# 01_Introduction_and_Bayesian Classification

## 1. Introduction

There are mainly three types of classical machine learning problems:

- Classification
- Regression and
- Clustering.

## 2. Classification: Bayesian Classifier

A classifier is something that can generate probability such that $p(y_1|x) > p(y_2|x)$.

### Introduction of Bayesian Classification

**Theorem: Bayes's Theorem**

$$p(w|x) = \frac{p(x|w)p(w)}{p(x)}$$

**Note:**

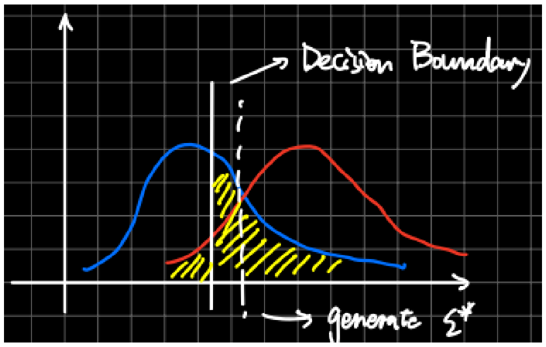
For a practical problem, $p(x|w)$ $p(w)$ $p(x)$ are actually unknown

## Classification Error

### Bayes’ Error

**Bayes's Error** is the theoretically minimum attainable error for a given classification problem.

The following is a illustration using **posterior probability distribution**



- Practically, for given classes, we do not know their true distribution, so that we cannot get $\epsilon^*$
- $\epsilon^*$ **does not depend on classification policy**, it is inertial property of true distribution of true classes

### Misclassification Cost

Sometimes, different misclassification has different costs. We use $\lambda_{ji}$ to modify the cost of mis-classify $w_j$ to $w_i$

Then the total risk can be presented by

$$
\begin{aligned}
l^i(x) &= \sum_{j=1}^{L} \lambda_{ji} p\left(w_j \mid x\right) (\text{ risk for assign } x \text{ to } w_j) \\
r^i &= \int l^i(x) p(x) dx \\
r &= \sum_i r^i = \sum_i \int \sum_{j=1} \lambda_{ji} p\left(w_j \mid x\right) dx
\end{aligned}
$$

By introducing the misclassification cost, we can then talk about a criterion, i.e. **minimize total risk**. Which means we need to optimize the classification due to following policy: we assign $x$ to $w_i$, when

$$\sum_j \lambda_{ji} p(w_i|x) dx \leq \sum_j \lambda_{jk} p(w_j|x)$$

## Bayesian Classifier

We will use Bayesian theorem to classify objects. Which means we need to know each part of Bayesian theorem, but actually, we can only estimate them:

- $\hat{p}(y) = \frac{N_y}{N}$

- $\hat{p}(\vec{x})=\sum_{i=1}^{c}\hat{p}\left(\vec{x}\mid y_i\right)\cdot\hat{p}\left(y_i\right)$
- $\hat{p}(\vec{x}|y_i)$

It can be seen that, $\hat{p}(\hat{x}|y_i)$ is still unknown. Then based on how to deal with $\hat{p}(\hat{x}|y_i)$, we can divide Bayesian Classifier into parametric version and non-parametric version.

## 3. Parametric Classifier

One classical solution of parametric classifier is we use **Gaussian Distribution** to estimate $\hat{p}(x|y_i)$

$$p(\vec{x})=\frac{1}{\sqrt{(2\lambda)^p\det(\Sigma)}}\exp\left(-\frac{1}{2}(x-\mu)^\top\Sigma^{-1}(x-\mu)\right)$$

The mean $\mu$ and the covariance matrix $\Sigma$ can be estimated by:

$$\hat{\mu}=\frac{1}{N}\sum_{i=1}^{N}\vec{x}_i\quad\sum_{i}^{N}=\frac{1}{N}\cdot\sum_{i=1}^{N}\left(\overrightarrow{x_i}-\hat{\mu}\right)\left(\vec{x}_1-\hat{\mu}\right)^\top$$

### Quadratic Classifier

In Quadratic Discriminant, we use the following Discriminant:

**Model: Quadratic Classifier**

$$f(\mathbf{x})=\log p\left(y_1\mid\mathbf{x}\right)-\log p\left(y_2\mid\mathbf{x}\right)=x^TWx+w^Tx+w_0$$

where

$$\begin{aligned}\mathbf{W}&=\frac{1}{2}\left(\Sigma_2^{-1}-\Sigma_1^{-1}\right)\\\mathbf{w}&=\mu_1^T\Sigma_1^{-1}-\mu_2^T\Sigma_2^{-1}\\w_0&=-\frac{1}{2}\log\det\Sigma_1-\frac{1}{2}\mu_1^T\Sigma_1^{-1}\mu_1+\log p\left(y_1\right)\\&\quad+\frac{1}{2}\log\det\Sigma_2+\frac{1}{2}\mu_2^T\Sigma_2^{-1}\mu_2-\log p\left(y_2\right)\end{aligned}$$

### Linear Classifier

When the $\Sigma$ are uninvertible matrices, we then use the following covariance instead. The Quadratic Discriminant will degenerated to Linear Discriminant.

$$\hat{\Sigma}_k=\frac{1}{c}\cdot\sum_{k=1}^{c}\hat{\Sigma}_k$$

- When two classes have the same $\Sigma$, the classifier will degenerated to a linear classifier

**Model: LDA**

$$\begin{aligned} f(x) &= w^T x + w_0 \\ \mathbf{w} &= \hat{\Sigma}^{-1} \left( \hat{\mu}_2 - \hat{\mu}_1 \right) \\ w_0 &= \frac{1}{2} \hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log \frac{p\left(y_1\right)}{p\left(y_2\right)} \end{aligned}$$

## Nearest Mean Classifier

But sometimes, the covariance is so extreme that is still invertible. Then we use covariance as follows:

$$\hat{\Sigma} = \sigma^2 \mathbb{I}$$

The classifier then become **Nearest Mean Classifier**

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T \mathbf{x} + w_0 \\ \mathbf{w} &= \hat{\mu}_2 - \hat{\mu}_1 \\ w_0 &= \frac{1}{2} \hat{\mu}_2^T \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\mu}_1 + \sigma^2 \log \frac{p\left(y_1\right)}{p\left(y_2\right)} \end{aligned}$$

**Note:**

It only uses distance to the mean of each of the classes and $w_0$ actually add some bias. That is the decision boundary will not be the middle line of the line between two mean point

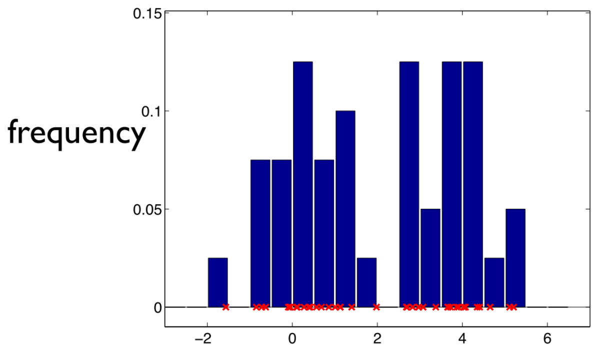
# 4. Non-Parametric Classifier

Sometimes the Gaussian Assumption may not be suitable, so we will need try other ways to **build the distribution estimation**

## Histogram Method

In histogram method, we split the data and count number $K_N$, then we have

$$\hat{p}(\mathbf{x}) = \hat{p}(\hat{\mathbf{x}}) \approx \frac{1}{h} \frac{k_N}{N}, \quad |\mathbf{x} - \hat{\mathbf{x}}| \leq h/2$$



## Parzen Density Estimation

In parzen density estimation method:

**Model: Parzen Density Estimation**

- Define shape of cells, add the density $K(r, h)$
- locate cells on training data
- Add contribution of cells
- Get the density estimation of test object $z$

$$\hat{p}(\mathbf{z} \mid h) = \frac{1}{n} \sum_{i=1}^{n} K\left(\left\|\mathbf{z} - \mathbf{x}_i\right\|, h\right)$$

One of the classical choice of the shape is

$$K(r, h) = \begin{cases} 0 & \text{if } |r| > h \\ 1/V & \text{if } |r| \leq h \end{cases}$$

## Nearest Neighbor Classification

In nearest neighbor classification, the density is estimated by:

$$\hat{p}(x|w_m) = \frac{k_m}{n \cdot V_k} \quad \longrightarrow V_k \text{ is the volum of } k\text{-th } neighbor$$

We can prove that the classification can then degenerated to compare the number of neighbor of each class

$$\begin{aligned} &\hat{p}\left(x \mid w_m\right) = \frac{k_m}{n_m \cdot V_k}; \quad p\left(w_m\right) = \frac{n_m}{n} \\ &\hat{p}\left(w_m \mid x\right) = \frac{p\left(x \mid w_m\right) \cdot p\left(w_m\right)}{p(x)} = \frac{\frac{k_m}{n_m \cdot V_k} \cdot \frac{n_m}{n}}{p(x)} \\ &\hat{p}\left(w_m \mid x\right) > \hat{p}\left(w_n \mid k\right) \Rightarrow k_m > k_n \end{aligned}$$

**Note:**

There is an implicitly hidden probability $p(x)$, the $p(x)$ is related to intuition: $p(x) = \frac{1}{V_k}$, that is the density of the volumn of kth neighbor.

**Method: KNN**

- locate the cell on the test object $\vec{x}$
- grow the cell until it covers $k$ objects
- find the majority of the K neighborhoods

## 5. Important Thing about Implementation

**Scale Features**

## Summary

- Bayes Error
- Bayesian Classification
  - Bayesian Rule
  - Lack true distribution:
    - parametric: assume Gaussian Distribution
      - QDA, LDA NMC
    - non-parametric:
      - histogram density
      - parzen density
      - knn
- Always Scale Features