ASSIGNMENT – 39

**MACHINE LEARNING**

**Q13 and Q15 are subjective answer type questions, Answer them briefly.**

**13. Explain the term regularization?**

Regularization means to make things regular or acceptable. This is exactly why we use it for applied machine learning. In the context of machine learning, regularization is the process which regularizes or shrinks the coefficients towards zero. In simple words, regularization discourages learning a more complex or flexible model, to prevent overfitting.

The basic idea is to penalize the complex models i.e. adding a complexity term that would give a bigger loss for complex models. To understand it, let’s consider a simple relation for linear regression. Mathematically, it is stated as below:  
Y≈ W\_0+ W\_1 X\_1+ W\_2 X\_(2 )+⋯+W\_P X\_P

Where Y is the learned relation i.e. the value to be predicted. X\_1,X\_(2 ),〖…,X〗\_P , are the features deciding the value of Y.  
W\_1,W\_(2 ),〖…,W〗\_P , are the weights attached to the features X\_1,X\_(2 ),〖…,X〗\_P respectively.  
W\_0 represents the bias.

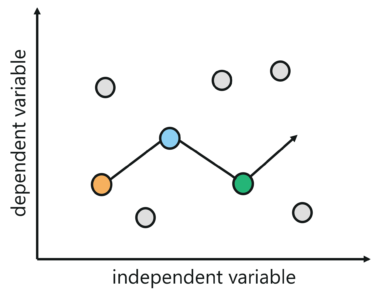
Now, in order to fit a model that accurately predicts the value of Y, we require a loss function and optimized parameters i.e. bias and weights.

The loss function generally used for linear regression is called the residual sum of squares (RSS). According to the above stated linear regression relation, it can be given as:  
RSS= ∑\_(j=1)^m (Y\_i-W\_0-∑\_(i=1)^n W\_i X\_ji )^2

We can also call RSS as the linear regression objective without regularization.

Now, the model will learn by the means of this loss function. Based on our training data, it will adjust the weights (coefficients). If our dataset is noisy, it will face overfitting problems and estimated coefficients won’t generalize on the unseen data.This is where regularization comes into action. It regularizes these learned estimates towards zero by penalizing the magnitude of coefficients.

There are two main regularization techniques, namely Ridge Regression and Lasso Regression. They both differ in the way they assign a penalty to the coefficients.

****

**Ridge Regression (L2 Regularization)**

This regularization technique performs L2 regularization. It modifies the RSS by adding the penalty (shrinkage quantity) equivalent to the square of the magnitude of coefficients.  
∑\_(j=1)^m (Y\_i-W\_0-∑\_(i=1)^n W\_i X\_ji )^2+ α∑\_(i=1)^n W\_i^2=RSS+ α∑\_(i=1)^n W\_i^2

Now, the coefficients are estimated using this modified loss function.

In the above equation, you may have noticed the parameter α (alpha) along with shrinkage quantity. This is called a tuning parameter that decides how much we want to penalize our model. In other terms, tuning parameter balances the amount of emphasis given to minimizing RSS vs minimizing the sum of the square of coefficients.

Let’s see how the value of α alpha affects the estimates produced by ridge regression.

**When** α=0, the penalty term has no effect. It means it returns the residual sum of the square as loss function which we choose initially i.e. we will get the same coefficients as simple linear regression.

**When** α=∞, the ridge regression coefficient will be zero because the modified loss function will ignore the core loss function and minimize coefficients square and eventually end up taking the parameter’s value as 0.

**When** 0<α<∞, for simple linear regression, the ridge regression coefficient will be somewhere between 0 and 1.That’s the reason for selecting a good value of α (alpha) is critical. The coefficient methods produced by ridge regression regularization technique are also known as the L2 norm.

**Lasso Regression (L1 Regularization)**

This regularization technique performs L1 regularization. It modifies the RSS by adding the penalty (shrinkage quantity) equivalent to the sum of the absolute value of coefficients.  
∑\_(j=1)^m (Y\_i-W\_0-∑\_(i=1)^n W\_i X\_ji )^2+ α∑\_(i=1)^n |W\_i |=RSS+ α∑\_(i=1)^n |W\_i |  
Now, the coefficients are estimated using this modified loss function.

Lasso Regression is different from ridge regression as it uses absolute coefficient values for normalization.

As loss function only considers absolute coefficients (weights), the optimization algorithm will penalize high coefficients. This is known as the L1 norm.

Here, α (alpha) is again a tuning parameter, works like that of ridge regression and provides a trade-off between balancing RS magnitude of coefficients.

Like ridge regression, α (alpha) in lasso regression can take various values as follows:

**When** α=0, we will get the same coefficients as simple linear regression.

**When** α=∞, the lasso regression coefficient will be zero.

**When** 0<α<∞, for simple linear regression, the lasso regression coefficient will be somewhere between 0 and 1.

It is appearing very similar to ridge regression, but let’s have a look at both techniques with a different perspective.

Think of ridge regression as solving an equation, where the sum of squares of weights(coefficients) is less than or equal to s. According to this, considering there are 2 parameters in a given problem, the ridge regression is expressed by  
W\_1^2+ W\_2^2≤s

It implies that ridge regression coefficients have the smallest loss function for all point that li within the circle given by the above equation.

Similarly, think of lasso regression as solving an equation, where the sum of modulus of weights(coefficients) is less than or equal to s. According to this, considering there are 2 parameters in a given problem, the lasso regression is expressed by  
|W\_1 |+ |W\_2 |≤s

It implies that ridge regression coefficients have the smallest loss function for all point that li within the diamond given by the above equation.  
The following image describes the above equations:

In this image we can see, Constraint functions (blue area); left one is for lasso whereas the right one is for the ridge, along with contours (green eclipse) for loss function i.e. RSS.

In the above case, for both regression techniques, the coefficient estimates are given by the first point at which contours (an eclipse) contacts the constraint (circle or diamond) region.

The ridge regression coefficient estimates will be exclusively non-zero. Why? Because ridge regression has a circular constraint, having no sharp points, the eclipse will not intersect the constraint on an axis.

On the other hand, the lasso constraint, because of diamond shape, has corners at each of the axes hence the eclipse will often intersect at each of the axes. Due to that, at least one of the coefficients will equal zero.

The above scenario shows that ridge regression will shrink the coefficients very close to 0 but will never make them exactly 0, which means the final model will include all predictors. This is a disadvantage of ridge regression, called model interpretability.

However, lasso regression, when α is sufficiently large, will shrink some of the coefficients estimates to exactly 0. That’s the reason lasso provides sparse solutions.

14. Which particular algorithms are used for regularization?

Regularization is a set of techniques which can help avoid overfitting in neural networks, thereby improving the accuracy of deep learning models when it is fed entirely new data from the problem domain. There are various regularization techniques, some of the most popular ones are — L1, L2, dropout, early stopping, and data augmentation.

The characteristic of a good machine learning model is its ability to generalise well from the training data to any data from the problem domain; this allows it to make good predictions on the data that model has never seen. To define generalisation, it refers to how well the model has learnt the concepts to apply to any data rather than just with the specific data it was trained on during the training process.

On the flip side, if the model is not generalised, a problem of overfitting emerges. In overfitting, the machine learning model works on the training data too well but fails when applied to the testing data. It even picks up the noise and fluctuations in the training data and learns it as a concept. This is where regularization steps in and makes slight changes to the learning algorithm so that the model generalises better. Some of the [regularization](https://analyticsindiamag.com/regularization-in-machine-learning-a-detailed-guide/) techniques are as follows:

**L2 and L1 Regularization**

L2 and L1 are the most common types of regularization. Regularization works on the premise that smaller weights lead to simpler models which in results helps in avoiding overfitting. So to obtain a smaller weight matrix, these techniques add a ‘regularization term’ along with the loss to obtain the cost function.

Cost function = Loss + Regularization term

The difference between L1 and L2 regularization techniques lies in the nature of this regularization term. In general, the addition of this regularization term causes the values of the weight matrices to reduce, leading simpler models.

In L2, we depict cost function [as](https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/)



Here, lambda is the regularization parameter which is the sum of squares of all feature weights. L2 technique forces the weight to reduce but never makes them zero. Also referred to as ridge regularization, this technique performs best when all the input features influence the output, and all the weights are of almost equal size.

In the L1 regularization technique,

L2

Unlike in the case of L2 regularization, where weights are never reduced to zero, in L1 the absolute value of the weights are penalised. This technique is useful when the aim is to compress the model. Also called Lasso regularization, in this technique, insignificant input features are assigned zero weight and useful features with non-zero.

**Dropout**

Another most frequently used regularization technique is [dropout](https://analyticsindiamag.com/everything-you-should-know-about-dropouts-and-batchnormalization-in-cnn/). It essentially means that during the training, randomly selected neurons are turned off or ‘dropped’ out. It means that they are temporarily obstructed from influencing or activating the downward [neuron](https://analyticsindiamag.com/fruit-recognition-using-the-convolutional-neural-network/) in a forward pass, and none of the weights updates is applied on the backward pass.

So if neurons are randomly [dropped](https://analyticsindiamag.com/guide-to-building-a-resnet-model-with-without-dropout/) out of the network during training, the other neurons step in and make the predictions for the missing neurons. This results in independent internal representations being learned by the network, making the network less sensitive to the specific weight of the neurons. Such a network is better generalised and has fewer chances of producing overfitting.

**Early Stopping**

It is a kind of cross-validation strategy where one part of the training set is used as a validation set, and the performance of the model is gauged against this set. So if the performance on this validation set gets worse, the training on the model is immediately stopped.

The main idea behind this technique is that while fitting a neural network on training data, consecutively, the model is evaluated on the unseen data or the validation set after each iteration. So if the performance on this validation set is decreasing or remaining the same for the certain iterations, then the process of model training is stopped.

**Data Augmentation**

The simplest way to reduce overfitting is to increase the data, and this technique helps in doing so.

[Data augmentation](https://analyticsindiamag.com/why-does-image-data-augmentation-work-as-a-regularizer-in-deep-learning/) is a regularization technique, which is used generally when we have images as data sets. It generates additional data artificially from the existing training data by making minor changes such as rotation, flipping, cropping, or blurring a few pixels in the image, and this process generates more and more data. Through this regularization technique, the model variance is reduced, which in turn decreases the regularization error.

15. Explain the term error present in linear regression equation?

There are three error metrics that are commonly used for evaluating and reporting the performance of a regression model; they are:

* Mean Squared Error (MSE).
* Root Mean Squared Error (RMSE).
* Mean Absolute Error (MAE)

There are many other metrics for regression, although these are the most commonly used. we can see the full list of regression metrics supported by the scikit-learn Python machine learning library here:

* [Scikit-Learn API: Regression Metrics](https://scikit-learn.org/stable/modules/classes.html#regression-metrics).

In the next section, let’s take a closer look at each in turn.

## **Metrics for Regression**

In this section, we will take a closer look at the popular metrics for regression models and how to calculate them for your predictive modelling project.

### Mean Squared Error

[Mean Squared Error](https://en.wikipedia.org/wiki/Mean_squared_error), or MSE for short, is a popular error metric for regression problems.

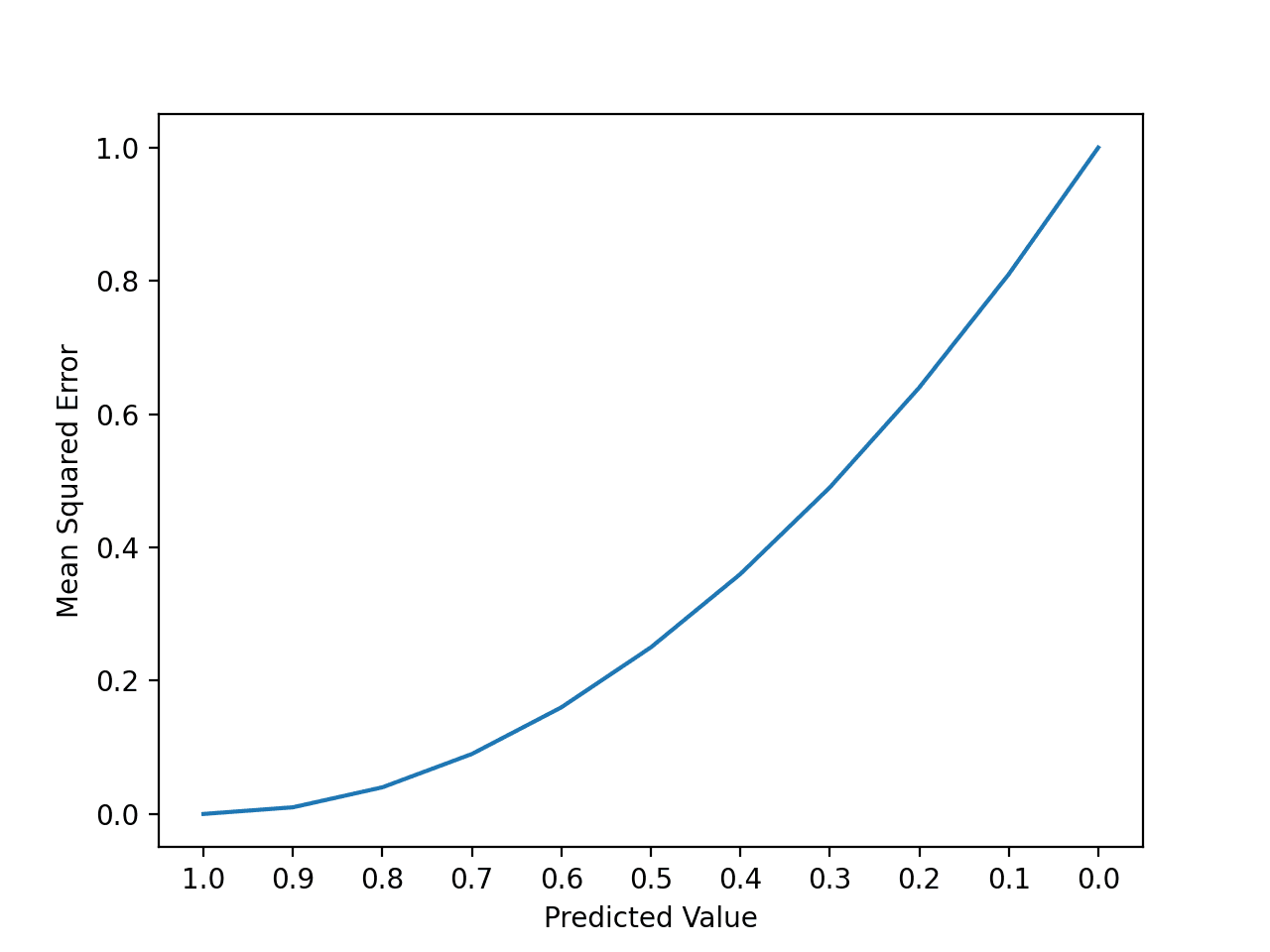
It is also an important loss function for algorithms fit or optimized using the least squares framing of a regression problem. Here “least squares” refers to minimizing the mean squared error between predictions and expected values.

The MSE is calculated as the mean or average of the squared differences between predicted and expected target values in a dataset.

* MSE = 1 / N \* sum for i to N (y\_i – yhat\_i)^2

Where y\_i is the i’th expected value in the dataset and yhat\_i is the i’th predicted value. The difference between these two values is squared, which has the effect of removing the sign, resulting in a positive error value.

The squaring also has the effect of inflating or magnifying large errors. That is, the larger the difference between the predicted and expected values, the larger the resulting squared positive error. This has the effect of “punishing” models more for larger errors when MSE is used as a loss function. It also has the effect of “punishing” models by inflating the average error score when used as a metric.



**Root Mean Squared Error**

The [Root Mean Squared Error](https://en.wikipedia.org/wiki/Root-mean-square_deviation), or RMSE, is an extension of the mean squared error.

Importantly, the square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

For example, if your target variable has the units “*dollars*,” then the RMSE error score will also have the unit “*dollars*” and not “*squared dollars*” like the MSE.

As such, it may be common to use MSE loss to train a regression predictive model, and to use RMSE to evaluate and report its performance.

The RMSE can be calculated as follows:

* RMSE = sqrt(1 / N \* sum for i to N (y\_i – yhat\_i)^2)

Where *Yi* is the i’th expected value in the dataset, *yhat\_i* is the i’th predicted value, and *sqrt()* is the square root function.

We can restate the RMSE in terms of the MSE as:

* RMSE = sqrt (MSE)

Note that the RMSE cannot be calculated as the average of the square root of the mean squared error values. This is a common error made by beginners and is an example of [Jensen’s inequality](https://machinelearningmastery.com/a-gentle-introduction-to-jensens-inequality/).

You may recall that the square root is the inverse of the square operation. MSE uses the square operation to remove the sign of each error value and to punish large errors. The square root reverses this operation, although it ensures that the result remains positive.

The root mean squared error between your expected and predicted values can be calculated using the [mean\_squared\_error () function](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_squared_error.html) from the scikit-learn library.

By default, the function calculates the MSE, but we can configure it to calculate the square root of the MSE by setting the “squared” argument to False.

**Mean Absolute Error**

[Mean Absolute Error](https://en.wikipedia.org/wiki/Mean_absolute_error), or MAE, is a popular metric because, like RMSE, the units of the error score match the units of the target value that is being predicted.

Unlike the RMSE, the changes in MAE are linear and therefore intuitive.

That is, MSE and RMSE punish larger errors more than smaller errors, inflating or magnifying the mean error score. This is due to the square of the error value. The MAE does not give more or less weight to different types of errors and instead the scores increase linearly with increases in error.

As its name suggests, the MAE score is calculated as the average of the absolute error values. Absolute or *abs ()* is a mathematical function that simply makes a number positive. Therefore, the difference between an expected and predicted value may be positive or negative and is forced to be positive when calculating the MAE.

The MAE can be calculated as follows:

* MAE = 1 / N \* sum for i to N abs(y\_i – yhat\_i)

Where *y\_i* is the i’th expected value in the dataset, *yhat\_i* is the i’th predicted value and *abs()* is the absolute function.

