Avinash Navlani
July 12th, 2018

SCIKIT-LEARN +1

15

28

Support Vector Machines with Scikit-learn

In this tutorial, you'll learn about Support Vector Machines, one of the most popular and widely used supervised machine learning algorithms.

SVM offers very high accuracy compared to other classifiers such as logistic regression, and decision trees. It is known for its kernel trick to handle nonlinear input spaces. It is used in a variety of applications such as face detection, intrusion detection, classification of emails, news articles and web pages, classification of genes, and handwriting recognition.

In this tutorial, you will be using scikit-learn in Python. If you would like to learn more about this Python package, I recommend you take a look at our Supervised Learning with scikit-learn course.

SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

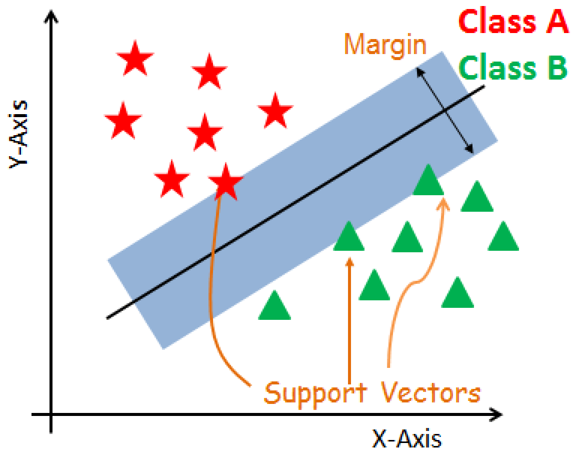
In this tutorial, you are going to cover following topics:

- Support Vector Machines

15
28

Support Vector Machines

Generally, Support Vector Machines is considered to be a classification approach, it but can be employed in both types of classification and regression problems. It can easily handle multiple continuous and categorical variables. SVM constructs a hyperplane in multidimensional space to separate different classes. SVM generates optimal hyperplane in an iterative manner, which is used to minimize an error. The core idea of SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes.



Support Vectors

Support vectors are the data points, which are closest to the hyperplane. These points will define the separating line better by calculating margins. These points are more relevant to the construction of the classifier.

Hyperplane

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

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

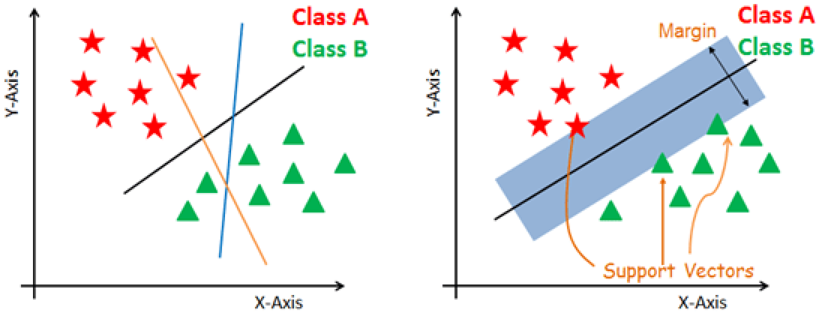
classes, then it is considered a good margin, a smaller margin is a bad margin.

How does SVM work?

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum marginal hyperplane in the following steps:

1. Generate hyperplanes which segregates the classes in the best way. Left-hand side figure showing three hyperplanes black, blue and orange. Here, the blue and orange have higher classification error, but the black is separating the two classes correctly.

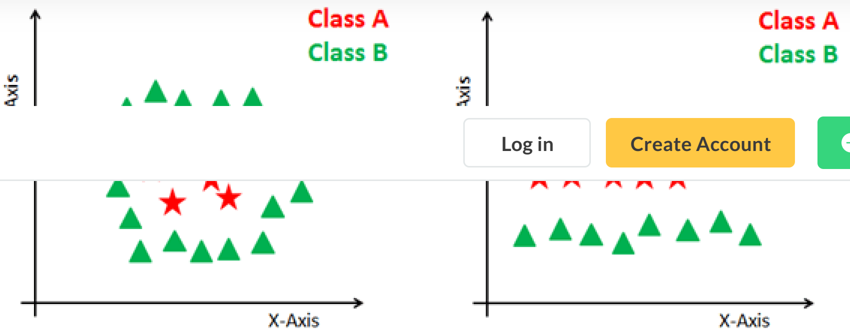
2. Select the right hyperplane with the maximum segregation from the either nearest data points as shown in the right-hand side figure.



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).

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). Now you can easily segregate these points using linear separation.



15
28

SVM Kernels

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick. Here, the kernel takes a low-dimensional input space and transforms it into a higher dimensional space. In other words, you can say that it converts nonseparable problem to separable problems by adding more dimension to it. It is most useful in non-linear separation problem. Kernel trick helps you to build a more accurate classifier.

- **Linear Kernel** A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

```
K(x, xi) = sum(x * xi)
```

- **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(x,xi) = 1 + sum(x * xi)^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

```
K(x,xi) = exp(-gamma * sum((x – xi^2))
```

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.

15
28

Classifier Building in Scikit-learn

Until now, you have learned about the theoretical background of SVM. Now you will learn about its implementation in Python using scikit-learn.

In the model the building part, you can use the cancer dataset, which is a very famous multi-class classification problem. This dataset is computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

The dataset comprises 30 features (mean radius, mean texture, mean perimeter, mean area, mean smoothness, mean compactness, mean concavity, mean concave points, mean symmetry, mean fractal dimension, radius error, texture error, perimeter error, area error, smoothness error, compactness error, concavity error, concave points error, symmetry error, fractal dimension error, worst radius, worst texture, worst perimeter, worst area, worst smoothness, worst compactness, worst concavity, worst concave points, worst symmetry, and worst fractal dimension) and a target (type of cancer).

This data has two types of cancer classes: malignant (harmful) and benign (not harmful). Here, you can build a model to classify the type of cancer. The dataset is available in the scikit-learn library or you can also download it from the UCI Machine Learning Library.

Loading Data

Let's first load the required dataset you will use.

```
#Load dataset
```

COMMUNITY
BETA
Tutorials
NEW
DATACAMP

15
28

Exploring Data

After you have loaded the dataset, you might want to know a little bit more about it. You can check feature and target names.

```
# print the names of the 13 features
print("Features: ", cancer.feature_names)

# print the label type of cancer('malignant' 'benign')
print("Labels: ", cancer.target_names)
```

```
Features:  ['mean radius' 'mean texture' 'mean perimeter' 'mean ar
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension'
Labels:  ['malignant' 'benign']
```

Let's explore it for a bit more. You can also check the shape of the dataset using shape.

```
# print data(feature)shape
cancer.data.shape
```

```
(569, 30)
```

Let's check top 5 records of the feature set.

```
# print the cancer data features (top 5 records)
print(cancer.data[0:5])
```

Want to leave a comment?

```
[[1.799e+01 1.038e+01 1.228e+02 1.001e+03 1.184e-01 2.776e-01 3.00
  1.471e-01 2.419e-01 7.871e-02 1.095e+00 9.053e-01 8.589e+00 1.53
```

15

28

```
  1.733e+01 1.846e+02 2.019e+03 1.622e-01 6.656e-01 7.119e-01 2.65
  4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 1.326e+03 8.474e-02 7.864e-02 8.69
  7.017e-02 1.812e-01 5.667e-02 5.435e-01 7.339e-01 3.398e+00 7.40
  5.225e-03 1.308e-02 1.860e-02 1.340e-02 1.389e-02 3.532e-03 2.49
  2.341e+01 1.588e+02 1.956e+03 1.238e-01 1.866e-01 2.416e-01 1.86
  2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 1.203e+03 1.096e-01 1.599e-01 1.97
  1.279e-01 2.069e-01 5.999e-02 7.456e-01 7.869e-01 4.585e+00 9.40
  6.150e-03 4.006e-02 3.832e-02 2.058e-02 2.250e-02 4.571e-03 2.35
  2.553e+01 1.525e+02 1.709e+03 1.444e-01 4.245e-01 4.504e-01 2.43
  3.613e-01 8.758e-02]
 [1.142e+01 2.038e+01 7.758e+01 3.861e+02 1.425e-01 2.839e-01 2.41
  1.052e-01 2.597e-01 9.744e-02 4.956e-01 1.156e+00 3.445e+00 2.72
  9.110e-03 7.458e-02 5.661e-02 1.867e-02 5.963e-02 9.208e-03 1.49
  2.650e+01 9.887e+01 5.677e+02 2.098e-01 8.663e-01 6.869e-01 2.57
  6.638e-01 1.730e-01]
 [2.029e+01 1.434e+01 1.351e+02 1.297e+03 1.003e-01 1.328e-01 1.98
  1.043e-01 1.809e-01 5.883e-02 7.572e-01 7.813e-01 5.438e+00 9.44
  1.149e-02 2.461e-02 5.688e-02 1.885e-02 1.756e-02 5.115e-03 2.25
  1.667e+01 1.522e+02 1.575e+03 1.374e-01 2.050e-01 4.000e-01 1.62
  2.364e-01 7.678e-02]]
```

Let's take a look at the target set.

```
# print the cancer labels (0:malignant, 1:benign)
print(cancer.target)
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1
 1 0 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 0 0 1 1 1 1 0 1
 1 1 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 0 1 1
 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 0 1 1 1 1 0 1 1 0
 1 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1
 1 0 1 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1 1 0 1 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 0 0 1 1 1 1 1 1 1 1
 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0
```

Want to leave a comment?

```
 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1
 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 0 0 0 0 0 1]
```

Splitting Data

15
28

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Split the dataset by using the function `train_test_split()`. you need to pass 3 parameters features, target, and test_set size. Additionally, you can use random_state to select records randomly.

```
# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(cancer.data, c
```

Generating Model

Let's build support vector machine model. First, import the SVM module and create support vector classifier object by passing argument kernel as the linear kernel in `SVC()` function.

Then, fit your model on train set using `fit()` and perform prediction on the test set using `predict()`.

```
#Import svm model
from sklearn import svm

#Create a svm Classifier
```

```
clf.fit(X_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Let's estimate how accurately the classifier or model can predict the breast cancer of patients.

predicted values.

15
28

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics

# Model Accuracy: how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9649122807017544
```

Well, you got a classification rate of 96.49%, considered as very good accuracy.

For further evaluation, you can also check precision and recall of model.

```
# Model Precision: what percentage of positive tuples are labeled
print("Precision:",metrics.precision_score(y_test, y_pred))

# Model Recall: what percentage of positive tuples are labelled as
print("Recall:",metrics.recall_score(y_test, y_pred))
```

```
Precision: 0.9811320754716981
Recall: 0.9629629629629629
```

Well, you got a precision of 98% and recall of 96%, which are considered as very good values.

Tuning Hyperparameters

- **Kernel**: The main function of the kernel is to transform the given dataset input data into the required form. There are various types of functions such as linear, polynomial, and radial basis function (RBF).

dimension. In some of the applications, it is suggested to use a more complex kernel to separate the classes that are curved or nonlinear.

COMMUNITY
BETA
Tutorials
NEW
DATACAMP
15
28

- **Regularization**: Regularization parameter in python's Scikit-learn C parameter used to maintain regularization. Here C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimization how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term. A smaller value of C creates a small-margin hyperplane and a larger value of C creates a larger-margin hyperplane.

- **Gamma**: A lower value of Gamma will loosely fit the training dataset, whereas a higher value of gamma will exactly fit the training dataset, which causes over-fitting. In other words, you can say a low value of gamma considers only nearby points in calculating the separation line, while the a value of gamma considers all the data points in the calculation of the separation line.

Advantages

SVM Classifiers offer good accuracy and perform faster prediction compared to Naïve Bayes algorithm. They also use less memory because they use a subset of training points in the decision phase. SVM works well with a clear margin of separation and with high dimensional space.

Disadvantages

SVM is not suitable for large datasets because of its high training time and it also takes more time in training compared to Naïve Bayes. It works poorly with overlapping classes and is also sensitive to the type of kernel used.

Conclusion

Congratulations, you have made it to the end of this tutorial!

In this tutorial, you covered a lot of ground about Support vector machine algorithm, its working, kernels, hyperparameter tuning, model

COMMUNITY
BETA
Tutorials
NEW
DATACAMP

package. You have also covered its advantages and disadvantages. I hope you have learned something valuable!

To learn more about this type of classifiers, you should take a look at our Linear Classifiers in Python course. It introduces other types of regression and loss functions, as well as Support Vector Machines.

I look forward to hearing any feedback or questions. You can ask the question by leaving a comment and I will try my best to answer it.

15
28

COMMENTS

Rohit Jagannath
Can you cross check the expression(Z is the squared sum of both x and y: z=x^2=y^2) for correction?

Also, Can you try to do the same with Train, Test and Validate split?

3 REPLY 14/10/2018 03:14 PM

Avinash Navlani
Thanks for the feedback and spotting the mistake. It should be z=x^2+y^2.

Yes, we can do this. Already, SVM performs analysis in multidimensional dataset to classify.

3 REPLY 17/10/2018 11:20 AM

Saad Munir
How to develop a data-set yourself for the SVM classifier? Also is there any pre defined data set of word documents that can be used for Microsoft word document carving?

2 REPLY 14/11/2018 11:44 PM

Avinash Navlani
You can take any dataset and try out SVM classifier, tune your hyperparameters. I have no idea about MS word document carving.

1 REPLY 17/11/2018 05:47 AM

Want to leave a comment?

```
K(x,xi) = exp(-gamma * sum((x – xi^2)) should be
```

```
K(x,xi) = exp(-gamma * sum((x – xi)^2)
```

▲ 3 REPLY | 07/12/2018 08:21 AM

Avinash Navlani
Yup, You are correct.

▲ 1 REPLY | 08/12/2018 07:53 AM

COMMUNITY

BETA

Tutorials

NEW

DATACAMP

15

▲ 28

Rohit Kumar
Can you make an article of using SVM with Autoencoder. That'll be very useful on how to use svm as a classifier with autoencoder

▲ 3 REPLY | 16/12/2018 10:00 AM

Avinash Navlani
Thanks for your feedback!

I will try when I get time. Right now i am busy with other articles.

▲ 1 REPLY | 19/12/2018 10:42 PM

Guillermo Viñas
Dear, you have an example of multivariate analysis type MIMO, to detect models and predict variables based on neural networks in python or some alternative in python that allows to solve the indicated?

▲ 1 REPLY | 26/12/2018 12:19 PM

haider ali
can you show how to apply this SVM code om multi-class classification. thank you

▲ 2 REPLY | 31/03/2019 01:33 AM

Paian Simarmata
hi avinash .

i have a project to predict the maturity of roasted coffee bean from its color.

since the coffee bean roasting have three types of roasting ==> light, medium and dark with also produce different color maturity.

i have to take the image of the roasted coffee bean realtime using webcam and sent the image to be processed and determine the color of maturity so the machine will stop roasting.

my question is can i use SVM to classify the image of the coffee bean color ?

i hope you can help me.

▲ 2 REPLY | 06/04/2019 05:40 AM

Avinash Navlani

▲ 1 REPLY | 07/04/2019 10:02 AM

Paulo Oliveira
Thanks for the post!
I have implemented and concluded your tutorial! Now my model is trained right? but how can i test with a new input, and see what the svm think it is?

▲ 3 REPLY | 02/06/2019 03:50 PM

Bahare Samadi
Hi, I have 27 samples that I would like to do binary classification. Each sample is a 2D- matrice. (100 x 96). I try to use the SVM, but it was not possible, since I have arrays with dim 3.
ValueError: Found array with dim 3. Estimator expected <= 2. Could you please advice ?

▲ 2 REPLY | 13/06/2019 11:56 AM

Yogi Pratama
Bagaimana saya membuat 3 klasfifikasi dlam contoh kasus adalah, klasifikasi (Normal, Kurang Normal, dan Tidak Normal). Apakah bisa? Terimakasih sebelumnya

▲ 1 REPLY | 13/06/2019 11:12 PM

Want to leave a comment?