GeeksforGeeks

What are Support Vector Machines?

Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line.

In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems.

Support Vector Machine for Multi-CLass Problems

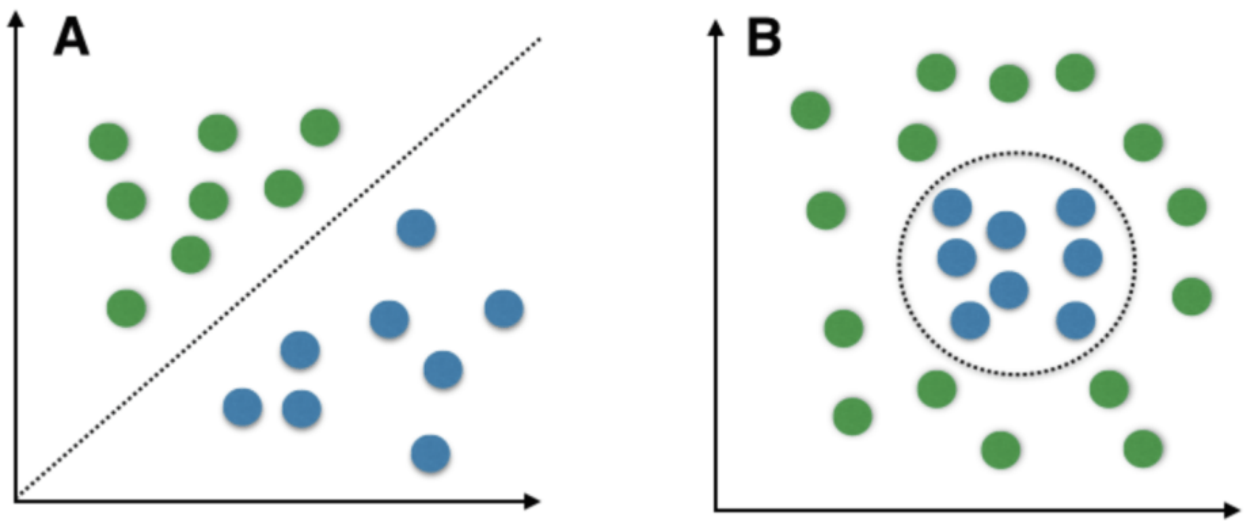
To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be :

- The data point belongs to that class OR
- The data point does not belong to that class.

For example, in a class of fruits, to perform multi-class classification, we can create a binary classifier for each fruit. For say, the 'mango' class, there will be a binary classifier to predict if it IS a mango OR it is NOT a mango. The classifier with the highest score is chosen as the output of the SVM.

SVM for complex (Non Linearly Separable)

SVM works very well without any modifications for linearly separable data. **Linearly Separable Data** is any data that can be plotted in a graph and can be separated into classes using a straight line.



A: Linearly Separable Data B: Non-Linearly Separable Data

We use **Kernelized SVM** for non-linearly separable data. Say, we have some non-linearly separable data in one dimension. We can transform this data into two-dimensions and the data will become linearly separable in two dimensions. This is done by mapping each 1-D data point to a corresponding 2-D ordered pair.
So for any non-linearly separable data in any dimension, we can just map the data to a higher dimension and then make it linearly separable. This is a very powerful and general transformation.

A **kernel** is nothing a measure of similarity between data points. The **kernel function** in a kernelized SVM tell you, that given two data points in the original feature space, what the similarity is between the points in the newly transformed feature space.

There are various kernel functions available, but two of are very popular :

- **Radial Basis Function Kernel (RBF):** The similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below. RBF is the default kernel used in SVM.

$$K(x, x') = exp(-\gamma||x - x'||)$$

- **Polynomial Kernel:** The Polynomial kernel takes an additional parameter, 'degree' that controls the model's complexity and computational cost of the transformation

A very interesting fact is that SVM does not actually have to perform this actual transformation on the data points to the new high dimensional feature space. This is called the **kernel trick**.

The Kernel Trick:
Internally, the kernelized SVM can compute these complex transformations just in terms of similarity calculations between pairs of points in the higher dimensional feature space where the transformed feature representation is implicit.
This similarity function, which is mathematically a kind of complex dot product is actually the kernel of a kernelized SVM. This makes it practical to apply SVM, when the underlying feature space is complex, or even infinite-dimensional. The kernel trick itself is quite complex and is beyond the scope of this article.
Important Parameters in Kernelized SVC (Support Vector Classifier)

1. **The Kernel** : The kernel, is selected based on the type of data and also the type of transformation. By default, the kernel is Radial Basis Function Kernel (RBF).
2. **Gamma** : This parameter decides how far the influence of a single training example reaches during transformation, which in turn affects how tightly the decision boundaries end up surrounding points in the input space. If there is a small value of gamma, points farther apart are considered similar. So more points are grouped together and have smoother decision boundaries (may be less accurate). Larger values of gamma cause points to be closer together (may cause overfitting).
3. **The 'C' parameter** : This parameter controls the amount of regularization applied on the data. Large values of C mean low regularization which in turn causes the training data to fit very well (may cause overfitting). Lower values of C mean higher regularization which causes the model to be more tolerant of errors (may lead to lower accuracy).

Pros of Kernelized SVM:

2. They are versatile : different kernel functions can be specified, or custom kernels can also be defined for specific datatypes.
3. They work well for both high and low dimensional data.

Cons of Kernelized SVM:

1. Efficiency (running time and memory usage) decreases as size of training set increases.
2. Needs careful normalization of input data and parameter tuning.
3. Does not provide direct probability estimator.
4. Difficult to interpret why a prediction was made.

CONCLUSION
Now that you know the basics of how an SVM works, you can go to the following link to learn how to implement SVM to classify items using Python :

https://www.geeksforgeeks.org/classifying-data-using-support-vector-machinessvms-in-python/

Recommended Posts:

Classifying data using Support Vector Machines(SVMs) in Python
Major Kernel Functions in Support Vector Machine (SVM)
SymPy | Permutation.support() in Python
copyreg — Register pickle support functions
NumPy | Vector Multiplication
Vector Projection using Python
Learning Vector Quantization
How to get the magnitude of a vector in NumPy?
How to create a vector in Python using NumPy
Introduction to Brython
Introduction to MoviePy
Introduction to Tkinter
Introduction to PySimpleGUI
Introduction to CherryPy
Introduction to Python3
Introduction to OpenCV

Introduction to Convolutions using Python

Introduction to Bokeh in Python

alokesh985
Check out this Author's contributed articles.

Article Tags : Machine Learning Python

Practice Tags : Machine Learning

1

0

☐ To-do ☐ Done

No votes yet.

Feedback/ Suggest Improvement Improve Article