Machine Learning Foundations
(機器學習基石)

Lecture 9: Linear Regression

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?

Lecture 8: Noise and Error

learning can happen
with **target distribution** $P(y|\mathbf{x})$ and **low E_{in} w.r.t. err**

3 **How** Can Machines Learn?

Lecture 9: Linear Regression

- Linear Regression Problem
- Linear Regression Algorithm
- Generalization Issue
- Linear Regression for Binary Classification

4 How Can Machines Learn Better?

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

+1

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of customer', approximate the desired credit limit with a weighted sum:
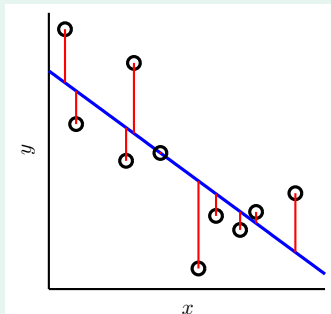
$$y \approx \sum_{i=0}^{d} w_i x_i$$

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

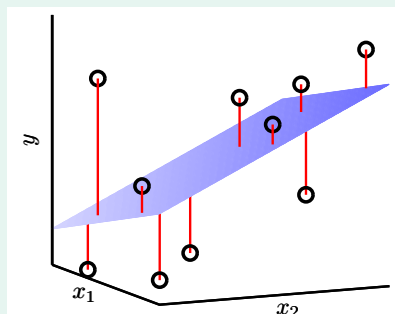$h(\mathbf{x})$: like **perceptron**, but without the sign

sign(Σwx)

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 <u>small</u> residuals

minimize

The Error Measure

popular/historical error measure:

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

in-sample

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

out-of-sample

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

We may have noise.

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

Fun Time

Consider using linear regression hypothesis $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict the credit limit of customers $\mathbf{x}$. Which feature below shall have a positive weight in a **good hypothesis** for the task?

1. birth month
2. monthly income
3. current debt
4. number of credit cards owned

Fun Time

Consider using linear regression hypothesis $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict the credit limit of customers $\mathbf{x}$. Which feature below shall have a positive weight in a **good hypothesis** for the task?

1. birth month
2. monthly income
3. current debt
4. number of credit cards owned

Reference Answer: ②

Customers with higher monthly income should naturally be given a higher credit limit, which is captured by the positive weight on the 'monthly income' feature.

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

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

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|X\mathbf{w} - \mathbf{y}\|^2$$
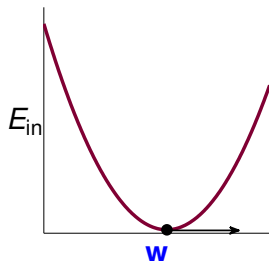
- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of 'best' $\mathbf{w}$

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

—not possible to 'roll down'

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

LIN: linear regression

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

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

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**)

A is a symmetric matrix.

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

Optimal Linear Regression Weights

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

First scenario:

invertible $X^TX_{(d+1, d+1)}$ $\leftarrow X_{(N, d+1)}$

- **easy!** <u>unique solution</u>

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

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

Second scenario:

singular X^TX

- **<u>many</u>** optimal solutions
- one of the solutions

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

by defining $X^\dagger$ in other ways

practical suggestion:
use **well-implemented** † **routine**
instead of $\left(X^TX\right)^{-1}X^T$
for numerical stability when **almost-singular**

Linear Regression Algorithm

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

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

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**

Fun Time

After getting $\mathbf{w}_{\text{LIN}}$, we can calculate the predictions $\hat{y}_n = \mathbf{w}_{\text{LIN}}^T \mathbf{x}_n$. If all $\hat{y}_n$ are collected in a vector $\hat{\mathbf{y}}$ similar to how we form $\mathbf{y}$, what is the matrix formula of $\hat{\mathbf{y}}$?

1. $\mathbf{y}$
2. $\mathbf{X}\mathbf{X}^T\mathbf{y}$
3. $\mathbf{X}\mathbf{X}^\dagger\mathbf{y}$
4. $\mathbf{X}\mathbf{X}^\dagger\mathbf{X}\mathbf{X}^T\mathbf{y}$

Fun Time

After getting $\mathbf{w}_{\text{LIN}}$, we can calculate the predictions $\hat{y}_n = \mathbf{w}_{\text{LIN}}^T \mathbf{x}_n$. If all $\hat{y}_n$ are collected in a vector $\hat{\mathbf{y}}$ similar to how we form $\mathbf{y}$, what is the matrix formula of $\hat{\mathbf{y}}$?

1 $\mathbf{y}$

2 $\mathrm{X}\mathrm{X}^T\mathbf{y}$

3 $\mathrm{X}\mathrm{X}^{\dagger}\mathbf{y}$

4 $\mathrm{X}\mathrm{X}^{\dagger}\mathrm{X}\mathrm{X}^T\mathbf{y}$

Reference Answer: ③

Note that $\hat{\mathbf{y}} = \mathrm{X}\mathbf{w}_{\text{LIN}}$. Then, a simple substitution of $\mathbf{w}_{\text{LIN}}$ reveals the answer.

Is Linear Regression a 'Learning Algorithm'?

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

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

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

Benefit of Analytic Solution:
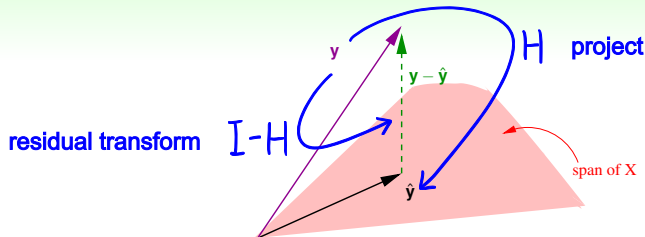## 'Simpler-than-VC' Guarantee

$$\overline{E_{in}} = \underset{\mathcal{D} \sim P^N}{\mathcal{E}} \left\{ E_{in}(\mathbf{w}_{LIN} \text{ w.r.t. } \mathcal{D}) \right\} \overset{\text{to be shown}}{=} \text{noise level} \cdot \left(1 - \frac{d+1}{N} \right)$$

$$E_{in}(\mathbf{w}_{LIN}) = \frac{1}{N} \| \mathbf{y} - \underbrace{\hat{\mathbf{y}}}_{\text{predictions}} \|^2 = \frac{1}{N} \| \mathbf{y} - X \underbrace{X^\dagger \mathbf{y}}_{\mathbf{w}_{LIN}} \|^2$$

$$= \frac{1}{N} \| (\underbrace{I}_{\text{identity}} - XX^\dagger)\mathbf{y} \|^2$$

call $XX^\dagger$ the hat matrix H
because it puts $\wedge$ on $\mathbf{y}$

Geometric View of Hat Matrix

residual transform $I-H$ **project**

y H

$y - \hat{y}$

span of X

$\hat{y}$

in $\mathbb{R}^N$

- $\hat{y} = X w_{\text{LIN}}$ within the span of X columns

- $y - \hat{y}$ smallest: $y - \hat{y} \perp$ span

- H: project y to $\hat{y} \in$ span

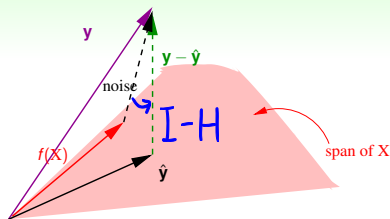- $I - H$: transform y to $y - \hat{y} \perp$ span

$$\begin{bmatrix} | & | & & | \\ x_0 & x_1 & \cdots & x_d \\ | & | & & | \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

$$= w_0 \begin{bmatrix} | \\ x_0 \\ | \end{bmatrix} + w_1 \begin{bmatrix} | \\ x_1 \\ | \end{bmatrix} + \cdots$$

claim: $\text{trace}(I - H) = N - (d + 1)$. **Why? :-)**

An Illustrative 'Proof'

- if **y** comes from some ideal $f(X) \in$ span plus **noise**
- **noise** transformed by $I - H$ to be $\mathbf{y} - \hat{\mathbf{y}}$

$$E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 \quad = \quad \frac{1}{N}\|(I - H)\textbf{noise}\|^2$$
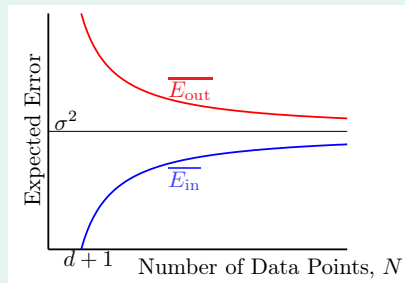$$= \quad \frac{1}{N}\big(N - (d+1)\big)\|\textbf{noise}\|^2$$

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

The Learning Curve

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

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

- 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'!

Fun Time

Which of the following property about H is not true?

1. H is symmetric
2. $H^2 = H$ (double projection = single one)
3. $(I - H)^2 = I - H$ (double residual transform = single one)
4. none of the above

Fun Time

Which of the following property about H is not true?

1. H is symmetric
2. $H^2 = H$ (double projection = single one)
3. $(I - H)^2 = I - H$ (double residual transform = single one)
4. none of the above

Reference Answer: ④

You can conclude that ② and ③ are true by their physical meanings! **:-)**

Linear Classification vs. Linear Regression

Linear Classification

$$\mathcal{Y} = \{-1, +1\}$$
$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$$
$$\text{err}(\hat{y}, y) = [\![\hat{y} \neq y]\!]$$

NP-hard to solve in general

Linear Regression

$$\mathcal{Y} = \mathbb{R}$$
$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$
$$\text{err}(\hat{y}, y) = (\hat{y} - y)^2$$

efficient analytic solution

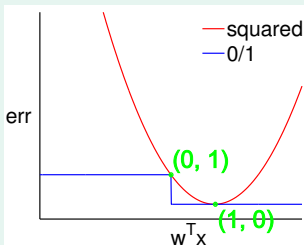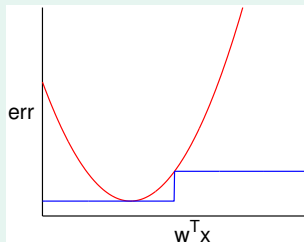$\{-1, +1\} \subset \mathbb{R}$: linear regression for classification?
Can we treat {-1, +1} as real numbers and apply linear regression on classification problem?

1. run LinReg on binary classification data $\mathcal{D}$ (**efficient**)
2. return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{LIN}}^T\mathbf{x})$

but explanation of this **heuristic**?

Relation of Two Errors

$$\text{err}_{0/1} = \left[\!\left[\text{sign}(\mathbf{w}^T\mathbf{x}) \neq y\right]\!\right] \quad \text{err}_{\text{sqr}} = \left(\mathbf{w}^T\mathbf{x} - y\right)^2$$

desired $y = 1$

desired $y = -1$

$$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$$

Linear Regression for Binary Classification

$$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$$

$$\text{classification } E_{\text{out}}(\mathbf{w}) \overset{\text{VC}}{\leq} \text{classification } E_{\text{in}}(\mathbf{w}) + \sqrt{\cdots\cdots}$$
$$\leq \text{regression } E_{\text{in}}(\mathbf{w}) + \sqrt{\cdots\cdots}$$

- (loose) upper bound err_{sqr} as $\widehat{\text{err}}$ to approximate $\text{err}_{0/1}$
- trade **bound tightness** for **efficiency**

$\mathbf{w}_{\text{LIN}}$: useful baseline classifier,
or as **initial PLA/pocket vector** $\mathbf{w_0}$

Fun Time

Which of the following functions are upper bounds of the pointwise 0/1 error $[\![\text{sign}(\mathbf{w}^T\mathbf{x}) \neq y]\!]$ for $y \in \{-1, +1\}$?

1. $\exp(-y\mathbf{w}^T\mathbf{x})$
2. $\max(0, 1 - y\mathbf{w}^T\mathbf{x})$
3. $\log_2(1 + \exp(-y\mathbf{w}^T\mathbf{x}))$
4. all of the above

Fun Time

Which of the following functions are upper bounds of the pointwise 0/1 error $[\![\text{sign}(\mathbf{w}^T\mathbf{x}) \neq y]\!]$ for $y \in \{-1, +1\}$?

1. $\exp(-y\mathbf{w}^T\mathbf{x})$
2. $\max(0, 1 - y\mathbf{w}^T\mathbf{x})$
3. $\log_2(1 + \exp(-y\mathbf{w}^T\mathbf{x}))$
4. all of the above

Reference Answer: ④

Plot the curves and you'll see. Thus, all three can be used for binary classification. In fact, <u>all three functions connect to very important algorithms in machine learning</u> and we will discuss one of them soon in the next lecture. **Stay tuned. :-)**

Summary

1. When Can Machines Learn?
2. Why Can Machines Learn?

Lecture 8: Noise and Error

3. **How** Can Machines Learn?

Lecture 9: Linear Regression

- Linear Regression Problem
  use hyperplanes to approximate real values
- Linear Regression Algorithm
  analytic solution with pseudo-inverse
- Generalization Issue
  $$E_{out} - E_{in} \approx \frac{2(d+1)}{N} \text{ on average}$$
- Linear Regression for Binary Classification
  0/1 error $\leq$ squared error

- **next: binary classification, regression, and then?**

4. How Can Machines Learn Better?