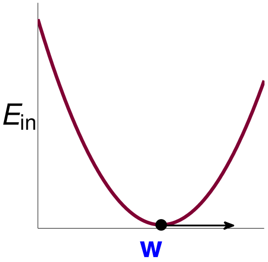
next: how to minimize $E_{\text{in}}(\mathbf{w})$?

Questions?

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

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

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



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

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

—not possible to 'roll down'

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

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

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

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**)

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

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

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

- 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:

use **well-implemented † 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 $\underbrace{\mathrm{X}^\dagger}_{(d+1)\times N}$

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

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!

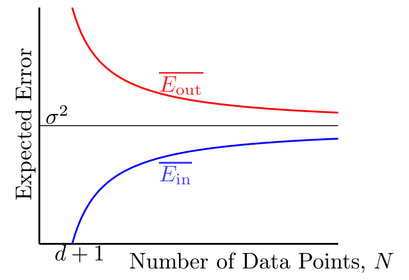
- good E_{in}? **yes, optimal!**
- good E_{out}? **yes, finite d_{VC} like perceptrons**
- improving iteratively? **somewhat, within an iterative pseudo-inverse routine**

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

The Learning Curves of Linear Regression

(proof skipped this year)

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



- 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**

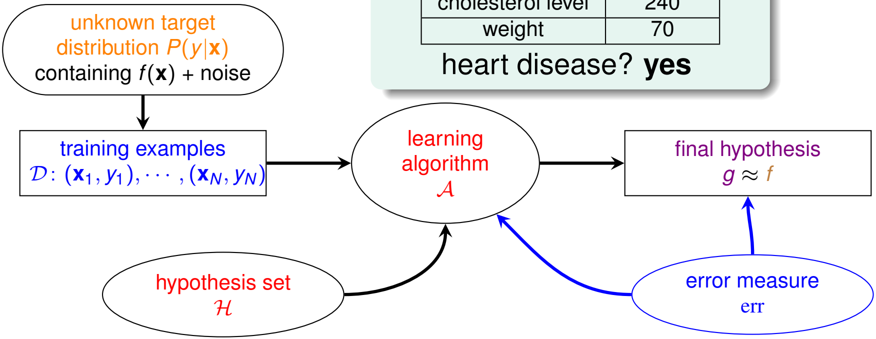
linear regression (LinReg):
learning 'happened'!

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:

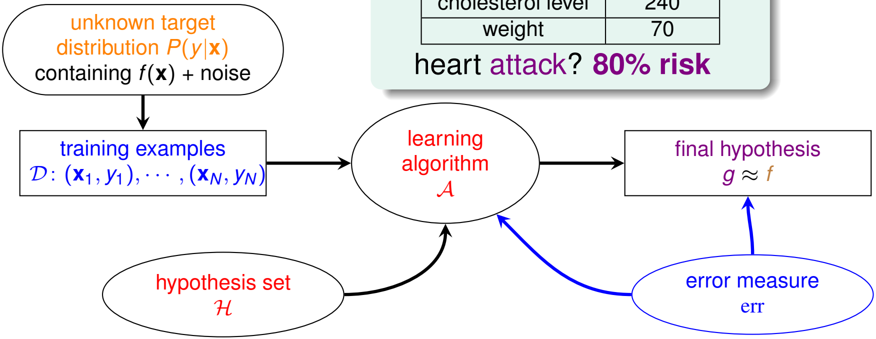
$$\text{ideal } f(\mathbf{x}) = \text{sign}\left(P(+1|\mathbf{x}) - \frac{1}{2}\right) \in \{-1, +1\}$$

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**



'soft' binary classification:

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

Soft Binary Classification

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

ideal (noiseless) data

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

actual (noisy) data

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

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

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

actual (noisy) data

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

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

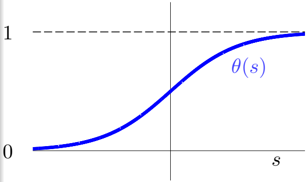
Logistic Hypothesis

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

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

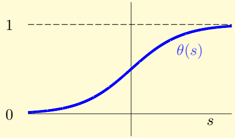
$$s = \sum_{i=0}^{d} w_i x_i$$

- convert the score to estimated probability by logistic function $\theta(s)$



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

Logistic Function



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

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

—smooth, monotonic, sigmoid function of s

logistic regression: use

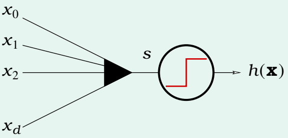
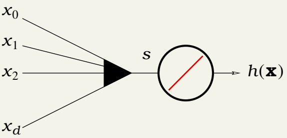
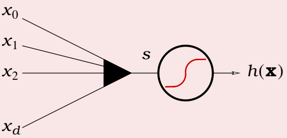
$$h(\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})}$$

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

Questions?

Three Linear Models

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

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$
		
plausible err = 0/1 (small flipping noise)	friendly err = squared (easy to minimize)	err = ?

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

Likelihood

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

probability that f generates $\mathcal{D}$

$$P(\mathbf{x}_1)P(\circ|\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)P(\times|\mathbf{x}_2) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)P(\times|\mathbf{x}_N)$$

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

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

probability that f generates $\mathcal{D}$

$$P(\mathbf{x}_1)f(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - f(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - f(\mathbf{x}_N))$$

likelihood that h generates $\mathcal{D}$

$$P(\mathbf{x}_1)h(\mathbf{x}_1) \times$$
$$P(\mathbf{x}_2)(1 - h(\mathbf{x}_2)) \times$$
$$\ldots$$
$$P(\mathbf{x}_N)(1 - h(\mathbf{x}_N))$$

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

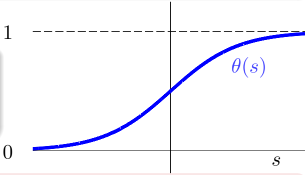
Likelihood of Logistic Hypothesis

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

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

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

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



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

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

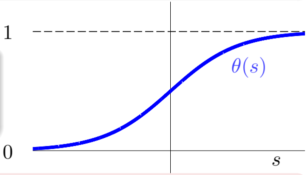
Likelihood of Logistic Hypothesis

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

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

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

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



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

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

Cross-Entropy Error

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

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \text{likelihood}(\mathbf{w}) \propto \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

$$\max_{\mathbf{w}} \quad \ln \prod_{n=1}^{N} \theta\left(y_n \mathbf{w}^T \mathbf{x}_n\right)$$

Cross-Entropy Error

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

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

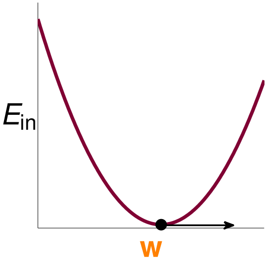
$$\Longrightarrow \quad \min_{\mathbf{w}} \underbrace{\frac{1}{N}\sum_{n=1}^{N} \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)}_{E_{\text{in}}(\mathbf{w})}$$

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

Questions?

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

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



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

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

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

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

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

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

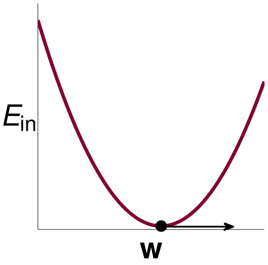
$$\begin{aligned}
\frac{\partial E_{\text{in}}(\mathbf{w})}{\partial w_i} &= \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\partial \ln(\square)}{\partial \square}\right)\left(\frac{\partial(1+\exp(\bigcirc))}{\partial \bigcirc}\right)\left(\frac{\partial -y_n\mathbf{w}^T\mathbf{x}_n}{\partial w_i}\right) \\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

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

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

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

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



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

- all $\theta(\cdot) = 0$: only if $y_n\mathbf{w}^T\mathbf{x}_n \gg 0$ —linear separable $\mathcal{D}$
- 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 $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

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

2. (try to) correct the mistake by

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

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 $\left(\mathbf{x}_{n(t)}, y_{n(t)}\right)$

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

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

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

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

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

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

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

Questions?

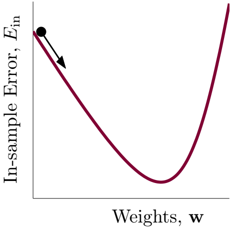
Iterative Optimization

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 $\eta > 0$:

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

- 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

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

  if η really small (Taylor expansion)

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:

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

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

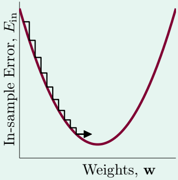
$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

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

gradient descent:
a simple & popular optimization tool

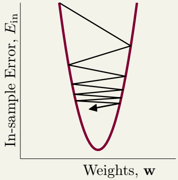
Choice of η

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)

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

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

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

2. update by

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

...until $\nabla E_{\text{in}}(\mathbf{w}_{t+1}) = 0$ or enough iterations

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

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

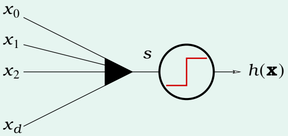
Questions?

Linear Models Revisited

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

linear classification

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

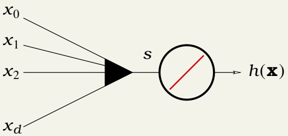


plausible err = 0/1

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

linear regression

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

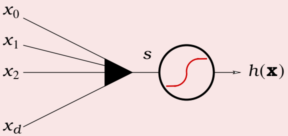


friendly err = squared

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

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

linear classification

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

$$\begin{aligned} &\text{err}_{0/1}(s, y) \\ = &\ [\![\text{sign}(s) \neq y]\!] \\ = &\ [\![\text{sign}(ys) \neq 1]\!] \end{aligned}$$

linear regression

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

$$\begin{aligned} &\text{err}_{\text{SQR}}(s, y) \\ = &\ (s - y)^2 \\ = &\ (ys - 1)^2 \end{aligned}$$

logistic regression

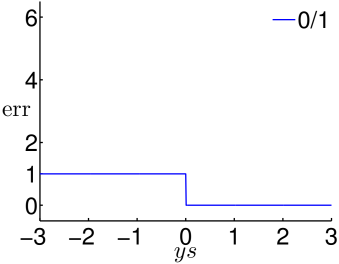
$$\begin{aligned} h(\mathbf{x}) &= \theta(s) \\ \text{err}(h, \mathbf{x}, y) &= -\ln h(y\mathbf{x}) \end{aligned}$$

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

(ys): classification correctness score

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	$\mathrm{err}_{\mathrm{SCE}}(s, y)$	$=$	$\log_2(1 + \exp(-ys))$

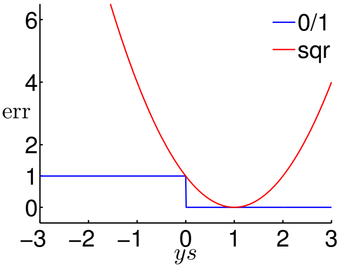


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1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

upper bound:
useful for designing algorithm

Visualizing Error Functions

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

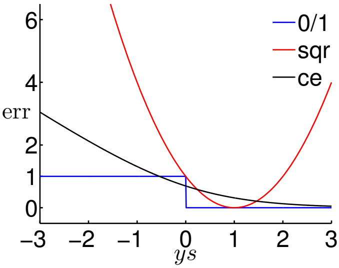


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1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

upper bound:
useful for designing algorithm

Visualizing Error Functions

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

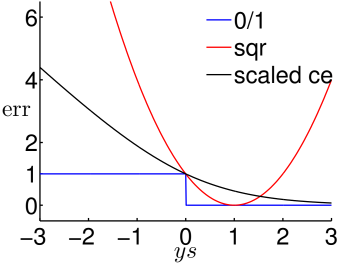


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\mathrm{err}_{\mathrm{SQR}} \rightarrow$ small $\mathrm{err}_{0/1}$
- ce: monotonic of ys
  small $\mathrm{err}_{\mathrm{CE}} \leftrightarrow$ small $\mathrm{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1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

upper bound:
useful for designing algorithm

Visualizing Error Functions

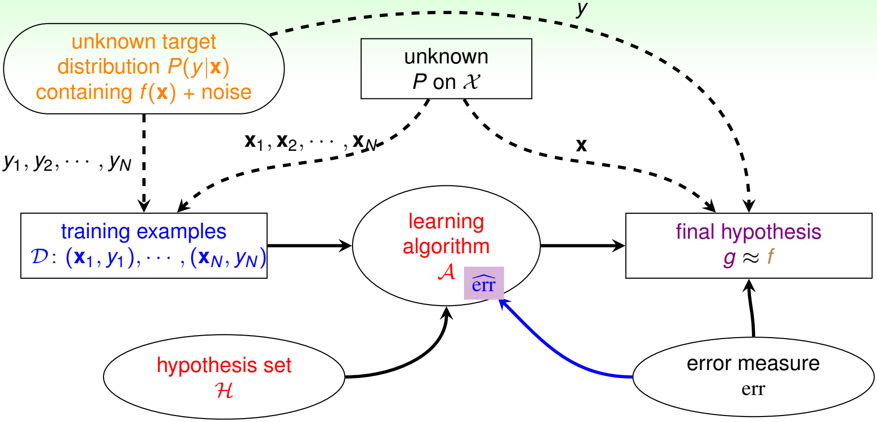
$$
\begin{array}{rrcl}
0/1 & \mathrm{err}_{0/1}(s,y) & = & [\![\mathrm{sign}(ys) \neq 1]\!] \\
\mathrm{sqr} & \mathrm{err}_{\mathrm{SQR}}(s,y) & = & (ys-1)^2 \\
\mathrm{ce} & \mathrm{err}_{\mathrm{CE}}(s,y) & = & \ln(1+\exp(-ys)) \\
\text{scaled ce} & \mathrm{err}_{\mathrm{SCE}}(s,y) & = & \log_2(1+\exp(-ys))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



- 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

Learning Flow with Algorithmic Error Measure



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

Theoretical Implication of Upper Bound

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

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

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

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

VC on $0/1$:

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

VC-Reg on CE :

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

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

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 $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{REG}}^T \mathbf{x})$

PLA

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

linear regression

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

logistic regression

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

- **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

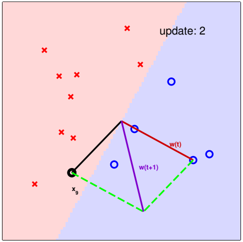
pick $(\mathbf{x}_n, y_n)$ and decide $\mathbf{w}_{t+1}$ by **the one example**

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

logistic regression

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

$O(N)$ time per iteration **:-(**



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 $\frac{1}{N}\sum_{n=1}^{N}$:
  view as expectation $\mathcal{E}$ over uniform choice of n!

stochastic gradient:

$$\nabla_{\mathbf{w}}\ \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n) \text{ with random } n$$

true gradient:

$$\nabla_{\mathbf{w}}\ E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}}\ \nabla_{\mathbf{w}}\ \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)$$

Stochastic Gradient Descent (SGD)

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

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps,
  average true gradient $\approx$ average stochastic gradient
- pros: **simple & cheaper computation :-)**
  —useful for **big data** or **online learning**
- cons: less stable in nature

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:

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

PLA:

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

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

two practical rule-of-thumb:

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

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**