~~# yolo v7 training~~

~~=> It needs a annotated~~ data

For Linear Regression (SLR) ordinary least square method is used

SVR (~~super~~ support vector regression)

=> In this we are going to study ~~super~~ support vector regression mostly linear SVR

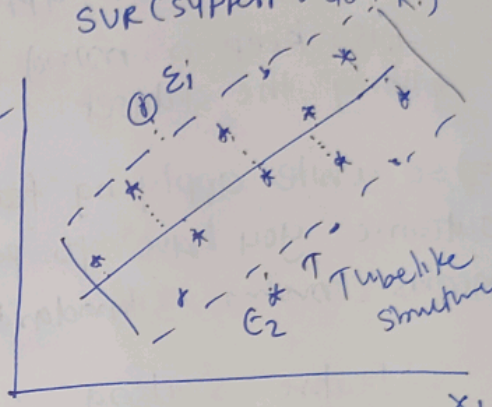
SVR = Support vector regression

=> In SVR along with line it has tube around that line

SLR

this tube is called as absolute insensitive tube

SVR (support vector R.)



=> Here in SVR distance betn point and line not came to play as point and line both resides in same insensitive tube

=> the point outside tube will calculate on behind of outer edge of tube

ε_i => is distance betn outside point & tube

=> To find best tube

$$= \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\varepsilon_i + \varepsilon_i^*) \longrightarrow \min$$

=> Just formula we use in SLR

=> outside point in tube will determine position & location of tube

=> Each outside point can be plotted shown as vector thus this model called as support vector regression

non-linear SVR

In 3D SVR is much complex than which it looks in 2D

(Code after this will be on non linear SVR)

Using of feature scaling depends on model which you're using.

Feature Scaling in SVR

Sometime there is need arises to apply feature scaling on y i.e dep on depent variable vector.

(⇒ so for this also the SVR model for value of different range so SVR fail to perform well

mainly we only apply feature scaling on X (matrix (feature))

(Also keep in mind we apply feature scaling after splitting the dataset.

⇒ so while applying feature scaling to y (dvv) in final outcome you have to perform inverse feature scaling means converting standardized value to final value

Feature Scaling

Imp standard scalar needs 2D array as input ⊛

Ex $y = [y] \Rightarrow$ 1D array converted to 2D array

Reshape array:- If you have numpy array say b

np — arr — so you can apply reshape on it

b = b.reshape(no. of rows, no. of columns)

2 to convert horizontal array to vertical use

b = b.reshape(len(b), 1) → In each no. column or element in a row

Ad use standard Scalar.

So as data are diffrnt so you have use diffrnt Standard scalar object for X & y

#Training SVR model

Code

```
from sklearn.svm import SVR
#create object
regressor = SVR(kernal = 'rbf');
```

‼ In SVR model we have to pass kernal as argument

we can use linear kernal or nonlinear kernal according to our need

Ex:- guassian RBF (see it's plot online)

$$k(\vec{x}, \vec{i}) = e^{\left(\frac{\|\vec{x} - \vec{i}\|^2}{2\sigma^2}\right)}$$

⇒ There are mainly type of kernal all can read online (RBF) one which we are going to use

that will build model & now train it

```
regressor.fit(x,y);
```

⇒ and thus you builded your model

#after doing model

now while predicting you have to perform something called reverse standardization in order to convert standard value to true original one

(*) If you want some prediction from model so you have to feed input is same way & same form that of (X).

and output is same which you feed as ytrain format, it it is feature scaled so will be the output so just do inverse standarization with same object who did standaridization

```
(Og) = SC.reverse_transform(value)
```

‼ this will original value which is non Standaridised

Ex m = regressor.predict(SC1.transform([[6.5]]).

↑ feed (X) & you get (y)

#now in order to find y do inverse transform on standfeatur object of y

value = SC2.inversetransform([m])

↑

⇒ Inverse transform also take 2D input.

#How to plot

find inversetransform of x & y

plt.scatter(xinverse, yinverse, color = "red").

plt.plot(X, pred, color = "yellow"),

↑

Code for pred

pred = model.predict(X) → X already in form

| regressor

pred is y so inverse transform it to y

pred = SC2.inversetransform(pred)

You get pred.