

K- Nearest Neighbor (K-NN Algorithm)

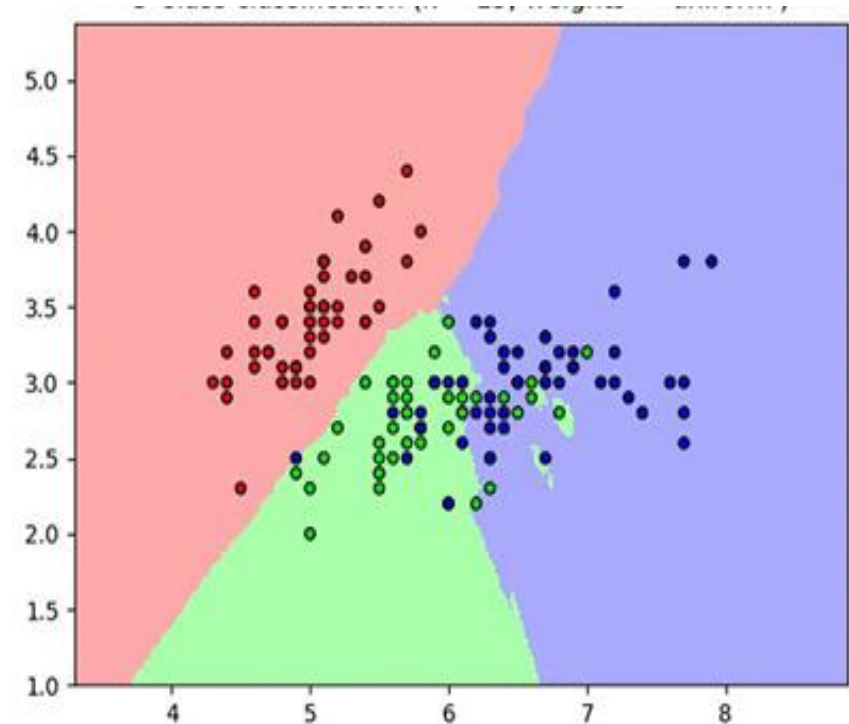
TIET, PATIALA

K-Nearest Neighbor - Introduction

- K-Nearest Neighbor (K-NN) is an instance-based (memory-based) supervised algorithm.
- **Instance-based learning** (sometimes called **memory-based learning**) is a family of learning algorithms that, instead of performing explicit generalization (in the training phase), compare new problem instances with instances seen in training, which have been stored in memory.
- It is called instance-based because it constructs hypotheses directly from the training instances themselves.
- The computation is postponed until a new instance is observed, these algorithms are sometimes referred to as "lazy."

K-Nearest Neighbor – Introduction (Contd...)

- K-NN is based on the fact that the similar objects lie in the close proximity to each other in the feature space.
- The figure illustrates that the objects of the same class lie quite close to each other and hence a new test case can be compared with k closest training examples in data set.



K-Nearest Neighbor – Introduction (Contd...)

- The ***k*-nearest neighbors algorithm (*k*-NN)**, is used for classification and regression.
- The output depends on whether k-NN is used for classification or regression:
 1. In ***k*-NN classification**, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.
 2. In ***k*-NN regression**, the output is the property value for the object. This value is the average of the values of k nearest neighbors.

Working of K- Nearest Neighbor Algorithm

- K-Nearest Neighbor algorithm works in the following steps:
 1. Determine parameter K = number of nearest neighbors.
 2. Calculate the distance between the query-instance and all the training samples.
 3. Sort the distance and determine nearest neighbors based on the K -th minimum distance.
 4. Gather the category Y of the nearest neighbors.
 5. Use simple majority of the category of nearest neighbors as the label of the query instance in case of classification and average of the values of k nearest neighbors in case of regression.

(For classification, K must be set as even number – so as to avoid tie during simple majority)

How to set value of K ?

- In KNN, finding the value of k is very crucial. A small value of k means that noise will have a higher influence on the result and a large value make it computationally expensive.
- If we choose our $K = 1$, then our algorithm behaves as over fitting and it gives a non - smooth decision surface.
- As K increases, our decision surface gets smoother and, if we choose K as very large, then our algorithm behaves as underfitting and it gives a smooth decision surface and everything becomes one class which is the majority class in our dataset.
- So, we should choose K wisely such that it should neither be overfitting nor be underfitting .

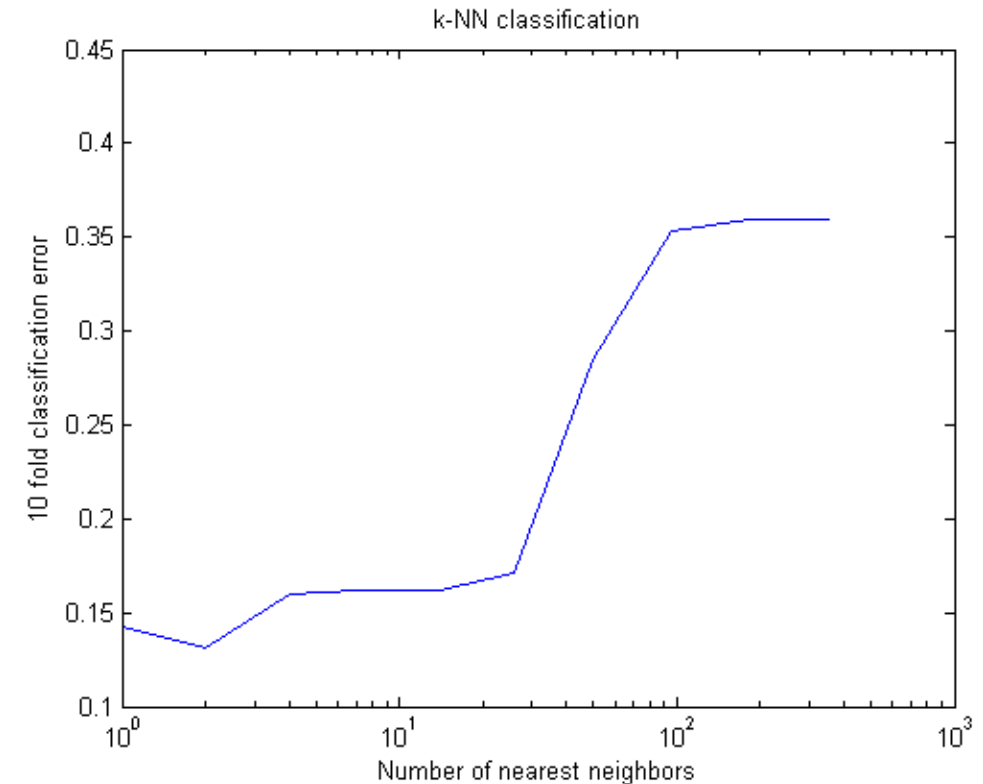
How to set value of K ?

Following approaches are used for setting value of K:

1. An odd number in case of classification.
2. Another simple approach to select k is set $k = \sqrt{n/2}$, where n = number of data points in training data.
3. The best approach is to plot the mean square error (or error rate) in labeling/prediction for each value of K. Chose the first value for which error rate is minimum.

This technique can be combined with k-fold cross validation (as shown in the figure)

We can tune the number of k in kNN and plot with respect to the cross-validation error



Distance Metrics for Continuous Variables

- Following association metrics are used for continuous variables:
- X_{ik} is the i^{th} instance of variable k , and p are the total number of features

Association Metric	Formula
1. Minkowski Distance	$d(X_i, X_j) = \left(\sum_{k=1}^p X_{ik} - X_{jk} ^m \right)^{1/m}$
2. Manhattan/ City-Block Distance	$d(X_i, X_j) = \sum_{k=1}^p X_{ik} - X_{jk} $
3. Euclidean Distance	$d(X_i, X_j) = \sqrt{\sum_{k=1}^p X_{ik} - X_{jk} ^2}$

Distance Metrics for Continuous Variables

Association Metric	Formula
4. Canberra Distance	$d(X_i, X_j) = \sum_{k=1}^p \frac{ X_{ik} - X_{jk} }{X_{ik} + X_{jk}}$
5. Czekanowski Coefficient	$d(X_i, X_j) = 1 - \frac{2 \sum_{k=1}^p \min(X_{ik}, X_{jk})}{\sum_{k=1}^p (X_{ik} + X_{jk})}$

Distance Metrics for Categorical Features

- If x and y are the feature vectors for two categorical variables, then following metrics are used for K-NN Classifier:

Association Metric	Formula
1. Cosine Distance	$d(x, y) = 1 - \frac{x \cdot y}{ x y }$
2. Tanimoto Coefficient	$d(x, y) = 1 - \frac{x \cdot y}{ x ^2 + y ^2 - x y }$
3. Jaccard Distance	$d(x, y) = 1 - \frac{ x \cap y }{ x \cup y }$
4. Sorensen-Dice Coefficient	$d(x, y) = 1 - \frac{2 x \cap y }{ x + y }$
5. Hamming Distance	$d(x, y) = \sum_{i=1}^k x_i - y_i $

Distance Metrics -Example

Consider the following Bag-of-Words Representation of three documents:

	i	think	therefore	am	can	you	don't	know	who
I think therefore I am	2	1	1	1	0	0	0	0	0
Can you think	0	1	0	0	1	1	0	0	0
I don't think therefore I don't know who I am	3	1	1	1	0	0	2	1	1

Cosine Distance:

$$d_{12} = 1 - \frac{1}{\sqrt{2^2 + 1 + 1 + 1}\sqrt{1 + 1 + 1}} = 1 - \frac{1}{\sqrt{7}\sqrt{3}} = 0.782$$

Tanimoto Distance:

$$d_{12} = 1 - \frac{1}{(2^2 + 1 + 1 + 1) + (1 + 1 + 1) - 1} = 1 - \frac{1}{7 + 3 - 1} = 0.889$$

$$\text{and } d_{13} = 1 - \frac{9}{7+18-9} = 0.438 \text{ and } d_{23} = 1 - \frac{1}{3+18-1} = 0.95$$

Distance Metrics –Example (Contd...)

Consider the following Bag-of-Words Representation of three documents:

	i	think	therefore	am	can	you	don't	know	who
I think therefore I am	2	1	1	1	0	0	0	0	0
Can you think	0	1	0	0	1	1	0	0	0
I don't think therefore I don't know who I am	3	1	1	1	0	0	2	1	1

Jaccard Distance:

$$d_{12} = 1 - \frac{1}{6} = 0.833$$

Note that Jaccard's metric ignores number of times each word occurs and essentially treats x_{iw} as binary indicator. $d_{13} = 1 - \frac{4}{7} = 0.429$ and $d_{23} = 1 - \frac{1}{9} = 0.889$.

Dice –Coefficient:

$$d_{12} = 1 - \frac{2+1}{4+3} = 0.714$$

K-NN Classifier- Numerical Example 1

- We have data from the questionnaires survey (to ask people opinion) and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here is four training samples:

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Y= Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

- Now the factory produces a new paper tissue that pass laboratory test with $X1 = 3$ and $X2 = 7$. Without another expensive survey, can we guess what the classification of this new tissue is?

Example 1- Solution

- 1. Determine parameter K = number of nearest neighbors**

Suppose use $K = 3$

- 2. Calculate the distance between the query-instance and all the training samples**

Coordinate of query instance is (3, 7), instead of calculating the distance we compute square distance which is faster to calculate (without square root)

Example 1- Solution (Contd...)

X1 = Acid Durability (seconds)	X2= Strength (kg/square meter)	Square Distance to query instance (3, 7)
7	7	$(7-3)^2+(7-7)^2=16$
7	4	$(7-3)^2+(4-7)^2=25$
3	4	$(3-3)^2+(4-7)^2=9$
1	4	$(1-3)^2+(4-7)^2=13$

Example 1- Solution (Contd...)

3. Sort the distance and determine nearest neighbors based on the K-th minimum distance

X1	X2	Square Distance to query instance (3, 7)	Rank minimum distance	Is it included in 3-Nearest neighbors?
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes
7	4	$(7-3)^2+(4-7)^2=25$	4	No
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes

Example 1- Solution (Contd...)

4. Gather the category Y of the nearest neighbors. Notice in the second row last column that the category of nearest neighbor (Y) is not included because the rank of this data is more than 3 (=K).

X1	X2	Square Distance to query instance (3, 7)	Rank minimum distance	Is it included in 3-Nearest neighbors?	Y= Category of nearest Neighbor
7	7	$(7-3)^2+(7-7)^2=16$	3	Yes	Bad
7	4	$(7-3)^2+(4-7)^2=25$	4	No	-
3	4	$(3-3)^2+(4-7)^2=9$	1	Yes	Good
1	4	$(1-3)^2+(4-7)^2=13$	2	Yes	Good

Use simple majority of the category of nearest neighbors as the prediction value of the query instance. We have 2 good and 1 bad, then we conclude that a new paper tissue that pass laboratory test with $X1 = 3$ and $X2 = 7$ is included in **good** category.

Advantages of K-NN Algorithm

1. **Prediction quality:**

- A kNN classifier is able to recover unstructured partitions of the space, as opposed to, say, a linear classifier that requires a linear separation between the classes.
- It can also adapt to different densities in the space, making it more robust than methods such as support vector machine (SVM) classification with radial basis function (RBF) kernel.

2. **Short cycles:**

- Another advantage of kNN is that there is little to no training involved.
- This means that iterating over different possible metrics / modifications of the input dataset is potentially faster when compared to a classifier that requires a heavy training procedure.

3. **Multi-Class Classification:**

- kNN can seamlessly handle a very large number of classes.
- For comparison, a linear model or a deep neural network with a cross-entropy loss must explicitly compute a score for each possible class, and choose the best one.

Advantages of K-NN Algorithm

4. Interpretability:

- The prediction of a kNN model is based on similarity to existing objects.
- As a result, the question “why was my example given class X?” is answered by “because similar items are labeled with X.”

Limitations of K-NN Algorithm

- **Costly inference:**
- The major disadvantage of kNN is its costly inference.
- To infer the label of an input query, we must find the data points closest to it.
- A naive solution would keep all data points in memory, and, given a query, compute the distance between it and all data points.
- For concrete quantities, if the training set contained n data points of d dimensions, this process requires $O(nd)$ arithmetic operations per query and $O(nd)$ memory.

Reducing Inference Cost

1. Subsampling:

- A very simple, yet often very effective way of reducing the inference cost is to subsample the data.
- For example, we might have an available dataset of, say, 10M data points, but we can do a good enough job with 100K data points.

2. Dimension reduction:

- For some distances, such as L2 and cosine distance, it's possible to reduce the dimension of the data points while maintaining the distance between a query point and any data point that is approximately the same.
- The quality of the approximation depends only on the dimension of the output.
- Small dimension means a crude approximation, yet it is often the case that we can obtain a good enough approximation for the distances while reducing the dimension.
- The main disadvantage of this method is that the output is dense, so for highly sparse data or data that had a low dimension to begin with, this might not be the best technique.

Reducing Inference Cost (Contd...)

3. Avoiding Regions Quickly:

- A common approach for disqualifying data points quickly is through clustering.
- If the center of a cluster is far away from the query we can disqualify the entire cluster without looking into all of its data points.
- For this technique, the data must be pre-processed to obtain $m \ll n$ centers, typically with k-means clustering.
- Then, when a query arrives we compute its distance to all of the centers, and disregard all points that belong to clusters with centers far away from the query.

