Machine Learning (CS60050) Spring 2020-2021

Instructor : Prof. Aritra Hazra

Scribe by Sourav Garai(20CD92R03)

January 28, 2021

1 Introduction

In Machine learning we explore different ways of how to predict from given data. We have looked into different algorithms like decision trees, Naive Bayes Classifier and Bayesian Networks. These algorithms are part of supervised learning where the data and the labels are given beforehand.

2 Parametric learning:

Previously we have derived MAP from MLE and a prior where the prior is determined from μ and σ which are known parameters. Also in MLE we try to determine the argmax θ which resembles the data probability. These are learned parameters thus the algorithm is said to be based on parametric learning.

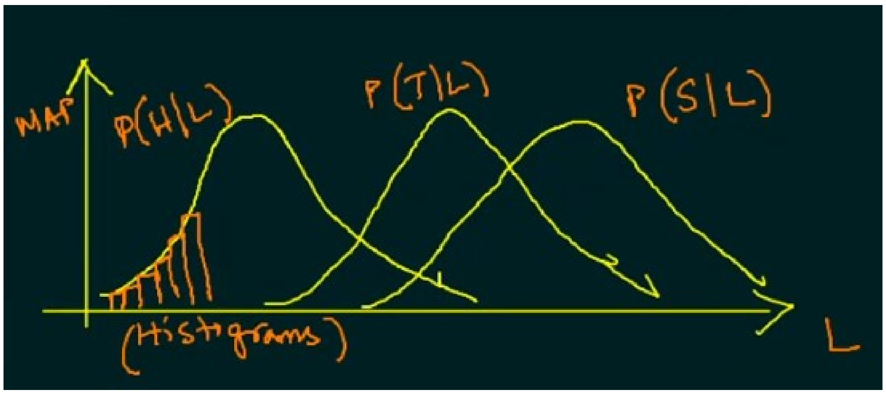


Figure 3.1: Histogram plot of fish length

3 Non-Parametric Learning

We will look into how we can predict directly from data instead of learning parameters. Let us consider a problem where we are given data of 3 types of fishes namely Hilsa, Tuna and Shark. The MAP values represented by P(H|L), P(T|L) and P(S|L) respectively.

3.1 Objective:

Now given a new fish of length l we need to find the type of the fish.

3.1.1 Approach:

If the length of the fish is $l = 2$ft we assume a length $\delta = 0.1$ and we get a window of $l \pm \delta$ that is 1.9 to 2.1. Then we see in the data for this window how many fishes are Hilsa, how many are Tuna and how many are Shark and compute the probability of each type of fish. The probabilities for Hilsa, Tuna and Shark respectively are $f_H = \frac{N_H}{N}, f_T = \frac{N_T}{N} and f_S = \frac{N_S}{N}$ where N is the total number of fishes. We take a vote of the largest probability and then predict this type as the type for the new fish.

This process is often known as the **Approximate MAP** as it is dependent on the relative count of the elements.

We can extend this to 2 dimensions say X and Y for length and width of the fish.

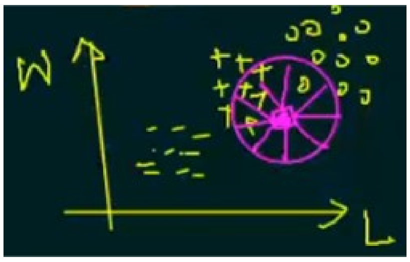


Figure 3.2: Length and width of fishes

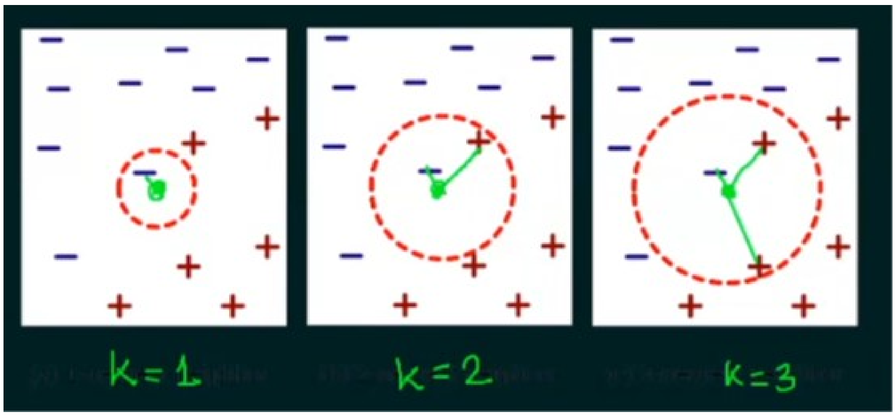


Figure 3.3: K Nearest Neighbour

Here we for an unknown fish we consider a circle and then find out in this circle how many fishes are Hilsa, how many are Tuna and how many are Shark and calculate the probability in the same way.

3.2 Nearest Neighbour

The principle is to create a boundary and see which of the neighbors are most frequent and then classify. We can consider the distances and then take the top K shortest distances. This is the intuition behind the **K-Nearest Neighbour** algorithm.

The distance between 2 points can be calculated in a number of ways. For instance let us consider the distance is equal to the euclidean distance. The value K is a user chosen parameter. The figure3.3 shows different values of K=1,2,3 We can see when K is 1 the new point is classified as -ve. When K=2

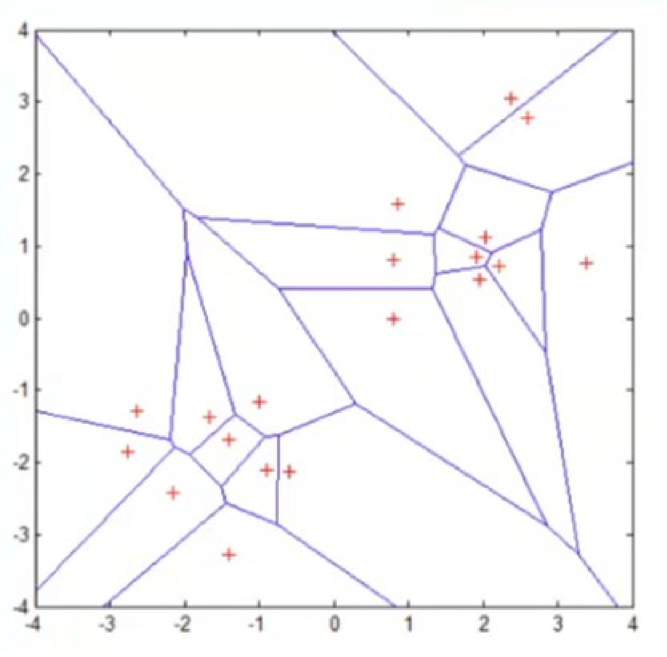


Figure 3.4: Caption

there is a tie and when K=3 the point is classified as +ve.

If we need to use K=1 there are efficient algorithms to do it. In computational geometry we have a Voronoi diagram were we can cluster the entire space(tessellation) such that given any point in the space we can directly get the nearest neighbour. So this is fairly straight forward as during test time we don't have to calculate the nearest distance for every point. This helps us save a lot of time during testing.

3.3 Mathematical formulation

Let f is a unknown labeling function to label the training examples.

$\hat{f}(q) = argmax_{v \in V} = \sum_{j=1}^{k} \delta(v, f(x_j))$

Where $f(x_i)$ is the label of training example x_i.

$\delta(a,b) = 1, a = b$ and $\delta(a,b) = 0, a \neq b$ otherwise.

3.4 Distance weighted K-Nearest Neighbour

Instead of just directly taking the top K nearest point contribution, we can multiply a weight to each point's contribution so that nearby points have more contribution than the far away points. The weight can be inverse of the distance of the new point q from the other training example points.

$\hat{f}(q) = argmax_{v \in V} \sum_{j=1}^{k} \frac{1}{d(q,x_{ij})^2} \delta(v, f(x_j)$
So further the point the lesser contribution it will have.

Generally if the weight of each point is W_j we have the formula
$\hat{f}(q) = argmax_{v \in V} \sum_{j=1}^{k} W_j \delta(v, f(x_j)$

3.4.1 Continuous values

For continuous values we take the weighted average of the k neighbours. For example if we want to calculate the temperature on a particular day example. January 28 given previous 100 years of temperature data for January 28.

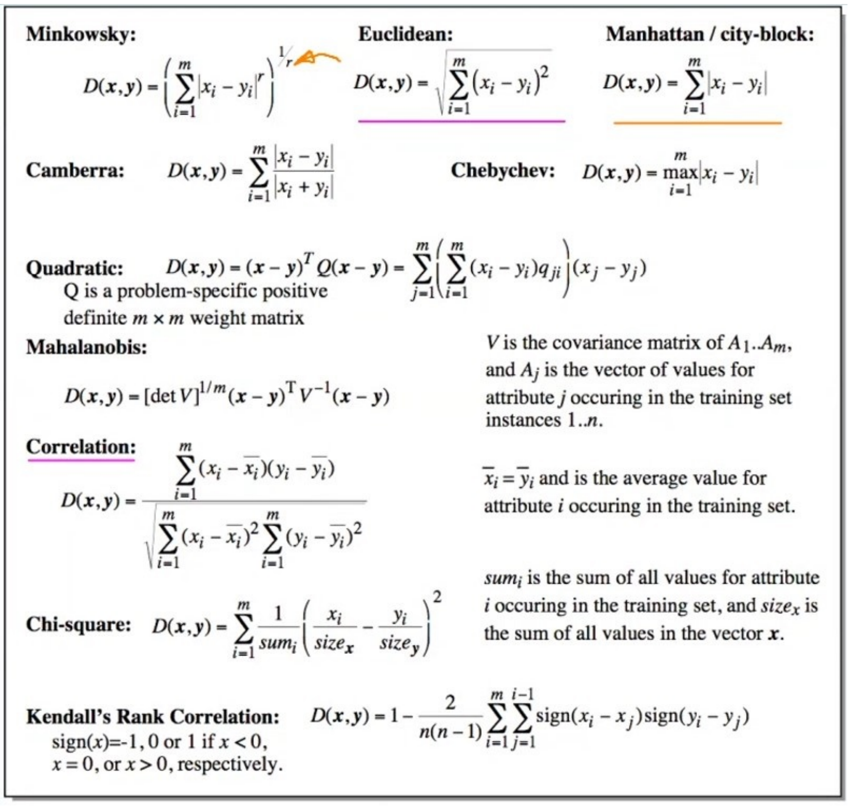
$\hat{f}(q) = \frac{\sum_{j=1}^{k} W_j f(x_{ij})}{\sum_{j=1}^{k} W_j}$

3.5 What is a good 'K'

Too Small: If the value of K is too small the prediction is based on few points and may be affected by locality error. So noise may detoriate the performance.

Too Large: If the value of K is too large the prediction will be influenced by the Prior(MAP). It will be too global in this case.

There is no fixed way to calculate the value of K but generally in practice we take $k = \sqrt{N}$ where N is the number of training examples. If there are lesser number of examples then $k = \frac{N}{10}$ is used. It is an open problem to find the value of K.



Minkowsky:

$$D(\boldsymbol{x},\boldsymbol{y}) = \left(\sum_{i=1}^{m} |x_i - y_i|^r\right)^{1/r}$$

Euclidean:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

Manhattan / city-block:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} |x_i - y_i|$$

Camberra:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{|x_i - y_i|}{|x_i + y_i|}$$

Chebychev:

$$D(\boldsymbol{x},\boldsymbol{y}) = \max_{i=1}^{m} |x_i - y_i|$$

Quadratic: Q is a problem-specific positive definite $m \times m$ weight matrix

$$D(\boldsymbol{x},\boldsymbol{y}) = (\boldsymbol{x}-\boldsymbol{y})^T Q(\boldsymbol{x}-\boldsymbol{y}) = \sum_{j=1}^{m}\left(\sum_{i=1}^{m}(x_i - y_i)q_{ji}\right)(x_j - y_j)$$

Mahalanobis:

$$D(\boldsymbol{x},\boldsymbol{y}) = [\det V]^{1/m}(\boldsymbol{x}-\boldsymbol{y})^{\mathrm{T}} V^{-1}(\boldsymbol{x}-\boldsymbol{y})$$

V is the covariance matrix of $A_1..A_m$, and A_j is the vector of values for attribute j occuring in the training set instances $1..n$.

Correlation:

$$D(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{m}(x_i - \overline{x_i})(y_i - \overline{y_i})}{\sqrt{\sum_{i=1}^{m}(x_i - \overline{x_i})^2 \sum_{i=1}^{m}(y_i - \overline{y_i})^2}}$$

$\overline{x_i} = \overline{y_i}$ and is the average value for attribute i occuring in the training set.

Chi-square:

$$D(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{m} \frac{1}{sum_i}\left(\frac{x_i}{size_{\boldsymbol{x}}} - \frac{y_i}{size_{\boldsymbol{y}}}\right)^2$$

sum_i is the sum of all values for attribute i occuring in the training set, and $size_{\boldsymbol{x}}$ is the sum of all values in the vector $\boldsymbol{x}$.

Kendall's Rank Correlation: sign(x)=-1, 0 or 1 if $x < 0$, $x = 0$, or $x > 0$, respectively.

$$D(\boldsymbol{x},\boldsymbol{y}) = 1 - \frac{2}{n(n-1)}\sum_{i=1}^{m}\sum_{j=1}^{i-1}\operatorname{sign}(x_i - x_j)\operatorname{sign}(y_i - y_j)$$

Figure 3.5: Distance Metrics

There are other ways of calculating the distance between 2 points namely Minkowsky, Manhattan, Quadratic, Correlation, Chi-square, etc.3.5

Which is a better distance metric depends on the data and the domain. It is always better to try out different metrics and figure out which works best for the data.

3.6 Issues with KNN algorithm

3.6.1 Curse of Dimensionality

Let us consider 2 pair of points (q,x) and (q,y) in multidimensional plane and measure the euclidean distance between them. Refer figure 3.6

We can see the euclidean distance of both the points are same and equals to $\sqrt{2}$. We can see that distance between x and q is much smaller than y

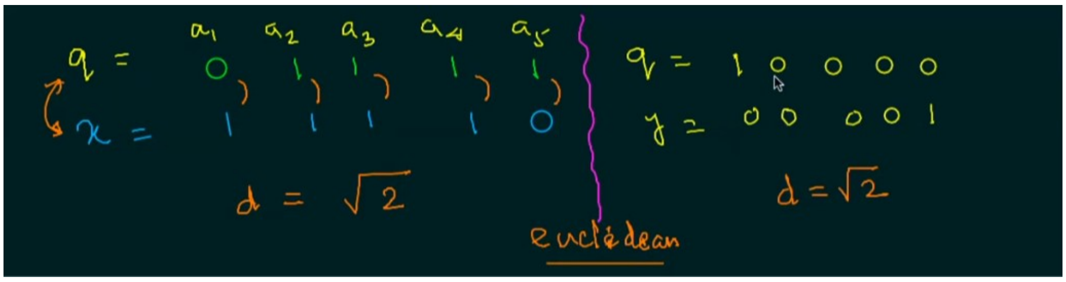


Figure 3.6: Euclidean Distance

and q. As a result both the points x and y are considered to be at equal distance from q which is clearly not correct. This problem arises in higher dimensional planes and known as the curse of dimensionality.

3.6.2 Scale of Attributes

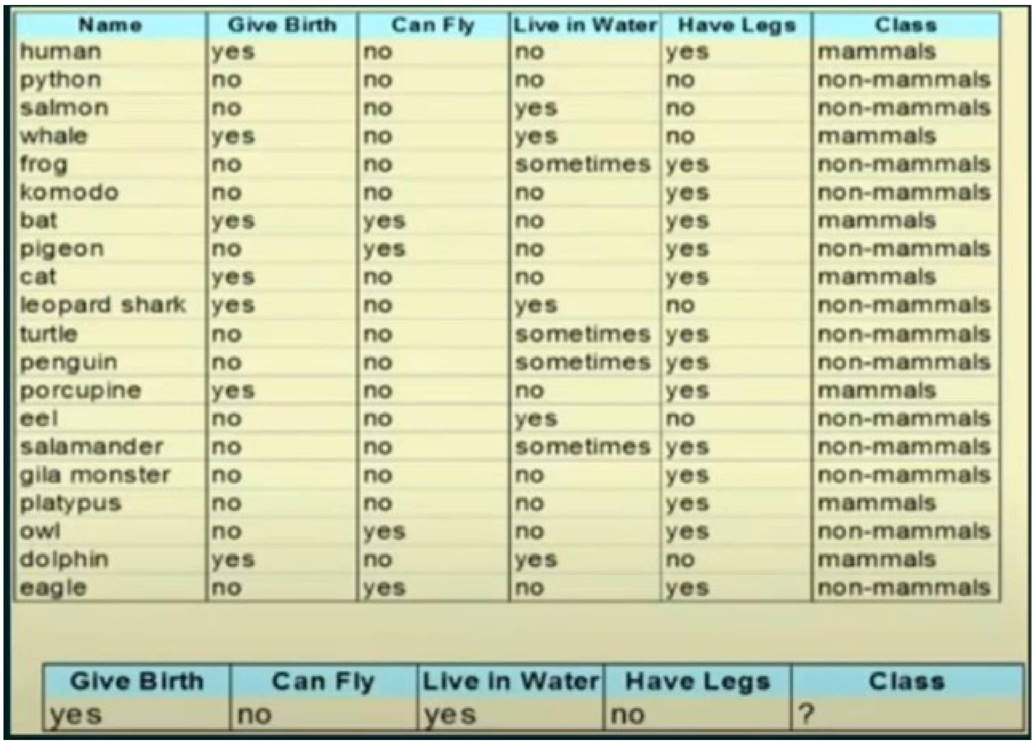
The attributes of the data may have different range. For example a medical data-set may have attributes age, blood-sugar, blood-pressure, etc. Here the value of age is between 0-100, blood sugar is between 50 to 200 and some other attribute may have data ranging from .01 to .09

In such a case we need to normalize the data such that all the values are between a fixed range say 0-1. The way we can do this is by calculating the mean and standard deviation of each attribute and then calculate the normalized value z where

$$z_{ij} = \frac{x_{ij} - \mu_i}{\sigma_i}$$
$$\mu_i = mean(X_i)$$
$$\sigma_i = StandardDeviation(X_i)$$

4 Example

From given data in figure 4.1 let us try to figure out the class of a unknown animal with the help of the features.



Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Figure 4.1: Example

Here number of training examples N=20 so we can take K=$\sqrt{20}$ or K=5 to prevent a tie

Now the testing data has attributes GiveBirth=Yes, CanFly=No, LiveIn-Water=Yes, HaveLegsa=No. So we look at examples with similar attribute values.

We find closest match with salmon, whale, leopard shark, eel and dolphin which sums to 3 non-mammals and 2 mammals.

Finally we predict the class of the animal to be non-mammal.

5 SUMMARY

- In non-parametric learning we do not need to learn any parameters.
- Given the raw data we find the top K points which are nearer to the target point that we need to predict. We give the label which has the highest count for the K points.
- The distance is weighted so that nearer points has a greater influence. There are different ways to calculate distance like Euclidean, Manhattan,etc.
- The value of K is user defined. Too small value will lead to localized errors, too large value will be biased to the prior. Practically $\sqrt{N}$ is used for large training examples(N) or $\frac{N}{10}$ for lesser number of training examples.
- There are 2 issues with the KNN algorithm
  - Curse Of Dimensionality
  - Computational Complexity