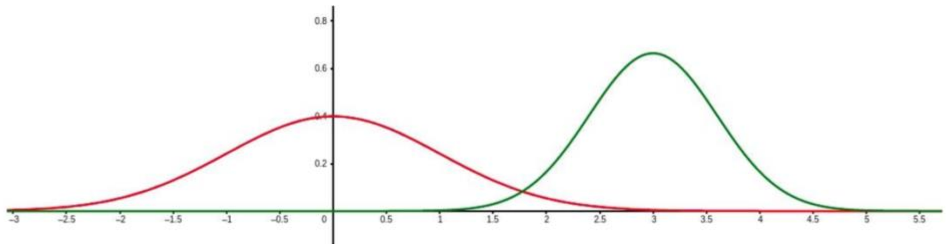
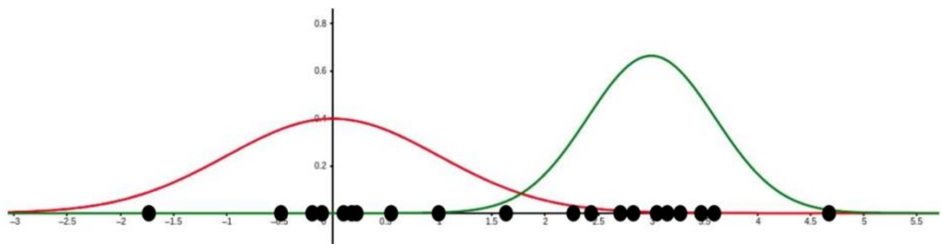
Soft Clustering:

Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$



Generate data using $\mathbf{N(\mu_1, \sigma_1)}$ and $\mathbf{N(\mu_2, \sigma_2)}$



Mixture Model:

X comes from a mixture model with k mixture components if the probability distribution of X is

$$P(X = x) = \sum_{j=1}^{k} P(C_j) P(X = x | C_j)$$

Mixture proportion
Represents the probability
of belonging to C_j

Probability of seeing x
when sampling from C_j

Gaussian Mixture Model:

A Gaussian Mixture Model (GMM) is a mixture model where

$$P(X = x | C_i) \sim N(\mu, \sigma)$$

GMM Clustering:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{n} \sum_{j=1}^{k} P(C_j) P(X_i \mid C_j)$$

It is a joint probability distribution of our data and we assume our data are independent

We can define the following:

$$\begin{aligned} l(\theta) &= \log(L(\theta)) \\ &= \sum_{i=1}^{n} \log(\sum_{j=1}^{k} P(C_j)P(X_i \mid C_j)) \end{aligned}$$

For $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k]^T$ and $\boldsymbol{\Sigma} = [\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_k]^T$

We can solve

$$\frac{d}{d\Sigma} l(\theta) = 0 \qquad \frac{d}{d\mu} l(\theta) = 0$$

And then we can get:

$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i)X_i}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^{n} P(C_j|X_i)(X_i - \hat{\mu}_j)^T(X_i - \hat{\mu}_j)}{\sum_{i=1}^{n} P(C_j|X_i)}$$

$$\hat{P}(C_j) = \frac{1}{n} \sum_{i=1}^{n} P(C_j|X_i)$$

$$\begin{aligned} P(C_j|X_i) &= \frac{P(X_i|C_j)}{P(X_i)} P(C_j) \\ &= \frac{P(X_i|C_j)P(C_j)}{\sum_{j=1}^{k} P(C_j)P(X_i|C_j)} \end{aligned}$$

Expectation Maximization Algorithm:

1. Start with random θ
2. Compute P(Cj | XI) for all Xi by using θ
3. Compute / Update θ from P(Cj | XI)
4. Repeat 2 & 3 until convergence

Naïve Bayes and SVM
Bayes Classifier is a probabilis8c framework for solving classifica8on problems
Condi8onal Probability:

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Bayesian Classifiers: Consider each atrribute and class label as random variables
Given a record with aRributes (A1, A2,…,An)
– Goal is to predict class C
– Specifically, we want to find the value of C that maximizes P(C| A1, A2,…,An)

Approach: compute the posterior probability P(C | A1, A2, …, An) for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C)P(C)}{P(A_1 A_2 \ldots A_n)}$$

Example:

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  – One for each (A_i,c_i) pair
- For (Income, Class=No):
  – If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$$

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)
=> Class = No

Probability estimation:

$$\text{Original}: P(A_i \mid C) = \frac{N_{ic}}{N_c}$$

$$\text{Laplace}: P(A_i \mid C) = \frac{N_{ic} + 1}{N_c + c}$$

$$\text{m - estimate}: P(A_i \mid C) = \frac{N_{ic} + mp}{N_c + m}$$

Naïve Bayes summary
• Robust to isolated noise points
• Handle missing values by ignoring the instance during probability estmate calculations
• Robust to irrelevant atrributes
• Independence assumption may not hold for some aRributes

Support Vector Machines:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \ge 1 \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \le -1 \end{cases}$$

Introduce slack variables:

- Need to minimize: $L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$

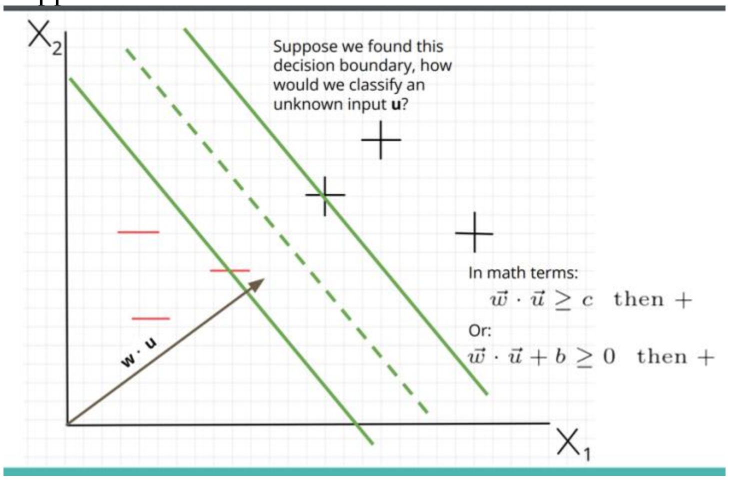
- Subject to:

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \ge 1 - \xi_\mathrm{i} \\ -1 & \text{if } \vec{\mathrm{w}} \bullet \vec{\mathrm{x}}_\mathrm{i} + \mathrm{b} \le -1 + \xi_\mathrm{i} \end{cases}$$

Kernel Suggestion:

We do not need to actually map explicitly each point to a high dimensional space!
We just need to have a func8on that computes the similarity (distance) in the mapped space given the points in the input space (without the need to do the mapping!)

Support Vector Machines



Ways to find the widest street:

We want our samples to lie beyond the street. That is:

$$\vec{w} \cdot \vec{x}_{+} + b \geq 1$$

$$\vec{w} \cdot \vec{x}_{-} + b \leq -1$$

Note: for an unknown u, we can have:

$$-1 < \vec{w} \cdot \vec{u} + b < 1$$

Introducing a variable:

$$y_i = \begin{cases} +1 & \text{if } x_i \text{ is a } + \text{sample} \\ -1 & \text{if } x_i \text{ is a } - \text{sample} \end{cases}$$

If we multiply our sample decision rules by this new variable:

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1$$

Meaning, for on the decision boundary, we want:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

We know that **WIDTH** $= (\vec{x}_+ - \vec{x}_-) \cdot \frac{\vec{w}}{\|\vec{w}\|}$ for $\vec{x}_-$ and $\vec{x}_+$ points on the boundary

And, since they are on the boundary, we know that

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

Hence, **WIDTH** $= \dfrac{2}{\|\vec{w}\|}$

Goal is to maximize the width:

$$\begin{aligned} \max\left(\frac{2}{\|\vec{w}\|}\right) &= \min(\|\vec{w}\|) \\ &= \min\left(\frac{1}{2}\|\vec{w}\|^2\right) \end{aligned}$$

Subject to:

$$y_i(\vec{w} \cdot \vec{x}_i + b) - 1 = 0$$

Can use Lagrange multipliers to form a single expression to find the extremum of

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum_i \alpha_i \left[y_i(\vec{x}_i \cdot \vec{w} + b) - 1\right]$$

$$\begin{aligned} \frac{\partial L}{\partial \vec{w}} &= \vec{w} - \sum_i \alpha_i y_i \vec{x}_i = 0 \\ &\implies \vec{w} = \sum_i \alpha_i y_i \vec{x}_i \end{aligned}$$

$$\begin{aligned} L &= \sum_i \alpha_i - \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_i \alpha_i y_i \vec{x}_i\right) \\ &= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \end{aligned}$$

To find Φ:

$$K(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$$

$$K(\vec{x}_i, \vec{x}_j) = (\vec{x}_i \cdot \vec{x}_j + 1)^n$$

$$K(\vec{x}_i, \vec{x}_j) = e^{\frac{\|\vec{x}_i - \vec{x}_j\|}{\sigma}}$$