**Unit-2 Chapter-4 Lecture-2.1.2**

**Mapped with CO1,3,4,5**

**Cross-Validation and Regularization**

**CROSS VALIDATION**

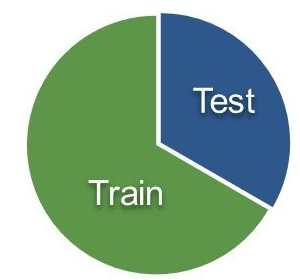
Once we are done with training our model, we just can’t assume that it is going to work well on data that it has not seen before. In other words, we cant be sure that the model will have the desired accuracy and variance in the production environment. We need some kind of assurance of the accuracy of the predictions that our model is putting out. For this, we need to validate our model. This process of deciding whether the numerical results quantifying hypothesized relationships between variables are acceptable as descriptions of the data is known as validation.

To evaluate the performance of any machine learning model we need to test it on some unseen data. Based on the model's performance on unseen data we can say whether our model is Under-fitting/Over-fitting/Well generalized. Cross-validation (CV) is one of the techniques used to test the effectiveness of machine learning models, it is also a resampling procedure used to evaluate a model if we have limited data. To perform CV we need to keep aside a sample/portion of the data on which is do not use to train the model, later us this sample for testing/validating. There are many methods

Below are the few common techniques used for CV.

1.    **Train Test Split approach**.

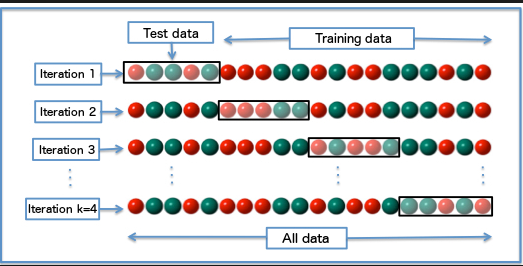
In this approach, we randomly split the complete data into training and test sets. Then Perform the model training on the training set and use the test set for validation purpose, ideally split the data into 70:30 or 80:20. With this approach, there is a possibility of high bias if we have limited data because we would miss some information about the data which we have not used for training. If our data is huge and our test sample and train sample has the same distribution then this approach is acceptable.



**Fig 6.1- Train test split**

**2. K-Folds Cross-Validation**:

K-Fold is popular and easy to understand, it generally results in a less biased model compare to other methods. Because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one of the best approaches if we have limited input data. This method follows the below steps.



**Fig 6.2- K-fold CV**

1.    Split the entire data randomly into k folds (value of k shouldn’t be too small or too high, ideally, we choose 5 to 10 depending on the data size). The higher value of K leads to a less biased model (but large variance might lead to overfitting), whereas the lower value of K is similar to the train-test split approach we saw before.

2.    Then fit the model using the K — 1 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.

3.    Repeat this process until every K-fold serves as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

If the estimator (model) is a classifier and ‘y’(target variable) is either binary/multiclass, then the ‘**Stratified fold**’ technique is used by default. In all other cases, the ‘**K\_Fold**’ technique is used as a default to split and train the model.

Similar to K\_Fold cross-validator, StratifiedKfold returns stratified folds, i.e while making the folds it maintains the percentage of samples for each class in every fold. So that model gets equally distributed data for training/test folds.

**REGULARIZATION**

**What is Regularization?**

In general, 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.

**How Does Regularization Work?**

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 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

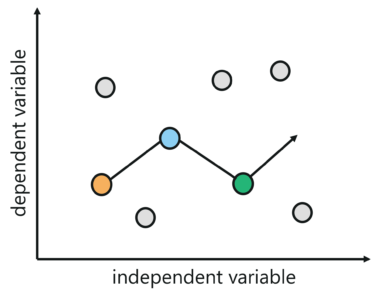
We can also call RSS as the linear regression objective without regularization.

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.

**Regularization Techniques**

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.



**Fig 6.3-Regularization**

**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, the 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 the 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 tradeoff 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 points 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 points 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.

**Book Reading and Video Material**

* Understanding Machine Learning: From Theory to Algorithms by Shai Shalev-Shwartz and Shai Ben-David-Cambridge University Press 2014 [Download](https://www.cse.huji.ac.il/~shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf) Buy at Amazon
* Introduction to Machine Learning – the Wikipedia guide [Download](http://datascienceassn.org/sites/default/files/Introduction%20to%20Machine%20Learning.pdf)
* [Weblink (towardsdatascience)](https://towardsdatascience.com/regularization-an-important-concept-in-machine-learning-5891628907ea)
* [Online Reading Material-1 (Research gate)](https://www.researchgate.net/publication/324701535_Cross-Validation)
* [Online Reading Material-2 (Standford)](http://cs229.stanford.edu/notes/cs229-notes5.pdf)
* [NPTEL Video](https://www.youtube.com/watch?v=nYCAH8b5AQ0)