# 02_Classification_using_Bayes_Theory

# 2.1 Bayes Decision Theory 贝叶斯决策理论

ℹ️ Basic Assumptions

- The decision problem is posed in probabilistic terms.
- ALL relevant probability values are known.

## 2.1.1 Process

Given:

1. A test sample $\mathbf{x}$.
    - Contains features $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_l \end{bmatrix}$.
    - Often reduced, removed some non-discriminative (un-useful) features.
2. A list of classes/patterns $\omega = \{\omega_1, \omega_2, \dots \omega_c\}$.
    - Defined by human-being.
3. A classification method $M$.
   - A database storing multiple samples with the same type of $x$.
   - Each sample is assigned to an arbitrary class $\omega_{any} \in \{\omega_1, \omega_2, \dots \omega_c\}$.

   Do:

- $\{P(\omega_1|\mathbf{x}), \cdots, P(\omega_c|\mathbf{x})\} \leftarrow classify(M, \mathbf{x}, \omega)$
- That is, for all the possible classes, find:
  - The probability that the given $x$ belongs to that class.

  Get:
- $\omega_{target}(\mathbf{x}) = \text{argmax}_i \Big[P(\omega_i|x)\Big], i \in [1, c]$.
- That is, assign $x$ a class/pattern from $\omega$ with the most probable one.

Example

MNIST database.

- Test sample:
    - $x =$ A $28 \times 28$ grayscale image of a hand-written number.
- Set of classes:
    - $\omega = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
- Classification Method:
    - Derived from 10,000 of $28 \times 28$ similar gray-scale images.

- Process:
  - Given an image, using the classification method, get a list of probabilities $P(\omega) = \{P(\omega_1), P(\omega_2), \dots, P(\omega_c)\}$.
  - Select the $\omega_i$ with the largest probability $P(\omega_i)$, that is $selected = argmax[P(\omega_i)]$.

### 2.1.2 Properties of Variables.

- The set of all classes $\omega$ :
  - $c$ available classes: $\omega = \{\omega_1, \omega_2, \dots, \omega_c\}$
- Prior Probabilities $P(\omega) = \{P(\omega_1), P(\omega_2), \dots, P(\omega_c)\}$ :
  - Probability Distribution of random variable $\omega_j$ in the database.
    - The fraction of samples in the database that belongs to class $\omega_j$.
    - $P(\omega)$ is the prior knowledge on $\omega = \{\omega_1, \omega_2, \dots, \omega_c\}$.
  - It is Non-Negative.
    - $\forall i \in [1, c], P(\omega_i) \geq 0$.
    - The probabilities of all classes are greater-or-equal to 0.
  - It is Normalized.
    - $\sum_{i=1}^{c} P(\omega_i) = 1$.
    - The sum of the prior probabilities of all classes is $1$.

## 2.2 Prior & Posterior Probabilities 先验与后验概率

### 2.2.1 Definition of Prior Probability 先验概率

- Decision *BEFORE* Observation (Naïve Decision Rule).
  - Don't care about test sample $x$.
  - Given $x$, always choose the class that:
    - has the most member in the database.
    - i.e., has the highest prior probability.
- Classification Process:
  1. $\omega = \{\omega_1, \omega_2, \dots, \omega_c\}$.
  2. By counting the number of members $Num(\omega_i)$ for each class $\omega_i \in \omega, i \in [1, c]$, we get the prior probabilities $P(\omega) = \{P(\omega_1), P(\omega_2), \dots, P(\omega_c)\}$.
  3. Then, classify $x$ directly into $\text{argmax}_i[P(\omega_i)]$.
- The decision is the same all the time obviously, and the prob. of a right guess is $\frac{1}{c}$.

### 2.2.2 Definition of Posterior Probability 后验概率

- Decision *WITH* Observation.
  - Cares about test sample $\mathbf{x}$.

- Considering $\mathbf{x}$, as well as the prior probabilities $P(\omega) = \{P(\omega_1), P(\omega_2), \dots, P(\omega_c)\}$ ,
    - and give $\mathbf{x}$ the class with the biggest posterior probability.

Posterior Probability of a class $\omega_j$ on test sample $\mathbf{x}$:
- Given test sample $x$,
- how possible does $x$ could be classified into class $\omega_j$.

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{P(\mathbf{x})}$$

$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Pior}}{\text{Evidence}}$$

where:

Likelihood - $p(\mathbf{x}|\omega_j)$:
- *Known*
- The fraction of samples stored in the database that
    - is same to $\mathbf{x}$, and
    - is labeled to class $\omega_j$.

Prior probability of class $\omega_j$ - $P(\omega_j)$:
- *Known*
- The fraction of samples stored in the database that
    - is not necessarily same to $\mathbf{x}$, and
    - is labeled to class $\omega_j$.

Evidence - $P(\mathbf{x})$:
- *Irrelevant*
- Unconditional density of $\mathbf{x}$.
- $P(\mathbf{x}) = \sum_{j=1}^{N} p(\mathbf{x}|\omega_j) \cdot P(\omega_j)$

## Special Cases

1. Equal Prior Probability

$$P(\omega_1) = P(\omega_2) = \cdots = P(\omega_c) = \frac{1}{c}$$

- The amount of members in each class is same.
- Posterior probabilities $P(\omega_j|\mathbf{x})$ only depend on likelihoods $P(\mathbf{x}|\omega_j)$.

2. Equal Likelihood

$$P(\mathbf{x}|\omega_1) = P(\mathbf{x}|\omega_2) = \cdots = P(\mathbf{x}|\omega_c)$$

- The amount of members *that's same to* $\mathbf{x}$ in each class is same.

- Posterior probabilities $P(\omega_j|\mathbf{x})$ only depend on priors $P(\omega_j)$.
- Back to Naïve Decision Rule.

## 2.2.3 Classification Examples

Given:

1. Test sample $x \in \{+, -\}$.
2. A list of classes $\omega = \{\omega_1 = cancer, \omega_2 = no\_cancer\}$.
3. Classification Method $M$, with known probabilities:
   - Prior Probabilities:
   - $P(\omega_1) = 0.008$
   - $P(\omega_2) = 1 - P(\omega_1) = 0.992$
   - Likelihoods:
   - For class $\omega_1 = cancer$: $P(+|\omega_1) = 0.98$, $P(-|\omega_1) = 0.02$
   - For class $\omega_2 = no\_cancer$: $P(+|\omega_2) = 0.03$, $P(-|\omega_2) = 0.97$.

   Classification:

- Given a test sample $x = +$.
  - The prob. that this person gets cancer is:
    - $P(\omega_1|+) = \dfrac{P(+|\omega_1) \times P(\omega_1)}{P(+)} = \dfrac{0.98 \times 0.008}{P(+)} = \dfrac{0.00784}{P(+)}$.
  - The prob. that this person doesn't gets cancer is:
    - $P(\omega_2|+) = \dfrac{P(+|\omega_2) \times P(\omega_2)}{P(+)} = \dfrac{0.03 \times 0.992}{P(+)} = \dfrac{0.02976}{P(+)}$
  - Therefore, the classification result would be:
    - $\omega_{target} = argmax_i[P(\omega_i|+)]$
      $= argmax_i[\dfrac{P(+|\omega_i) \times P(\omega_i)}{P(x)}]$
      $= argmax_i[P(+|\omega_i) \times P(\omega_i)]$
      $= \omega_2$, for $0.00784 < 0.02976$
    - That is, $no\_cancer$.

# 2.3 Loss Functions 决策成本函数

## 2.3.0 Why do we use loss functions?

- Different selection errors may have differently significant consequences, i.e., "losses" or "costs". 不同决策的成本、后果不同。
  - In pure Naïve Bayes classification, we only consider probability.
  - However,
    - we can tolerate "non-cancer" being classified into "cancer",
    - while it's more lossy to classify "cancer" into "non-cancer".
  - There is a need to consider this kind of "loss" into our decision method.

- We want to know if the Bayes decision rule is optimal.
  - Need a evaluation method
  - calc how many error you make, sum together

## 2.3.1 Probability of Error

For only two classes:

- If $P(\omega_1|x) > P(\omega_2|x)$, $x \leftarrow \omega_1$. Prob. of error: $P(\omega_2|x)$.
- If $P(\omega_1|x) < P(\omega_2|x)$, $x \leftarrow \omega_2$. Prob. of error: $P(\omega_1|x)$.

## 2.3.2 Loss Function (i.e., "Cost Function")

### Basics

- An action $\alpha_i$ for a given $\mathbf{x}$ is:
  - To assign the test pattern $\mathbf{x}$ with the class $\omega_i$
- The loss $\lambda(\alpha_i|\omega_j)$ denotes the cost of:
  - Assigning a random test sample as $\omega_i$,
  - while the actual class of the sample is $\omega_j$.
  - For instance, $\lambda(\alpha_{\text{cancer}}|\omega_{\text{no\_cancer}})$ is the cost of diagnosing a patient without cancer as "having cancer".

### Expected Loss & Bayes Risk

- Expected Loss (Average Loss, Conditional Risk) 期望成本
  - We don't actually know the true class of $\omega_j$ for a random sample $\mathbf{x}$, so we use the Expected Loss, i.e., the "average loss".
  - We consider the average loss of classifying a random sample into $\omega_i$ by considering:
    - For all class $\omega_j \in \omega$, the loss of classifying $\omega_i$ into $\omega_j$, and
    - The probability that the random sample $\mathbf{x} \in \omega_j$, i.e., $P(\omega_j|\mathbf{x})$.

The expected loss of classifying a random sample $\mathbf{x}$ into $\omega_i$ is:

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) \cdot P(\omega_j|\mathbf{x})$$

where:

- $\lambda(\alpha_i|\omega_j)$ is the cost of classifying $\mathbf{x}$ into $\omega_i$ with $\mathbf{x}$ belonging to $\omega_j$ actually.
- $P(\omega_i|\mathbf{x})$ is the (posterior) probability that $\mathbf{x}$ belongs to class $\omega_j$.
  - Computed during Naïve Bayes Classification with $P(\omega_j)$ and $P(\mathbf{x}|\omega_j)$.
- Bayes Risk 贝叶斯风险
  - The modified measurement of the original Bayes Rule.

- Consider the importance of each error.
- Consider minimum loss, instead of maximum probability.
- Bayes Risk finds the action that gives the *minimum expected loss* classifying $\mathbf{x}$.

$$\begin{aligned}\alpha(\mathbf{x}) &= \text{argmin}_{\alpha_i \in A} R(\alpha_i|\mathbf{x}) \\ &= \text{argmin}_{\alpha \in A} \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) \cdot P(\omega_j|\mathbf{x})\end{aligned}$$

### Derivation: A 2-class problem.

#### Given

- The test sample $\mathbf{x}$.
- Two classes: $\omega = \{\omega_1, \omega_2\}$
- Calculated posterior probabilities during Naive Bayes:
  - $P(\omega_1|\mathbf{x})$, $P(\omega_2|\mathbf{x})$
- Loss Matrix:
  - $\begin{pmatrix}\lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22}\end{pmatrix}$
  - where $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$

  #### Do
- $\omega^* = \text{argmin}_{\alpha_i \in A} R(\alpha_i|\mathbf{x})$
- The condition of choosing $\alpha_1$ is:

$$\begin{aligned}& R(\alpha_1|\mathbf{x}) < R(\alpha_2|\mathbf{x}) \\ \iff & \lambda_{11}P(\omega_1|\mathbf{x}) + \lambda_{12}P(\omega_2|\mathbf{x}) < \lambda_{21}P(\omega_1|\mathbf{x}) + \lambda_{22}P(\omega_2|\mathbf{x}) \\ \iff & (\lambda_{21} - \lambda_{11}) \cdot P(\omega_1|\mathbf{x}) > (\lambda_{12} - \lambda_{22}) \cdot P(\omega_2|\mathbf{x}) \\ \iff & \frac{P(\omega_1|\mathbf{x})}{P(\omega_2|\mathbf{x})} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \\ \iff & \frac{P(\mathbf{x}|\omega_1) \cdot P(\omega_1)}{P(\mathbf{x}|\omega_2) \cdot P(\omega_2)} > \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \\ \iff & \frac{P(\mathbf{x}|\omega_1)}{P(\mathbf{x}|\omega_2)} > \frac{(\lambda_{12} - \lambda_{22}) \cdot P(\omega_2)}{(\lambda_{21} - \lambda_{11}) \cdot P(\omega_1)} = \theta\end{aligned}$$

## 2.3.3 Examples

### Minimum Prob. Error and Minimum Risk

Remark: The Gaussian Distribution.

$$x \in \mathbb{R} \sim Gaussian(\mu, \sigma): \ P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Given:

- Random distributions of samples in 2 classes $\omega_1$ and $\omega_2$ respectively.
  - $\omega_1$: $\mu = 0$, $\sigma = \frac{1}{\sqrt{2}} \implies P(x|\omega_1) = \frac{1}{\sqrt{\pi}} e^{-x^2}$
  - $\omega_2$: $\mu = 1$, $\sigma = \frac{1}{\sqrt{2}} \implies P(x|\omega_2) = \frac{1}{\sqrt{\pi}} e^{-(x-1)^2}$
- Loss Matrix:
  - $\begin{pmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1.0 \\ 0.5 & 0 \end{pmatrix}$

  Do:

## Minimum Error

The threshold value $x_0$ where the two distributions are equal

- i.e., minimum probability of error

$$P(x_0|\omega_1) = P(x_0|\omega_2)$$

$$\implies \frac{1}{\sqrt{\pi}} e^{-x_0^2} = \frac{1}{\sqrt{\pi}} e^{-(x_0-1)^2}$$

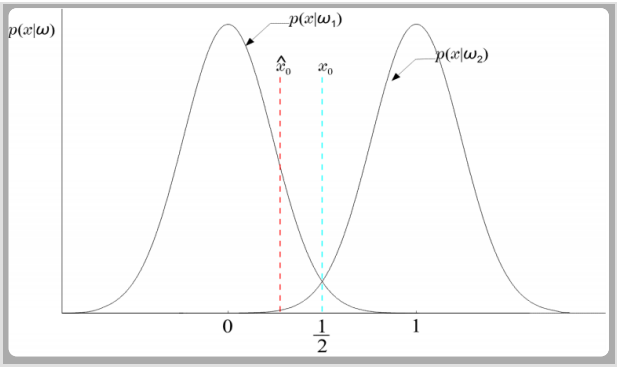
$$\implies x_0 = -x_0 + 1$$

$$\implies x_0 = \frac{1}{2}$$

## Minimum Risk

The threshold $\hat{x}_0$ for minimum $R(\alpha_i|x)$.

$$
\begin{aligned}
& R(\alpha_1|\hat{x}_0) = R(\alpha_2|\hat{x}_0) \\
\Longrightarrow\ & \lambda_{11} \cdot P(\omega_1|\hat{x}_0) + \lambda_{12} \cdot P(\omega_2|\hat{x}_0) = \lambda_{21} \cdot P(\omega_1|\hat{x}_0) + \lambda_{22} \cdot P(\omega_2|\hat{x}_0) \\
\Longrightarrow\ & (\lambda_{21} - \lambda_{11}) \cdot P(\omega_1|\hat{x}_0) = (\lambda_{12} - \lambda_{22}) \cdot P(\omega_2|\hat{x}_0) \\
\Longrightarrow\ & \frac{P(\omega_1|\hat{x}_0)}{P(\omega_2|\hat{x}_0)} = \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \\
\Longrightarrow\ & \frac{P(\hat{x}_0|\omega_1) \cdot P(\omega_1)}{P(\hat{x}_0|\omega_2) \cdot P(\omega_2)} = \frac{\lambda_{12} - \lambda_{22}}{\lambda_{21} - \lambda_{11}} \\
\Longrightarrow\ & \frac{P(\hat{x}_0|\omega_1)}{P(\hat{x}_0|\omega_2)} = \frac{(\lambda_{12} - \lambda_{22}) \cdot P(\omega_2)}{(\lambda_{21} - \lambda_{11}) \cdot P(\omega_1)} \\
\Longrightarrow\ & \frac{P(\hat{x}_0|\omega_1)}{P(\hat{x}_0|\omega_2)} = \frac{(1-0) \times \frac{1}{2}}{(0.5-0) \times \frac{1}{2}} = 2 \\
\Longrightarrow\ & P(\hat{x}_0|\omega_1) = 2P(\hat{x}_0|\omega_2) \\
\Longrightarrow\ & \frac{1}{\sqrt{\pi}} e^{-\hat{x}_0^2} = \frac{2}{\sqrt{\pi}} e^{-(\hat{x}_0-1)^2} \\
\Longrightarrow\ & -\hat{x}_0^2 = \ln 2 - \hat{x}_0^2 + 2\hat{x}_0 - 1 \\
\Longrightarrow\ & 2\hat{x}_0 = 1 - \ln 2 \\
\Longrightarrow\ & \hat{x}_0 = \frac{1 - \ln 2}{2}
\end{aligned}
$$




# 2.4 Discriminant Functions 判别函数

## 2.4.1 Definition of Discriminant Function

A Discriminant Function is a function $f$ that satisfies the following property:

- If:
  - $f(\cdot)$ monotonically increases, and
  - $\forall i \neq j,\ f\Big(P(\omega_i|\mathbf{x})\Big) > f\Big(P(\omega_j|\mathbf{x})\Big)$

    - Then:
        - $\mathbf{x} \leftarrow \omega_i$
- That is, the function is able to "tell", or "discriminate" a certain $\omega_i$ from others.
    - i.e., it separates $\omega_i$ and $\neg\omega_i$.

A sample usage of a discriminant function: Given two classes $\omega_i$ and $\omega_j$, define $g(\mathbf{x}) \equiv P(\omega_i|\mathbf{x}) - P(\omega_j|\mathbf{x}) = 0$.

- $g(\mathbf{x}) = 0$: Decision Surface;
- $g(\mathbf{x}) > 0$: Region $R_i$ where $P(\omega_i|\mathbf{x}) > P(\omega_j|\mathbf{x})$;
- $g(\mathbf{x}) < 0$: Region $R_i$ where $P(\omega_i|\mathbf{x}) < P(\omega_j|\mathbf{x})$;

## 2.4.2 Property of Discriminant Function

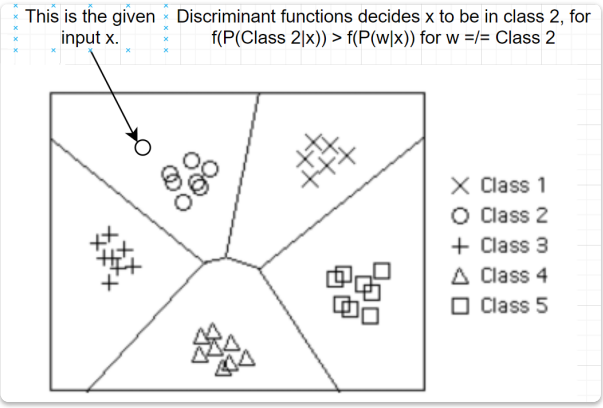
1. One function per class.
    1. A discriminant function is able to "tell" a certain one $\omega_i$ specifically for any input $x$.
2. Various discriminant functions $\rightarrow$ Identical classification results. 样式各异，结果相同。
    1. It is correct to say, the discriminant functions:
        1. *Preserves* the original monotonically increase of its inputs.
        2. But changes the changing rate by *processing* the inputs.
    2. i.e.,
        1. "$\forall i \neq j, f(g_i(x)) > f(g_j(x)) \wedge f \nearrow$ "and "$\forall i \neq j, g_i(x) > g_j(x)$" are equivalent in decision.
        2. Changing growth rate of input:
            1. $f(g_i(x)) = k \cdot g_i(x)$, a linear change.
            2. $f(g_i(x)) = \ln g_i(x)$, a log change, i.e., it grows, but slower as it proceed.
        3. Therefore, the discriminant function may vary, but the output is always the same.
3. Examples of discriminant functions:
    1. Minimum Risk: $g_i(x) = -R(\alpha_i|x) = -\lambda(\alpha_i|x) \times P(\omega_i|x)$, for $i \in [1, c]$
    2. Minimum Error Rate: $g_i(x) = P(\omega_i|x)$, for $i \in [1, c]$

## 2.4.3 Decision Region 决策区域

- $c$ discriminant functions $\implies$ $c$ decision regions
    - $g_i(x) \implies R_i \subset R^d, i \in [1, c]$
- One function per decision region that is distinct and mutual-exclusive.
    - A decision region is defined as: $R_i = \{x | x \in R^d : \forall i \neq j, g_i(x) > g_j(x)\}$, where
    - $\forall i \neq j, R_i \cap R_j = \emptyset$, and $\cup_{i=1}^{c} R_i = R^d$

## 2.4.4 Decision Boundaries 决策边界

- "Surface" in feature space, where ties occur among 2 or more largest discriminant functions.
- $x_0$ is on the decision boundary/surface if and only if
  - $\exists \omega_i, \omega_j \in \omega, g_i(x_0) = g_j(x_0)$.




# 2.5 Bayesian Classification for Normal Distributions

## 2.5.1 Multi-Dimensional Normal Distribution 高维正态分布

### 1-D Case 多类别，一维数据

- There are several classes:
  - Each class has its own distribution of data samples.
  - i.e., each class has its own $\mu$ and $\sigma$.
- For a specific class, there are plenty of data samples:
  - Each sample is a *scalar*, that is a $1 \times 1$ "matrix", which is a "plain number".
  - The samples follows a Normal Distribution.

Suppose data samples in a specific class $\omega_i$ conforms a normal distribution:

$$x \sim N(\mu_i, \sigma_i) : P(x|\omega_i) = \frac{1}{\sigma_i \sqrt{2\pi}} ^{\frac{(x-\mu_i)^2}{2\sigma^2}}$$

where:

- $\mu_i$ is the mean value, $\mu_i = E(x)$
- $\sigma_i^2$ is the variance, $\sigma_i^2 = E\Big[(x-\mu)^2\Big]$

### Multivariate Case 多类别，高维数据

- There are several classes:
  - Each class has its own distribution of data samples,
  - i.e., each class has its own $\mu$ and $\sigma$.
- For a specific class, there are plenty of data samples:
  - Each sample is a *vector*, that is a $d \times 1$ matrix, where $d$ is the dimension of data.

- The samples follow a $d$-dimensional Normal Distribution.

Suppose data samples in a specific class $\omega_i$ conforms a normal distribution:

$$\mathbf{x} \sim N(\mu_i, \Sigma_i): \ P(\mathbf{x}|\omega_i) = \frac{1}{|\Sigma_i|^{\frac{1}{2}} \cdot (2\pi)^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)}$$

Regular Variables:

- $d$-dimensional random variable: $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$
- $d$-dimensional mean vector: $\mu_i = \begin{bmatrix} \mu_{i1} \\ \mu_{i2} \\ \vdots \\ \mu_{id} \end{bmatrix} = \begin{bmatrix} E(x_{i1}) \\ E(x_{i2}) \\ \vdots \\ E(x_{id}) \end{bmatrix}$
- $d \times d$ covariate matrix: $\sigma_i = \begin{pmatrix} \sigma_{i11} & \sigma_{i12} & \cdots & \sigma_{i1d} \\ \sigma_{i21} & \sigma_{i22} & \cdots & \sigma_{i2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{id1} & \sigma_{id2} & \cdots & \sigma_{idd} \end{pmatrix} = E\Big[(\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^\top\Big]$

Explanation of exponent $-\frac{1}{2}(\mathbf{x} - \mu_i)^\top \Sigma_i^{-1}(\mathbf{x} - \mu_i)$:

- $(X - \mu_i)^\top = [(x_1 - \mu_{i1}) \quad (x_2 - \mu_{i2}) \quad \cdots \quad (x_d - \mu_{id})]$
- $\Sigma_i^{-1} = \begin{pmatrix} \sigma'_{i11} & \sigma'_{i12} & \cdots & \sigma'_{i1d} \\ \sigma'_{i21} & \sigma'_{i22} & \cdots & \sigma'_{i2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma'_{id1} & \sigma'_{id2} & \cdots & \sigma'_{idd} \end{pmatrix}$, the inverse of the covariance matrix.
- $(X - \mu_i) = \begin{bmatrix} x_1 - \mu_{i1} \\ x_2 - \mu_{i2} \\ \cdots \\ x_d - \mu_{id} \end{bmatrix}$

The exponent as a whole:

$$
\begin{aligned}
&-\frac{1}{2}(X-\mu_i)^\top \Sigma_i^{-1}(X-\mu_i) \\
&= -\frac{1}{2}\begin{bmatrix}(x_1-\mu_{i1}) & (x_2-\mu_{i2}) & \cdots & (x_d-\mu_{id})\end{bmatrix}
\begin{pmatrix}
\sigma'_{i11} & \sigma'_{i12} & \cdots & \sigma'_{i1d} \\
\sigma'_{i21} & \sigma'_{i22} & \cdots & \sigma'_{i2d} \\
\vdots & \vdots & \ddots & \vdots \\
\sigma'_{id1} & \sigma'_{id2} & \cdots & \sigma'_{idd}
\end{pmatrix}
\begin{bmatrix}x_1-\mu_{i1} \\ x_2-\mu_{i2} \\ \cdots \\ x_d-\mu_{id}\end{bmatrix} \\
&= -\frac{1}{2}\begin{bmatrix}a_1 & a_2 & \cdots a_d\end{bmatrix}
\begin{bmatrix}x_1-\mu_{i1} \\ x_2-\mu_{i2} \\ \cdots \\ x_d-\mu_{id}\end{bmatrix} \\
&= y \geq 0
\end{aligned}
$$

### Example: 2-D Case

$$
\mathbf{x} \sim N(\mu_i, \sigma_i) : P(\mathbf{x}|\omega_i = \frac{1}{|\Sigma_i|^{\frac{1}{2}} \cdot (2\pi)} e^{-\frac{1}{2}(x_1-\mu_{i1} \quad x_2-\mu_{i2})\Sigma_i^{-1}\begin{pmatrix}x_1-\mu_{i1} \\ x_2-\mu_{i2}\end{pmatrix}}
$$

where:

- 2-dimensional random variable: $\mathbf{x} = \begin{pmatrix}x_1 \\ x_2\end{pmatrix}$.
- 2-dimensional mean vector: $\mu_i = \begin{pmatrix}\mu_{i1} \\ \mu_{i2}\end{pmatrix}$
- $2 \times 2$ covariate matrix $\Sigma_i$:

$$
\begin{aligned}
\Sigma_i &= E\Big[(\mathbf{x}-\mu_i)(\mathbf{x}-\mu_i)^\top\Big] \\
&= E\Big[\begin{pmatrix}x_1-\mu_{i1} \\ x_2-\mu_{i2}\end{pmatrix}(x_1-\mu_{i1} \quad x_2-\mu_{i2})\Big] \\
&= E(\begin{bmatrix}(x_1-\mu_{i1})^2 & (x_1-\mu_{i1})(x_2-\mu_{i2}) \\ (x_1-\mu_{i1})(x_2-\mu_{i2}) & (x_2-\mu_{i2})^2\end{bmatrix}) \\
&= \begin{bmatrix}E\Big[(x_1-\mu_{i1})^2\Big] & E\Big[(x_1-\mu_{i1})(x_2-\mu_{i2})\Big] \\ E\Big[(x_1-\mu_{i1})(x_2-\mu_{i2})\Big] & E\Big[(x_2-\mu_{i2})^2\Big]\end{bmatrix} \\
&= \begin{bmatrix}\sigma_1^2 & \sigma \\ \sigma & \sigma_2^2\end{bmatrix}
\end{aligned}
$$

## 2.5.2 Minimum-error-rate classification

### Recall:

- Minimum-error-rate means that we ignore the "cost" of each decision.
- In other words, we only select the classes based on probabilities.

## Pattern of Discriminant Function

The discriminant function of MER classification could be given by:

$$\forall i \in [1, c] \cap \mathbb{N}^+,\ g_i(\mathbf{x}) = \ln P(\omega_i|\mathbf{x})$$

Namely,

$$g_i(\mathbf{x}) = \ln P(\omega_i|\mathbf{x})$$

$$\Longrightarrow g_i(\mathbf{x}) = \ln\Big[P(\mathbf{x}|\omega_\mathbf{i}) \cdot P(\omega_i)\Big]$$

$$\Longrightarrow g_i(\mathbf{x}) = \ln\Big[P(\mathbf{x}|\omega_i)\Big] + \ln\Big[P(\omega_i)\Big]$$

$$\Longrightarrow g_i(\mathbf{x}) = \ln\Big[\frac{1}{|\Sigma_i|^{\frac{1}{2}} \cdot (2\pi)^{\frac{d}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i)}\Big] + \ln\Big[P(\omega_i)\Big]$$

$$\Longrightarrow g_i(\mathbf{x}) = \ln\Big[\frac{1}{|\Sigma_i|^{\frac{1}{2}} \cdot (2\pi)^{\frac{d}{2}}}\Big] - \frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big]$$

$$\Longrightarrow g_i(\mathbf{x}) = \Big(-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\Sigma_i|\Big) - \frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big]$$

Here, $-\frac{d}{2}\ln(2\pi)$ is a constant, which could be ignored.

★ The discriminant function is then updated as:

$$g_i(\mathbf{x}) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big]$$

## Case I: $\Sigma_i = \sigma^2 I$

That is:

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_{|\omega|} = \sigma^2 I = \begin{bmatrix} \sigma^2 & 0 & \cdots & 0 \\ 0 & \sigma^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma^2 \end{bmatrix}$$

- All the classes have a *Common Covariance Matrix* of $\sigma^2 I$.
- The common covariate matrix is *isotropic (各向同性的)* with respect to any class.
  - i.e., the variance is the same in all directions.
  - i.e., no directional preference in the spread of distribution

Therefore, we have:

$$|\Sigma_i| = \sigma^{2d}$$

$$\Sigma_i^{-1} = \frac{1}{\sigma^2} I$$

This is the original discriminant function:

$$g_i(\mathbf{x}) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big]$$

Here, $-\frac{1}{2}\ln|\Sigma_i| = -\frac{1}{2}\ln|\sigma^2 I|$ is a constant, therefore can be ignored:

$$\begin{aligned}
g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}(\mathbf{x}-\mu_i)^\top \cdot (\frac{1}{\sigma^2} I) \cdot (\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{(\mathbf{x}-\mu_i)^\top (\mathbf{x}-\mu_i)}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{(\mathbf{x}^\top-\mu_i^\top)(\mathbf{x}-\mu_i)}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{\mathbf{x}^\top\mathbf{x} - \mathbf{x}^\top\mu_i - \mu_i^\top\mathbf{x} + \mu_i^\top\mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{\mathbf{x}^\top\mathbf{x} - 2\mu_i^\top\mathbf{x} + \mu_i^\top\mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{\|\mathbf{x}-\mu_i\|^2}{2\sigma^2} + \ln\Big[P(\omega_i)\Big]
\end{aligned}$$

Note: $\|\cdot\|^2$ denotes the *Euclidean Distance.*
Moreover $\mathbf{x}^\top\mathbf{x}$ is the same across all classes, therefore can be ignored:

$$\begin{aligned}
g_i(\mathbf{x}) &= -\frac{-2\mu_i^\top\mathbf{x} + \mu_i^\top\mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= \frac{\mu_i^\top\mathbf{x}}{\sigma^2} - \frac{\mu_i^\top\mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] \\
&= \Big(\frac{\mu_i}{\sigma^2}\Big)^\top\mathbf{x} + \Big(-\frac{\mu_i^\top\mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big]\Big)
\end{aligned}$$

Namely,

$$g_i(\mathbf{x}) = \mathbf{w}_i^\top\mathbf{x} + w_{i0}$$

where:

- $\mathbf{w}_i = \frac{\mu_i}{\sigma^2}$ is the weight vector, and

- $w_{i0} = -\frac{\mu_i^\top \mu_i}{2\sigma^2} + \ln\Big[P(\omega_i)\Big]$ is the threshold / bias scalar.

This is a Linear Discriminant Function. The decision surface is thus:

$$
\begin{aligned}
& g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \\
\implies & \mathbf{w}_i^\top \mathbf{x} + w_{i0} - (\mathbf{w}_j^\top \mathbf{x} + w_{j0}) = 0 \\
\implies & (\mathbf{w}_i - \mathbf{w}_j)^\top \mathbf{x} + (w_{i0} - w_{j0}) = 0 \\
\implies & (\frac{\mu_i - \mu_j}{\sigma^2})^\top \mathbf{x} + (w_{i0} - w_{j0}) = 0 \\
\implies & (\mu_i - \mu_j)^\top \mathbf{x} + \sigma^2(w_{i0} - w_{j0}) = 0 \\
\implies & (\mu_i - \mu_j)^\top \mathbf{x} + \sigma^2(\frac{-\mu_i^\top \mu_i}{2\sigma^2} - \frac{-\mu_j^\top \mu_j}{2\sigma^2} + \ln\Big[P(\omega_i)\Big] - \ln\Big[P(\omega_j)\Big]) = 0 \\
\implies & (\mu_i - \mu_j)^\top \mathbf{x} - \frac{1}{2}(\mu_i^\top \mu_i - \mu_j^\top \mu_j) + \sigma^2 \ln\Big[\frac{P(\omega_i)}{P(\omega_j)}\Big] = 0 \\
\implies & (\mu_i - \mu_j)^\top \mathbf{x} - \frac{1}{2}(\mu_i - \mu_j)^\top (\mu_i + \mu_j) + \sigma^2 \ln\Big[\frac{P(\omega_i)}{P(\omega_j)}\Big] = 0 \\
\implies & \mathbf{x} - \frac{1}{2}(\mu_i + \mu_j) + \sigma^2 \ln\Big[\frac{P(\omega_i)}{P(\omega_j)}\Big] \cdot \frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2} = 0 \\
\implies & \mathbf{x} = \frac{1}{2}(\mu_i + \mu_j) - \sigma^2 \ln\Big[\frac{P(\omega_i)}{P(\omega_j)}\Big] \cdot \frac{\mu_i - \mu_j}{\|\mu_i - \mu_j\|^2} \in \mathbb{R}^2
\end{aligned}
$$

## Case II: $\Sigma_i = \Sigma$

That is:

$$\Sigma_1 = \Sigma_2 = \cdots = \Sigma_{|\omega|} = \Sigma$$

- All the classes have a *Common Covariance Matrix* of $\Sigma$.
- More general than Case I.

This is the original discriminant function:

$$g_i(\mathbf{x}) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\mathbf{x} - \mu_i)^\top \Sigma_i^{-1}(\mathbf{x} - \mu_i) + \ln\Big[P(\omega_i)\Big]$$

Here, $-\frac{1}{2}\ln|\Sigma_i| = -\frac{1}{2}\ln|\Sigma|$ is constant, which could be ignored:

$$
\begin{aligned}
g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}(\mathbf{x}^\top - \mu_i^\top)\Big(\Sigma^{-1}\mathbf{x} - \Sigma^{-1}\mu_i\Big) + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}(\mathbf{x}^\top\Sigma^{-1}\mathbf{x} - \mathbf{x}^\top\Sigma^{-1}\mu_i - \mu_i^\top\Sigma^{-1}\mathbf{x} + \mu_i^\top\Sigma^{-1}\mu_i) + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}(\mathbf{x}^\top\Sigma^{-1}\mathbf{x} - 2\mu_i^\top\Sigma^{-1}\mathbf{x} + \mu_i^\top\Sigma^{-1}\mu_i) + \ln\Big[P(\omega_i)\Big]
\end{aligned}
$$

Here, $\mathbf{x}^\top\Sigma^{-1}\mathbf{x}$ is the same across all classes, thus can be ignored:

$$
g_i(\mathbf{x}) = \mu_i^\top\Sigma^{-1}\mathbf{x} + (-\frac{\mu_i^\top\Sigma^{-1}\mu_i}{2} + \ln\Big[P(\omega_i)\Big])
$$

Namely,

$$
g_i(\mathbf{x}) = \mathbf{w}_i^\top\mathbf{x} + w_{i0}
$$

where:

- $\mathbf{w}_i = \mu_i$ is the weight vector;
- $w_{i0} = -\frac{1}{2}\mu_i^\top\Sigma^{-1}\mu_i + \ln\Big[P(\omega_i)\Big]$ is the threshold / bias scalar.

## Case III: $\Sigma_i$ is arbitrary

In most cases, for each class $\omega_i$, $\Sigma_i$, the covariance/spread of data in this class is arbitrary. This is the original discriminant function:

$$
g_i(\mathbf{x}) = -\frac{1}{2}\ln|\Sigma_i| - \frac{1}{2}(\mathbf{x}-\mu_i)^\top\Sigma_i^{-1}(\mathbf{x}-\mu_i) + \ln\Big[P(\omega_i)\Big]
$$

We can derive that:

$$
\begin{aligned}
g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x}-\mu_i)^\top \Sigma_i^{-1}(\mathbf{x}-\mu_i) - \frac{1}{2}\ln|\Sigma_i| + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}(\mathbf{x}^\top - \mu_i^\top)(\Sigma_i^{-1}\mathbf{x} - \Sigma_i^{-1}\mu_i) + (-\frac{1}{2}|\Sigma_i| + \ln\Big[P(\omega_i)\Big]) \\
&= -\frac{1}{2}(\mathbf{x}^\top\Sigma_i^{-1}\mathbf{x} - \mathbf{x}^\top\Sigma_i^{-1}\mu_i - \mu_i^\top\Sigma_i^{-1}\mathbf{x} + \mu_i^\top\Sigma_i^{-1}\mu_i) + (-\frac{1}{2}|\Sigma_i| + \ln\Big[P(\omega_i)\Big]) \\
&= -\frac{1}{2}(\mathbf{x}^\top\Sigma_i^{-1}\mathbf{x} - 2\mu_i^\top\Sigma_i^{-1}\mathbf{x} + \mu_i^\top\Sigma_i^{-1}\mu_i) + (-\frac{1}{2}|\Sigma_i| + \ln\Big[P(\omega_i)\Big]) \\
&= -\frac{1}{2}\mathbf{x}^\top\Sigma_I^{-1}\mathbf{x} + \mu_i^\top\Sigma_i^{-1}\mathbf{x} - \frac{1}{2}\mu_i^\top\Sigma_i^{-1}\mu_i - \frac{1}{2}|\Sigma_i| + \ln\Big[P(\omega_i)\Big] \\
&= -\frac{1}{2}\mathbf{x}^\top\Sigma_i^{-1}\mathbf{x} + \mu_i^\top\Sigma_i^{-1}\mathbf{x} + (-\frac{\mu_i^\top\Sigma_i^{-1}\mu_i + |\Sigma_i|}{2} + \ln\Big[P(\omega_i)\Big]) \\
&= \mathbf{x}^\top(-\frac{1}{2}\Sigma_i^{-1})\mathbf{x} + (\mu_i^\top\Sigma_i^{-1})\mathbf{x} + (-\frac{\mu_i^\top\Sigma_i^{-1}\mu_i + |\Sigma_i|}{2} + \ln\Big[P(\omega_i)\Big])
\end{aligned}
$$

Namely,

$$
g_i(\mathbf{x}) = \mathbf{x}^\top\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^\top\mathbf{x} + w_{i0}
$$

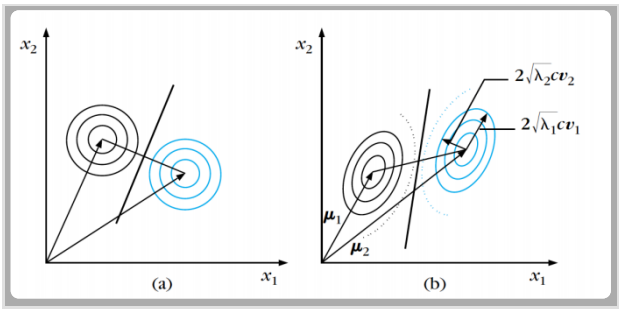
where:

- $\mathbf{W}_i = -\frac{1}{2}\Sigma_i^{-1}$ is the Quadratic matrix.
- $\mathbf{w}_i = \mu_i^\top\Sigma_i^{-1}$ is the weight vector.
- $w_{i0} = -\frac{\mu_i^\top\Sigma_i^{-1}\mu_i + |\Sigma_i|}{2} + \ln\Big[P(\omega_i)\Big]$ is the threshold / bias scalar.

## Summary

Again, for special covariance matrices:

- $\Sigma_i = \sigma^2 I$:
  - Assign $x$ to $\omega_i$ if there is a smaller Euclidean Distance: $d_{Euclidean} = \|X - \mu_i\|$
- $\Sigma_i = \Sigma$:
  - Assign $x$ to $\omega_i$ if there is a smaller Mahalanobis Distance: $d_{Mahalanobis} = \sqrt{(X-\mu_i)^\top\Sigma^{-1}(X-\mu_i)}$




# 2.5.3 Examples

Given:

- Two classes: $\omega_1, \omega_2$
- Prior probabilities:
  - $P(\omega_1) = P(\omega_2)$.
- Posterior probabilities:
  - $P(\mathbf{x}|\omega_1) \sim N(\mu_1, \Sigma)$
  - $P(\mathbf{x}|\omega_2) \sim N(\mu_2, \Sigma)$
  - where:
  - $\mu_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mu_2 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$
  - $\Sigma = \begin{pmatrix} 1.1 & 0.3 \\ 0.3 & 1.9 \end{pmatrix}$

  Do:
- Classify $\mathbf{x} = \begin{pmatrix} 1.0 \\ 2.2 \end{pmatrix}$ using Bayes Classification.

  Solve:

  Compute inverse of covariance matrix:

$$\begin{pmatrix} 1.1 & 0.3 \\ 0.3 & 1.9 \end{pmatrix}^{-1} = \frac{1}{1.1 \times 1.9 - 0.3^2} \begin{pmatrix} 1.9 & -0.3 \\ -0.3 & 1.1 \end{pmatrix}$$

$$= \frac{1}{2} \begin{pmatrix} 1.9 & -0.3 \\ -0.3 & 1.1 \end{pmatrix}$$

$$\Sigma^{-1} = \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix}$$

  Compute *Mahalanobis distance* using $\mu_1$ and $\mu_2$.

$$d^2(\mathbf{x}, \mu_i) = (\mathbf{x} - \mu_i)^\top \Sigma^{-1} (\mathbf{x} - \mu_i)$$

$$\mathbf{x} - \mu_1 = \begin{pmatrix} 1.0 \\ 2.2 \end{pmatrix}, \ \mathbf{x} - \mu_2 = \begin{pmatrix} -2.0 \\ -0.8 \end{pmatrix}$$

$$d^2(\mathbf{x}, \mu_1) = \begin{pmatrix} 1.0 & 2.2 \end{pmatrix} \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix} \begin{pmatrix} 1.0 \\ 2.2 \end{pmatrix} = 2.952$$

$$d^2(\mathbf{x}, \mu_2) = \begin{pmatrix} -2.0 & -0.8 \end{pmatrix} \begin{pmatrix} 0.95 & -0.15 \\ -0.15 & 0.55 \end{pmatrix} \begin{pmatrix} -2.0 \\ -0.8 \end{pmatrix} = 3.672$$

  Therefore, classify $\mathbf{x} \leftarrow \omega_1$, since $d^2(\mathbf{x}, \mu_2) > d^2(\mathbf{x}, \mu_1)$.

# 03_Estimation_Methods

# 3.1 $l$-norms and Distance Metrics

## 3.1.1 $l$-norms

$\mathbf{x}$ is a column vector in $\mathbb{R}^N$ space.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{bmatrix}$$

### $l_0$-norm

$$\|\mathbf{x}\|_0 \equiv \sum_{i=1}^{N} |x_i|^0$$

$$= |x_1|^0 + |x_2|^0 + \cdots + |x_N|^0$$

- $l_0$-norm is the *number of non-zero elements* in vector $X$.
  - Defined that $0^0 = 0$.
- Application:
  - $\|\mathbf{x}\|_0$ is very small $\iff$ The vector $\mathbf{x}$ is very sparse/shallow.
  - Minimize $\|\mathbf{x} - \mathbf{y}\|_0$ $\iff$ Minimize the difference between $\mathbf{x}$ and $\mathbf{y}$.

### $l_1$-norm (Taxicab Norm / Manhattan Norm)

$$\|\mathbf{x}\|_1 \equiv \sum_{i}^{N} |x_i|$$

$$= |x_1| + |x_2| + \cdots + |x_N|$$

- $l_1$-norm is the *sum of elements' absolute values* in vector $X$.
- Application:
  - Minimize $\|\mathbf{x}\|_1$ $\iff$ Minimize total value of non-zero element sums. Similar results as minimize $\|\mathbf{x}\|_0$.

### $l_2$-norm (Euclidean Norm)

$$\|\mathbf{x}\|_2 \equiv (\sum_{i=1}^{N} |x_i|^2)^{\frac{1}{2}}$$

$$= \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$$

- $l_2$-norm can be expressed as matrix format $\|\mathbf{x}\|_2 \equiv \sqrt{\mathbf{x}^\top \mathbf{x}}$
- Application:
  - Minimize $\|X\|_2 \iff$ Make matrix more sparse.

## $l_\infty$-norm (Maximum Norm)

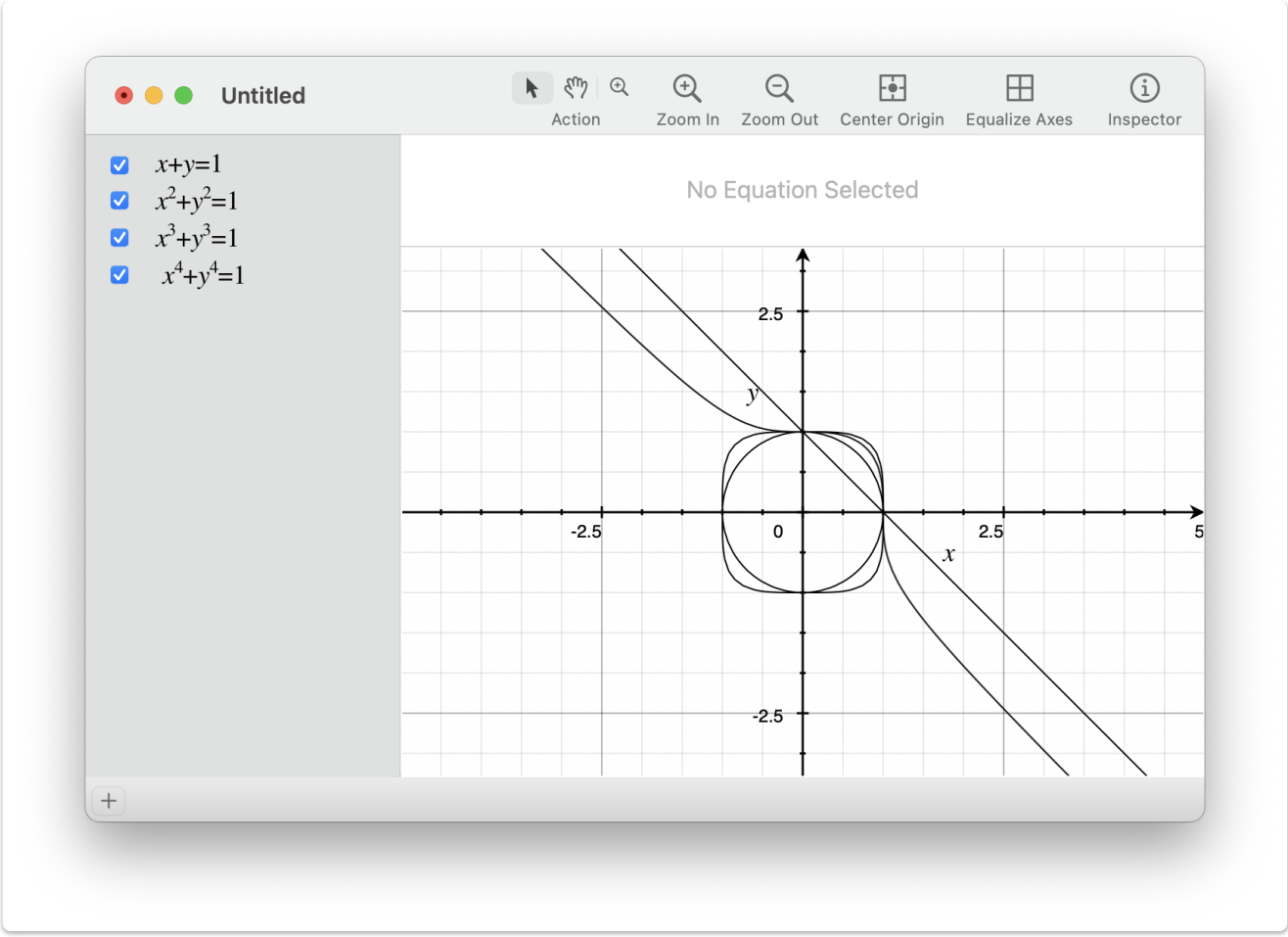
$$\|\mathbf{x}\|_\infty \equiv \max(|x_1|, |x_2|, \cdots, |x_N|)$$

- $l_\infty$-norm takes the *maximum of absolute values of elements* in vector $\mathbf{x}$.
- $l_\infty$-norm is also called
  - Maximum norm

## $l_p$-norm

$$\|X\|_p \equiv \left(\sum_{i=1}^{N} |x_i|^p\right)^{\frac{1}{p}} = (|x_1|^2 + |x_2|^2 + \cdots + |x_N|^p)^{\frac{1}{p}}$$

- $l_p$-norm is a general form of $l$-norm, where $p \geq 0$.
  - $p = 0$, $l_0$-norm,
  - $p = 1$, $l_1$-norm,
  - $p = 2$, $l_2$-norm,
  - ...,
  - $p \to \infty$, $l_\infty$-norm.




## 3.1.2 Distance Metrics

### Euclidean Distance

#### Given

- Two datasets $\mathbf{x}, \mathbf{y}$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{bmatrix}, \ \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix} \in \mathbb{R}^N$$

#### Do

Euclidean Distance:

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_N - y_N)^2}$$

$$= \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

$$= \sqrt{(\mathbf{x} - \mathbf{y})^\top (\mathbf{x} - \mathbf{y})}$$

- The straight-line distance between X and Y.

- Also called $L_2$ distance.

### Mahalanobis Distance

Given

- An observation.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_N \end{bmatrix}$$

- A set of observations with

  - mean $\mu = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_N \end{bmatrix}$

  - Covariance matrix $\Sigma$

  Do

Mahalanobis Distance

$$d_M(\mathbf{x}, \mu) = \sqrt{(\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu)}$$

- It is a measure of distance between:
  - a point, and
  - a distribution
- It reverts to Euclidean Distance when $\Sigma = I$.

## 3.2 Parameter Estimation

Recall the Bayes Formula:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{P(\mathbf{x})}$$

All we have initially are the training samples.

- We don't directly "know" the prior & posterior probabilities.
- Therefore, we need to retrieve prior probability $P(\omega_j)$ and posterior probability $P(X|\omega_j)$ from training samples.

Collect training samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ distributed according to the unknown $P(\mathbf{x}|\omega_j)$.

Assumed that these samples are Independent and identically distributed (i.i.d.).

- Independent: $\mathbf{x}_i$ and $\mathbf{x}_j$ does not influence each other.
- Identical: $\mathbf{x}_i \neq \mathbf{x}_j,\ \forall i \neq j$.

Our next goal is to estimate $\mu_j$ and $\Sigma_j$, hyper parameters of the posterior distribution $P(\mathbf{x}|\omega_j)$.

- Parametric Form
  - Maximum Likelihood Estimation (MLE)
  - Bayesian Estimation (BE)
- Nonparametric Form

# 3.3 Maximum Likelihood Estimation

## 3.3.1 Find the best $\theta$ : Log-Likelihood.

Given

- The set of i.i.d. training Examples:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$$

- where:
  - $\forall k = 1, 2, \cdots, N,\ \mathbf{x}_k \sim P(\mathbf{x}|\theta)$
- $\theta$ are the parameters to be estimated.

  Do

  We derive the objective function:

$$p(\mathcal{X}|\theta) \equiv p(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N|\theta)$$

$$= \prod_{k=1}^{N} p(\mathbf{x}_k|\theta)$$

  $p(X|\theta)$ is the Likelihood of $\theta$ with respect to $X$.

To find a best $\theta$, we derive a Maximum Likelihood estimation:

$$\hat{\theta}_{ML} = \text{argmax}_\theta\, p(\mathcal{X}|\theta)$$

$$= \text{argmax}_\theta \prod_{k=1}^{N} p(\mathbf{x}_k|\theta)$$

That is, we want to find the $\theta$ that gives the Maximum Likelihood on $\mathcal{X}$.

### Log-likelihood

For optimization purposes, we derive a log-likelihood function that preserves the monotonicity of the original MLE.

$$L(\theta) = \ln p(\mathcal{X}|\theta)$$

$$= \ln \prod_{k=1}^{N} p(\mathbf{x}_k|\theta)$$

$$= \sum_{k=1}^{N} \ln p(\mathbf{x}_k|\theta)$$

As the monotonicity is preserved, we would derive that

$$\hat{\theta}_{ML} = \text{argmax}_\theta \, p(\mathcal{X}|\theta)$$

$$= \text{argmax}_\theta \, L(\theta)$$

$$= \text{argmax}_\theta \sum_{k=1}^{N} \ln \, p(\mathbf{x}_k|\theta)$$

Equivalently, we find the $\theta$ that gives the *maximum* $L(\theta)$ now.

To find the $\theta$ that maximizes $L(\theta)$, we find:

$$\hat{\theta}_{ML} : \frac{\partial L(\theta)}{\partial \theta} = 0$$

That is,

$$\hat{\theta}_{ML} : \sum_{k=1}^{N} \frac{\partial \Big[\ln p(x_k|\theta)\Big]}{\partial \theta} = 0$$

## 3.3.2 $\mu$ unknown, $\Sigma$ known; $\theta = \{\mu\}$.

### Univariate & Multivariate Case ($x \in \mathbb{R}^{N^+}$)

The distribution:

$$p(\mathbf{x}_k|\mu) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_k-\mu)^\top \Sigma^{-1}(\mathbf{x}_k-\mu)}$$

The log likelihood:

$$
\begin{aligned}
\ln p(\mathbf{x}_k|\mu) &= -\frac{1}{2}\ln\Big((2\pi)^d|\Sigma|\Big) - \frac{1}{2}(\mathbf{x}_k-\mu)^\top\Sigma^{-1}(\mathbf{x}_k-\mu) \\
&= -\frac{1}{2}\Big((2\pi)^d|\Sigma|\Big) - \frac{1}{2}(\mathbf{x}_k^\top-\mu^\top)\Sigma^{-1}(\mathbf{x}_k-\mu) \\
&= -\frac{1}{2}\Big((2\pi)^d|\Sigma|\Big) - \frac{1}{2}(\mathbf{x}_k^\top\Sigma^{-1}-\mu^\top\Sigma^{-1})(\mathbf{x}_k-\mu) \\
&= -\frac{1}{2}\Big((2\pi)^d|\Sigma|\Big) - \frac{1}{2}(\mathbf{x}_k^\top\Sigma^{-1}\mathbf{x}_k-\mathbf{x}_k^\top\Sigma^{-1}\mu-\mu^\top\Sigma^{-1}\mathbf{x}_k+\mu^\top\Sigma^{-1}\mu) \\
&= -\frac{1}{2}\Big((2\pi)^d|\Sigma|\Big) - \frac{1}{2}\mathbf{x}_k^\top\Sigma^{-1}\mathbf{x}_k+\mu^\top\Sigma^{-1}\mathbf{x}_k-\frac{1}{2}\mu^\top\Sigma^{-1}\mu
\end{aligned}
$$

The constant terms are:

- $-\frac{1}{2}\Big((2\pi)^d|\Sigma|\Big)$
- $-\frac{1}{2}\mathbf{x}_k^\top\Sigma^{-1}\mathbf{x}_k$, since $\mathbf{x}_k$ is pre-defined.

Therefore:

$$
\begin{aligned}
\frac{\partial}{\partial\mu}\ln p(\mathbf{x}_k|\mu) &= \frac{\partial}{\partial\mu}(\mu^\top\Sigma^{-1}\mathbf{x}_k-\frac{1}{2}\mu^\top\Sigma^{-1}\mu) \\
&= \Sigma^{-1}\mathbf{x}_k-\Sigma^{-1}\mu \\
&= \Sigma^{-1}(\mathbf{x}_k-\mu)
\end{aligned}
$$

As we required

$$\frac{\partial}{\partial \mu} L(\mu) = 0$$

$$\implies \sum_{k=1}^{N} \frac{\partial}{\partial \mu} \ln p(\mathbf{x}_k|\mu) = 0$$

$$\implies \sum_{k=1}^{N} \Sigma^{-1}(\mathbf{x}_k - \mu) = 0$$

$$\implies \Sigma^{-1} \sum_{k=1}^{N} (\mathbf{x}_k - \mu) = 0$$

$$\implies \left(\sum_{k=1}^{N} \mathbf{x}_k\right) - N\mu = 0$$

$$\implies N\mu = \sum_{k=1}^{N} \mathbf{x}_k$$

$$\implies \mu = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k$$

★ That is, the $\mu$ that produces the maximum likelihood over the dataset is:

$$\hat{\mu}_{ML} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{x}_k$$

## 3.3.3 $\mu$ unknown, $\Sigma$ unknown; $\theta = \{\mu, \Sigma\}$

### Univariate Case ($x \in \mathbb{R}$)

$$p(x_k|\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_k - \mu)^2}{2\sigma^2}}$$

where:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} \mu \\ \sigma^2 \end{bmatrix}$$

The log likelihood:

$$\ln p(x_k|\theta) = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(x_k - \mu)^2$$

$$= -\frac{1}{2}\ln(2\pi\theta_2) - \frac{1}{2\theta_2}(x_k - \theta_1)^2$$

Therefore:

$$\frac{\partial}{\partial \theta} \ln p(x_k|\theta) = \begin{bmatrix} \frac{\partial L(\theta)}{\partial \theta_1} \\ \frac{\partial L(\theta)}{\partial \theta_2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{(x_k - \theta_1)}{\theta_2} \\ -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \end{bmatrix}$$

Again, to find the $\theta$ that minimizes the MLE, we let

$$\frac{\partial}{\partial \theta} L(\theta) = 0$$

$$\implies \sum_{k=1}^{N} \frac{\partial}{\partial \theta} \ln p(x_k|\theta) = 0$$

$$\implies \begin{bmatrix} \sum_{k=1}^{N} \frac{x_k - \theta_1}{\theta_2} \\ \sum_{k=1}^{N} \left( -\frac{1}{2\theta_2} + \frac{(x_k - \theta_1)^2}{2\theta_2^2} \right) \end{bmatrix} = 0$$

*For the first term:*

$$\sum_{k=1}^{N} \frac{x_k - \theta_1}{\theta_2} = 0$$

$$\implies \sum_{k=1}^{N} (x_k - \theta_1) = 0$$

$$\implies (\sum_{k=1}^{N} x_k) - N\theta_1 = 0$$

$$\implies \theta_1 = \frac{1}{N} \sum_{k=1}^{N} x_k$$

★ That is, the optimal $\mu$ is:

$$\hat{\mu}_{ML} = \frac{1}{N} \sum_{k=1}^{N} x_k$$

*For the second term:*

$$\sum_{k=1}^{N} -\frac{1}{2\theta_2} + \frac{(x_k-\theta_1)^2}{2\theta_2^2} = 0$$

$$\Longrightarrow \sum_{k=1}^{N} -\theta_2 + (x_k-\theta_1)^2 = 0$$

$$\Longrightarrow -N\theta_2 + \sum_{k=1}^{N}(x_k-\theta_1)^2 = 0$$

$$\Longrightarrow \theta_2 = \frac{1}{N}\sum_{k=1}^{N}(x_k-\theta_1)^2$$

★ That is, the optimal $\sigma^2$ is:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{k=1}^{N}(x_k-\hat{\mu})^2$$

### Multivariate Case ($x \in \mathbb{R}^D, D > 1$)

$$p(\mathbf{x}_k|\theta) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}_k-\mu)^\top \Sigma^{-1}(\mathbf{x}_k-\mu)}$$

where:

$$\theta = \begin{bmatrix}\mu \\ \Sigma\end{bmatrix} = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix}$$

★ Similarly, we have the optimized parameters as:

$$\hat{\mu} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{k=1}^{N}(\mathbf{x}_k-\hat{\mu})(\mathbf{x}_k-\hat{\mu})^\top$$

# 3.4 Bayesian Estimation

## 3.4.0 Difference between BE and MLE

- In ML estimation, $\theta$ was considered a parameter with a fixed value.
- In Bayesian estimation however, $\theta$ is considered an unknown random vector.
  - which is described by a P.D.F $p(\theta)$.

## 3.4.1 Find the best $\theta$ : Bayes Formula

### Given

- The set of i.i.d. training examples:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$$

- where:
  - $\forall k = 1, 2, \cdots, N,\ \mathbf{x}_k \sim P(\mathbf{x}|\theta)$
- $\theta$ are the parameters to be estimated.

### Do

As $\theta$ is regarded to be random, we compute the maximum of $p(\theta|\mathcal{X})$. From Bayes formula, we know that:

$$p(\theta|\mathcal{X}) = \frac{p(\mathcal{X}|\theta) \cdot P(\theta)}{P(\mathcal{X})}$$

We find the with the best Maximum Aposterior Probability.

$$\begin{aligned}\hat{\theta}_{MAP} &= \text{argmax}_\theta\, p(\theta|\mathcal{X}) \\ &= \text{argmax}_\theta\, p(\mathcal{X}|\theta) \cdot P(\theta)\end{aligned}$$

Similarly, find the max:

$$\frac{\partial}{\partial\theta} \ln p(\theta|\mathcal{X}) = 0$$

$$\implies \frac{\partial}{\partial\theta} \ln\Big(p(\mathcal{X}|\theta) \cdot P(\theta)\Big) = 0$$

## 3.4.2 $\mu$ unknown, $\sigma$ known

### Univariate case

$$p(x_k|\mu) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_k-\mu)^2}{2\sigma^2}}$$

where $\mu$ conforms a normal distribution:

$$\mu \sim N(\mu_0, \sigma_0) :\ p(\mu) = \frac{1}{\sigma_0\sqrt{2\pi}} e^{-\frac{(\mu-\mu_0)^2}{2\sigma_0^2}}$$

We could therefore know that:

$$\ln p(x_k|\mu) = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}(x_k - \mu)^2$$

$$\ln P(\mu) = -\frac{1}{2}\ln(2\pi\sigma_0^2) - \frac{1}{2\sigma^2}(\mu - \mu_0)^2$$

Therefore,

$$\frac{\partial}{\partial \mu} \ln p(x_k|\mu) = \frac{(x_k - \mu)}{\sigma^2}$$

$$\frac{\partial}{\partial \mu} \ln P(\mu) = \frac{(\mu - \mu_0)}{\sigma_0^2}$$

Therefore, the optimal $\mu$ could be obtained by:

$$\frac{\partial}{\partial \mu} \ln\Big(p(\mathcal{X}|\mu) \cdot P(\mu)\Big) = 0$$

$$\Longrightarrow \frac{\partial}{\partial \mu} \ln\Big[\prod_{k=1}^{N} p(\mathbf{x}_k|\mu) \cdot P(\mu)\Big] = 0$$

$$\Longrightarrow \frac{\partial}{\partial \mu} \Big[\sum_{k=1}^{N} \ln p(\mathbf{x}_k|\mu)\Big] + \frac{\partial}{\partial \mu} \ln P(\mu) = 0$$

$$\Longrightarrow \Big[\sum_{k=1}^{N} \frac{\partial}{\partial \mu} \ln p(\mathbf{x}_k|\mu)\Big] + \frac{\partial}{\partial \mu} \ln P(\mu) = 0$$

$$\Longrightarrow (\sum_{k=1}^{N} \frac{x_k - \mu}{\sigma^2}) - (\frac{\mu - \mu_0}{\sigma_0^2}) = 0$$

$$\Longrightarrow \frac{1}{\sigma^2}(\sum_{k=1}^{N} x_k) - \frac{N}{\sigma^2}\mu - \frac{1}{\sigma_0^2}\mu + \frac{1}{\sigma_0^2}\mu_0 = 0$$

$$\Longrightarrow \sigma_0^2(\sum_{k=1}^{N} x_k) - (\sigma_0^2 N + \sigma^2)\mu + \sigma^2\mu_0 = 0$$

$$\Longrightarrow \sigma_0^2(\sum_{k=1}^{N} x_k) + \sigma^2\mu_0 = (\sigma_0^2 N + \sigma^2)\mu$$

$$\Longrightarrow \mu = \frac{\sigma_0^2(\sum_{k=1}^{N} x_k) + \sigma^2\mu_0}{\sigma_0^2 N + \sigma^2}$$

$$\Longrightarrow \mu = \frac{\frac{\sigma_0^2}{\sigma_2}\sum_{k=0}^{N} x_k + \mu_0}{\frac{\sigma_0^2}{\sigma^2}N + 1}$$

★ That is, the optimal $\mu$ by Bayesian Estimation is:

$$\hat{\mu}_{BE} = \frac{\frac{\sigma_0^2}{\sigma_2}\sum_{k=0}^{N} x_k + \mu_0}{\frac{\sigma_0^2}{\sigma^2}N + 1}$$

# 04_Dimension_Reduction

# 4.0 A Quick View

### What does it do?

Dimension Reduction:

- Reduces the dimension of data.
  - Changes the data representation into a lower-dimensional one.
- It preserves the structure of the data.
- Usually unsupervised.

### Why do we need DR?

- Computation Complexity
- Pre-processing stage before further learning
- Data Visualization
- Data Interpretation

# 4.1 Singular Value Decomposition (SVD) 奇异值分解

## 4.1.0 Why SVD?

- *Redundancy* within dimensions of a single data sample. 多维间存在冗余信息
  - In a set of high-dimensional data samples, not all dimensions are useful.
  - There may be redundancies among some dimensions.
    - That is, some dimensions are highly related.
    - e.g., Suppose in a data set, for most data samples, $x_2 = 2x_1 + 3$. Therefore we only need $x_1$ since it could already describe $x_2$ with itself. This creates a redundancy.
  - SVD picks out main features, and project data into lower dimensions to remove such redundancies.
- Existence of *noise* samples. 存在噪声数据
  - Among data, smaller eigenvalues always comes with unimportant features.
  - By ignoring these data, we could reduce the noise when we are reducing data dimension.
  - That's why, during the process of SVD, we need to *sort* the eigenvalues.

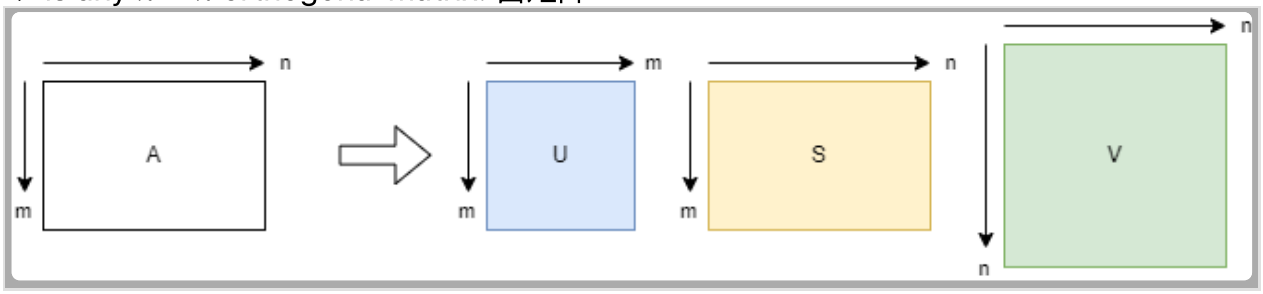
## 4.1.1 Definition

- Suppose that matrix $A \in \mathbb{R}^{m \times n}$ contains a set of training data.
  - $m$: Dimensions within a data sample.
    - That is, a column vector of $A$ represents a data sample.
  - $n$: The number of data samples.
  - In fact, the role of $m \times n$ could be reversed.
    - In the current version, $A$ is a "fat" matrix; In the reversed version, $A$ is a "tall" matrix.
- The Singular Value Decomposition process could be described as follows.

$$A_{m \times n} = U_{m \times m} S_{m \times n} V^{\top}_{n \times n}$$

where,

- $A$ is any $m \times n$ matrix.
- $U$ is any $m \times m$ orthogonal matrix. 酉矩阵、正交矩阵
  - $U^{\top} = U^{-1}$
  - $UU^{\top} = U^{\top}U = I$
- $S$ is any $m \times n$ diagonal matrix. 对角矩阵
  - Singular values $\sigma_1 > \sigma_2 > \cdots > \sigma_{\min(m,n)} > 0$ is the *main* diagonal of $S$.
    - $S = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_m & \cdots & 0 \end{bmatrix}$
  - $\sigma_1^2 > \sigma_2^2 > \cdots > \sigma_{\min(m,n)^2}$ are the eigenvalues of $AA^{\top}$ and $A^{\top}A$.
- $V$ is any $n \times n$ orthogonal matrix. 酉矩阵




## Left Singular Matrix $U$

- When we look at the Left Singular Matrix $U$, we pay attention to its *Column Vectors*.
  - Since $U \in \mathbb{R}^{m \times m}$, it has $m$ column vectors. They are the **Left Singular Vectors**.
  - These column vectors represents the *Main Directions* of the *Row Space* of matrix $A$.
    - Row space: The space of Row Vectors, consider the row number, i.e. the height of the matrix.
  - In other words, $U$ denotes the relationships among the dimensions in data samples.
- How exactly?

- Each column vector of $U$ is a unit vector, and $U$'s column vectors are all perpendicular 垂直 to each other, with dot product of $0$.
- Each column vector of $U$ represents a *co-tendency* among all the features within a data sample.
- The more left the column vector is located, the more important it is.

## 4.1.2 Calculation Procedures

### Problem Setup

#### Given

- A matrix $A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix}$.

#### Do

- Find $U$, $S$, and $V$ for Singular Value Decomposition.

### Basic Knowledge

$$
\begin{aligned}
AA^\top &= \left(USV^\top\right)\left(USV^\top\right)^\top \\
&= \left(USV^\top\right)\left(VS^\top U^\top\right) \\
&= US\left(V^\top V\right)S^\top U^\top \\
&= USS^\top U
\end{aligned}
$$

$$
\begin{aligned}
A^\top A &= \left(USV^\top\right)^\top\left(USV^\top\right) \\
&= \left(VS^\top U^\top\right)\left(USV^\top\right) \\
&= VS^\top\left(U^\top U\right)SV^\top \\
&= VS^\top SV^\top
\end{aligned}
$$

### Step 1. Calculate $AA^\top$ and $A^\top A$

Known that:

$$
A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix},\ A^\top = \begin{bmatrix} 2 & -1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}
$$

Therefore, we could get:

$$
AA^\top = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix}\begin{bmatrix} 2 & -1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & -2 \\ -2 & 5 \end{bmatrix}
$$

$$
A^\top A = \begin{bmatrix} 2 & -1 \\ 0 & 2 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix} = \begin{bmatrix} 5 & -2 & 2 \\ -2 & 4 & 0 \\ 2 & 0 & 1 \end{bmatrix}
$$

## Step 2. Eigenvalues and $S$

As we obtained $AA^\top$ and $A^\top A$, we can get their common eigenvalues, and construct $S$ matrix.

- The eigenvalues $AA^\top$ and $A^\top A$ are essentially the same, except for the zero-eigenvalue.

From the definition of Eigen Values:

$$AA^\top = \lambda I$$

where $\lambda$ is the eigenvalue of $AA^\top$. Calculate the eigen values:

$\implies |AA^\top - \lambda I| = 0$

$\implies \left| \begin{pmatrix} 5 & -2 \\ -2 & 5 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right| = 0$

$\implies \begin{vmatrix} 5-\lambda & -2 \\ -2 & 5-\lambda \end{vmatrix} = 0$

$\implies (5-\lambda)^2 - 4 = 0$

$\implies \lambda^2 - 10\lambda + 21 = 0$

$\implies (\lambda - 3)(\lambda - 7) = 0$

$\implies \begin{cases} \lambda_1 = 7 \\ \lambda_2 = 3\text{'} \end{cases} \begin{cases} \sigma_1 = \sqrt{\lambda_1} = \sqrt{7} \\ \sigma_2 = \sqrt{\lambda_2} = \sqrt{3} \end{cases}$

Calculate Eigenvalues for $A^\top A$:

$|A^\top A - \lambda I| = 0$

$\implies \left| \begin{pmatrix} 5 & -2 & 2 \\ -2 & 4 & 0 \\ 2 & 0 & 1 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right| = 0$

$\implies \begin{vmatrix} 5-\lambda & -2 & 2 \\ -2 & 4-\lambda & 0 \\ 2 & 0 & 1-\lambda \end{vmatrix} = 0$

$\implies (5-\lambda)\Big[(4-\lambda)(1-\lambda)\Big] - (-2)\Big[-2(1-\lambda)\Big] + 2\Big[-2(4-\lambda)\Big] = 0$

$\implies (5-\lambda)(\lambda^2 - 5\lambda + 4) - 4(1-\lambda) - 4(4-\lambda) = 0$

$\implies (5-\lambda)(\lambda^2 - 5\lambda + 4) + 8\lambda - 20 = 0$

$\implies (5\lambda^2 - 25\lambda + 20 - \lambda^3 + 5\lambda^2 - 4\lambda) + 8\lambda - 20 = 0$

$\implies (-\lambda^3 + 10\lambda^2 - 29\lambda + 20) + 8\lambda - 20 = 0$

$$\implies -\lambda^3 + 10\lambda^2 - 21\lambda = 0$$

$$\implies (\lambda^2 - 10\lambda + 21)\lambda = 0$$

$$\implies (\lambda - 3)(\lambda - 7)(\lambda - 0) = 0$$

$$\implies \begin{cases} \lambda_1 = 7 \\ \lambda_2 = 3, \\ \lambda_3 = 0 \end{cases} \begin{cases} \sigma_1 = \sqrt{\lambda_1} = \sqrt{7} \\ \sigma_2 = \sqrt{\lambda_2} = \sqrt{3} \\ \sigma_3 = \sqrt{\lambda_3} = 0 \end{cases}$$

Therefore, the diagonal matrix $S$ would be:

$$S = \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \end{pmatrix} = \begin{pmatrix} \sqrt{7} & 0 & 0 \\ 0 & \sqrt{3} & 0 \end{pmatrix}$$

## Step 3. Find $U$

We need to find $U$ using the eigenvalues we obtained from Step 2. Again, by the property of eigenvalues of a matrix:

$$\forall x \in \mathbb{R}^m,\ (AA^\top - \lambda I)x = 0$$

For $\lambda_1 = 7$:
$(AA^\top - \lambda I)x_1 = 0$

$$\implies \left( \begin{pmatrix} 5 & -2 \\ -2 & 5 \end{pmatrix} - 7 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) x_1 = 0$$

$$\implies \begin{pmatrix} -2 & -2 \\ -2 & -2 \end{pmatrix} x_1 = 0$$

$$\implies \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} x_1 = 0 \ \textit{(row 2 - row 1)}$$

$$\implies x_1 = \begin{pmatrix} a \\ -a \end{pmatrix}$$

$$\implies u_1 = \frac{x_1}{\|x_1\|} = \frac{1}{\sqrt{x_1^\top x_1}} x_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

For $\lambda_2 = 3$:
$(AA^\top - \lambda I)x_2 = 0$

$$\implies \left( \begin{pmatrix} 5 & -2 \\ -2 & 5 \end{pmatrix} - 3 \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \right) x_2 = 0$$

$$\implies \begin{pmatrix} 2 & -2 \\ -2 & 2 \end{pmatrix} x_2 = 0 \ \textit{(row 2 + row 1)}$$

$$\implies \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix} x_2 = 0$$

$$\implies x_2 = \begin{pmatrix} a \\ a \end{pmatrix}$$

$$\implies u_2 = \frac{x_2}{\|x_2\|} = \frac{1}{\sqrt{x_2^\top x_2}} x_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

Construct matrix $U$:

$$U = \begin{pmatrix} | & | \\ u_1 & u_2 \\ | & | \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}$$

For $\lambda_3 = 0$

## Step 4. Finding $V$

For $\lambda_1 = 7$:
$(A^\top A - \lambda_1 I)x_3 = 0$

$$\implies \left( \begin{pmatrix} 5 & -2 & 2 \\ -2 & 4 & 0 \\ 2 & 0 & 1 \end{pmatrix} - 7 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) x_3 = 0$$

$$\implies \begin{pmatrix} -2 & -2 & 2 \\ -2 & -3 & 0 \\ 2 & 0 & -6 \end{pmatrix} x_3 = 0$$

$$\implies \begin{cases} -2x_{31} - 3x_{32} = 0 \\ 2x_{31} - 6x_{33} = 0 \end{cases}$$

$$\implies \begin{cases} x_{32} = -\frac{2}{3} x_{31} \\ x_{33} = \frac{1}{3} x_{31} \end{cases}$$

$$\implies x_3 = \begin{pmatrix} x_{31} \\ -\frac{2}{3} x_{31} \\ \frac{1}{3} x_{31} \end{pmatrix} = \begin{pmatrix} 3a \\ -2a \\ a \end{pmatrix}, \|x_3\| = a\sqrt{3^2 + (-2)^2 + 1^2} = \sqrt{14} \cdot a$$

$$\implies v_1 = \frac{x_3}{\|x_3\|} = \begin{pmatrix} \frac{3}{\sqrt{14}} \\ \frac{-2}{\sqrt{14}} \\ \frac{1}{\sqrt{14}} \end{pmatrix}$$

For $\lambda_2 = 3$:
$(A^\top A - \lambda_2 I)x_4 = 0$

$$\implies \left( \begin{pmatrix} 5 & -2 & 2 \\ -2 & 4 & 0 \\ 2 & 0 & 1 \end{pmatrix} - 3 \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \right) x_4 = 0$$

$$\implies \begin{pmatrix} 2 & -2 & 2 \\ -2 & 1 & 0 \\ 2 & 0 & -2 \end{pmatrix} x_4 = 0$$

$$\implies \begin{cases} -2x_{41} + x_{42} = 0 \\ 2x_{41} - 2x_{43} = 0 \end{cases}$$

$$\implies \begin{cases} x_{42} = 2x_{41} \\ x_{43} = x_{41} \end{cases}$$

$$\implies x_4 = \begin{pmatrix} x_{41} \\ 2x_{41} \\ x_{41} \end{pmatrix} = \begin{pmatrix} a \\ 2a \\ a \end{pmatrix}, ||x_4|| = a\sqrt{1^2 + 2^2 + 1^2} = \sqrt{6} \cdot a$$

$$\implies v_2 = \frac{x_4}{\|x_4\|} = \begin{pmatrix} \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{6}} \end{pmatrix}$$

For $\lambda_3 = 0$:
$(A^\top A - 0I)x_5 = 0$

$$\implies \begin{pmatrix} 5 & -2 & 2 \\ -2 & 4 & 0 \\ 2 & 0 & 1 \end{pmatrix} x_5 = 0$$

$$\implies \begin{cases} -2x_{51} + 4x_{52} = 0 \\ 2x_{51} + x_{53} = 0 \end{cases}$$

$$\implies \begin{cases} x_{52} = \frac{1}{2}x_{51} \\ x_{53} = -2x_{51} \end{cases}$$

$$\implies x_5 = \begin{pmatrix} x_{51} \\ \frac{1}{2}x_{51} \\ -2x_{51} \end{pmatrix} = \begin{pmatrix} a \\ \frac{1}{2}a \\ -2a \end{pmatrix}, ||x_5|| = a\sqrt{1^2 + (\frac{1}{2})^2 + (-2)^2} = \frac{\sqrt{21}}{2}$$

$$\implies v_3 = \frac{x_5}{\|x_5\|} = \begin{pmatrix} \frac{2}{\sqrt{21}} \\ \frac{1}{\sqrt{21}} \\ \frac{-4}{\sqrt{21}} \end{pmatrix}$$

Construct matrix $V$:

$$V = \begin{pmatrix} | & | & | \\ v_1 & v_2 & v_3 \\ | & | & | \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{14}} & \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{21}} \\ \frac{-2}{\sqrt{14}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{21}} \\ \frac{1}{\sqrt{14}} & \frac{1}{\sqrt{6}} & \frac{-4}{\sqrt{21}} \end{pmatrix}$$

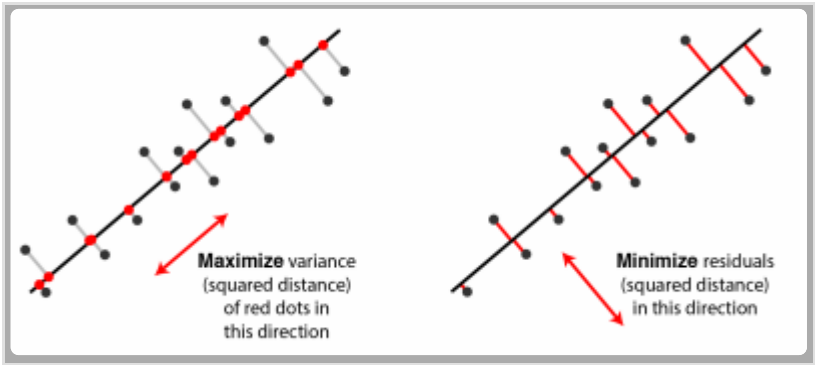
Transpose matrix $V$:

$$V^\top = \begin{pmatrix} - & v_1^\top & - \\ - & v_2^\top & - \\ - & v_3^\top & - \end{pmatrix} = \begin{pmatrix} \frac{3}{\sqrt{14}} & \frac{-2}{\sqrt{14}} & \frac{1}{\sqrt{14}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{21}} & \frac{1}{\sqrt{21}} & \frac{-4}{\sqrt{21}} \end{pmatrix}$$

### Step 5. Complete SVD

$$A = \begin{bmatrix} 2 & 0 & 1 \\ -1 & 2 & 0 \end{bmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \sqrt{7} & 0 & 0 \\ 0 & \sqrt{3} & 0 \end{pmatrix} \begin{pmatrix} \frac{3}{\sqrt{14}} & \frac{-2}{\sqrt{14}} & \frac{1}{\sqrt{14}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{21}} & \frac{1}{\sqrt{21}} & \frac{-4}{\sqrt{21}} \end{pmatrix}$$

# 4.2 Principle Component Analysis (PCA) 主成分分析




## 4.2.0 Why PCA?

- Project data from higher dimension to lower dimension, while preserving a low projection error.
- Maximizes *data variance* in low-dimensional representation.
- Simple & Non-parametric method of extracting relevant information from confusing data.
- Reduce a complicate dataset to a lower dimension.

### Problem Setup

#### Given

- An $n \times m$ training data set $X = \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{pmatrix}$
  - where $x^{(i)} \in \mathbb{R}^n$
  - that is, $X = \begin{pmatrix} x_1^{(1)} & x_1^{(2)} & \cdots & x_1^{(m)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(m)} \\ \vdots & \vdots & \ddots & \vdots \\ x_n^{(1)} & x_n^{(2)} & \cdots & x_n^{(m)} \end{pmatrix}$
  - Structural Analysis:
    - $n$ is the *dimension* of a data sample. Each row is a feature.

- $m$ is the *amount* of data sample. Each column is a data sample.

Do

- Reduces the dataset from $n$-dimensions to $k$-dimensions.
  - That is, to convert each feature from a $n$-d vector to a $k$-d vector;
  - Namely, to convert $X$ from an $n \times m$ matrix into a $k \times m$ matrix.

## 4.2.1 Data Pre-processing: Mean Normalization

Given

- The $n \times m$ training data set $X = \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{pmatrix}$

  Do

1. Calculate feature mean for all the vectors:
   - $\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \cdots \\ \mu_m \end{pmatrix}$
   - where $\mu_j = \sum_{i=1}^{n} x_j^{(i)}$.
   - A mean of a feature with respect to all the data samples.
2. Feature scaling:
   - For each row of $X$, that is a set of a specific feature of each data sample,
     - Reduce each value on this row by the row mean.
     - $\begin{pmatrix} x_j^{(1)} - \mu_j & x_j^{(2)} - \mu_j & \cdots & x_j^{(m)} - \mu_j \end{pmatrix}$

### What it does:

What we eventually get is:

- A scaled version of a dataset.
- Since different features may have their own range of values, which could vary, we need to normalize all features into a unified range of values.
  - E.g.: House Size is around 200 squared meters, while the price could be around 30,000.

## 4.2.2 Reduce Data Dimension

Given

- The normalized version of dataset $X$.

  Do

1. Compute the covariance matrix by:

$$\Sigma_{n\times n} = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}(x^{(i)})^\top = \frac{1}{m}XX^\top$$

2. Compute eigenvectors using *Singular Value Decomposition* on the covariate matrix $\Sigma$.

$$U_{n\times n}S_{n\times m}V_{m\times m} = svd(\Sigma)$$

3. Take the first $k$ columns from $U$.

$$U = \begin{pmatrix} | & | & & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} & \cdots & u^{(n)} \\ | & | & & | & & | \end{pmatrix} \in \mathbb{R}^{n\times n}$$

$$\implies U_{reduce} = \begin{pmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{pmatrix} \in \mathbb{R}^{n\times k}$$

4. We want to reduce $x^{(i)} \in \mathbb{R}^n \to z^{(i)} \in \mathbb{R}^k$ by:

$$z^{(i)} = U_{reduce}^\top x^{(i)}$$

   Namely,

$$\begin{pmatrix} - & (u^{(1)})^\top & - \\ - & (u^{(2)})^\top & - \\ & \vdots & \\ - & (u^{(k)})^\top & - \end{pmatrix}_{k\times n} \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_k^{(1)} \\ \vdots \\ x_n^{(1)} \end{pmatrix}_{n\times 1} = \begin{pmatrix} z_1^{(i)} \\ z_2^{(i)} \\ \vdots \\ z_k^{(i)} \end{pmatrix}_{k\times 1}$$

## 4.2.3 Choosing $k$

### Reconstruct Original Data

After PCA, we obtain $z^{(i)} = U_{reduce}^\top x^{(i)}$. We can reconstruct the original data from $z^{(i)}$ by:

$$\widetilde{x}^{(i)} = U_{reduce} z^{(i)}$$

The reconstruction comes with information loss. We will choose $k$ based on the information loss.

### Choosing $k$ - Slow

Average Squared Projection Error:

$$\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}-\tilde{x}^{(i)}||^2$$
$$=\frac{1}{m}\sum_{i=1}^{m}(x^{(i)}-\tilde{x}^{(i)})^\top(x^{(i)}-\tilde{x}^{(i)})$$

Total variation of data:

$$\frac{1}{m}\sum_{i=1}^{m}||x^{(i)}||^2$$
$$=\frac{1}{m}\sum_{i=1}^{m}{x^{(i)}}^\top x^{(i)}$$

Choose the target dimension number $k$ to be the smallest value so that:

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}-\tilde{x}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2}\leq 0.01$$

i.e., $99\%$ of the variance is retained.

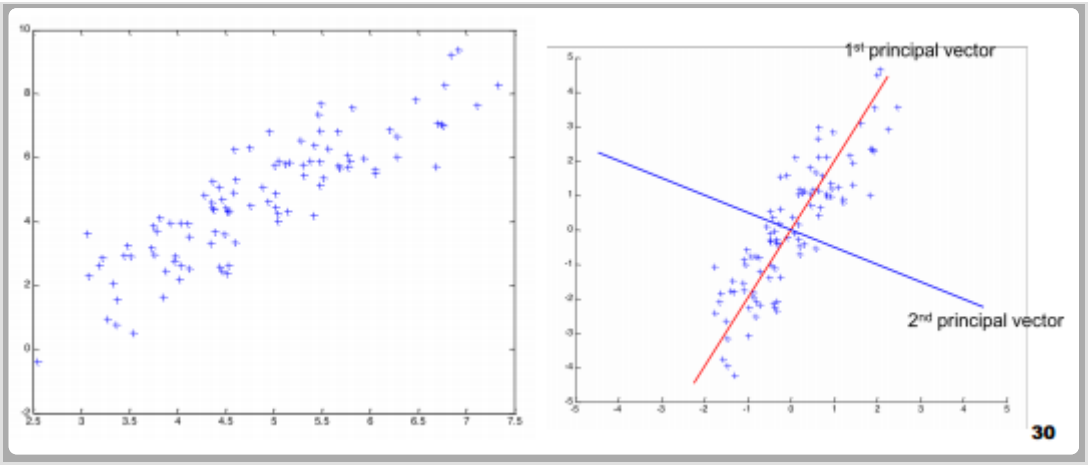
### Choosing $k$ - Fast

After performing SVD on $\Sigma=\frac{1}{m}XX^\top$, we have obtained $U$, $S$, and $V$.
Focusing on $S$, we pick the smallest $k$ for:

$$\frac{\sum_{i=1}^{k}s_{ii}}{\sum_{i=1}^{k}s_{ii}}\geq 0.99$$

i.e., $99\%$ of the variance is retained.

## 4.2.4 Results




In this example, the training data $X$ is of shape $2\times m$, thus the covariate matrix $C=\frac{1}{m}XX^\top$ is of shape $2\times 2$. Performing SVD on $C$:

$$C_{2\times 2}=U_{2\times 2}S_{2\times 2}V_{2\times 2}^\top$$

There are $2$ eigenvalues, with 2 principle vectors. The reduced $U$ would be of shape $2\times 1$.

### Red Line: 1st Principal Vector

Corresponds to the *largest* eigenvalue, indicating the most significant direction the data variates.

- That is, on this line, the projected data varies the most.

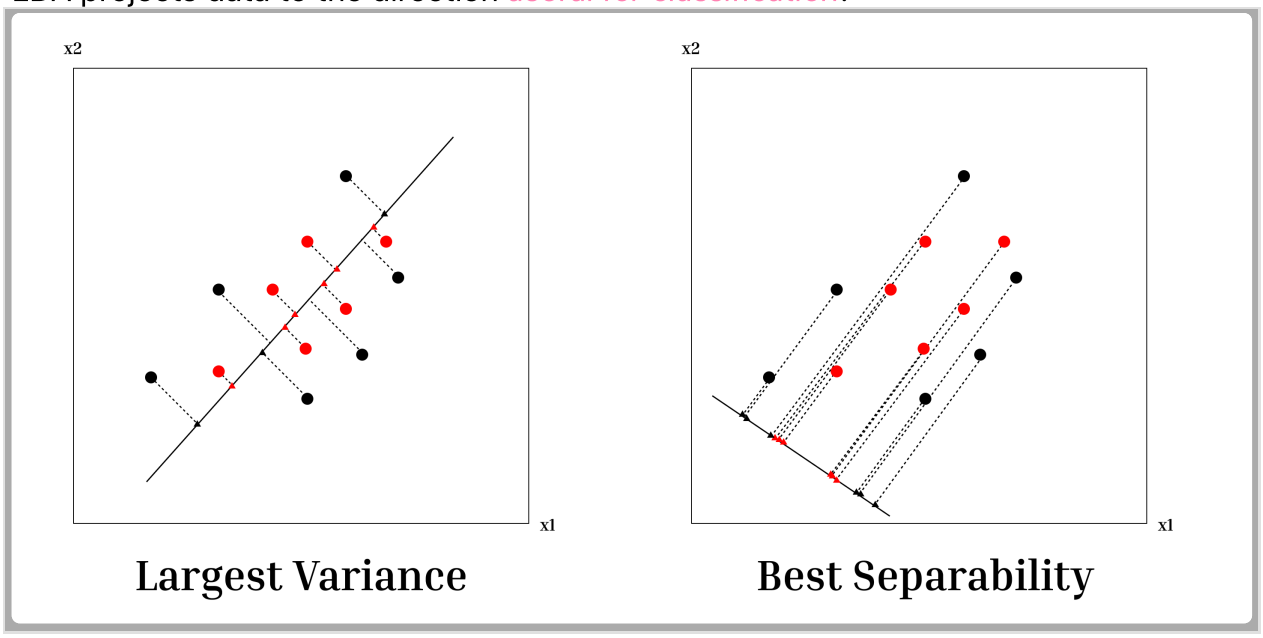
### Blue Line: 2nd Principal Vector

Corresponds to the *second largest* eigenvalue, being *perpendicular* to the first one.

# 4.3 Linear Discriminant Analysis (LDA) 线性判别分析

## 4.3.0 Problems of PCA

- The *directions* of maximum variance may be useless for classification.
  - I may indeed variates, but the classes could be completely mixed together.
- LDA solves this problem by:
  - not seeking the best variance,
  - but seeking the **best separability.**
- LDA projects data to the direction *useful for classification*.




## 4.3.1 LDA

**Given**

- A set of $d$-dimensional samples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$. From which,
  - $N_1$ samples belong to class $\omega_1$.
  - $N_2$ samples belong to class $\omega_2$.

  **Do**
- We seek a set of scalar $\mathbf{y} = \{y_1, y_2, \cdots, y_N\} \subset \mathbb{R}$ by projecting the $N$ samples in $x$ onto a line.

$$y_i = \mathbf{w}^\top \mathbf{x}_i \subset \mathbb{R}$$

- Namely,

$$y_i = \begin{pmatrix} w_1 & w_2 & \cdots & w_d \end{pmatrix} \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{id} \end{pmatrix}$$

  $y_i$ is the projected value of $\mathbf{x}_i$ in the new space.
- LDA selects the line that maximizes the *separability* of the scalars.
- In this new space, values of $y$ could be easily separated.

## 4.3.2 Measure of Separation

Supposed that we have a obtained such a line.

Sample Means of each class in $x$-space:

$$\mu_i = \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{x} \in \mathbb{R}^d$$

Sample Means of each class in $y$-space (projected mean):

$$\begin{aligned} \widetilde{\mu}_i &= \frac{1}{N_i} \sum_{y \in \omega_i} y \\ &= \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{w}^\top \mathbf{x} \\ &= \mathbf{w}^\top \mu_i \end{aligned}$$

### Distance of Means

The distance between the project mean is:

$$|\widetilde{\mu}_1 - \widetilde{\mu}_2| = |\mathbf{w}^\top (\mu_1 - \mu_2)| \in \mathbb{R}$$

Ignoring the standard deviation within classes.

### Scatter

Fisher's solution is to *maximize* the difference between the means of each class.

- The means of each class is normalized by a measure of the within-class scatter.
- The scatter is equivalent to the *variance* of each class.

The within-class scatter of a class $\omega_i$

$$\tilde{s}_i^2 = \sum_{y \in \omega_i} (y - \widetilde{\mu}_i)^2$$

The total within-class scatter of all the project samples would be

$$(\tilde{s}_1^2 + \tilde{s}_1^2)$$

★ The criterion function would be:

$$\mathcal{J}(\mathbf{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{s_1^2 + s_2^2}$$

We need to find the optimal $\mathbf{w}$ that maximizes the criterion function $\mathcal{J}(\mathbf{w})$.

## 4.3.3 Represent $\mathcal{J}(\mathbf{w})$ with $\mathbf{w}$

We want to find the optimal $\mathbf{w}$ such that the criterion function $\mathcal{J}(\mathbf{w})$ is maximized.

- Given that $\mathcal{J}(\mathbf{w}) = \frac{|\tilde{\mu}_1 - \tilde{\mu}_2|^2}{s_1^2 + s_2^2}$,
- we need to use $\mathbf{w}$ to represent the scatters.

### Within-Class Scatter

The scatter/variance in $x$-space:

$$S_i = \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^\top \in \mathbb{R}^d \times \mathbb{R}^d$$

The within-class scatter matrix:

$$S_W = S_1 + S_2$$

To express the scatter in $y$-space with $\mathbf{w}$:

$$\begin{aligned}
\tilde{s}_i^2 &= \frac{1}{N_i} \sum_{y \in \omega_i} (y - \tilde{\mu}_i)^2 \\
&= \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} (\mathbf{w}^\top \mathbf{x} - \mathbf{w}^\top \mu_i)^2 \\
&= \frac{1}{N_i} \sum_{\mathbf{x} \in \omega_i} \mathbf{w}^\top (\mathbf{x} - \mu_i)(\mathbf{x} - \mu_i)^\top \mathbf{w} \\
&= \mathbf{w}^\top S_i \mathbf{w}
\end{aligned}$$

★ That is,

$$\tilde{s}_1^2 + \tilde{s}_1^2 = \mathbf{w}^\top S_W \mathbf{w}$$

### Between-Class Scatter

The between-class scatter:

$$S_B = |\mu_1 - \mu_2|^2 = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top$$

The difference between the projected means:

$$
\begin{aligned}
(\tilde{\mu}_1 - \tilde{\mu}_2) &= (\mathbf{w}^\top \mu_1 - \mathbf{w}^\top \mu_2)^2 \\
&= \mathbf{w}^\top (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top \mathbf{w} \\
&= \mathbf{w}^\top S_B \mathbf{w}
\end{aligned}
$$

### The optimal $\mathbf{w}$

The optimal $\mathbf{w}$ will be:

$$
\begin{aligned}
\mathbf{w}^* &= \text{argmax}_{\mathbf{w}} \mathcal{J}(\mathbf{w}) \\
&= \text{argmax}_{\mathbf{w}} \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}
\end{aligned}
$$

## 4.3.4 Find the optimal $\mathbf{w}$

To find the optimal $\mathbf{w}$, we find that:

$$
\frac{d}{d\mathbf{w}} \mathcal{J}(\mathbf{w}) = 0
$$

$$
\implies \frac{d}{d\mathbf{w}} \left( \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}} \right) = 0
$$

$$
\implies \frac{1}{(\mathbf{w}^\top S_W \mathbf{w})^2} \cdot \left( \mathbf{w}^\top S_W \mathbf{w} \frac{d}{d\mathbf{w}} (\mathbf{w}^\top S_B \mathbf{w}) - \mathbf{w}^\top S_B \mathbf{w} \frac{d}{d\mathbf{w}} (\mathbf{w}^\top S_W \mathbf{w}) \right) = 0
$$

- 上导下不导-上不导下导

$$
\implies \frac{1}{(\mathbf{w}^\top S_W \mathbf{w})^2} \left( \mathbf{w}^\top S_W \mathbf{w} (2 S_B \mathbf{w}) - \mathbf{w}^\top S_B \mathbf{w} (2 S_W \mathbf{w}) \right) = 0
$$

$$
\implies \mathbf{w}^\top S_W \mathbf{w} (2 S_B \mathbf{w}) - \mathbf{w}^\top S_B \mathbf{w} (2 S_W \mathbf{w}) = 0
$$

$$
\implies S_B \mathbf{w} - \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}} (S_W \mathbf{w}) = 0
$$

$$
\implies S_B \mathbf{w} - \mathcal{J}_{\max} S_W \mathbf{w} = 0
$$

Set constant $\lambda = \mathcal{J}_{\max} = \frac{\mathbf{w}^\top S_B \mathbf{w}}{\mathbf{w}^\top S_W \mathbf{w}}$

$$
\implies S_B \mathbf{w} = \lambda S_W \mathbf{w}
$$

$$
\implies S_W^{-1} S_B \mathbf{w} = \lambda \mathbf{w}
$$

Know that $S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top$ is the between-class scatter matrix.

- Therefore, $S_B \mathbf{w} = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top \mathbf{w} = \alpha(\mu_1 - \mu_2)$
- where $\alpha = (\mu_1 - \mu_2)^\top \mathbf{w} \in \mathbb{R}$, that is $\alpha$ is a scalar.
- i.e., $S_B \mathbf{w}$ points to the same direction as $\mu_1 - \mu_2$

$$
\implies S_W^{-1} (\mu_1 - \mu_2) = \lambda \mathbf{w}
$$

★ $\implies \mathbf{w} = S_W^{-1}(\mu_1 - \mu_2)$

# Example:

Compute LDA projection of the following 2D dataset.

- $X_1 = \{(4,1), (2,4), (2,3), (3,6), (4,4)\}$
- $X_2 = \{(9,10), (6,8), (9,5), (8,7), (10,8)\}$

LDA Solution:

## Step 1: Data Arrangements

Arrange data into 2 separate matrices

$$X_1 = \begin{pmatrix} 4 & 2 & 2 & 3 & 4 \\ 1 & 4 & 3 & 6 & 4 \end{pmatrix}$$
$$X_2 = \begin{pmatrix} 9 & 6 & 9 & 8 & 10 \\ 10 & 8 & 5 & 7 & 8 \end{pmatrix}$$

## Step 2: Class Statistics

Sample means:

$$\mu_1 = \begin{pmatrix} \frac{4+2+2+3+4}{5} \\ \frac{1+4+3+6+4}{5} \end{pmatrix} = \begin{pmatrix} 3.0 \\ 3.6 \end{pmatrix}$$
$$\mu_2 = \begin{pmatrix} \frac{9+6+9+8+10}{5} \\ \frac{10+8+5+7+8}{5} \end{pmatrix} = \begin{pmatrix} 8.4 \\ 7.6 \end{pmatrix}$$

Sample Variants:

$$
\begin{aligned}
S_1 =& \frac{1}{5}\Bigg(\begin{bmatrix}1\\-2.6\end{bmatrix}\begin{bmatrix}1 & -2.6\end{bmatrix}+\\
&\begin{bmatrix}-1\\0.4\end{bmatrix}\begin{bmatrix}-1 & 0.4\end{bmatrix}+\\
&\begin{bmatrix}-1\\-0.6\end{bmatrix}\begin{bmatrix}-1 & -0.6\end{bmatrix}+\\
&\begin{bmatrix}0\\2.4\end{bmatrix}\begin{bmatrix}0 & 2.4\end{bmatrix}+\\
&\begin{bmatrix}1\\0.4\end{bmatrix}\begin{bmatrix}1 & 0.4\end{bmatrix}\Bigg)\\
=& \frac{1}{5}\begin{pmatrix}1 & -1 & -1 & 0 & 1\\ -2.6 & 0.4 & -0.6 & 2.4 & 0.4\end{pmatrix}\begin{pmatrix}1 & -2.6\\ -1 & 0.4\\ -1 & -0.6\\ 0 & 2.4\\ 1 & 0.4\end{pmatrix}\\
=& \frac{1}{5}\begin{pmatrix}4 & -2\\ -2 & 13.2\end{pmatrix}\\
=& \begin{pmatrix}0.8 & -0.4\\ -0.4 & 2.64\end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
S_2 &= \frac{1}{5}\begin{pmatrix}0.6 & -2.4 & 0.6 & -0.4 & 1.6\\ 2.4 & 0.4 & -2.6 & -0.6 & 0.4\end{pmatrix}\begin{pmatrix}0.6 & 2.4\\ -2.4 & 0.4\\ 0.6 & -2.6\\ -0.4 & -0.6\\ 1.6 & 0.4\end{pmatrix}\\
&= \frac{1}{5}\begin{pmatrix}9.2 & -0.2\\ -0.2 & 13.2\end{pmatrix}\\
&= \begin{pmatrix}1.84 & -0.04\\ -0.04 & 2.64\end{pmatrix}
\end{aligned}
$$

## Step 3: Between & Within Class Scatters

Within-class scatters:

$$
\begin{aligned}
S_W &= S_1 + S_2\\
&= \begin{pmatrix}0.8 & -0.4\\ -0.4 & 2.64\end{pmatrix} + \begin{pmatrix}1.84 & -0.04\\ -0.04 & 2.64\end{pmatrix}\\
&= \begin{pmatrix}2.64 & -0.44\\ -0.44 & 5.28\end{pmatrix}
\end{aligned}
$$

Between-class Scatter:

$$
\begin{aligned}
S_B &= (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top\\
&= \begin{pmatrix}-5.4\\ -4\end{pmatrix}\begin{pmatrix}-5.4 & -4\end{pmatrix}\\
&= \begin{pmatrix}29.16 & 21.6\\ 21.6 & 16\end{pmatrix}
\end{aligned}
$$

## Step 4: Calculate LDA Projection

Inverse of the between-class scatter matrix:

$$\begin{aligned} S_W^{-1} &= \begin{pmatrix} 2.64 & -0.44 \\ -0.44 & 5.28 \end{pmatrix}^{-1} \\ &= \frac{1}{5.28 \times 2.64 - 0.44^2} \begin{pmatrix} 5.28 & 0.44 \\ 0.44 & 2.64 \end{pmatrix} \\ &= \begin{pmatrix} 0.3841 & 0.0320 \\ 0.0320 & 0.1921 \end{pmatrix} \end{aligned}$$

The LDA projection $\mathbf{w}$

$$\begin{aligned} \mathbf{w} &= S_W^{-1}(\mu_1 - \mu_2) \\ &= \begin{pmatrix} 0.3841 & 0.0320 \\ 0.0320 & 0.1921 \end{pmatrix} \begin{pmatrix} -5.4 \\ -4 \end{pmatrix} \\ &= \begin{pmatrix} -2.2021 \\ -0.9412 \end{pmatrix} \end{aligned}$$

Therefore, the LDA projection line would be:

$$y = \begin{pmatrix} -2.2021 & -0.9412 \end{pmatrix} \mathbf{x}$$

# 05_Support_Vector_Machine

# 5.0 A Quick View

## What does it do?

- Find an optimized separating plane to
  - Separate samples of 2 classes
  - Maximize the margins

## Summary

| | $G$ | Constraint | Solution |
|---|---|---|---|
| Reg | $G=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda_i\lambda_j)\cdot(y_iy_j)\cdot\mathbf{x}_i^\top\mathbf{x}_j$ | $\begin{cases}\lambda_i\geq 0\\ \sum_{i=1}^{N}\lambda_iy_i=0\end{cases}$ | $f(\mathbf{x})=(\sum_{i=1}^{N}\lambda_iy$ |
| SfMg | $G=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda_i\lambda_j)\cdot(y_iy_j)\cdot\mathbf{x}_i^\top\mathbf{x}_j$ | $\begin{cases}0\leq\lambda_i\leq C\\ \sum_{i=1}^{N}\lambda_iy_i=0\end{cases}$ | $f(\mathbf{x})=(\sum_{i=1}^{N}\lambda_iy$ |
| NLin | $G=\sum_{i=1}^{N}\lambda_i-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\lambda_i\lambda_j)\cdot(y_iy_j)\cdot K(\mathbf{x}_i,\mathbf{x}_j)$ | $\begin{cases}\lambda_i\geq 0\\ \sum_{i=1}^{N}\lambda_iy_i=0\end{cases}$ | $f(\mathbf{x})=\sum_{i=1}^{N}\lambda_iy_i$ |

# 5.1 Margin

## 5.1.1 Motive

### Given

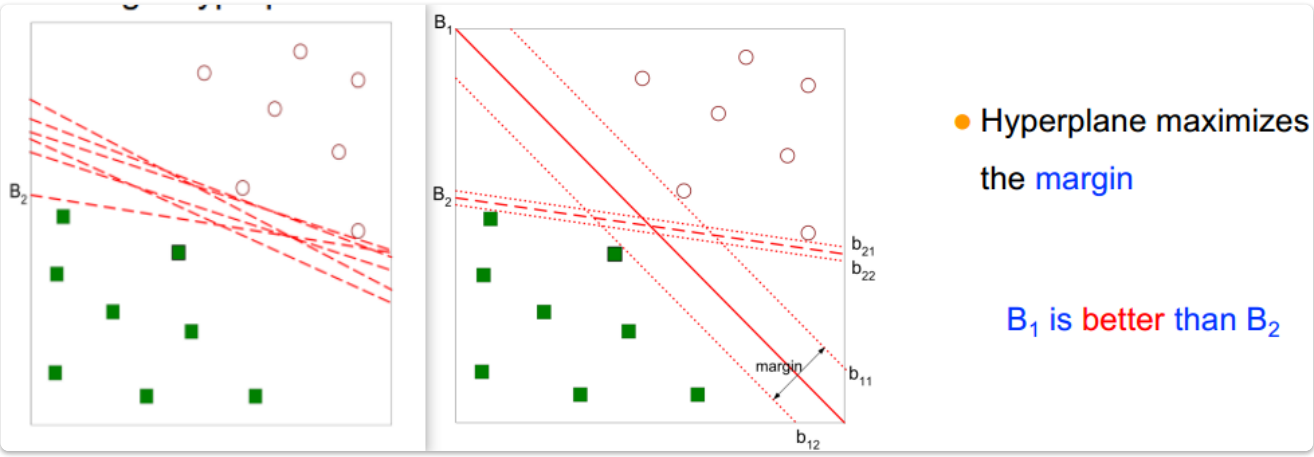
- A set of multi-dimensional linearly separable classes
  - $\mathcal{X}=\{\mathbf{x}_1,\mathbf{x}_2,\cdots,\mathbf{x}_N\}$
- Two classes to categorize samples in $\mathcal{X}$:
  - $\omega=\{\omega_{(1)},\omega_{(2)}\}$
- A mapping relation of $\mathcal{D}\in\mathcal{X}\times\omega$
  - $\mathcal{D}=\{\langle\mathbf{x}_1,\omega_1\rangle\ \langle\mathbf{x}_2,\omega_2\rangle\ \cdots,\langle\mathbf{x}_N,\omega_N\rangle\}$

### Do

- Find a hyperplane $\mathbf{w}^\top\mathbf{x}+b=0$ that separates the two classes.

- $\mathbf{w}$ is the normal vector of this hyperplane.

From multiple possible solutions, we want the one that maximizes the margin.




## 5.1.2 Distance to Hyperplane

The distance from each sample $\mathbf{x}_i$ to the hyperplane is:

$$r = \frac{\mathbf{w}^\top \mathbf{x}_i + b}{||\mathbf{w}||}$$

*Proof.* Suppose that $\mathbf{x}_p$ is the projection of a data sample $\mathbf{x}_i$ on the hyperplane $\mathbf{w}^\top \mathbf{x} + b = 0$. Therefore,

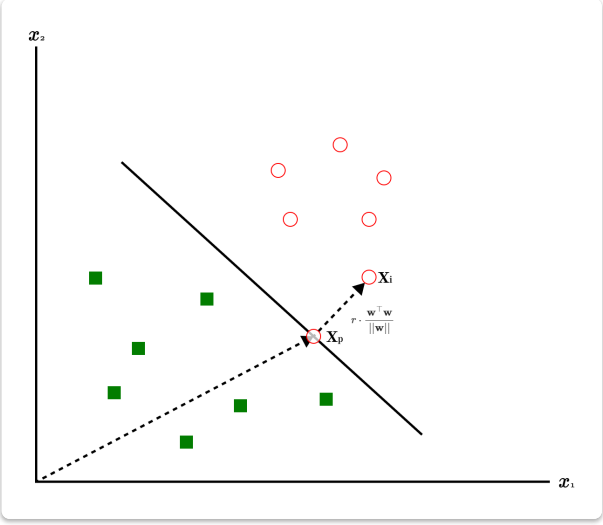
$$\mathbf{x}_i = \mathbf{x}_p + r \cdot \frac{\mathbf{w}}{||\mathbf{w}||}$$

namely,

$$\begin{aligned}
\mathbf{w}^\top \mathbf{x}_i + b &= \mathbf{w}^\top (\mathbf{x}_p + r \cdot \frac{\mathbf{w}}{||\mathbf{w}||}) + b \\
&= \mathbf{w}^\top \mathbf{x}_p + b + r \cdot \frac{\mathbf{w}^\top \mathbf{w}}{||\mathbf{w}||} \\
&= 0 + r \cdot \frac{\mathbf{w}^\top \mathbf{w}}{||\mathbf{w}||} \\
&= r \cdot \frac{\mathbf{w}^\top \mathbf{w}}{||\mathbf{w}||}
\end{aligned}$$

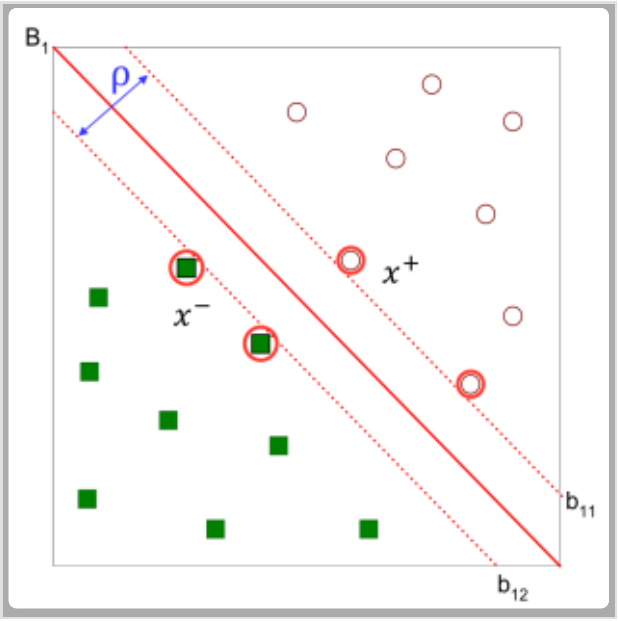
Thus,

$$\begin{aligned}
r &= (\mathbf{w}^\top \mathbf{x}_i + b) \cdot \frac{||\mathbf{w}||}{\mathbf{w}^\top \mathbf{w}} \\
&= (\mathbf{w}^\top \mathbf{x}_i + b) \cdot \frac{||\mathbf{w}||}{||\mathbf{w}||^2} \\
&= \frac{\mathbf{w}^\top \mathbf{x}_i + b}{||\mathbf{w}||}
\end{aligned}$$




## 5.1.3 Support Vectors and Margin

- Support Vectors are:
  - A subset of training samples
  - Samples closes to the hyperplane
- Margin $\rho$ is the distance between support vectors.
  - The hyperplane is to maximize the margin $\rho$.




In the above graph, there are 3 hyperplanes:

$$
\begin{aligned}
B_1 &: \mathbf{w}^\top \mathbf{x} + b = 0 \\
b_{11} &: \mathbf{w}^\top \mathbf{x} + b = +1 \\
b_{12} &: \mathbf{w}^\top \mathbf{x} + b = -1
\end{aligned}
$$

Where $\mathbf{x}^+$ and $\mathbf{x}^-$ lies on the hyperplanes $b_{11}$ and $b_{12}$. Then:

$$
\begin{aligned}
\mathbf{w}^\top (\mathbf{x}^+ - \mathbf{x}^-) &= \mathbf{w}^\top \mathbf{x}^+ - \mathbf{w}^\top \mathbf{x}^- \\
&= (\mathbf{w}^\top \mathbf{x}^+ + b) - (\mathbf{w}^\top \mathbf{x}^- + b) \\
&= (1) - (-1) \\
&= 2
\end{aligned}
$$

The margin would be:

$$\begin{aligned}\rho &= \frac{\mathbf{w}^\top(\mathbf{x}^+ - \mathbf{x}^-)}{\|\mathbf{w}\|} \\ &= \frac{2}{\|\mathbf{w}\|}\end{aligned}$$

# 5.2 Quadratic Optimization

## 5.2.1 Formulation

Let

- $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ be the data set
- $y = \{y_1, y_2, \cdots, y_N\} \subset \{-1, 1\}^N$ be the class labels of the corresponding data in $\mathcal{X}$.

Do

- Find the optimal $\mathbf{w}$ such that:
  - $\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized, and
  - $\begin{cases}\mathbf{w}^\top \mathbf{x}_i + b \geq 1 & \text{if } y_i = +1 \\ \mathbf{w}^\top \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1\end{cases}$ for $i = 1, 2, \cdots, N$

Maximizing the margin $\rho = \frac{2}{\|\mathbf{w}\|}$ is equivalent to minimizing:

$$\frac{1}{2}\|\mathbf{w}\|^2 = \frac{1}{2}\mathbf{w}^\top \mathbf{w}$$

★ The formulated quadratic optimization problem of SVM is:

- Minimize: $\frac{1}{2}\|\mathbf{w}\|^2$
- With constraint: $y_i(\mathbf{w}^\top + b) \geq 1,\ \forall\, 1, 2, \cdots, N$

## 5.2.2 Lagrangian of Quadratic Optimization

★ The Lagrangian of the quadratic optimization problem is:

$$L(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \sum_{i=1}^{N} \lambda_i (1 - y_i(\mathbf{w}^\top \mathbf{x}_i + b))$$

where $\lambda_1, \lambda_2, \cdots, \lambda_N \geq 0$ is the Lagrangian multiplier of all the data points in $\mathcal{X}$ respectively.

★ At the end, only the support vector's Lagrangian multiplier would be non-zero.

- That is $\lambda_i \neq 0$ if and only if $\mathbf{x}_i$ is a support vector.
- Non-support vectors won't contribute to the hyper plane.

Suppose that we have already found a series of such Lagrangian multipliers. To optimize $L$, we compute the partial derivatives of $L$ with respect to $\mathbf{w}$ and $b$.

### Optimize $L$ w.r.t. $\mathbf{w}$.

$$\begin{aligned}\frac{dL}{d\mathbf{w}} &= \mathbf{w} + \frac{dL}{d\mathbf{w}} \sum_{i=1}^{N} \lambda_i - (\lambda_i y_i)\mathbf{w}^\top \mathbf{x}_i - (\lambda_i y_i b) \\ &= \mathbf{w} + \sum_{i=1}^{N} -\lambda_i y_i \mathbf{x}_i \\ &= \mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i\end{aligned}$$

Let $\frac{dL}{d\mathbf{w}} = 0$.

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

### Optimize $L$ w.r.t. $b$

$$\begin{aligned}\frac{dL}{db} &= 0 + \frac{dL}{db} \sum_{i=1}^{N} \lambda_i - (\lambda_i y_i)\mathbf{w}^\top \mathbf{x}_i - (\lambda_i y_i b) \\ &= \sum_{i=1}^{N} -\lambda_i y_i \\ &= -\sum_{i=1}^{N} \lambda_i y_i\end{aligned}$$

Let $\frac{dL}{db} = 0$.

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

- ★ Therefore, the optimized weight and bias of this quadratic is:

$$\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$$

- ★ with a constraint of:

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

  with respect to $\lambda_1, \lambda_2, \cdots, \lambda_N \geq 0$.

## 5.2.3 Dual Problem

### Get $\lambda$ with optimized $L$

Substitute $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{N} \lambda_i y_i = 0$ into $L(\mathbf{w}, b)$ would result in:

$$
\begin{aligned}
L(\mathbf{w}, b) =& \frac{1}{2}\Big(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i\Big)^\top \Big(\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i\Big) \\
&+ \sum_{i=1}^{N} \lambda_i (1 - y_i \Big( \Big(\sum_{j=1}^{N} \lambda_j y_j \mathbf{x}_j\Big)^\top \mathbf{x}_i + b\Big)) \\
\\
=& \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j \\
&+ \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} (\lambda_i y_i) \cdot \Big( \sum_{j=1}^{N} (\lambda_j y_j) \cdot \mathbf{x}_j^\top \mathbf{x}_i + b\Big) \\
\\
=& \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j \\
&+ \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} \Big( \sum_{j=1}^{N} (\lambda_i y_i) \cdot (\lambda_j y_j) \cdot \mathbf{x}_j^\top \mathbf{x}_i + b(\lambda_i y_i)\Big) \\
\\
=& \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j \\
&+ \sum_{i=1}^{N} \lambda_i - \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i y_i) \cdot (\lambda_j y_j) \cdot \mathbf{x}_j^\top \mathbf{x}_i \\
&+ b \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i y_i) \\
\\
=& -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j \\
&+ \sum_{i=1}^{N} \lambda_i \\
&+ b \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i y_i) \\
\\
=& \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j
\end{aligned}
$$

The original criterion function is now with respect to only $\lambda$. That is:

$$
G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j
$$

Optimize $G(\lambda)$ by computing:

$$
\frac{dG}{d\lambda_i}, \ \forall i = 2, 3, \cdots, N
$$

Check the solutions if they satisfy the constraint of:

$$\begin{cases} \lambda_i > 0 \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases}$$

### 5.2.4 Solutions: Regular

Maximize:

$$G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j$$

with constraint:

$$\begin{cases} \lambda_i \geq 0 \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases}$$

The dual solution is:

$$f(\mathbf{x}) = (\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i)^\top \mathbf{x} + b$$

with:

$$\begin{cases} \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \\ b = y_k - (\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^\top) \mathbf{x}_k \end{cases}$$

## 5.3 Soft Margin Classification

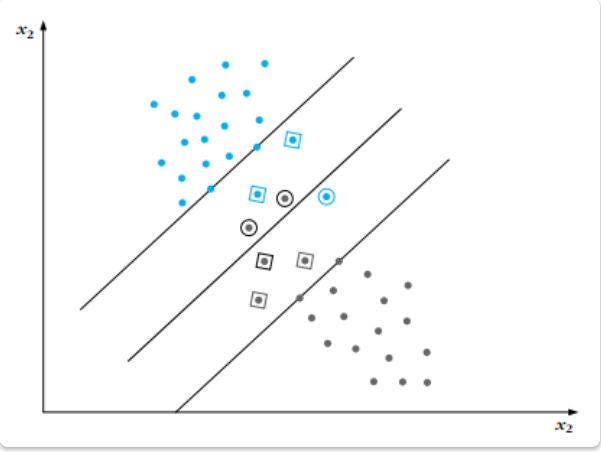
### 5.3.1 Problem Setup

There exists conditions that training samples can't be separated.

- In this case, *no* hyperplane could satisfy $y_i(\mathbf{w}^\top \mathbf{x} + b) > 1, \ \forall \mathbf{x}$.
- i.e., $\neg \exists \mathbf{w}, b, \forall \mathbf{x}, \ y_i(\mathbf{w}^\top \mathbf{x} + b) > 1$

Training samples belong to one of the three possible categories.

- Correctly Classified: Samples outside the margin.
  - $y_i(\mathbf{w}^\top + b) > 1$
- Margin Violation: Samples within the margin, but correctly classified.
  - $y_i(\mathbf{w}^\top + b) > 1$
- Misclassified samples:
  - $y_i(\mathbf{w}^\top + b) < 0$




## 5.3.2 Slack Variables & Parameter C 松弛因子与C参数

### Assignment of $\xi_i$

Assign slack variables $\xi_1, \xi_2, \cdots, \xi_N \geq 0$ to all the samples in $\mathcal{X}$.

- Correctly Classified: $\xi_i = 0$
- Margin Violation: $0 \leq \xi_i \leq 1$
- Misclassified Variables: $\xi_i > 1$

About slack variables $\xi_i$.

- $\xi_i$ allows misclassification of difficult or noisy samples.
  - The resulting is called a Soft Margin.
  - If $\xi_i$ is sufficiently large, every constraint will be forced to be satisfied.
- $\xi_i$ is based on the output of the discriminant function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$.
- $\xi_i$ approximates the number of mis-classified samples.

### Intuitive Optimization

The optimization problem becomes:

- Minimize: $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \xi_i$
- With constraint: $y_i(\mathbf{w}^\top + b) \geq 1 - \xi_i, \ \forall\, 1, 2, \cdots, N$

The parameter $C$ is a user-selected regularization parameter.

- A trade-off parameter between:
  - error and
  - margin
- Effects of parameter $C$:
  - Smaller $C$ = Large Margin & More error, allows constraints to be easily ignored.
  - Larger $C$ = Narrow Margin & Less error, constraints is hard to ignore.
  - $C = \infty$ = Hard Margin, which enforces all constraints.

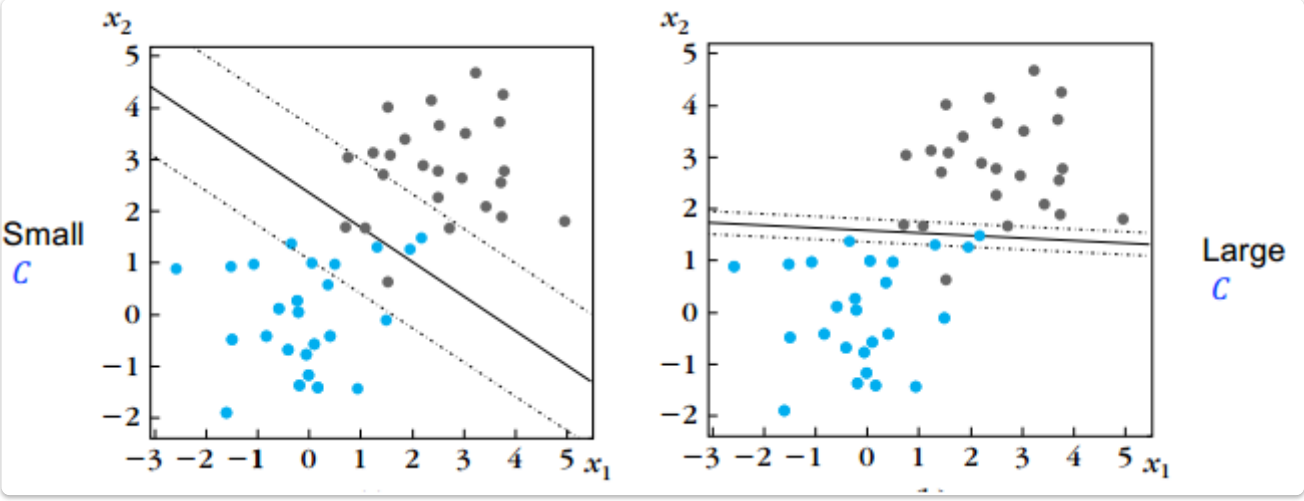
### Corresponding Dual Problem

Which transfer to the dual problem as

$$G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j$$

with the constraints of:

$$\begin{cases} 0 \leq \lambda_i \leq C \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases}$$




## 5.3.3 Solutions: Soft Margin

Maximize:

$$G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j$$

with constraint:

$$\begin{cases} 0 \leq \lambda_i \leq C \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases}$$

The dual solution is:

$$f(\mathbf{x}) = (\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i)^\top \mathbf{x} + b$$

with:

$$\begin{cases} \mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \\ b = y_k(1 - \xi_k) - (\sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^\top) \mathbf{x}_k \end{cases}$$

# 5.4 Non-Linear SVM

# 5.4.0 Basics

## Why Non-Linear SVM?

- Datasets may be too hard for linear separation.

## What does it do?

- Transform data into a higher dimensional space $\mathbf{H}$,
  - via a mapping function $\mathbf{\Phi}$
  - such that the data appears of the form $\mathbf{\Phi}(\mathbf{x}_i)\mathbf{\Phi}(\mathbf{x}_j)$.
- Linear separation in $\mathbf{H}$ is *equivalent* to non-linear separation in the original input space.

## Problems

- High dimensionality results in high computation burden.
- Hard to obtain a good estimation.

# 5.4.1 Kernel Function

> A kernel of two data is:
>
> - The inner product between the vectors
> - $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$

Suppose that each data point is mapped into high-dimensional space via transformation of:

$$\mathbf{\Phi} : \mathbf{x} \mapsto \phi(\mathbf{x})$$

Then, the kernel of two data becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

We could compute $K(\mathbf{x}_i, \mathbf{x}_j)$ without computing $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ explicitly.

# 5.4.2 Compute Kernel

We could compute kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ in the original space. Suppose that the original space is of 2 dimensions. Let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2$, then $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2$.
*Proof.*

$$
\begin{aligned}
K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^\top \mathbf{x}_j)^2 \\
&= (1 + \begin{bmatrix} x_{i1} & x_{i2} \end{bmatrix} \begin{bmatrix} x_{j1} \\ x_{j2} \end{bmatrix})^2 \\
&= \Big(1 + (x_{i1}x_{j1} + x_{i2}x_{j2})\Big)^2 \\
&= 1 + (x_{i1}x_{j1} + x_{i2}x_{j2})^2 - 2((x_{i1}x_{j1} + x_{i2}x_{j2})) \\
&= x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 1 \\
&= \begin{bmatrix} x_{i1}^2 & x_{i2}^2 & \sqrt{2}x_{i1}x_{i2} & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} & 1 \end{bmatrix} \begin{bmatrix} x_{j1}^2 \\ x_{j2}^2 \\ \sqrt{2}x_{j1}x_{j2} \\ \sqrt{2}x_{j1} \\ \sqrt{2}x_{j2} \\ 1 \end{bmatrix} \\
&= \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)
\end{aligned}
$$

where

$$
\phi(\mathbf{x}) = \phi(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ 1 \end{bmatrix}
$$

## 5.4.3 Kernel Examples

| | Kernel | Mapping |
|---|---|---|
| Linear | $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$ | $\mathbf{\Phi} : \mathbf{x} \mapsto \phi(\mathbf{x}) = \mathbf{x}$ |
| Polynomial | $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^p$ | $\mathbf{\Phi} : \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathbb{R}^{\frac{(d+p)!}{p!d!}}$ |
| Sigmoid | $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^\top \mathbf{x}_j + \beta_1)$ | |
| Gaussian | $K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{\|x_i - x_j\|^2}{2\sigma}}$ | $\mathbf{\Phi} : \mathbf{x} \mapsto \phi(\mathbf{x}) \in \mathbb{R}^\infty$ |

## 5.4.4 Solutions: Non-Linear

Maximize:

$$
G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot K(\mathbf{x}_i, \mathbf{x}_j)
$$

with constraint:

$$\begin{cases} \lambda_i \geq 0 \\ \sum_{i=1}^{N} \lambda_i y_i = 0 \end{cases}$$

The dual solution:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

# 5.5 Examples

## Non-Linear

### Given

- Suppose we have five 1-D datapoints:
  - $x_1 = 1, x_2 = 2, x_3 = 3, x_4 = 5, x_5 = 6$
- Their corresponding classes are:
  - $y_1 = 1, y_2 = 1, y_3 = -1, y_4 = -1, y_5 = 1$
- Use the degree-2 polynomial kernel:
  - $K(x_i, x_j) = (x_i x_j + 1)^2$

### Do

*Step 1.* Find $\lambda_i$.

Maximize:

$$G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot (1 + x_i x_j)^2$$

with constraint:

$$\sum_{i=1}^{N} \lambda_i y_i = 0$$

Using a quadratic problem solver, we obtain:

$$\lambda_1 = 0,\ \lambda_2 = 2.5,\ \lambda_3 = 0,\ \lambda_4 = 7.333,\ \lambda_5 = 4.833$$

*Step 2.* Calculate.

The support vectors are:

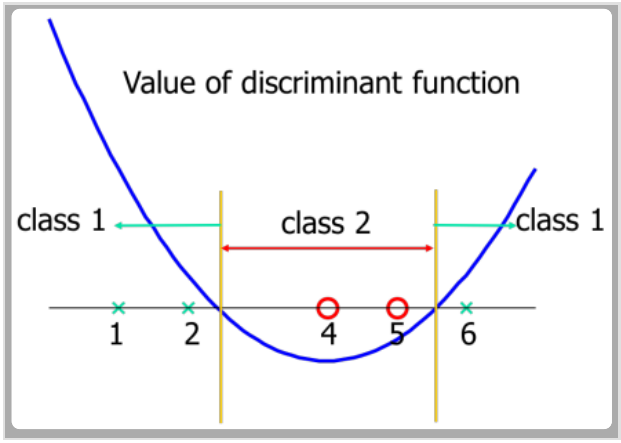
$$x_2 = 2,\ x_4 = 5,\ x_5 = 6$$

The discriminant function is:

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{N} \lambda_i y_i K(x_i, x) + b \\
&= \sum_{i=1}^{N} \lambda_i y_i (1 + x_i x)^2 + b \\
&= 2.5 \cdot 1 \cdot (1 + 2x)^2 + 7.333 \cdot (-1) \cdot (1 + 5x)^2 + 4.833 \cdot 1 \cdot (1 + 6x)^2 + b \\
&= 0.6667x^2 - 5.333x + b
\end{aligned}
$$

Given that $f(6) = 1$, $b = 9$.
We obtained the discriminant function of:

$$f(x) = 0.6667x^2 - 5.333x + 9$$




## Linear

### Given

- Suppose we have three 2-D data points.
  - $\mathbf{x}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$, $\mathbf{x}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$, $\mathbf{x}_3 = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$
- Their corresponding labels are:
  - $y_1 = 1$, $y_2 = -1$, $y_3 = -1$
- Use the trivial kernel:
  - $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$

### Do

*Step 1.* Find $\lambda$.

$$G(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\lambda_i \lambda_j) \cdot (y_i y_j) \cdot \mathbf{x}_i^\top \mathbf{x}_j$$

*1-1.* Constraint:

$$
\begin{aligned}
\sum_{i=1}^{N} \lambda_i y_i = 0 &\implies \lambda_1 - \lambda_2 - \lambda_3 = 0 \\
&\implies \lambda_1 = \lambda_2 + \lambda_3
\end{aligned}
$$

*1-2.* Express $G(\lambda)$ with $\lambda_1$, $\lambda_2$ and $\lambda_3$.

$$\frac{1}{2}\sum_{i=1}^{N}(\lambda_i\lambda_j)\cdot(y_iy_j)\cdot\mathbf{x}_i^\top\mathbf{x}_j=\sum_{i=1}^{N}\lambda_i-G(\lambda)$$

$$\begin{aligned}\Longrightarrow\frac{1}{2}\sum_{i=1}^{N}\Big(&\\&(\lambda_i\lambda_1)\cdot(y_iy_1)\cdot\mathbf{x}_i^\top\mathbf{x}_1+\\&(\lambda_i\lambda_2)\cdot(y_iy_2)\cdot\mathbf{x}_i^\top\mathbf{x}_2+\\&(\lambda_i\lambda_3)\cdot(y_iy_3)\cdot\mathbf{x}_i^\top\mathbf{x}_3\\\Big)&=\sum_{i=1}^{N}\lambda_i-G(\lambda)\end{aligned}$$

$$\begin{aligned}\Longrightarrow\frac{1}{2}\Big(&\\&(\lambda_1\lambda_1)\cdot(y_1y_1)\cdot\mathbf{x}_1^\top\mathbf{x}_1+(\lambda_1\lambda_2)\cdot(y_1y_2)\cdot\mathbf{x}_1^\top\mathbf{x}_2+(\lambda_1\lambda_3)\cdot(y_1y_3)\cdot\mathbf{x}_1^\top\mathbf{x}_3+\\&(\lambda_2\lambda_1)\cdot(y_2y_1)\cdot\mathbf{x}_2^\top\mathbf{x}_1+(\lambda_2\lambda_2)\cdot(y_2y_2)\cdot\mathbf{x}_2^\top\mathbf{x}_2+(\lambda_2\lambda_3)\cdot(y_2y_3)\cdot\mathbf{x}_2^\top\mathbf{x}_3+\\&(\lambda_3\lambda_1)\cdot(y_3y_1)\cdot\mathbf{x}_3^\top\mathbf{x}_1+(\lambda_3\lambda_2)\cdot(y_3y_2)\cdot\mathbf{x}_3^\top\mathbf{x}_2+(\lambda_3\lambda_3)\cdot(y_3y_3)\cdot\mathbf{x}_3^\top\mathbf{x}_3\\\Big)&=\sum_{i=1}^{N}\lambda_i-G(\lambda)\end{aligned}$$

$$\begin{aligned}\Longrightarrow\frac{1}{2}\Big(&\\&\lambda_1^2\cdot(1\cdot1)\cdot\begin{bmatrix}2&1\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix}+\lambda_1\lambda_2\cdot(1\cdot-1)\begin{bmatrix}2&1\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}+\lambda_1\lambda_3\cdot(1\cdot-1)\cdot\begin{bmatrix}2&1\end{bmatrix}\begin{bmatrix}3\\3\end{bmatrix}+\\&\lambda_2\lambda_1\cdot(-1\cdot1)\cdot\begin{bmatrix}1&2\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix}+\lambda_2^2\cdot(-1\cdot-1)\begin{bmatrix}1&2\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}+\lambda_2\lambda_3\cdot(-1\cdot-1)\cdot\begin{bmatrix}1&2\end{bmatrix}\begin{bmatrix}3\\3\end{bmatrix}+\\&\lambda_3\lambda_1\cdot(-1\cdot1)\cdot\begin{bmatrix}3&3\end{bmatrix}\begin{bmatrix}2\\1\end{bmatrix}+\lambda_3\lambda_2\cdot(-1\cdot-1)\begin{bmatrix}3&3\end{bmatrix}\begin{bmatrix}1\\2\end{bmatrix}+\lambda_3^2\cdot(-1\cdot-1)\cdot\begin{bmatrix}3&3\end{bmatrix}\begin{bmatrix}3\\3\end{bmatrix}\\\Big)&=\sum_{i=1}^{N}\lambda_i-G(\lambda)\end{aligned}$$

$$\begin{aligned}\Longrightarrow\frac{1}{2}\Big(&\\&5\lambda_1^2-4\lambda_1\lambda_2-9\lambda_1\lambda_3\\&-4\lambda_1\lambda_2+5\lambda_2^2+9\lambda_2\lambda_3\\&-9\lambda_1\lambda_3+9\lambda_3\lambda_3+18\lambda_3^2\\\Big)&=\lambda_1+\lambda_2+\lambda_3-G(\lambda)\end{aligned}$$

$$\begin{aligned}\Longrightarrow G(\lambda)=&-\frac{1}{2}(5\lambda_1^2+5\lambda_2^2+18\lambda_3^2-8\lambda_1\lambda_2-18\lambda_1\lambda_3+18\lambda_2\lambda_3)\\&+(\lambda_1+\lambda_2+\lambda_3)\end{aligned}$$

$$\begin{aligned}\Longrightarrow G(\lambda)=&-\frac{1}{2}(5(\lambda_2+\lambda_3)^2+5\lambda_2^2+18\lambda_3^2-8(\lambda_2+\lambda_3)\lambda_2-18(\lambda_2+\lambda_3)\lambda_3+18\lambda_2\lambda_3)\\&+((\lambda_2+\lambda_3)+\lambda_2+\lambda_3)\end{aligned}$$

$$\Longrightarrow G(\lambda)=-\lambda_2^2-\frac{5}{2}\lambda_3^2-\lambda_2\lambda_3+2\lambda_2+2\lambda_3$$

*1-3.* Minimize $G(\lambda)$

$$\frac{\partial G}{\partial \lambda_3} = 0$$

$$\implies \frac{\partial}{\partial \lambda_3}\left(-\lambda_2^2 + (2-\lambda_3)\lambda_2 + (2\lambda_3 - \frac{5}{2}\lambda_3^2)\right) = 0$$

$$\implies 2\lambda_2 + \lambda_3 - 2 = 0$$

$$\frac{\partial G}{\partial \lambda_3} = 0$$

$$\implies \frac{\partial}{\partial \lambda_3}\left(-\frac{5}{2}\lambda_3^2 + (2-\lambda_2)\lambda_3 + (2\lambda_2 - \lambda_2^2)\right) = 0$$

$$\implies \lambda_2 + 5\lambda_3 - 2 = 0$$

*1-4.* Summarize.

$$\begin{cases} \lambda_1 - \lambda_2 - \lambda_3 = 0 \\ 2\lambda_2 + \lambda_3 = 2 \\ \lambda_2 + 5\lambda_3 = 2 \end{cases}$$

$$\implies \begin{bmatrix} 1 & -1 & -1 \\ 0 & 2 & 1 \\ 0 & 1 & 5 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}$$

$$\implies \begin{cases} \lambda_1 = \frac{10}{9} \\ \lambda_2 = \frac{8}{9} \\ \lambda_3 = \frac{2}{9} \end{cases}$$

*Step 2.* Calculate.

The discriminant function is

$$\begin{aligned} f(\mathbf{x}) &= \sum_{i=1}^{N} \lambda_i y_i \mathbf{x}_i^\top \mathbf{x} + b \\ &= (\frac{10}{9}[2 \quad 1] - \frac{8}{9}[1 \quad 2] - \frac{2}{9}[3 \quad 3])\mathbf{x} + b \\ &= \left[\frac{2}{3} \quad -\frac{4}{3}\right]\mathbf{x} + b \end{aligned}$$

Get $b$:

$$f(\mathbf{x}_1) = 1$$

$$\implies f(\begin{bmatrix} 2 \\ 1 \end{bmatrix}) = 1$$

$$\implies \begin{bmatrix} \frac{2}{3} & -\frac{4}{3} \end{bmatrix} \begin{bmatrix} 2 \\ 1 \end{bmatrix} + b = 1$$

$$\implies \frac{4}{3} - \frac{4}{3} + b = 1$$

$$\implies b = 1$$

Therefore, the discriminant function would be:

$$f(\mathbf{x}) = \begin{bmatrix} \frac{2}{3} & -\frac{4}{3} \end{bmatrix} \mathbf{x} + 1$$

# 06_Perceptron_and_Neural_Networks

# 6.1 Perceptron Algorithm

## 6.1.1 Problem Setup

Given:

- A set of $l$-dimensional data samples:

$$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^l$$

- Two classes:

$$\omega = \{\omega_1, \omega_2\}$$

- A classification relation:

$$R : \mathcal{X} \mapsto \omega$$

  Do:
  Learn a decision hyper-plane:

$$g(\mathbf{x}) : \ \mathbf{w}^\top \mathbf{x} + w_0 = 0$$

  where $\mathbf{w}$ is the $l$-dimensional column weight vector and $w_0$ is the threshold.

Assume that the classes are linearly separable. Therefore:

$$\exists\, \mathbf{w}, w_0, \begin{cases} \mathbf{w}^\top \mathbf{x} + w_0 > 0 & \mathbf{x} \in \omega_1 \\ \mathbf{w}^\top \mathbf{x} + w_0 < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

We could omit the additional threshold by letting:

$$\mathbf{w}_{\text{new}} = \begin{bmatrix} \mathbf{w}_{\text{old}} \\ w_0 \end{bmatrix}$$

$$\mathbf{x}_{\text{new}} = \begin{bmatrix} \mathbf{x}_{\text{old}} \\ 1 \end{bmatrix}$$

Here:

- The original $w_0$ is inserted into the original $\mathbf{w}$, letting the new weight vector to grow by $1$ dimension.
- Correspondingly, all the data samples are expanded by $1$ dimension, with the new dimension being $1$.
  Therefore, we could conclude that:

$$\mathbf{w}_{\text{old}}^{\top}\mathbf{x}_{\text{old}} + w_0 \equiv \mathbf{w}_{\text{new}}^{\top}\mathbf{x}_{\text{new}}$$

★ In general, the original linear separability could be expressed by:

$$\exists\, \mathbf{w} \in \mathbb{R}^{l+1}, \quad \begin{cases} \mathbf{w}^{\top}\mathbf{x} > 0 & \mathbf{x} \in \omega_1 \\ \mathbf{w}^{\top}\mathbf{x} < 0 & \mathbf{x} \in \omega_2 \end{cases}$$

### Introduction to Perceptron Algorithm

Our goal is to compute such a solution. To reach this goal, we:

- Define a Cost Function.
- Choose an algorithm to *minimize* the cost function.
  - The minimum corresponds to a hyperplane solution.

## 6.1.2 Perceptron Cost Function

ℹ A "cost" is defined by the following:

$$\mathcal{J}(\mathbf{w}) = \sum_{\mathbf{x}\in\mathcal{Y}} \delta_{\mathbf{x}} \mathbf{w}^{\top}\mathbf{x}$$

where:

- $\mathcal{Y} \subset \mathcal{X}$ is the training vectors that's been *wrongly classified* by $\mathbf{w}$.
  - $\mathcal{Y} \in \emptyset$ means a solution is achieved.
- $\delta_{\mathbf{x}} = \begin{cases} -1 & \text{if } \mathbf{x} \in \omega_1 \\ +1 & \text{if } \mathbf{x} \in \omega_2 \end{cases}$

The gradient of the cost function:

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial}{\partial \mathbf{w}} \sum_{\mathbf{x}\in\mathcal{Y}} \delta_{\mathbf{x}} \mathbf{w}^{\top}\mathbf{x}$$

$$= \sum_{\mathbf{x}\in\mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x}$$

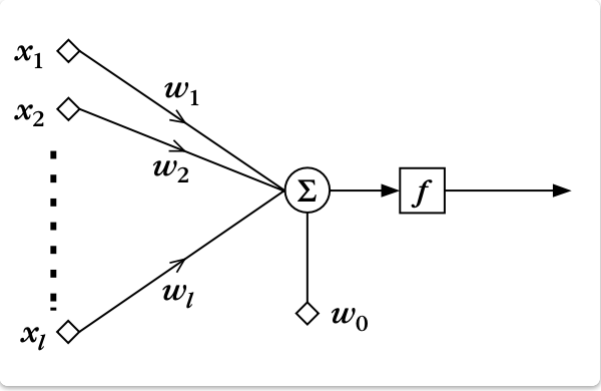
★ The gradient descent algorithm would be:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho_t \frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}}\bigg|_{\mathbf{w}=\mathbf{w}_t}$$

$$= \mathbf{w}_t - \rho_t \sum_{\mathbf{x}\in\mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x}$$

### What exactly is a perceptron?

A demonstration of a single perceptron:




A perceptron is:

- A linear combination of inputs and weights.

## 6.1.3 Example: A single update step of weight

Given:

- Current weight with bias:

$$\mathbf{w}_t = \begin{bmatrix} w_1 \\ w_2 \\ w_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix}$$

- Wrongly classified data samples:

$$\mathbf{x}_1 = \begin{bmatrix} -0.2 \\ 0.75 \end{bmatrix},\ \mathbf{x}_2 = \begin{bmatrix} 0.4 \\ 0.05 \end{bmatrix} \in \mathcal{Y}$$

  where $\mathbf{x}_1 \in \omega_2,\ \mathbf{x}_2 \in \omega_1$

- The learning rate of the current step:

$$\rho_t = \rho = 0.7$$

  Do:

  Update the weights of the separating plane:

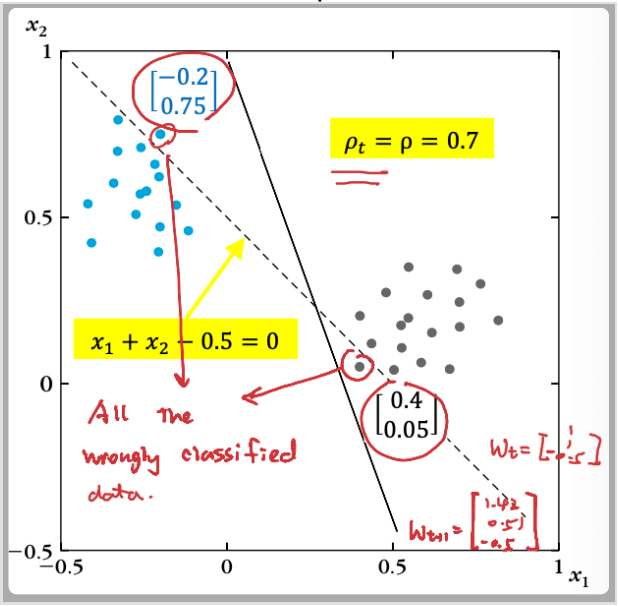
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho \sum_{\mathbf{x} \in \mathcal{Y}} \delta_{\mathbf{x}} \mathbf{x}$$

$$= \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7 \times \left( (+1) \begin{bmatrix} -0.2 \\ 0.75 \\ 1 \end{bmatrix} + (-1) \begin{bmatrix} 0.4 \\ 0.05 \\ 1 \end{bmatrix} \right)$$

$$= \begin{bmatrix} 1 \\ 1 \\ -0.5 \end{bmatrix} - 0.7 \times \begin{bmatrix} -0.2 - 0.4 \\ 0.75 - 0.05 \\ 1 - 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1.42 \\ 0.51 \\ -0.5 \end{bmatrix}$$

Visualization of this update:




# 6.2 XOR Problem and Multi-Layer Perceptron

## 6.2.1 XOR Problem

ℹ The XOR Problem illustrates:

- The inefficiency of a *Single Layer Perceptron* when,
- being faced with *Linear-Inseparable* data sets.

**Given:**

- Data samples:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathcal{X}$$

```
- where: $x_1, x_2\in\{0,1\}$
```

- The class of the data sample:

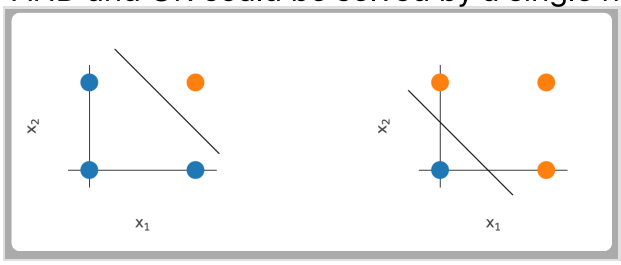
$$\omega_{\text{AND}}(\mathbf{x}) = x_1 \wedge x_2$$

$$\omega_{\text{OR}}(\mathbf{x}) = x_1 \vee x_2$$

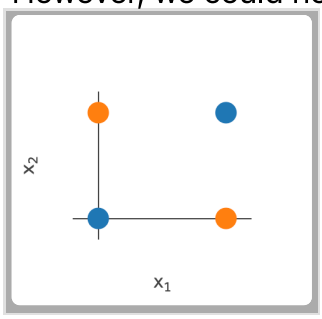
$$\omega_{\text{XOR}}(\mathbf{x}) = x_1 \oplus x_2$$

Do:

AND and OR could be solved by a single hyperplane.



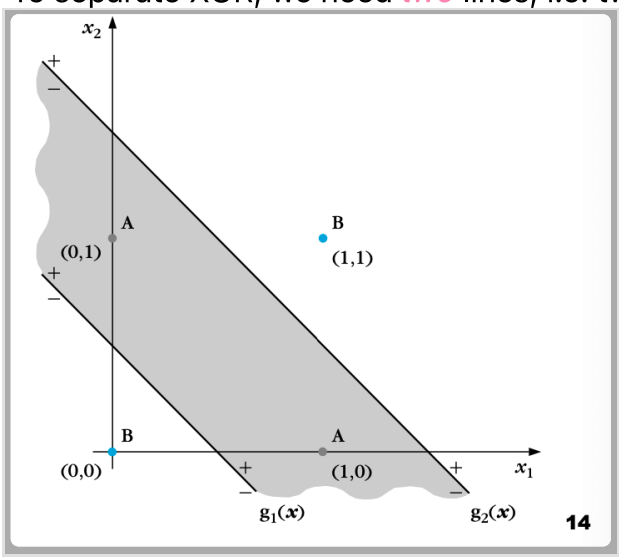

However, we could not decide a hyperplane that could separate XOR.




## 6.2.2 Two-Layer Perceptron

### Key Point 1: Intuitive Solution of XOR Problem

To separate XOR, we need *two* lines, i.e. two hyperplanes.




The two hyperplanes:

$$g_1(\mathbf{x}) = \mathbf{w}_1^\top \mathbf{x} = 0$$

$$g_2(\mathbf{x}) = \mathbf{w}_2^\top \mathbf{x} = 0$$

Intuitively, we could get the observation solution by:

$$g_1(\mathbf{x}) > 0 \ \wedge \ g_2(\mathbf{x}) < 0 \implies \mathbf{x} \in \omega_1$$

$$g_1(\mathbf{x}) < 0 \ \vee \ g_2(\mathbf{x}) > 0 \implies \mathbf{x} \in \omega_2$$

After, we activate the outputs by:

$$y_i(\mathbf{x}) = f(g_i(\mathbf{x}))$$

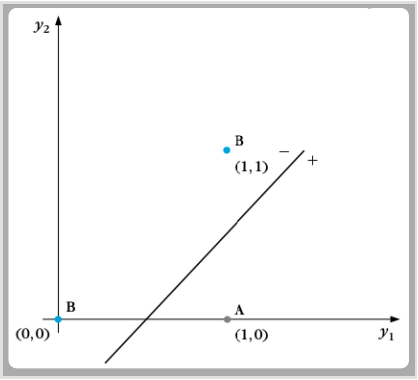
$$= \begin{cases} 0 & \text{if } g_i(\mathbf{x}) < 0 \\ 1 & \text{if } g_i(\mathbf{x}) > 0 \end{cases}$$

we could convert the solution as:

$$y_1(\mathbf{x}) = 1 \ \wedge \ y_2(\mathbf{x}) = 0 \implies \mathbf{x} \in \omega_1$$

$$y_1(\mathbf{x}) = 0 \ \vee \ g_2(\mathbf{x}) = 1 \implies \mathbf{x} \in \omega_2$$

With this expression, we could convert the solution into:



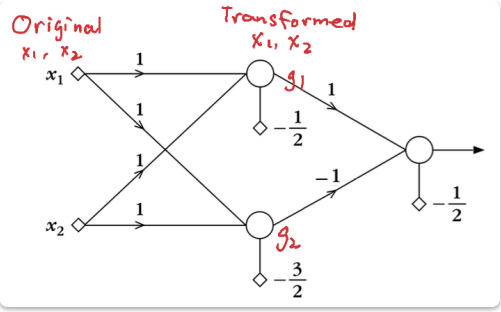

> *Note that $y_2 = 1 \implies y_1 \neq 0$, regions upper than the second line can't be lower than the first line. Therefore, the coordinate $(0, 1)$ is empty.*

This is a linear-separable case, which could be solved by the 1-layer perceptron.

- There are 2 axes.
- The value on each axes corresponds to:
  - The relative position of the original datapoint,
  - with respect to the separating hyperplane defined in the first layer.
  - e.g., $y_1 = 1$ means that the point is above the plane $g_1$.

## Key Point 2: More details with the XOR Problem's Solution

The previous XOR problem solution could be illustrated in the following *2-layer perceptron.*




### First Layer

The first layer has two neurons:

- Each neuron takes a 2-dimensional input, i.e., the data sample.
- Each neuron mimics a *Hyperplane.*
- The *output* of each neuron tells the *Region* separated by the hyperplane.

$$\text{First Neuron: } f(\begin{bmatrix}1 & 1 & -\frac{1}{2}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix}) = f(x_1 + x_2 - \frac{1}{2}) \in \{0,1\}$$

$$\text{Second Neuron: } f(\begin{bmatrix}1 & 1 & -\frac{3}{2}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\1\end{bmatrix}) = f(x_1 + x_2 - \frac{3}{2}) \in \{0,1\}$$

  The first layer *Maps* the linear inseparable data into linear separable ones.
- This is done with the help of $f$, the Activation Function.

$$f(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

### Second Layer

The second layer has only one neuron:

- It mimics the hyperplane that separates the *mapped data.*

$$\text{Output Neuron: } f(\begin{bmatrix}1 & -1 & -\frac{1}{2}\end{bmatrix}\begin{bmatrix}y_1\\y_2\\1\end{bmatrix}) = f(y_1 - y_2 - \frac{1}{2}) \in \{0,1\}$$

## 6.2.3 Three-Layer Perceptron

### Key Point 1: Two-Phase Process of Two-Layered Network

Let's take a look at the general form of a two-layered network
Given:

- Three hyperplanes:

$$g_1(\mathbf{x}) = \begin{bmatrix} 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0$$

$$g_2(\mathbf{x}) = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0$$

$$g_3(\mathbf{x}) = \begin{bmatrix} 0 & 1 & -\frac{1}{4} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = 0$$

Do:

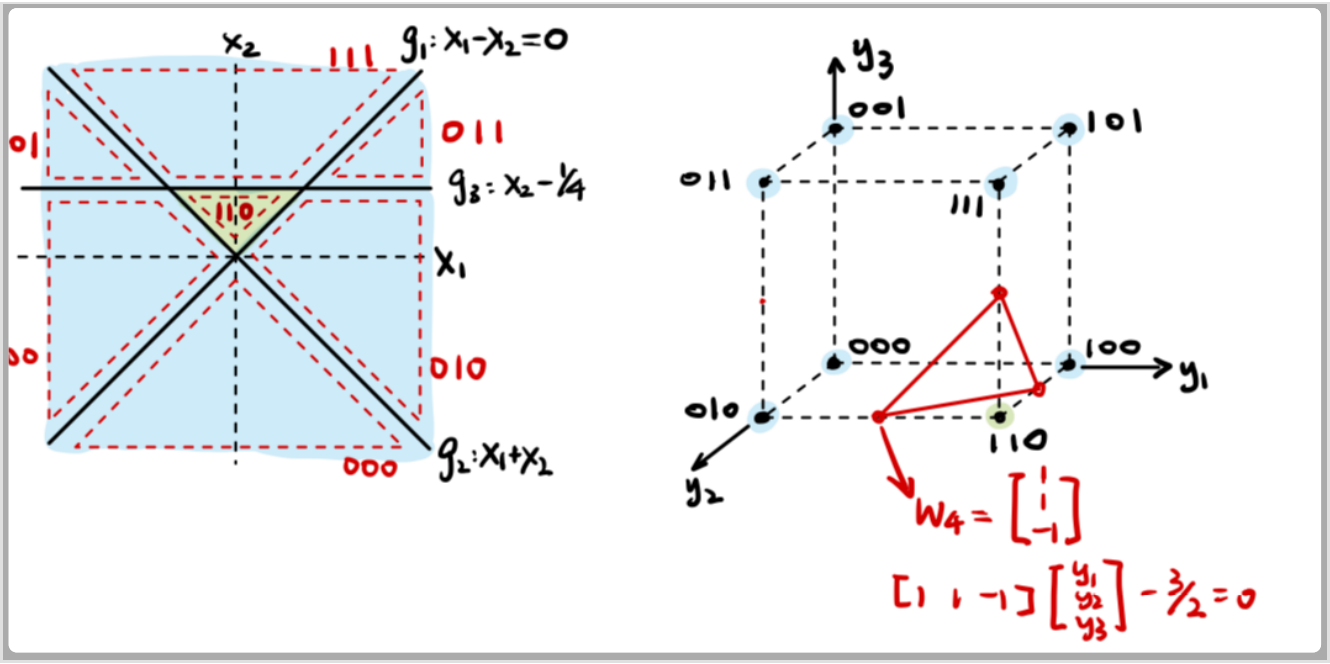
- Separate the central regions from others using a two-layered perceptron.

$$g_1(\mathbf{x}) > 0 \ \wedge \ g_2(\mathbf{x}) > 0 \ \wedge \ g_3(\mathbf{x}) < 0$$

- Namely, the regions named $110$.

*Phase 1.*

- The original space is of $l$ dimensions.
- Suppose there are $p$ hyperplanes (of $l$ dimensions) in the space.
- In the first layer, $l$-dimensional space will be transformed into a $p$-dimensional space.
  - The transformation is done with the help of Activation Function.




- By observation:
  - Regions in $l$-d space $\rightarrow$ Cube vertices in $p$-d space.
  - Separating regions in $l$-d space $\rightarrow$ Separating vertices in $p$-d space.

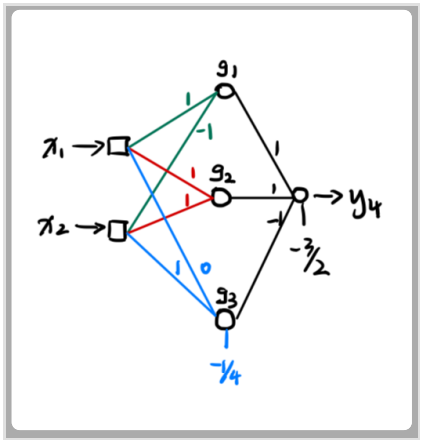
  *Phase 2.*

- The second layer, i.e., the output layer, separates vertices in $p$-dimensional space.
  - The single perceptron of the second layer mimics the hyperplane in the $p$-d space.
- In the given example, the hyperplane in $p$-d space is:

$$g_4(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -1 & -\frac{3}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

*Summary.*

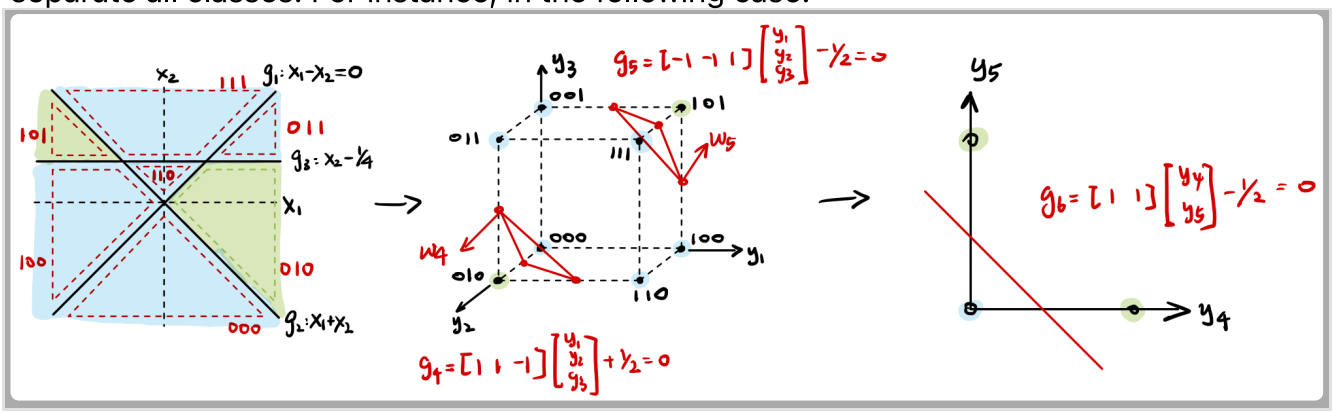
The overall structure of the two-layered neural network is:




- The first layer mimics the three lines, transforming $2$-d into $3$-d.
- The second layer mimics the hyperplane in $3$-d space that separates the vertices.

## Key Point 2: Deficiencies Two-Layered Network

By the example above, we could predict that the two-layered neural network can't separate all classes. For instance, in the following case:




- For disconnected regions in the $l$-space, their corresponding vertices are also "disconnected" in the $p$-space.
- We need $2$ hyperplanes in the $p$-space to separate them!
  Therefore, we need a 3-Phase Process.

## Key Point 3: Three-Phase Process of Three-Layered Network

*Phase 1.*

- In the $2$-d space, there are $3$ hyperplanes.
- The first layer uses $3$ perceptrons to mimic the $3$ hyperplanes.
- The original $2$-d space is thus transformed into a $3$-d space.

*Phase 2.*

- In the $3$-d space, we realized that we need to use two $p$-dimensional separating planes ($p = 2$) to separate the two vertices $010$ and $101$.

$$g_4(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

$$g_5(\mathbf{x}) = \begin{bmatrix} -1 & -1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ 1 \end{bmatrix} = 0$$

*Phase 3.*

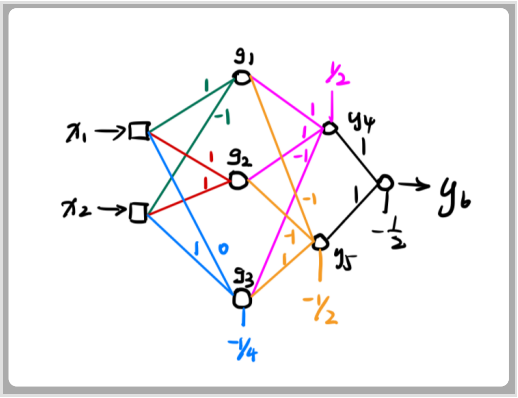
- The second layer transposes the intermediate $3$-d space into a new $2$-d space, considering there are $2$ separating planes.
- Separate the new linear-separable $2$-d vertices with the hyperplane:

$$g_6(\mathbf{x}) = \begin{bmatrix} 1 & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} y_4 \\ y_5 \\ 1 \end{bmatrix} = 0$$

The separation process is thus done.

*Summary.*

The overall structure of the three-layered neural network is:

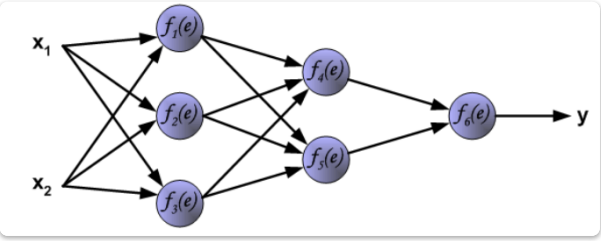



- The first layer mimics the three lines, transforming $2$-d into $3$-d.
- The second layer mimics the two hyperplanes in the $3$-d space, transforming $3$-d into $2$-d.
- The third layer, i.e., the output layer, mimics the separating line in the $2$-d space from the second layer.

# 6.3 Back Propagation Algorithm

# 6.3.1 Forward Propagation

We construct a neural network as follows.




## Key Point 1: Detailed Explanation of NN Structure

There are three layers in this neural network. Initially, weights are randomized.

### Layer 1: $2 \to 3$

- Weights:

$$\mathbf{W}_1 = \begin{bmatrix} \mathbf{w}_{11}^\top \\ \mathbf{w}_{12}^\top \\ \mathbf{w}_{13}^\top \end{bmatrix} = \begin{bmatrix} w_{11,1} & w_{11,2} & b_{11} \\ w_{12,1} & w_{12,2} & b_{12} \\ w_{13,1} & w_{13,2} & b_{13} \end{bmatrix}$$

### Layer 2: $3 \to 2$

- Weights:

$$\mathbf{W}_2 = \begin{bmatrix} \mathbf{w}_{21}^\top \\ \mathbf{w}_{22}^\top \end{bmatrix} = \begin{bmatrix} w_{21,1} & w_{21,2} & w_{21,3} & b_{21} \\ w_{22,1} & w_{22,2} & w_{22,3} & b_{22} \end{bmatrix}$$

### Layer 3: $2 \to 1$

- Weights:

$$\mathbf{W}_3 = \mathbf{w}_{31}^\top = \begin{bmatrix} w_{31,1} & w_{31,2} & b_{31} \end{bmatrix}$$

## Key Point 2: Forward Propagation Process

- Original Data with ground truth:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}, y_d$$

### Layer 1:

$$
\begin{aligned}
\mathbf{y}_1' &= f(\mathbf{W}_1\mathbf{x}) \\
&= f(\begin{bmatrix} w_{11,1} & w_{11,2} & b_{11} \\ w_{12,1} & w_{12,2} & b_{12} \\ w_{13,1} & w_{13,2} & b_{13} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}) \\
&= f(\begin{bmatrix} w_{11,1} \cdot x_1 + w_{11,2} \cdot x_2 + b_{11} \\ w_{12,1} \cdot x_1 + w_{12,2} \cdot x_2 + b_{12} \\ w_{13,1} \cdot x_1 + w_{13,2} \cdot x_2 + b_{13} \end{bmatrix}) \\
&= f(\begin{bmatrix} g_{11} \\ g_{12} \\ g_{13} \end{bmatrix}) \\
&= \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \end{bmatrix} \\
\mathbf{y}_1 &= \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix}
\end{aligned}
$$

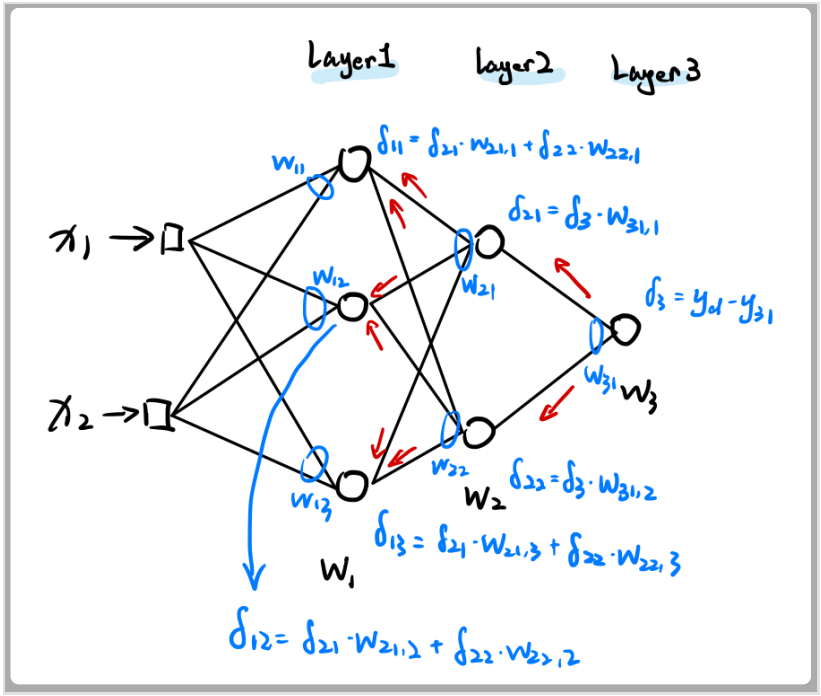
## Layer 2:

$$
\begin{aligned}
\mathbf{y}_2' &= f(\mathbf{W}_2\mathbf{y}_1) \\
&= f(\begin{bmatrix} w_{21,1} & w_{21,2} & w_{21,3} & b_{21} \\ w_{22,1} & w_{22,2} & w_{22,3} & b_{22} \end{bmatrix} \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix}) \\
&= f(\begin{bmatrix} w_{21,1} \cdot y_{11} + w_{21,2} \cdot y_{12} + w_{21,3} \cdot y_{13} + b_{21} \\ w_{22,1} \cdot y_{11} + w_{22,2} \cdot y_{12} + w_{22,3} \cdot y_{13} + b_{22} \end{bmatrix}) \\
&= f(\begin{bmatrix} g_{21} \\ g_{22} \end{bmatrix}) \\
&= \begin{bmatrix} y_{21} \\ y_{22} \end{bmatrix} \\
\mathbf{y}_2 &= \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix}
\end{aligned}
$$

## Layer 3:

$$
\begin{aligned}
\mathbf{y}_3' &= f(\mathbf{W}_3\mathbf{y}_2) \\
&= f(\begin{bmatrix} w_{31,1} & w_{31,2} & b_{31} \end{bmatrix} \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix}) \\
&= f(w_{31,1} \cdot y_{21} + w_{31,2} \cdot y_{22} + b_{31}) \\
&= f(g_{31}) \\
&= y_{31} \\
\mathbf{y}_3 &= y_{31}
\end{aligned}
$$

## Key Point 3: Back Propagation Algorithm




### Layer 3:

$$
\delta_3 = y_d - y_{31}
$$

### Layer 2:

$$
\delta_{21} = \delta_3 \cdot w_{31,1}
$$

$$
\delta_{22} = \delta_3 \cdot w_{31,2}
$$

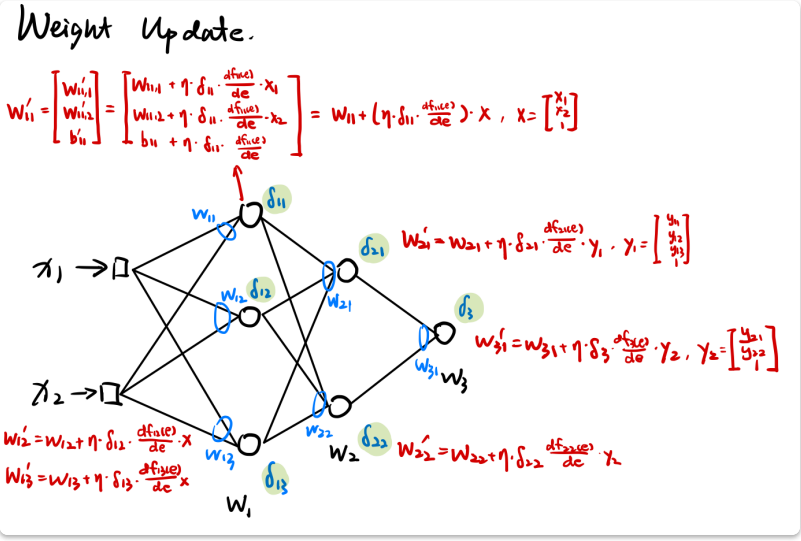
### Layer 1:

$$
\delta_{11} = \delta_{21} \cdot w_{21,1} + \delta_{22} \cdot w_{22,1}
$$

$$
\delta_{12} = \delta_{21} \cdot w_{21,2} + \delta_{22} \cdot w_{22,2}
$$

$$
\delta_{13} = \delta_{21} \cdot w_{21,3} + \delta_{22} \cdot w_{22,3}
$$

## Key Point 4: Weight Update




### Layer 1

$$\mathbf{w}'_{11} = \begin{bmatrix} w'_{11,1} \\ w'_{11,2} \\ b'_{11} \end{bmatrix}$$

$$= \begin{bmatrix} w_{11,1} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot x_1 \\ w_{11,2} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot x_2 \\ b_{11} + \eta \cdot \delta_{11} \cdot \dfrac{df_{11}(e)}{de} \cdot 1 \end{bmatrix}$$

$$= \mathbf{w}_{11} + \eta \cdot \delta_{11} \cdot \frac{df_{11}(e)}{de} \cdot \mathbf{x}$$

$$\mathbf{w}'_{12} = \mathbf{w}_{12} + \eta \cdot \delta_{12} \cdot \frac{df_{12}(e)}{de} \cdot \mathbf{x}$$

$$\mathbf{w}'_{13} = \mathbf{w}_{13} + \eta \cdot \delta_{13} \cdot \frac{df_{13}(e)}{de} \cdot \mathbf{x}$$

where,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

### Layer 2

$$\mathbf{w}'_{21} = \mathbf{w}_{21} + \eta \cdot \delta_{21} \cdot \frac{df_{21}(e)}{de} \cdot \mathbf{y}_1$$

$$\mathbf{w}'_{22} = \mathbf{w}_{22} + \eta \cdot \delta_{22} \cdot \frac{df_{22}(e)}{de} \cdot \mathbf{y}_1$$

where,

$$\mathbf{y}_1 = \begin{bmatrix} y_{11} \\ y_{12} \\ y_{13} \\ 1 \end{bmatrix}$$

### Layer 3

$$\mathbf{w}'_{31} = \mathbf{w}_{31} + \eta \cdot \delta_3 \cdot \frac{df_{31}(e)}{de} \cdot \mathbf{y}_2$$

where,

$$\mathbf{y}_2 = \begin{bmatrix} y_{21} \\ y_{22} \\ 1 \end{bmatrix}$$