K Nearest Neighbors

K nearest neighbors 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 1970s as a non-parametric technique.

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 neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. If values are continuous, the Euclidean distance, Manhattan distance, and Minkowski function is used to classify the new case. Euclidean distance function is mostly used.

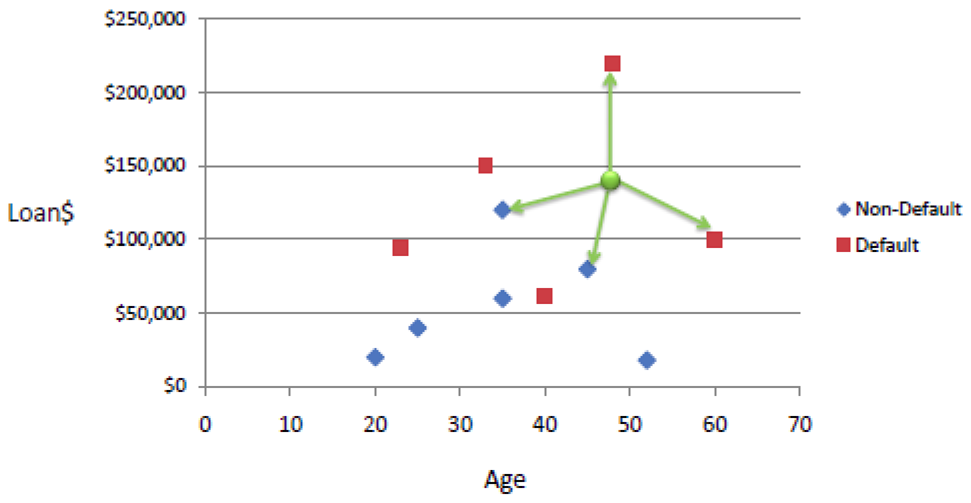
$$Euclidean\ distance = \sqrt{\sum_{i=1}^{k} (X_i - Y_i)^2}$$

But in the instance of categorical variables 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.

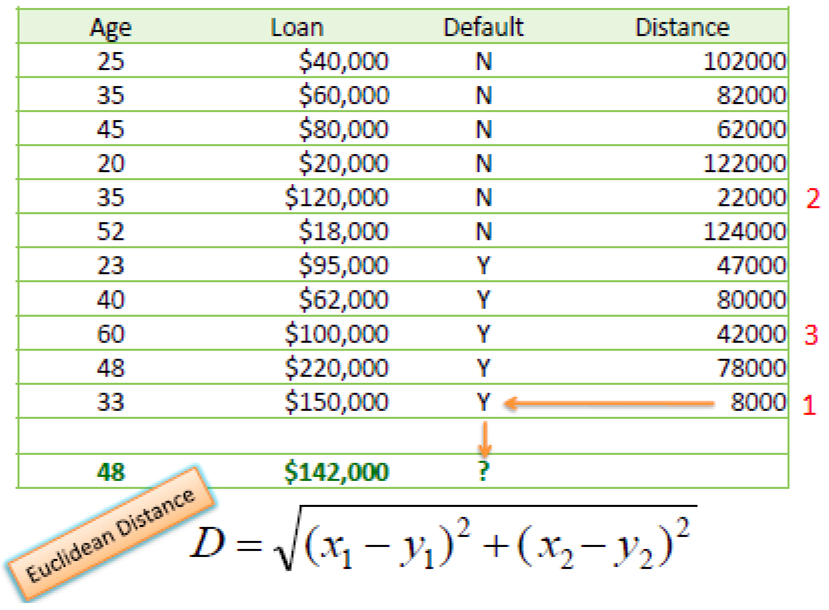
$$Hamming\ Distance,\ D_H = \sum_{i=1}^{k} \left| x_i - Y_i \right|$$
$$x = y \Rightarrow D = 0$$
$$x \neq y \Rightarrow D = 1$$

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. Tht 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	2
52	$18,000	N	124000	
23	$95,000	Y	47000	
40	$62,000	Y	80000	
60	$100,000	Y	42000	3
48	$220,000	Y	78000	
33	$150,000	Y	8000	1
48	**$142,000**	**?**		

Euclidean Distance

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

$$D = \sqrt{(48 - 33)^2 + (142000 - 150000)^2} = 8000.01 \Rightarrow Default = Y$$

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. equation for standardized distance,

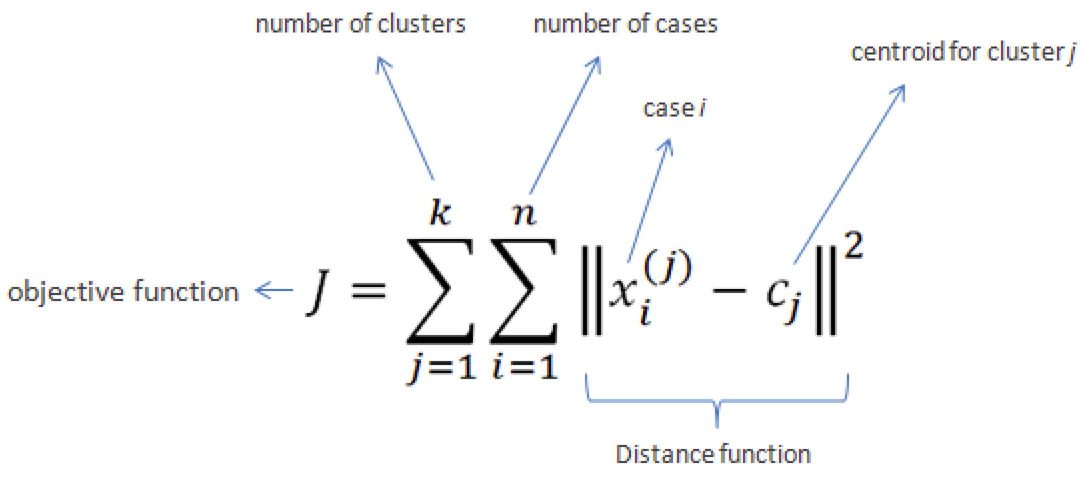
$$X_s = \frac{X - Min}{Max - Min}$$

Here goes the list of some papers in which k-NN was used :

1) The Distance-Weighted k-Nearest-Neighbor Rule (DOI : 10.1109/TSMC.1976.5408784)
2) A fuzzy K-nearest neighbor algorithm (DOI : 10.1109/TSMC.1985.6313426)
3) Distance Metric Learning for Large Margin Nearest Neighbor Classification (http://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification.pdf)

K-Means Clustering

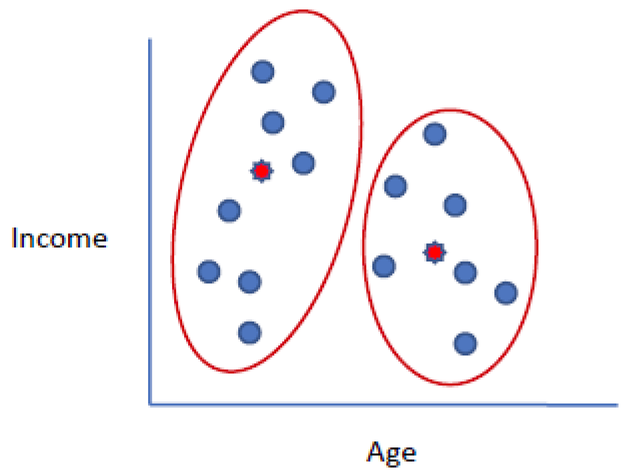
K-Means clustering intends to partition *n* objects into *k* clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly *k* different clusters of greatest possible distinction. The best number of clusters *k* leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:



$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \underbrace{\left\| x_i^{(j)} - c_j \right\|^2}_{\text{Distance function}}$$

Algorithm

1. Clusters the data into *k* groups where *k* is predefined.
2. Select *k* points at random as cluster centers.
3. Assign objects to their closest cluster center according to the *Euclidean distance* function.
4. Calculate the centroid or mean of all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.



K-Means is relatively an efficient method. However, we need to specify the number of clusters, in advance and the final results are sensitive to initialization and often terminates at a local optimum. Unfortunately there is no global theoretical method to find the optimal number of clusters. A practical approach is to compare the outcomes of multiple runs with different k and choose the best one based on a predefined criterion. In general, a large k probably decreases the error but increases the risk of overfitting.

Example:

Suppose we want to group the visitors to a website using just their age (one-dimensional space) as follows:

$$n = 16$$

$$15, 15, 16, 19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65$$

Initial clusters (random centroid or average):

$$k = 2,\ C_1 = 16,\ C_2 = 22$$

$$Distance1 = \left|X_i - C_1\right|$$

$$Distance2 = \left|X_i - C_2\right|$$

Iteration 1: $C_1 = 15.33,\ C_2 = 36.25$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	**15.33**
15	16	22	1	7	1	
16	16	22	0	6	1	
19	16	22	9	3	2	**36.25**
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

Iteration 2: $C_1 = 18.56, \ C_2 = 45.90$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	**18.56**
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	
19	15.33	36.25	3.67	17.25	1	
19	15.33	36.25	3.67	17.25	1	
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	**45.9**
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

Iteration 3: $C_1 = 19.50, \ C_2 = 47.89$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	**19.50**
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	
20	18.56	45.9	1.44	25.9	1	
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	**47.89**
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	
43	18.56	45.9	24.44	2.9	2	
44	18.56	45.9	25.44	1.9	2	
60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

Iteration 4: $C_1 = 19.50, \; C_2 = 47.89$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	**19.50**
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	
19	19.5	47.89	0.50	28.89	1	
19	19.5	47.89	0.50	28.89	1	
20	19.5	47.89	0.50	27.89	1	
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	**47.89**
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

No change between iterations 3 and 4 has been noted. By using clustering, 2 groups have been identified 15-28 and 35-65. The initial choice of centroids can affect the output clusters, so the algorithm is often run multiple times with different starting conditions in order to get a fair view of what the clusters should be.

List of papers using K-Means algorithm:

1) Constrained K-means Clustering with Background Knowledge
2) The global *k*-means clustering algorithm (doi : https://doi.org/10.1016/S0031-3203(02)00060-2)
3) Web-scale k-means clustering (doi : https://doi.org/10.1145/1772690.1772862)

Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used. Linear Regression is used for finding linear relationship between target and one or more predictors. There are two types of linear regression - Simple and Multiple.

Simple Linear Regression

Simple linear regression is useful for finding relationship between two continuous variables. One is predictor or independent variable and the other one is response or dependent variable. It looks for statistical relationship but not deterministic relationship. Relationship between two variables is said to be deterministic if one variable can be accurately expressed by the other. For example, using temperature in degree Celsius it is possible to accurately predict Fahrenheit. Statistical relationship is not accurate in determining the relationship between two variables. For example, relationship between height and weight.

The core idea is to obtain a line that best fits the data. The best fit line is the one for which total prediction error(all data points) is as small as possible. Error is the distance between the point and the regression line.

Real-time example :

We have a dataset which contains information about the relationship between ‘number of hours studied’ and ‘marks obtained’. Many students have been observed and their hours of study and obtained grade are recorded. This will be our training data. Goal is to design a model that can predict marks if given the number of hours studied. Using the training data, a regression line is obtained which will give minimum error. This linear equation is then used for any new data. That is, if we give number of hours studied by a student as an input, our model should predict their mark with minimum error.

$$Y(pred) = b_0 + b_1 * x$$

The values b_0 and b_1 must be chosen so that they minimize the error. If sum of squared error is taken as a metric to evaluate the model, then the goal is to obtain a line that best reduces the error.

$$Error = \sum_{i=1}^{n} (actualOutput - predictedOutput)^2$$

If we don't square the error, then positive and negative point will cancel out each other.

For model with one predictor,

$$b_0 = \bar{y} - b_1\bar{x}$$

Figure 3: Intercept Calculation

$$b_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Exploring 'b1'

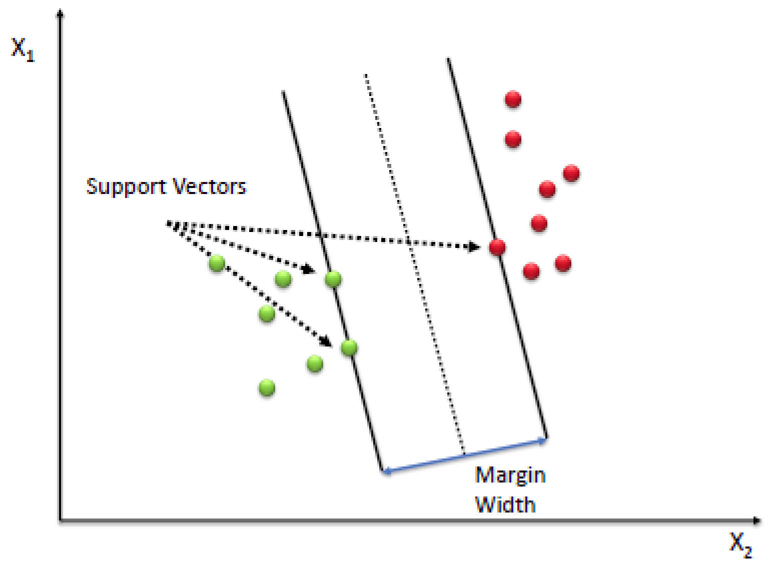
- If b1 > 0, then x(predictor) and y(target) have a positive relationship. That is increase in x will increase y.
- If b1 < 0, then x(predictor) and y(target) have a negative relationship. That is increase in x will decrease y.

Exploring ‘b0’

- If the model does not include x=0, then the prediction will become meaningless with only b0. For example, we have a dataset that relates height(x) and weight(y). Taking x=0(that is height as 0), will make equation have only b0 value which is completely meaningless as in real-time height and weight can never be zero. This resulted due to considering the model values beyond its scope.
- If the model includes value 0, then ‘b0’ will be the average of all predicted values when x=0. But, setting zero for all the predictor variables is often impossible.
- The value of b0 guarantee that residual have mean zero. If there is no ‘b0’ term, then regression will be forced to pass over the origin. Both the regression co-efficient and prediction will be biased.

Support Vector Machine - Classification (SVM)

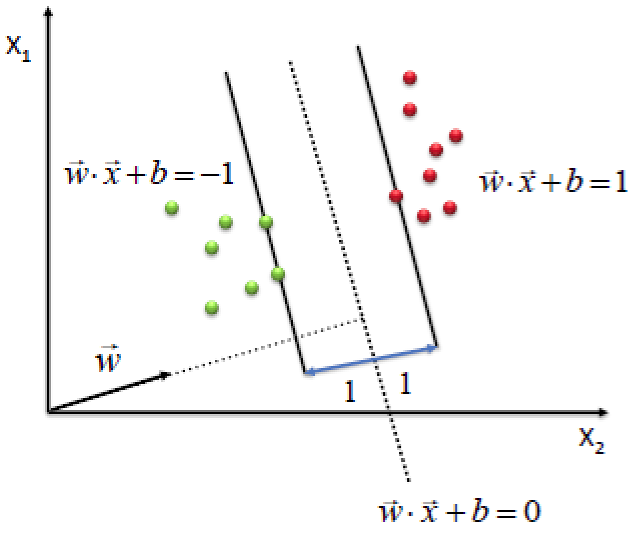
A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors.



Algorithm

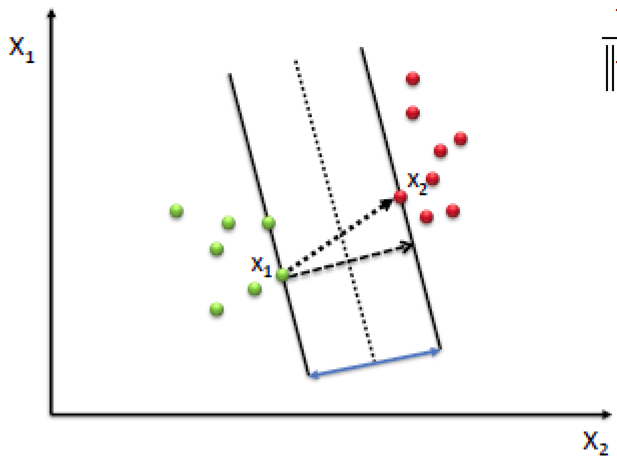
1. Define an optimal hyperplane: maximize margin
2. Extend the above definition for non-linearly separable problems: have a penalty term for misclassifications.
3. Map data to high dimensional space where it is easier to classify with linear decision surfaces: reformulate problem so that data is mapped implicitly to this space.

To define an optimal hyperplane we need to maximize the width of the margin (*w*).


X_1
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X_2
$\vec{w}\cdot\vec{x}+b=0$

$$\boxed{\max \frac{2}{\|w\|}}$$

$$\begin{aligned}&s.t.\\&(w\cdot x+b)\geq 1, \forall x \text{ of class } 1\\&(w\cdot x+b)\leq -1, \forall x \text{ of class } 2\end{aligned}$$



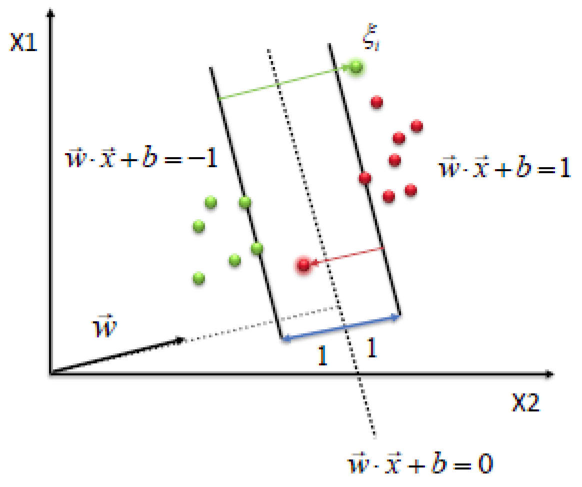
$$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$$

$$\begin{aligned}&w\cdot x_2+b=1\\&w\cdot x_1+b=-1\\&w\cdot x_2+b-w\cdot x_1-b=1-(-1)\\&w\cdot x_2-w\cdot x_1=2\\&\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}\end{aligned}$$

We find *w* and *b* by solving the following objective function using Quadratic Programming.

$$\min \frac{1}{2}\|w\|^2$$
$$s.t.\ y_i(w\cdot x_i+b)\geq 1,\ \forall x_i$$

The beauty of SVM is that if the data is linearly separable, there is a unique global minimum value. An ideal SVM analysis should produce a hyperplane that completely separates the vectors (cases) into two non-overlapping classes. However, perfect separation may not be possible, or it may result in a model with so many cases that the model does not classify correctly. In this situation, SVM finds the hyperplane that maximizes the margin and minimizes the misclassifications.

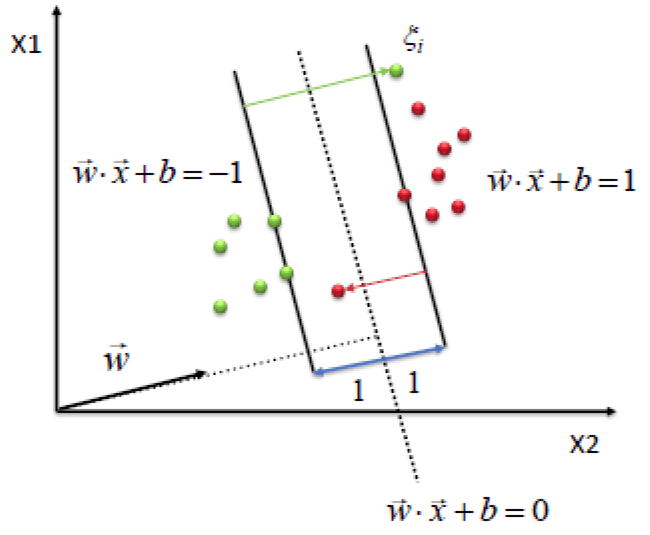

X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$

slack variable:

ξ_i

Allow some instances to fall off the margin, but penalize them

The algorithm tries to maintain the slack variable to zero while maximizing margin. However, it does not minimize the number of misclassifications (NP-complete problem) but the sum of distances from the margin hyperplanes.


X1
ξ_i
$\vec{w}\cdot\vec{x}+b=-1$
$\vec{w}\cdot\vec{x}+b=1$
$\vec{w}$
1
1
X2
$\vec{w}\cdot\vec{x}+b=0$

Constraint becomes :

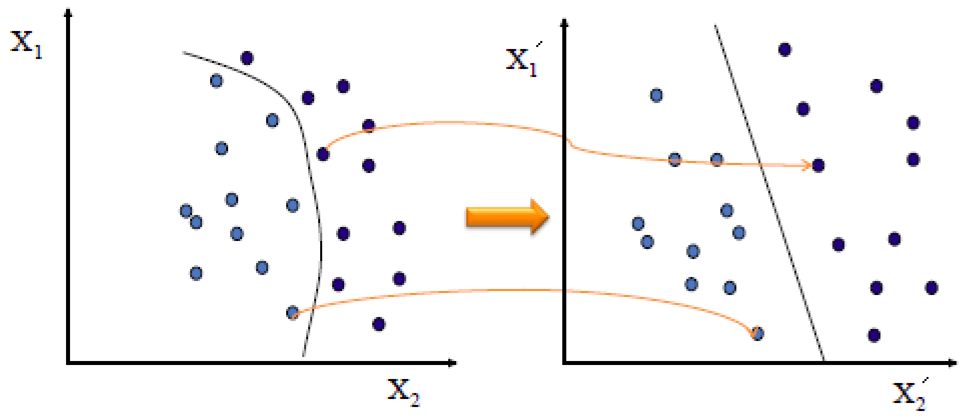
$$y_i(w\cdot x_i+b)\geq 1-\xi_i,\ \forall x_i$$
$$\xi_i\geq 0$$

Objective function
penalizes for misclassified instances and those within the margin

$$\min\frac{1}{2}\|w\|^2+C\sum_i\xi_i$$

C trades-off margin width and misclassifications

The simplest way to separate two groups of data is with a straight line (1 dimension), flat plane (2 dimensions) or an N-dimensional hyperplane. However, there are situations where a nonlinear region can separate the groups more efficiently. SVM handles this by using a kernel function (nonlinear) to map the data into a different space where a hyperplane (linear) cannot be used to do the separation. It means a non-linear function is learned by a linear learning machine in a high-dimensional feature space while the capacity of the system is controlled by a parameter that does not depend on the dimensionality of the space. This is called *kernel trick* which means the kernel function transform the data into a higher dimensional feature space to make it possible to perform the linear separation.



Linear SVM $x_i \cdot x_j$

Non-linear SVM $\phi(x_i) \cdot \phi(x_j)$

Kernel function $k(x_i \cdot x_j)$

Map data into new space, then take the inner product of the new vectors. The image of the inner product of the data is the inner product of the images of the data. Two kernel functions are shown below.

Polynomial

$$k(\mathrm{x}_i, \mathrm{x}_j) = (\mathrm{x}_i . \mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i, \mathrm{x}_j) = \exp\left(- \frac{\left\| \mathrm{x}_i - \mathrm{x}_j \right\|^2}{2\sigma^2} \right)$$

List of paper using SVM:

1) Least Squares Support Vector Machine Classifiers(DOI : 10.1023/A:1018628609742)
2) Fuzzy Support Vector Machines
3) Semi-Supervised Support Vector Machines (Source : http://papers.nips.cc/paper/1582-semi-supervised-support-vector-machines.pdf)