Regularization

Sometimes the machine learning model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted. This problem can be deal with the help of a regularization technique.

Regularization a way to reduce overfitting in linear models. To handle overfitting in linear models, we can reduce the weightage of the coefficients of input variables towards zero.

In simple words, we reduce the magnitude of the features by keeping the same number of features. It works by adding a penalty or complexity term to the cost function such that coefficients are penalized in a way that model doesn't overfits. Hence, the model will be less likely to fit the noise of the training data and will improve the generalization abilities of the model.

Cost function = Loss (rmse/binary cross entropy) + Regularization term

The bigger the penalization the smaller the coefficients.

The L1 regularization or Lasso (Least Absolute Shrinkage and Selection Operator)

The penalty added to the cost function is the sum of the absolute values of weights/coefficients. It adds a penalty for non-zero coefficients. As a result, for high values of λ, many coefficients are exactly zeroed under lasso.

Since L1 regularization will shrink some parameters to zero, some variables will not play any role in the model.

Error(L1)= Error + $+\lambda \sum_{j=0}^{p} |w_j|$

Here, if lambda is zero then you can imagine we get back OLS. However, if lambda is very large then it will add too much weight and it will lead to under-fitting. Having said that it's important how lambda is chosen

The L2 regularization or Ridge

The L2 parameter norm penalty is commonly known as weight decay. The penalty added to the cost function is the sum of the squared values of weights/coefficients.

The L2 regularization will force the parameters to be relatively small but never will shrink to zero.

Error(L2) = $Min(\sum_{i=1}^{n}(y_i - w_i x_i)^2 + p\sum_{i=1}^{n}(w_i)^2)$

The L1/L2 regularization (also called Elastic net)

Elastic-net is a mix of both L1 and L2 regularizations. A penalty is applied to the sum of the absolute values and to the sum of the squared values.

Error(L1,L2) = $\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$

In addition to setting and choosing a lambda value elastic net also allows us to tune the alpha parameter where $\boldsymbol{\alpha}$ = 0 corresponds to ridge and $\boldsymbol{\alpha}$ = 1 to lasso. Simply put, if you plug in 0 for alpha, the penalty function reduces to the L1 (ridge) term and if we set alpha to 1 we get the L2

(lasso) term. Therefore, we can choose an alpha value between 0 and 1 to optimize the elastic net. Effectively this will shrink some coefficients and set some to 0 for sparse selection.

Lasso, Ridge and Elastic net are only applicable for linear models and not for ensemble models. Pruning, Early stopping are ways to achieve regularisation and reduce overfitting in ensemble models. We have drop out for implementing regularisation in neural networks

Dropout

A simple and powerful regularization technique for neural networks and deep learning models is dropout. Dropout is a technique where randomly selected neurons are ignored/dropped-out randomly during training. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. This in turn results in a network that is capable of better generalization and is less likely to over fit the training data. For example, if use chose the dropout value to be 0.20, then there is 20 % chance of a neuron in each layer to be dead and not play its role in the neutral network. If neuron is dead, it won't part any role but if is active, it is fully effective.