LINEAR MODEL

Bùi Tiến Lên

2023

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

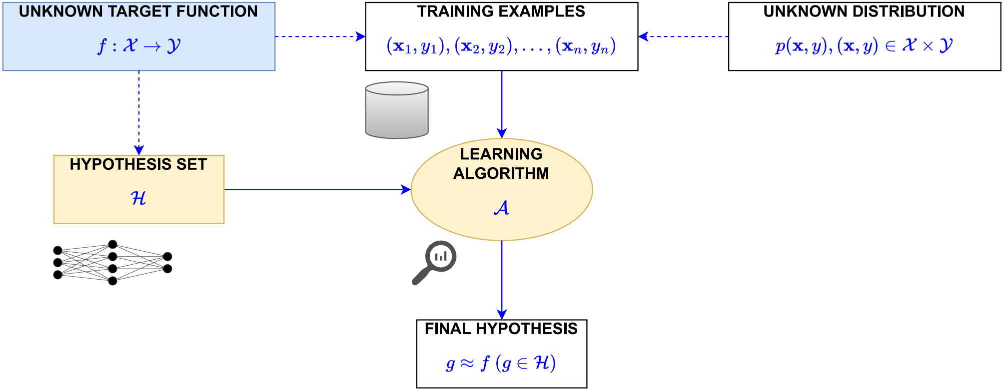
Contents

Notation

symbol	meaning
$a, b, c, N \ldots$	scalar number
$\boldsymbol{w}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y} \ldots$	column vector
$\boldsymbol{X}, \boldsymbol{Y} \ldots$	matrix
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{N}$	set of natural numbers
$\mathbb{R}^D$	set of vectors
$\mathcal{X}, \mathcal{Y}, \ldots$	set
$\mathcal{A}$	algorithm

operator	meaning
$\boldsymbol{w}^\top$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse

Learning diagram

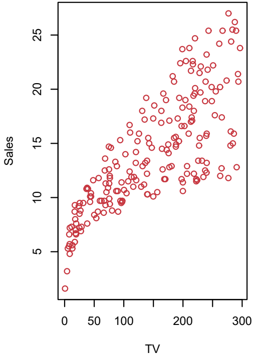


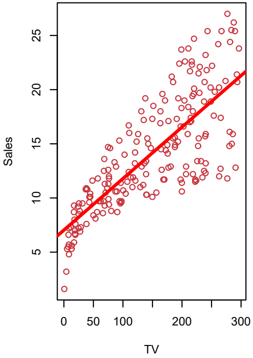
Linear Regression

- Simple Linear Model
- Weighted Linear Model
- Linear Basis Function Model

Problem 1

- Consider the Advertising data set $\mathcal{D}_{train}$ consists of the **sales** of that product in 200 different markets, along with advertising budgets for the product in each of those markets for the media **TV**. Find the *relationship* between **TV** (input) and **sales** (output)





Linear Regression Model

Concept 1

A **linear regression** is a model that assumes a **linear relationship** between **inputs** and the **output**.

Problem Statement

- **The requirement** is to build a system that can take a vector $\boldsymbol{x} \in \mathbb{R}^{D+1}$ as **input** and **predict** the value of a scalar $y \in \mathbb{R}$ as its **output**
- **The hypothesis set $\mathcal{H}$**

$$y \approx \hat{y} = h_{\boldsymbol{w}}(\mathbf{x}) = \boldsymbol{w}^\mathsf{T}\boldsymbol{x} \tag{1}$$

where $\hat{y}$ be the value that our model (function) predicts y and $\boldsymbol{w} \in \mathbb{R}^{D+1}$ is a *vector of parameters* of the model

Problem Statement (cont.)

- **Task** $\boldsymbol{T}$: to predict y from $\boldsymbol{x}$ by outputting $\hat{y} = h_{\boldsymbol{w}}(\mathrm{x}) = \boldsymbol{w}^\intercal \boldsymbol{x}$
- The train set $\mathcal{D}_{train}$ denoted as $(\boldsymbol{X}, \boldsymbol{y})$ including N samples $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots (\boldsymbol{x}_N, y_N)\}$, construct the matrix $\boldsymbol{X}$ and the vectors $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$

$$\underbrace{\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\intercal \\ \boldsymbol{x}_2^\intercal \\ \vdots \\ \boldsymbol{x}_N^\intercal \end{bmatrix}}_{\text{input data matrix}}, \quad \underbrace{\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}, \quad \underbrace{\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix}}_{\text{output vector}} \tag{2}$$

Problem Statement (cont.)

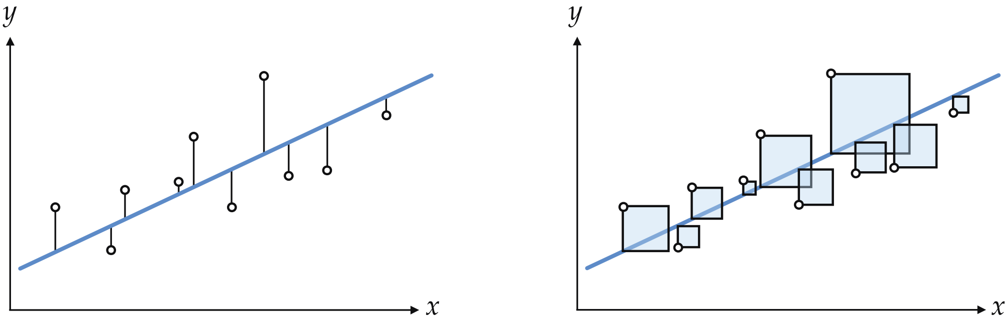
- **Performance measure P**:

Concept 2

The *mean squared error* MSE_{train} of the model on the train set $\mathcal{D}_{train}$

$$MSE_{train} = \frac{1}{N} \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2 = \frac{1}{N} \sum_{n=1}^{N} (\hat{y}_n - y_n)^2 \tag{3}$$

Problem Statement (cont.)



Problem Statement (cont.)

- **The learning goal**: find the vector of parameter $\boldsymbol{w}$ such that

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}}(MSE_{train}) \tag{4}$$

Solving Problem

Solution

- Compute the gradient of MSE_{train}

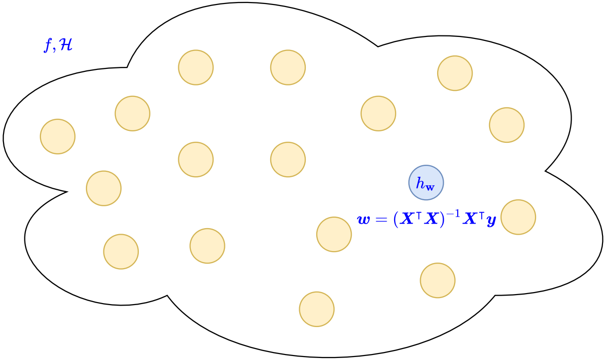
$$\begin{aligned} \nabla_{\boldsymbol{w}}(MSE_{train}) &= \nabla_{\boldsymbol{w}}(\boldsymbol{w}^\intercal\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} - \boldsymbol{w}^\intercal\boldsymbol{X}^\intercal\boldsymbol{y} - \boldsymbol{y}^\intercal\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^\intercal\boldsymbol{y}) \\ &= 2\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{X}^\intercal\boldsymbol{y} \end{aligned} \tag{5}$$

- If MSE_{train} reach the min value then $\nabla_{\boldsymbol{w}}(MSE_{train}) = 0$

$$\begin{aligned} \nabla_{\boldsymbol{w}}(MSE_{train}) &= 0 \\ \boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\intercal\boldsymbol{y} &= 0 \\ \boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{w} &= \boldsymbol{X}^\intercal\boldsymbol{y} \\ \boldsymbol{w} &= (\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}\boldsymbol{X}^\intercal\boldsymbol{y} \end{aligned} \tag{6}$$

■

Solving Problem (cont.)



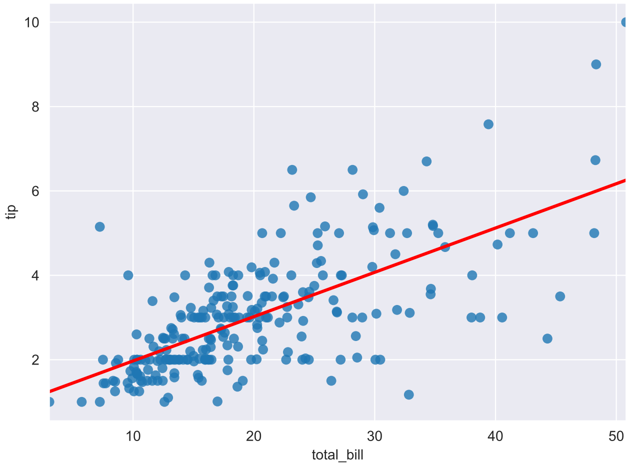
Programming Example

- Use seaborn to read tips dataset and find the linear relationship between total_bill and tip

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style("darkgrid")
tips = sns.load_dataset("tips")
sns.regplot(x="total_bill", y="tip", data=tips, ci=None, line_kws
    ={'color':'red'})
plt.show()
```

Programming Example (cont.)



Word Example

1. Find the linear regression function $y = f(x) = w_0 + w_1 x$ given the following data set $\mathcal{D}$

input x	target y
1	2
2	3
3	3
4	5

2. Find the linear regression function $y = f(\mathbf{x}) = f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$ given the following data set $\mathcal{D}$

input $\mathbf{x}$	target y
(1, 1)	1
(2, 3)	3
(3, 4)	4
(4, 3)	5

Discussion

- D is a large number
- Online learning
- Limitations of **the model** (**hypothesis set**)

Weighted Linear Model

- In some cases the observations may be weighted; for example, they may not be equally reliable. In this case, we find the vector of parameters $\boldsymbol{w}$ to minimize the weighted sum of squares of errors

$$E_{train} = \sum_{n=1}^{N} a_n(\hat{y}_n - y_n)^2 \tag{7}$$

Solving Problem

1. Construct the matrices $\boldsymbol{X}$, $\boldsymbol{A}$ and the vector $\boldsymbol{y}$

$$\underbrace{\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\intercal \\ \boldsymbol{x}_2^\intercal \\ \vdots \\ \boldsymbol{x}_N^\intercal \end{bmatrix}}_{\text{input data matrix}}, \quad \underbrace{\boldsymbol{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_N \end{bmatrix}}_{\text{weight matrix}}, \quad \underbrace{\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}} \tag{8}$$

2. Calculate the vector of parameters

$$\boldsymbol{w} = (\boldsymbol{X}^\intercal \boldsymbol{A} \boldsymbol{X})^{-1} \boldsymbol{X}^\intercal \boldsymbol{A} \boldsymbol{y} \tag{9}$$

Linear in What?

- **Linearity in the weights**

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1 + \ldots + w_D x_D \tag{10}$$

Linear Basis Function Models

Concept 3

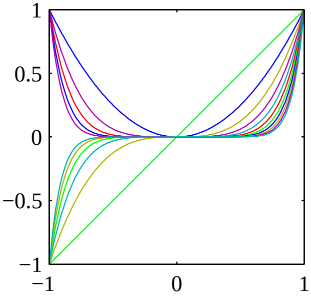
A linear basis function model is a linear combination of **fixed nonlinear functions** of the input variables

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0\phi_0(\boldsymbol{x}) + w_1\phi_1(\boldsymbol{x}) + \ldots + w_M\phi_M(\boldsymbol{x}) \tag{11}$$

where $\phi_i(\boldsymbol{x})$ are basis functions

Some types of basis functions

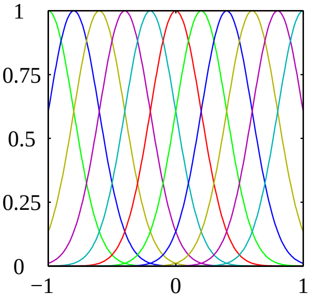
- Polynomial



$$\phi_j(x) = x^j \tag{12}$$

Some types of basis functions (cont.)

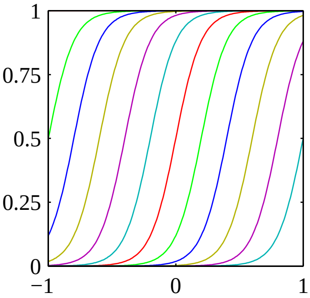
- Gaussian



$$\phi_j(x;\mu_j,s_j)=\exp\left(-\frac{(x-\mu_j)^2}{s_j^2}\right) \tag{13}$$

Some types of basis functions (cont.)

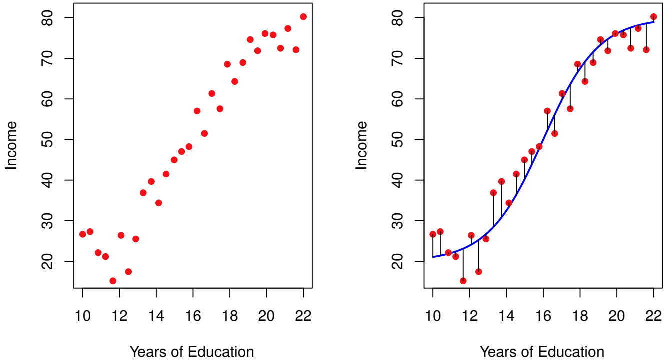
- Sigmoid



$$\phi_j(x; \mu_j, s_j) = \frac{1}{1 + e^{-\frac{x-\mu_j}{s_j}}} \tag{14}$$

Problem 2

- Find the relationship between `Years of Education` and `Income` based on the given data



Problem Statement

- **The requirement** is to build a system that can take a vector $\boldsymbol{x} \in \mathbb{R}^D$ as **input** and **predict** the value of a scalar $y \in \mathbb{R}$ as its **output**
- The hypothesis set $\mathcal{H}$

$$y \approx \hat{y} = h_{\boldsymbol{w}}(\mathbf{x}) = \boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}) \tag{15}$$

where $\hat{y}$ be the value that our model (function) predicts y, $\boldsymbol{w} \in \mathbb{R}^{M+1}$ is a *vector of parameters* of the model and $\boldsymbol{\phi}$ is a set of $M+1$ basis functions

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \phi_1(\boldsymbol{x}) \\ \vdots \\ \phi_M(\boldsymbol{x}) \end{bmatrix} \tag{16}$$

Problem Statement (cont.)

- **Task T**: to predict y from $\boldsymbol{x}$ by outputting $\hat{y} = h_{\boldsymbol{w}}(\mathrm{x}) = \boldsymbol{w}^\intercal \phi(\boldsymbol{x})$
- **Performance measure P**:
  The *mean squared error* MSE_{train} of the model on the train set $\mathcal{D}_{train}$ including N samples $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \dots (\boldsymbol{x}_N, y_N)\}$
- **The learning goal**: find the vector of parameter $\boldsymbol{w}$ such that

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}}(MSE_{train})$$

Solving Problem

1. Construct the matrix $\mathbf{\Phi}$ and the vector $\boldsymbol{y}$

$$\mathbf{\Phi} = \underbrace{\begin{bmatrix} \phi(\boldsymbol{x}_1)^\intercal \\ \phi(\boldsymbol{x}_2)^\intercal \\ \vdots \\ \phi(\boldsymbol{x}_N)^\intercal \end{bmatrix}}_{\text{design matrix}}, \quad \boldsymbol{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}} \tag{17}$$

2. Calculate the vector of parameters

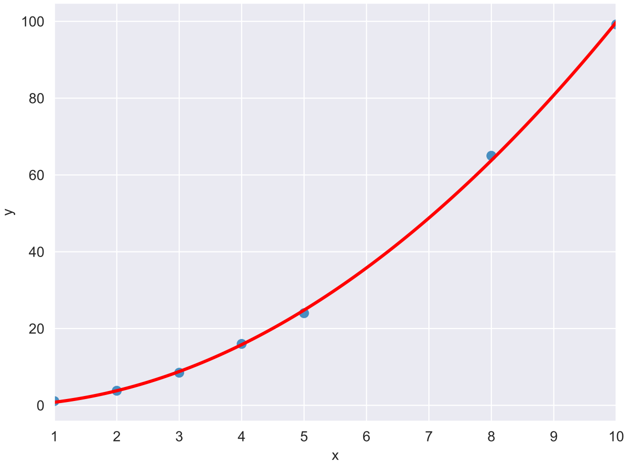
$$\boldsymbol{w} = (\mathbf{\Phi}^\intercal \mathbf{\Phi})^{-1} \mathbf{\Phi}^\intercal \boldsymbol{y} \tag{18}$$

Programming Example

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

sns.set_style("darkgrid")
x = [1, 2, 3, 4, 5, 8, 10]
y = [1.1, 3.8, 8.5, 16, 24, 65, 99.2]
sns.regplot(x, y, order=2, ci=None, line_kws={'color':'red'})
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```

Programming Example (cont.)



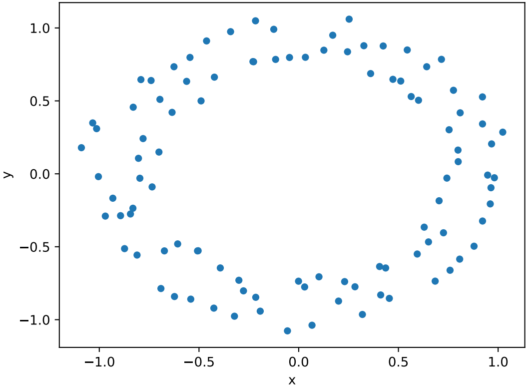
Word Example

- Find a polynomial regression function $y = f(x) = w_0 + w_1 x + w_2 x^2$ given the data set $\mathcal{D}$

input x	target y
1	2
2	3
3	3
4	5

Puzzle

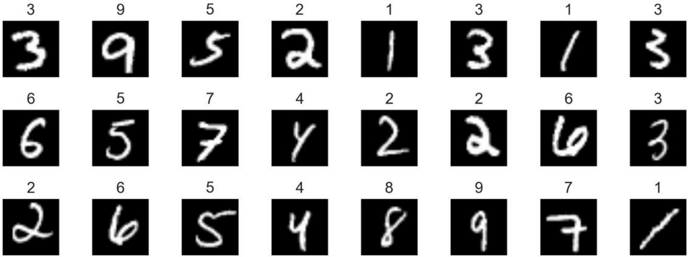
- What basis functions?



Classification

A real data set

- Some 16-by-16 pixel grayscale image from the MNIST database



Input representation

Input representation or **feature extraction**

- **“raw” input**

  pixels $\boldsymbol{x}^\mathsf{T} = \begin{pmatrix} x_0 & x_1 & \dots & x_{256} \end{pmatrix}$

  linear model $\boldsymbol{w}^\mathsf{T} = \begin{pmatrix} w_0 & w_1 & \dots & w_{256} \end{pmatrix}$

- **Feature extraction**: extract useful information

  intensity and symmetry $\boldsymbol{x}^\mathsf{T} = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$

  linear model $\boldsymbol{w}^\mathsf{T} = \begin{pmatrix} w_1 & w_2 \end{pmatrix}$

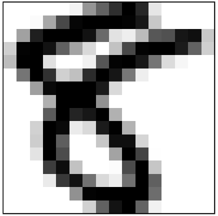
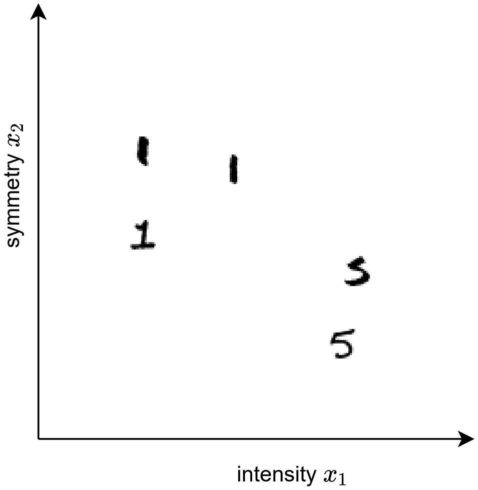


Illustration of Features



The Problem with Categorical Data

- Some algorithms can work with categorical data directly.
  - For example, a decision tree can be learned directly from categorical data with no data transform required
- Many machine learning algorithms cannot operate on label data directly.
  - They require all input variables and output variables to be numeric.
- There are two common types of conversion: integer encoding and one-hot encoding

Integer Encoding

- For categorical variables where no such ordinal relationship exists, the integer encoding is not enough.

id	color
1	red
2	green
3	blue
4	red
...	...

$\xrightarrow{\text{Integer encoding}}$

id	color
1	1
2	2
3	3
4	1
...	...

One-Hot Encoding

- One-hot encoding ensures that machine learning does not assume that higher numbers are more important.

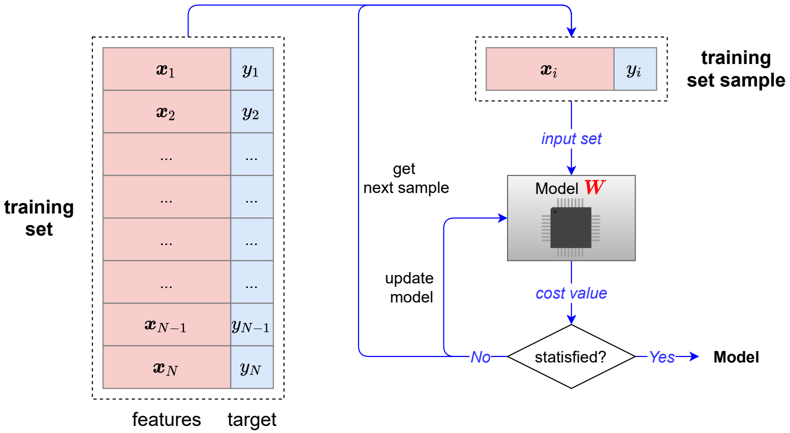
id	color
1	red
2	green
3	blue
4	red
…	…

One-hot encoding →

id	color_red	color_green	color_blue
1	1	0	0
2	0	1	0
3	0	0	1
4	1	0	0
…	…	…	…

Classifier Training

- **Select** the learning model for **classifier**, e.g., Perceptron
- **Train** the classifier/model using a training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$



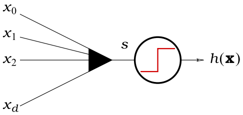
Logistic Regression

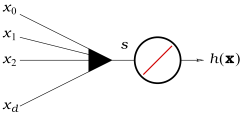
- Binary Classification
- Evaluation
- Multi-class Classification

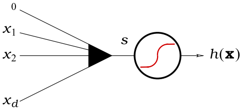
A Third Linear Model

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

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = sign(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \sigma(s)$







The logistic function

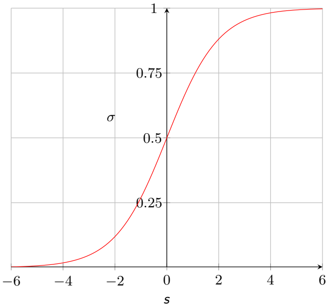
- The formula

$$\sigma(s) = \frac{1}{1+e^{-s}} \tag{19}$$

- The **logistic function** converts a score to a probability
- Properties

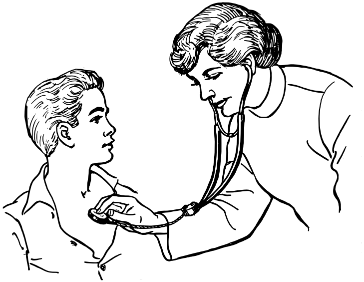
$$\sigma(-s) = 1 - \sigma(s)$$

$$\sigma'(s) = \sigma(s)(1-\sigma(s))$$



Probability Interpretation

- $h(\boldsymbol{x}) = \sigma(s)$ can be interpreted as a probability
- For example, prediction of heart attacks
  - Input $\boldsymbol{x}$: cholesterol level, age, weight, etc.
  - The signal $s = \boldsymbol{w}^T\boldsymbol{x}$: risk score
  - $\sigma(s)$: probability of a heart attack



score $\longrightarrow$ probability of heart attack

Problem Statement

- The target function f is the probability distribution

$$f : \mathbb{R}^D \to [0, 1]$$

- Hypothesis set $h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T \boldsymbol{x})$ and the conditional probability

$$P(y \mid \boldsymbol{x}, \boldsymbol{w}) = \begin{cases} h_{\boldsymbol{w}}(\boldsymbol{x}) & \text{for} \quad y = 1 \\ 1 - h_{\boldsymbol{w}}(\boldsymbol{x}) & \text{for} \quad y = 0 \end{cases} \tag{20}$$

Error measure

We define error measurement based on likelihood

- For each $(\boldsymbol{x}, y)$, y is generated by probability $h_{\boldsymbol{w}}(\boldsymbol{x})$.
- Plausible error measure based on **likelihood** of y given $\boldsymbol{x}$ and $\boldsymbol{w}$

$$P(y \mid \boldsymbol{x}, \boldsymbol{w}) = z^y (1 - z)^{1-y} \tag{21}$$

where

$$z = h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^\intercal \boldsymbol{x}) \tag{22}$$

- Likelihood of $\mathcal{D} = \{(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_N, y_N)\}$ given $\boldsymbol{w}$ is

$$\prod_{n=1}^{N} P(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) \tag{23}$$

Error measure (cont.)

- **Learning goal**: maximizing likelihood

$$\begin{aligned} & \text{Maximize} && \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n, \mathbf{w}) \\ \Leftrightarrow\ & \text{Minimize} && -\log \prod_{n=1}^{N} P(y_n \mid \mathbf{x}_n, \mathbf{w}) \\ \Leftrightarrow\ & \text{Minimize} && -\sum_{n=1}^{N} (y_n \log z_n + (1 - y_n) \log(1 - z_n)) \end{aligned} \tag{24}$$

Learning Algorithm (Gradient Descent)

1. Initialize the weights (parameters) at $t = 0$ $\boldsymbol{w}_0$
2. For $t = 1, 2, 3, \ldots$ do

   2.1 Compute the outputs z_n for each $\boldsymbol{x}_n$ $(n = 1, \ldots, N)$

   $$z_n = \sigma(\boldsymbol{w}_t^T \boldsymbol{x}_n) \tag{25}$$

   2.2 Compute the gradient

   $$\nabla_{\boldsymbol{w}} E = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n (z_n - y_n) \tag{26}$$

   2.3 Update the weights

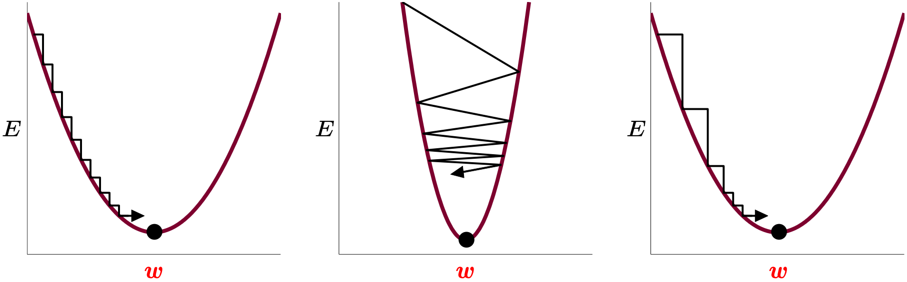
   $$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla_{\boldsymbol{w}} E \tag{27}$$

   where η is a learning rate (*hyper-parameter*)

   Iterate the next step until $\boldsymbol{w}$ is not changes
3. Return the final weights $\boldsymbol{w}$

Learning Rate

- How η affects the algorithm?



η too small η too large η right

Programming Example

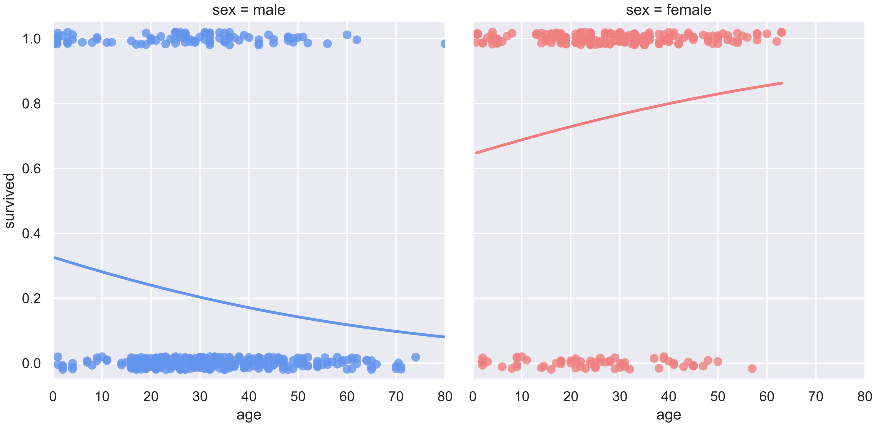
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="darkgrid")

# Load the example titanic dataset
df = sns.load_dataset("titanic")

# Make a custom palette with gendered colors
pal = dict(male="#6495ED", female="#F08080")

# Show the survival proabability as a function of age and sex
g = sns.lmplot(x="age", y="survived", col="sex", hue="sex", data=df,
    palette=pal, y_jitter=.02, logistic=True, ci=None)
g.set(xlim=(0, 80), ylim=(-.05, 1.05))
plt.show()
```

Programming Example (cont.)



Evaluation

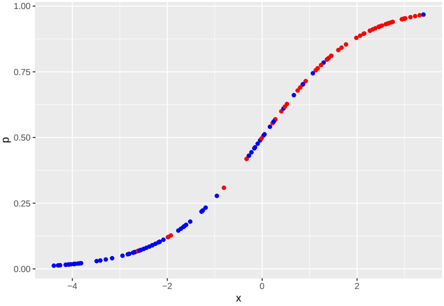
- Consider two-class problem with two classes $\oplus$ and $\ominus$
- The performance of logistic regression model is based on a threshold *th*

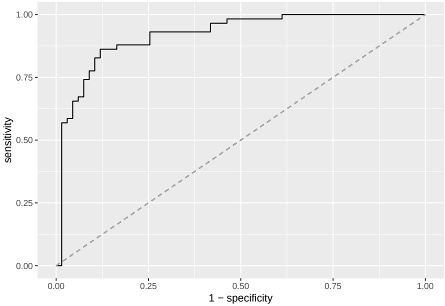
$$\begin{aligned} y \text{ } is \oplus \text{ } if \; P(y \mid \boldsymbol{x}) &\geq th \\ y \text{ } is \ominus \text{ } if \; P(y \mid \boldsymbol{x}) &< th \end{aligned} \tag{28}$$

  - High threshold: high specificity, low sensitivity
  - Low threshold: low specificity, high sensitivity
- We should select the best threshold for the trade-off between the cost of *false positives* vs *false negatives*

ROC Curve

- The receiver operating characteristic (ROC) curve is plot which shows the performance of a binary classifier as function of its cut-off threshold.
- It essentially shows the *true positive rate* (*sensitivity*) against the *false positive rate* (1-*specificity*) for various threshold values.
- The area under the curve (AUC) is an aggregated measure of performance.

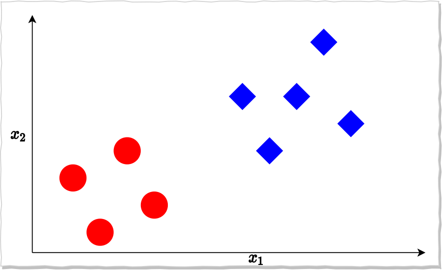


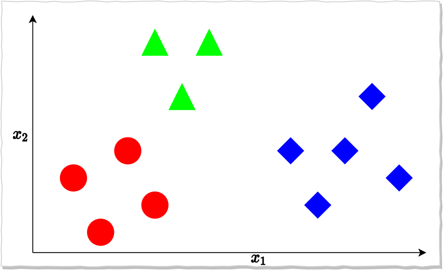


Multi-class classification problems

- **Email foldering/tagging**: Work (1), Friends (2), Family (3), Hobby (4)
- **Medical diagrams**: Not ill, Cold, Flu
- **Weather**: Sunny, Cloudy, Rain, Snow

Visual of Binary vs Multi-class classification

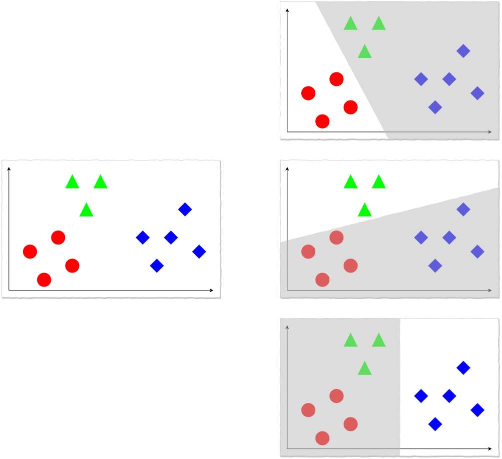




Approaches

- One-vs-one
- Hierarchical
- **One-vs-all**

One-vs-all



- Class ○ (1):

$$h^{(1)}(\boldsymbol{x}) = P(y = 1 \mid \boldsymbol{x}, \boldsymbol{w}_1)$$

- Class △ (2):

$$h^{(2)}(\boldsymbol{x}) = P(y = 2 \mid \boldsymbol{x}, \boldsymbol{w}_2)$$

- Class ♢ (3):

$$h^{(3)}(\boldsymbol{x}) = P(y = 3 \mid \boldsymbol{x}, \boldsymbol{w}_3)$$

One-vs-all (cont.)

Learning

- Train a logistic regression classifier $h^{(i)}(\boldsymbol{x})$ for each class i to predict the probability that $y = i$.

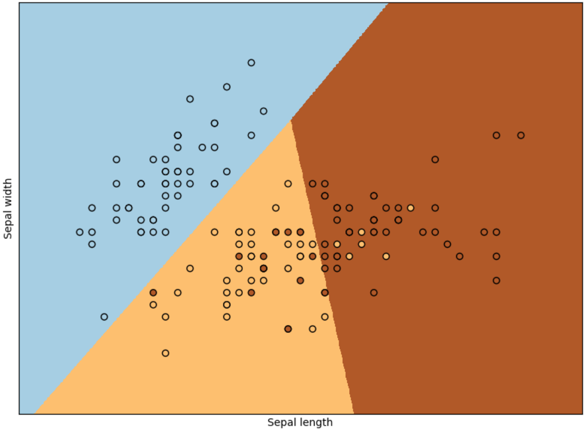
Prediction

- On a new input $\boldsymbol{x}$, to make a prediction, pick the class i that maximizes

$$\arg\max_i (h^{(i)}(\boldsymbol{x})) \tag{29}$$

Decision boundaries and decision regions

- Logistic regression for **Iris** dataset



Softmax Regression

- Softmax Regression
- Cross Entropy vs. MSE

Softmax Regression

Concept 4

Softmax regression is a generalization of logistic regression that we can use for multi-class classification

- In Softmax regression, we replace the sigmoid function by the so-called softmax function $\phi(\cdot) = \{\phi_1, ..., \phi_C\}$.

Score function

Concept 5

The **score function** f that maps the raw features to class scores.

$$\boldsymbol{z} = f(\boldsymbol{x}; W, \boldsymbol{b}) = W\boldsymbol{x} + \boldsymbol{b} \tag{30}$$

input image

$$\begin{array}{ccccccccc} W & & \boldsymbol{x} & & \boldsymbol{b} & & \boldsymbol{z} & \\ \begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix} & \times & \begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix} & + & \begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix} & = & \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} & \begin{array}{l} \text{cat score} \\ \text{dog score} \\ \text{ship score} \end{array} \end{array}$$

Score function (cont.)

- Use bias trick $W_0 = \boldsymbol{b}$ to represent the two parameters $W, \boldsymbol{b}$ as one

$$\boldsymbol{z} = f(\boldsymbol{x}; W) = W\boldsymbol{x} \tag{31}$$

$$\underset{W}{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}} \times \underset{\boldsymbol{x}}{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}} + \underset{\boldsymbol{b}}{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}} \Leftrightarrow \underset{\text{new } W}{\begin{bmatrix} 1.1 & 0.2 & -0.5 & 0.1 & 2.0 \\ 3.2 & 1.5 & 1.3 & 2.1 & 0.0 \\ -1.2 & 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}} \times \underset{\text{new } \boldsymbol{x}}{\begin{bmatrix} 1 \\ 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}$$

Softmax function

Concept 6

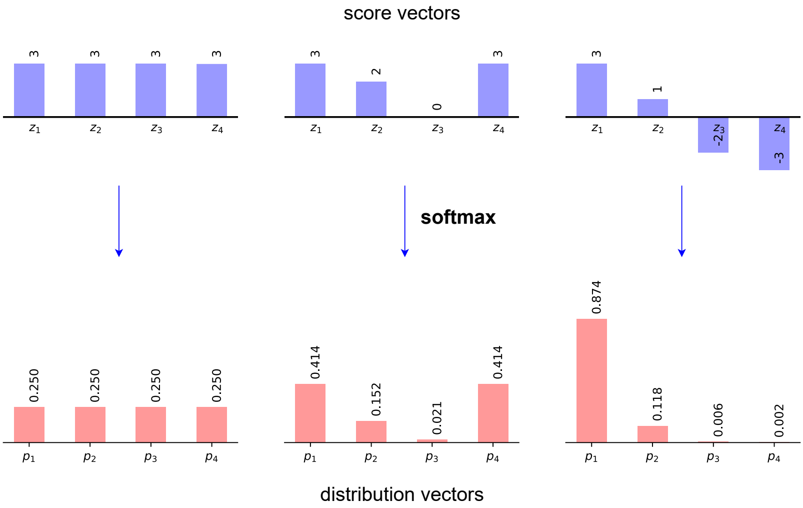
The **softmax function** converts a score vector $\boldsymbol{z} = (z_1, ..., z_C)$ to a discrete distribution vector $\boldsymbol{p} = (p_1, ..., p_C)$

$$p_i = P(y = i \mid \boldsymbol{z}) = \phi_i(\boldsymbol{z}) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i \in [1, ..., C] \tag{32}$$

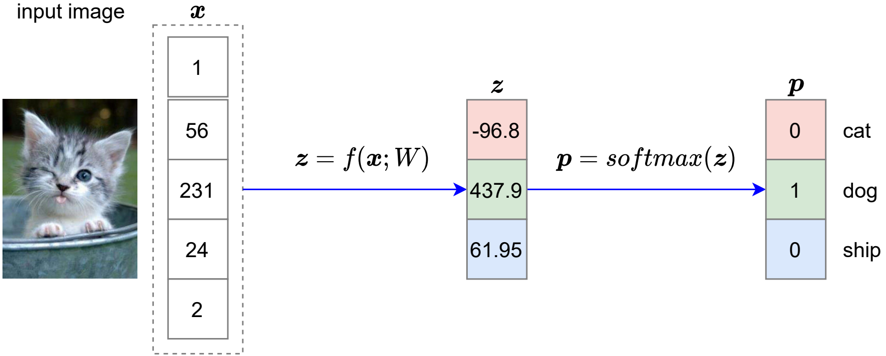
where

$$z_i = w_{i0} + w_{i1}x_1 + ... + w_{iD}x_D = \boldsymbol{w}_i^T \boldsymbol{x} \tag{33}$$

Softmax function (cont.)



Softmax function (cont.)



Problem Statement

- Given $\mathcal{D} = \{(\boldsymbol{x}_1, y_1) \ldots (\boldsymbol{x}_N, y_N)\}$ where $y_n \in \{1, \ldots, C\}$.
- Denote $\boldsymbol{t}_n$ is a **one-hot encoding** of y_n (or target discrete distribution)
- **Learning goal**: Find a softmax function $\phi_W(\cdot) = \{\phi_1, \ldots, \phi_C\}$ that minimize

$$\arg\min_{W} E(\phi_W) = \arg\min_{W} \sum_{n=1}^{N} CE(\boldsymbol{p}_n, \boldsymbol{t}_n) \tag{34}$$

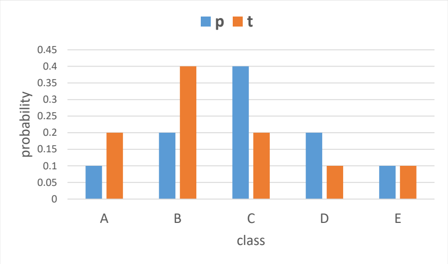
Cross Entropy

Concept 7

Cross-entropy (CE) is a measure of the difference between two probability distributions. The cross-entropy between a "true" distribution $\boldsymbol{t} = (t_1, ..., t_C)$ and an estimated distribution $\boldsymbol{p} = (p_1, ..., p_C)$ is defined as

$$CE(\boldsymbol{p}, \boldsymbol{t}) = -\sum_{i=1}^{C} t_i \log p_i \tag{35}$$

Cross Entropy (cont.)



$$\left.\begin{aligned} \boldsymbol{p} &= (0.1, 0.2, 0.4, 0.2, 0.1) \\ \boldsymbol{t} &= (0.2, 0.4, 0.2, 0.1, 0.1) \end{aligned}\right\} \rightarrow CE(\boldsymbol{p}, \boldsymbol{t}) = 1.678$$

- $CE > 0$
- $CE(\boldsymbol{p}, \boldsymbol{t}) \neq CE(\boldsymbol{t}, \boldsymbol{p})$
- CE minimize if $p_i = t_i, \forall i$

MSE

Concept 8

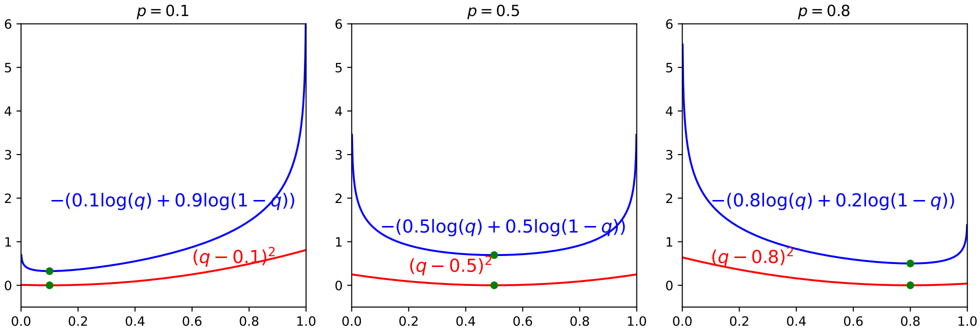
Mean squared error (MSE) is a measure of the average of the squares of the errors.

$$MSE(\boldsymbol{p}, \boldsymbol{t}) = \frac{1}{C}\sum_{i=1}^{C}(p_i - t_i)^2 \tag{36}$$

- $MSE \geq 0$
- $MSE(\boldsymbol{p}, \boldsymbol{t}) = MSE(\boldsymbol{t}, \boldsymbol{p})$
- $MSE = 0$ if $p_i = t_i, \forall i$

Cross Entropy vs. MSE

- Consider three "true" binary distributions $\boldsymbol{p} = (0.1, 0.9), (0.5, 0.5)$ and $(0.8, 0.2)$



Learning Algorithm

1. Initialize the weights W_0 (parameters) at $t = 0$
2. For $t = 1, 2, 3, \ldots$ do

   2.1 Compute the ouput distribution $\boldsymbol{p}_n$ for each $\boldsymbol{x}_n$ ($n = 1 \ldots N$)

   $$\boldsymbol{p}_n = softmax(W_t \boldsymbol{x}_n) \tag{37}$$

   2.2 Compute the gradient

   $$\nabla_W E = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{p}_n - \boldsymbol{t}_n) \boldsymbol{x}_n^\intercal \tag{38}$$

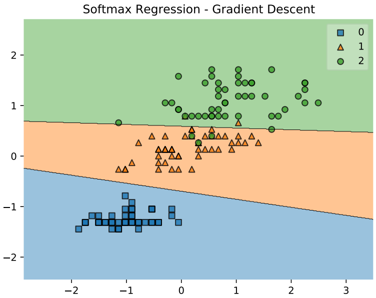
   2.3 Update the weights

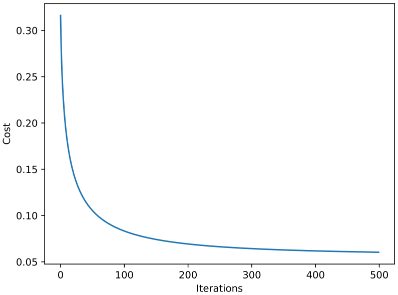
   $$W_{t+1} = W_t - \eta \nabla_W E \tag{39}$$

   Iterate the next step until W is not change
3. Return the final weights W

Example

- Softmax regression for **Iris** dataset





Capacity, Overfitting and Underfitting

- Model Capacity
- Model vs. Data
- Bias-Variance
- Tradeoff of Capacity
- Regularization
- Tradeoff of Regularization

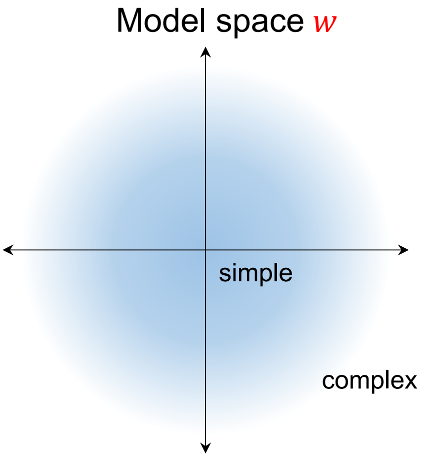
Model Capacity

Concept 9

Capacity is model complexity.

The most common ways to estimate the capacity of a model:

- VC dimension
- The number of parameters
- The norm of parameters



Model vs. Data

Concept 10

Data is divided into three sets: **training set**, **validation set** and **test set**.

Concept 11

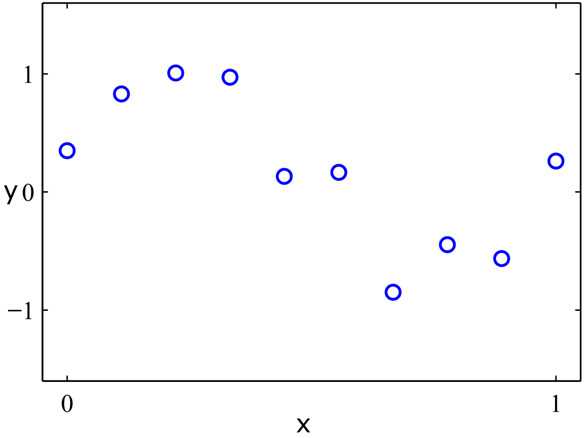
Models can be too limited. We can't find a function that fits the data well. This is called **underfitting**.

Concept 12

Models can also be too rich. We don't just model the data, but also the underlying noise. This is called **overfitting**.

Model vs. Data (cont.)

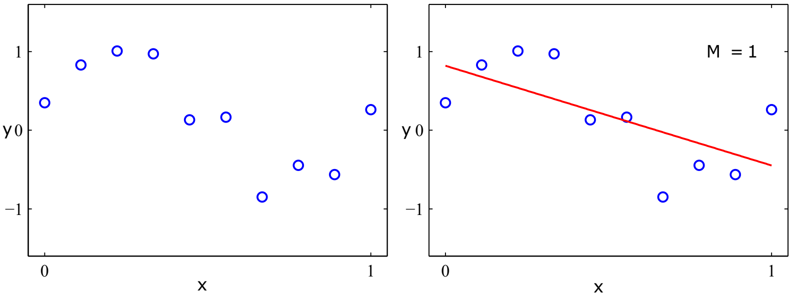
- Given the data set $\mathcal{D} = \{(x_1, y_1), ...(x_{10}, y_{10})\}$ shown in the following figure, find the best regression function to the data



Model vs. Data (cont.)

- Consider the simple hypothesis set

$$\mathcal{H}_1 = \{h \mid y = h(x) = w_0 + w_1 x\} \tag{40}$$

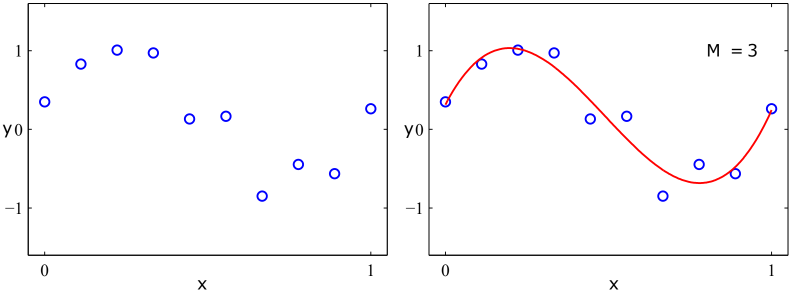


Model vs. Data (cont.)

- Consider the hypothesis set

$$\mathcal{H}_3 = \{h \mid y = h(x) = w_0 + w_1 x + w_2 x^2 + w_3 x^3\} \tag{41}$$

(note that $\mathcal{H}_1 \subset \mathcal{H}_3$)

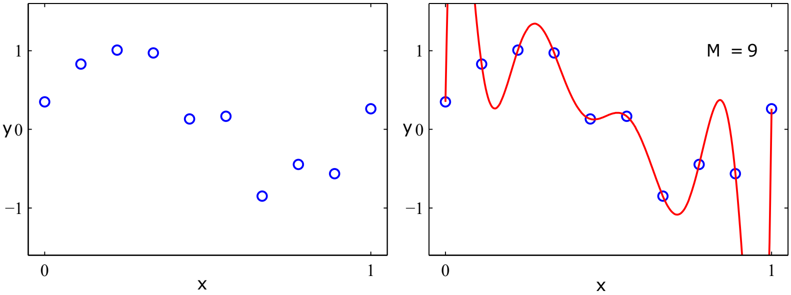


Model vs. Data (cont.)

- Consider the hypothesis set

$$\mathcal{H}_9 = \{h \mid y = h(x) = w_0 + w_1 x + w_2 x^2 + \ldots + w_9 x^9\} \tag{42}$$

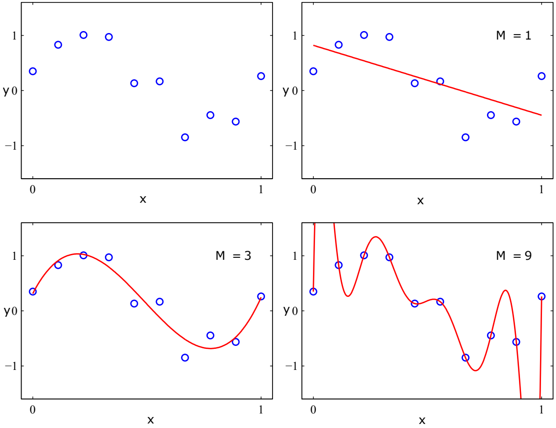
(note that $\mathcal{H}_1 \subset \mathcal{H}_3 \subset \mathcal{H}_9$)



Model vs. Data (cont.)

Which one

- Under-fitting
- Over-fitting
- **Appropriate fitting**



Model vs. Data (cont.)

	$M = 1$	$M = 3$	$M = 9$
w_0	0.82	0.31	0.35
w_1	-1.27	7.99	232.37
w_2		-25.43	-5321.83
w_3		17.37	48568.31
w_4			-231639.30
w_5			640042.26
w_6			-1061800.52
w_7			1042400.18
w_8			-557682.99
w_9			125201.43

Model Performance

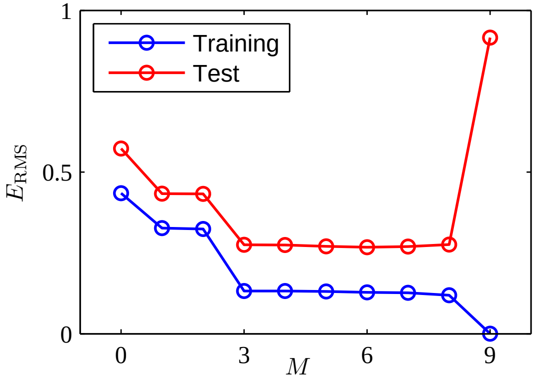


Figure 1: Graphs of the root-mean-square error evaluated on the **training set** and on an **independent test set** for various values of M

What happen if increasing N

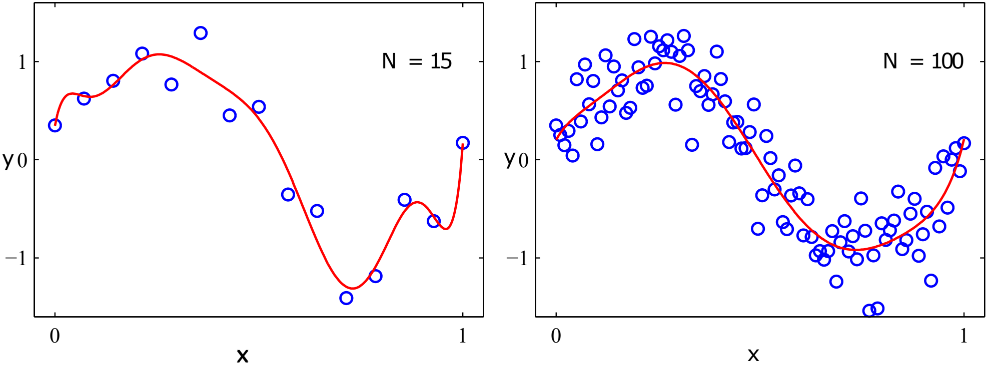


Figure 2: Using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem

Errors in Learning Model

- **Bias errors**: error due to assumption in the model
  - High bias to signify underfitting
- **Variance errors**: It measures the variability in the results given by model when the dataset is changed
  - High variance to signify overfitting

$$\text{Expected error} = \text{Bias} + \text{Variance} + \text{Irreducible Error} \tag{43}$$

Errors in Learning Model (cont.)

Concept 13

Given a learning model $\langle \mathcal{H}, \mathcal{A} \rangle$, we define the "average" hypothesis $\overline{g}(x)$

$$\overline{g}(x) = \mathbb{E}_{\mathcal{D}} \left[g_{\mathcal{D}}(x) \right] \tag{44}$$

where $g_{\mathcal{D}}(x)$ is the "**best**" hypothesis given the data set $\mathcal{D}$

- Bias of learning model

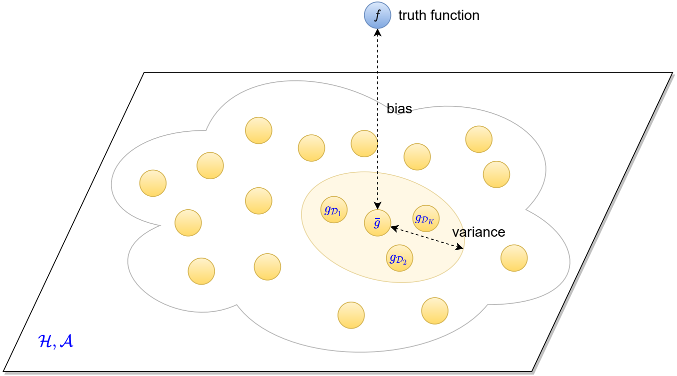
$$\text{Bias} = \mathbb{E}_x \left[\left(\overline{g}(x) - f(x) \right)^2 \right] \tag{45}$$

  where $f(x)$ is the "**truth**" function

- Variance of learning model

$$\text{Variance} = \mathbb{E}_x \left[\mathbb{E}_{\mathcal{D}} \left[\left(g_{\mathcal{D}}(x) - \overline{g}(x) \right)^2 \right] \right] \tag{46}$$

Errors in Learning Model (cont.)



- Given many independent data sets $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$, we can estimate $\overline{g}(x)$ by

$$\overline{g}(x) \approx \frac{1}{K} \sum_{i=1}^{K} g_{\mathcal{D}_i}(x) \tag{47}$$

Example: two learning models

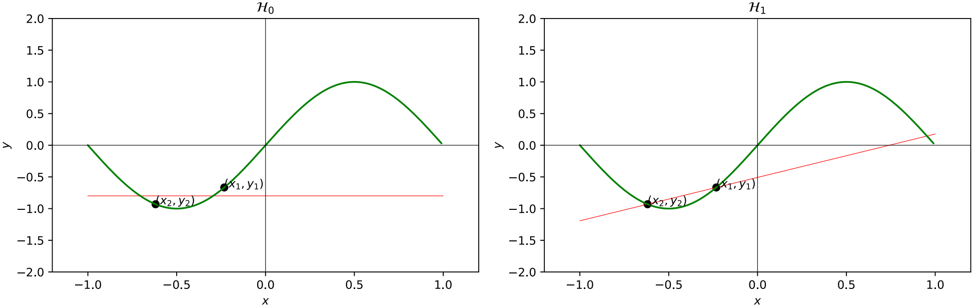
- Consider a target function sine

$$\begin{array}{rcccc} f & : & [-1,1] & \rightarrow & \mathbb{R} \\ & & x & \rightarrow & \sin(\pi x) \end{array} \tag{48}$$

- We generate 100 data sets $\{\mathcal{D}_i\}$, $i = 1, ..., 100$, each containing $N = 2$ data points, independently from the sinusoidal curve $f(x) = \sin(\pi x)$. For each data set $\mathcal{D}_i$, we fit the data using one of two models
  - $\mathcal{H}_0$: set of all lines of the form $h(x) = b$
  - $\mathcal{H}_1$: set of all lines of the form $h(x) = ax + b$
  - Note that $\mathcal{H}_0 \subset \mathcal{H}_1$

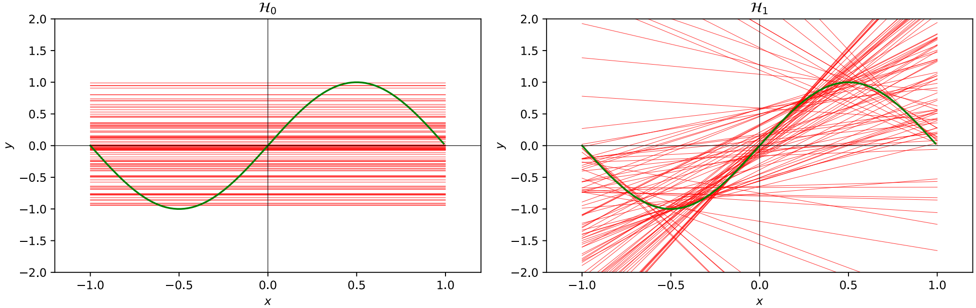
Example: two learning models (cont.)

- Given a data set $\mathcal{D}_i = \{(x_1, y_1), (x_2, y_2)\}$.
  - For $\mathcal{H}_0$, we choose the constant hypothesis that best fits the data (the horizontal line at the midpoint, $b = (y_1 + y_2)/2$).
  - For $\mathcal{H}_1$, we choose the line that passes through the two data points (x_1, y_1) and (x_2, y_2).



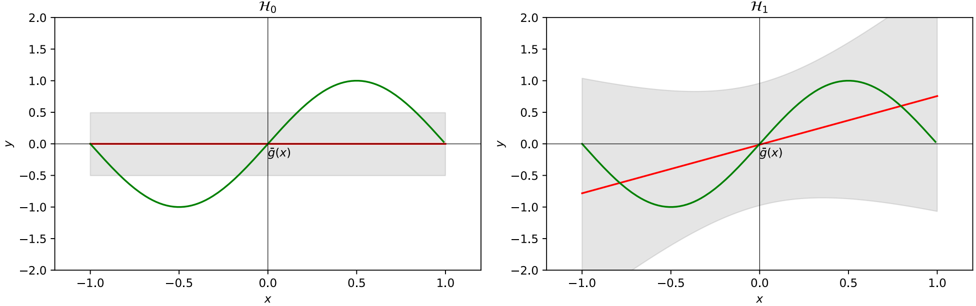
Example: two learning models (cont.)

- Repeating this process with 100 data sets $\{\mathcal{D}_i\}$, $i = 1, \ldots, 100$,



Example: two learning models (cont.)

- The bias-variance for each learning model



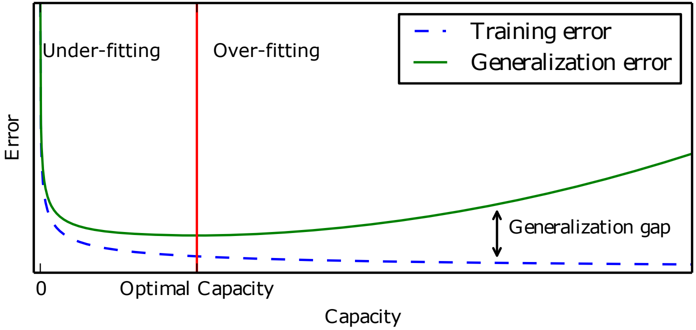
- bias=0.50
- var=**0.25**

- bias=**0.21**
- var=1.69

Generalization and Capacity

The criteria determining how well a machine learning model will perform:

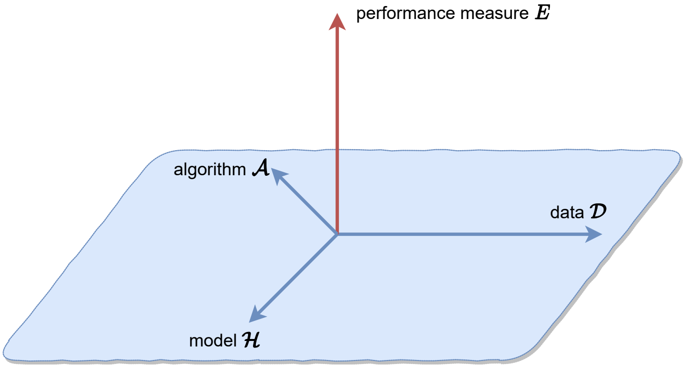
1. Make the training error small.
2. Make the gap between training and test (generalization) error small.



Regularization

Concept 14

Regularization is any modification we make to a learning model that is intended to reduce its generalization error but not its training error.



Addressing

High bias	High variance
Obtain more features	Decrease number of features
Decrease regularization λ	Increase regularization λ
Extend model	Obtain more data
Train longer	Stop early
New model architecture	New model architecture

Regularization for Linear Regression

- Using regularized MSE_{train} for linear regression

$$\boldsymbol{w} = \arg\min_{\boldsymbol{w}} \left[MSE_{train} + \lambda \left(\frac{1}{N} \boldsymbol{w}^T \boldsymbol{w} \right) \right] \tag{49}$$

where λ is the regularization coefficient (*hyper-parameter*) that controls the relative importance of the data-dependent error MSE_{train} and the regularization term $\lambda \left(\frac{1}{N} \boldsymbol{w}^T \boldsymbol{w} \right)$

- Solving for $\boldsymbol{w}$, we obtain

$$\boldsymbol{w} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{50}$$

Example: one learning model

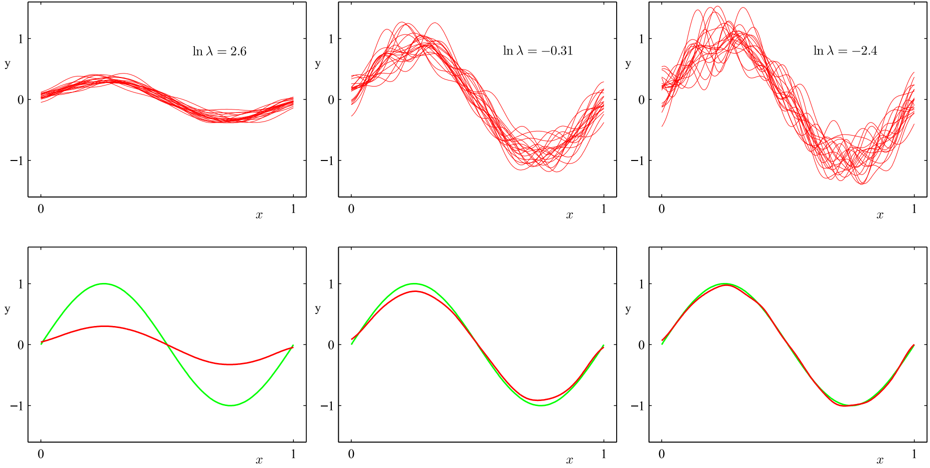
- Consider a target function sine

$$\begin{array}{rcccc} f & : & [-1,1] & \rightarrow & \mathbb{R} \\ & & x & \rightarrow & \sin(2\pi x) \end{array} \tag{51}$$

- We generate 100 data sets $\{\mathcal{D}_i\}$, $i = 1, ..., 100$, each containing $N = 25$ data points, independently from the sinusoidal curve $f(x) = \sin(2\pi x)$. For each data set $\mathcal{D}_i$, we fit a model with 24 Gaussian basis functions by minimizing the regularized error function

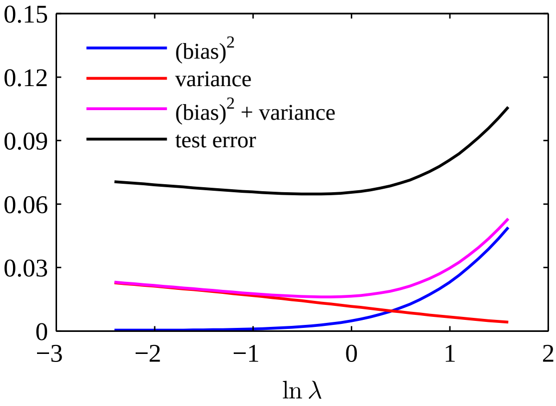
Example: one learning model (cont.)

- Illustration of the dependence of bias and variance on model regularization coefficient



Example: one learning model (cont.)

- Summary of the dependence of bias and variance on model regularization coefficient



References

Goodfellow, I., Bengio, Y., and Courville, A. (2016).
Deep learning.
MIT press.

Lê, B. and Tô, V. (2014).
Cở sở trí tuệ nhân tạo.
Nhà xuất bản Khoa học và Kỹ thuật.

Russell, S. and Norvig, P. (2021).
Artificial intelligence: a modern approach.
Pearson Education Limited.