foundation

(https://datascience.foundation)

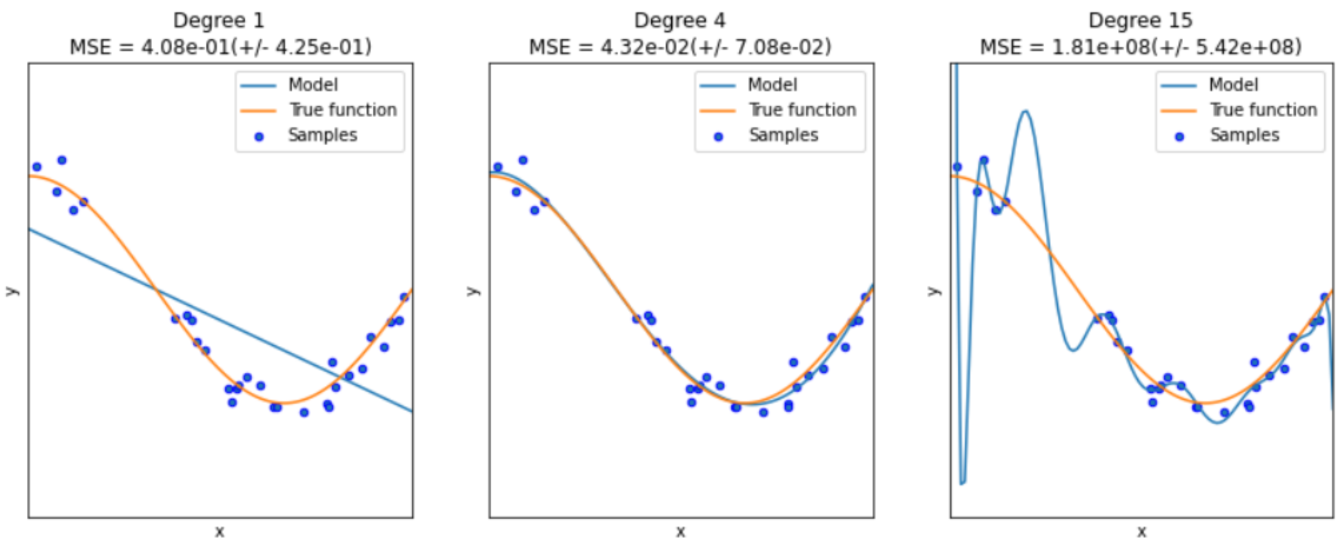
Underfitting and Overfitting in Machine Learning

A DSF Whitepaper   13 June 2020   Mayank Tripathi

Author Profile(https://datascience.foundation/individual/view/1096) Other Articles(https://datascience.foundation/portfolio/mayanktripathi) Follow

Share with your network:

f 1   in 6



Believe it or not, in the real world we will never have a Clean and Perfect Dataset. Each dataset will have some strange or missing parts or imbalanced data. Or we as a Machine Learning Developers will introduce some errors or deficiencies to our model. One of the main reasons for this is that we want our model to be able to describe an underlying pattern. Unfortunately the nature of real life data is that it comes with some level of noise and outliers, and for the most part we want the model to capture the signal in the data and not the noise.

Now, to capture the underlying pattern in data accurately, the model may have to incorporate the noise. The implication is that the derived model fits the modelling data well but does not generalize well enough to other samples that have not been included in the modelling process.

There is terminology to describe how well a machine learning model learns and generalizes to new data, this is overfitting and underfitting. The goal of a good machine learning model is to generalize well from the training dataset to any dataset from the problem domain. This allows us to make predictions based on a dataset the model has never seen.

In statistics a fit is referred to as, how close your model is to the target class / function / value. Overfitting and underfitting are the two biggest causes of poor performance of machine learning algorithms or models.

Let's understand what is Best Fit, Overfitting and Underfitting? Followed by some code.

What is Overfitting & Underfitting?

Overfitting refers to the scenario where a machine learning model can't generalize or fit well on unseen dataset. A clear sign of machine learning overfitting is if its error on the testing or validation dataset is much greater than the error on training dataset.

Overfitting is a term used in statistics that refers to a modeling error that occurs when a function corresponds too closely to a dataset. As a result, overfitting may fail to fit additional data, and this may affect the accuracy of predicting future observations.

Overfitting happens when a model learns the detail and noise in the training dataset to the extent that it negatively impacts the performance of the model on a new dataset. This means that the noise or random fluctuations in the training dataset is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new datasets and negatively impact the model's ability to generalize.

The opposite of overfitting is underfitting.

Underfitting refers to a model that can neither model the training dataset nor generalize to new dataset. An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training dataset.

Underfitting is often not discussed as it is easy to detect given a good performance metric.

Understanding it with Examples

Example 1:

Let's say three students have prepared for a mathematics examination.

The first student has only studied Addition mathematic operations and skipped other mathematics operations such as Subtraction, Division, Multiplication etc.

The second student has a particularly good memory. Thus, second student has memorized all the problems presented in the textbook.

And the third student has studied all mathematical operations and is well prepared for the exam.

In the exam student one will only be able to solve the questions related to Addition and will fail in problems or questions asked related to other mathematics operations.

Student two will only be able to answer questions if they happened to appear in the textbook (as he has memorized it) and will not be able to answer any other questions.

Student three will be able to solve all the exam problems reasonably well.

Machine Learning algorithms have similar behavior to our three students, sometimes the model generated by the algorithm are similar to the first student. They learn from only from a small part of the training dataset, in such cases the model is **Underfitting.**

Sometimes the model will memorize the entire training dataset, like the second student. They perform very well on known instances, but faulter badly on unseen data or unknown instances. In such cases the model is said to be **Overfitting.**

And when model does well in both the training dataset and on the unseen data or unknown instances like student three, it is a good fit.

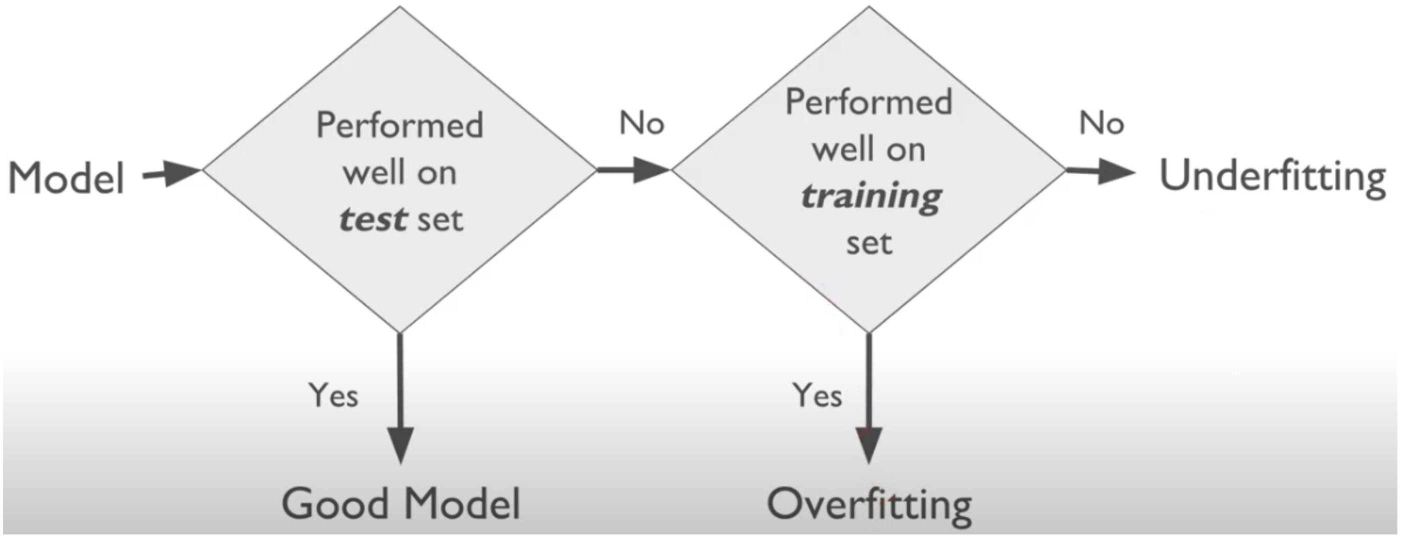
Example 2:

Let's have another example and consider that you have visited a city "X" and took a ride in a taxi. On speaking to friends, you later realize that the taxi driver charged you twice or three times more than the standard fare. This occurred as you were new in the city and driver quite literally took you for a ride.

Also, you purchased some items from a street vendor, and you again ended up paying more than they were worth. You finally decide that the people in the city "X" are dishonest. Which is a human trait, people often generalize. Machine learning models also have this weakness if we are not careful to avoid bias during the development stages: modeling, selecting algorithms, features, training dataset etc.

Suppose in the same city "X" another taxi driver charged you reasonably and as per the meter, but based on experience, you consider that this driver has also charged more. This is called Overfitting.

From above two examples we can say that, if model performs well on test or unseen dataset then it is a best fit or good model. And if did not perform well on test or unseen dataset but did well on training dataset then it is an Overfit model. And any model that did not do well in the training dataset nor in test dataset then it is a Underfit model.


Model
Performed well on *test* set
No
Performed well on *training* set
No
Underfitting
Yes
Good Model
Yes
Overfitting

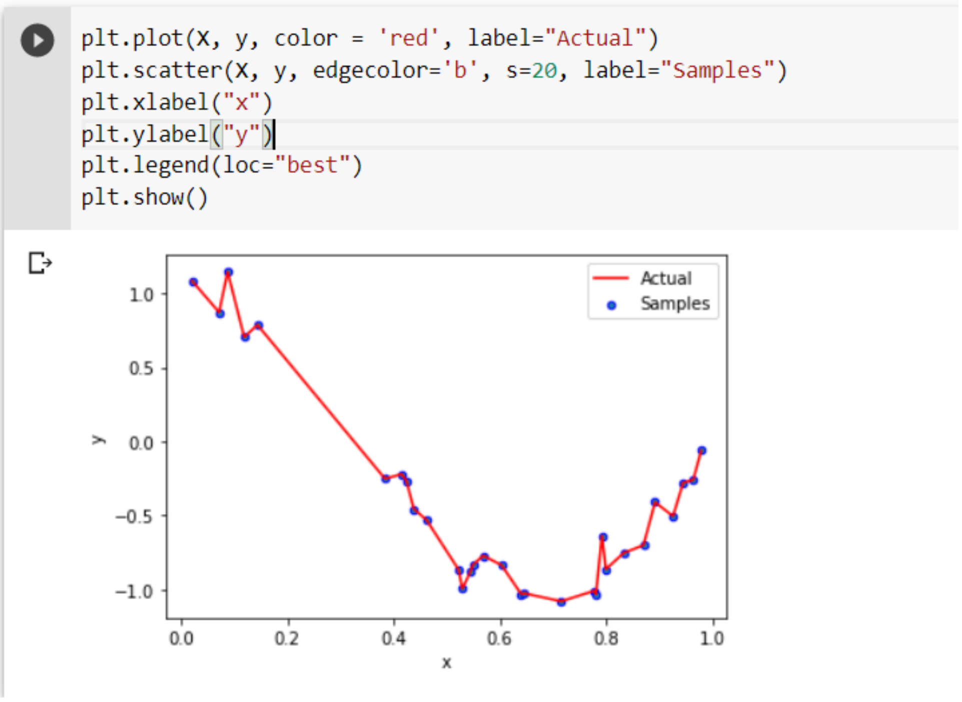
Next will see an example with python code.

Example 3:

This example demonstrates the problems of underfitting and overfitting and how we can use linear regression with polynomial features to approximate nonlinear functions.

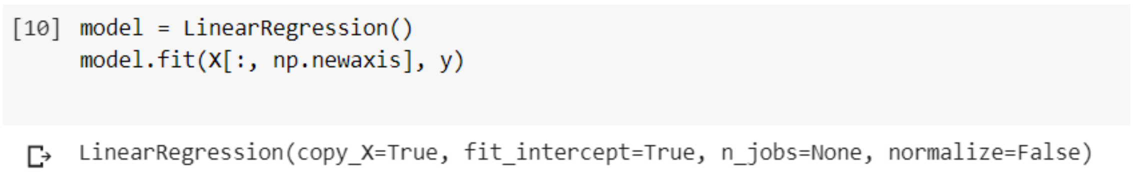
Let us generate two variables say X and y. X will have some random number / sample, where as y is a part of cosine function.

Based on the data the graph will look like… simple plotting X and y.



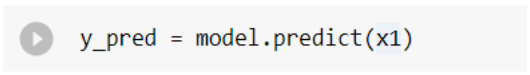
```
plt.plot(X, y, color = 'red', label="Actual")
plt.scatter(X, y, edgecolor='b', s=20, label="Samples")
plt.xlabel("x")
plt.ylabel("y")
plt.legend(loc="best")
plt.show()
```

Let's train our model using Linear Regression, predict and visualize it.



```
[10] model = LinearRegression()
     model.fit(X[:, np.newaxis], y)
```

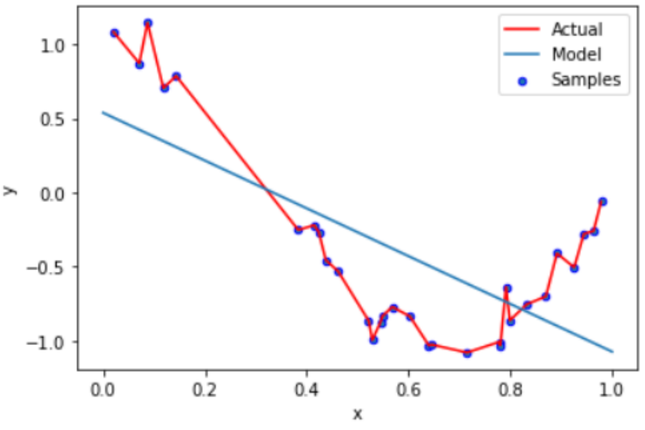
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```



```
y_pred = model.predict(x1)
```

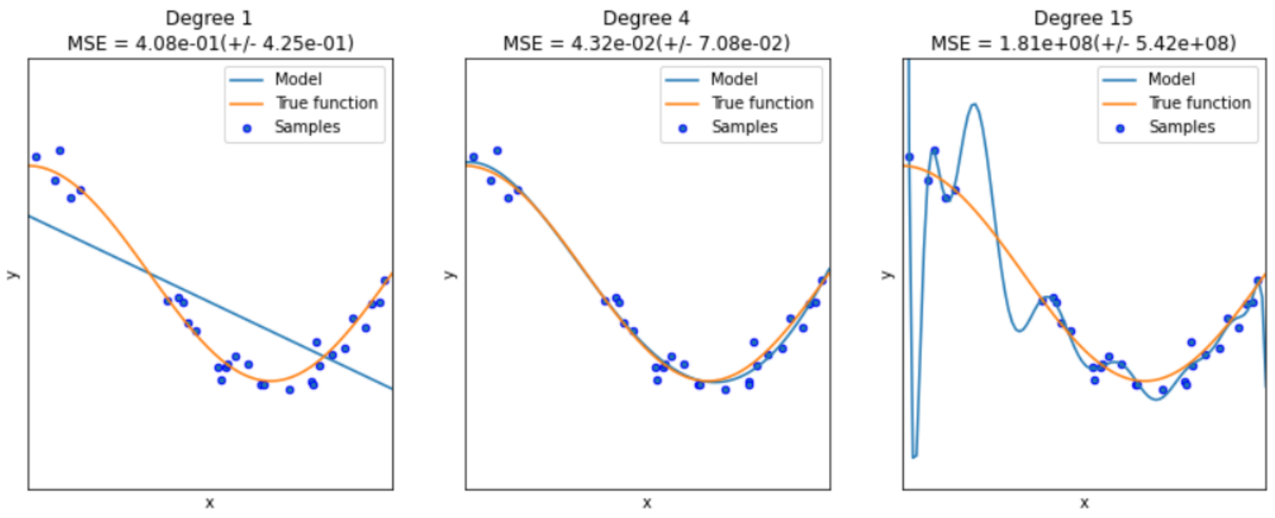
Let's visualize the predicted model.

```
[13] # plt.scatter(X, y, s=10)
     # plt.plot(x1, y_pred, color='r')
     # plt.show()

     plt.plot(X, y, color = 'red', label="Actual")
     plt.scatter(X, y, edgecolor='b', s=20, label="Samples")
     plt.plot(x1, y_pred, label="Model")

     plt.xlabel("x")
     plt.ylabel("y")
     plt.legend(loc="best")
     plt.show()
```



We can see that the straight line is unable to capture the patterns in the data. This is an example of under-fitting. The error will be huge in this model.

Let's consider the Polynomial feature with some degrees and train our model.



We can see that a linear function (polynomial with degree 1) is not enough to fit the training samples. This is called **underfitting.**

A polynomial of degree 4 approximates the true function almost perfectly. This is called Best Fit or Good Model.

However, for higher degrees (with degree 15) the model will overfit the training dataset, i.e. it learns the noise of the training data.

We evaluate quantitatively overfitting / underfitting by using cross-validation.

We also calculate the mean squared error (MSE) on the validation dataset, the higher the less likely the model generalizes correctly from the training dataset.

Full code can be referenced from https://colab.research.google.com/drive/1XzngJPT8WUyHFW-JxhrawVhPurJlv1cA?usp=sharing (https://colab.research.google.com/drive/1XzngJPT8WUyHFW-JxhrawVhPurJlv1cA?usp=sharing)

Detecting Overfitting or Underfitting

A key challenge of detecting any kind of fit (be it underfitting or best fit or overfitting), is almost impossible before you test the data. It can help address the inherent characteristics of overfitting, which is the inability to generalize a dataset. The data can therefore be separated into different subsets to make it easy for training and testing. The data is split into two main parts, i.e., a test dataset and a training dataset.

Splitting technique may vary based on the type of dataset and one can use any splitting technique.

If our model does much better on the training dataset than on the test dataset, then we're likely **overfitting** For example, our model performed with a 99% accuracy on the training dataset but only 50-55% accuracy on the test dataset. It is Overfitting the model and did not performed well on unseen dataset.

If our model does much better on the test dataset than on the training dataset, then we are likely **underfitting.**

And if our model does well on both the training and test datasets then we have the best fit. For example, our model performed 90% accuracy on the training dataset and performs 88% - 92% accuracy on the test dataset. It is the best fit model.

Another simple way to detect this is by using cross-validation. This attempts to examine the trained model with a new data set to check its predictive accuracy. Given a dataset, some portion of this is held back (say 30%) while the rest is used in training the model. Once the model has been trained the reserved data is then used to check the accuracy of the model compared to the accuracy of derived from the data used in training. A significant variance in these two flags overfitting.

How to Prevent Overfitting or Underfitting

Detecting overfitting or underfitting is useful, but it does not solve the problem. Fortunately, you have several options to try. Here are a few of the most popular solutions.

The remedy for Underfitting, is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.

To prevent overfitting, there are various ways and a few of them are shown below.

- Cross-validation:
  - Cross-validation is a powerful preventative measure against overfitting.
  - Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.
  - In standard k-fold cross-validation, we partition the data into k subsets, called folds. Then, we iteratively train the algorithm on k-1 folds while using the remaining fold as the test set (called the "holdout fold").
  - Cross-validation allows you to tune hyperparameters with only your original training dataset. This allows you to keep your test dataset as a truly unseen dataset for selecting your final model.
- Train with more data
  - It won't work every time, but training with more data can help algorithms detect the signal better.
  - As the user feeds more training data into the model, it will be unable to overfit all the samples and will be forced to generalize to obtain results.
  - Users should continually collect more data as a way of increasing the accuracy of the model.
  - However, this method is considered expensive, and, therefore, users should ensure that the data being used is relevant and clean.
  - Of course, that is not always the case. If we just add more noisy data, this technique will not help. That is why you should always ensure your data is clean and relevant.
- Data augmentation
  - An alternative to training with more data is data augmentation, which is less expensive compared to the former.
  - If you are unable to continually collect more data, you can make the available data sets appear diverse.
  - Data augmentation makes a data sample look slightly different every time it is processed by the model. The process makes each data set appear unique to the model and prevents the model from learning the characteristics of the data sets.
- Reduce Complexity or Data Simplification
  - Overfitting can occur due to the complexity of a model, such that, even with large volumes of data, the model still manages to overfit the training dataset.
  - The data simplification method is used to reduce overfitting by decreasing the complexity of the model to make it simple enough that it does not overfit.
  - Some of the actions that can be implemented include pruning a decision tree, reducing the number of parameters in a Neural Networks, and using dropout on a Neural Networks.
  - Simplifying the model can also make the model lighter and run faster.
- Regularization

- Regularization refers to a broad range of techniques for artificially forcing your model to be simpler.
    - The method will depend on the type of learner you are using. For example, you could prune a decision tree, use dropout on a neural network, or add a penalty parameter to the cost function in regression.
    - Oftentimes, the regularization method is a hyperparameter as well, which means it can be tuned through cross-validation.
    - To learn about Regularization refer to the article https://datascience.foundation/datatalk/regularization-machine-learning-teacher (https://datascience.foundation/datatalk/regularization-machine-learning-teacher) written and very well explained by Abhishek Mishra.
- Ensembling
    - Ensembles are machine learning methods for combining predictions from multiple separate models. There are a few different methods for ensembling, but the two most common are: Boosting and Bagging.
    - Boosting works by using simple base models to increase their aggregate complexity. It trains a large number of weak learners arranged in a sequence, such that each learner in the sequence learns from the mistakes of the learner before it.
    - Boosting attempts to improve the predictive flexibility of simple models.
    - Boosting combines all the weak learners in the sequence to bring out one strong learner.
    - Bagging works by training many strong learners arranged in a parallel pattern and then combining them to optimize their predictions.
    - Bagging attempts to reduce the chance of overfitting complex models.
    - Bagging then combines all the strong learners together to "smooth out" their predictions.
- Early Stopping
    - When you're training a learning algorithm iteratively, you can measure how well each iteration of the model performs.
    - Up until a certain number of iterations, new iterations improve the model. After that point, however, the model's ability to generalize can weaken as it begins to overfit the training data.
    - Early stopping refers stopping the training process before the learner passes that point.
    - Today, this technique is mostly used in deep learning while other techniques (e.g. regularization) are preferred for classical machine learning.
- You need to add regularization in case of Linear and SVM models.
- In decision tree models you can reduce the maximum depth.
- While in Neural Networks, you can introduce dropout layer to reduce overfitting.

Quick Summary

Overfitting is a modeling error that introduces bias to the model because it is too closely related to the data set.

Overfitting makes the model relevant to its data set only, and irrelevant to any other data sets.

Some of the methods used to prevent overfitting include ensembling, data augmentation, data simplification, and cross-validation.

References

1. https://datascience.foundation/datatalk/regularization-machine-learning-teacher (https://datascience.foundation/datatalk/regularization-machine-learning-teacher)
2. Wikipedia.org (https://en.wikipedia.org/wiki/Overfitting) (https://en.wikipedia.org/wiki/Overfitting)
3. https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html (https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html)

Comments: