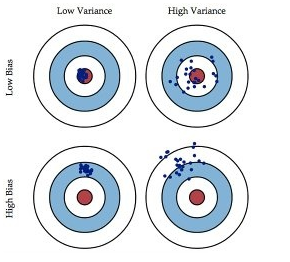
**R2 vs MSE**

**R2, predicted R2, adjusted R2**

**To avoid over-fitting**

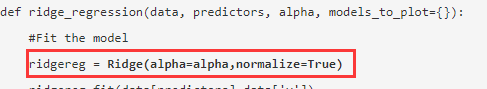
**Variance and bias**



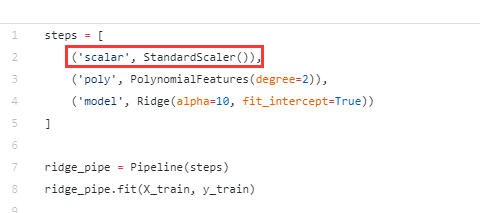
To do : bias and variance in cost function.

**Sample size**

**Normlalize**



Or



**Regularization of Linear Models with SKLearn**

<https://medium.com/coinmonks/regularization-of-linear-models-with-sklearn-f88633a93a2>

**Difference between Ridge and Lasso**

the value of alpha is iterated over a range of values and the one giving higher cross-validation score is chosen. (<https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/>)

1. Key Difference

Ridge: It includes all (or none) of the features in the model. Thus, the major advantage of ridge regression is coefficient shrinkage and reducing model complexity.

Lasso: Along with shrinking coefficients, lasso performs feature selection as well. (Remember the ‘selection‘ in the lasso full-form?) As we observed earlier, some of the coefficients become exactly zero, which is equivalent to the particular feature being excluded from the model.

Traditionally, techniques like stepwise regression were used to perform feature selection and make parsimonious models. But with advancements in Machine Learning, ridge and lasso regression provide very good alternatives as they give much better output, require fewer tuning parameters and can be automated to a large extend.

2. Typical Use Cases

Ridge: It is majorly used to prevent overfitting. Since it includes all the features, it is not very useful in case of exorbitantly high #features, say in millions, as it will pose computational challenges.

Lasso: Since it provides sparse solutions, it is generally the model of choice (or some variant of this concept) for modelling cases where the #features are in millions or more. In such a case, getting a sparse solution is of great computational advantage as the features with zero coefficients can simply be ignored.

Its not hard to see why the stepwise selection techniques become practically very cumbersome to implement in high dimensionality cases. Thus, lasso provides a significant advantage.

3. Presence of Highly Correlated Features

Ridge: It generally works well even in presence of highly correlated features as it will include all of them in the model but the coefficients will be distributed among them depending on the correlation.

Lasso: It arbitrarily selects any one feature among the highly correlated ones and reduced the coefficients of the rest to zero. Also, the chosen variable changes randomly with change in model parameters. This generally doesn’t work that well as compared to ridge regression.

This disadvantage of lasso can be observed in the example we discussed above. Since we used a polynomial regression, the variables were highly correlated. ( Not sure why? Check the output of data.corr() ). Thus, we saw that even small values of alpha were giving significant sparsity (i.e. high #coefficients as zero).

Along with Ridge and Lasso, Elastic Net is another useful techniques which combines both L1 and L2 regularization. It can be used to balance out the pros and cons of ridge and lasso regression. I encourage you to explore it further.

**When do we need neural network**

model selection: cross validation, feature selection and evaluation

**Fit exponential**

https://stackoverflow.com/questions/3433486/how-to-do-exponential-and-logarithmic-curve-fitting-in-python-i-found-only-poly

For fitting y = AeBx, take the logarithm of both side gives log y = log A + Bx. So fit (log y) against x.

Note that fitting (log y) as if it is linear will emphasize small values of y, causing large deviation for large y. This is because polyfit (linear regression) works by minimizing ∑i (ΔY)2 = ∑i (Yi − Ŷi)2. When Yi = log yi, the residues ΔYi = Δ(log yi) ≈ Δyi / |yi|. So even if polyfit makes a very bad decision for large y, the "divide-by-|y|" factor will compensate for it, causing polyfit favors small values.

This could be alleviated by giving each entry a "weight" proportional to y. polyfit supports weighted-least-squares via the w keyword argument.

**Power fitting**

**Choose final model**

**Random forest**

Explanation:

https://www.quora.com/How-does-random-forest-work-for-regression-1

<https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd>

**Random forest regression importance**

<https://medium.com/the-artificial-impostor/feature-importance-measures-for-tree-models-part-i-47f187c1a2c3>

<https://explained.ai/rf-importance/index.html#6>

To visualise the random forest tree.

https://towardsdatascience.com/how-to-visualize-a-decision-tree-from-a-random-forest-in-python-using-scikit-learn-38ad2d75f21c