Lab #5: SVM (Support Vector Machine)

This lab is to deal with SVM to classification tasks and compare its performance with other competitive algorithms. In general, SVM is one of the most popular and widely used supervised machine learning algorithms.

The core idea of SVM is to find a **maximum marginal hyperplane** that best divides the dataset into classes.

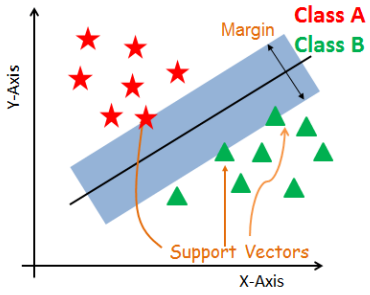

Fig.1. SVM idea

Support Vectors: Support vectors are the data points, which are closest to the hyperplane.

Hyperplane: A hyperplane is a decision plane which separates between a set of objects having different class memberships.

Margin: A margin is a gap between the two lines on the closest class points. This is calculated as the perpendicular distance from the line to support vectors or closest points.

- If the margin is larger in between the classes, then it is considered a good margin, a smaller margin is a bad margin.

Dealing with non-linear and inseparable planes: Some problems can't be solved using linear hyperplane, as shown in the figure below (left-hand side).

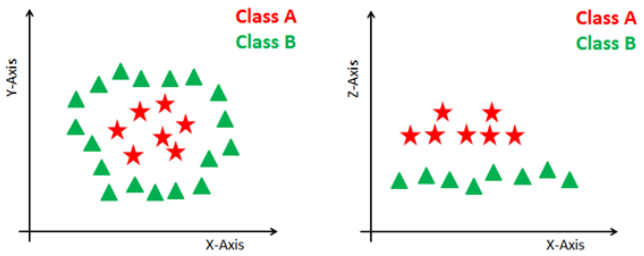

Fig.2. Kernel trick

In such situation, SVM uses a kernel trick to transform the input space to a higher dimensional space as shown on the right. The data points are plotted on the x-axis and z-axis

(Z is the squared sum of both x and y: $Z = x^2 + y^2$). Then it can be easily segregate these points using linear separation.

SVM Kernels

SVM uses a technique called the kernel trick which transfroms data points from a low-dimensional input space and to a higher dimensional space. In other words, it can be said that it converts nonseparable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem.

- **Linear Kernel**: A linear kernel can be used as normal dot product any two given observations.

$$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = (\mathrm{x}^{(i)^T} * \mathrm{x}^{(j)})$$

- **Polynomial Kernel**: A polynomial kernel is a more generalized form of the linear kernel. The polynomial kernel can distinguish curved or nonlinear input space.

$$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = \left(1 + \mathrm{x}^{(i)^T} * \mathrm{x}^{(j)}\right)^d$$

Where d is the degree of the polynomial. $d = 1$ is similar to the linear transformation. The degree needs to be manually specified in the learning algorithm.

- **Radial Basis Function Kernel**: The Radial basis function kernel is a popular kernel function commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space.

$$K(\mathrm{x}^{(i)}, \mathrm{x}^{(j)}) = e^{-\gamma \|\mathrm{x}^{(i)} - \mathrm{x}^{(j)}\|^2}$$

Here γ is a parameter ($\gamma \in [0,1]$). A higher value of gamma will perfectly fit the training dataset, which causes over-fitting. $\gamma = 0.1$ is considered to be a good default value. The value of gamma needs to be manually specified in the learning algorithm.

==========

SVM (https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html)

Syntax:

class sklearn.svm.SVC(*, *C=1.0*, *kernel='rbf'*, *degree=3*, *gamma='scale'*, *coef0=0.0*, *shrinking=True*, *probability=False*, *tol=0.001*, *cache_size=200*, *class_weight=None*, *verbose=False*, *max_iter=- 1*, *decision_function_shape='ovr'*, *break_ties=False*, *random_state=None*)

where,

C: float, optional (default=1.0)

Penalty parameter C of the error term.

kernel: string, optional (default='**rbf**')

Specifies the kernel type to be used in the algorithm. It must be one of '**linear**', '**poly**', '**rbf**', '**sigmoid**', '**precomputed**'. If none is given, '**rbf**' will be used.

degree: int, optional (default=3)

Degree of kernel function. It is significant only in '**poly**' and '**sigmoid**'.

gamma: float, optional (default=0.0)

Kernel coefficient for '**rbf**' and '**poly**'. If gamma is 0.0 then 1/n_features will be used instead.

coef0: float, optional (default=0.0)

Independent term in kernel function. It is only significant in '**poly**' and '**sigmoid**'.

probability: boolean, optional (default=False):

Whether to enable probability estimates. This must be enabled prior to calling predict_proba.

shrinking: boolean, optional (default=True):

Whether to use the shrinking heuristic. Shrinking Heuristic is to **accelerate the optimization process** by reducing the number of data points that need to be examined during each iteration. Shrinking Heuristic works by identifying data points that are likely to be support vectors (data points close to the decision boundary) or contribute significantly to the margin (the region separating different classes). Instead of evaluating all data points in each iteration, the algorithm focuses on these potentially important data points.

tol: float, optional (default=1e-3) :

Tolerance for stopping criterion.

cache_size: float, optional:

Specify the size of the kernel cache (in MB)

class_weight: {dict, 'auto'}, optional

Set the parameter C of class i to class_weight[i]*C for SVC. If not given, all classes are supposed to have weight one. The 'auto' mode uses the values of y to automatically adjust weights inversely proportional to class frequencies.

scale_C: bool, default: True

Scale C with number of samples. It makes the setting of C independent of the number of samples. To match libsvm commandline one should use scale_C=False.

Usage:

```
#Import svm model
from sklearn import svm
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = clf.predict(X_test)
```