Featured Machine learning

Overfitting and underfitting in machine learning

October 17, 2022 15 min

Your Email

Back to blog

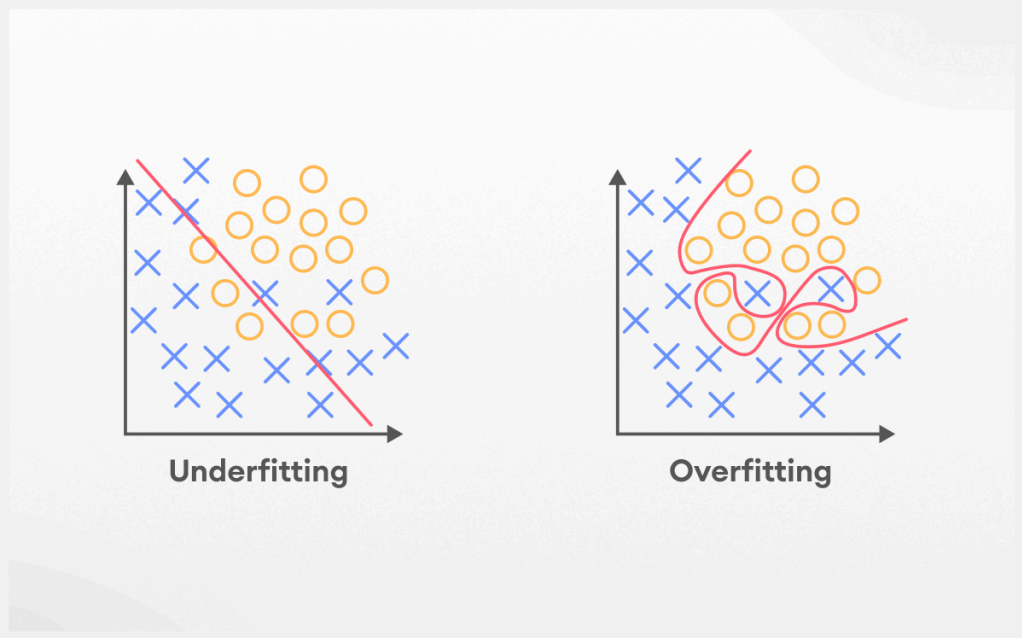
Contents

Overfitting and underfitting are two of the biggest reasons why machine learning algorithms and models don't get good results. Understanding why they emerge in the first place and taking action to prevent them may boost your model performance on many levels. Let's better explore the difference between overfitting and underfitting through a hypothetical example.

Assume two children must take a math exam. Due to time constraints, the first child only learned addition and was unable to learn subtraction, multiplication, or division. The second child had a phenomenal memory but was not very good at math, so instead, he memorized all the problems in the problem book. During the exam, the first child solved only addition-related math problems and was not able to tackle math problems involving the other three basic arithmetic operations. On the other hand, the second child was solely capable of solving problems he memorized from the math problem book and was unable to answer any other questions. In this case, if the math exam questions were from another textbook and included questions related to all kinds of basic arithmetic operations, both children would not manage to pass it.

Machine learning algorithms sometimes demonstrate behavior similar to these two children. There are times when they learn only from a small part of the training dataset (similar to the kid who learned only addition). In such cases, the model is underfitting. In other cases, machine learning models memorize the entire training dataset (like the second child) and perform superbly on known instances but fail on unseen data. In situations like this, the model is overfitting. Overfitting and

In this article, we will cover generalization, bias-variance tradeoffs, and how they are connected to overfitting and overfitting principles. We will also explore the differences between overfitting and underfitting, how to detect and prevent them, as well as will dive deeper into models prone to overfitting and underfitting.



What is generalization

In supervised learning, the main goal is to use training data to build a model that will be able to make accurate predictions based on new, unseen data, which has the same characteristics as the initial training set. This is known as generalization. Generalization relates to how effectively the concepts learned by a machine learning model apply to particular examples that were not used throughout the training. You want to create a model that can generalize as precisely as possible. However, in the real world, this is a tricky problem.

Generalization error

the validation set, and in the end, evaluate the performance of your best model on the testing set. The error rate on new cases is called the generalization error (or out-of-sample error), and by evaluating your models on the validation set, you get an estimate of this error. This value tells you how well your models perform on instances it has never iterated on.

A model's generalization error (also known as a prediction error) can be expressed as the sum of three very different errors: Bias error, variance error, and irreducible error.

Note: The irreducible error occurs due to the noisiness of the data itself. The only way to reduce this part of the error is to clean up the data (e.g., fix the data sources, such as broken sensors, or detect and remove outliers).

The concept of bias: Bias error

This type of error results from incorrect assumptions, such as thinking that the data is linear when it is actually quadratic. Bias is defined as a systematic error that happens in the machine learning model as a result of faulty ML assumptions. Bias is also the average squared difference between predictions of the model and actual data. Models with a higher percentage of bias will not match the training data. On the other hand, models with lower bias rates will coincide with the training dataset. Characteristics of a high-bias model include:

- Failure to capture proper data trends
- Potential towards underfitting
- More generalized/overly simplified
- High error rate

For a more detailed overview of bias in machine learning and other relevant topics, check out our blog.

The concept of variance: Variance error

Variance, as a generalization error, occurs due to the model's excessive sensitivity to small variations in the training data. In supervised learning, the model learns from training data. So, if you change the training data, the model will also be affected. The variance shows the amount by which the performance of the predictive model will be impacted when evaluating based on the validation data. If your model can generalize well, it shouldn't change too much from one split to another. For example, a model with many degrees of freedom (such as a high-degree polynomial model) is likely to have high variance, while linear models will probably have lower variance. A high-variance model typically has the following qualities:

- Noise in the data set
- Potential towards overfitting

Bias-variance tradeoff

Bias/variance in machine learning relates to the problem of simultaneously minimizing two error sources (bias error and variance error).

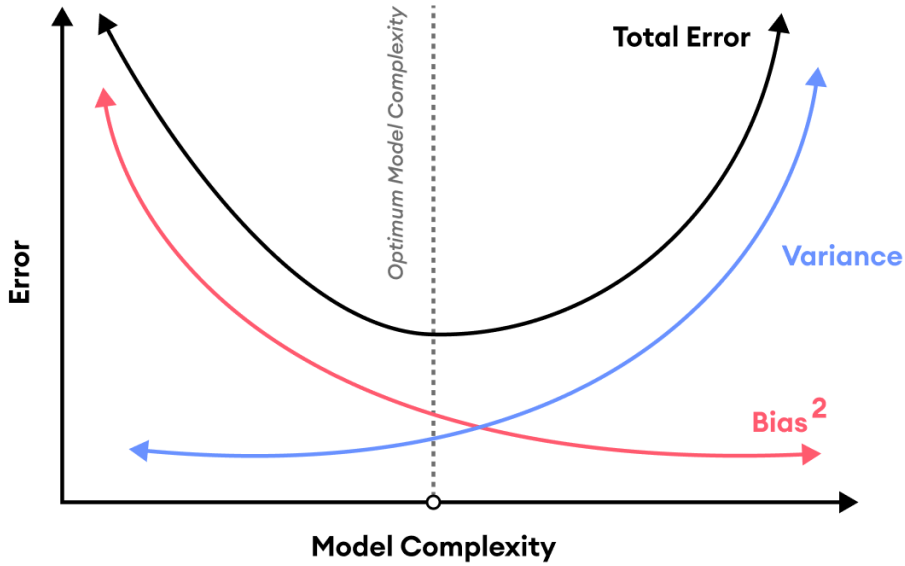


Figure 1. Bias/variance tradeoff: Image source

As demonstrated in Figure 1, if the model is too simple (e.g., linear model), it will have high bias and low variance. In contrast, if your model is very complex and has many parameters, it will have low bias and high variance. If you decrease the bias error, the variance error will increase and vice versa. This correlation is known as the bias-variance tradeoff.

In order to find a balance between underfitting and overfitting (the best model possible), you need to find a model which minimizes the total error.

Total Error = Bias^2+ Variance + Irreducible Error

To understand the math behind this equation, check out the following resource.

What is underfitting

Underfitting occurs when a model is not able to make accurate predictions based on training data and hence, doesn't have the capacity to generalize well on new data. Another case of underfitting is when a model is not able to learn enough from training data (Figure 2), making it difficult to capture the dominating trend (the model is unable to create a mapping between the input and the target variable).

Machine learning models with underfitting tend to have poor performance both in training and testing sets (like the child who learned only addition and was not able to solve problems related to other basic arithmetic operations both from his math

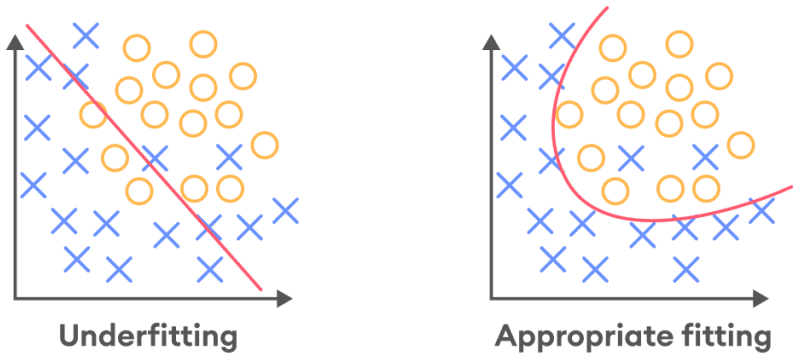


Figure 2. Underfitting model vs. good-fitting mode: Image source

What is overfitting

A model is considered overfitting when it does extremely well on training data but fails to perform on the same level on the validation data (like the child who memorized every math problem in the problem book and would struggle when facing problems from anywhere else). An overfitting model fails to generalize well, as it learns the noise and patterns of the training data to the point where it negatively impacts the performance of the model on new data (figure 3). If the model is overfitting, even a slight change in the output data will cause the model to change significantly. Models that are overfitting usually have low bias and high variance (Figure 5).

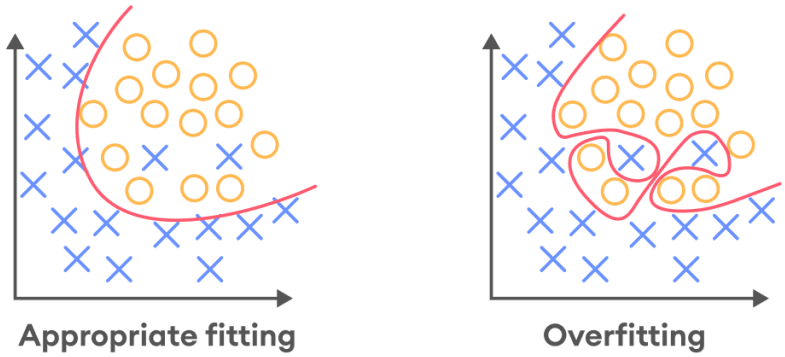


Figure 3. Good-fitting model vs. overfitting model: Image source

One of the core reasons for overfitting are models that have too much capacity. A model's capacity is described as the ability to learn from a particular dataset and is measured through Vapnik-Chervonenkis (VC) dimension. To understand the VC dimension, let's first define shattering.

Shattering is a model's ability to categorize a group of points. The model may provide a function that divides the points into two discrete classes while avoiding overlapping. Shattering is different from simple classification because it

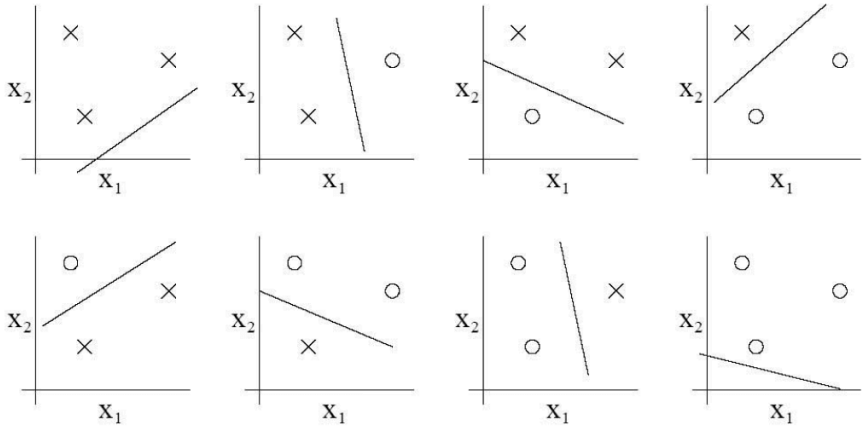


Figure 4. Shattering with 3 points: Image source

For any of the eight possible labeling of points presented in Figure 5, you can find a linear classifier that obtains "zero training error" on them. Moreover, it is obvious there is no set of four points this hypothesis class can shatter, so for this example, the VC dimension is 3.

Below you can see a diagram that provides a visual understanding of overfitting and underfitting. Your main goal as a machine learning engineer is to build a model that generalizes well and perfectly predicts correct values (in the dart's analogy, this will be the center of the target).

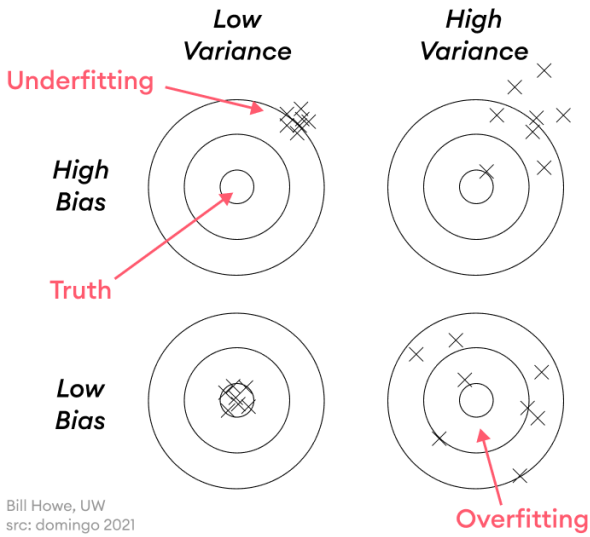


Figure 5. Overfitting and underfitting using a dart analogy: Image source

Underfitting in machine learning: How to detect underfitting

You already have a basic understanding of what underfitting and overfitting in machine learning are. The next goal is to understand how to detect and prevent them.

validation will be considerably high. In other words, for an underfitting dataset, the training and the validation error will be high.

2) **Oversimplistic prediction graph:** If a graph with the data points and the fitted curve is plotted, and the classifier curve is oversimplistic, then, most probably, your model is underfitting. In those cases, a more complex model should be tried out.

How to avoid underfitting

There are several things you can do to prevent underfitting in AI and machine learning models:

1) **Train a more complex model** – Lack of model complexity in terms of data characteristics is the main reason behind underfitting models. For example, you may have data with upwards of 100000 rows and more than 30 parameters. If you train data with the Random Forest model and set max depth (max depth determines the maximum depth of the tree) to a small number (for example, 2), your model will definitely be underfitting. Training a more complex model (in this respect, a model with a higher value of max depth) will help us solve the problem of underfitting.

2) **More time for training** - Early training termination may cause underfitting. As a machine learning engineer, you can increase the number of epochs or increase the duration of training to get better results.

3) **Eliminate noise from data** – Another cause of underfitting is the existence of outliers and incorrect values in the dataset. Data cleaning techniques can help deal with this problem.

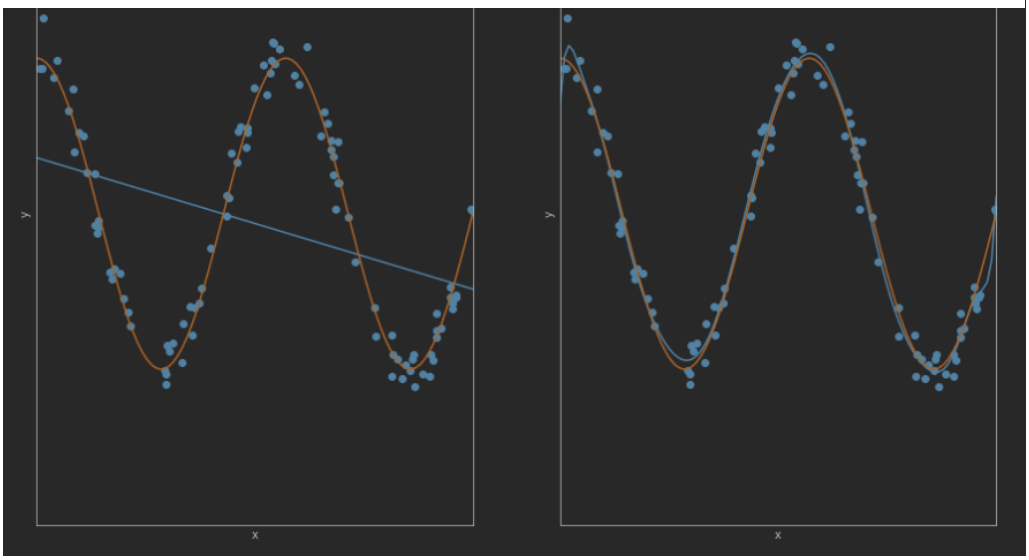
4) **Adjust regularization parameters** - the regularization coefficient can cause both overfitting and underfitting models.

5) **Try a different model** – if none of the above-mentioned principles work, you can try a different model (usually, the new model must be more complex by its nature). For example, you can try to replace the linear model with a higher-order polynomial model.

Model overfitting vs. underfitting: Models prone to underfitting

Some models are more prone to underfitting than others. Some examples of models that are usually underfitting include linear regression, linear discriminant analysis, and logistic regression. As you can guess from the above-mentioned names, linear models are often too simple and tend to underfit more compared to other models. However, this is not always the case, as models can also overfit - this typically happens when there are more features than the number of instances in the training data.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
#this allows us to create a random dataset
X = np.sort(np.random.rand(100))
#Lets create a true function
true_f = lambda X: np.cos(3.5 * np.pi * X)
y = true_f(X) + np.random.randn(100) * 0.1
degrees = [1,15]
plt.figure(figsize=(15, 10))
for i in range(len(degrees)):
   ax = plt.subplot(1, len(degrees), i+1)
   plt.setp(ax, xticks=(), yticks=())
   polynomial_features = PolynomialFeatures(degree=degrees[i], includ
   linear_regression = LinearRegression()
   #creating a structure for operation
   pipeline = Pipeline([("polynomial_features", polynomial_features),
   pipeline.fit(X[:, np.newaxis], y)
   #Testing
   X_test = np.linspace(0, 1, 100)
   yhat = pipeline.predict(X_test[:, np.newaxis])
   plt.plot(X_test, yhat,label="Model")
   plt.plot(X_test, true_f(X_test), label="True function")
   plt.scatter(X, y, label="Samples")
   plt.xlabel("x")
   plt.ylabel("y")
   plt.xlim((0, 1))
   plt.ylim((-2, 2))
   plt.legend(loc="best")
   plt.title("Degree %d" % degrees[i])
plt.show()
```



Overfitting in machine learning: How to detect overfitting

In machine learning and AI, overfitting is one of the key problems an engineer may face. Some of the techniques you can use to detect overfitting are as follows:

1) Use a resampling technique to estimate model accuracy. The most popular resampling technique is k-fold cross-validation. It allows you to train and test your model k-times on different subsets of training data and build up an estimate of the performance of a machine learning model on unseen data. The drawback here is that it is time-consuming and cannot be applied to complex models, such as deep neural networks.

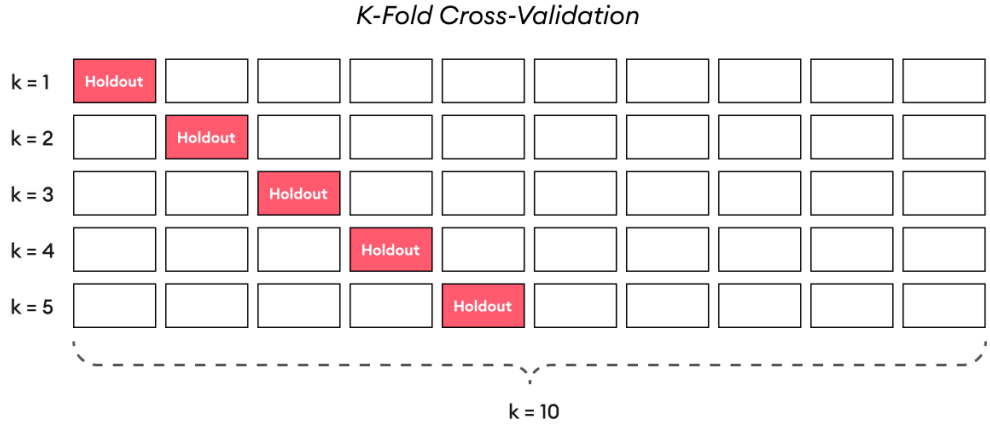


Figure 6. K-fold cross-validation: Image source

2) Hold back a validation set. Once a model is trained on the training set, you can evaluate it on the validation dataset, then compare the accuracy of the model in the training dataset and the validation dataset. A significant variance in these two results allows assuming that you have an overfitted model.

3) Another way to detect overfitting is by starting with a simplistic model that will serve as a benchmark. With this approach, if you try more complex algorithms, you

preferred over more complex ones (if your model is not getting significantly better after using a much more complex model, it is preferable to use a simpler model).

How to prevent overfitting

There are numerous ways to overcome overfitting in machine learning models. Some of those methods are listed below.

1) **Adding more data** – Most of the time, adding more data can help machine learning models detect the "true" pattern of the model, generalize better, and prevent overfitting. However, this is not always the case, as adding more data that is inaccurate or has many missing values can lead to even worse results.

2) **Early stopping** – In iterative algorithms, it is possible to measure how the model iteration performance. Up until a certain number of iterations, new iterations improve the model. After that point, however, the model's ability to generalize can deteriorate as it begins to overfit the training data. Early stopping refers to stopping the training process before the learner passes that point.

3) **Data augmentation** – In machine learning, data augmentation techniques increase the amount of data by slightly changing previously existing data and adding new data points or by producing synthetic data from a previously existing dataset.

4) **Remove features** – You can remove irrelevant aspects from data to improve the model. Many characteristics in a dataset may not contribute much to prediction. Removing non-essential characteristics can enhance accuracy and decrease overfitting.

5) **Regularization** – Regularization refers to a variety of techniques to push your model to be simpler. The approach you choose will be determined by the model you are training. For example, you can add a penalty parameter for a regression (L1 and L2 regularization), prune a decision tree or use dropout on a neural network.

6) **Ensembling** – Ensembling methods merge predictions from numerous different models. These methods not only deal with overfitting but also assist in solving complex machine learning problems (like combining pictures taken from different angles into the overall view of the surroundings). The most popular ensembling methods are boosting and bagging.

- **Boosting** – In boosting method, you train a large number of weak learners (constrained models) in sequence, and each sequence learns from the mistakes of the previous sequence. Then you combine all weak learners into a single strong learner.
- **Bagging** is another technique to reduce overfitting. It trains a large number of strong learners (unconstrained models) and then combines them all in order to optimize their predictions.

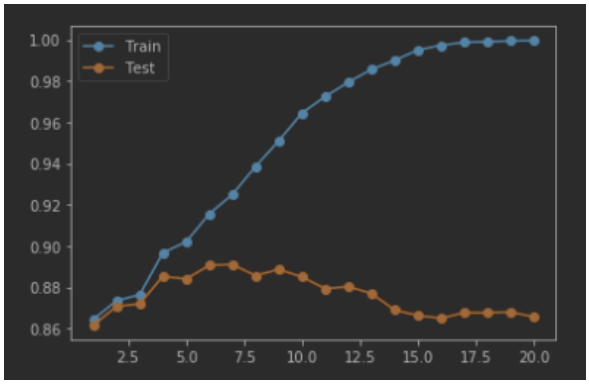
As mentioned earlier, a model is acknowledged as overfitting when it does extremely well on training data but fails to perform on that level for the test data. Nonparametric and nonlinear models, which are more flexible when learning a target function, are more prone to overfitting problems. As a result, many nonparametric machine learning methods contain parameters or approaches to limit the amount of detail learned by the model. Models such as decision trees and neural networks are more prone to overfitting.

To demonstrate that this model is prone to overfitting, let's look at the following example. In this example, random make classification() function was used to define a binary (two class) classification prediction problem with 10,000 examples (rows) and 20 input features (columns). The results are shown below.

```
# evaluate decision tree performance on train and test sets with diff
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from matplotlib import pyplot
# create dataset
X, y = make_classification(n_samples=10000, n_features=20, random_sta
# split into train test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
# define lists to collect scores
train_scores, test_scores = list(), list()
# define the tree depths to evaluate
values = [i for i in range(1, 21)]
# evaluate a decision tree for each depth
for i in values:
        # configure the model
        model = DecisionTreeClassifier(max_depth=i)
        # fit model on the training dataset
        model.fit(X_train, y_train)
        # evaluate on the train dataset
        train_yhat = model.predict(X_train)
        train_acc = accuracy_score(y_train, train_yhat)
        train_scores.append(train_acc)
        # evaluate on the test dataset
        test_yhat = model.predict(X_test)
        test_acc = accuracy_score(y_test, test_yhat)
        test_scores.append(test_acc)
        # summarize progress
```

```
pyplot.plot(values, test_scores, '-o', label='Test')
pyplot.legend()
pyplot.show()
```

```
>1, train: 0.865, test: 0.862
>2, train: 0.873, test: 0.871
>3, train: 0.876, test: 0.872
>4, train: 0.897, test: 0.885
>5, train: 0.902, test: 0.884
>6, train: 0.916, test: 0.891
>7, train: 0.925, test: 0.891
>8, train: 0.939, test: 0.886
>9, train: 0.951, test: 0.889
>10, train: 0.964, test: 0.885
>11, train: 0.972, test: 0.879
>12, train: 0.979, test: 0.880
>13, train: 0.986, test: 0.877
>14, train: 0.990, test: 0.869
>15, train: 0.995, test: 0.866
>16, train: 0.997, test: 0.865
>17, train: 0.999, test: 0.868
>18, train: 0.999, test: 0.868
>19, train: 0.999, test: 0.868
>20, train: 1.000, test: 0.866
```



As you can see, starting from depth 7, the model starts overfitting. You can use techniques like pruning to limit model overfitting.

Key takeaways

Generalization is the model's ability to make accurate predictions on new, unseen data that has the same characteristics as the training set. However, if your model

A model is underfitting when it is not able to make accurate predictions on training data, and it also doesn't have the capacity to generalize well on new data. Underfitting usually occurs when we deal with linear models (ordinary least squares, logistic models), and it can be prevented by training a more complex model, selecting more features, eliminating noise from the data, and adjusting regularization.

On the other hand, if a machine learning model is overfitted, it fails to perform that well on the test data, as opposed to the training data. Nonparametric and nonlinear models, which are more flexible when learning a target function, are more prone to overfitting problems. Some of the overfitting prevention techniques include data augmentation, regularization, early stoppage techniques, cross-validation, ensembling, etc.

Author: Tigran P., Independent Machine Learning Researcher

Tigran P.
Independent Machine Learning Researcher

Recommended for you

No items found.

Stay connected

Subscribe to receive new blog posts and latest discoveries in the industry from SuperAnnotate