

Lecture 14

K Nearest Neighbors – Classification

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

KNN is a **non-parametric, lazy** learning algorithm. Its purpose is to use a database in which the data points are separated into several classes to predict the classification of a new sample point.

For a new case of dependent values (query point), we would like to estimate the outcome based on the *KNN* examples. *KNN* achieves this by finding *K* examples that are closest in distance to the query point, hence, the name *k-Nearest Neighbors*. For regression problems, *KNN* predictions are based on averaging the outcomes of the *K* nearest neighbors; for classification problems, a majority of voting is used.

When we say a technique is **non-parametric** it means that it does not make any assumptions on the underlying data distribution. In other words, the model structure is determined from the data. If you think about it, it's pretty useful, because in the "real world", most of the data does not obey the typical theoretical assumptions made (as in linear regression models, for example). Therefore, KNN could and probably should be one of the first choices for a classification study when there is little or no prior knowledge about the distribution data.

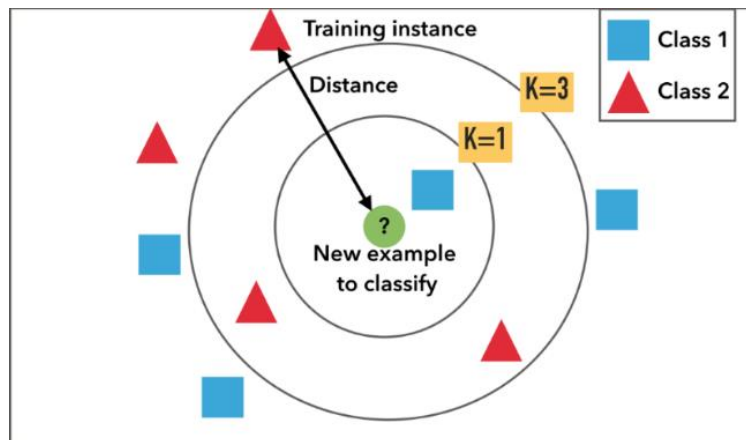
Eager Vs. Lazy Learners

Eager learners mean when given training points will construct a generalized model before performing prediction on given new points to classify. You can think of such learners as being ready, active and eager to classify unobserved data points.

Lazy Learning means there is no need for learning or training of the model and all of the data points used at the time of prediction. Lazy learners wait until the last minute before classifying any data point. Lazy learner stores merely the training dataset and waits until classification needs to perform. Only when it sees the test tuple does it.

KNN is a **lazy** algorithm (as opposed to an *eager* algorithm). Does that mean that KNN does nothing? What this means is that it does not use the training data points to do any *generalization*. In other words, there is *no explicit training phase* or it is very minimal. This also means that the training phase is pretty fast. Lack of generalization means that KNN keeps all the training data. To be more exact, all (or most) the training data is needed during the testing phase.

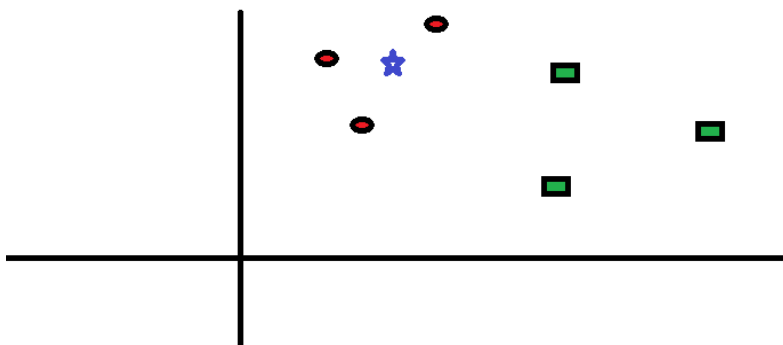
KNN Algorithm is based on **feature similarity**: How closely out-of-sample features resemble our training set determines how we classify a given data point:



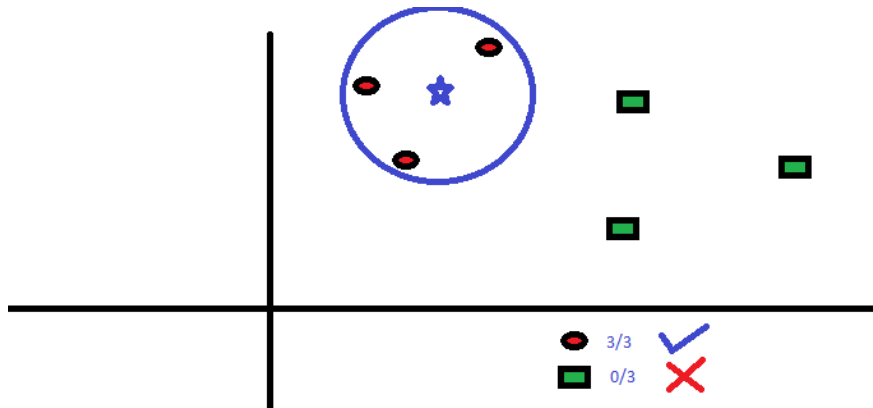
Example of k -NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of red triangles. If $k = 3$ (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example $k = 5$ it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle).

How does the KNN algorithm work?

Let's take a simple case to understand this algorithm. Following is a spread of red circles (RC) and green squares (GS) :



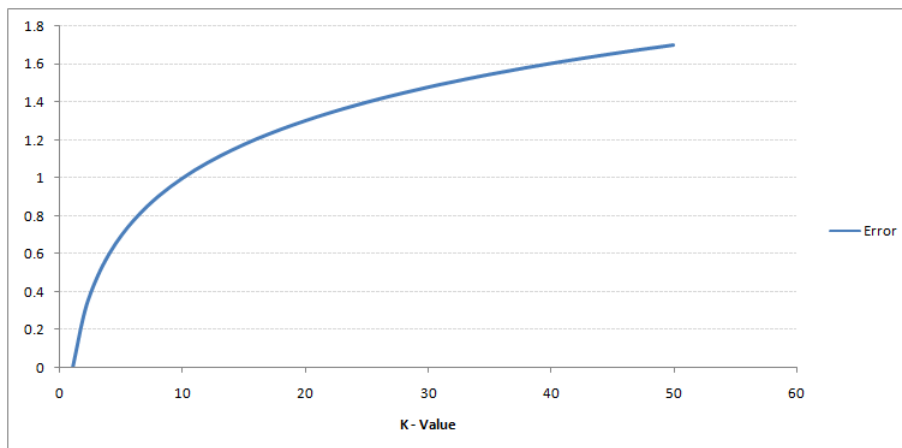
You intend to find out the class of the blue star (BS). BS can either be RC or GS and nothing else. The “K” in KNN algorithm is the nearest neighbors we wish to take vote from. Let’s say $K = 3$. Hence, we will now make a circle with BS as center just as big as to enclose only three datapoints on the plane. Refer to following diagram for more details:

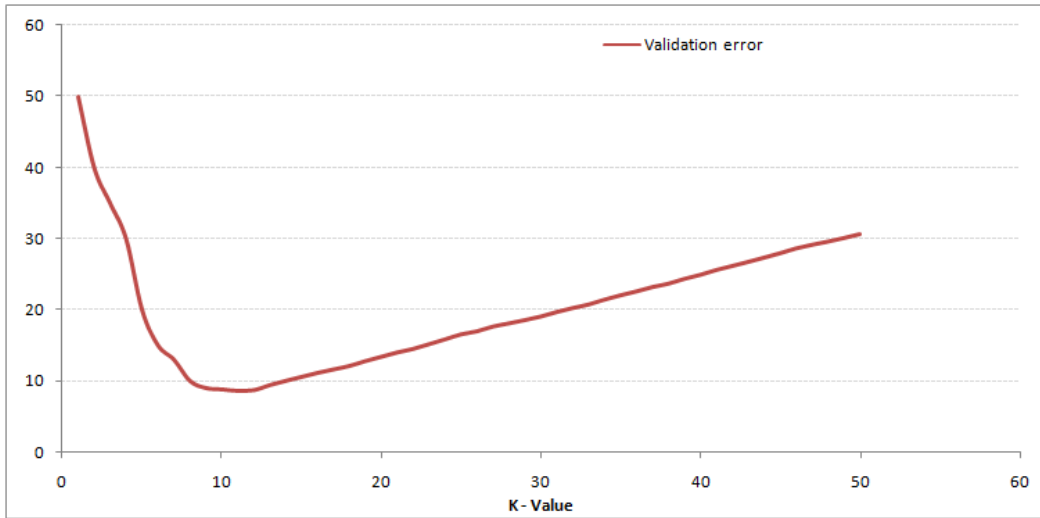


The three closest points to BS is all RC. Hence, with good confidence level we can say that the BS should belong to the class RC. Here, the choice became very obvious as all three votes from the closest neighbor went to RC. The choice of the parameter K is very crucial in this algorithm. Next we will understand what are the factors to be considered to conclude the best K .

How do we choose the factor K ?

First let us try to understand what exactly does K influence in the algorithm. If you watch carefully, you can see that the boundary becomes smoother with increasing value of K . With K increasing to infinity it finally becomes all blue or all red depending on the total majority. The training error rate and the validation error rate are two parameters we need to access on different K -value. Following is the curve for the training error rate with varying value of K :





This makes the story more clear. At $K=1$, we were overfitting the boundaries. Hence, error rate initially decreases and reaches a minima. After the minima point, it then increase with increasing K . To get the optimal value of K , you can segregate the training and validation from the initial dataset. Now plot the validation error curve to get the optimal value of K . This value of K should be used for all predictions.

Breaking it Down – Pseudo Code of KNN

KNN can be used for **classification** — the output is a class membership (predicts a class — a discrete value). An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. It can also be used for **regression** — output is the value for the object (predicts continuous values). This value is the average (or mean) of the values of its k nearest neighbors.

We can implement a KNN model by following the below steps:

The KNN Algorithm

1. Load the data
2. Initialize K to your chosen number of neighbors. (generally odd number)
3. For each example in the data
 3. Calculate the distance between the query example and the current example from the data.
 3. Add the distance and the index of the example to an ordered collection
 4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
 5. Pick the first K entries from the sorted collection
 6. Get the labels of the selected K entries
 7. If regression, return the mean of the K labels
 8. If classification, return the mode of the K labels

Algorithm

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If $K = 1$, then the case is simply assigned to the class of its nearest neighbor.

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

It should also be noted that all three distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

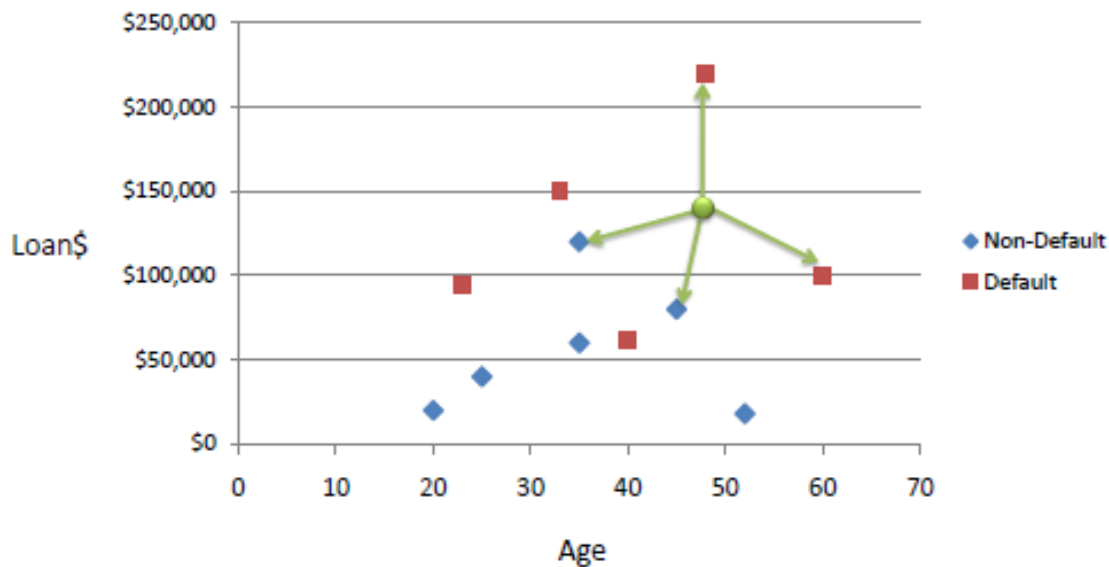
$$x \neq y \Rightarrow D = 1$$

X	Y	Distance
Male	Male	0
Male	Female	1

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee. Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. That produces much better results than 1NN.

Example:

Consider the following data concerning credit default. Age and Loan are two numerical variables (predictors) and Default is the target.



We can now use the training set to classify an unknown case (Age=48 and Loan=\$142,000) using Euclidean distance. If K=1 then the nearest neighbor is the last case in the training set with Default=Y.

Age	Loan	Default	Distance
25	\$40,000	N	102000
35	\$60,000	N	82000
45	\$80,000	N	62000
20	\$20,000	N	122000
35	\$120,000	N	22000
52	\$18,000	N	124000
23	\$95,000	Y	47000
40	\$62,000	Y	80000
60	\$100,000	Y	42000
48	\$220,000	Y	78000
33	\$150,000	Y	8000
48	\$142,000	?	

Euclidean Distance

$$D = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$D = \text{Sqrt}[(48-33)^2 + (142000-150000)^2] = 8000.01 \gg \text{Default}=Y$$

With K=3, there are two Default=Y and one Default=N out of three closest neighbors. The prediction for the unknown case is again Default=Y.

A few Applications and Examples of KNN

Credit ratings — collecting financial characteristics vs. comparing people with similar financial features to a database. By the very nature of a credit rating, people who have similar financial details would be given similar credit ratings. Therefore, they would like to be able to use this existing database to predict a new customer's credit rating, without having to perform all the calculations.

Should the bank give a loan to an individual? Would an individual default on his or her loan? Is that person closer in characteristics to people who defaulted or did not default on their loans?

In political science — classing a potential voter to a “will vote” or “will not vote”, or to “vote Democrat” or “vote Republican”.

More advance examples could include handwriting detection (like OCR), image recognition and even video recognition.

Some pros and cons of KNN

Pros:

No assumptions about data — useful, for example, for nonlinear data

Simple algorithm — to explain and understand/interpret

High accuracy (relatively) — it is pretty high but not competitive in comparison to better supervised learning models

Versatile — useful for classification or regression

Cons:

Computationally expensive — because the algorithm stores all of the training data

High memory requirement

Stores all (or almost all) of the training data

Prediction stage might be slow (with big N)

Sensitive to irrelevant features and the scale of the data