EXPLAINERS

Overfitting in Machine Learning: What It Is and How to Prevent It

Did you know that there's one mistake…

…that thousands of data science beginners unknowingly commit?

And that this mistake can single-handedly ruin your machine learning model?

No, that's not an exaggeration. We're talking about one of the trickiest obstacles in applied machine learning: *overfitting*.

But don't worry:

In this guide, we'll walk you through exactly what overfitting means, how to spot it in your models, and what to do if your model is overfit.

By the end, you'll know how to deal with this tricky problem once and for all.

Table of Contents

Examples of Overfitting

Let's say we want to predict if a student will land a job interview based on her resume.

Now, assume we train a model from a dataset of 10,000 resumes and their outcomes.

Next, we try the model out on the original dataset, and it predicts outcomes with 99% accuracy… wow!

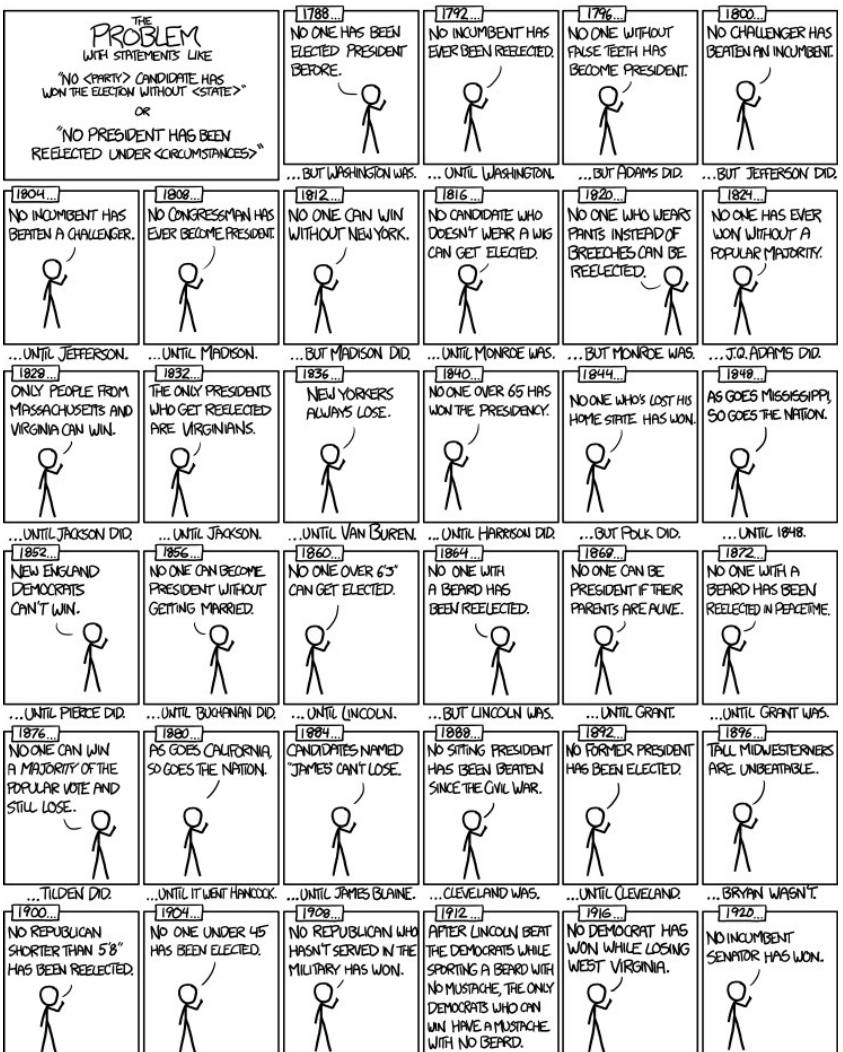
But now comes the bad news.

When we run the model on a new ("unseen") dataset of resumes, we only get 50% accuracy… uh-oh!

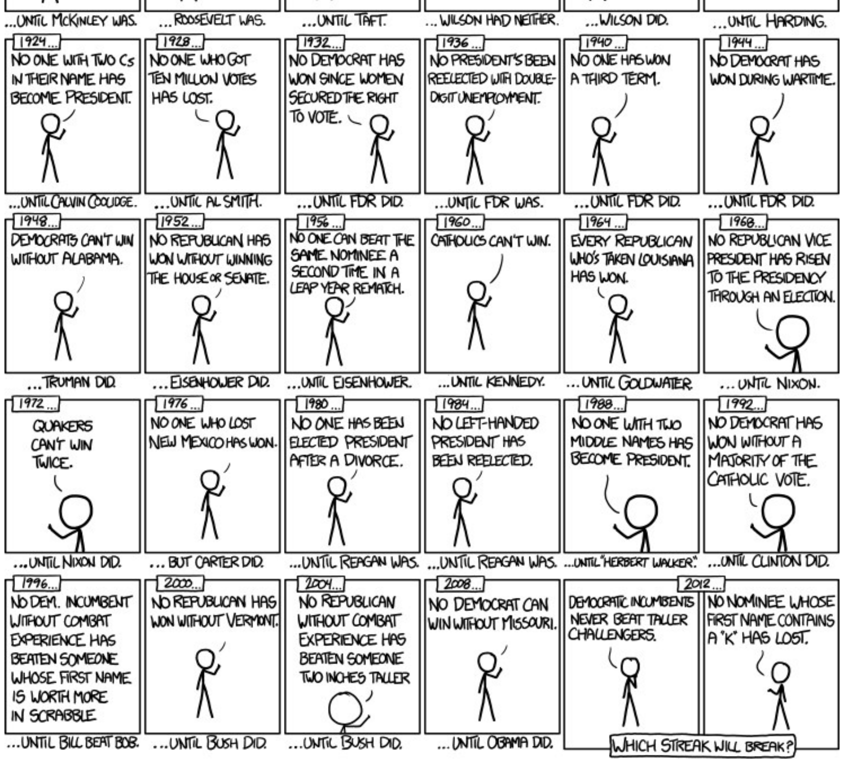
Our model doesn't *generalize* well from our training data to unseen data.

This is known as overfitting, and it's a common problem in machine learning and data science.

In fact, overfitting occurs in the real world all the time. You only need to turn on the news channel to hear examples:



THE PROBLEM WITH STATEMENTS LIKE "NO <PARTY> CANDIDATE HAS WON THE ELECTION WITHOUT <STATE>" OR "NO PRESIDENT HAS BEEN REELECTED UNDER <CIRCUMSTANCES>"
1788... NO ONE HAS BEEN ELECTED PRESIDENT BEFORE.
...BUT WASHINGTON WAS.
1792... NO INCUMBENT HAS EVER BEEN REELECTED.
...UNTIL WASHINGTON.
1796... NO ONE WITHOUT FALSE TEETH HAS BECOME PRESIDENT.
...BUT ADAMS DID.
1800... NO CHALLENGER HAS BEATEN AN INCUMBENT.
...BUT JEFFERSON DID.
1804... NO INCUMBENT HAS BEATEN A CHALLENGER.
...UNTIL JEFFERSON.
1808... NO CONGRESSMAN HAS EVER BECOME PRESIDENT.
...UNTIL MADISON.
1812... NO ONE CAN WIN WITHOUT NEW YORK.
...BUT MADISON DID.
1816... NO CANDIDATE WHO DOESN'T WEAR A WIG CAN GET ELECTED.
...UNTIL MONROE WAS.
1820... NO ONE WHO WEARS PANTS INSTEAD OF BREECHES CAN BE REELECTED.
...BUT MONROE WAS.
1824... NO ONE HAS EVER WON WITHOUT A POPULAR MAJORITY.
...J.Q. ADAMS DID.
1828... ONLY PEOPLE FROM MASSACHUSETTS AND VIRGINIA CAN WIN.
...UNTIL JACKSON DID.
1832... THE ONLY PRESIDENTS WHO GET REELECTED ARE VIRGINIANS.
...UNTIL JACKSON.
1836... NEW YORKERS ALWAYS LOSE.
...UNTIL VAN BUREN.
1840... NO ONE OVER 65 HAS WON THE PRESIDENCY.
...UNTIL HARRISON DID.
1844... NO ONE WHO'S LOST HIS HOME STATE HAS WON.
...BUT POLK DID.
1848... AS GOES MISSISSIPPI, SO GOES THE NATION.
...UNTIL 1848.
1852... NEW ENGLAND DEMOCRATS CAN'T WIN.
...UNTIL PIERCE DID.
1856... NO ONE CAN BECOME PRESIDENT WITHOUT GETTING MARRIED.
...UNTIL BUCHANAN DID.
1860... NO ONE OVER 6'3" CAN GET ELECTED.
...UNTIL LINCOLN.
1864... NO ONE WITH A BEARD HAS BEEN REELECTED.
...BUT LINCOLN WAS.
1868... NO ONE CAN BE PRESIDENT IF THEIR PARENTS ARE ALIVE.
...UNTIL GRANT.
1872... NO ONE WITH A BEARD HAS BEEN REELECTED IN PEACETIME.
...UNTIL GRANT WAS.
1876... NO ONE CAN WIN A MAJORITY OF THE POPULAR VOTE AND STILL LOSE.
...TILDEN DID.
1880... AS GOES CALIFORNIA, SO GOES THE NATION.
...UNTIL IT WENT HANCOCK.
1884... CANDIDATES NAMED "JAMES" CAN'T LOSE.
...UNTIL JAMES BLAINE.
1888... NO SITTING PRESIDENT HAS BEEN BEATEN SINCE THE CIVIL WAR.
...CLEVELAND WAS.
1892... NO FORMER PRESIDENT HAS BEEN ELECTED.
...UNTIL CLEVELAND.
1896... TALL MIDWESTERNERS ARE UNBEATABLE.
...BRYAN WASN'T.
1900... NO REPUBLICAN SHORTER THAN 5'8" HAS BEEN REELECTED.
1904... NO ONE UNDER 45 HAS BEEN ELECTED.
1908... NO REPUBLICAN WHO HASN'T SERVED IN THE MILITARY HAS WON.
1912... AFTER LINCOLN BEAT THE DEMOCRATS WHILE SPORTING A BEARD WITH NO MUSTACHE, THE ONLY DEMOCRATS WHO CAN WIN HAVE A MUSTACHE WITH NO BEARD.
1916... NO DEMOCRAT HAS WON WHILE LOSING WEST VIRGINIA.
1920... NO INCUMBENT SENATOR HAS WON.


...UNTIL MCKINLEY WAS.
...ROOSEVELT WAS.
...UNTIL TAFT.
...WILSON HAD NEITHER.
...WILSON DID.
...UNTIL HARDING.
1924...
NO ONE WITH TWO Cs IN THEIR NAME HAS BECOME PRESIDENT.
...UNTIL CALVIN COOLIDGE.
1928...
NO ONE WHO GOT TEN MILLION VOTES HAS LOST.
...UNTIL AL SMITH.
1932...
NO DEMOCRAT HAS WON SINCE WOMEN SECURED THE RIGHT TO VOTE.
...UNTIL FDR DID.
1936...
NO PRESIDENT'S BEEN REELECTED WITH DOUBLE-DIGIT UNEMPLOYMENT.
...UNTIL FDR WAS.
1940...
NO ONE HAS WON A THIRD TERM.
...UNTIL FDR DID.
1944...
NO DEMOCRAT HAS WON DURING WARTIME.
...UNTIL FDR DID.
1948...
DEMOCRATS CAN'T WIN WITHOUT ALABAMA.
...TRUMAN DID.
1952...
NO REPUBLICAN HAS WON WITHOUT WINNING THE HOUSE OR SENATE.
...EISENHOWER DID.
1956...
NO ONE CAN BEAT THE SAME NOMINEE A SECOND TIME IN A LEAP YEAR REMATCH.
...UNTIL EISENHOWER.
1960...
CATHOLICS CAN'T WIN.
...UNTIL KENNEDY.
1964...
EVERY REPUBLICAN WHO'S TAKEN LOUISIANA HAS WON.
...UNTIL GOLDWATER.
1968...
NO REPUBLICAN VICE PRESIDENT HAS RISEN TO THE PRESIDENCY THROUGH AN ELECTION.
...UNTIL NIXON.
1972...
QUAKERS CAN'T WIN TWICE.
...UNTIL NIXON DID.
1976...
NO ONE WHO LOST NEW MEXICO HAS WON.
...BUT CARTER DID.
1980...
NO ONE HAS BEEN ELECTED PRESIDENT AFTER A DIVORCE.
...UNTIL REAGAN WAS.
1984...
NO LEFT-HANDED PRESIDENT HAS BEEN REELECTED.
...UNTIL REAGAN WAS.
1988...
NO ONE WITH TWO MIDDLE NAMES HAS BECOME PRESIDENT.
...UNTIL "HERBERT WALKER."
1992...
NO DEMOCRAT HAS WON WITHOUT A MAJORITY OF THE CATHOLIC VOTE.
...UNTIL CLINTON DID.
1996...
NO DEM. INCUMBENT WITHOUT COMBAT EXPERIENCE HAS BEATEN SOMEONE WHOSE FIRST NAME IS WORTH MORE IN SCRABBLE.
...UNTIL BILL BEAT BOB.
2000...
NO REPUBLICAN HAS WON WITHOUT VERMONT.
...UNTIL BUSH DID.
2004...
NO REPUBLICAN WITHOUT COMBAT EXPERIENCE HAS BEATEN SOMEONE TWO INCHES TALLER.
...UNTIL BUSH DID.
2008...
NO DEMOCRAT CAN WIN WITHOUT MISSOURI.
...UNTIL OBAMA DID.
2012...
DEMOCRATIC INCUMBENTS NEVER BEAT TALLER CHALLENGERS.
NO NOMINEE WHOSE FIRST NAME CONTAINS A "K" HAS LOST.
WHICH STREAK WILL BREAK?

Overfitting Electoral Precedence (source: XKCD)

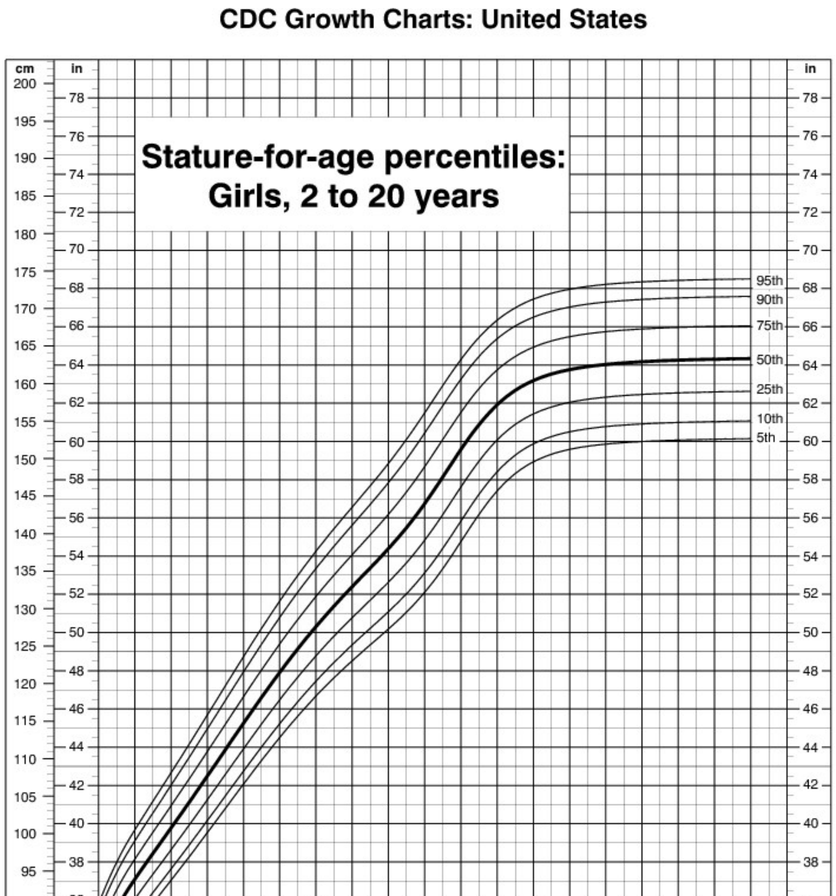
Signal vs. Noise

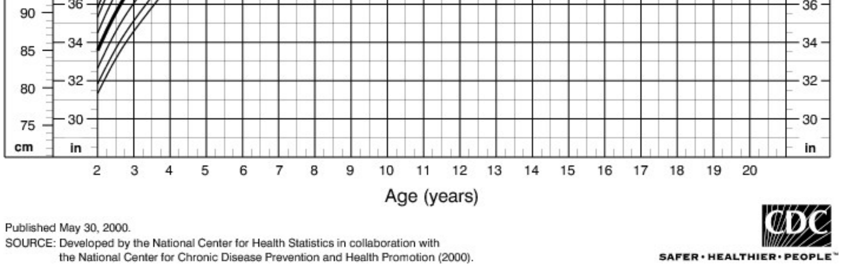
You may have heard of the famous book The Signal and the Noise by Nate Silver.

In predictive modeling, you can think of the "signal" as the true underlying pattern that you wish to learn from the data.

"Noise," on the other hand, refers to the irrelevant information or randomness in a dataset.

For example, let's say you're modeling height vs. age in children. If you sample a large portion of the population, you'd find a pretty clear relationship:





Height vs. Age (source: CDC)

This is the signal.

However, if you could only sample one local school, the relationship might be muddier. It would be affected by outliers (e.g. kid whose dad is an NBA player) and randomness (e.g. kids who hit puberty at different ages).

Noise interferes with signal.

Here's where machine learning comes in. A well functioning ML algorithm will separate the signal from the noise.

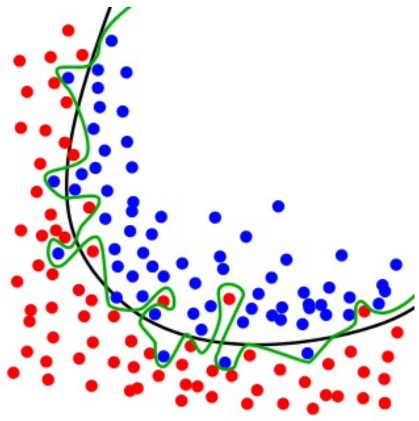
If the algorithm is too complex or flexible (e.g. it has too many input features or it's not properly regularized), it can end up "memorizing the noise" instead of finding the signal.

This overfit model will then make predictions based on that noise. It will perform unusually well on its training data… yet very poorly on new, unseen data.

Goodness of Fit

In statistics, *goodness of fit* refers to how closely a model's predicted values match the observed (true) values.

A model that has learned the noise instead of the signal is considered "overfit" because it fits the training dataset but has poor fit with new datasets.



While the black line fits the data well, the green line is overfit.

Overfitting vs. Underfitting

We can understand overfitting better by looking at the opposite problem, underfitting.

Underfitting occurs when a model is too simple – informed by too few features or regularized too much – which makes it inflexible in learning from the dataset.

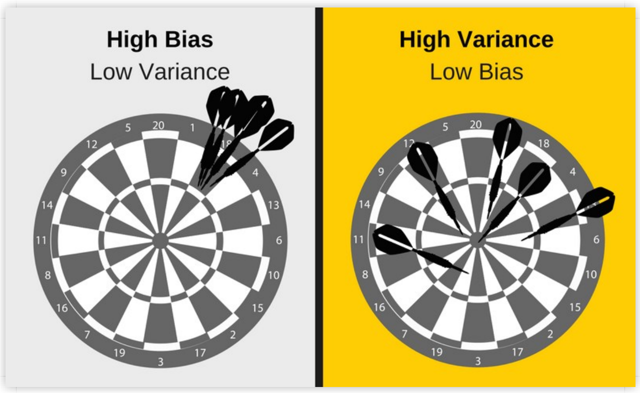
Simple learners tend to have less variance in their predictions but more bias towards wrong outcomes (see: The Bias-Variance Tradeoff).

On the other hand, complex learners tend to have more variance in their predictions.

Both bias and variance are forms of prediction error in machine learning.

Typically, we can reduce error from bias but might increase error from variance as a result, or vice versa.

This trade-off between too simple (high bias) vs. too complex (high variance) is a key concept in statistics and machine learning, and one that affects all supervised learning algorithms.

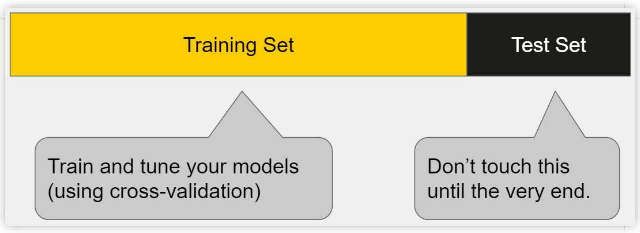


Bias vs. Variance (source: EDS)

How to Detect Overfitting

A key challenge with overfitting, and with machine learning in general, is that we can't know how well our model will perform on new data until we actually test it.

To address this, we can split our initial dataset into separate *training* and *test* subsets.


Training Set
Test Set
Train and tune your models (using cross-validation)
Don't touch this until the very end.

Train-Test Split

This method can approximate of how well our model will perform on new data.

If our model does much better on the training set than on the test set, then we're likely overfitting.

For example, it would be a big red flag if our model saw 99% accuracy on the training set but only 55% accuracy on the test set.

If you'd like to see how this works in Python, we have a full tutorial for machine learning using Scikit-Learn.

Another tip is to start with a very simple model to serve as a benchmark.

Then, as you try more complex algorithms, you'll have a reference point to see if the additional complexity is worth it.

This is the Occam's razor test. If two models have comparable performance, then you should usually pick the simpler one.

How to Prevent Overfitting

Detecting overfitting is useful, but it doesn't solve the problem. Fortunately, you have several options to try.

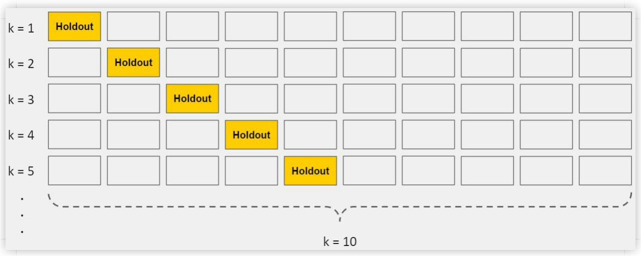
Here are a few of the most popular solutions for overfitting:

Cross-validation

Cross-validation is a powerful preventative measure against overfitting.

The idea is clever: 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").



K-Fold Cross-Validation

Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.

We have another article with a more detailed breakdown of cross-validation.

Train with more data

It won't work every time, but training with more data can help algorithms detect the signal better. In the earlier example of modeling height vs. age in children, it's clear how sampling more schools will help your model.

Of course, that's not always the case. If we just add more noisy data, this technique won't help. That's why you should always ensure your data is clean and relevant.

Remove features

Some algorithms have built-in feature selection.

For those that don't, you can manually improve their generalizability by removing irrelevant input features.

An interesting way to do so is to tell a story about how each feature fits into the model. This is like the data scientist's spin on software engineer's rubber duck debugging technique, where they debug their code by explaining it, line-by-line, to a rubber duck.

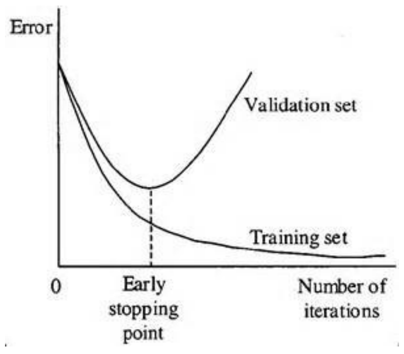
If anything doesn't make sense, or if it's hard to justify certain features, this is a good way to identify them.
In addition, there are several feature selection heuristics you can use for a good starting point.

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.

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're 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.

We have a more detailed discussion here on algorithms and regularization methods.

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:

Bagging attempts to reduce the chance overfitting complex models.

- It trains a large number of "strong" learners in parallel.
- A strong learner is a model that's relatively unconstrained.
- Bagging then combines all the strong learners together in order to "smooth out" their predictions.

Boosting attempts to improve the predictive flexibility of simple models.

- It trains a large number of "weak" learners in sequence.
- A weak learner is a constrained model (i.e. you could limit the max depth of each decision tree).
- Each one in the sequence focuses on learning from the mistakes of the one before it.
- Boosting then combines all the weak learners into a single strong learner.

While bagging and boosting are both ensemble methods, they approach the problem from opposite directions.

Bagging uses complex base models and tries to "smooth out" their predictions, while boosting uses simple base models and tries to "boost" their aggregate complexity.

Next Steps

Whew! We just covered quite a few concepts:

- Signal, noise, and how they relate to overfitting.
- Goodness of fit from statistics
- Underfitting vs. overfitting
- The bias-variance tradeoff
- How to detect overfitting using train-test splits
- How to prevent overfitting using cross-validation, feature selection, regularization, etc.

Hopefully seeing all of these concepts linked together helped clarify some of them.

To truly master this topic, we recommend getting hands-on practice.

While these concepts may feel overwhelming at first, they will 'click into place' once you start seeing them in the context of real-world code and problems.

So here are some additional resources to help you get started:

- Python machine learning tutorial
- Plotting overfitting and underfitting with Scikit-Learn
- More real-world examples of overfitting

Now, go forth and learn! (Or have your code do it for you!)

« Previous Post
Datasets for Data Science and Machine Learning

Next Post »
8 Fun Machine Learning Projects for Beginners