Regression Trees 03

Dr. Saad

Model Assessment

It is crucial to understand that in statistical learning methods:

- No method dominates all others over all possible data-sets.
- In each problem, a method can outperform all others
- In the same problem, but different a data set, A statistical method can outperform the other. for this reason:
- A data analyst must try different methods
- Assess each method
- Decide on which method they will proceed with based on its **performance**.

Statistical Analysis: is a challenging domain, not just based on statistical learning methods, but on many characteristics:

- **The mind**: we understand problems differently.
- **Computing skills**: How tweak the elements of a specific algorithm to get the best out of it.
- **Expertise**: Selecting the convenient algorithms for certain project.
- **Analytic skills**: Feature engineering, feature selection ...

Model evaluation is based on its predictions on **new (unseen) data**, which we call **test data**.

1. **MEAN SQUARED ERROR**: This measure in commonly used in regression.

$$\textsf{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

* If MSE is calculated on the train data we call it **Train MSE** * If MSE is calculated on the test data we call it **Test MSE**, which is the one we are interested in.

- We would like to have as small the **Test MSE** as possible.
- If we have different algorithms, we would pick up the one with smallest **Test MSE**.

Overfitting: Where the algorithm fits the noise or **the patterns that are happened by random chance** in the training data, it other words **tracks almost every point**. Overfitting is known to have **small Train MSE** and **Large Test MSE**.

Underfitting: The algorithm is not flexible enough to catch the true form of the data. In this case we have **large Train MSE** and **Large Test MSE**.

Generally: we are after a situation where we neither **overfit** nor **underfit** (most of the work is done here trying to find the best algorithm that fits this situation.)

- If a model does not **overfit**, we say it **generalizes well**

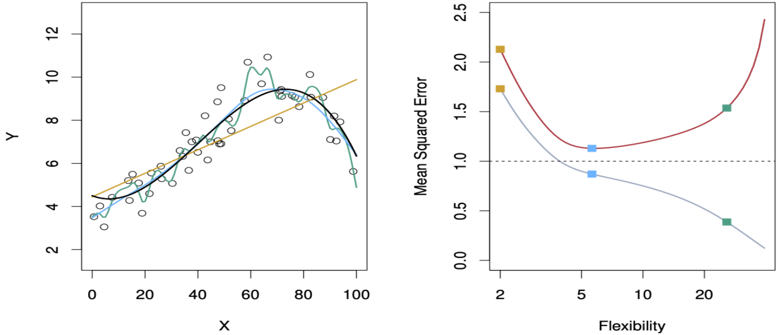


Figure 1: An example showing overfitting and underfitting: Adopted from (An Introduction to Statistical Learning with Applications in R)

Left: **Orange**: linear Regression, **Blue and Green**: smoothing spline. **Black:** the true function form.

right: **Gray curve** Train MSE, **Red curve** Test MSE

1. Linear Regression is inflexible
2. The more flexible the function the more it fits the observations closely. which is too **wigly**
3. Train MSE is always less than Test MSE
4. Train MSE declines as flexibility increases
5. Test MSE declines as flexibility increases but at some point it levels off and then starts to increase (**This is a sign of overfitting**.) This is known as a **U-shape in Test MSE**
6. The wiggly curve has the smallest **Train MSE** but the **worst Test MSE** as well as linear regression.
7. Linear Regression **orange** shows underfitting.
8. The wiggly curve **the green** shows overfitting.
9. The blue curve is the closest one to the true form, in this case it is the best fit.

Result: The best function can be any function like **Logistic Regression, Random Forest, or Neural Network ...**

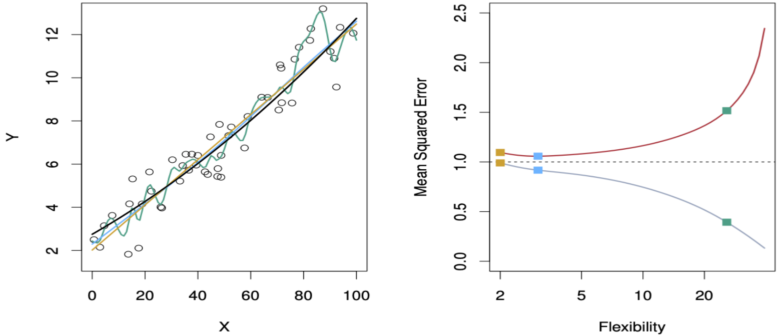


Figure 2: An example showing overfitting and underfitting: Adopted from (An Introduction to Statistical Learning with Applications in R)

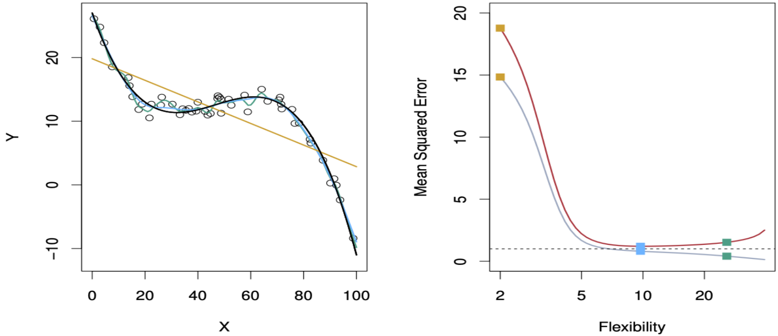


Figure 3: An example showing overfitting and underfitting: Adopted from (An Introduction to Statistical Learning with Applications in R)

The Expected value of Test MSE is composed of the components (variance and bias plus the variance error), sometimes it is called **a generalization error**, the formula is shown below:

$$\mathbf{E}(\mathbf{MSE})^{\mathbf{2}} = \mathbf{Var}\left(\hat{\mathbf{y}}\right) + \left[\mathbf{Bias}\left(\hat{\mathbf{y}}\right)\right]^{\mathbf{2}} + Var\left(\varepsilon\right)$$

$$MSE = (y - \hat{y})$$

- This quantity is the **expected Test Mean Squared Error**.

- It refers to the average test **MSE** from repeatedly estimated function using a large number of training sets, then tested on test sets.
- The previous formula tells us that we need to find an algorithm that minimizes the **test (MSE)**. That can happen when we have both **low variance** and **low bias**
- This formula is always positive, the variance is positive plus a positive value of the bias.

1. **Variance**: refers to the amount by which a **fitted function** would change if we estimated it using a **different dataset**.

- We should not have a function that varies too much between training sets.
- If a statistical method (say Decision Tree) has a **high variance**, fitting the same method or algorithm on a different training set would result in totally different results.
- More flexible algorithms usually have higher variance like **decision trees**.

Example:

1. in figure.1: the green curve has **high variance** because it is too flexible. If we change only few points, the fitted function would change considerably.
2. The linear regression (orange in figure 1) has **low variance**.

2. **Bias**: Refers to the **error** that is introduced by approximating a **real-life** problem. In other words: if the real form of a function is quadratic but we fit a linear regression, thus, we have made an **error**, precisely we would always have **biased results**.

- Generally, more flexible algorithms have **low-bias**

Example:

In figure 3: The true form of the function is non-linear. Thus, no matter how many training observations we are given, it will not be possible to produce an accurate estimate using **Linear Regression**. In this Case, Linear Regression has **high-bias**

In figure 2: The linear regression would be a good fit.

- Flexible algorithms have **high variance** but **low bias**
- The change in **variance** and **bias** can be captured using **MSE** (formula in the previous slide).
- Increasing the flexibility of a method would result in fast decrease in bias more than the increase in **variance**. We see Test MSE declining
- At some point increasing flexibility will have less effect on **bias** but results in a significant increase in **variance**. The Test MSE will start increasing.
- We seek trading between increasing flexibility to a certain point where we have **low variance and low-bias**.

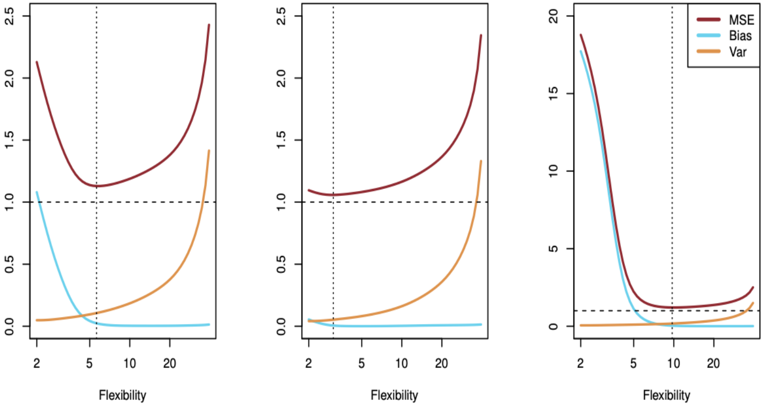


Figure 4: An example showing Bias-Variance TradeOff: Adopted from (An Introduction to Statistical Learning with Applications in R)

Of course, the true form of the model is unknown, for this reason we follow the next steps:

1. Collecting the data needed for the specific problem
2. Splitting the data into train/test (also called **hold-out set**) (sometimes the data is split into three parts Train/Validation and test)
3. Fitting the data on train set
4. evaluate the model on the test set or
5. Using **Cross Validation Technique** Which is very effective in practice.

Remedies:

- In case of overfitting: Decrease the flexibility.
- In case of underfitting: Increase the flexibility, or gather more features or data points ...

Regression Tree **(School Grades project)**:

The response is final_grade in math (numeric: from 0 to 20, output target).

The Goal: Fitting a regression tree based on the next features

- **age** : student's age (numeric: from 15 to 22)
- **address**: student's home address type (binary: 'U' - urban or 'R' - rural)
- **studytime** weekly study time (numeric: 1. $<$2 hours, 2. 2 to 5 hours, 3. 5 to 10 hours, or 4. $>$10 hours)
- **schoolsup** extra educational support (binary: yes or no)
- **famsup** family educational support (binary: yes or no)
- **paid** extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)

Parameter: Can be estimated from the data, like linear regression parameters.

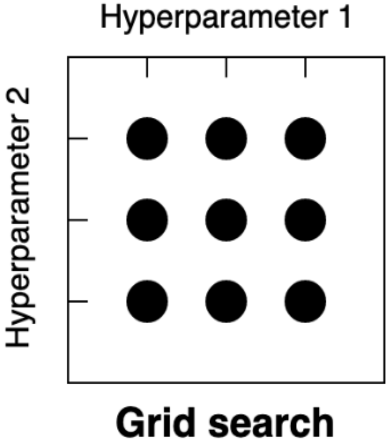
In more detail: **Parameters** are components of the final model that are learned through the modeling process. Crucially, you do not set these. You cannot set these. The algorithm discovers them through undertaking its steps.

Hyperparameter: it is defined by the modeler beforehand like minsplit or maxdepth in decision trees.

In more detail: **Hyperparameters** are something that you set before the modeling process begins. You can think of them like the knobs and dials on an old radio. You tune the different dials and buttons and hope that a nice tune comes out. The algorithm does not learn the value of these during the modeling process

Grid: the set of hyperparameter combinations to iterate over during the Grid Search.

Grid Search: An exhaustive technique that searches over all combinations of hyperparameters. In other words, running a model for every cell in the grid.



The goal of a grid search:

is to evaluate a large number of parameter settings, by training models on each combination of hyperparameter values, to find the combination that produces the best model. That is what we call **Model Tunning**

Best model: Is selected based on a statistical metric such as **RMSE**.

Why Even Bother to do hyperparameter searching?:

R functions come with default setting, but they are not always the optimal, just a good start, But training a sequence of models with various hyperparameter settings with the goal of finding the best one is a typical task in any machine learning pipeline.

Steps of Grid Search:

1. Choose the hyperparameter(s) to **tune**
2. Choose a minimum and maximum values for each hyperparameter. (Some guesswork must be done here). Run a small grid, see if the optimum lies at either endpoints, and then expand the grid in that direction.
3. Create a data frame using `expand.grid()` function, or any other convenient function.
4. Train a sequence of models by looping over the grid.
5. Evaluate the model by generating prediction on the validation data set.
6. Select the winner model by a chosen statistical metric, **RMSE** in case of regression.