LINEAR MODEL

Bùi Tiến Lên

01/09/2019

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

Contents

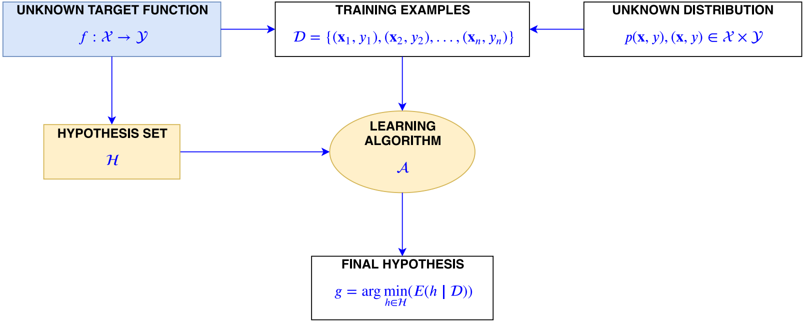
Notation

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
$\boldsymbol{w}^\mathsf{T}$	transpose
$\boldsymbol{X}\boldsymbol{Y}$	matrix multiplication
$\boldsymbol{X}^{-1}$	inverse

Learning Goal

- Learning diagram revisited

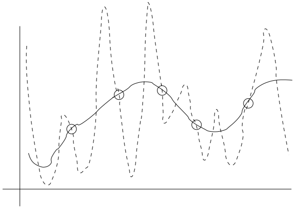


Inductive Bias

Theorem 1 (No Free Lunch Theorems)

An unbiased learner can never generalize.

- The converse of the Inductive Learning Hypothesis is that generalization only possible if we make some assumptions, or introduce some priors. We need an Inductive Bias.
- **Consider**: arbitrarily wiggly functions or random truth tables or non-smooth distributions.

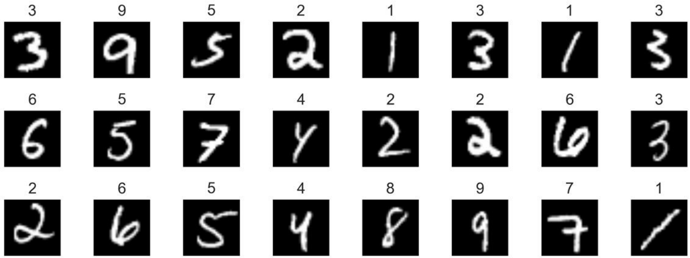


0 0 0	0
0 0 1	?
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	?
1 1 0	1
1 1 1	?

Linear 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**

$$\text{pixels} \quad \boldsymbol{x}^\top = \begin{pmatrix} x_0 & x_1 & \dots & x_{256} \end{pmatrix}$$

$$\text{linear model} \quad \boldsymbol{w}^\top = \begin{pmatrix} w_0 & w_1 & \dots & w_{256} \end{pmatrix}$$

- **Feature extraction**: extract useful information

$$\textit{intensity and symmetry} \quad \boldsymbol{x}^\top = \begin{pmatrix} x_1 & x_2 \end{pmatrix}$$

$$\text{linear model} \quad \boldsymbol{w}^\top = \begin{pmatrix} w_1 & w_2 \end{pmatrix}$$

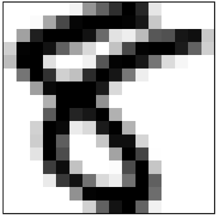
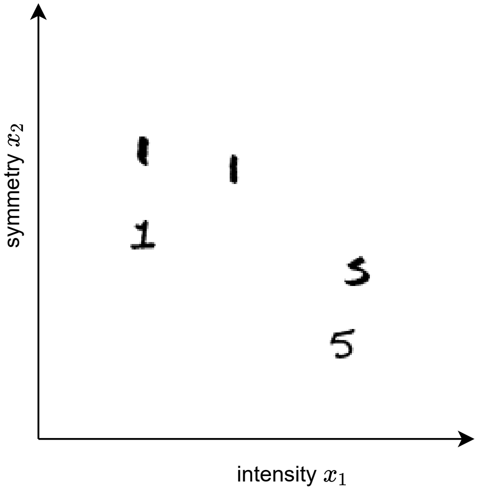
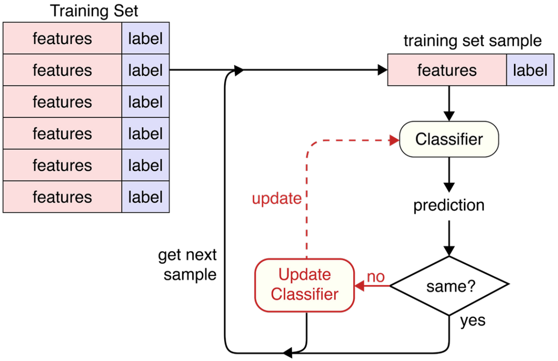


Illustration of Features



Classifier Training

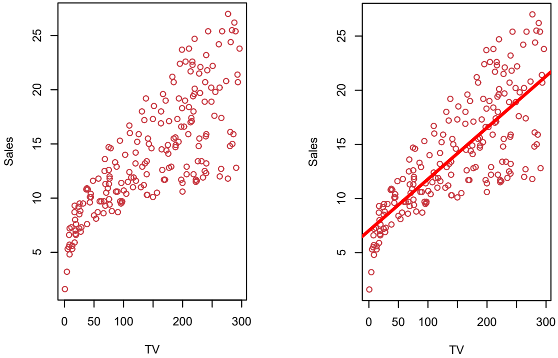
- **Select** the learning model for **classifier**, e.g., Perceptron
- **Train** the classifier



Linear Regression

Simple Problem

- 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** and **sales**



Solving Problem by Learning

- **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 $\mathcal{H}$

$$y \approx \hat{y} = h_{\boldsymbol{w}}(\mathbf{x}) = \boldsymbol{w}^\intercal \boldsymbol{x} \quad (1)$$

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

Solving Problem by Learning (cont.)

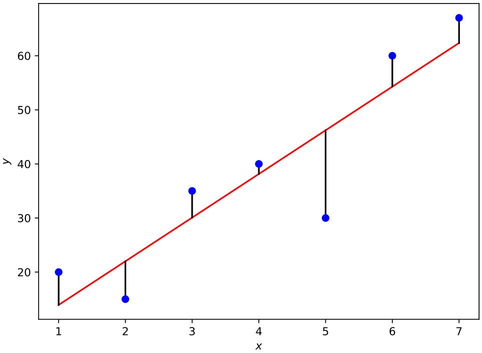
- **Task $\boldsymbol{T}$**: to predict y from $\boldsymbol{x}$ by outputting $\hat{y} = h_{\boldsymbol{w}}(\mathrm{x}) = \boldsymbol{w}^\intercal \boldsymbol{x}$
- **Performance measure $\boldsymbol{P}$**:
  The *mean squared error* MSE_{train} of the model on the train set $\mathcal{D}_{train}$ denoted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

$$MSE_{train} = \frac{1}{N} \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|^2 \tag{2}$$

Construct the matrix $\boldsymbol{X}$ and the vectors $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$

$$\underbrace{\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^\intercal \\ \boldsymbol{x}_2^\intercal \\ \vdots \\ \boldsymbol{x}_N^\intercal \end{bmatrix}}_{\text{input data matrix}}, \quad \underbrace{\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}}, \quad \underbrace{\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_N \end{bmatrix}}_{\text{output vector}} \tag{3}$$

Solving Problem by Learning (cont.)



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

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

Solving Problem by Learning (cont.)

Solution

- Compute the gradient of MSE_{train}

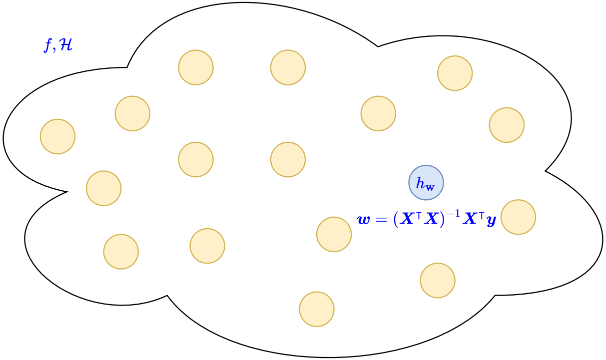
$$\begin{aligned}\nabla_{\boldsymbol{w}}(MSE_{train}) &= \nabla_{\boldsymbol{w}}(\boldsymbol{w}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} - \boldsymbol{w}^\top\boldsymbol{X}^\top\boldsymbol{y} - \boldsymbol{y}^\top\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^\top\boldsymbol{y}) \\ &= 2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w}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

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

$$\begin{aligned}\nabla_{\boldsymbol{w}}(MSE_{train}) &= 0 \\ \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} - \boldsymbol{X}^\top\boldsymbol{y} &= 0 \\ \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{w} &= \boldsymbol{X}^\top\boldsymbol{y} \\ \boldsymbol{w} &= (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y} \qquad (4)\end{aligned}$$

■

Solving Problem by Learning (cont.)



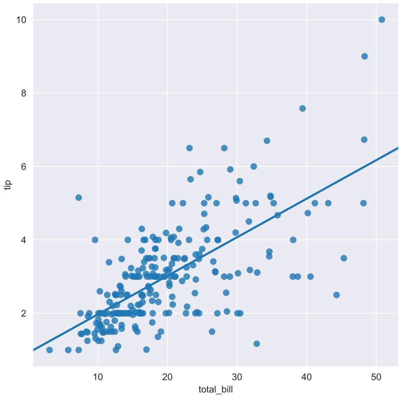
Programming Example

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

```
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)
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 $y = f(\boldsymbol{x}) = f(x_1, x_2) = w_0 + w_1 x_1 + w_2 x_2$ given the following data set $\mathcal{D}$

input $\boldsymbol{x}$	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**)

Linear in What?

- **Linearity in the weights**

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 x_1 + \ldots + w_D x_D \tag{5}$$

- We extend **the model** by introducing **linear combinations** of **fixed nonlinear functions** of the input variables

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = w_0 + w_1 \phi_1(\boldsymbol{x}) + \ldots + w_M \phi_M(\boldsymbol{x}) \tag{6}$$

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

Some types of basis functions

- Power

$$\phi_j(x) = x^j \tag{7}$$

- Gaussian

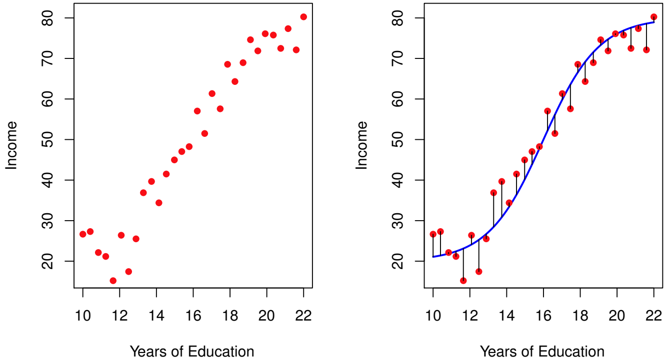
$$\phi_j(x) = \exp\left(-\frac{(x-\mu_j)^2}{s^2}\right) \tag{8}$$

- Sigmoid

$$\phi_j(x) = \frac{1}{1 + e^{-\frac{x-\mu_j}{s}}} \tag{9}$$

Another Problem

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



Solving Problem by Learning

- **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 $\mathcal{H}$

$$y \approx \hat{y} = h_{\boldsymbol{w}}(\mathbf{x}) = \boldsymbol{w}^\intercal \boldsymbol{\phi}(\boldsymbol{x}) \tag{10}$$

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

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} \phi_0(\boldsymbol{x}) \\ \phi_1(\boldsymbol{x}) \\ \vdots \\ \phi_M(\boldsymbol{x}) \end{bmatrix} \tag{11}$$

Solving Problem by Learning (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 $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2) \ldots (\boldsymbol{x}_N, y_N)\}$
- **The learning goal**: find the vector of parameter $\boldsymbol{w}$ such that

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

Solving Problem by Learning (cont.)

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

$$\underbrace{\mathbf{\Phi} = \begin{bmatrix} \phi(\boldsymbol{x}_1)^\intercal \\ \phi(\boldsymbol{x}_2)^\intercal \\ \vdots \\ \phi(\boldsymbol{x}_N)^\intercal \end{bmatrix}}_{\text{design matrix}}, \quad \underbrace{\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\text{target vector}} \tag{12}$$

2. Calculate the vector of parameters

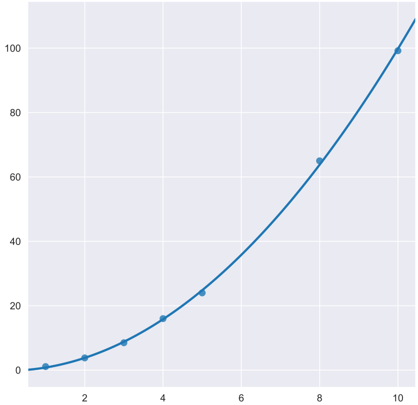
$$\boldsymbol{w} = (\mathbf{\Phi}^\intercal \mathbf{\Phi})^{-1} \mathbf{\Phi}^\intercal \boldsymbol{y} \tag{13}$$

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)
plt.show()
```

Programming Example (cont.)



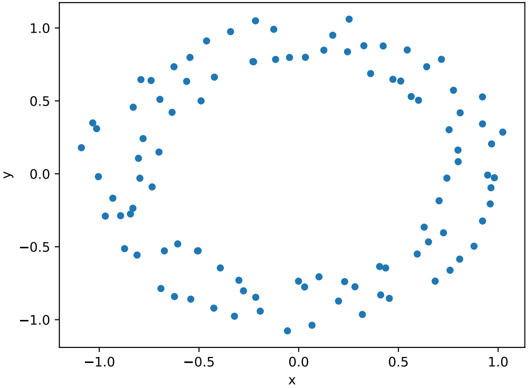
Word Example

- 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?



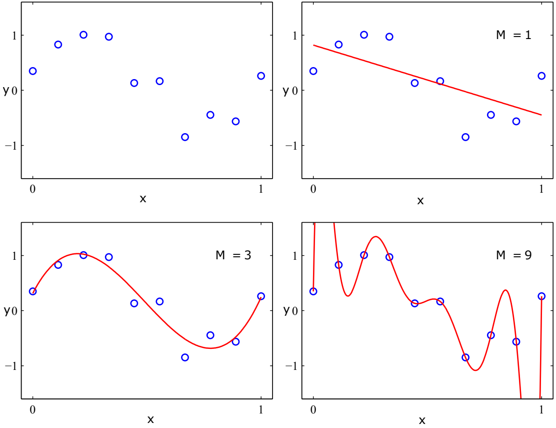
Capacity, Overfitting and Underfitting

Model Training

We consider three hypothesis sets (polynomial functions) $\mathcal{H}_1$,$\mathcal{H}_3$ and $\mathcal{H}_9$ and the results of fitting the models to the data set $\mathcal{D}$

Which one

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



Model Training (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

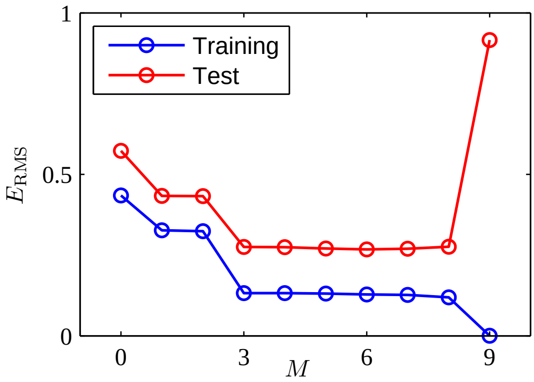


Figure 1: Graphs of the root-mean-square error evaluated on the **training set** and on an **independent test set** for various values of M

What happen if increasing N

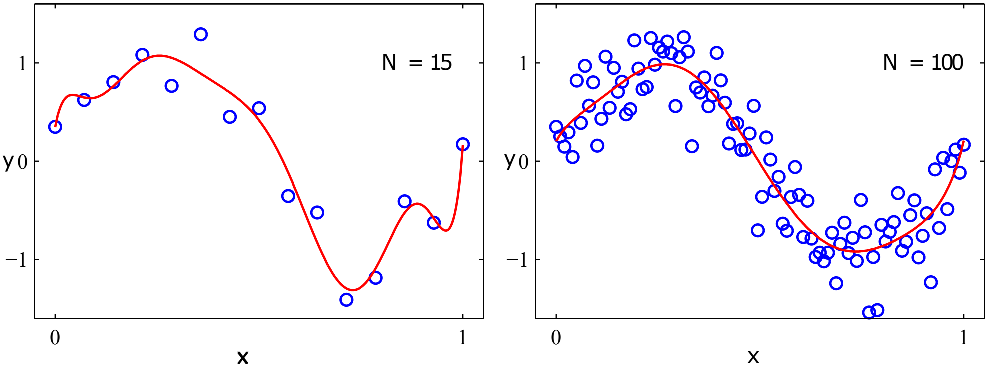
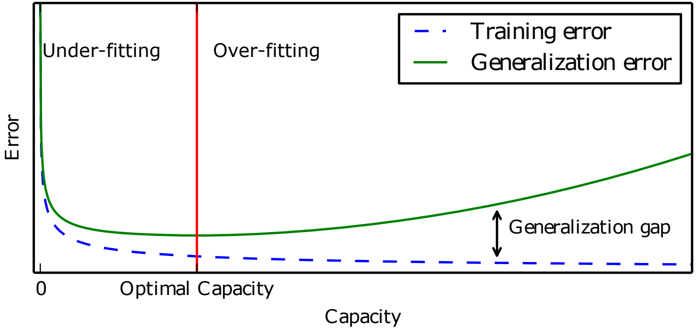


Figure 2: Using the $M = 9$ polynomial for $N = 15$ data points (left plot) and $N = 100$ data points (right plot). We see that increasing the size of the data set reduces the over-fitting problem

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.

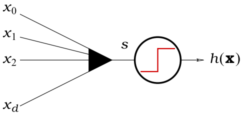


Logistic Regression

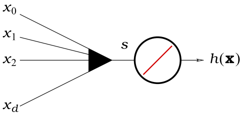
A Third Linear Model

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

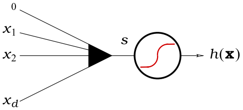
linear classification
$h(\boldsymbol{x}) = sign(s)$



linear regression
$h(\boldsymbol{x}) = s$



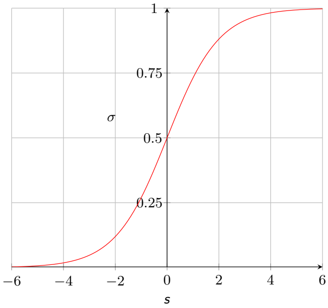
logistic regression
$h(\boldsymbol{x}) = \sigma(s)$



The logistic function

The formula

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



Some 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

Problem Statement

- The target function f is the probability

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

- Data $\mathcal{D} = \{(\boldsymbol{x}_1, y_1) ... (\boldsymbol{x}_N, y_N)\}$ with binary $y_i \in \{-1, 1\}$ generated by noisy target

$$P(y \mid \boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}) & \text{for} \quad y = +1 \\ 1 - f(\boldsymbol{x}) & \text{for} \quad y = -1 \end{cases}$$

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

Error measure

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

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

where $z = h_{\boldsymbol{w}}(\boldsymbol{x}) = \sigma(\boldsymbol{w}^T\boldsymbol{x})$

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

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

Error measure (cont.)

- Maximizing Likelihood

$$
\begin{array}{lll}
 & \text{Maximize} & \prod_{n=1}^{N} P(y_n \mid \boldsymbol{x}_n, \boldsymbol{w}) \\
\Leftrightarrow & \text{Minimize} & -\log \prod_{n=1}^{N} P(y_n \mid \boldsymbol{x}_n, \boldsymbol{w})
\end{array}
$$

- We define error measurement (*cross-entropy error*)

$$
E(h_{\boldsymbol{w}}) = -\sum_{n=1}^{N} (y_n \log z_n + (1 - y_n)\log(1 - z_n)) \tag{14}
$$

Learning Algorithm

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

   2.1 Compute the gradient

$$\nabla_{\boldsymbol{w}} E = -\sum_{n=1}^{N} \frac{y_n \boldsymbol{x}_n}{1 + e^{y_n \boldsymbol{w}_t^\intercal \boldsymbol{x}_n}}$$

   2.2 Update the weights

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

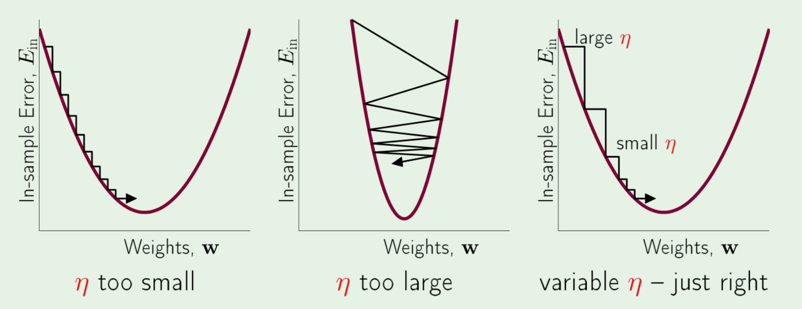
   where η is a learning rate (*hyper-parameter*)

   2.3 Iterate the next step until $\boldsymbol{w}$ is not change

3. Return the final weights $\boldsymbol{w}$

Learning Rate

- How η affects the algorithm?



Programming Example

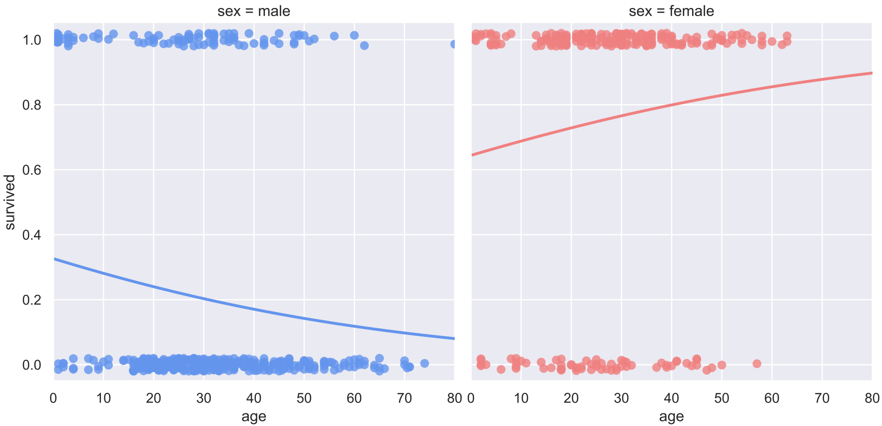
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="darkgrid")

# 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 proability 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.)



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.

Nguyen, T. (2018).
Artificial intelligence slides.
Technical report, HCMC University of Sciences.

Russell, S. and Norvig, P. (2016).
Artificial intelligence: a modern approach.
Pearson Education Limited.