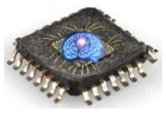
EECE-6822 Machine Learning



Linear Regression

Cris Ababei
Dept. of Electrical and Computer Engineering
MARQUETTE UNIVERSITY
BE THE DIFFERENCE.

PART 1

Linear Regression

Outline

- Linear Regression
- Regression for Classification
- Nonlinear Transformation
- Linear Regression with Basis Functions
- Generalization (feasibility of learning)
- Split Data into Train/Test Portions
- Cross Validation

3

3

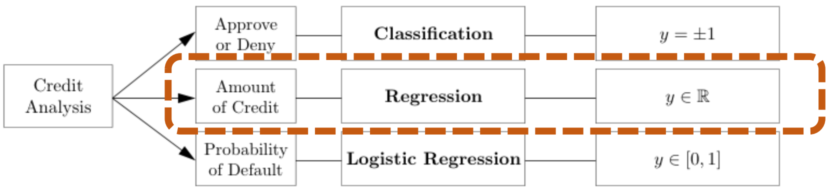
What is Linear Regression?

- Regression = real-valued output
- Linear regression
  - A type of supervised machine-learning algorithm that learns from labelled datasets and maps the data points with most optimized linear functions which can be used for prediction on new datasets.
  - Assumes that there is a linear relationship between the input and output, meaning the output changes at a constant rate as the input changes. This relationship is represented by a straight line.
- Example:
  - We want to predict a student's exam score based on how many hours they studied (it was observed that as students study more hours, their scores go up ☺).
  - Independent variable (input): Hours studied; it is a factor we control/observe.
  - Dependent variable (output): Exam score; depends on how many hours were studied.

4

4

Three Learning Problems



- Linear models are perhaps ***the*** fundamental model.
- The linear model is the first model to try.

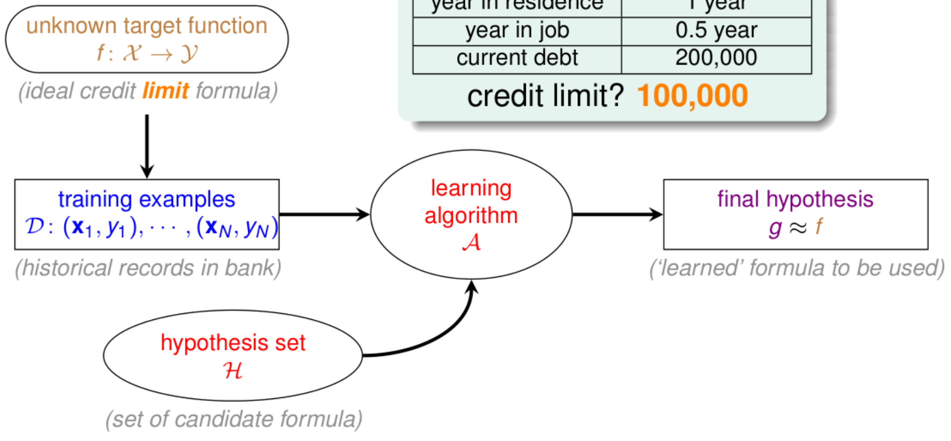
5

5

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**

6

6

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:

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

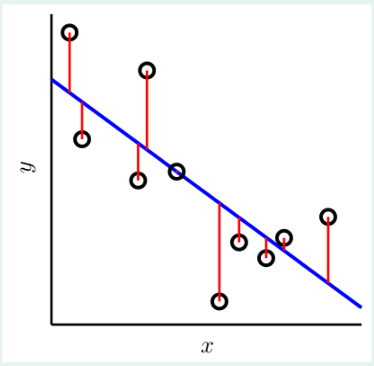
- linear regression hypothesis: $h(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$

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

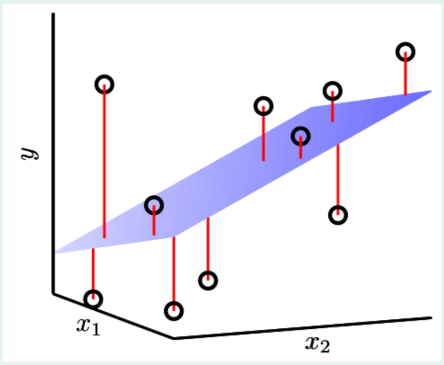
7

Illustration of Linear Regression

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



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

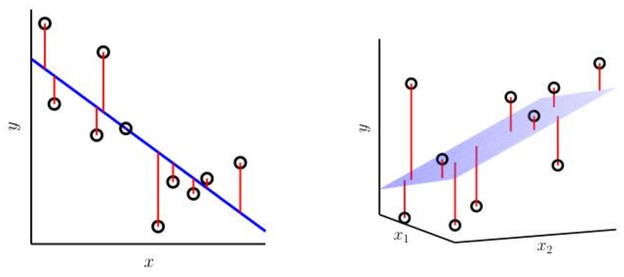


linear regression:
find lines/hyperplanes with small residuals

3

8

Least Squares Linear Regression



$y = f(\mathbf{x}) + \epsilon \qquad \longleftarrow$ noisy target $P(y|\mathbf{x})$

in-sample error $\qquad E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n) - y_n)^2$

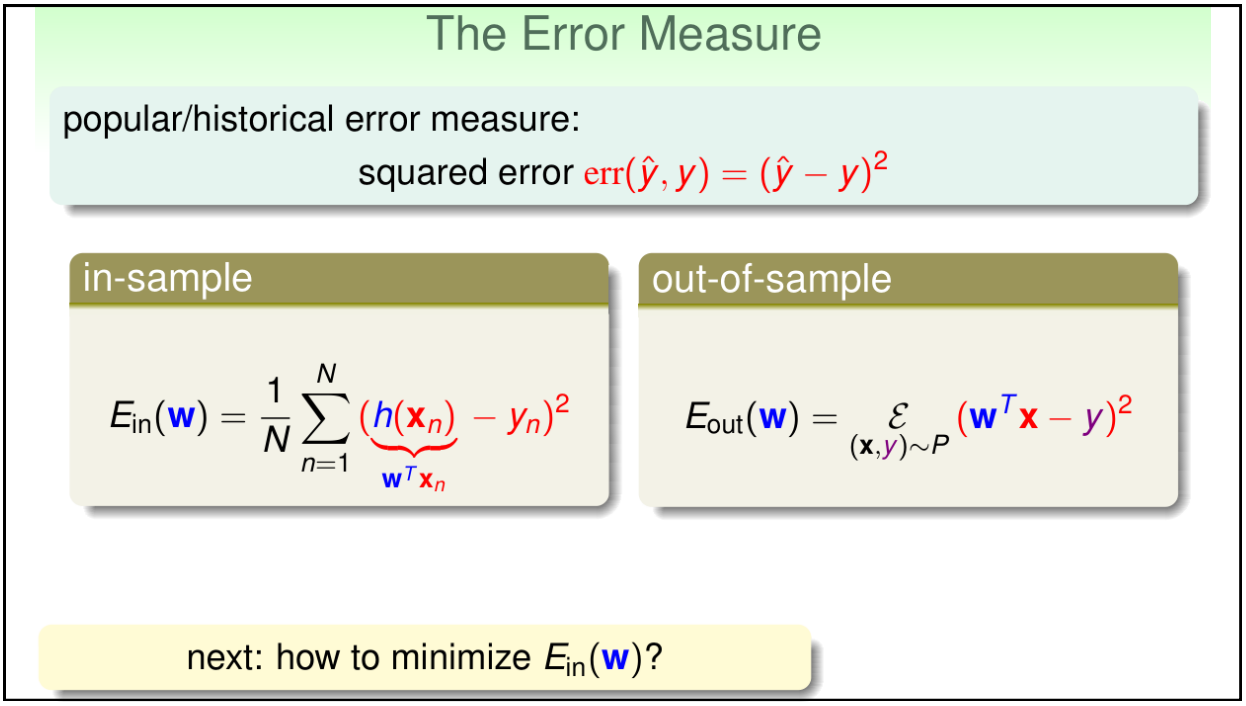
out-of-sample error $\qquad E_{\text{out}}(h) = \mathbb{E}_{\mathbf{x}}[(h(\mathbf{x}) - y)^2]$

$\Bigg\} \quad h(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\mathbf{x}$

Recall:

The out-of-sample error E_{out} measures how well our training on $\mathcal{D}$ has *generalized* to data that we have not seen before. E_{out} is based on the performance over the entire input space $\mathcal{X}$. Intuitively, if we want to estimate the value of E_{out} using a sample of data points, these points must be 'fresh' test points that have not been used for training, similar to the questions on the final exam that have not been used for practice.

The in sample error E_{in}, by contrast, is based on data points that have been used for training. It expressly measures training performance, similar to your performance on the practice problems that you got before the final exam.



The Error Measure

popular/historical error measure:

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

in-sample

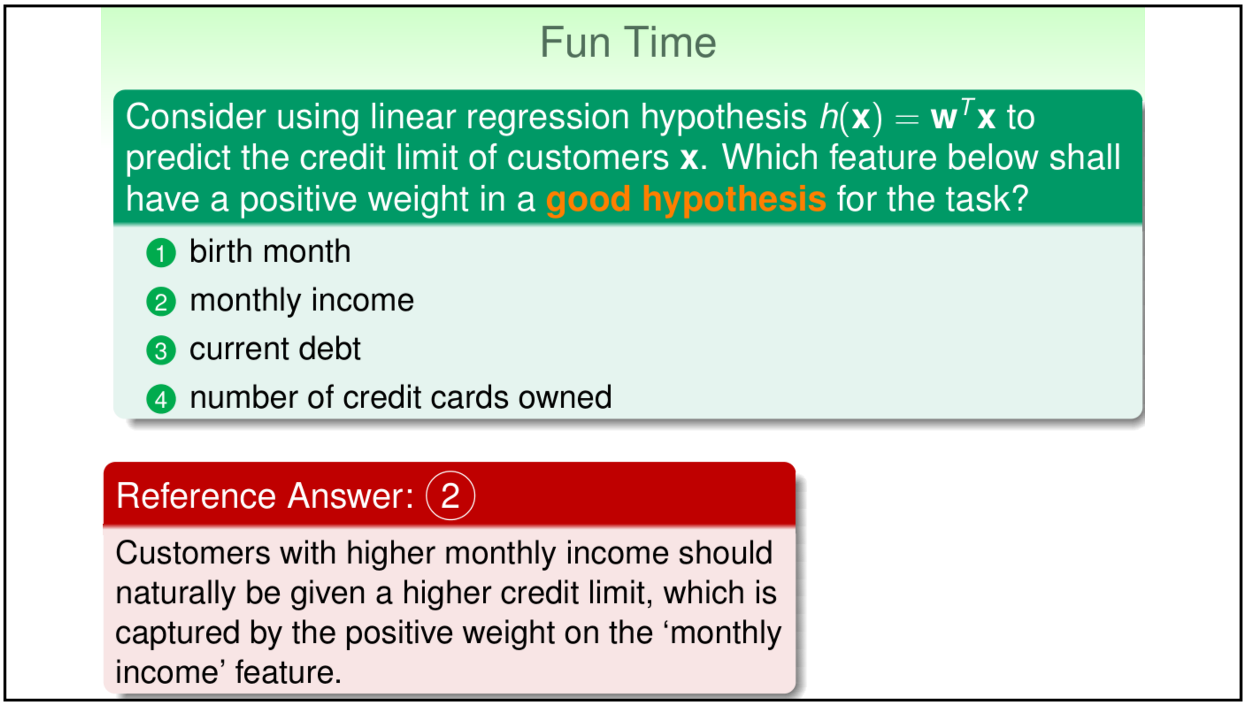
$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n)^2$$

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

11



Fun Time

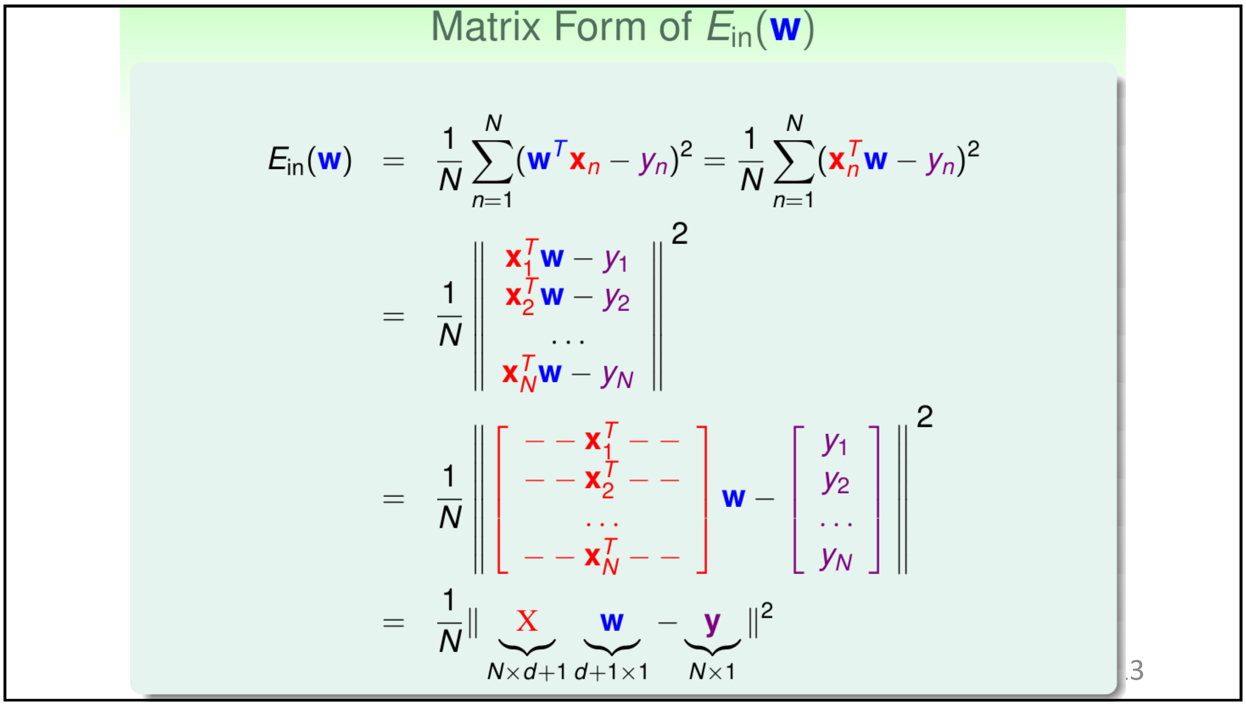
Consider using linear regression hypothesis $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ to predict the credit limit of customers $\mathbf{x}$. Which feature below shall have a positive weight in a **good hypothesis** for the task?

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.

12

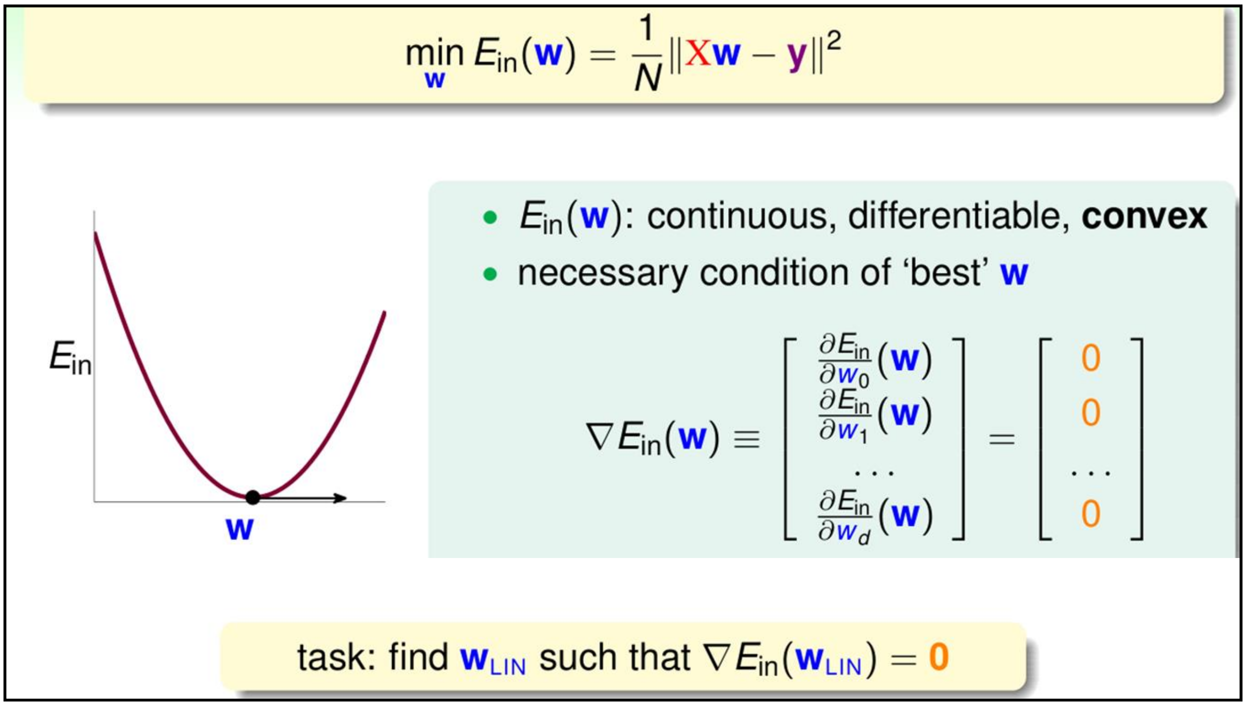


Matrix Form of $E_{\text{in}}(\mathbf{w})$

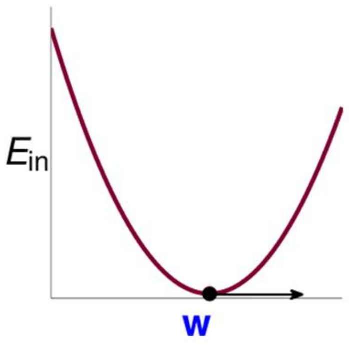
$$
\begin{aligned}
E_{\text{in}}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - y_n)^2 \\
&= \frac{1}{N}\left\| \begin{matrix} \mathbf{x}_1^T\mathbf{w} - y_1 \\ \mathbf{x}_2^T\mathbf{w} - y_2 \\ \dots \\ \mathbf{x}_N^T\mathbf{w}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

.3

13



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

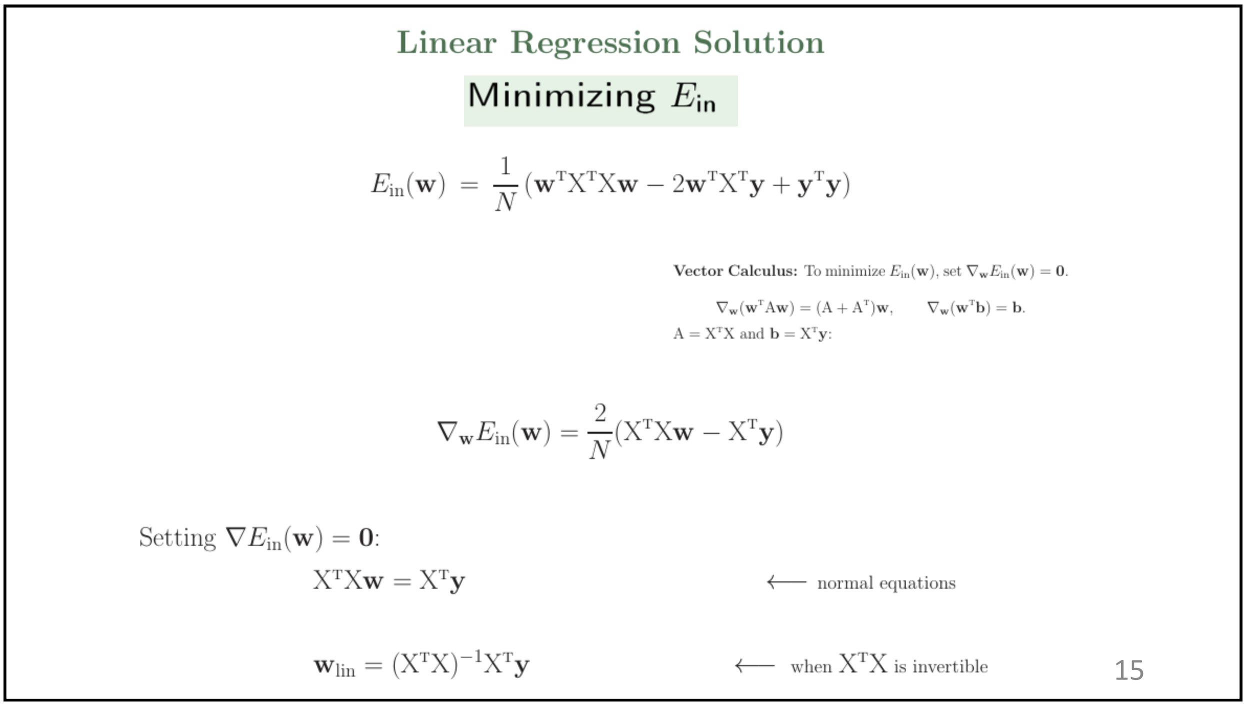


- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of 'best' $\mathbf{w}$

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

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

14



Linear Regression Solution

Minimizing E_{in}

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

Vector Calculus: To minimize $E_{\text{in}}(\mathbf{w})$, set $\nabla_{\mathbf{w}}E_{\text{in}}(\mathbf{w}) = \mathbf{0}$.

$$\nabla_{\mathbf{w}}(\mathbf{w}^{\mathrm{T}}\mathrm{A}\mathbf{w}) = (\mathrm{A} + \mathrm{A}^{\mathrm{T}})\mathbf{w}, \qquad \nabla_{\mathbf{w}}(\mathbf{w}^{\mathrm{T}}\mathbf{b}) = \mathbf{b}.$$

$\mathrm{A} = \mathrm{X}^{\mathrm{T}}\mathrm{X}$ and $\mathbf{b} = \mathrm{X}^{\mathrm{T}}\mathbf{y}$:

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

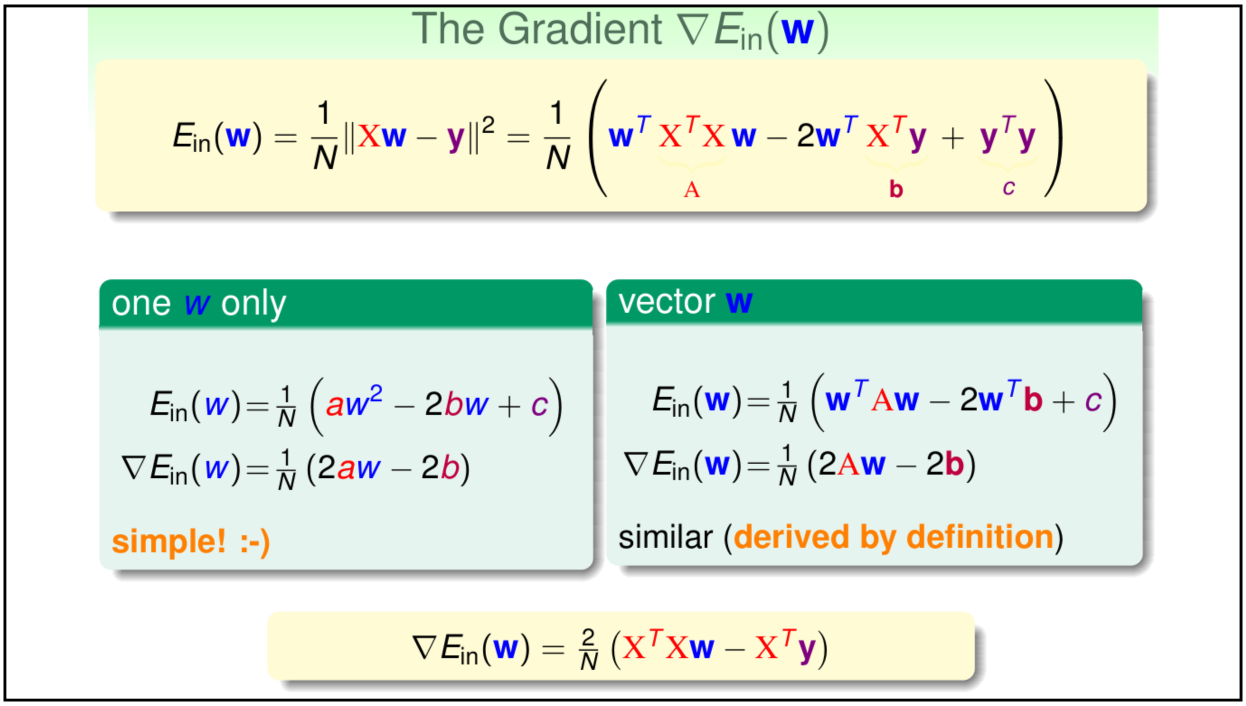
Setting $\nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$:

$$\mathrm{X}^{\mathrm{T}}\mathrm{X}\mathbf{w} = \mathrm{X}^{\mathrm{T}}\mathbf{y} \qquad \longleftarrow \text{ normal equations}$$

$$\mathbf{w}_{\text{lin}} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathbf{y} \qquad \longleftarrow \text{ when } \mathrm{X}^{\mathrm{T}}\mathrm{X} \text{ is invertible}$$

15

15



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

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

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

simple! :-)

vector $\mathbf{w}$

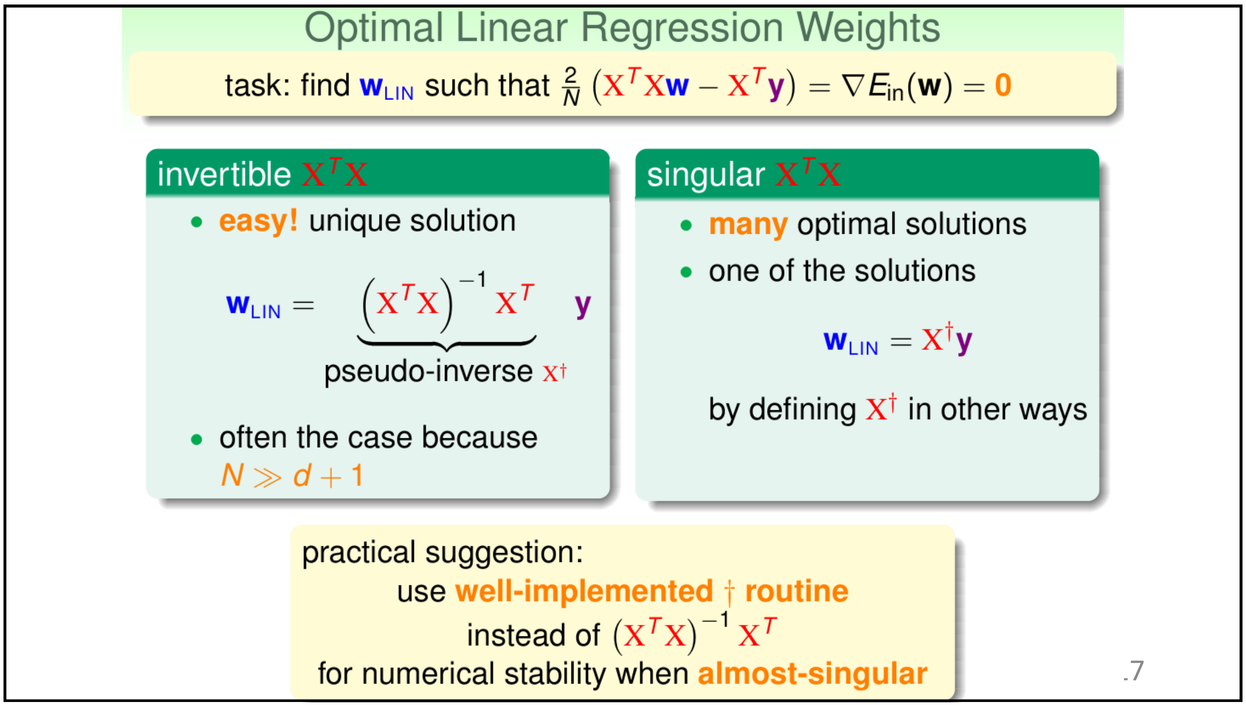
$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(\mathbf{w}^T\mathrm{A}\mathbf{w} - 2\mathbf{w}^T\mathbf{b} + c\right)$$

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

similar (**derived by definition**)

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

16



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

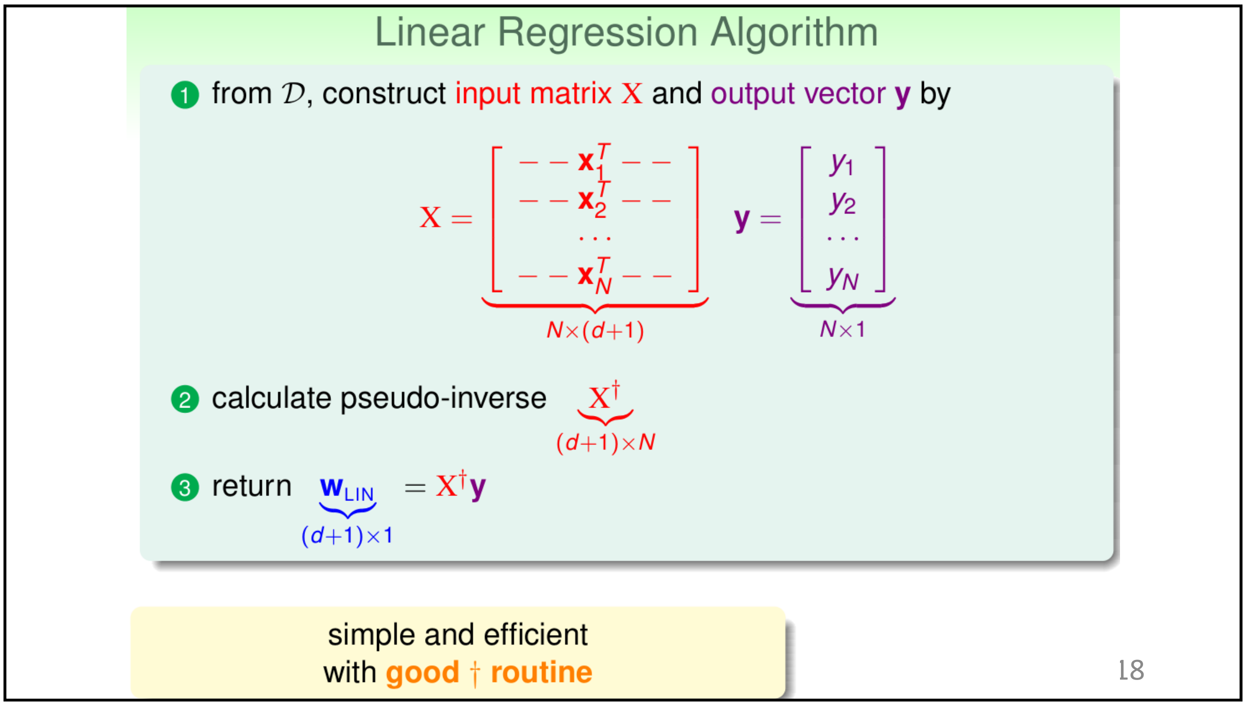
$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^\dagger\mathbf{y}$$

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

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

7

17



Linear Regression Algorithm

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

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

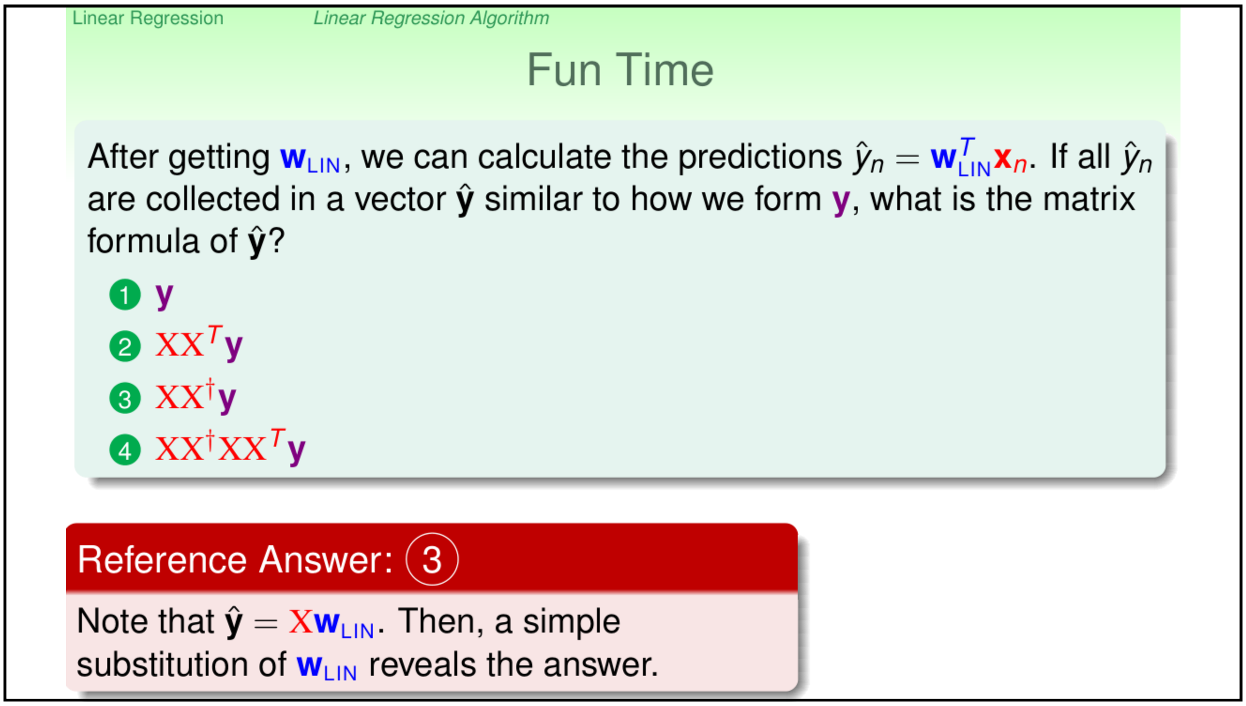
2. calculate pseudo-inverse $\underbrace{\mathrm{X}^\dagger}_{(d+1)\times N}$

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

simple and efficient
with **good $\dagger$ routine**

18

18



Fun Time

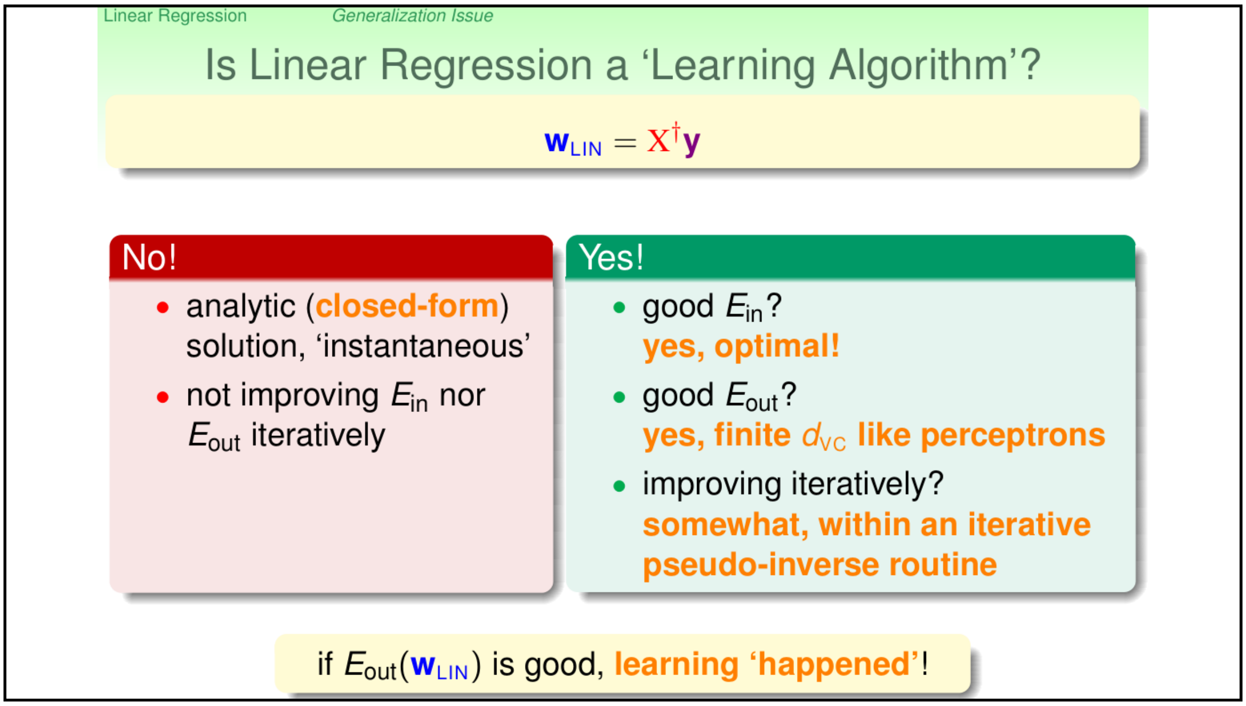
After getting $\mathbf{w}_{\text{LIN}}$, we can calculate the predictions $\hat{y}_n = \mathbf{w}_{\text{LIN}}^T \mathbf{x}_n$. If all $\hat{y}_n$ are collected in a vector $\hat{\mathbf{y}}$ similar to how we form $\mathbf{y}$, what is the matrix formula of $\hat{\mathbf{y}}$?

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.

19



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'**!

20

Conclusion so far:
A way to train a linear regression model

Takeaway

- Using a "closed form" equation (the Normal Equation) that directly computes the model parameters that best fit the model to the training set (i.e., the model parameters that minimize the cost function over the training set).

$$\boxed{\hat{w}_{LS} = \hat{w}_{MLE} = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1}\boldsymbol{X}^T\boldsymbol{y}}$$

21

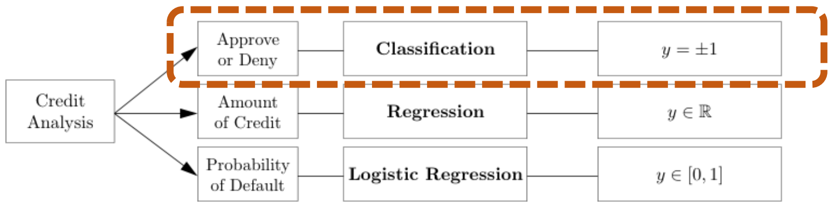
21

Regression for Classification

22

22

Three Learning Problems



- Linear models are perhaps ***the*** fundamental model.
- The linear model is the first model to try.

Linear Regression for Classification

Linear regression can learn *any* real valued target function.

For example $y_n = \pm 1$. (± 1 are real values!)

Use linear regression to get $\mathbf{w}$ with $\mathbf{w}^{\mathrm{T}}\mathbf{x}_n \approx y_n = \pm 1$

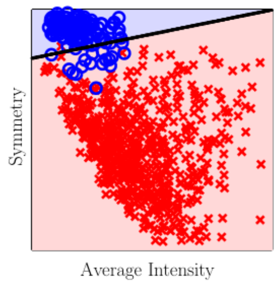
Then $\text{sign}(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n)$ will likely agree with $y_n = \pm 1$.

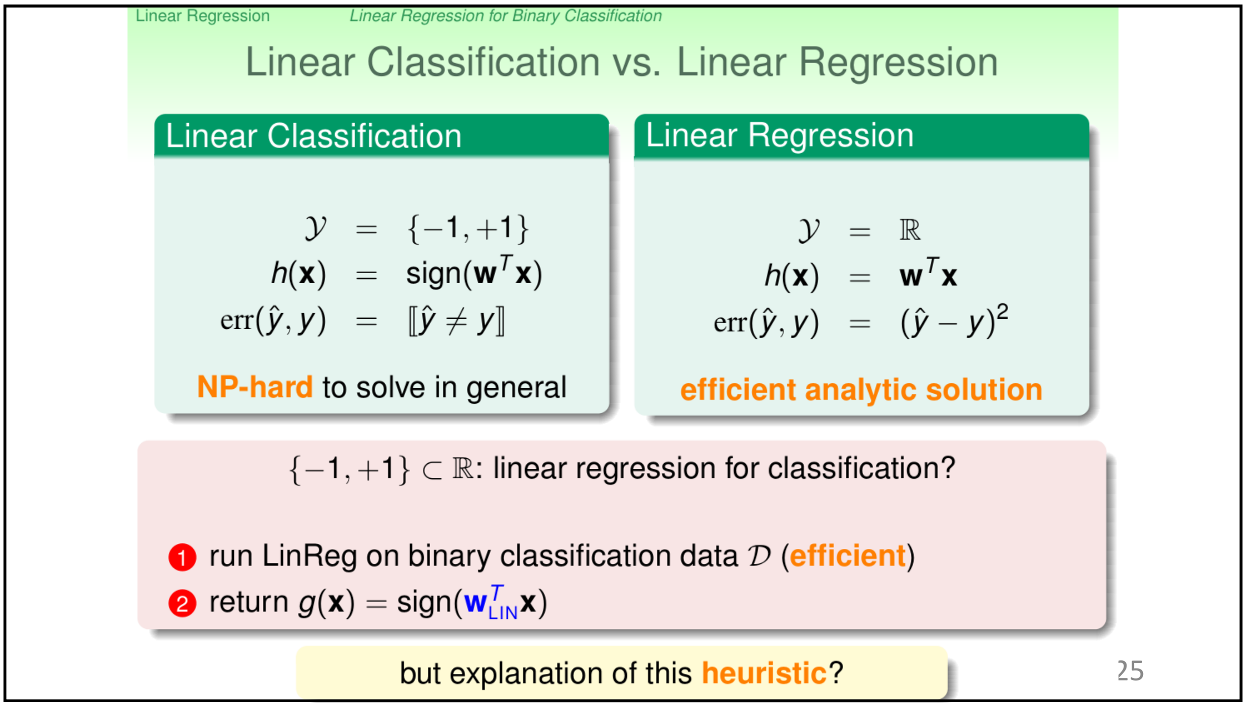
These can be good initial weights for classification.

Example.

Classifying 1 from not 1

(multiclass $\rightarrow$ 2 class)





Linear Classification vs. Linear Regression

Linear Classification

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

NP-hard to solve in general

Linear Regression

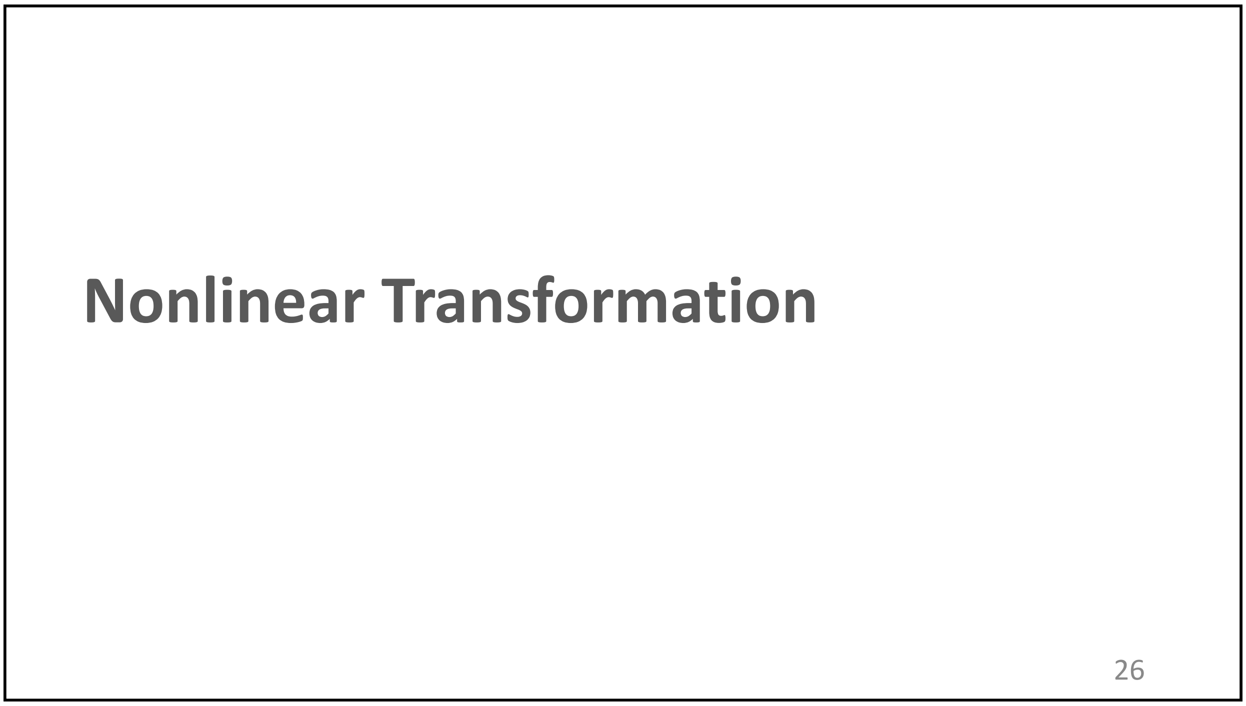
$$\begin{aligned} \mathcal{Y} &= \mathbb{R} \\ h(\mathbf{x}) &= \mathbf{w}^T\mathbf{x} \\ \text{err}(\hat{y}, y) &= (\hat{y} - y)^2 \end{aligned}$$

efficient analytic solution

$\{-1,+1\} \subset \mathbb{R}$: linear regression for classification?

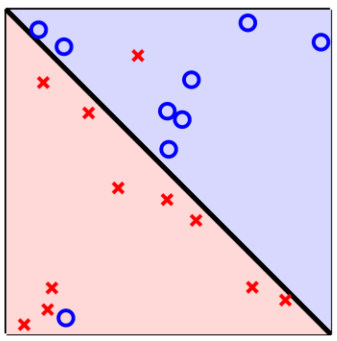
1. run LinReg on binary classification data $\mathcal{D}$ (efficient)
2. return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{LIN}}^T\mathbf{x})$

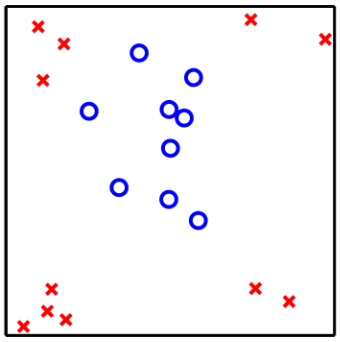
but explanation of this heuristic?



Nonlinear Transformation

The Linear Model has its Limits


(a) Linear with outliers

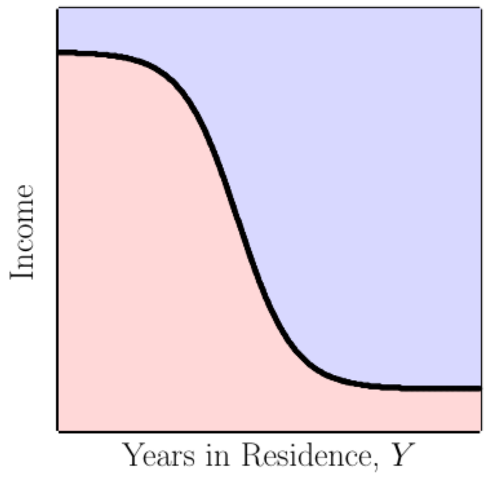

(b) Essentially nonlinear

To address (b) we need something more than linear.

27

27

Change Your Features



$Y \gg 3$ years

no additional effect beyond $Y = 3$;

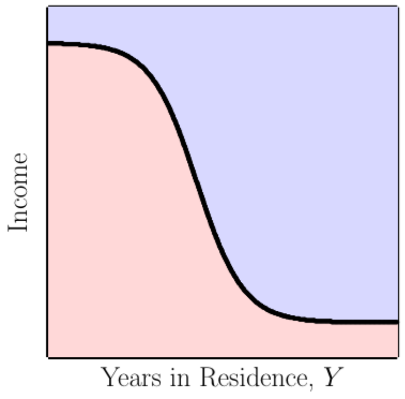
$Y \ll 0.3$ years

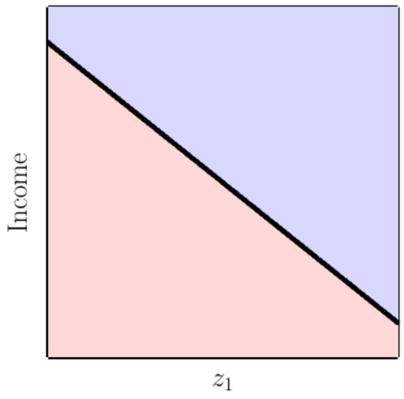
no additional effect below $Y = 0.3$.

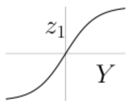
28

28

Change Your Features Using a Transform

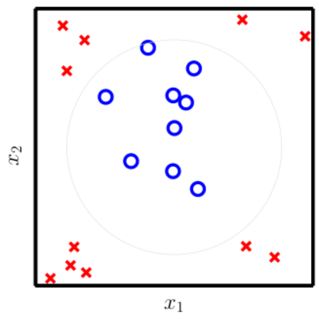




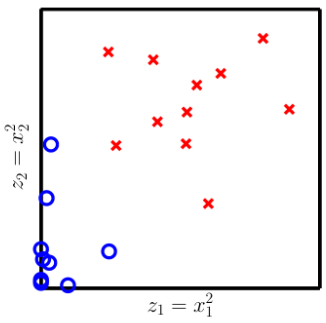


Mechanics of the Feature Transform I

Transform the data to a $\mathcal{Z}$-space in which the data is separable.



$\longrightarrow$

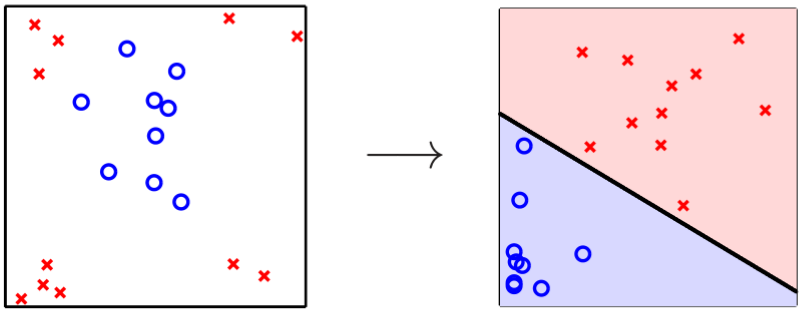


$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \longrightarrow \mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix} = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \end{bmatrix}$$

Mechanics of the Feature Transform II

Separate the data in the $\mathcal{Z}$-space with $\tilde{\mathbf{w}}$:

$$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$$



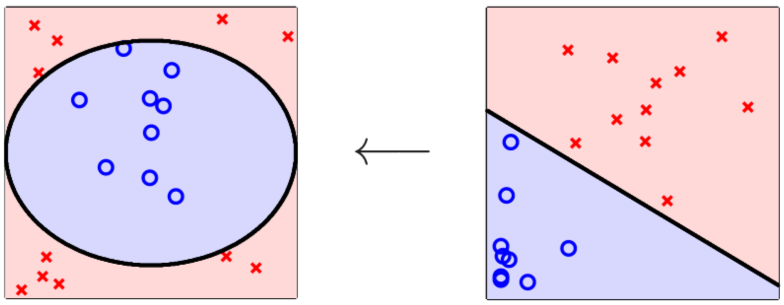
31

Mechanics of the Feature Transform III

To classify a new $\mathbf{x}$, first transform $\mathbf{x}$ to $\mathbf{\Phi}(\mathbf{x}) \in \mathcal{Z}$-space and classify there with $\tilde{g}$.

$$\begin{aligned} g(\mathbf{x}) &= \tilde{g}(\mathbf{\Phi}(\mathbf{x})) \\ &= \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{\Phi}(\mathbf{x})) \end{aligned}$$

$$\tilde{g}(\mathbf{z}) = \text{sign}(\tilde{\mathbf{w}}^{\mathrm{T}}\mathbf{z})$$



32

The General Feature Transform

$\underline{\mathcal{X}\text{-space is } \mathbb{R}^d}$

$\underline{\mathcal{Z}\text{-space is } \mathbb{R}^{\tilde{d}}}$

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_d \end{bmatrix} \qquad \mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \vdots \\ \Phi_{\tilde{d}}(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ \vdots \\ z_{\tilde{d}} \end{bmatrix}$$

$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ $\qquad$ $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N$

$y_1, y_2, \ldots, y_N$ $\qquad$ $y_1, y_2, \ldots, y_N$

no weights

$$\tilde{\mathbf{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{\tilde{d}} \end{bmatrix}$$

$g(\mathbf{x}) = \text{sign}(\tilde{\mathbf{w}}^{\text{T}}\mathbf{\Phi}(\mathbf{x}))$

33

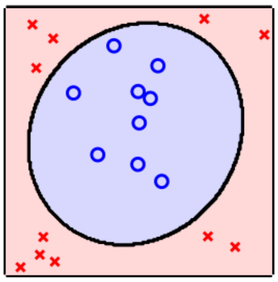
33

Must Choose Φ BEFORE Your Look at the Data

After constructing features carefully, **before** seeing the data ...

... if you think linear is not enough, try **the 2nd order polynomial transform**.

$$\begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = \mathbf{x} \longrightarrow \mathbf{\Phi}(\mathbf{x}) = \begin{bmatrix} 1 \\ \Phi_1(\mathbf{x}) \\ \Phi_2(\mathbf{x}) \\ \Phi_3(\mathbf{x}) \\ \Phi_4(\mathbf{x}) \\ \Phi_5(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1x_2 \\ x_2^2 \end{bmatrix}$$



34

34

The General Polynomial Transform Φ_k

We can get even fancier: degree-k polynomial transform:

$$\begin{aligned}
\Phi_1(\mathbf{x}) &= (1, x_1, x_2),\\
\Phi_2(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2),\\
\Phi_3(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3),\\
\Phi_4(\mathbf{x}) &= (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, x_1^2x_2^2, x_1x_2^3, x_2^4),\\
&\vdots
\end{aligned}$$

– Dimensionality of the feature space increases rapidly (d_{VC})!

– Similar transforms for d-dimensional original space.

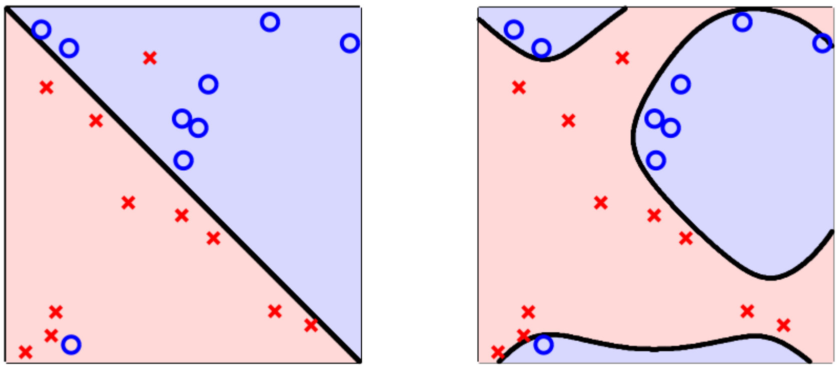
– Approximation-generalization tradeoff

Higher degree gives lower (even zero) E_{in} but worse generalization.

35

35

Be Careful with Feature Transforms



High order polynomial transform leads to “nonsense”.

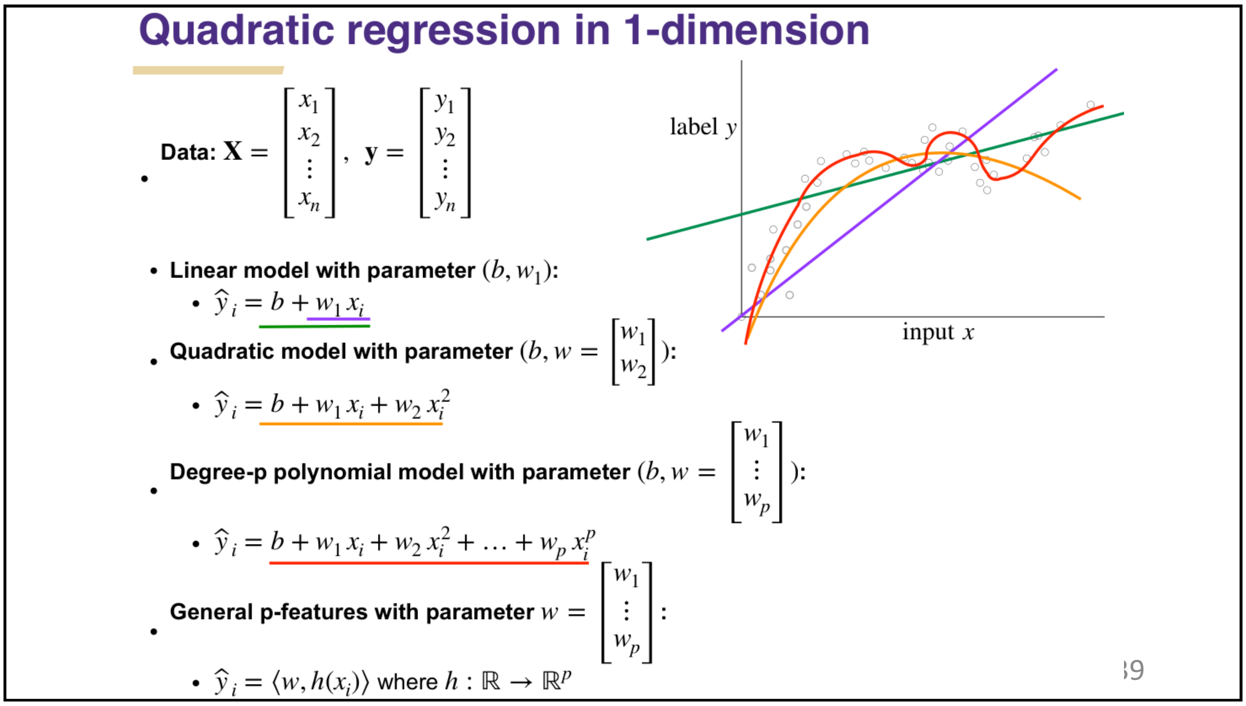
36

36

Use the Linear Model!

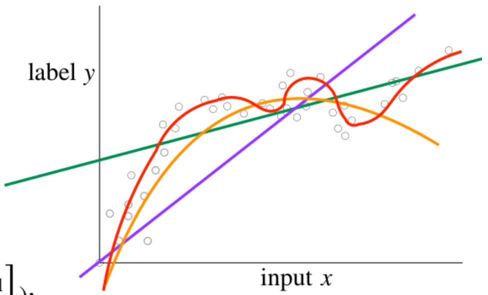
- First try a linear model – simple, robust and works.
- Algorithms can tolerate error plus you have nonlinear feature transforms.
- Choose a feature transform *before* seeing the data. Stay simple.
  Data snooping is hazardous to your E_{out}.
- Linear models are fundamental in their own right; they are also the building blocks of many more complex models like support vector machines.
- Nonlinear transforms also apply to regression and logistic regression.

Linear Regression with Basis Functions

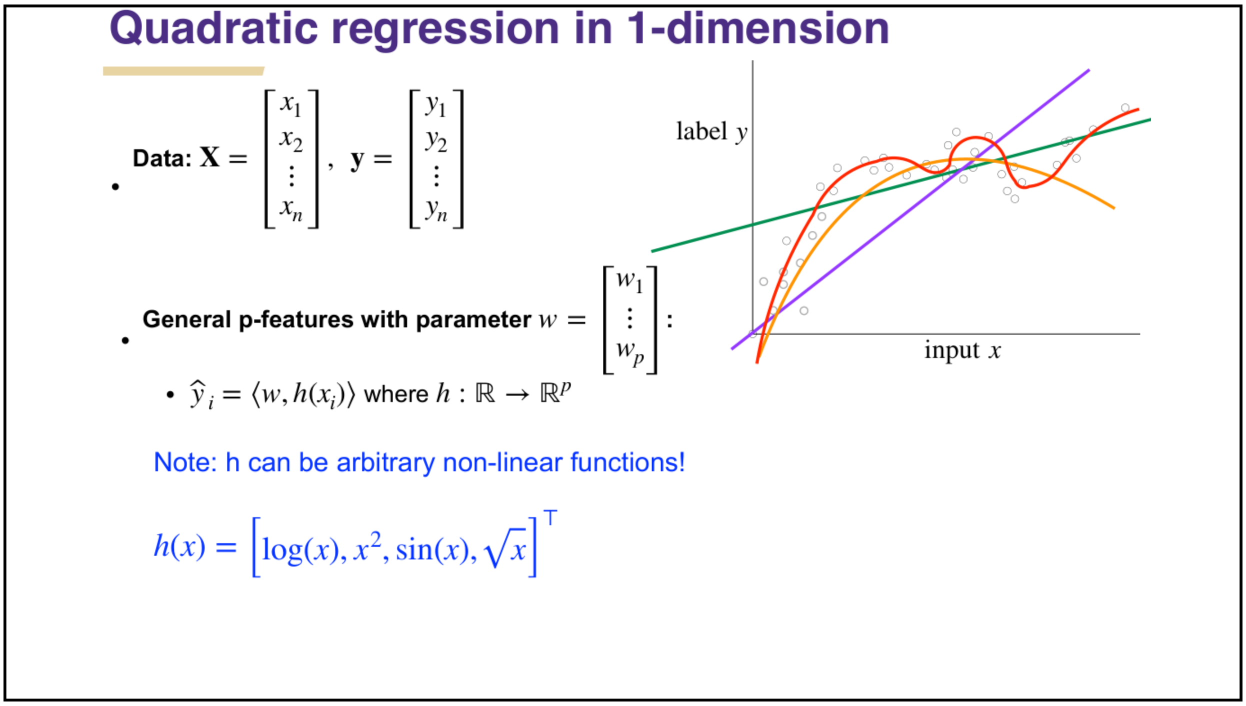


Quadratic regression in 1-dimension

- Data: $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- Linear model with parameter (b, w_1):
  - $\hat{y}_i = \underline{b + \underline{w_1 x_i}}$
- Quadratic model with parameter $(b, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$:
  - $\hat{y}_i = \underline{b + w_1 x_i + w_2 x_i^2}$
- Degree-p polynomial model with parameter $(b, w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix})$:
  - $\hat{y}_i = \underline{b + w_1 x_i + w_2 x_i^2 + \ldots + w_p x_i^p}$
- General p-features with parameter $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

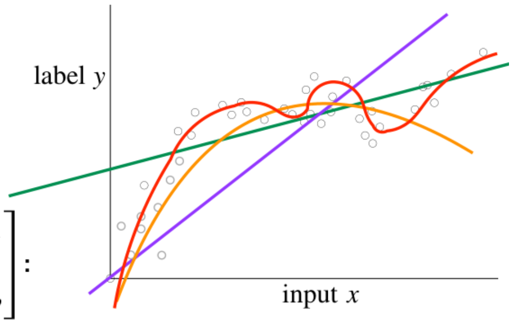


39



Quadratic regression in 1-dimension

- Data: $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- General p-features with parameter $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\hat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$



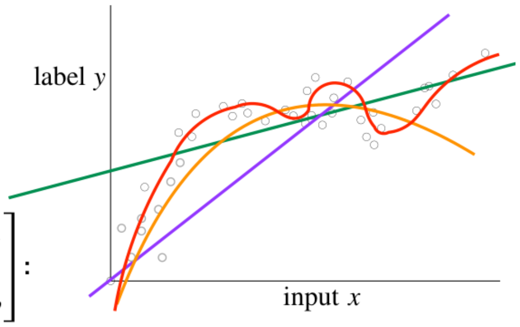
Note: h can be arbitrary non-linear functions!

$$h(x) = \left[\log(x), x^2, \sin(x), \sqrt{x}\right]^\top$$

40

Quadratic regression in 1-dimension

- Data: $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- General p-features with parameter $w = \begin{bmatrix} w_1 \\ \vdots \\ w_p \end{bmatrix}$:
  - $\widehat{y}_i = \langle w, h(x_i) \rangle$ where $h : \mathbb{R} \to \mathbb{R}^p$

How do we learn w?

$$\mathbf{H} = \begin{bmatrix} --h(x_1)^\top -- \\ \vdots \\ --h(x_n)^\top -- \end{bmatrix} \in \mathbb{R}^{n \times p}$$

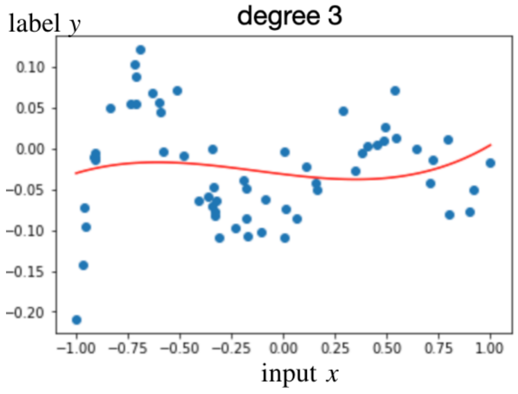
$$\widehat{w} = \arg\min_w \|\mathbf{H}w - \mathbf{y}\|_2^2$$

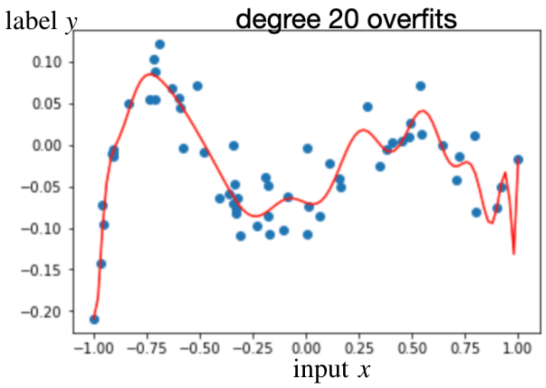
For a new test point x, predict
$\widehat{y} = \langle \widehat{w}, h(x) \rangle$

41

Which p should we choose?

- First instance of class of models with different representation power = model complexity





- How do we determine which is better model?

42

- **Recall: Feasibility of Learning** is split into two questions:
  1. Can we make sure that **Eout(g)** is close enough to **Ein(g)?**
  2. Can we make sure **Ein(g)** is small enough?

- **Generalization:**
  - We say a predictor generalizes if it performs as well on unseen data as on training data
  - The data used to train a predictor is training data or in-sample data
  - We want the predictor to work on test data or out-of-sample data
  - We say a predictor fails to generalize if it performs well on in-sample data, but, it does not perform well on out-of-sample data

For more insights read: Abu-Mostafa 2.2

43

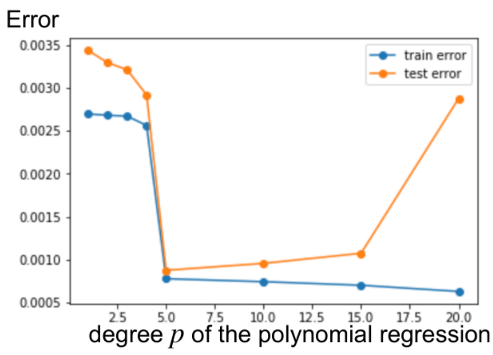
43

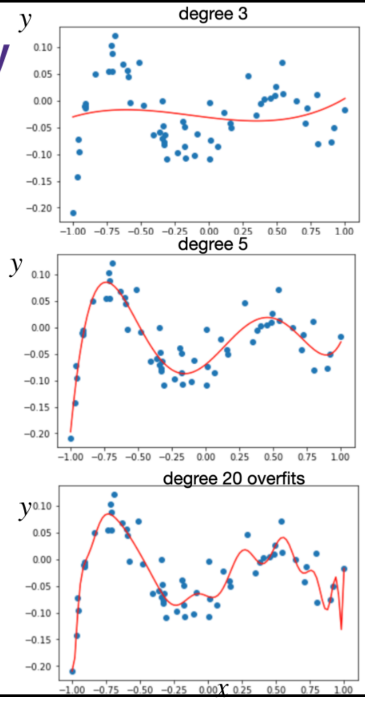
Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set (e.g., 90/10)
- **training set** used to train the model
  - minimize $\mathscr{L}_{\text{train}}(w) = \dfrac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$
- **test set** used to evaluate the model
  - $\mathscr{L}_{\text{test}}(w) = \dfrac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$
- this assumes that test set is similar to unseen data
- **test set should never be used in training or picking unknowns**

44

Train/test error vs. complexity





- Degree $p = 5$, since it achieves **minimum test error**
- **Train error** monotonically decreases with model complexity
- **Test error** has a U shape. There is a best value

test set should never be used in training or picking degree

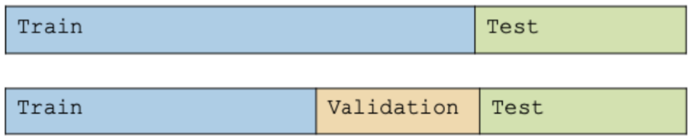
Cross Validation

How do we pick the number of basis functions?

Validation Set

Takeaway

> Simplest approach is to add another dataset partition called the **validation set**



> Each set has its own role
  > **Train:** Used to train a model of each model complexity
  > **Validation:** Used as a hold-out to evaluate each model. Generally choose model complexity with lowest validation error.
  > **Test:** Once the model is chosen, can get an estimate of future error by test. This would be what you report. **Only do this once!**

Computational cost of LOO - High

(LOO) Leave-one-out cross validation

> **Consider a validation set with 1 example:**
  - **D – training data**
  - **D\j – training data with j th data point (x_j ,y_j) moved to validation set**

> **Learn model $f_{D\backslash j}$ with $D\backslash j$ dataset**

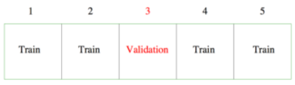
> **Estimate true error** as squared error on predicting $\mathbf{y_j}$:
  - **Unbiased estimate of error$_{true}$($f_{D\backslash j}$)!**

> **LOO cross validation**: Average over all data points *j*:
  - **For each data point you leave out, learn a new model $f_{D\backslash j}$**
  - **Estimate error** as:

$$\text{error}_{LOO} = \frac{1}{n}\sum_{j=1}^{n}(y_j - f_{\mathcal{D}\backslash j}(x_j))^2$$

Use *k*-fold cross validation

> **Randomly divide training data into *k* equal parts**
  - **$D_1,\ldots,D_k$**



> **For each *i***
  - **Learn model $f_{D\backslash D_i}$ using data point not in D_i**
  - **Estimate error of $f_{D\backslash D_i}$ on validation set D_i:**

$$\text{error}_{\mathcal{D}_i} = \frac{1}{|\mathcal{D}_i|} \sum_{(x_j,y_j)\in\mathcal{D}_i} (y_j - f_{\mathcal{D}\backslash\mathcal{D}_i}(x_j))^2$$

> **k-fold cross validation error is average** over data splits:

$$error_{k-fold} = \frac{1}{k}\sum_{i=1}^{k} error_{\mathcal{D}_i}$$

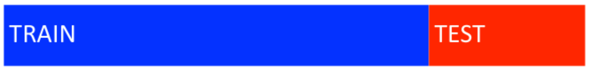
> **k-fold cross validation properties:**
  - **Much faster to compute** than LOO
  - **More (pessimistically) biased** – using much less data, only *n(k-1)/k*
  - **Usually, k = 10**

49

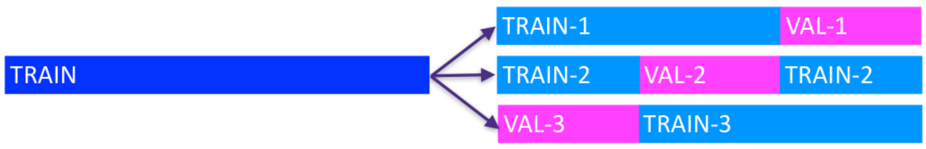
49

Recap

> **Given a dataset, begin by splitting into**



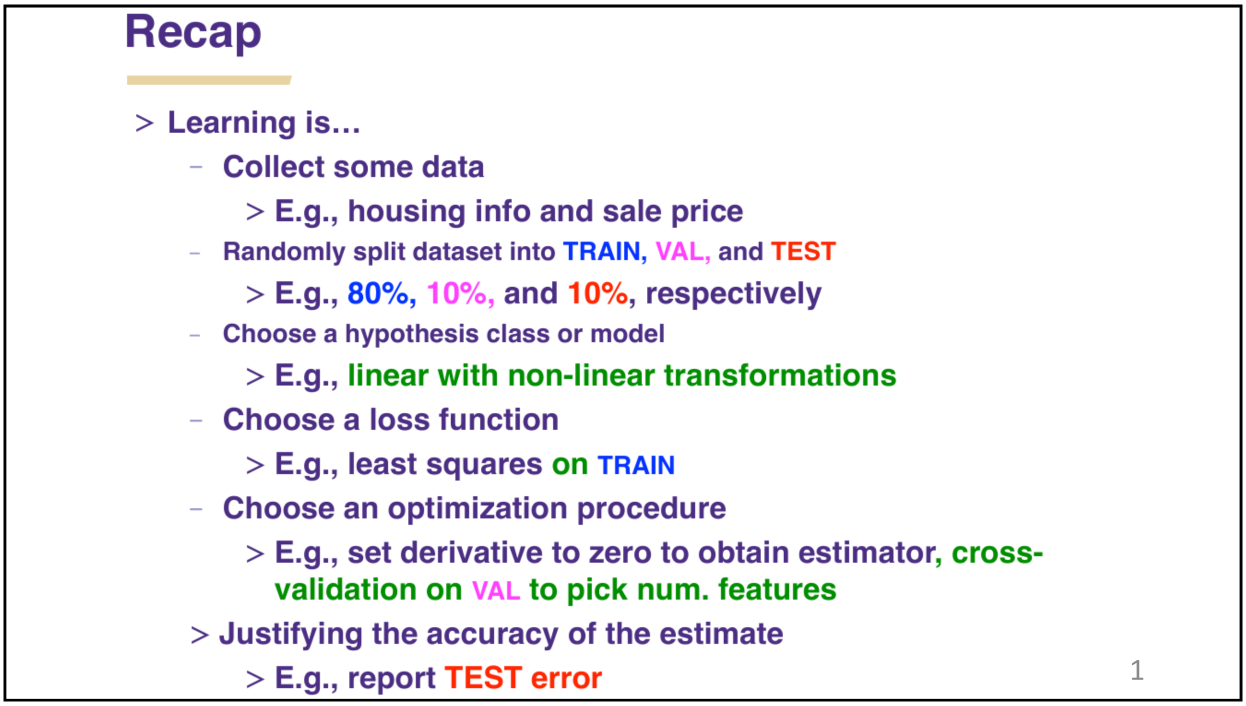
> **Model selection: Use k-fold cross-validation on TRAIN to train predictor and choose magic parameters such as degree**



> **Model assessment: Use TEST to assess the accuracy of the model you output**
  - **Never ever ever ever ever train or choose parameters based on the test data**

50

50

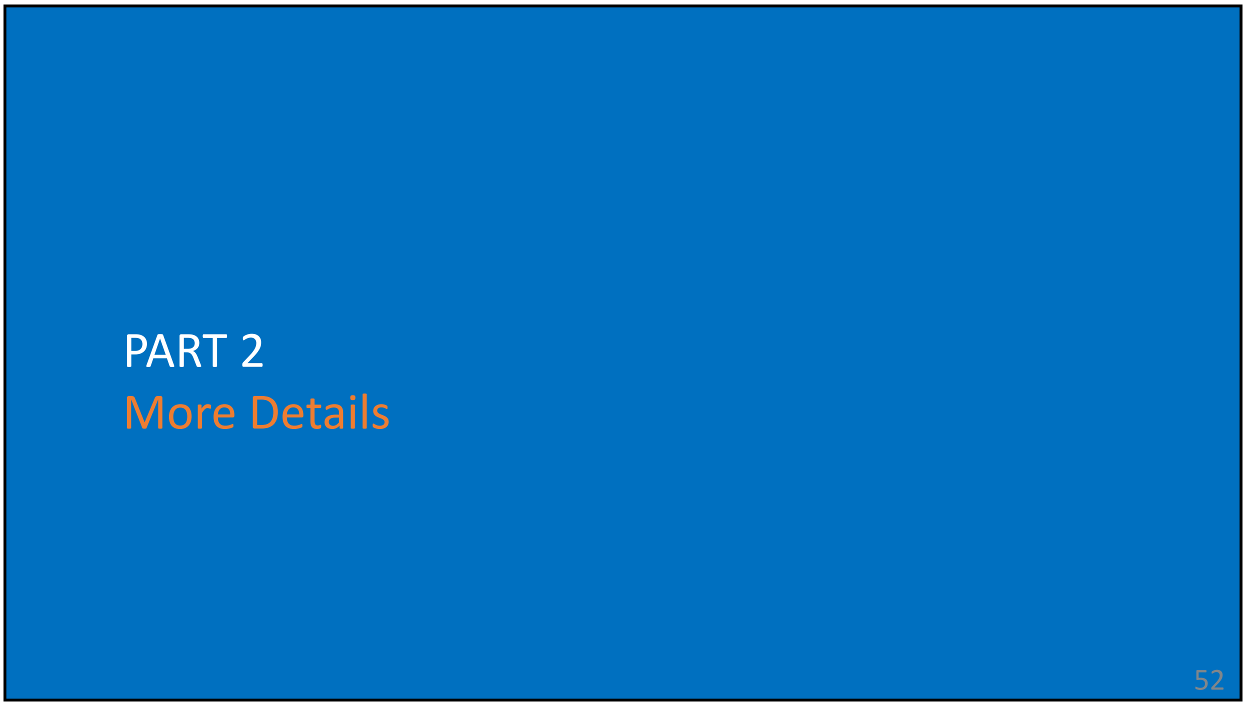


Recap

- > Learning is…
  - – Collect some data
    - > E.g., housing info and sale price
  - – Randomly split dataset into TRAIN, VAL, and TEST
    - > E.g., 80%, 10%, and 10%, respectively
  - – Choose a hypothesis class or model
    - > E.g., linear with non-linear transformations
  - – Choose a loss function
    - > E.g., least squares on TRAIN
  - – Choose an optimization procedure
    - > E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features
  - > Justifying the accuracy of the estimate
    - > E.g., report TEST error

1

51



PART 2

More Details

52

52

Details

- See hand-written notes and discussion in class.

53

53

PART 3

Code Time!

54

54

Code Time

- See demonstration and discussion in class.
- See also links in the “Assignment” for this Lecture.
- Linear regression
  - linear_regression_demo.ipynb (uwash)
- Regression with Basis Functions
  - demo_polynomial.ipynb (uwash)

55

55

Conclusion

Takeaways

56

56

Conclusion

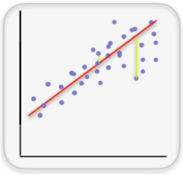
- Linear Regression
  - **Simplicity and interpretability**: Easy to understand and interpret - a starting point for learning about ML.
  - **Predictive ability**: Helps predict future outcomes based on past data - useful in many fields like finance, healthcare, marketing, etc.
  - **Basis for other models**: Many advanced algorithms, like neural networks, build on the concepts of linear regression.
  - **Efficiency**: Computationally efficient - works well for problems with a linear relationship.
  - **Widely used**: One of the most widely used techniques in both statistics and machine learning for regression tasks.
  - **Analysis**: Provides insights into relationships between variables (e.g., how much one variable influences another).

57

57

The regression problem in matrix notation

$$\begin{aligned}\widehat{w}_{LS} &= \arg\min_w ||\mathbf{y} - \mathbf{X}w||_2^2 \\ &= (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\end{aligned}$$



Dealing with an offset

$$\widehat{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\widehat{b}_{LS} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

A new house is about to be listed. What should it sell for?

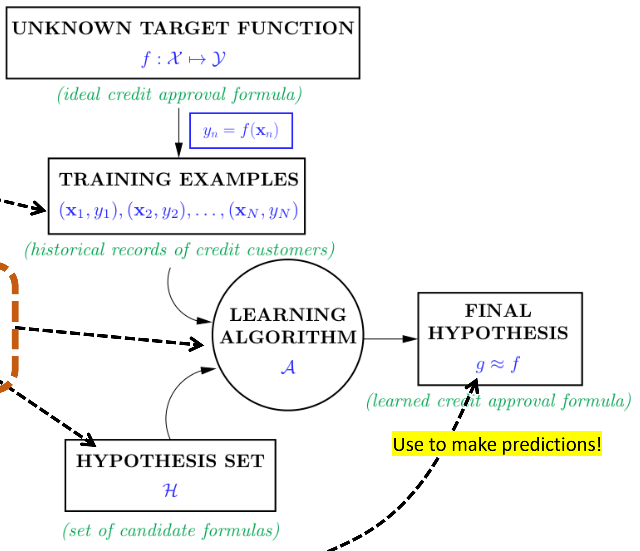
$$\hat{y}_{\text{new}} = x_{\text{new}}^T \hat{w}_{LS} + \hat{b}_{LS}$$

58

58

Learning is…

- > Learning is…
  - Collect some data
    - > E.g., housing info and sale price
  - Randomly split dataset into TRAIN, VAL, and TEST
    - > E.g., 80%, 10%, and 10%, respectively
  - Choose a hypothesis class or model
    - > E.g., linear with non-linear transformations
  - Choose a loss function
    - > E.g., least squares on TRAIN
  - Choose an optimization procedure
    - > E.g., set derivative to zero to obtain estimator, cross-validation on VAL to pick num. features
- > Justifying the accuracy of the estimate
  - > E.g., report TEST error



UNKNOWN TARGET FUNCTION
$f : \mathcal{X} \mapsto \mathcal{Y}$
(ideal credit approval formula)
$y_n = f(\mathbf{x}_n)$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(learned credit approval formula)
Use to make predictions!
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

59

References and Credits

Many of the teaching materials for this course have been adapted from various sources. We are very grateful and thank the following professors, researchers, and practitioners for sharing their teaching materials (in no particular order):

- Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Hsuan-Tien Lin. https://amlbook.com/slides.html
- Ethem Alpaydin. https://www.cmpe.boun.edu.tr/~ethem/i2ml3e/
- Natasha Jaques. https://courses.cs.washington.edu/courses/cse446/25sp/
- Lyle Ungar. https://alliance.seas.upenn.edu/~cis520/dynamic/2022/wiki/index.php?n=Lectures.Lectures
- Aurelien Geron. https://github.com/ageron/handson-ml3
- Sebastian Raschka. https://github.com/rasbt/machine-learning-book
- Trevor Hastie. https://www.statlearning.com/resources-python
- Andrew Ng. https://www.youtube.com/playlist?list=PLoROMvodv4rMiGQp3WXShtMGgzqpfVfbU
- Richard Povineli. https://www.richard.povinelli.org/teaching
- … and many others.

60

60