SVR - Support Vector Regression

Tuesday, January 17, 2017 8:13 AM

Data

Position	Level	Salary
Business Analyst	1	45000
Junior Consultant	2	50000
Senior Consultant	3	60000
Manager	4	80000
Country Manager	5	110000
Region Manager	6	150000
Partner	7	200000
Senior Partner	8	300000
C-level	9	500000
CEO	10	1000000

Python

Here, we use the same preprocessing steps as before. Since the data is small, we are not going to further divide it into training and test sets.

```
# SVR - Support Vector Regression
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:,1:2].values
y = dataset.iloc[:, 2].values
```

SVR Model requires the incoming data to already be feature scaled. It does not do feature scaling on it's own. This is a very important step !

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_Y = StandardScaler()
X = sc_X.fit_transform(X)
Y = sc_Y.fit_transform(y)
```

Now, we train the SVR model by importing the model from the sklearn.svm library.

```
# Build a SVR Model on the scaled feature set
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X, Y)
```

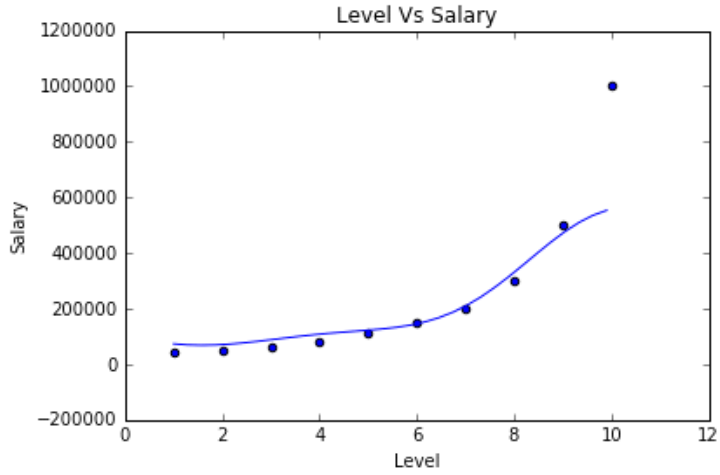
Here, the most important step is the choice of Kernels. We use 'rbf' by default but it can be changed. Other options include 'poly', 'linear', 'sigmoid', etc.

We then Visualize how the model has learnt the feature mapping by plotting the predictions of the model on a grid of X values

```
# Visualizing SVR predictions
# Generating grid of X values to obtain a nice smooth curve during plot of the predictions.
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), y)
plt.plot(sc_X.inverse_transform(X_grid), sc_Y.inverse_transform(regressor.predict(X_grid)))
plt.title('Level Vs Salary')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.show()
```

Here, since the model input and model output are scaled, we need to use Feature scaling operations to convert them to the correct representations for the model to process. Hence the use of sc_Y.inverse_transform() and sc_X.inverse_transform()



Note - SVR removes anomalies. Hence the outlier for salary of CEO is not fit by the model properly. We can change the penalty parameter to be more inclusive or less inclusive of outliers. Example : If we use the value of C = 10.0 (default is 1.0), then we get

```
# Build a SVR Model on the scaled feature set
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf', C = 10.0)
regressor.fit(X, Y)
```

Finally for predicting a new point, we use the same feature scaling and model for making the prediction.

```
# Predict a new Point
y_new_pred = sc_Y.inverse_transform(regressor.predict(sc_X.transform(np.array([[6.5]]))))
print(y_new_pred)
```

Result - 170370.0204065

This is a very good estimate.

R

For R, we need to first install the library e1071. We can do this using the following command

```
Packages.install('e1071')
```

We need to then import the library as follows

```
Library(e1071)
```

Data preprocessing is the same as before. Very simple

```
dataset = read.csv('POsition_Salaries.csv')
dataset = dataset[2:3]
```

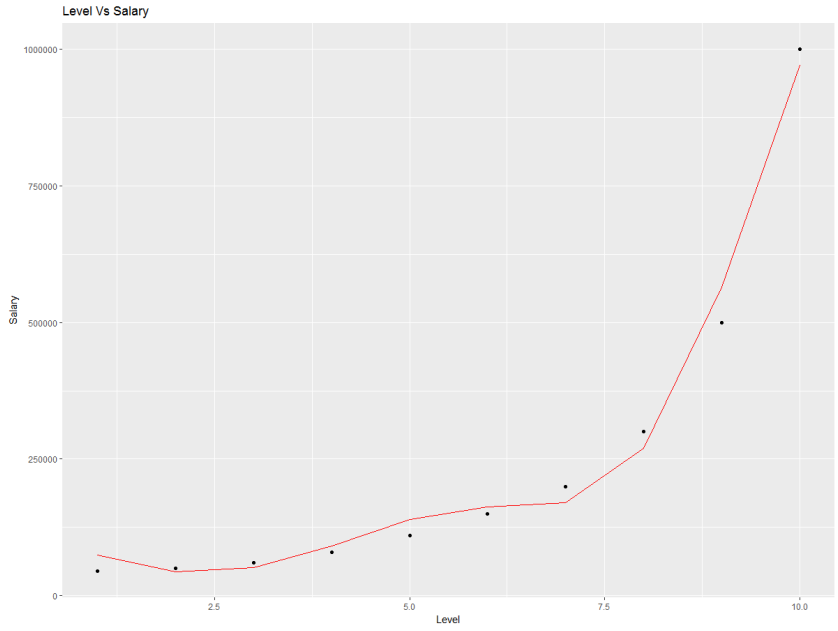
We can build a SVR model now. For creating the SVR model, we use the svm function and pass the type as 'eps-regression' in it. This makes the SVM model act as a regression model. Cost similar to what we did in python is used for finding a balance between overfitting and underfitting of data. Cost is the 'C'-constant of the regularization term in the Lagrange formulation.

```
# Fitting SVR Regressor to the dataset
```

```
svr_regressor = svm(formula = Salary ~ Level,
                    data = dataset,
                    type = 'eps-regression',
                    cost = 10.0)
```

Finally we plot the model output to gain an understanding of how the model is performing

```
# Plotting the model
library(ggplot2)
ggplot() +
  geom_point(aes(dataset$Level, dataset$Salary)) +
  geom_line(aes(dataset$Level, predict(svr_regressor, data = dataset)), color = 'red') +
  ggtitle('Level Vs Salary') + xlab('Level') + ylab('Salary')
```



We finally predict value for a new data point as follows.

```
# Predict a new Data point
value = predict(svr_regressor, data.frame(Level = 6.5))
```

Result - 163414

This is a good estimate for the given level.