Support Vector Regression (SVR)

Mohan M J

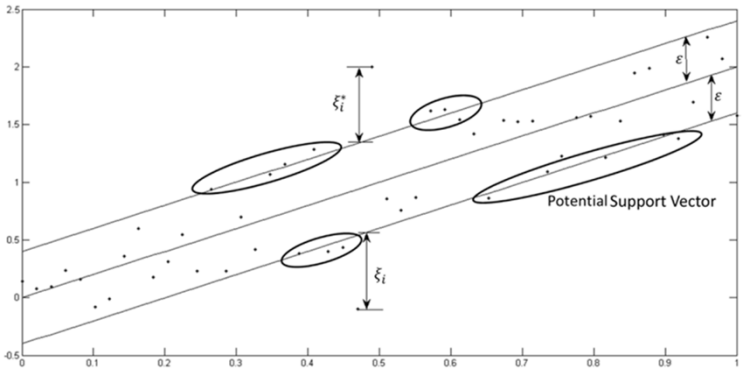
Introduction

- The regression problem is a generalization of the classification problem, in which the model returns a continuous-valued output, as opposed to an output from a finite set.
- In other words, a regression model estimates a continuous-valued multivariate function
- SVMs represent this optimal hyperplane with support vectors
- In a classification problem the vector X is used to define a hyperplane that separates the two different classes
- These vectors are used to perform linear regression
- The closest to the test points are referred to as support vectors

Introduction

- SVM generalization to SVR is accomplished by introducing an ε-insensitive region around the function, called the ε -tube
- This tube reformulates the optimization problem to find the tube that best approximates the continuous-valued function, while balancing model complexity and prediction error
- The hyperplane is represented in terms of support vectors, which are training samples that lie outside the boundary of the tube.
- points outside the tube are penalized, but those within the tube, either above or below the function, receive no penalty.

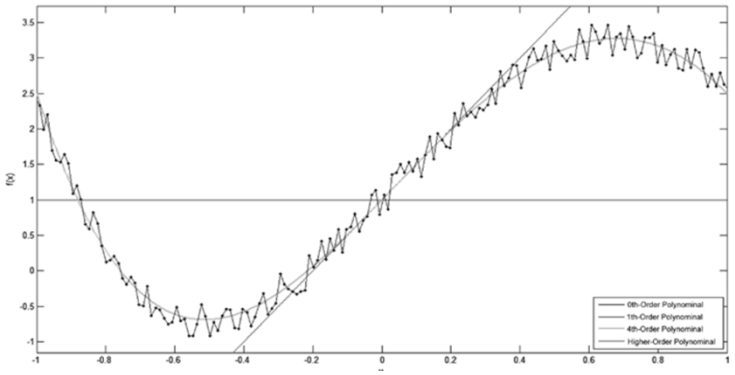
SVR



$$y = f(x) = < w, x > + b = \sum_{j=1}^{M} w_j x_j + b,\ y,\ b \in \mathbb{R}, x,\ w \in \mathbb{R}^M$$

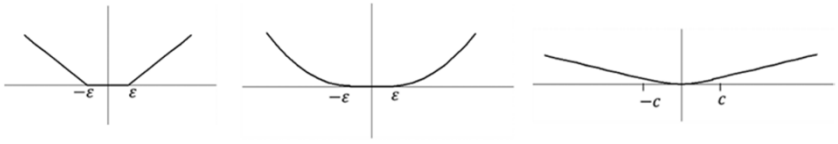
$$f(x) = \begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x \\ 1 \end{bmatrix} = w^T x + b \quad x, w \in \mathbb{R}^{M+1}$$

SVR



$$f(x,w) = \sum_{i=1}^{M} w_i x^i,\ x \in \mathbb{R}, w \in \mathbb{R}^M.$$

SVR



linear *quadratic* *Huber*

$$L_\varepsilon\left(y, f(x,w)\right) = \begin{cases} 0 & |y - f(x,w)| \le \varepsilon; \\ |y - f(x,w)| - \varepsilon & otherwise, \end{cases}$$

$$L_\varepsilon\left(y, f(x,w)\right) = \begin{cases} 0 & |y - f(x,w)| \le \varepsilon; \\ \left(|y - f(x,w)| - \varepsilon\right)^2 & otherwise, \end{cases}$$

$$L\left(y, f(x,w)\right) = \begin{cases} c|y - f(x,w)| - \dfrac{c^2}{2} & |y - f(x,w)| > c \\ \dfrac{1}{2}|y - f(x,w)|^2 & |y - f(x,w)| \le c \end{cases}$$

Python Code

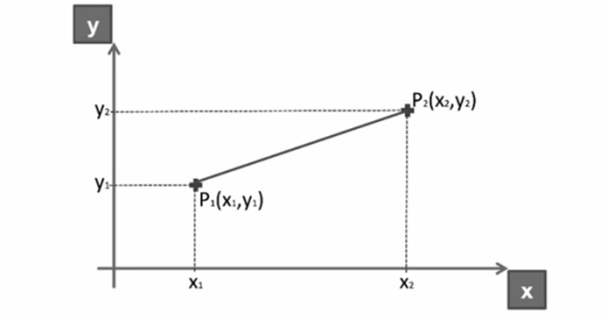
```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# Importing the dataset
dataset = pd.read_csv('Position_Salaries.csv')
X = dataset.iloc[:, 1:2].values
X
y = dataset.iloc[:, 2:].values
y
```

Python Code: Feature Scaling

```
# Feature Scaling SVR model don't have inbuilt
feature scaling
from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
X= sc_X.fit_transform(X)
sc_y = StandardScaler()
y = sc_y.fit_transform(y)
```

Feature Scaling

- Many of the ML models are based on Euclidian Distance
- Large number dominate the smaller



Euclidean Distance between P_1 and $P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Feature Scaling

Change the variables to same scale

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

Python Code: SVR Regression

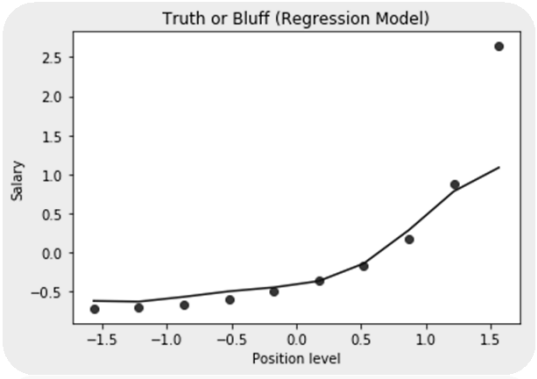
```
# Fitting the Regression Model to the dataset
from sklearn.svm import SVR
myModel = SVR(kernel='rbf') #Gaussian Kernel
myModel.fit(X,y)

# Predicting a new result
y_pred =
sc_y.inverse_transform(myModel.predict(sc_X.transfo
rm(np.array(6.5))))
y_pred
```

Python Code: Visualize

```
# Visualising the Regression results
plt.scatter(X, y, color = 'red')
plt.plot(X, myModel.predict(X), color = 'blue')
plt.title('Truth or Bluff (Regression Model)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

CEO is considered as outlier!



THANKS