Assignment - 1

Foundations of Machine Learning

Tuhin Dutta

AI21MTECH02002

$\boxed{1}$ Given n points, we imagine two classes of data each of n/2 points overlapped to some extent in 2D space -

(a) Using all the available data, the training error varies ~~between~~ on the choice of 'K' in K-NN classifier.

K = 1, for each data point, x, in the training set, we want to find another point, x', that has the least shortest distance from x. Here the possible distance is always 0 (zero) which means the nearest neighbour is the data point itself, $x = x'$.

Overfitting

For this reason, the training error will be zero.

K = n, Here the training error become very high and in some specific cases be even 100% !

underfitting

This is because for uneven datasets where one class is having very few samples ~~but the other~~ ~~class is having~~ than the other class then by majority voting the new test sample of the other class would always be classified incorrectly. ~~The~~ ~~Generally, if K = n then model ends up predicting the~~ For this reason, the training error ~~may be~~ can reach 100%.

Model ends up predicting the majority level of the whole data-set.

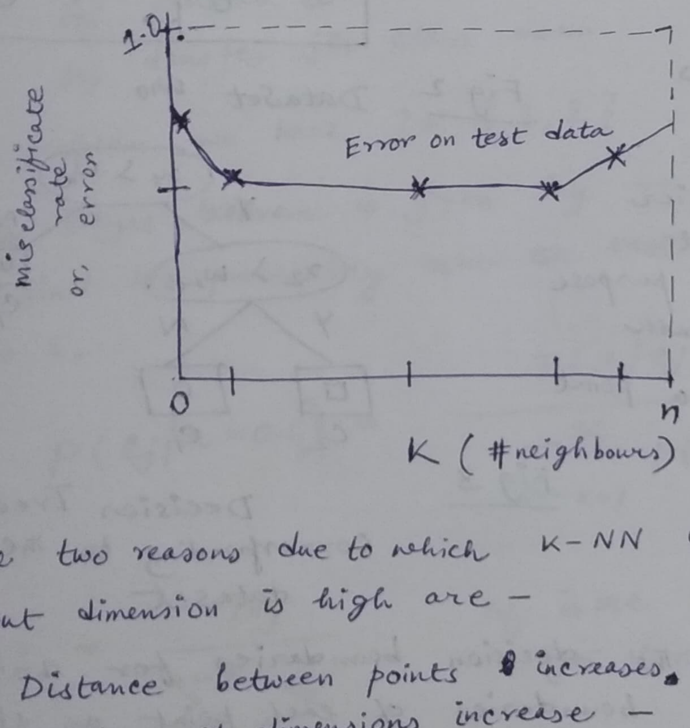
~~but~~ K > 1 and K < n, Generally K value is chosen empirically so that neither it overfits (K=1) nor it underfits (K=n).

(b) Generalization error is the expected value of the misclassification rate when averaged over future data. It can be approximated by computing the misclassification rate on a large independent test set not used during model training. The misclassification error can be defined as follows -

$$\text{error} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(f(x_i) \neq y_i \right)$$

where, $f(x)$ is the classifier.

$$\mathbb{I}(e) = \begin{cases} 1 & e == \text{TRUE} \\ 0 & e == \text{FALSE} \end{cases}$$ is the indicator function.



Here we plot the test error rate Vs K. It has a U-shaped curve. For complex model (small K) the method overfits. and for simple mode (big K) the method underfits.

(c) The two reasons due to which K-NN is undesirable when input dimension is high are -

i) Distance between points increases exponentially as the number of dimensions increase - we know K-NN classifier makes the assumption that similar points share similar labels. Unfortunately, in high-dimension spaces, points that are drawn from a probability distribution, never tend to be close to each other. For K-NN it requires a point to be close in every dimension. As gaps between data increases errors also increase and K-NN will not be able to predict/classify accurately based on measured distance.

As the number of dimensions increases the size of the data space increases and hence for K-NN to work correctly the amount of data needed to maintain density should also increase.

ii> Space complexity increases as dimensions increase -

K-NN classifier requires all data to be stored in memory so if we have n data points and each point of m dimension then the space complexity will be of the order $O(nm)$. This will increase as $(m >> n)$ and hence will be difficult to store and query.

(d)

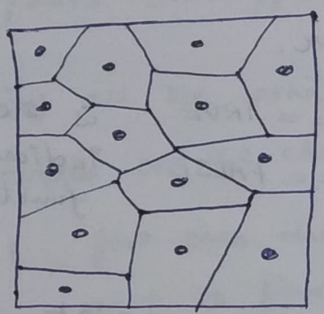


Fig 1

1-NN decision boundaries under euclidean distance

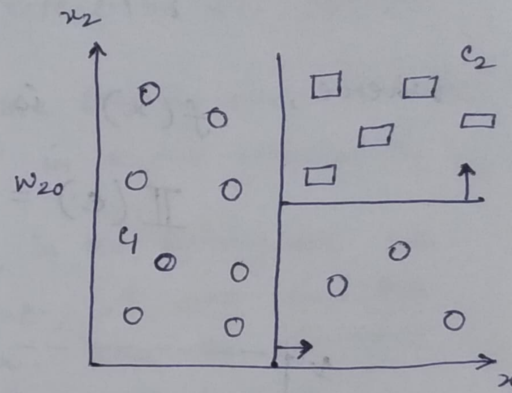
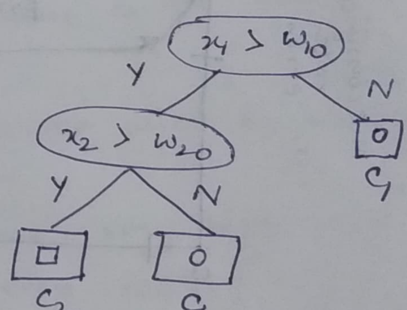


Fig 2 DataSet w_{10}



1-NN is a simple classifier which divides the input space for classification purpose into a convex region each of which corresponds to a point in the instance set.

Fig 3

Decision Tree corresponding to the above dataset

Now we can see, that the decision boundaries for 1-NN correspond to the cell boundaries of each point as shown in Fig 1 but they are not parallel to the co-ordinate axes.

Fig 2 is the dataset corresponding to the decision trees in Fig 3 and we can clearly observe that the ~~bou~~ boundaries are always parallel to co-ordinate axes based on the kind of questions asked at each node of the decision tree.

~~So~~ Now, since the univariate decision node splits along one-axis and successive splits are orthogonal to each other, it is not possible to mimic the same using 1-NN classifiers.

Hence it is not possible, to build a decision tree which classifies exactly according to the 1-NN scheme using euclidean distance measure.

---

[2] (a) The Gaussian likelihood is given by -

$$P(x|C_j) = \frac{1}{\sqrt{2\pi \sigma_j^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_j}{\sigma_j}\right)^2}$$

where C_j denotes j^{th} class

σ_j denotes j^{th} class variance

μ_j denotes j^{th} class mean $\quad \mu_j = \frac{1}{N_j}\sum_{i=1}^{N_j} x_i$

As per problem, we have $j = \{1, 2\}$

~~classifier~~ Bayes theorem is given by —

To find the probability that the test point $x = 0.6$ belongs to class 1,

$$P(C_j | x = 0.6) = \frac{P(x|C_j)\, P(C_j)}{\sum_{K=1}^{K} P(x|C_K)\, P(C_K)}$$

$$P(C_j) = \frac{N_j}{\sum_{K=1}^{K} N_K}$$

N_j is the # samples in the j^{th} class data set.

~~Given~~ let us find the maximum likelihood parameters,

$N_1 = 10$

$\mu_1 = 2.6/10$

$= 0.26$

$\sigma_1^2 = 0.0149$

$p_1 = P(C_1) = \frac{10}{14} = 0.714$

$N_2 = 4$

$\mu_2 = 3.45/4$

$= 0.8625$

$\sigma_2^2 = 0.0092$

$p_2 = P(C_2) = \frac{4}{14}$

$= 0.2857$

we have to find,

$$P(C_1 \mid x=0.6) = \frac{P(x=0.6 \mid C_1)\; P(C_1)}{P(x/C_1)\,P(C_1) + P(x/C_2)\,P(C_2)}$$

let us find the posterior, prior and evidence –

$$P(x=0.6/C_1) = \frac{1}{\sqrt{2\pi \sigma_1^2}}\, e^{-\frac{1}{2}\left(\frac{x-\mu_1}{\sigma_1}\right)^2}$$

$$= 0.0675466$$

$$P(x=0.6/C_2) = \frac{1}{\sqrt{2\pi \sigma_2^2}}\, e^{-\frac{1}{2}\left(\frac{x-\mu_2}{\sigma_2}\right)^2}$$

$$= 0.0983161$$

Therefore,

$$P(C_1 \mid x=0.6) = 0.631945$$

Also, we already have found,

$$p_1 = P(C_1) = 0.714$$

$$p_2 = P(C_2) = 0.2857$$

(Ans.)

(b) let $y = \begin{cases} 1 & \text{represent class politics} \\ 0 & \text{represent class sports} \end{cases}$

x = (goal, football, golf, defense, offence, wicket, office, strategy)

x_j = j^{th} word among the n words.

$n = 8$

m = total # samples
= 12

Assuming Bernoulli distribution, ~~estimate~~ the parameters, namely,

$P(y)$

$P(x_j = 1 | y = 0)$

$P(x_j = 1 | y = 1)$

Using joint likelihood, we can estimate these parameters,

$P(x_1, \dots, x_8 | y) = P(x_1 | y) \, P(x_2 | y, x_1) \, P(x_3 | y, x_1, x_2) \dots$

[Assume x_j are conditionally independent]

$P(x_1, \dots x_8 | y) = P(x_1 | y) \, P(x_2 | y) \, P(x_3 | y) \dots$

$$\boxed{P(x_1, \dots x_8 | y) = \prod_{j=1}^{n} P(x_j | y)} \quad \dots\dots (1)$$

~~or~~

~~$= \prod_{i=1}^{m} \prod_{j=1}^{n} P($~~

Therefore, the joint likelihood is calculated as —

$$\mathcal{L}(y, x_j | y) = \prod_{i=1}^{m}\prod_{j=1}^{n} P(x_j^i, y^i)$$

$$= \prod_{i=1}^{m}\prod_{j=1}^{n} P(x_j^i | y^i)\, P(y^i) \quad \text{.....} (2)$$

$$p(y) = \frac{\sum_i I\{y^i = 1\}}{m} = \frac{6}{12} = 0.5$$

$I(\cdot)$ is the indicator function

$$P(x_j = 1 | y = 1)$$

$$= \frac{\sum_i I\{y^i = 1, x_j^i = 1\}}{\sum_i I\{y^i = 1\}}$$

$$= \frac{1}{6}\begin{bmatrix} 2 & 1 & 1 & 5 & 5 & 1 & 4 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 2/6 & 1/6 & 1/6 & 5/6 & 5/6 & 1/6 & 4/6 & 5/6 \end{bmatrix}$$

$$P(x_j = 1 | y = 0)$$

$$= \frac{\sum_i I\{y^i = 0, x_j^i = 1\}}{\sum_i I\{y^i = 0\}}$$

$$= \frac{1}{6}\begin{bmatrix} 4 & 4 & 1 & 4 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 4/6 & 4/6 & 1/6 & 4/6 & 1/6 & 1/6 & 0 & 1/6 \end{bmatrix}$$

To predict we maximize as —

$$\arg\max_y P(y | \vec{x}) = \arg\max_y P(\vec{x} | y)\, P(y)$$

Given document, $\vec{x} = (1, 0, 0, 1, 1, 1, 1, 0)$

$$P(y=1 / \vec{x}) = P(\vec{x} \mid y=1)\, P(y=1)$$

$$= \prod_{j=1}^{n} P(x_j^i \mid y^i = 1)\, P(y^i = 1)$$

$$= \prod_{j=1}^{n} P(x_j = 1 / y=1)^{x_j^i} \cdot \left(1 - P(x_j = 1 / y = 0)\right)^{1 - x_j^i} , \cancel{P(y)}\ P(y^i = 1) \quad \ldots\ldots (3)$$

Also,

$$P(y=0 / \vec{x}) = P(\vec{x} / y=0)\, P(y=0)$$

$$= \prod_{j=1}^{n} P(x_j^i / y^i = 0)\, P(y^i = 0)$$

$$= \prod_{j=1}^{n} P(x_j = 1 \mid y = 0)^{x_j^i} \cdot \left(1 - P(x_j = 1 \mid y = 0)\right)^{1 - x_j^i} \cdot \left(1 - P(y^i)\right) \quad \ldots\ldots (4)$$

Taking log on both sides for (3), we get —

$$\log P(y=1 \mid \vec{x}) = \sum_{j=1}^{n=8} \left[x_j^i \log\left(P(x_j = 1 \mid y=1)\right) + (1 - x_j^i) \log\left(1 - P(x_j = 1 \mid y=1)\right)\right] + \log\left(P(y^i = 1)\right)$$

~~Compute using numpy python,~~

Here $i = 1$.

After computing, we get —

$$\log\left(P(y=1 / \vec{x})\right) = 0.128 + 0.16 = 0.288$$

$\log P(y=0 \mid \vec{x})$

$$= \sum_{j=1}^{n=8} \left[x_j^i \log(P(x_j=1 \mid y=0)) + (1-x_j^i) \log(1-P(x_j=1 \mid y=0)) \right] + \log(1-P(y))$$

After computing, we get —

$= -1.192 + 0.374$

$\Rightarrow \log(P(y=0 \mid \vec{x})) = -0.818$

since, $\log(P(y=1 \mid \vec{x})) > \log(P(y=0 \mid \vec{x}))$

we classify $\vec{x}$ as politics. (Ans.)