Chapter 4: Nonparametric Techniques

4.1 Introduction

Potential problems for parametric techniques:

- The assumed parametric form **may not fit the ground-truth density** encountered in practice, e.g:
  - **Assumed parametric form**: Unimodal(such as Gaussian pdf)
  - **Ground-truth density**: Multimodal(such as mixture of Gaussian pdfs)

So here comes the nonparametric techniques: $p(x|w_j)$ doesn't have parametric form. We use the data to speak for themselves.

1. **Parzen Windows**
2. k_n**-nearest-neighbor**

4.2 Probability Density Estimation

General Settings

- **Feature space**: $\mathcal{F} = \mathbf{R}^d$
- **Feature vector**: $x \in \mathcal{F}$
- **pdf function**: $x \sim p(\cdot)$

Fundamental Fact: The probability (**a smoothed/averaged version of** $p(x)$) of a vector x **falling into a region** $\mathcal{R} \subset \mathcal{F}$:

$$P = Pr[x \in R] = \int_{\mathcal{R}} p(x^{'})dx^{'}$$

Given n examples (i.i.d.)

$$x_1, x_2, \cdots, x_n$$

with $x_i \sim p(\cdot) \quad (1 \leq i \leq n)$

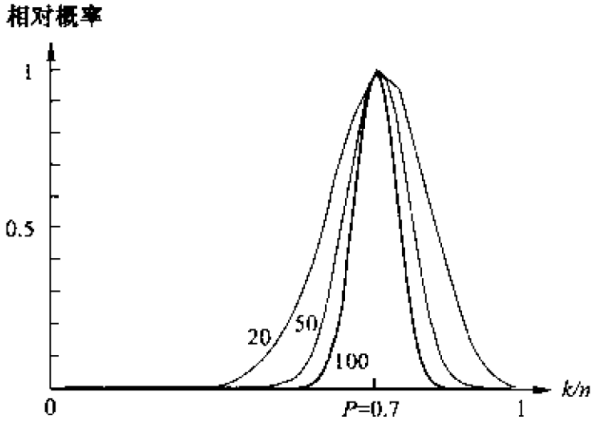
Let X be the (discrete) **random variable** representing the number of examples falling into $\mathcal{R}$, so X will take **Binomial Distribution**: $X \sim \mathcal{B}(n, P)$ and we have:

$$Pr[X = r] = \binom{n}{r} P^r (1-P)^{n-r}$$
$$\mathcal{E}[X] = nP \quad \Rightarrow \quad P = \frac{\mathcal{E}[X]}{n}$$

If we assume that $p(x)$ is constant and region $\mathcal{R}$ is so small that $p(\cdot)$ hardly varies with $\mathcal{R}$, then we have:

$$P = Pr(x \in \mathcal{R}) = \int_R p(x^{'})dx^{'} \approx p(x) \int_{\mathcal{R}} 1 dx^{'} = p(x)V$$

- x is a point in $\mathcal{R}$
- V is the volume enclosed by $\mathcal{R}$



- k/n: success / total
- all curve converge to a **single peak** at $\mathcal{E}(X)$
- when n get larger, curve gets sharper around $\mathcal{E}(X)$

X **peaks sharply** about $\mathcal{E}(X)$ when n is large enough.

Let k be the actual value of X after observing the i.i.d. training examples $x_1, x_2, \cdots, x_n$

Given those evidence, we have $k \approx \mathcal{E}[X]$, hence we can conclude that:

$$p(x) = \frac{k/n}{V}$$

If we fixed the value of V and are able to get more training examples, then the value of k/n will converge to the true value of $p(x)$, which is a **smoothed version**:

$$\frac{P}{V} = \frac{\int_{\mathcal{R}} p(x^{'})dx^{'}}{\int_{\mathcal{R}} dx^{'}}$$

However, we hope to estimate $p(x)$ which is not a smoothed version, so we need to make V approach 0, which lead to the following problems:

- $V \to 0$, the region $\mathcal{R}$ will become so small that there my be no training examples falling into the region, which makes no sense.
- If 1 or 2 sample accidentally fall into the region $\mathcal{R}$, then the estimation will become infinite, which is also not reasonable.

Theoretically, we construct a series fo regions $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_m$ which contain the dot x to overcome the limitations. The first region use 1 sample, the second region use 2 samples, and so on. Hence we have:

$$p_n(x) = \frac{k_n/n}{V_n}$$

- V_n: the volume of region $\mathcal{R}_n$
- n: the number of training examples
- k_n: the number of training examples falling into $\mathcal{R}_n$

Requirements to be satisfied, which make $p_n(x)$ converge to $p(x)$:

1. $\lim_{n\to\infty} V_n = 0$: ensure the space-smoothed P/V converge to $p(x)$ under the situation of uniform convergence in the region and continuity of $p(\cdot)$ at point x.
2. $\lim_{n\to\infty} k_n = \infty$: meaningful only when $p(x) \neq 0$ and guarantees that the ratio of frequencies can converge to the probability P
3. $\lim_{n\to\infty} k_n/n = 0$

Methods:

1. **Parzen Windows**: Fix V_n and determine k_n
2. k_n**-nearest-neighbor**: Fix k_n and determine V_n

4.3 Parzen Windows

Assumptions:

- $\mathcal{R}_n$: a d-dimensional hypercube
- h_n: the length of the edge of $\mathcal{R}_n$

Determine k_n with **window function**, a.k.a. **kernel function**, **potential function**, etc.

- **Properties of window functions**
  - non-negativity: $\varphi(\mathbf{u}) \geq 0$
  - uniformity: $\int \varphi(\mathbf{u}) d\mathbf{u} = 1$
- **Extension**
  - $\varphi(\cdot)$ is not limited to be the following hypercube window function but any pdf function that satisfies the above two properties.

Define window function:

$$\varphi(\mathbf{u}) = \begin{cases} 1 & |u_j| \leq \frac{1}{2},\ j = 1, \ldots, d, \\ 0 & \text{otherwise} \end{cases}$$

$\varphi(\mathbf{u})$ defines a **unit hypercube** centered at the origin and $V_n = h_n^d$

$\varphi(\frac{x - x_i}{h_n}) = 1 \quad \Leftrightarrow \quad x_i$ falls within the hypercube of volume V_n centered at x with edge length h_n

- $x - x_i$: the relative position between the sample point x_i and the target piont x
- h_n: bandwidth parameter which controls the width of the window

Hence the total number of the sample points falling into the hypercube can be calculated as:

$$k_n(x) = \sum_{i=1}^{n} \varphi(\frac{x - x_i}{h_n})$$

According to the probability density, we have:

$$p_n(x) = \frac{k_n/n}{V_n} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi(\frac{x - x_i}{h_n})$$

- **meaning**: the average contribution of the sample points to the target point x
- The kernel function φ weights the influence of each sample point x_i on x:
    1. If x_i is very close to x (i.e., $|x - x_i|$ is small), then $\varphi(\frac{x - x_i}{h_n})$ will approach 1.
    2. If x_i is far from x, $\varphi(\frac{x - x_i}{h_n})$ will approach 0.
    3. $\varphi(\cdot)$ is a pdf function $\quad \Rightarrow \quad p_n(x)$ is a pdf function.

The proof of 3 is as follows:

- **Notice**: x is a multi-dimensional vector and the window bandwidth h_d do the same for all the dimensions. When using the substitution $\mathbf{u} = \frac{x - x_i}{h_n}$, we get $dx = h_n^d d\mathbf{u}$

$$\varphi(\cdot) \geq 0 \Rightarrow p_n(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi(\frac{x - x_i}{h_n}) \geq 0$$

$$\begin{aligned} \int p_n(x) dx &= \frac{1}{nV_n} \sum_{i=1}^{n} \int \varphi(\frac{x - x_i}{h_n}) dx \\ &= \frac{1}{nV_n} \sum_{i=1}^{n} \int h_n^d \varphi(\frac{x - x_i}{h_n}) d(\frac{x - x_i}{h_n}) \\ &= \frac{V_n}{nV_n} \sum_{i=1}^{n} \int \varphi(\mathbf{u}) d\mathbf{u} = 1 \end{aligned}$$

Above all we can conclude that:

- **windows function (being pdf)** $\varphi(\cdot)$ + **window width** h_n + **training data** x_i = **Parzen pdf** $p_n(\cdot)$

Now we exploit the effect of window width h_n on the Parzen pdf $p_n(x)$. We define function $\delta_n(x)$ as:

$$\delta_n(x) = \frac{1}{V_n}\varphi(\frac{x}{h_n}) = \frac{1}{h_n^d}\varphi(\frac{x}{h_n})$$

- window width h_n affect two key attributes of partial contribution function $\delta_n(\cdot)$:
  1. **Amplitude**: determined by $\frac{1}{h_n^d}$, the larger h_n, the smaller the amplitude.
  2. **Width**: determined by $\frac{x}{h_n}$, the larger h_n, the wider the width. (**larger width cause slower change so as to increse effective range**)
- $\delta_n(\cdot)$ being a pdf function

$$\begin{aligned} \delta_n(\cdot) &\geq 0 \\ \int \delta_n(x)dx &= \int \frac{1}{h_n^d}\varphi(\frac{x}{h_n})dx \\ &= \int \frac{1}{h_n^d} \cdot \varphi(\mathbf{u}) \cdot h_n^d d\mathbf{u} = 1 \end{aligned}$$

We can rewrite $p_n(x)$ as:

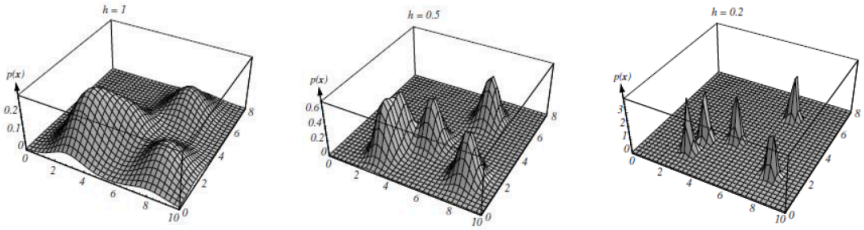
$$p_n(x) = \frac{1}{n}\sum_{i=1}^{n}\delta_n(x - x_i)$$

- $p_n(x)$: **superposition** of n sample contributions interpolations
- x_i: contributes to $p_n(x)$ based on the **distance** between x and x_i
- **Influence**:
  - h_n **very large**
    1. $\delta_n(x)$ being board with small amplitude
    2. $p_n(x)$ will be the superposition of n board, slowly changing functions, i.e. **being smooth with low resolution**
    3. may cause **underfitting** and be unable to capture local structure
  - h_n **very small**
    1. $\delta_n(x)$ being sharp with large amplitude
    2. $p_n(x)$ will be the superposition of n sharp pulses, i.e. **being variable/unstable with high resolution**
    3. may cause **overfitting** and can be more affected by noise

A compromised value of h_n should be sought for limited number of training examples.

Suppose $\varphi(\cdot)$ being a 2-d *Gaussian pdf* and n=5

The shape of $p_n(x)$ with decreasing values of h_n



h=1.0 h=0.5 h=0.2

4.4 k_n-nearest-neighbor

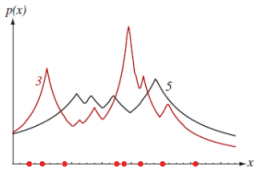
$$p_n(x) = \frac{k_n/n}{V_n}$$

- Fix k_n then determine V_n
- **Procedure**: specify $k_n \rightarrow$ centere a cell about $x \rightarrow$ grow the cell until capturing k_n nearest exmaples $\rightarrow$ return cell volumes as V_n
- The principled rule to specify k_n (the **sufficient and necessary conditions** of $p_n(x)$ converging to $p(x)$):
    1. $lim_{n\rightarrow\infty} k_n = \infty$
    2. $lim_{n\rightarrow\infty} \frac{k_n}{n} = 0$
- A rule-of-thumb (经验法则) choice for k_n: $k_n = \sqrt{n}$

Eight points in one dimension (n=8, d=1)

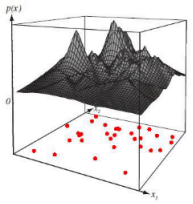
red curve: k_n=3

black curve: k_n=5



Thirty-one points in two dimensions (n=31, d=2)

black surface: k_n=5



- 1-dimension example
    - Smaller k_n produces detailed but noisier estimates
    - Larger k_n creates smoother estimates but with reduced local detail.
- 2-dimension example
    - The $k_n = 5$ estimation dynamically adapts to the data density, showing higher values in dense areas and lower values in sparse regions.

4.5 Nearest-Neighbor Rule

4.5.1 Definition

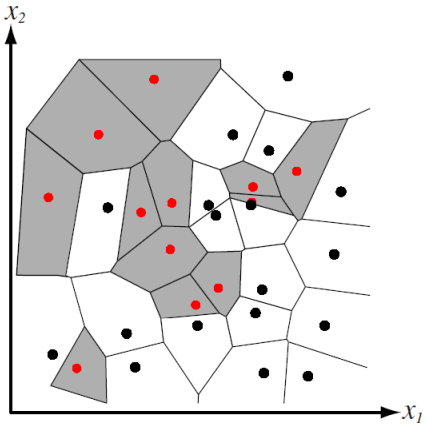
Given:

- The **label space**: $\Omega = \{\omega_1, \omega_2, \ldots, \omega_c\}$
- A set of n labeled training examples: $\mathcal{D}^n = \{(\mathbf{x}_i, \theta_i) \mid 1 \leq i \leq n\}$, where:
  - $\mathbf{x}_i \in \mathbb{R}^d$: The i-th feature vector in d-dimensional space
  - $\theta_i \in \Omega$: The label associated with $\mathbf{x}_i$

For a test example $\mathbf{x}$:

1. Identify its nearest neighbor $\mathbf{x}'$: $\mathbf{x}' = \arg\min_{\mathbf{x}_i \in \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}} D(\mathbf{x}_i, \mathbf{x})$,
   - $D(\mathbf{a}, \mathbf{b})$: distance metric between two vectors $\mathbf{a}$ and $\mathbf{b}$, such as the Euclidean distance.
2. Assign the label θ' associated with $\mathbf{x}'$ to $\mathbf{x}$.

4.4.2 Voronoi tessellation (维诺图)



Each training example $\mathbf{x}$ leads to a cell in the Voronoi tessellation

- any point in the cell is closer to $\mathbf{x}$ than to any other training examples
- partition the feature space into n cells
- any point in the cell share the same class label as $\mathbf{x}$

4.4.3 Error Rate of Nearest Neighbor Rule

$$P(e) = \int P(e|x)p(x)dx$$

- $P(e)$: The **average probability of error** based on nearest-neighbor rule
- $P(e \mid x)$: The probability of making an errorneous classification on x based on nearest-neighbor rule
- $p(x)$: The probability density function of the feature space, which describes how data points are distributed

According to the **Bayes Decision Rule**, we can minimize the error rate at each point x by choosing the class label w_m with the maximum posterior probability:

$$P^*(e|x) = 1 - P(w_m|x)$$

- $P(w_m|x)$: the posterior probability of the most likely class at x, i.e. $P(w_m|x) = \max_{1 \leq i \leq c} P(w_i|x)$

- $P^*(e|x)$: the **minimum** possible value of $P(e|x)$, i.e. the one given by Bayesian decision rule

The Bayes risk(under zero-one loss) can be written as:

$$P^*(e) = \int P^*(e|x)p(x)dx$$

4.4.4 Error Bounds of Nearest Neighbor Rule

$$P^*(e) \leq P(e) \leq P^*(e)\big(2 - \frac{c}{c-1}P^*(e)\big)$$

- $P_n(e)$: The probability of error under the NN rule with n training examples
- $P = \lim_{n\to\infty} P_n(e)$: the asymptotic probability of error as the number of training examples approaches infinity
- c: the number of class labels

A conditional probability of error $P(e|x, x^{'})$ relying on testing sample x and the nearest neighbor $x^{'}$:

$$P(e|x) = \int P(e|x, x^{'})p(x^{'}|x)dx^{'}$$

When $\theta \neq \theta^{'}_{n'}$ it produce a classification error:

$$\begin{aligned} P_n(e|x, x^{'}_n) &= 1 - \sum_{i=1}^{c} P(\theta = w_i, \theta^{'}_n = w_i|x, x^{'}_n) \\ &= 1 - \sum_{i=1}^{c} P(w_i|x)P(w_i|x^{'}_n) \end{aligned}$$

According to the fact that $p(x^{'}|x)$ peaks sharp around x but stay low at other locations, we can assume $p(x^{'}|x)$ approach a **Dirac delta function** δ centered at x when n approaches infinity. Hence we have:

- When $\delta(x - x_0)$ being a integral kernel, we have : $\int f(x)\delta(x - x_0)dx = f(x_0)$
- When $n \to \infty$, $P(w_i|x^{'}_n \to P(w_i|x))$, hence we have $\sum_{i=1}^{c} P(w_i|x)P(w_i|x^{'}_n) \to \sum_{i=1}^{c} P^2(w_i|x)$

$$\begin{aligned} \lim_{n\to\infty} P_n(e|x, x^{'}_n) &= \int \Big[1 - \sum_{i=1}^{c} P(w_i|x)P(w_i|x^{'}_n)\Big]\delta(x^{'} - x)dx^{'} \\ &= 1 - \sum_{i=1}^{c} P^2(w_i|x) \end{aligned}$$

Hence we have asymptotic probability of NN error:

$$
\begin{aligned}
P &= \lim_{n\to\infty} P_n(x) \\
&= \lim_{n\to\infty} \int P_n(e|x)p(x)dx \\
&= \int \Big[1 - \sum_{i=1}^{c} P^2(w_i|x)\Big]p(x)dx
\end{aligned}
$$

Proof of the lower bound:

$P^*(e)$ is the minimum possible value of $P(e)$ given by the Bayesian decision rule, which is the optimal classifier.

Proof of the upper bound:

$$
\sum_{i=1}^{c} P^2(w_i|x) = P^2(w_m|x) + \sum_{i\neq m} P^2(w_i|x)
$$

- **Constraints**:
  - $P(w_i|x) \geq 0$
  - $\sum_{i\neq m} P^(w_i|x) = 1 - P(w_m|x) = P^*(e|x)$

To make the minimum of $\sum_{i=1}^{c} P^2(w_i|x)$, we let:

$$
P(w_i|x) = \begin{cases} \frac{P^*(e|x)}{c-1} & i \neq m \\ 1 - P^*(e|x) & i = m \end{cases}
$$

Hence we have :

$$
\begin{aligned}
\sum_{i=1}^{c} P^2(w_i|x) &\geq (1 - P^*(e|x))^2 + \frac{P^{*2}(e|x)}{c-1} \\
\therefore 1 - \sum_{i=1}^{c} P^2(w_i|x) &\leq 2P^*(e|x) - \frac{c}{c-1}P^{*2}(e|x)
\end{aligned}
$$

Directly we can get: $P \leq 2P^*$

Notice that:

$$
\begin{aligned}
Var[P^*(e|x)] &= \mathcal{E}(P^{*2}(e|x)) - E(P^*(e|x))^2 \\
&= \int P^{*2}(e|x)p(x)dx - (\int P^*(e|x)p(x)dx)^2 \\
&= \int P^{*2}(e|x)p(x)dx - P^{*2} \geq 0
\end{aligned}
$$

therefore we have:

$$
\int P^{*2}(e|x)p(x)dx \geq P^{*2}
$$

According to the fact we get before:

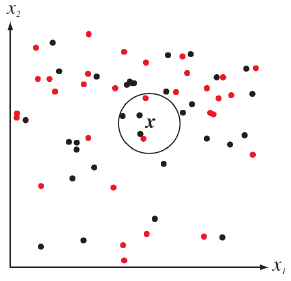
$$
\begin{aligned}
P &= \int \left[1 - \sum_{i=1}^{c} P^2(w_i|x)\right] p(x)dx \\
&\leq \int \left[2P^*(e|x) - \frac{c}{c-1} P^{*2}(e|x)\right] p(x)dx \\
&= 2P^* - \int \frac{c}{c-1} P^{*2}(e|x)p(x)dx \\
&\leq 2P^* - \frac{c}{c-1} P^{*2} = P^*(2 - \frac{c}{c-1} P^*)
\end{aligned}
$$

4.4.5 k_n-nearest-neighbor Rule

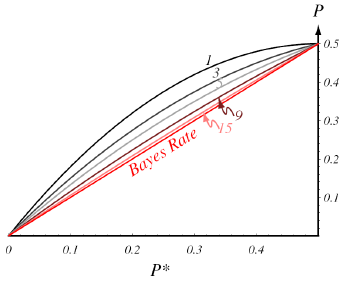
Classification with k-nearest-neighbor rule

- Given the label space $\Omega = \omega_1, \omega_2, \ldots, \omega_c$ and a set of n labeled training examples $D^n = (x_i, \theta_i) \quad 1 \leq i \leq n$, where $x_i \in \mathbb{R}^d$ and $\theta_i \in \Omega$:
  - For test example x, identify $S' = \{x_i \mid x_i \text{ is among the } KNN \text{ of } x\}$ and then assign the most frequent label w.r.t. S', i.e., $\arg\max_{\omega_i \in \Omega} \sum_{x_i \in S'} 1_{\theta_i = \omega_i}$ to x.
    - 1_θ : an indicator function which returns 1 if predicate θ right, and 0 otherwise.

For binary classification problem ($c = 2$), and odd value of k is generally used to avoid ties.



c=2, k=5



the error rate of kNN rule (i.e. P) upper-bounded by the Bayes risk (i.e. P*) for binary classification (c=2)

4.4.6 Computational Complexity of KNN Rule

Given n labeled training examples in d-dimensional space, the computational complexity of **classifying one test example** is $O(dn)$

General ways of reducing computational burden:

- **Calculate Partial Distance**: $D_r(a, b) = \left(\sum_{j=1}^{r} (a_j - b_j)^2\right)^{\frac{1}{2}} \ (r < d)$
  - Using a subset r of the total **d-dimensional space**
  - As more dimensions are gradually added, the values of partial distances are strictly **non-decreasing**
- **Pre-structing**: create some form of search tree, where nearest neighbors are **recursively** identified following the tree structure (each root node corresponds to a brunch of the

disciminant region)
    - This may cause error because when the testing sample is near the boundary of the region, the nearest neighbor may not be found in the discriminated region.
- **Editing/Pruning/Condensing**: eliminate "redundant" ("useless") examples from the training set, e.g. example surrounded by training examples of the same class label

4.5 Distance Metric

4.5.1 Properties of Distance Metric

The NN/kNN rule depends on the use of distance metric to identify nearest neighbors.

Four properties of distance metric:

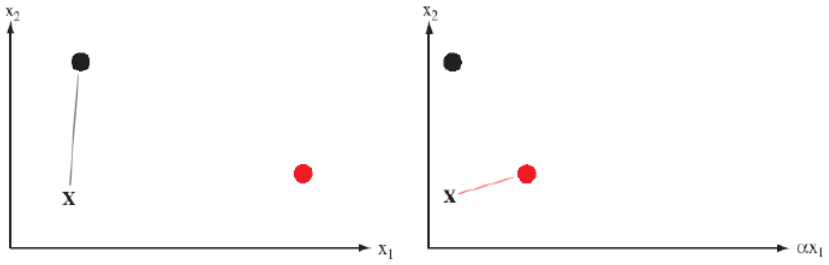
- **Non-negativity**: $D(a,b) \geq 0$
- **Reflexivity (自反性)**: $D(a,b) = 0 \Leftrightarrow a = b$
- **Symmetry**: $D(a,b) = D(b,a)$
- **Triangle Inequality**: $D(a,b) + D(b,c) \geq D(a,c)$

4.5.2 Euclidean Distance

Euclidean distance:

$$D(a,b) = \left(\sum_{j=1}^{d} (a_j - b_j)^2 \right)^{\frac{1}{2}}$$

Potential Issue:



Scaling the features ➔ change the distance relationship

- **Issue**: If the coordinate system is transformed, the new distance relationship calculated by Euclidean distance may be very different from the original one.
- **Possible solution**: normalize each feature into equal-sized intervals, e.g. [0,1]

Minkowski Distance Metric (a.k.a. L_k norm)

$$L_k(a,b) = \left(\sum_{j=1}^{d} |a_j - b_j|^k \right)^{\frac{1}{k}} \quad (k > 0)$$

- $k = 2$: Euclidean Distance
- $k = 1$: Manhattan Distance (city block distance)
- $k = \infty$: L_∞ distance

- $L_\infty(a, b) = \max_{1 \le j \le d} |a_j - b_j|$

4.5.3 Distance Metric Between Sets

Tanimoto distance:

$$D_{\text{Tanimoto}}(S_1, S_2) = \frac{n_1 + n_2 - 2n_{12}}{n_1 + n_2 - n_{12}}$$
$$(n_1 = |S_1|, n_2 = |S_2|, n_{12} = |S_1 \cap S_2|)$$

Hausdorff Distance:

$$D_H(S_1, S_2) = \max\left(\max_{s_1 \in S_1} \min_{s_2 \in S_2} D(s_1, s_2), \max_{s_2 \in S_2} \min_{s_1 \in S_1} D(s_2, s_1) \right)$$
$$(D(s_1, s_2) : \text{any distance metric between } s_1 \text{ and } s_2)$$

- $\max_{s_1 \in S_1} \min_{s_2 \in S_2} D(s_1, s_2)$: for every s_1 in S_1, find the maximum value of the minimum distance between s_1 and all s_2 in S_2

Eample: Hausdorff Distance between two sets of feature vectors

$$S_1 = \left\{(0.1, 0.2)^t, (0.3, 0.8)^t\right\} \quad S_2 = \left\{(0.5, 0.5)^t, (0.7, 0.3)^t\right\}$$

- $D_H(S_1, S_2) = \max\left(\max_{s_1 \in S_1} \min_{s_2 \in S_2} D(s_1, s_2), \max_{s_2 \in S_2} \min_{s_1 \in S_1} D(s_2, s_1) \right)$
  - $\max_{s_1 \in S_1} \min_{s_2 \in S_2} D(s_1, s_2)$
    1. for $s_1 = (0.1, 0.2)^t$
       - $d((0.1, 0.2)^t, (0.5, 0.5)^t) = 0.5$
       - $d((0.1, 0.2)^t, (0.7, 0.3)^t) = \sqrt{0.37}$
    2. for $s_1 = (0.3, 0.8)^t$
       - $d((0.3, 0.8)^t, (0.5, 0.5)^t) = \sqrt{0.13}$
       - $d((0.3, 0.8)^t, (0.7, 0.3)^t) = \sqrt{0.41}$
    3. $\max_{s_1 \in S_1} \min_{s_2 \in S_2} D(s_1, s_2) =$
       $\max\left(\min(0.5, \sqrt{0.37}), \min(\sqrt{0.13}, \sqrt{0.41})\right) =$
       $\max(0.5, \sqrt{0.13}) = 0.5$
  - $\max_{s_2 \in S_2} \min_{s_1 \in S_1} D(s_2, s_1)$
    1. for $s_2 = (0.5, 0.5)^t$
       - $d((0.5, 0.5)^t, (0.1, 0.2)^t) = 0.5$
       - $d((0.5, 0.5)^t, (0.3, 0.8)^t) = \sqrt{0.13}$
    2. for $s_2 = (0.7, 0.3)^t$
       - $d((0.7, 0.3)^t, (0.1, 0.2)^t) = \sqrt{0.37}$
       - $d((0.7, 0.3)^t, (0.3, 0.8)^t) = \sqrt{0.41}$
    3. $\max_{s_2 \in S_2} \min_{s_1 \in S_1} D(s_2, s_1) =$
       $\max\left(\min(0.5, \sqrt{0.13}), \min(\sqrt{0.37}, \sqrt{0.41})\right) =$
       $\max(\sqrt{0.13}, \sqrt{0.37}) = \sqrt{0.37}$
  - $D_H(S_1, S_2) = \max(\sqrt{0.5}, \sqrt{0.37}) = \sqrt{0.37} \approx 0.61$