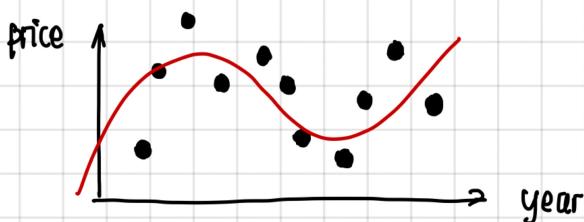


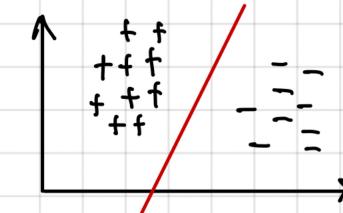
LECTURE 1 : INTRODUCTION

1. Q : What is (statistical) pattern recognition?
A : Automated recognition of patterns & regularities in data. It uses techniques developed in statistics & machine learning.
2. Q : Any differences to Machine Learning?
A : Machine learning : learn from data without being explicitly programmed. It's almost identical to pattern recognition, since learning implies finding patterns/regularities.
3. Q : What is machine learning ? How is it different from non-learning algorithms?
A : Non-learning algorithms are based on predetermined rules, developed by humans. The rules are fixed regardless the input data.
4. Q : What's wrong with non-learning algorithms?
A : (1) Humans (programmers) should know the solutions & then the rules.
Hence, problems humans can't solve will never be solved
(2) Different types of data require different rules
(3) It's human-centric & not data-centric
5. Typical tasks in pattern recognition :

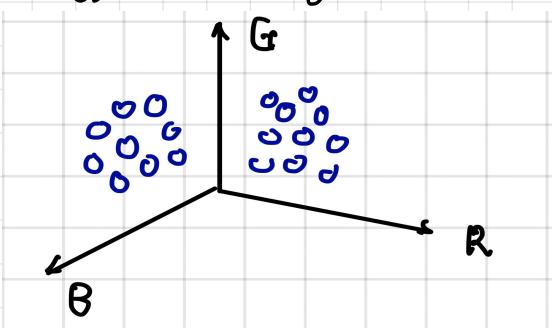
(1) Regression



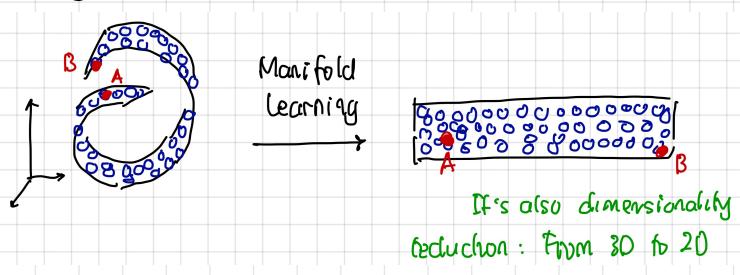
(2) Classification



(3) Clustering

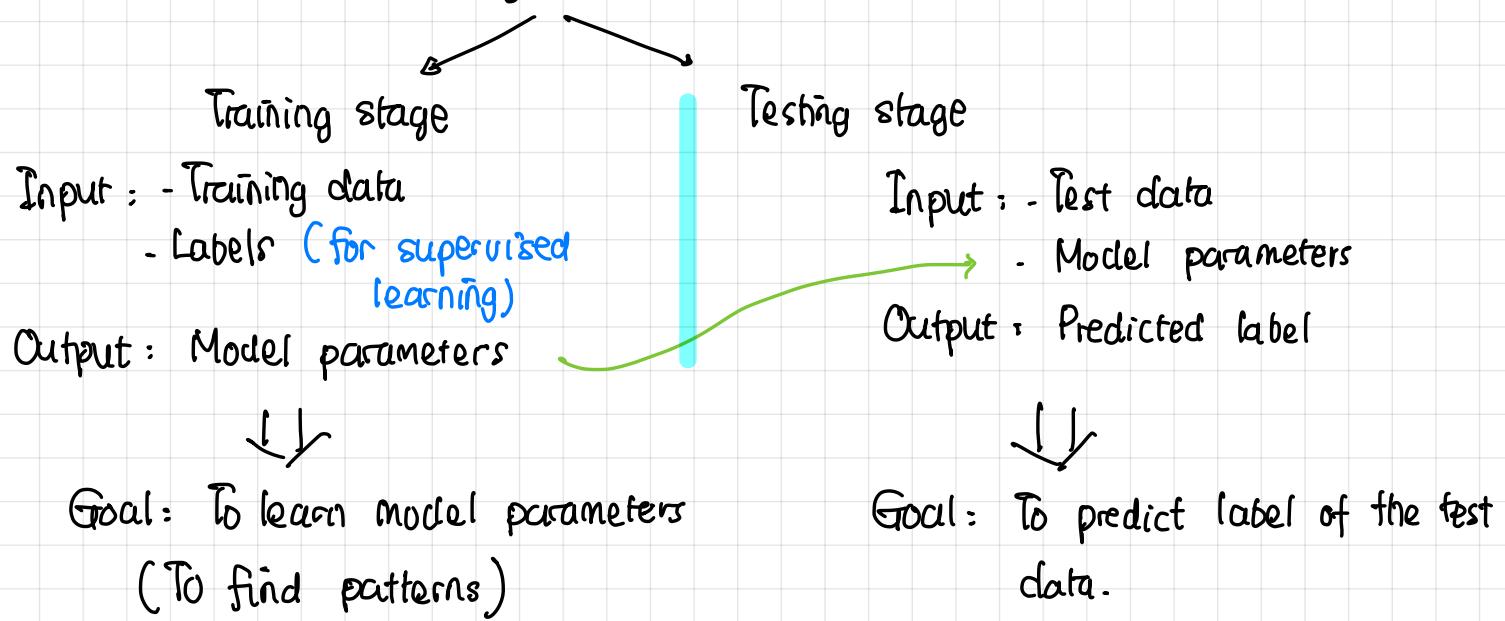


(4) Dimensionality Reduction



If's also dimensionality reduction : From 3D to 2D

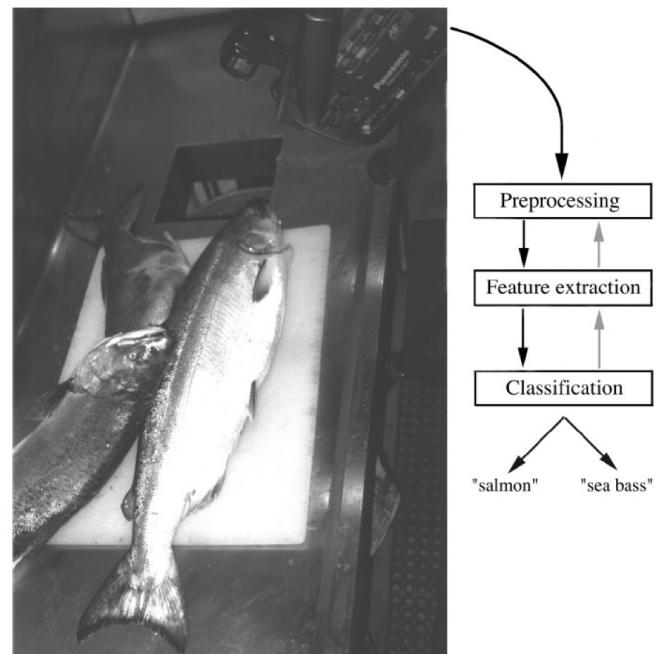
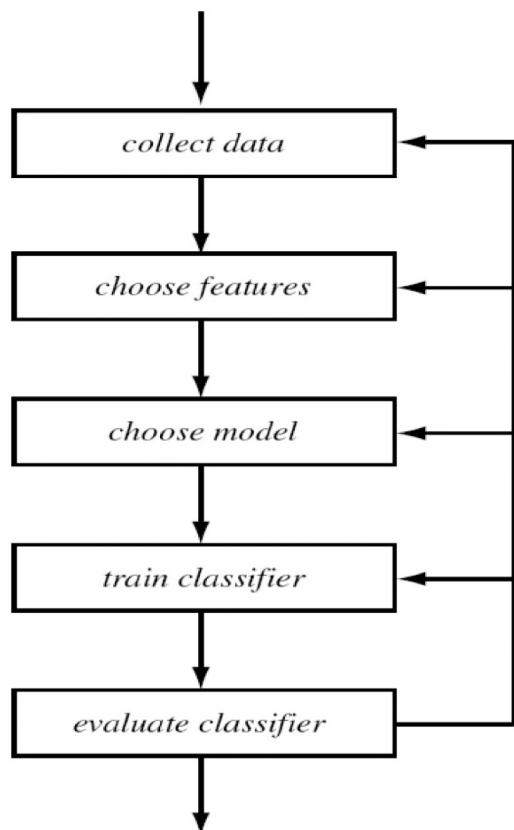
6. Common machine - learning framework :



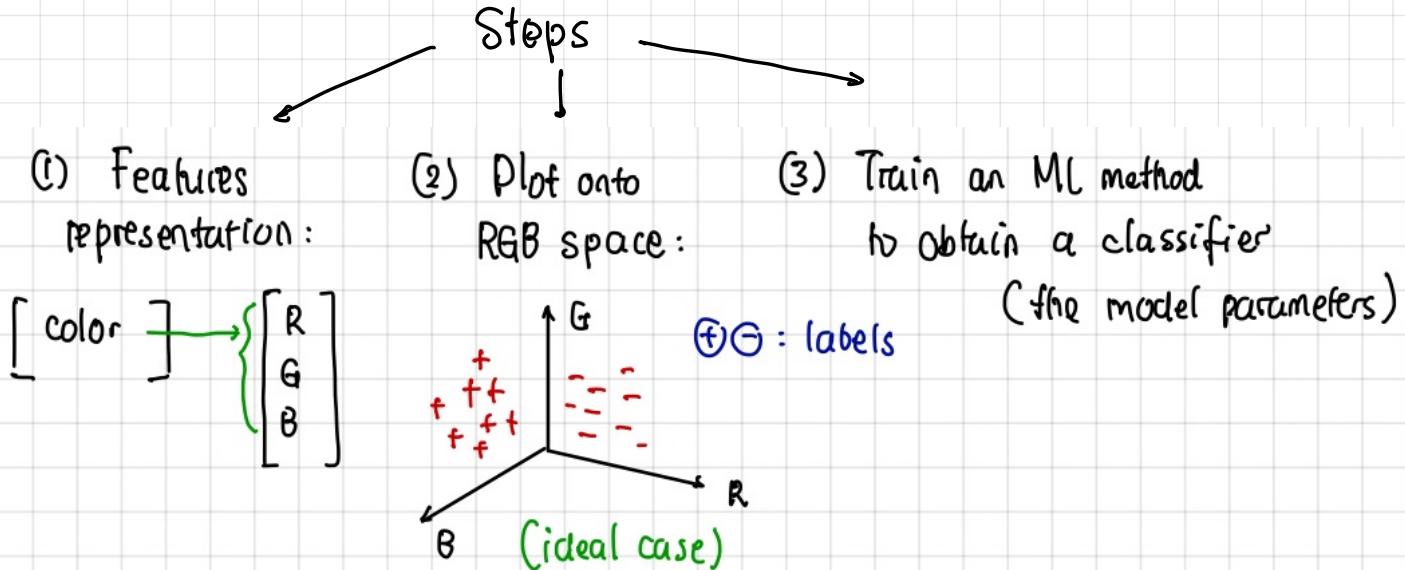
7. Types of learning (training stage) :

- (1) Supervised : Data + Labels (= labeled data)
- (2) Unsupervised : Unlabeled data
- (3) Semi-supervised : Labeled data + unlabeled data

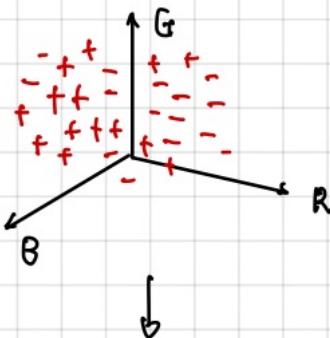
8. Design - cycle for supervised learning :



g. Example: To classify apples & oranges from their appearance:



Problems: color alone is insufficient:



Some apples are green/yellow
Some oranges are red/green

↓
Adding more features: [color
edge / contour]

The dimension becomes higher
(can't be observed / visualized)

Dimensionality reduction:
(to plot a high dimensionality
data to a lower one: 2D/3D)

Ideally: the features of different classes are separable in a higher dimension.

10. Overfitting problem :

A 2D coordinate system with a horizontal x-axis and a vertical y-axis. The plot contains several data points: five '+' symbols clustered in the upper-left region and five '-' symbols clustered in the lower-right region. A solid red line, representing a linear classifier, runs diagonally from the top-left towards the bottom-right, separating the two classes. The axes have arrows at their ends.

should be in the middle between the 2 classes

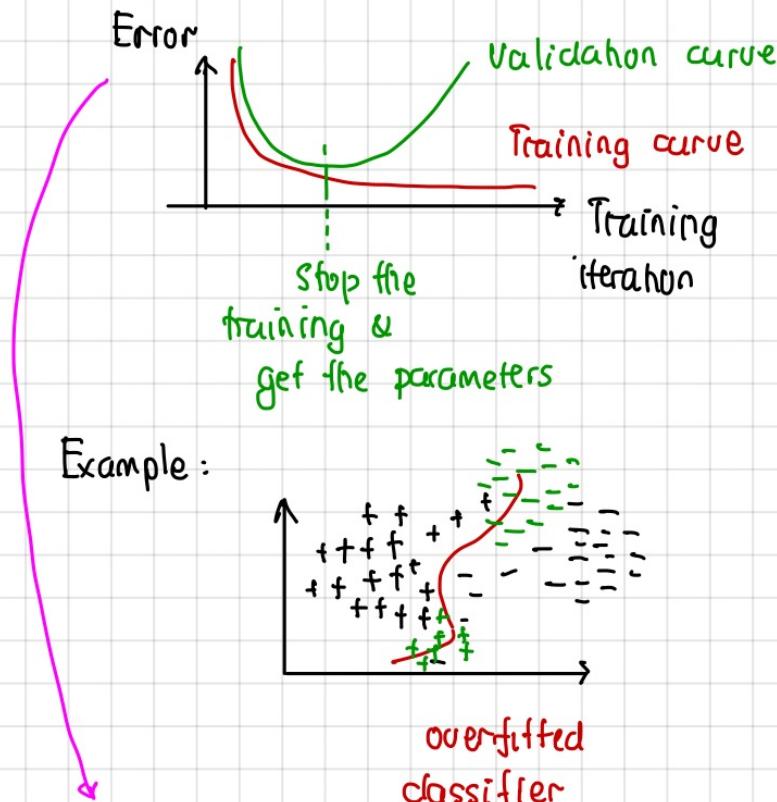
New data : test data .
the classification becomes wrong .

II. Stages in Machine Learning :

- ① Training
- ② Testing
- ③ Validation \rightarrow to avoid overfitting

12. Validation stage :

Assumption : the validation data can represent the distribution of the test data.



Error = classification error

13. Inductive Reasoning :

\Rightarrow Deductive :

1. All men are mortal
2. Socrates is a man
3. Socrates is mortal

\Rightarrow Inductive :

1. All swans we've seen is white
2. All swans are white



Generalization :

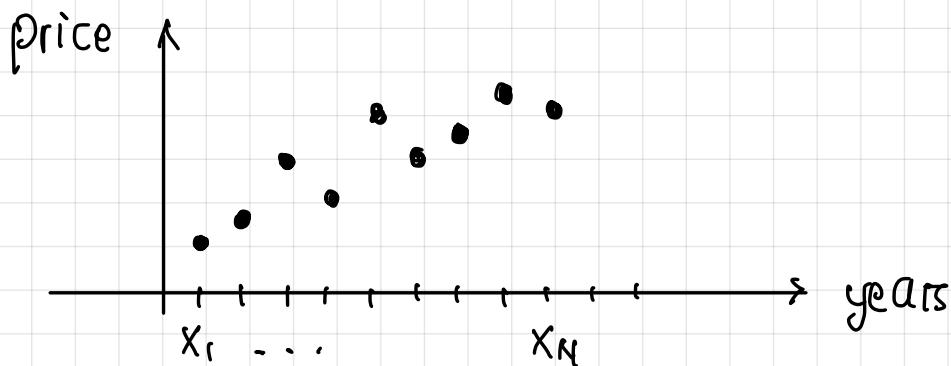


14. ML's drawbacks :

- (1) No data, no learning
- (2) Data dependency that can be bias
- (3) Inductive in nature (not always correct)

LEAST-SQUARES REGRESSION

Problem : Given a price list of a house over a period of time,
how can we predict the house's price in the future?



Solution:

(1) Data Modeling: $y(\bar{x}, w_0) = w_0$ (simplest model)

Assume: there is only one parameter to represent the whole prices.

where: $y_n = y(\bar{x}, w_0) =$ the predicted price using the model

(2) Error Function:

$$E(w_0) = \frac{1}{2} \sum_{n=1}^N (w_0 - t_n)^2$$

} Least Squares

where:

t_n = the true (observed) price at time n .

N = the number of data

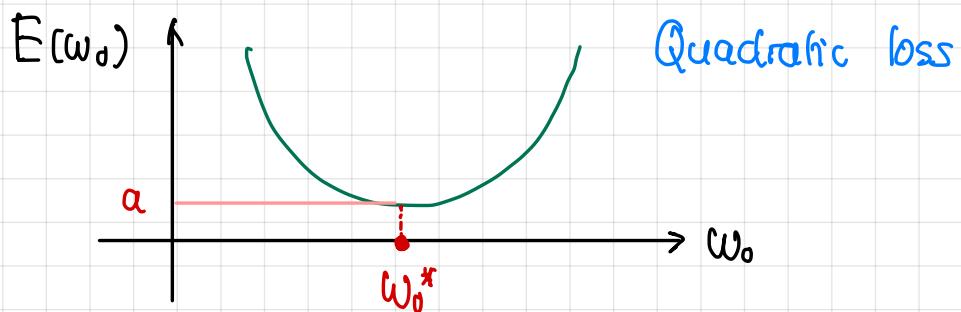
(3) Optimization (Minimization):

$$w_0^* = \underset{\{w_0\}}{\operatorname{argmin}} E(w_0) = \underset{\{w_0\}}{\operatorname{argmin}} \frac{1}{2} \sum_{n=1}^N (w_0 - t_n)^2$$

the best estimation
of w_0

all the candidate values of w_0

Finding w_0^* through optimization:



$$\alpha = \min_{\{w_0\}} E(w_0)$$

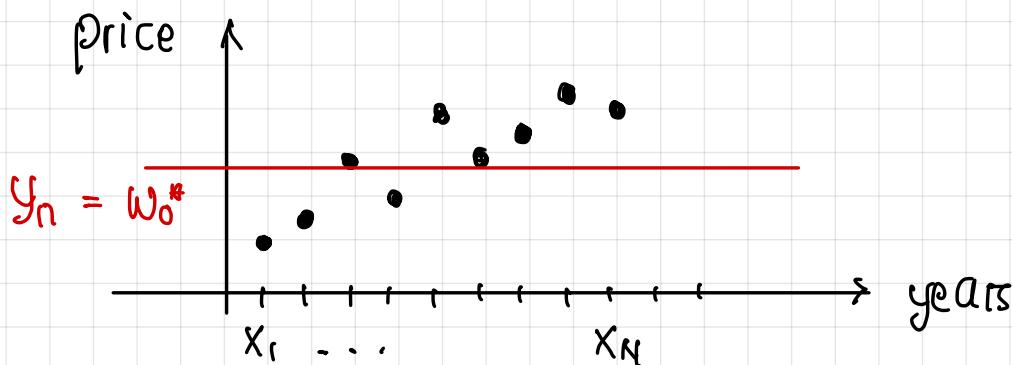
Q: How to get the best w_0 that minimizes the error?

A: Since the loss is quadratic:

$$\begin{aligned} \frac{\partial E(w_0)}{\partial w_0} &= 0 \\ &= \frac{1}{2} \sum_n \frac{\partial}{\partial w_0} (w_0 - t_n)^2 \\ &= \sum_n (w_0 - t_n) = 0 \end{aligned}$$

$$\sum_n w_0 = \sum_n t_n$$

$$w_0 = \frac{1}{N} \sum_n t_n \quad : \quad w_0 \text{ is the mean of the prices}$$



Better data modeling:

(1) Data modeling:

$$y(x_n, \bar{w}) = x_n w_r + w_0 \quad (\text{linear model})$$

where: $\bar{w} = \begin{bmatrix} w_0 \\ w_r \end{bmatrix} \rightarrow \text{model parameters}$

(2) Error Function:

$$E(\bar{w}) = \frac{1}{2} \sum_n (y(x_n, \bar{w}) - t_n)^2$$

(3) Optimization: $\bar{w}^* = \underset{\{\bar{w}\}}{\operatorname{arg\min}} E(\bar{w})$

Solving the optimization:

Least squares loss is a quadratic loss, hence:

$$\frac{\partial E(\bar{w})}{\partial \bar{w}} = 0$$

$$\text{meaning: } \frac{\partial E(\bar{w})}{\partial w_0} = 0$$

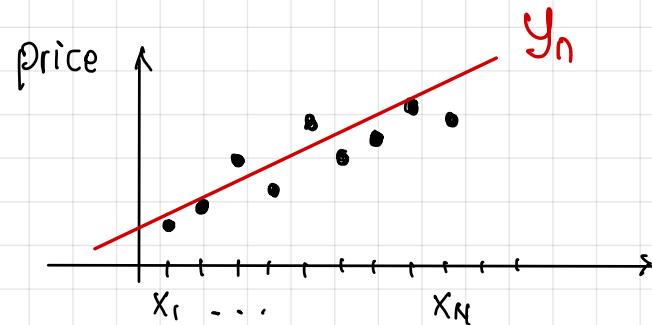
multivariable calculus

$$\begin{aligned} (1) \quad \frac{\partial E(\bar{w})}{\partial w_0} &= \frac{1}{2} \sum_n (w_0 + x_n w_r - t_n)^2 = 0 \\ &= \sum_n (w_0 + x_n w_r - t_n) = 0 \end{aligned}$$

$$w_0 = \frac{1}{N} \left(\sum_n t_n - \sum_n x_n w_r \right)$$

$$(2) \quad \frac{\partial E(\bar{w})}{\partial w_r} = \sum_n (w_0 + x_n w_r - t_n) x_n = 0$$

$$w_r = \left(\sum_n t_n x_n - w_0 x_n \right) / \sum_n x_n^2$$



Two equations and two unknowns (w_0 and w_r) \rightarrow solvable

• General Model: Polynomial Functions

Problem: What can we do if we do not know the shape of the data distribution in advance?

(i) Data modeling:

All possible functions:

$$\text{horizontal line: } y = w_0$$

$$\text{linear: } y = w_0 + w_1 x$$

$$\text{quadratic: } y = w_0 + w_1 x + w_2 x^2$$

$$\text{cubic: } y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

:

:

General form of polynomial functions:

$$y(x_n, \bar{w}) = \sum_{m=0}^{M-1} w_m x_n^m$$

$$\bar{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{M-1} \end{bmatrix}_{M \times 1}$$

M = the degree of polynomial



Vector form:

$$y_n = \bar{w}^T \bar{x}_n$$

1×1 $1 \times M$ $M \times 1$

$$y_n = y(x_n, \bar{w})$$

$$\bar{x}_n = \begin{bmatrix} x^0 \\ x^1 \\ \vdots \\ x^{M-1} \end{bmatrix}_{M \times 1}$$

Matrix form:

$$\bar{y} = \bar{X} \bar{w}$$

$N \times 1$ $N \times M$ $M \times 1$

$$\bar{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}_{N \times 1} ; \quad \bar{X} = \begin{bmatrix} x_1^0 & \dots & x_1^{M-1} \\ \vdots & \ddots & \vdots \\ x_N^0 & \dots & x_N^{M-1} \end{bmatrix}_{N \times M}$$

(2) Error function :

$$E(\bar{w}) = \frac{1}{2} \sum_{n=1}^N (y_n - f_n)^2$$

$$; y_n = y(x_n, \bar{w})$$

where :

t_n is the correct price

$y(x_n, \bar{w})$ is the estimated price : $y(x_n, \bar{w}) = \sum_{m=0}^M w_m x_n^m$

N = the number of the data.

In the training stage, our goal is to predict \bar{w} ,

from \bar{x} (data) and \bar{t} (label(s)).

In a vector form:

$$E(\bar{w}) = \frac{1}{2} (\bar{y} - \bar{t})^T (\bar{y} - \bar{t})$$

$$\text{where: } \bar{t} = [t_1, t_2, \dots, t_N]^T$$

(3) Optimization:

$$\bar{w}^* = \underset{\{\bar{w}\}}{\operatorname{argmin}} E(\bar{w}) = \underset{\{\bar{w}\}}{\operatorname{argmin}} \frac{1}{2} \sum_{n=1}^N (\bar{w}^T \bar{x}_n - t_n)^2$$

Since the cost function is quadratic: $\frac{d}{d\bar{w}} E(\bar{w}) = 0$

$$f(g(x))' = f'(g(x)) \cdot g'(x)$$

$$\begin{aligned} \frac{d}{d\bar{w}} \left[\frac{1}{2} \sum_{n=1}^N (\bar{w}^T \bar{x}_n - t_n)^2 \right] &= \frac{1}{2} \sum_{n=1}^N \frac{d}{d\bar{w}} (\bar{w}^T \bar{x}_n - t_n)^2 \bar{x}_n (\bar{w}^T \bar{x}_n - t_n) \\ &= \sum_{n=1}^N \bar{x}_n (\bar{w}^T \bar{x}_n - t_n) = \sum_n \bar{x}_n (\bar{w}^T \bar{x}_n) - \sum_n \bar{x}_n t_n \end{aligned}$$

Matrix form:

$$\frac{\partial}{\partial \bar{w}} E(\bar{w}) = \underbrace{\sum_n \bar{x}_n (\bar{w}^T \bar{x}_n)}_{X^T \bar{w}} - \underbrace{\sum_n \bar{x}_n t_n}_{X^T \bar{t}}$$

$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} t_1 + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} t_2 + \dots + \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} t_N$
 $X^T \bar{E}$
 $M \times N \quad N \times 1$
 $[x_1, x_2, \dots, x_N] \cdot [t_1, t_2, \dots, t_N]$
 \bar{t}
 (see the additional notes)

$$\frac{\partial}{\partial \bar{w}} E(\bar{w}) = X^T X \bar{w} - X^T \bar{t} = 0$$

$$X^T X \bar{w} = X^T \bar{t}$$

$$\bar{w} = \left(\frac{X^T X}{M \times M} \right)^{-1} X^T \bar{t}$$

→

$$\bar{w} = X^+ \bar{t}$$

- Inverse & pseudo inverse:

the determinant is not zero

If A is a square & non-singular matrix, then the inverse of A is A^{-1}

$$\text{where } AA^{-1} = A^{-1}A = I.$$

But if A is non-square $N \times M$ matrix (where $N > M$), how to compute A^{-1} ?

Solution:

pseudo-inverse:

$$A^+ = (A^T A)^{-1} A^T$$

Additional Notes

1. From scalar operations to vector operations):

$$\sum_{k=1}^K w_k x^k = [w_1 \ w_2 \ \dots \ w_M] \begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^M \end{bmatrix} = w_1 x^1 + w_2 x^2 + \dots + w_M x^M$$

$1 \times M$ $M \times 1$

$$= \bar{W}^T \bar{x}$$

$1 \times M$ $M \times 1$

2. From Vector Operations to Matrix Operations:

$$\sum_{n=1}^N \bar{x}_n \bar{x}_n^T = \sum_{n=1}^N \begin{bmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^M \end{bmatrix} \begin{bmatrix} x_n^1 & x_n^2 & \dots & x_n^M \end{bmatrix}$$

$M \times 1$ $1 \times M$

$$= \sum_{n=1}^N \underbrace{\begin{bmatrix} x_n^1 x_n^1 & \dots & x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ x_n^M x_n^1 & \dots & x_n^M x_n^M \end{bmatrix}}_M = \begin{bmatrix} \sum_{n=1}^N x_n^1 x_n^1 & \dots & \sum_{n=1}^N x_n^1 x_n^M \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^N x_n^M x_n^1 & \dots & \sum_{n=1}^N x_n^M x_n^M \end{bmatrix}$$

Let: $\bar{x} = \begin{bmatrix} x_1^1 & \dots & x_1^M \\ \vdots & \ddots & \vdots \\ x_N^1 & \dots & x_N^M \end{bmatrix}$

$N \times M$

$$\bar{x}^T \bar{x} = \begin{bmatrix} x_1^1 & \dots & x_N^1 \\ \vdots & \ddots & \vdots \\ x_1^M & \dots & x_N^M \end{bmatrix} \begin{bmatrix} x_1^1 & \dots & x_1^M \\ \vdots & \ddots & \vdots \\ x_N^1 & \dots & x_N^M \end{bmatrix}$$

$M \times N$ $N \times M$

$$\begin{aligned}
&= \begin{bmatrix} X_1^T X_1 + X_2^T X_2 + \dots + X_N^T X_N & \dots & X_1^T X_1^M & X_2^T X_2^M + \dots + X_N^T X_N^M \\ \vdots & \ddots & \ddots & \vdots \\ X_1^M X_1^T + X_2^M X_2^T + \dots + X_N^M X_N^T & \dots & X_1^M X_1^M + X_2^M X_2^M + \dots + X_N^M X_N^M \end{bmatrix} \\
&= \begin{bmatrix} \sum_{n=1}^N X_n^T X_n^T & \dots & \sum_{n=1}^N X_n^T X_n^M \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^N X_n^M X_n^T & \dots & \sum_{n=1}^N X_n^M X_n^M \end{bmatrix}
\end{aligned}$$

They are identical!

Recall:

$$\sum_{n=1}^N \bar{X}_n \bar{X}_n^T = \begin{bmatrix} \sum_{n=1}^N X_n^T X_n^T & \dots & \sum_{n=1}^N X_n^T X_n^M \\ \vdots & \ddots & \vdots \\ \sum_{n=1}^N X_n^M X_n^T & \dots & \sum_{n=1}^N X_n^M X_n^M \end{bmatrix}$$

5. Matrix inverse : $A A^{-1} = A^{-1} A = I$

A is invertible (or A^{-1} exists) if :

- ① A is a square matrix : A is a $N \times N$ matrix
- ② A is not a singular matrix (= the determinant is not zero)

6. What is a singular matrix?

A matrix whose determinant = 0 → What does it mean?

A matrix represents linear equations:

$$\begin{aligned} 2x + 5y &= 1 \\ 3x + 7y &= 4 \end{aligned} \quad \left\{ \begin{matrix} 2 & 5 \\ 3 & 7 \end{matrix} \right\} \rightarrow \det \neq 0$$

↓
Thus, this matrix is invertible.

Implies the solution of x & y can be found.

$$\left[\begin{matrix} 2 & 5 \\ 3 & 7 \end{matrix} \right] \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \left[\begin{matrix} 2 & 5 \\ 3 & 7 \end{matrix} \right]^{-1} \begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

If a matrix is singular, we can't solve the equations:

$$\begin{aligned} 2x + 5y &= 1 \\ 4x + 10y &= 2 \end{aligned} \quad \left\{ \begin{matrix} 2 & 5 \\ 4 & 10 \end{matrix} \right\} \rightarrow \det = 0$$

↳ these equations are basically the same equation. Meaning, we only have 1 equation with 2 unknowns (x & y).

7. Pseudo Inverse:

A matrix is invertible if it's square. What can we do if we have a non-square matrix: A with dimensions N x M?

if $N > M$, it means

we have more equations than the unknown parameters:

$$\begin{aligned} 2x + 5y &= 1 \\ 3x + 4y &= 3 \\ x + 7y &= 5 \end{aligned} \quad \left\{ \begin{matrix} N=3 \\ M=2 \end{matrix} \right. \quad (\text{i.e. } x \text{ & } y)$$

↓
This is called an over-determined system

if $N < M$, it means

we have fewer equations than the unknown parameters:

$$2x + 5y = 1$$

↓

$$N=1, M=2$$

↓
This is called an under-determined system.

Assuming A is a $N \times M$ matrix with $N > M$, how we can get the inverse?

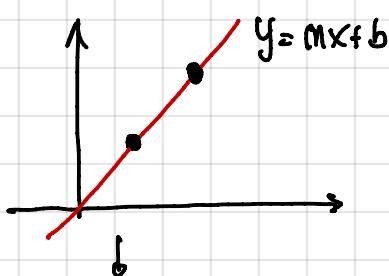
The solution is using the pseudo inverse:

$$A^+ = (A^T A)^{-1} A^T$$

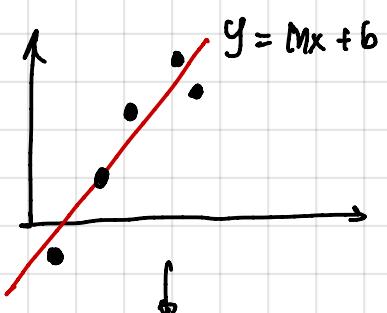


However, it's not the only way to compute the pseudo inverse. There are a few alternatives, and one of them is to use SVD (Singular Value Decomposition). In python, this can be realized by calling: `numpy.linalg.pinv()`.

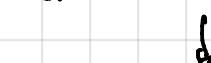
8. What is the geometrical meaning of an over-determined system?



The case when we have 2 unknowns
(i.e. m & b) and 2 equations (= the two dots)



The case of an over-determined system.



The number of unknowns (m & b)
is (much) less than the equations (5 dots)



Why don't the dots lie on the line?
→ outliers or noise.



outliers: when the model (in this case $y = mx + b$)
is just an approximation to the data.

noise: the model is correct & precise, but the
system has unexpected noise.

Questions :

Referring to our discussion in the previous pages :

1. Where is the machine learning framework (training & testing)?
2. Why come the simple least squares is called machine learning?
3. Where is the underfitting problem?
4. What does it mean by 'generalization' in Machine learning?

Additional Notes :

1. Derivative :

$$u = (f(x))^n \rightarrow \frac{du}{dx} = n f(x)^{n-1} \frac{df(x)}{dx}$$

example : $u = (5x + 6)^2 \rightarrow \frac{du}{dx} = 2(5x + 6)5 = 10(5x + 6)$

2. Partial derivative :

$$u = (f(x,y))^n \rightarrow \frac{\partial u}{\partial x} = n f(x,y)^{n-1} \frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial u}{\partial y} = n f(x,y)^{n-1} \frac{\partial f(x,y)}{\partial y}$$

example :

$$u = (5x + 6y)^3 \rightarrow \frac{\partial u}{\partial x} = 3(5x + 6y)^2 5 = 15(5x + 6y)^2$$