Assignment

Advanced Regression

Madhusudhan Anand (maddymaster@gmail.com)

23rd September, 2019

Question 1

Rahul built a logistic regression model with a training accuracy of 97% and a test accuracy of 48%. What could be the reason for the gap between the test and train accuracies, and how can this problem be solved?

Answer

Rahul's Logistic Regression model has high training accuracy of 97% and test accuracy of 48% and that is a significant gap. Often, it is mistaken that if a model performs well on the training data, it will produce good results on test data as well. Very often, that is not the case.Accuracy is defined has number of correct predictions divided by total number of predictions. So in case of Rahul, his it learned rules specifically for the train set, those rules do not generalize well beyond the train set and lead to a big gap and this condition is often related to as Overfitting.

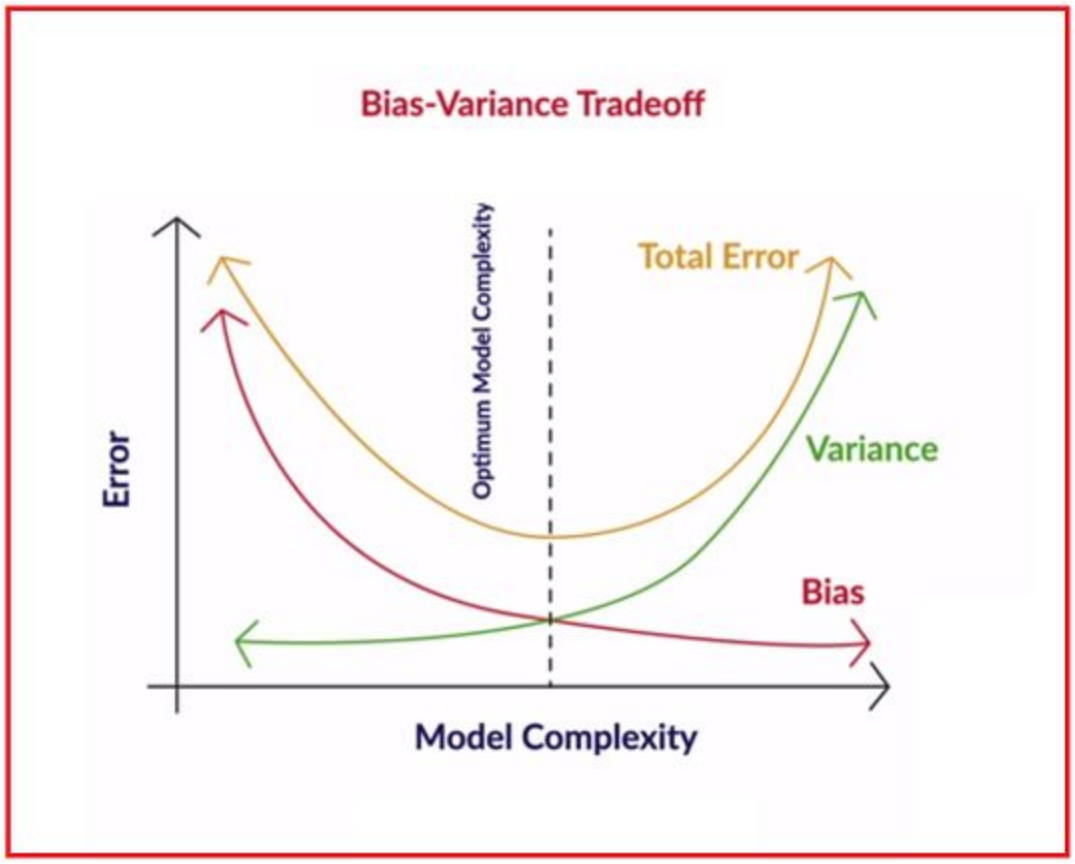
From the upgrad course, this is the definition of overfitting - Overfitting is a phenomenon where a model becomes too specific to the data it is trained on and fails to generalise to other unseen data points in the larger domain. A model that has become too specific to a training dataset has actually 'learnt' not just the hidden patterns in the data but also the noise and the inconsistencies in the data. In a typical case of overfitting, the model performs very well on the training data but fails miserably on the test data.

How to solve this?

Occam's razor is perhaps the most important thumb rule in machine learning, and incredibly 'simple' at the same time. When in dilemma, choose the simpler model.

Look at the bias and variance, and perhaps run cross validation in scenarios like these, adjust the hyperparameters, for example:

Bias Variance Tradeoffs



Having established that we need to find the right balance between model bias and variance, or simplicity and complexity, Rahul needs tools which can reduce or increase the complexity.

Rahul can try Regularization and Hyperparamters as options.

Regularization discourages the model from becoming too complex even if the model explains the (training) observations better. In the last session, you were introduced to this term which is used to find the optimal point between extreme complexity and simplicity. In this context, we will now discuss the use of hyperparameters of a model.

Hyperparameters are parameters that we pass on to the learning algorithm to control the complexity of the final model. Hyper parameter are choices that the algorithm designer makes to 'tune' the behavior of the learning algorithm. The choice of hyperparameters, therefore, has a lot of bearing on the final model produced by the learning algorithm.

Hyperparameters are a part of most learning algorithms which are used for training and regularization. In linear regression, as you will now see, the hyperparameter is used to regularize the model so that it does not become more complex than it should be.

Based on this new understanding Rahul should **be able to "Fine-tune" or regularize** the model so as to keep it optimally complex and reduce this gap and achieve a better result of accuracy

Question 2

List at least four differences in detail between L1 and L2 regularisation in regression.

Answer:

Regression model that uses L1 regularization is referred to as Lasso Regression and the model that uses L2 is called Ridge Regression

Lasso:

1. Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds "absolute value of magnitude" of coefficient as penalty term to the loss function.
2. Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether
3. if lambda is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.
4. Lasso works well for feature selection in case we have a huge number of features.

Ridge:

1. Ridge regression adds "squared magnitude" of coefficient as penalty term to the loss function
2. Ridge doesnt do feature selection

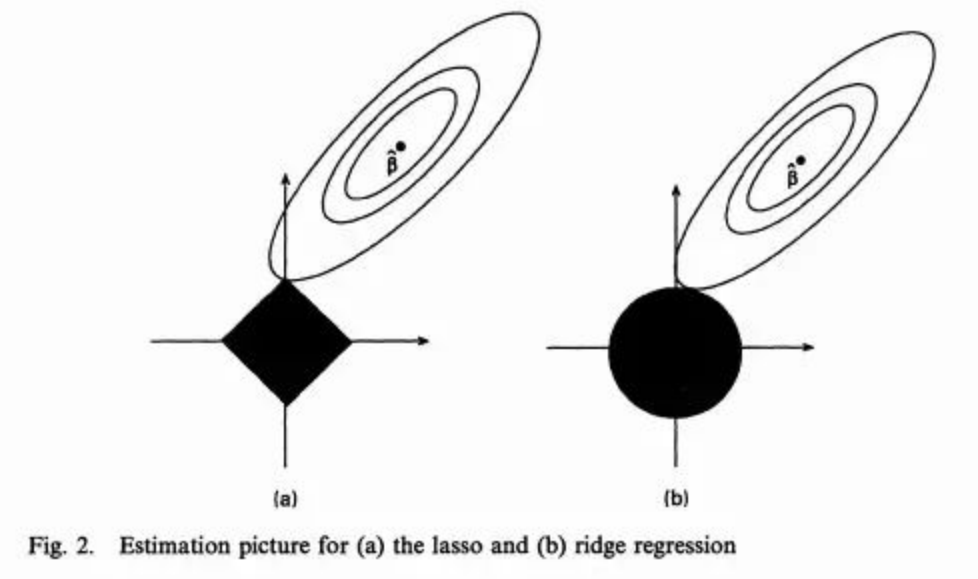
3. 

Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

The feasible point that minimizes the loss is more likely to happen on the coordinates on graph (a) than on graph (b) since graph (a) is more angular. This effect amplifies when your number of coefficients increases, i.e. from 2 to 200.

4. And another difference between L1 and L2 regularization is that L1 can yield sparse models while L2 doesn't. Sparse model is a great property to have when dealing with high-dimensional data, for at least 2 reasons.

- Model compression: increasingly important due to the mobile growth
- Feature selection: it helps to know which features are important and which features are not or redundant.

Question 3

Consider two linear models:

L1: y = 39.76x + 32.648628

And

L2: y = 43.2x + 19.8

Given the fact that both the models perform equally well on the test data set, which one would you prefer and why?

Answer: **I would use L2 because it just looks a lot simpler. We need to use a simpler model wherever possible**

4 unique points about using a simpler model where ever possible:

1. A simpler model is usually more generic than a complex model. This becomes important because generic models are bound to perform better on unseen datasets.
2. A simpler model requires less training data points. This becomes extremely important because in many cases one has to work with limited data points.
3. A simple model is more robust and does not change significantly if the training data points undergo small changes.
4. A simple model may make more errors in the training phase but it is bound to outperform complex models when it sees new data. This happens because of overfitting.

Question 4

How can you make sure that a model is robust and generalisable? What are the implications of the same for the accuracy of the model and why?

Answer: 1. Choosing simpler models at times is a good idea.

2. Simpler models require fewer training samples for effective training than the more complex ones and are consequently easier to train. In machine learning jargon, the sample complexity is lower for simpler models.

3. Simpler models are more robust — they are not as sensitive to the specifics of the training data set as their more complex counterparts are. Clearly we are learning a 'concept' using a model and not really the training data itself. So ideally the model must be immune to the specifics of the

training data provided and rather somehow pick out the essential characteristics of the phenomenon that is invariant across any training data set for the 6 PGDMLAI-Lecture Notes Model Selection problem. So it is generally better for a model to be not too sensitive to the specifics of the data set on which it has been trained. Complex models tend to change wildly with changes in the training data set. Again using the machine learning jargon simple models have low variance, high bias and complex models have low bias, high variance. Here 'variance' refers to the variance in the model and 'bias' is the deviation from the expected, ideal behaviour. This phenomenon is often referred to as the bias-variance tradeoff.

4. Simpler models make more errors in the training set — that's the price one pays for greater predictability. Complex models lead to overfitting — they work very well for the training samples, fail miserably when applied to other test samples.

Another thing we could do is bias-variance tradeoff

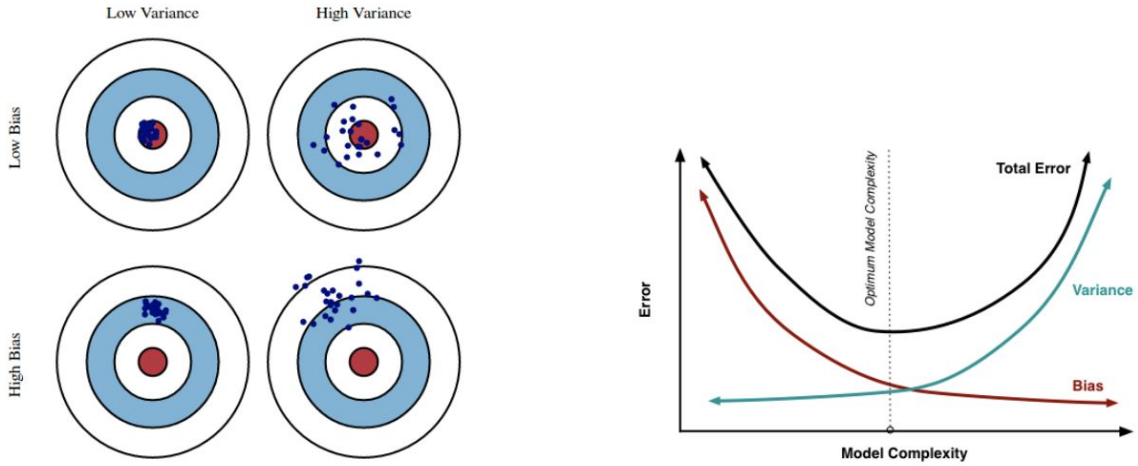


Figure 2: (Left) Illustration of Bias and Variance; (Right) Bias-Variance Tradeoff

However when data limited, which is often the case, notice that we are dealing with a fundamental dilemma — we need the training set to be as large as possible (small training set leaves us with the danger of missing out on crucial pieces of information required for the learning) yet we need a way to estimate the reliability of the model in test scenarios not covered by the training data. In both the methodologies — Hold-Out Strategy and Cross Validation — the basic idea is the same: to keep aside some data that will not in any way influence the model building. The part of the data that is kept aside is then used as a 'proxy' for the unknown (as far as the model we have built is concerned) test data on which we want to estimate the performance of the model. We also look at hyperparameters in this context. Refer Figure 3. A

typical Machine Learning scenario is where we have a data source (possibly implicit) and an underlying system that we are trying to mimic. For example the data source could be an email repository or an email server. The 'system' is a human gate-keeper who is trying to segregate spam emails from the rest. The machine learning model we are trying to build is essentially trying to mimic a human 'system' of discriminating between spam emails and the rest. There is a Learning Algorithm that takes samples from the data source (this will constitute the training data) and the expected responses from the 'system' and comes up with a model that behaves a lot like the system we are trying to mimic. The model will often be used to predict what the system would have presumably

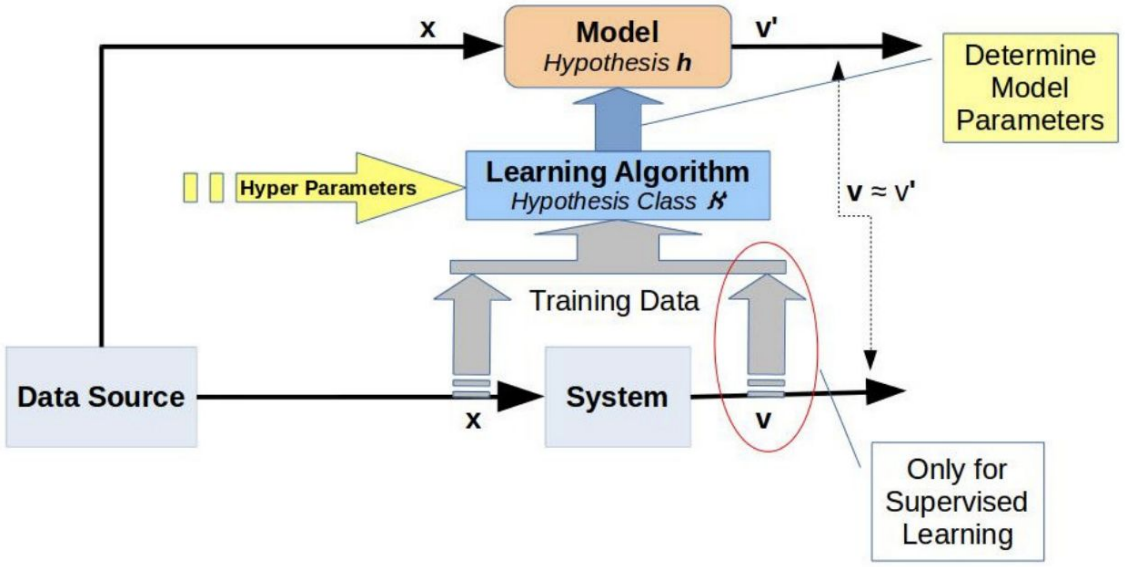


Figure 3: Machine Learning — A Broad Framework

Question 5

You have determined the optimal value of lambda for ridge and lasso regression during the assignment. Now, which one will you choose to apply and why?

Ridge regression adds "squared magnitude" of coefficient as penalty term to the loss function. Here the highlighted part represents L2 regularization element.

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

Cost function

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. This technique works very well to avoid over-fitting issue.

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds "absolute value of magnitude" of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Cost function

Again, if lambda is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

The key difference between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for feature selection in case we have a huge number of features.

Traditional methods like cross-validation, stepwise regression to handle overfitting and perform feature selection work well with a small set of features but these techniques are a great alternative when we are dealing with a large set of features.