# Day_5_Links

1. Simple Linear Regression Review: https://youtu.be/CtKeHnfK5uA
2. Read:
    a. DSFS (Data Science from Scratch), Ch. 15, Multiple Regression
    b. Read MLPR (Machine Learning Pocket Reference, Ch. 14, Regression: handson-ml2/04_training_linear_models.ipynb at master · ageron/handson-ml2 · GitHub
    c. Read HOML (Hands on Machine Learning), Ch. 4, Training Models: handson-ml2/04_training_linear_models.ipynb at master · ageron/handson-ml2 · GitHub
3. Multiple Linear Regression:
    a. DAT4/08_linear_regression.ipynb at master · justmarkham/DAT4
    b. Linear Regression in Python with Scikit-Learn
    c. In Depth: Linear Regression | Python Data Science Handbook
    d. Linear Regression — statsmodels
4. Read:
    a. MLPR, Ch. 15, Metrics and Regression Evaluation and Ch. 16, Explaining Regression Models. ch 15 code, ch 16 code
5. Poisson Regression:
    a. Poisson Regression / Regression of Counts: Definition
    b. An Illustrated Guide to the Poisson Regression Model
    c. Maximum Likelihood Estimation of Custom Models in Python with StatsModels
6. Negative Binomial Regression:
    a. Negative Binomial Regression: A Step by Step Guide | by Sachin Date
    b. Negative Binomial Example
7. Polynomial Regression:
    a. Polynomial Regression. This is my third blog in the Machine… | by Animesh Agarwal

# Discussion 1:

**Question:**
You have constructed a linear regression model and have discovered that the model under fits the training data. Is the model exhibiting high bias or high variance? What steps can we take to improve the model?

**Solution 1:**

- Data: We could investigate whether it is possible or not to obtain or generate more data if that is feasible and has proven to be not very costly be it computation or otherwise with a promise of high reward, then it would be beneficial to gather more data.

- Adjust hyperparameters(learning rate, number of epochs in GD)

A) We could increase the number of epochs because maybe it is simply not converging as fast as it should be (assuming that the cost function is convex) so in this case, we could increase the number of epoch and observe the difference.

B) we could also attempt to change the learning rate as it could be overshooting the global minimum or simply so slow it needs to be increased to converge faster.

- check the features: Re-examine our features and look into whether or not our features were actually predictive or not of our response variable, it could be the case that we have accidentally removed a valuable explanatory variable. We could even attempt to create more explanatory variables.

- Complexity: We could attempt to further increase the level of complexity as in some cases linear regression could be to simple of a model and needs something a little more complex (such as polynomial regression) depending on the data, of course, so we would first examine the data and then make an educated decision on what are the possible replacements.

**Solution 2:**

A model will be underfit is the bias is high, it will be overfit if the variance is high. If we have a model that underfits data then it is because we have not added enough complexity to it. For instance, heteroskedastic data will not fit a regular linear regression line, it may be necessary to have a d-dimensional version with multiple variables or use a non-linear line to represent the data. Underfitting occurs when there is still room for improvement on the test data. If the model is not powerful enough (as mentioned), is over-regularized (and there are many techniques to do this as mentioned in the  reading), or has simply not been trained long enough. This means the network has not learned the relevant patterns in the training data. It could also be that we sampled the data too sparsely, that we didn't use enough data in our training and this shows up when we try to cross-validate it, or when we try to simplify our assumption too much and take out variables that we shouldn't.

# Discussion 2:

**Question:**

You have constructed a polynomial regression model and have discovered that the model overfits the training data. Is the model exhibiting high bias or high variance? What steps can we take to mitigate the overfitting we are observing?

**Solution 1:**

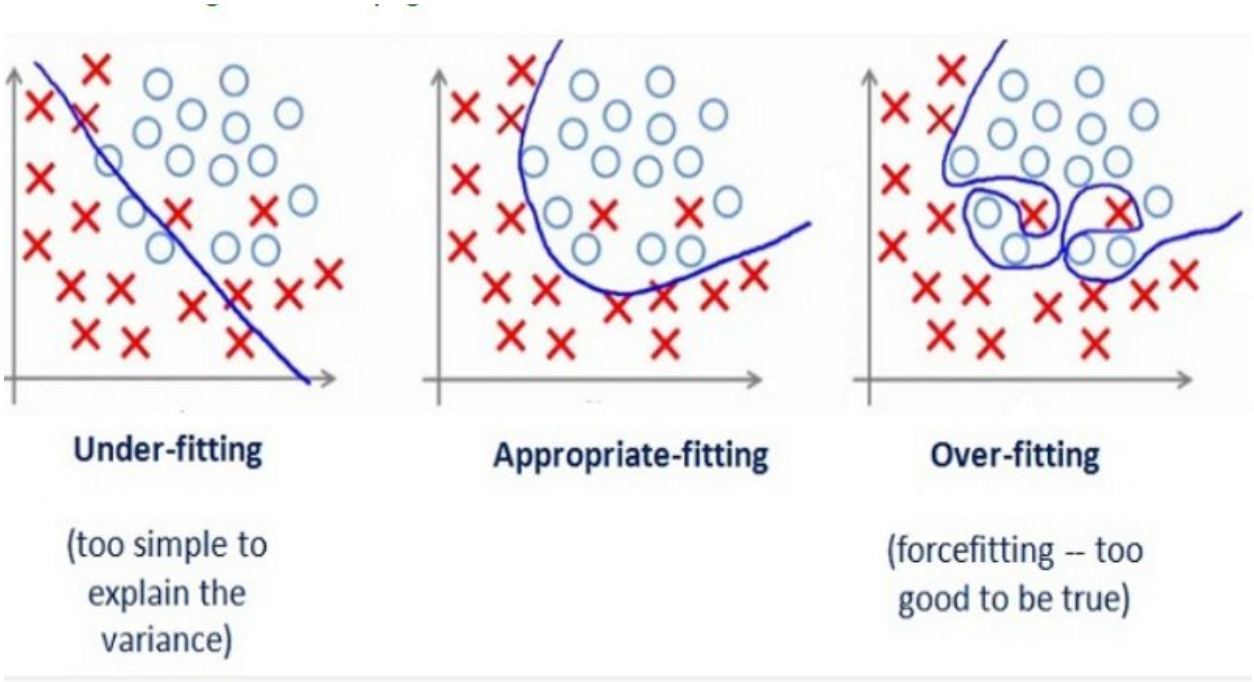
The **bias** is commonly defined as the difference between the expected value of the estimator and the parameter that we want to estimate: **Bias**=E[$\hat{\theta}$]$-\theta$.

**Bias** is an error due to erroneous or overly simplistic assumptions in the learning algorithm you are using. This can lead to the model underfitting your data, making it hard for it to have high predictive accuracy and for you to generalize your knowledge from the training set to the test set.

**Variance** is the measure of dispersion in a **data** set. In other words, it measures how spread out a **data** set is. It is calculated by first finding the deviation of each element in the **data** set from the **mean**, and then by squaring it. **Variance** is the average of all squared deviations.

**Variance** is error due to too much complexity in the learning algorithm you are using. This leads to the algorithm being highly sensitive to high degrees of variation in your training data, which can lead your model to overfit the data. You will be carrying too much noise from your training data for your model to be very useful for your test data.

Therefore, if you have constructed a polynomial regression model and have discovered that the model overfits the training dat, the model is exhibiting **high variance**.




2. What steps can we take to mitigate the overfitting we are observing?

1. ***Build A More simple Model***

The first and simplest solution to an overfitting problem is to train a more simple model to fix the problem. And, get more data in. and regularization.

2. ***Cross Validation***

In cross-validation, all the available or chosen data is not used in training the model. There are usually three folds that help in performing the cross-validation method- the training data, test data, and validation dataset. You can use a combination of Training and Test data alone, or use all three data folds.

There are many ways to work with these folds, and The Training data is usually 60% of the total dataset, the test dataset will be 20% and, the validation data set comprises of the remaining 20%.

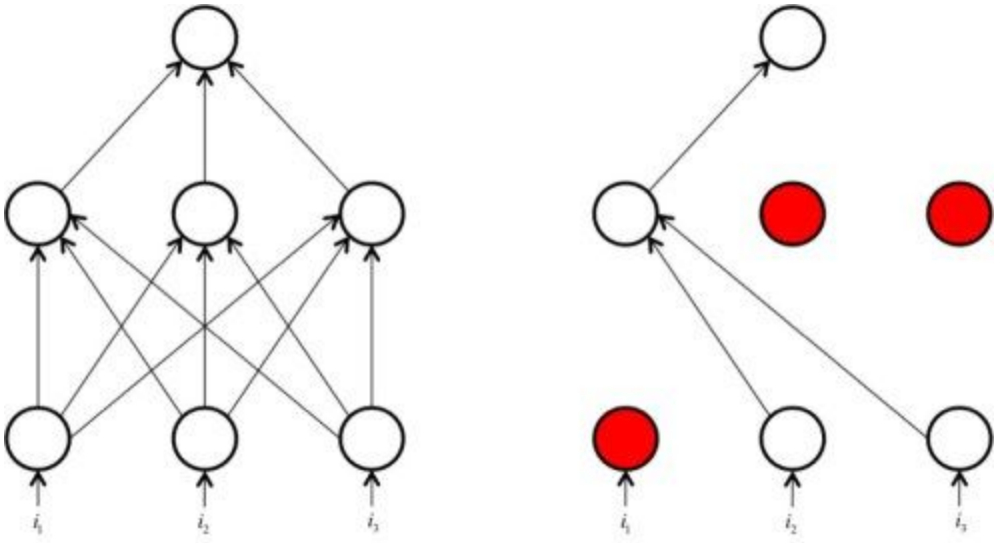
The quality of the trained model is tested by first training the model by using just the training data, and then compare that model with the model that is trained with the test data. In this manner, we can identify which data points bring about a better prediction.

3. ***Dropout***:

The drop-out method is used when working with neural networks

(Links to an external site.)

in deep learning. Dropout is a technique that is old and proven to help the accuracy of models. which makes some activations in a layer deactivated (equals 0). We can choose any amount of data from the dataset to create a dropout layer. Usually, this is in the range of 20 or 30 percent. Suppose, if we use 30% dropout, then the activations for a random 30% neurons in that particular layer gets deactivated. The deactivated neurons will not be propagated to the next layer of the network. We do this to avoid overfitting, as more noise will make the model robust.



*(Dropout method: Here, some neurons have been deactivated( red colored, right). Suppose the activation is x, then in dropout it is equated to zero)*

*4.* ***Gradient Noise****:*

This method involves adding gradient noise during the training, a method that proved to have increased the accuracy of a model. Refer to this paper- *Adding Gradient Noise Improves Learning for Very Deep Networks*

(Links to an external site.)

).

Adding noise sampled from Gaussian Distribution:

$$g_t \leftarrow g_t + N(0, \sigma_t^2)$$

5. ***Regularisation***:

Regularization is just yet another popular method of reducing the overfitting phenomenon. Used to resolve a problem of high variance, the technique involves penalising coefficients and weights, to get a higher accuracy for both training data and test data.

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left( t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i|$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left( t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \lambda \sum_{i=1}^{k} w_i^2$$

*(Here, w is a weight value, the red box represents the Regularization Term, and lambda is the Regularization Parameter, which get optimized during the training. The remaining is the loss function which calculates the least of the squares.)*

**Solution 2:**

Polynomial models regularly overfit data. A friend of mine works for KKR and he has people coming to him regularly saying that have found a now polynomial model or EMA or SMA that predicts the market. Often it is just overfitting a model. When the complexity of a situation is so great that the R-square is too high, that the cross-validation shows inconsistencies, but yet the model is terrible on anything but the training data - it is clearly overfit. This could be due to over-regularization, it could be due to keeping to many variables in the model that were chosen just from a training set with targets that were too specific, it could be because the p-values you started with were unrealistic.

One could also use early stopping, as in the gradient descent models, listed in the reading. Early stopping allows a broader model which is often more accurate and less overfit. One might also try to use other models and create an ensemble of models that each take a different perspective, none of them requiring such exact measure that they overfit the data.