

Ridge and Lasso Regression

Thursday, July 6, 2023 10:06 AM

When joining two points on a plot using best fit line,
We wanna join the two points and construct a line thru them
We might get a certain cost function in that process which we need to reduce

Now lets say we plot our predicted values (\hat{y}) against our observed values (y)
In that case, **Sum of residuals** is defined as

$$\sum_{i=1}^n (y - \hat{y})^2$$

If our prediction is seen to be above the actual value during testing, we can say that our current model is **OVERFITTING**. During **overfitting**:

- Wrt the training dataset we get low error
- Wrt the testing dataset we get a high error
- We always get a steep slope in the case of overfitting

During **underfitting**:

- High error during training
- High error during testing

Using **ridge and lasso regression**, we can convert the overfitting condition (high variance) into a model with low variance and low bias. So we go from an **overfitted model to a well generalised model**.

In the case of **linear regression**, we tried to reduce the cost function or the sum of residuals function

In the case of **ridge regression**, we add a **whole new parameter** (lambda times slope sq) to the cost function. We hence get a new function. Now we try to reduce the newly formed function as a **WHOLE**.

$$\left. \begin{array}{l} \sum_{i=1}^n (y - \hat{y})^2 \\ + \\ \lambda \times (\text{slope})^2 \end{array} \right\} \text{WHOLE}$$

So the process will be

- Converting a **steep slope** (in a unit movement in x axis, high movement in the slope)

So, during the training case of ridge regression, we know that the linear terms will be zero. During the first iteration, we take the new params as follows and hence get the resulting answer as 1.69. (JUST AN EXAMPLE)

$$\left. \begin{array}{l} \sum_{i=1}^n (y - \hat{y})^2 = 0 \\ + \\ \lambda \times (\text{slope})^2 \end{array} \right\} \text{WHOLE}$$
$$0 + 1 \times (1.3)^2 = 0 + 1.69 = 1.69$$

Remember how we would stop at this point in linear regression when the sum of residuals was 0. But here we aren't stopping, and instead getting 1.69

Now we try and get another best fit line that would reduce this value of 1.69.

So we take another best fit line

With lower steepness

So the slope in this new line will be smaller

Then the new iteration will lead to a result that is obviously lesser than 1.69

Since the new error value is lesser than the previous line value, we are now gonna proceed with the new best fit line.

Now this value for new slopes will keep going lower thru the process of experiments and all that.

The lambda times slope sq will keep getting added and in the end we **PENALISE THE STEEPNESS**

Altho we might have some bias (error during training), for the testing dataset the new best fit line will surely give us a more generalised model.

The extra and steeper slope will be penalized by the operation of lambda times slope sq.
This is ridge regression

LASSO!

Same as the ridge regression equation and methodology but
It uses **magnitude of slope** instead of slope sq.

$$\left. \begin{array}{l} \sum_{i=1}^n (y - \hat{y})^2 = 0 \\ + \\ \lambda \times |\text{slope}| \end{array} \right\}$$

It helps in the tasks of feature selection, it will only select the features that are more significant
The features in which the slope value is 0 is deselected and only the relevant features will be selected and moved on.

SVM-c

Used for classification as well as regression problems

Saturday, September 16, 2023

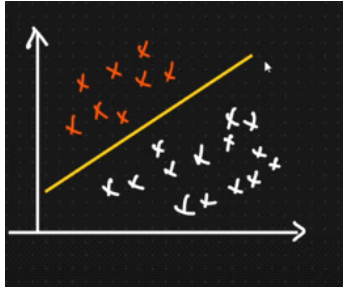
12:54 AM

For a classification problem using logistic regression, we have been using the process as follows

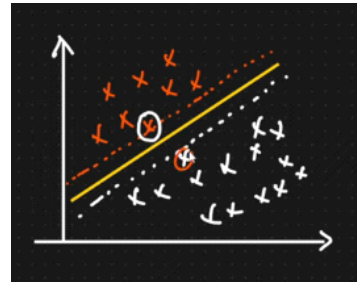
For scatter plots of points of two types red and white as follows

We draw a line that well divides the two input types

So that there is a distinction between the two



Using **SVM**, we make two more lines on either sides, called **marginal lines**



These two lines are called **support vectors**

Since in the above case, we don't have any errors, then we called the yellow plane as **hard marginal plane**

Errors then its called **soft marginal plane**

Margin Optimization:

SVM aims to maximize the margin between the **support vectors**

This helps to minimize the classification error.

Margin = $2 / ||w||$, where w is the weight vector perpendicular to the hyperplane

SVMs can be extended to handle non-linearly separable data by using a kernel function.


A kernel function maps the input data into a higher-dimensional space,

That will be making it easier to find a hyperplane that separates the data.

- linear kernel,
- polynomial kernel
- radial basis function (RBF) kernel.

Once the optimal hyperplane is found during training, SVM can be used for classification by simply evaluating the sign of the decision function:

scss

 Copy code

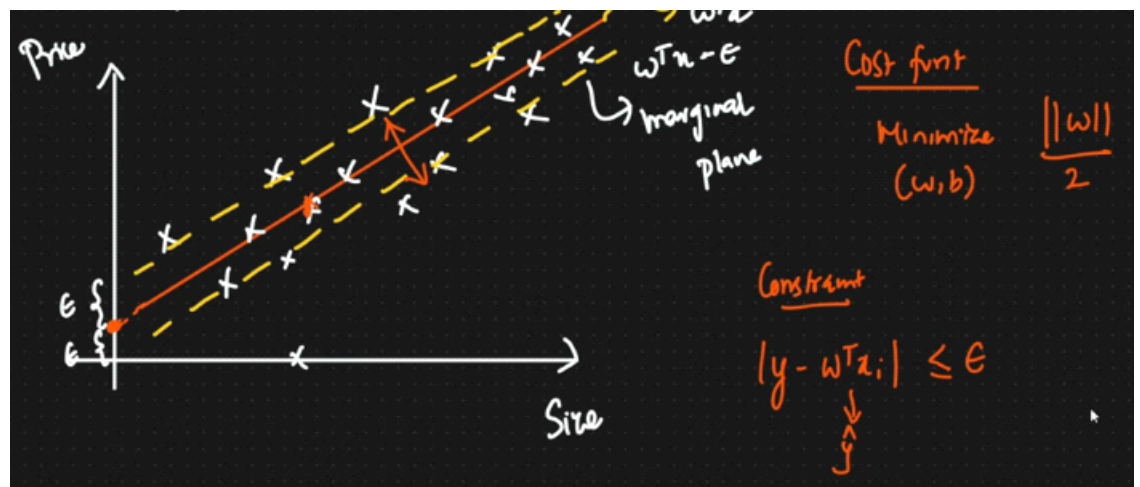
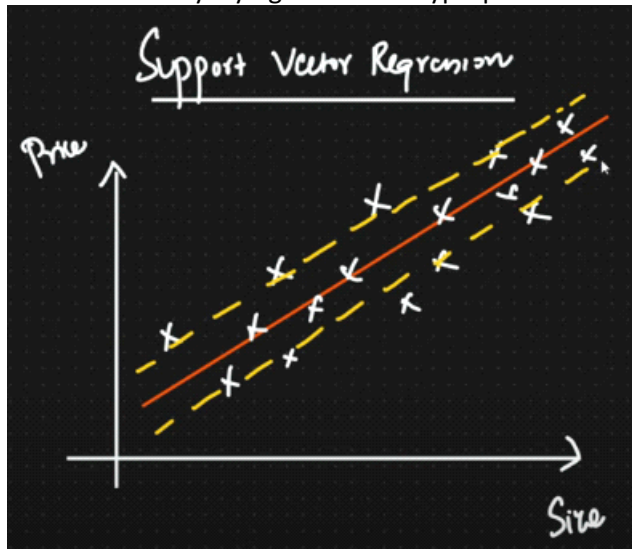
$$f(x) = w \cdot x + b$$

SVM-r

Saturday, September 16, 2023 1:14 AM

Suppose we have a plot between house price and house size

Then we start by trying to make a hyperplane that well divides the two input types



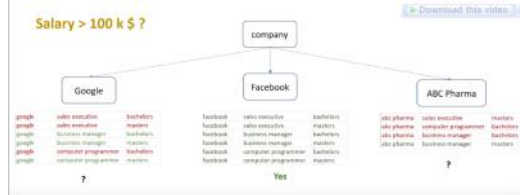
Decision Tree

Saturday, September 16, 2023 10:29 AM

Suppose we have a dataset in which we have to analyse if a persons salary is greater than 100 grand

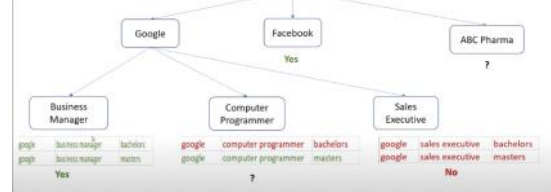
We will be based on company, position and qualification
Even in our minds, we will try to make decisions and build up reasoning based on all these factors that affect salary

Our mind would work something like this:

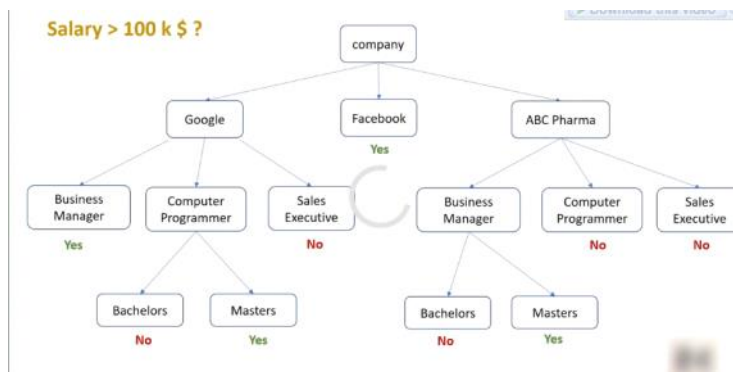


Here we see that FACEBOOK = 100 GRAND = RICH
But there are mixed points in google and abc

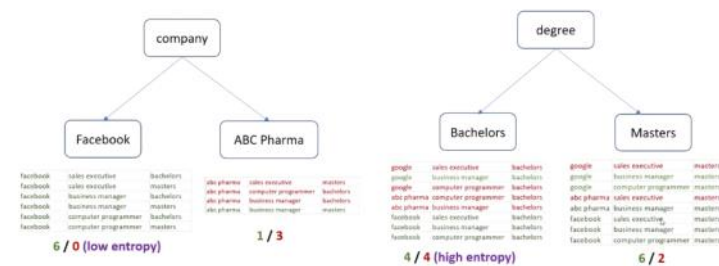
Now for google we ask further questions that may give us a yes no answer
Lets try looking at the position of the job:



So we see that a BM has 100 grand
And a computer programmer may or may not, depends on the qualification (sad)



The order of selection of these attributes will affect the performance of the attributes
What if we started with qualification rather than the company
We might have got a subset with a far more entropy (un ordered ness)



Low entropy = high info gain at every split

Gini Impurity

