https://www.ibm.com/cloud/learn/overfitting

Overfitting

Learn how to avoid overfitting, so that you can generalize data outside of your model accurately.

What is overfitting?

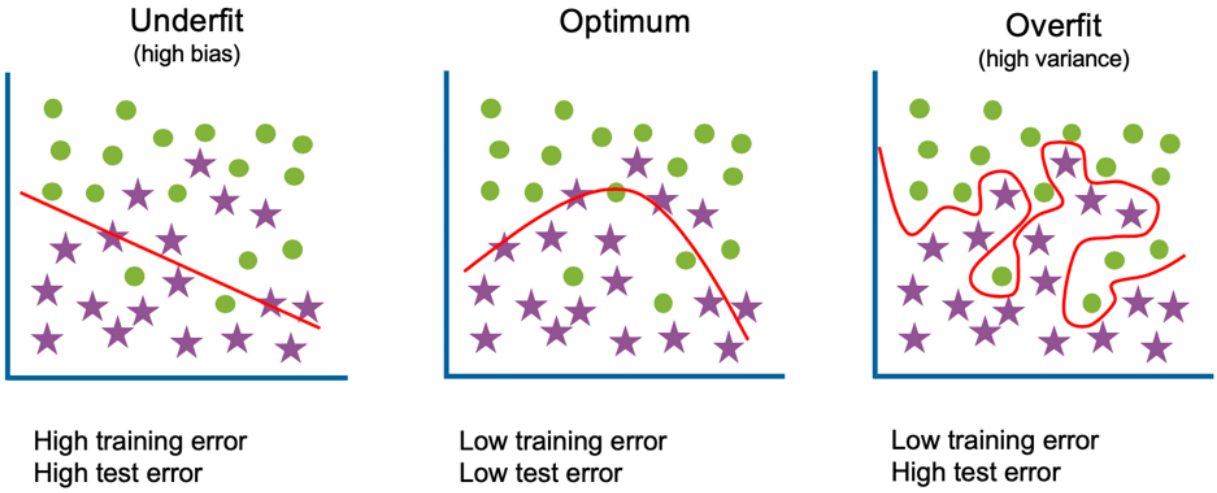
Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm unfortunately cannot perform accurately against unseen data, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data.

When machine learning algorithms are constructed, they leverage a sample dataset to train the model. However, when the model trains for too long on sample data or when the model is too complex, it can start to learn the “noise,” or irrelevant information, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes “overfitted,” and it is unable to generalize well to new data. If a model cannot generalize well to new data, then it will not be able to perform the classification or prediction tasks that it was intended for.

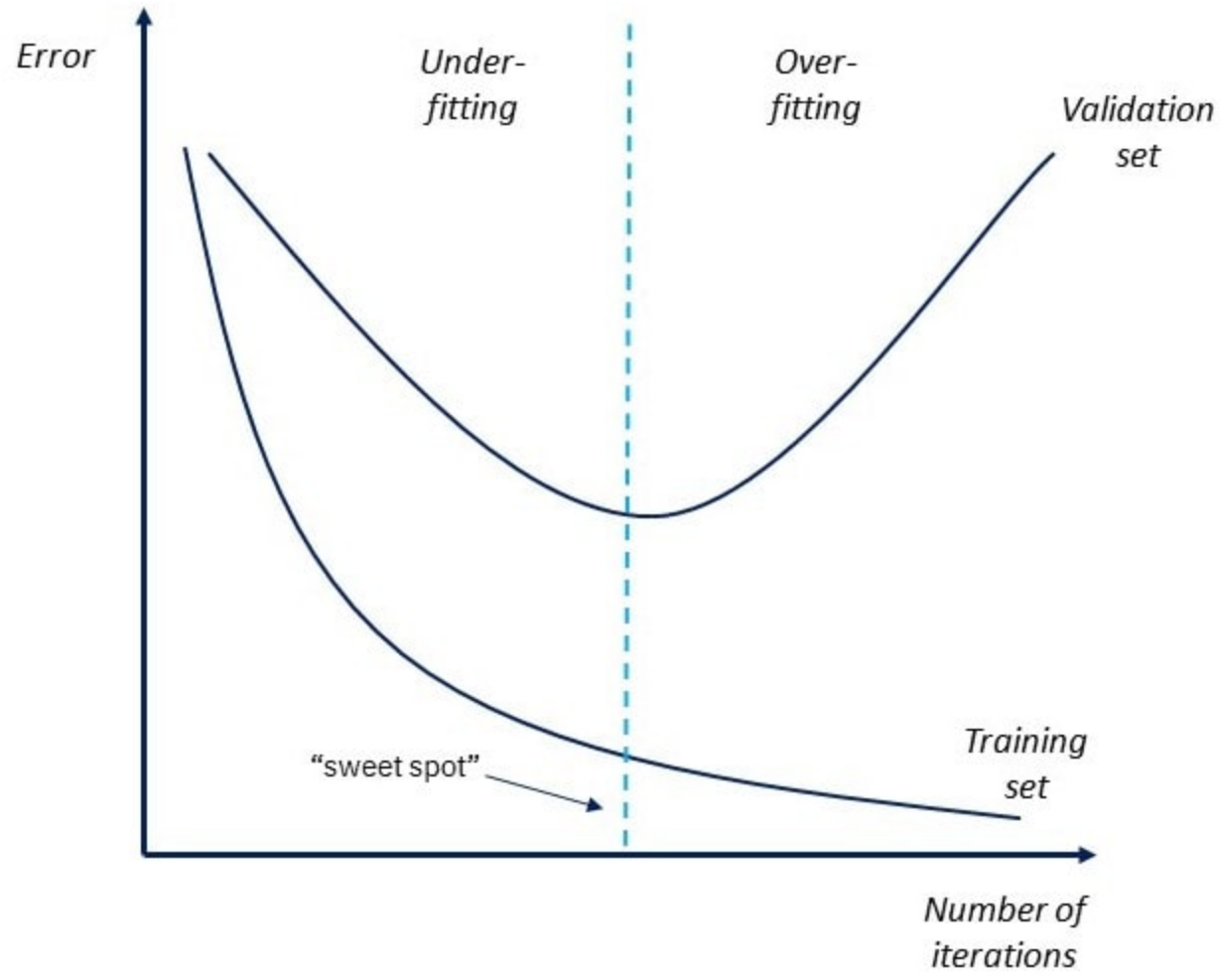
Low error rates and a high variance are good indicators of overfitting. In order to prevent this type of behavior, part of the training dataset is typically set aside as the “test set” to check for overfitting. If the training data has a low error rate and the test data has a high error rate, it signals overfitting.

Overfitting vs. underfitting

If overtraining or model complexity results in overfitting, then a logical prevention response would be either to pause training process earlier, also known as, “early stopping” or to reduce complexity in the model by eliminating less relevant inputs. However, if you pause too early or exclude too many important features, you may encounter the opposite problem, and instead, you may underfit your model. Underfitting occurs when the model has not trained for enough time or the input variables are not significant enough to determine a meaningful relationship between the input and output variables.



In both scenarios, the model cannot establish the dominant trend within the training dataset. As a result, underfitting also generalizes poorly to unseen data. However, unlike overfitting, underfitted models experience high bias and less variance within their predictions. This illustrates the bias-variance tradeoff, which occurs when as an underfitted model shifted to an overfitted state. As the model learns, its bias reduces, but it can increase in variance as becomes overfitted. When fitting a model, the goal is to find the “sweet spot” in between underfitting and overfitting, so that it can establish a dominant trend and apply it broadly to new datasets.

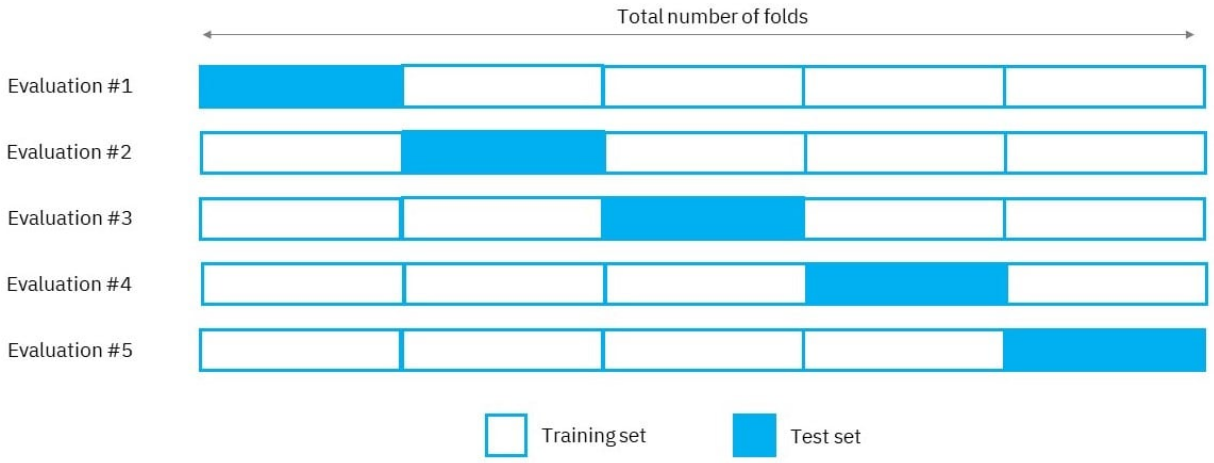


How to detect overfit models

To understand the accuracy of machine learning models, it's important to test for model fitness. K-fold cross-validation is one of the most popular techniques to assess accuracy of the model.

In k-folds cross-validation, data is split into k equally sized subsets, which are also called "folds." One of the k-folds will act as the test set, also known as the holdout set or validation set, and the remaining folds will train the model. This process repeats until each of the fold has acted as a holdout fold. After each evaluation, a score is retained and when all iterations have completed, the scores are averaged to assess the performance of the overall model.

For example, let's say we divided the dataset into five sub-groups. The process can be visualized, like this:



How to avoid overfitting

While using a linear model helps us avoid overfitting, many real-world problems are nonlinear ones. In addition to understanding how to detect overfitting, it is important to understand how to avoid overfitting altogether. Below are a number of techniques that you can use to prevent overfitting:

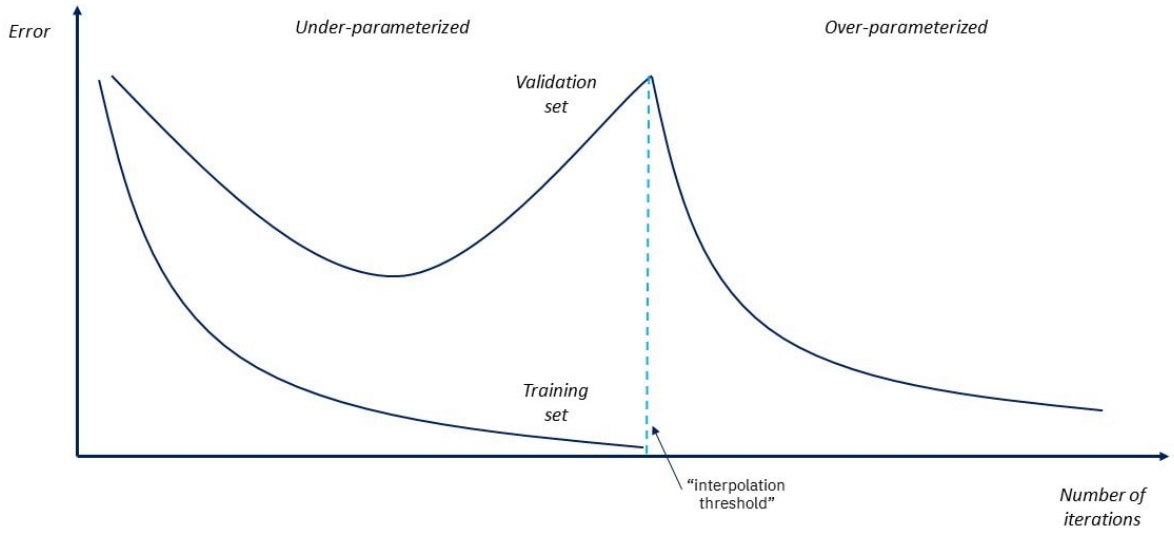
- **Early stopping:** As we mentioned earlier, this method seeks to pause training before the model starts learning the noise within the model. This approach risks halting the training process too soon, leading to the opposite problem of underfitting. Finding the "sweet spot" between underfitting and overfitting is the ultimate goal here.
- **Train with more data:** Expanding the training set to include more data can increase the accuracy of the model by providing more opportunities to parse out the dominant relationship among the input and output variables. That said, this is a more effective method when clean, relevant data is injected into the model. Otherwise, you could just continue to add more complexity to the model, causing it to overfit.
- **Data augmentation:** While it is better to inject clean, relevant data into your training data, sometimes noisy data is added to make a model more stable. However, this method should be done sparingly.
- **Feature selection:** When you build a model, you'll have a number of parameters or features that are used to predict a given outcome, but many times, these features can be redundant to others. Feature selection is the process of identifying the most important ones within the training data and then eliminating the irrelevant or redundant ones. This is commonly mistaken for dimensionality reduction, but it is different. However, both methods help to simplify your model to establish the dominant trend in the data.

- **Regularization:** If overfitting occurs when a model is too complex, it makes sense for us to reduce the number of features. But what if we don't know which inputs to eliminate during the feature selection process? If we don't know which features to remove from our model, regularization methods can be particularly helpful. Regularization applies a "penalty" to the input parameters with the larger coefficients, which subsequently limits the amount of variance in the model. While there are a number of regularization methods, such as L1 regularization, Lasso regularization, and dropout, they all seek to identify and reduce the noise within the data.
- **Ensemble methods:** Ensemble learning methods are made up of a set of classifiers—e.g. decision trees—and their predictions are aggregated to identify the most popular result. The most well-known ensemble methods are bagging and boosting. In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. After several data samples are generated, these models are then trained independently, and depending on the type of task—i.e. regression or classification—the average or majority of those predictions yield a more accurate estimate. This is commonly used to reduce variance within a noisy dataset.

See how to leverage some of these approaches within this IBM Developer image recognition tutorial.

Recent research

While the above is the established definition of overfitting, recent research (PDF, 1.2 MB) (link resides outside of IBM) indicates that complex models, such as deep learning models and neural networks, perform at a high accuracy despite being trained to "exactly fit or interpolate." This finding is directly at odds with the historical literature on this topic, and it explained through the "double descent" risk curve below. You can see that as the model learns past the threshold of interpolation, the performance of the model improves. The methods that we mentioned earlier to avoid overfitting, such as early stopping and regularization, can actually prevent interpolation.



IBM and overfitting

IBM Watson Studio is an open data platform which allows data scientists to build, run, test and optimize AI models at scale across any cloud. IBM Watson Studio empowers you to operationalize AI anywhere as part of IBM Cloud Pak® for Data. Unite teams, simplify AI lifecycle management and accelerate time to value with an open, flexible multicloud architecture.

To build machine learning models with accuracy, sign up for the IBMid and create your IBM Cloud account today.