Gurucharan M K Follow

Jul 11, 2020 · 6 min read ★ · Listen

Machine Learning Basics: Support Vector Regression

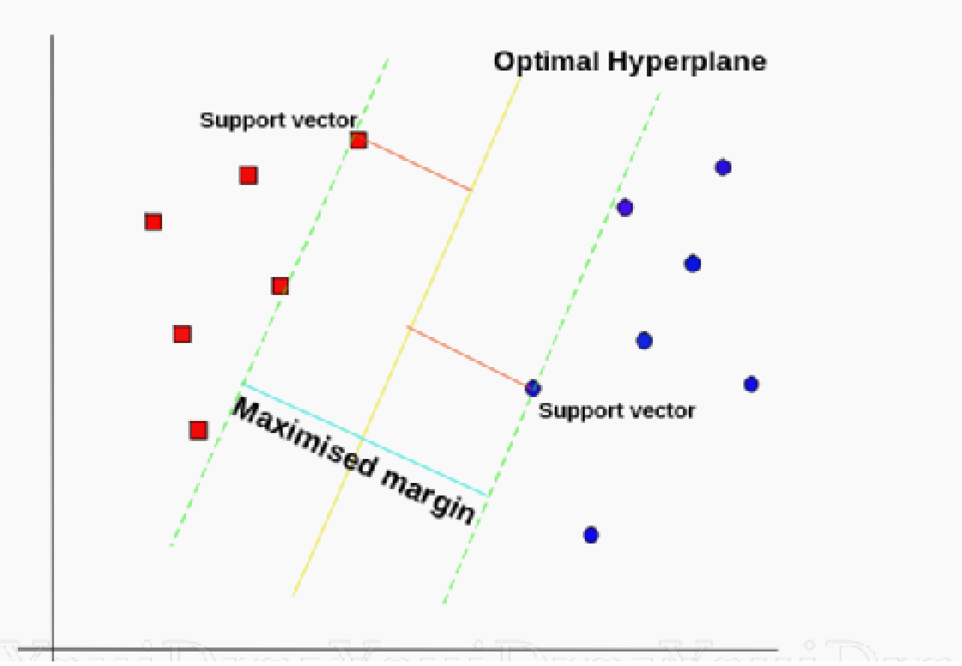
Learn to build a Support Vector Regression (SVR) model in Machine Learning and analyze the results.

In the previous stories, I explained the Machine Learning program for building Linear and Polynomial Regression model in Python. In this article, we will go through the program for building a Support Vector Regression model based on non-linear data.

Overview of SVR

Support Vector Machine (SVM) is a very popular Machine Learning algorithm that is used in both Regression and Classification. Support Vector Regression is similar to Linear Regression in that the equation of the line is `y= wx+b` In SVR, this straight line is referred to as ***hyperplane***. The data points on either side of the hyperplane that are closest to the hyperplane are called ***Support Vectors*** which is used to plot the boundary line.

Unlike other Regression models that try to minimize the error between the real and predicted value, the SVR tries to fit the best line within a threshold value (Distance between hyperplane and boundary line), **a**. Thus, we can say that SVR model tries satisfy the condition `-a < y-wx+b < a` . It used the points with this boundary to predict the value.



Source

For a non-linear regression, the kernel function transforms the data to a higher dimensional and performs the linear separation. Here we will use the ***rbf*** kernel.

In this example, we will go through the implementation of ***Support Vector Regression (SVM)***, in which we will predict th nt based on his or her number of hours put into study.

63 3

Problem Analysis

In this data, we have one independent variable *Hours of Study* and one dependent variable *Marks*. In this problem, we have to train a SVR model with this data to understand the correlation between the Hours of Study and Marks of the student and be able to predict the student's mark based on their number of hours dedicated to studies.

Step 1: Importing the libraries

In this first step, we will be importing the libraries required to build the ML model. The ***NumPy*** library and the ***matplotlib*** are imported. Additionally, we have imported the ***Pandas*** library for data analysis.

Step 2: Importing the dataset

In this step, we shall use pandas to store the data obtained from my github repository and store it as a Pandas DataFrame using the function “**pd.read_csv**”.

We go through our dataset and assign the independent variable (x) to the column “***Hours of Study***” and the dependent variable (y) to the last column, which is the “***Marks***” to be predicted.

```
dataset = pd.read_csv('https://raw.githubusercontent.com/mk-gurucharan/Regression/master/SampleData.csv')

X = dataset.iloc[:, 0].values
y = dataset.iloc[:, 1].values
y = np.array(y).reshape(-1,1)

dataset.head(5)

>>

Hours of Study   Marks
32.502345        31.707006
53.426804        68.777596
61.530358        62.562382
47.475640        71.546632
59.813208        87.230925
```

We use the corresponding .iloc function to slice the DataFrame to assign these indexes to X and Y. In this, the *Hours of Study* is taken as the independent variable and is assigned to X. The dependent variable that is to be predicted is the last column which is *Marks* and it is assigned to y. We will reshape the variable ***y*** to a column vector using `reshape(-1,1)`.

Step 3: Feature Scaling

Most of the data that are available usually are of varying ranges and magnitudes which makes building the model difficult. Thus, the range of the data needs to be normalized to a smaller range which enables the model to be more accurate when training. In this dataset, the data is normalized between to small values near zero. For example, the *score of 87.23092513 is normalized to 1.00475931 and score of 53.45439421 is*

```
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
X = sc_X.fit_transform(X.reshape(-1,1))
y = sc_y.fit_transform(y.reshape(-1,1))
```

Feature Scaling is mostly performed internally in most of the common Regression and Classification models. Support Vector Machine is not a commonly used class and hence the data is normalized to a limited range.

Step 4: Training the Support Vector Regression model on the Training set

In building any ML model, we always need to split the data into the training set and the test set. The SVR Model will be trained with the values of the *training set* and the predictions are tested on the *test set*. Out of 100 rows, 80 rows are used for training and the model is tested on the remaining 20 rows as given by the condition, `test_size=0.2`

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size
= 0.2)
```

Step 5: Training the Support Vector Regression model on the Training set

In this, the function ***SVM*** is imported and is assigned to the variable `regressor`. The kernel ***"rbf"*** (Radial Basis Function) is used. RBF kernel is used to introduce a non-linearity to the SVR model. This is done because our data is non-linear. The `regressor.fit` is used to fit the variables *X_train* and *y_train* by reshaping the data accordingly.

```
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train.reshape(-1,1), y_train.reshape(-1,1))
```

Step 6: Predicting the Test set Results

```
y_pred = regressor.predict(X_test)
y_pred = sc_y.inverse_transform(y_pred)
```

Step 7: Comparing the Test Set with Predicted Values

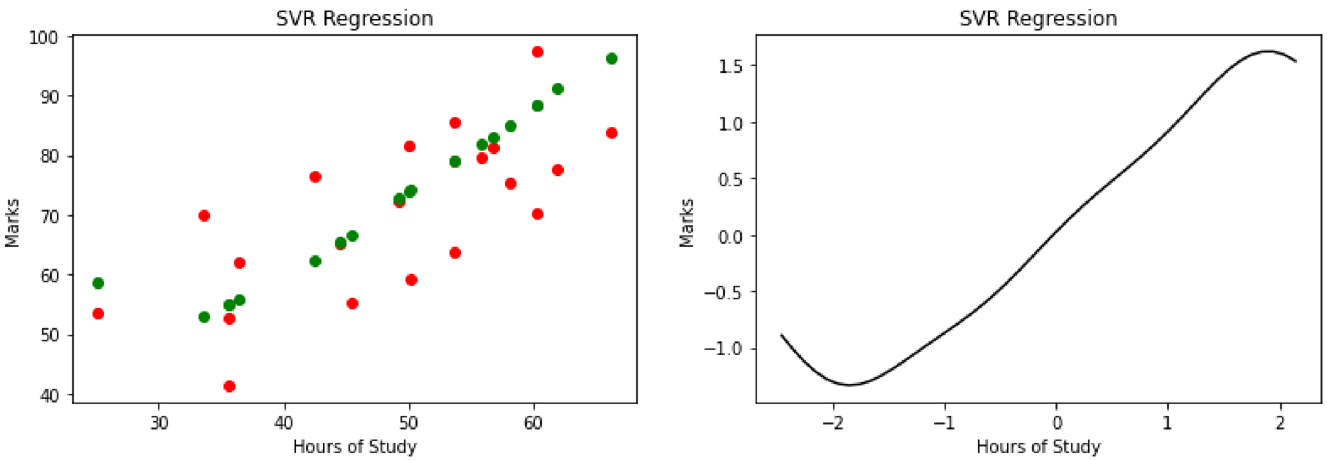
In this step, we shall display the values of *y_test* as ***Real Values*** and *y_pred* values as ***Predicted Values*** for each X_test against each other in a Pandas DataFrame.

```
df = pd.DataFrame({'Real
Values':sc_y.inverse_transform(y_test.reshape(-1)), 'Predicted
Values':y_pred})
df

>>
Real Values   Predicted Values
31.707006     53.824386
76.617341     61.430210
65.101712     63.921849
85.498068     80.773056
81.536991     72.686906
79.102830     60.357810
95.244153     89.523157
52.725494     54.616087
95.455053     82.003370
80.207523     81.575287
79.052406     67.225121
83.432071     73.541885
85.668203     78.033983
71.300880     76.536061
52.682983     63.993284
45.570589     53.912184
63.358790     76.077840
57.812513     62.178748
82.892504     64.172003
83.878565     93.823265
```

We can see that there is a significant deviation of the predicted values with the real values of the test set and hence we can conclude that this model is not the perfect fit for the following data.

Step 8: Visualising the SVR results

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(sc_X.inverse_transform(X_test),
sc_y.inverse_transform(y_test.reshape(-1)), color = 'red')
plt.scatter(sc_X.inverse_transform(X_test), y_pred, color = 'green')

plt.title('SVR Regression')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Hours of Study vs Marks (SVR)

In this graph, the Real values are plotted in “***Red***” color and the Predicted values are plotted in “***Green***” color. The plot of the SVR model is also shown in “***Black***” color.

I am attaching a link of my github repository where you can find the Google Colab notebook and the data files for your reference.

mk-gurucharan/Regression

GitHub is home to over 50 million developers working together to host and review code, manage projects, and build…

github.com

Hope I have been able to clearly explain the program for building a Support Vector Regression model with a non-linear dataset.

You can also find the explanation of the program for other Regression models below:

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Support Vector Regression
- Decision Tree Regression
- Random Forest Regression