Regularization in Machine Learning

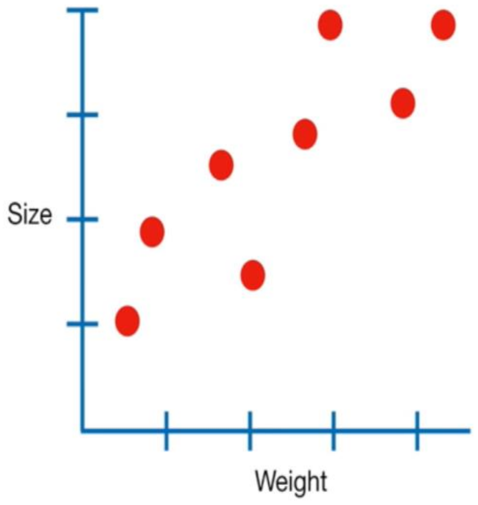
Regularization is set of strategies or techniques which are used to reduce the test error at the expense of training error.

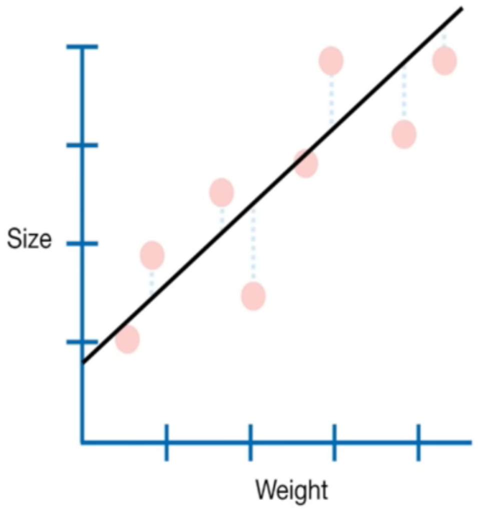
Rigde Regression :

In this topic we will the how regularization helps in solving various machine leaning models which if not used results in overfitting .

Before dive into the topic you must be familiar with the linear models .

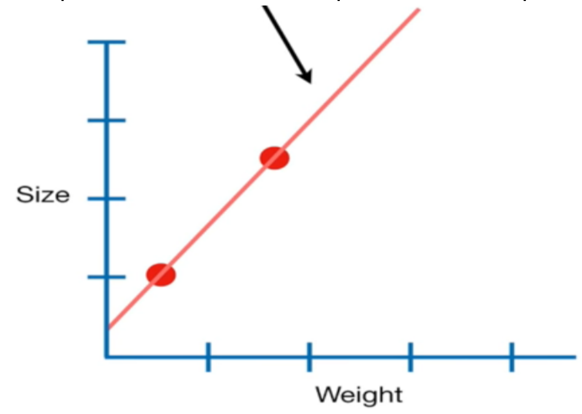
Lets take the case of Ridge Regression .In Ridge regression we have find a line (in case of 2 dimentional data Lets take here.) which has the least sum of squares of distances from the line.



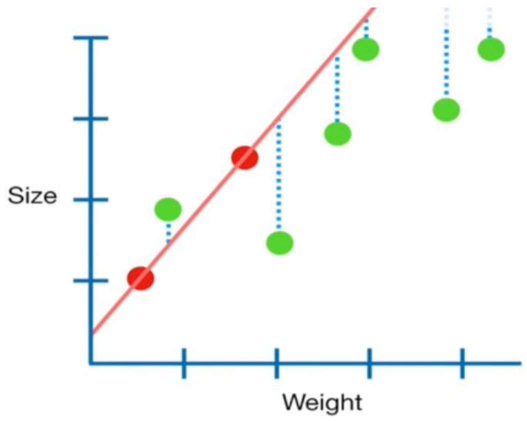


Here we have a line which will give us the values if we plug in the required values of size and weight.

If we have large of values then we can be fairly confident that Least Squares line actually reflects the relation between the size and weight .But what if we have only less data points Lets take only two data points,Since line overlaps the the data points minimum sum of residuals is 0.



The green dots representing the test data we can see although we have sum = 0 for training data we have higher least square sum for test data this means that the new line has high variance.



We say new line is over fit to the training data. In Ridge regression we add a small amount of bias to loss term which we are minimizing so the new line does not fully fit on the training data which further can see the huge drop in the variance of test data.

For p dimensional data points ***β*** is the coefficient estimates for different variables.

Y ≈ β0 + β1X1 + β2X2 + …+ βpXp

$$\mathrm{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

Loss Function :

Coefficients are estimated by minimizing the function

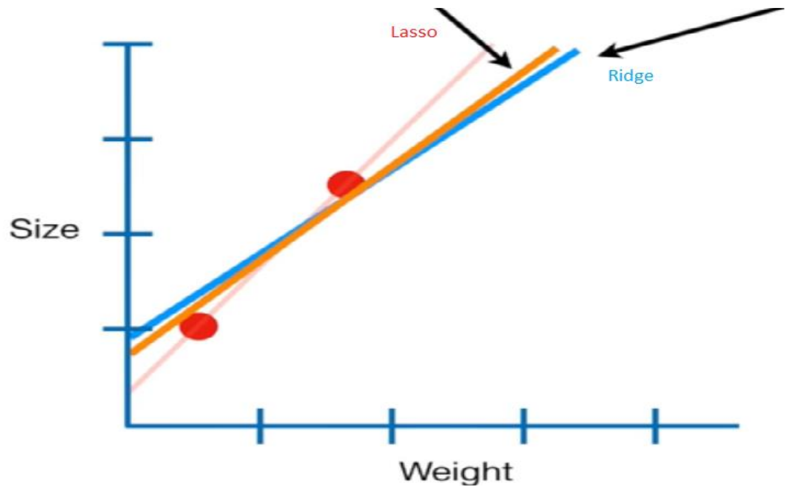
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \mathrm{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

Here Lambda is the critical parameter that has to tuned carefully so that coefficient prouduced are not very high nor very small , cross validation can be thought as Lambda estimation . The coefficients estimates produced by this method is also called L2 – Norm.

Lasso Regression :

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \;=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|.$$

In Lasso regression we use modulo $\boldsymbol{\beta}$ (It takes the absolute value) instead of sum of squares which means by little bias we are reducing the variance .It is clear that this version penalizes the whole term only when the coefficients are higher. The big difference between the Ridge and Lasso regression is that former can shrink the coefficients asymptotically close to zero while Lasso regression can shrink the slope all the way to zero.



Let’s take an example where we are predicting the size which depends on in following ways-

Size = y-intercept + slope* Weight + slope2 * High_Fat + slope3 * astrological Sign + Slope4 * AirSpeed of Swallow

We know size mostly dependent on the weight and high fat but very less or negligible dependent on Astrological Sign and Air Speed of Swallow. These two variable are worst way to predict size,

In Ridge Regression if we increase Lambda slope3 and slope4 can shrink a lot but they never be 0.

In contrast to Lasso regression if we increase the Lambda these term may go away and we left with size only dependent on weight and high fat.L1 – Norm encourages the values to be negative resulting in more sparsity .

Since Lasso regression let us to get rid of silly stuffs we can say it is a little better in reducing the variance than Ridge regression.

Regularization in DeepLearning

All types of neural network models whether Multilayer perceptron, Convolution Neural Network , Recurrent Neural networks which are very complex and they usually tries learn whole order of training data which includes the noise too so ultimately they will become prone to over fitting .

There are different optimization technique in deep learning to generalize the model better helps in perform well on unseen data.

L2 & L1 Regularization:

Larger weights are the sign of more complex neural networks which has over fit the training data.L1 and L2 norm is added as an optimization part to keep the weights smaller. Larger weights means it has learnt even smaller specifics of the training data which is not encouraged also it makes the model unstable producing the terrifying output.

In classical machine learning models we have seen L1 and L2 penalizes the coefficient but here in Neural network using L1/L2 regularizer means we have to penalize the weight matrices of each node.

Regularization term is added to the cost function (say, binary cross entropy for the classification or MSE for regression problem)

Cost function = Loss (say, binary cross entropy) + Regularization term

Due to the addition of regularization term it supresses the weights by going higher and makes the model with smaller weights which ultimately leads to the simpler model. The terms L1 and L2 differ by

L2,

$$Cost\,function \;=\; Loss \;+\; \frac{\lambda}{2m} \;*\; \sum \|w\|^2$$

The regularization term here also known as *weight decay* as it forces the weight to decay towards 0 but not 0 , as we have seen above.

In L1 ,

$$Cost\,function \;=\; Loss \;+\; \frac{\lambda}{2m} \;*\; \sum \|w\|$$

Lambda is regularization parameter

m is the number of inputs

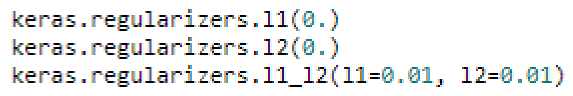
w weight matrix at a layer j

Summation from 1 to n where n is the number of layers

In deep learning neural networks has too many ,millions, parameters which is not necessary or very difficult to know about each of them. Combination of both L1 and L2 tells us which parameter is useless or needed .Both get the each set of separate parameters for both of them . Hyperparameter values can be found using the grid search method. This method can also be applied on classical machine learning models where we too many features to handel . **Elastic-Net** is the fancy name for this which provides practical advantage of trading-off between Lasso and Ridge.

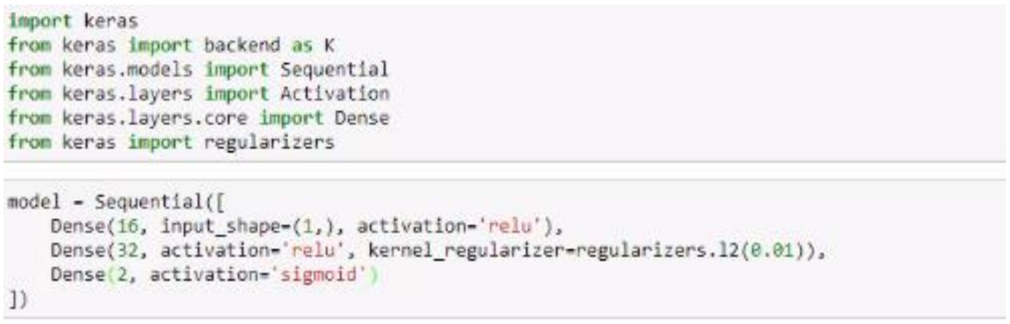
In Keras we have ,

```
from keras import regularizers

model.add(Dense(64, input_dim=64,

                kernel_regularizer=regularizers.l2(0.01)
```



```
keras.regularizers.l1(0.)
keras.regularizers.l2(0.)
keras.regularizers.l1_l2(l1=0.01, l2=0.01)
```

Regularizer as you can see ,needless to say ,it is added for each layers not for the entire model.

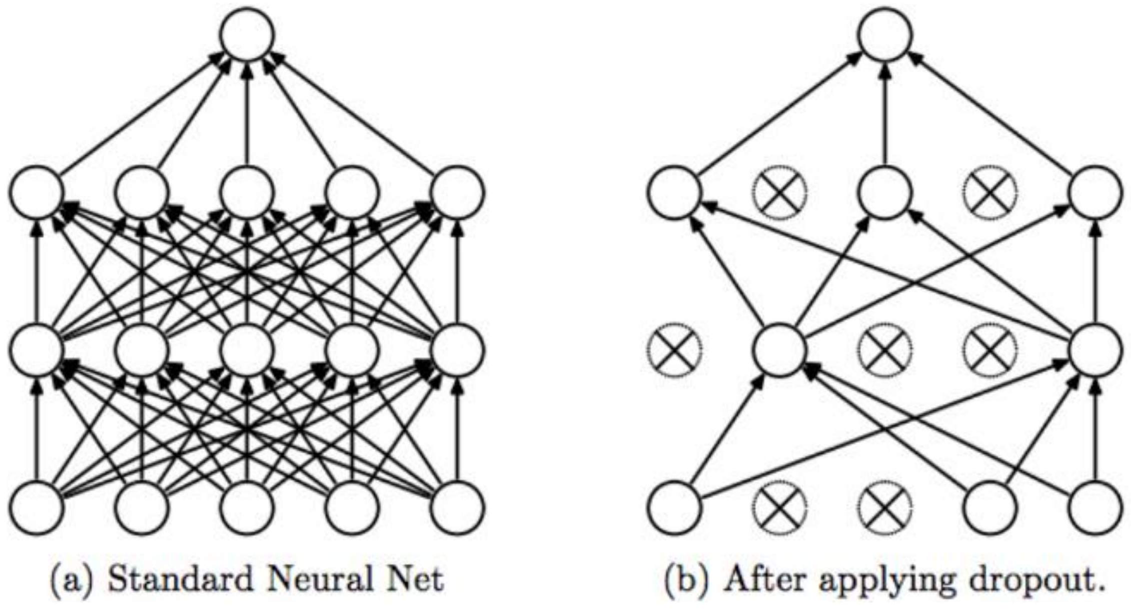


```
import keras
from keras import backend as K
from keras.models import Sequential
from keras.layers import Activation
from keras.layers.core import Dense
from keras import regularizers
```

```
model = Sequential([
    Dense(16, input_shape=(1,), activation='relu'),
    Dense(32, activation='relu', kernel_regularizer=regularizers.l2(0.01)),
    Dense(2, activation='sigmoid')
])
```

Dropouts:

Dropout is one of the most interesting regularization technique to prevent overfitting. As the name suggests it drops out the specified fraction of neurons randomly at each iteration from the neural network model to incorporate the randomness into the model .At each training state individual nodes are dropped out of the total with probability *1-p* or kept with probability *p.* This randomness leads to learn the useful features and better learning of model and generalize much better on the unseen data.



Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Dropping of neurons is only used during the training of model and entire model is used during the inference phase.

Dropout concept is extensively used in computer vision where you have large number of features(pixel values).

In keras ,

keras.layers.Dropout(rate, noise_shape=None, seed=None)

Example of Dropout in MLP

```
from keras.layers import Dense
from keras.layers import Dropout
...
model.add(Dense(32))
model.add(Dropout(0.5))
model.add(Dense(1))
```

Example of Dropout in LSTM

```
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
...
model.add(LSTM(32, dropout=0.5, recurrent_dropout=0.5))
model.add(Dense(1))
```

Example of Dropout in CNN

```
from keras.layers import Dense
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dropout
...
model.add(Conv2D(32, (3,3)))
model.add(Conv2D(32, (3,3)))
model.add(MaxPooling2D())
model.add(Dropout(0.5))
model.add(Dense(1))
```

Credits:

https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

StatQuest with Josh Strammer

https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/

https://machinelearningmastery.com/weight-regularization-to-reduce-overfitting-of-deep-learning-models/