Sequence Interpolation Overfitting

Evan Dramko

March 2025

TLDR: If you have sequential data where interpolation between a point's neighbors is close to the actual answer, then use a continuous block to make the test set rather than a random shuffle.

Processing sequential data has been a core application of machine learning since its inception. Natural language, genetic sequencing, and financial data are among the most common use cases of machine learning, and are all sequential in nature. However, a subtle mistake in the preparation of such datasets can lead to highly unrealistic training *and testing* results. The problem I am referring to specifically is when an improper split of training and test data allows for the model to "memorize" points throughout the temporal spread of the data and unrealistically interpolate the testing values from them.

Whew! That is a mouthful. Fortunately, the idea is actually very easy. Lets look at illustrative example to clarify it. Specifically, we look at *stock price prediction*: the estimation of future stock prices based on data about the market's history. Lets assume our labels are a stock's price at a given day. We record the closing price of the stock every day over the past four years. Often, we would see the user randomly assign points to either the training or test set. However, in this case, doing so would may lead to far higher results, even in testing, than we would expect in real world deployment. The reason for this becomes apparent if we look at a graph of stock prices.

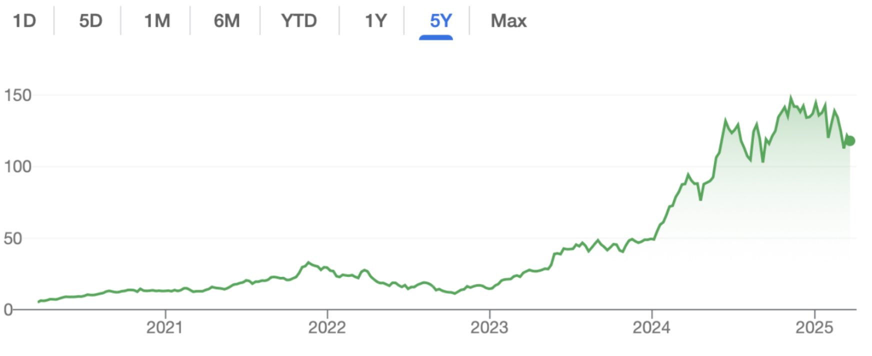

Image Sourced From Google "Market Summary"

In most cases, the price at any given day is likely close to the average between the prior and following day. In a random training/test split, it is quite likely that the neighbors or close by timestamps of any given test point will be in the training set. This means that the model can likely achieve a high accuracy on both the training and testing data by leveraging this interpolation and without learning anything else about the problem. However, in practical deployment we do not know the price of the stock in the future, making it impossible to apply this strategy.

There is a very easy way to handle this: instead of using a random split of the data, break off a continuous chunk of the data and use that as the training set. Now, the neighbors of the testing data do not appear in the training data. Once again, the testing data becomes a realistic indicator for how the model will perform in the real world.