Machine Learning: Core Concepts

Model Types

Linear regression is a way to predict the value of some a target variable by fitting a line that best describes the relationship between **Big X** and **little y** for the values we already have. If you remember `y = mx + b` from Algebra, it's the same thing; the `y` is the intercept, and the `b` is the beta coefficient. The beta coefficient tells us what change we can expect to see in `X` for every one-unit increase in `y`. If that doesn't seem familiar to you, don't worry about it; we'll give you everything you need to know.

Statistical Concepts

Cost Functions

When we train a model, we're solving an optimization problem. We provide training data to an algorithm and tell it to find the model or model parameters that best fit the data. But how can the algorithm judge what the "best" fit is? What criteria should it use?

A **cost function** (sometimes also called a loss or error function) is a mathematical formula that provides the score by which the algorithm will determine the best fit. Generally, the the goal is to minimize the cost function and get the lowest score. For linear models, these functions measure distance, and the model tries to to get the closest fit to the data. For tree-based models, they measure impurity, and the model tries to get the most terminal nodes.

Residuals

When we perform any type of regression analysis, we end up with a **line of best fit**. Because our data comes from the real world, it tends to be a little bit messy, so the data points usually don't fall *exactly* on this line. Most of the time, they're are scattered around it, and a **residual** is the vertical distance between each individual data point and the regression line. Each data point has only one residual which can be positive if it's above the regression line, negative if it's below the regression line, or zero if the line passes directly through the point. Think of it like this: the model describes theoretical line. That line doesn't really exist outside the model. The residuals, however, are true values; they represent the actual data that came from real observations.

Performance Metrics

In statistics, an *error* is the difference between a measurement and reality. There may not be any difference at all, but there's usually *something* not quite right, and we need to account for that in our model. To do that, we need to figure out the **mean absolute error (MAE)**. Absolute error is the

error in a single measurement, and mean absolute error is the average error over the course of several measurements.

Imagine that you're buying some bananas. The store charges for fruit based on weight, so you put your bananas on a scale before you head off to pay for them. The scale says they weigh 1.2 kilos, but your innate sense of weight tells you that they actually weight 0.9 kilos. The absolute error in that measurement would be 0.3 kilos. It can go the other way too: maybe you know the bananas weight 1.2 kilos, but the scale says they were 0.9 kilos. In that case, the absolute error would *still* be 0.3 kilos, because even though the numerical difference is -0.3, absolute values are always positive; all you have to do is disregard the `-` sign.

Let's keep going: you're sure the bananas don't weight 1.2 kilos, so you weigh them again. This time, the scale says 1.0 kilos. That's still wrong, so you weigh the bananas a third time, and now the scale says 2.3 kilos. Since the actual weight of your bananas hasn't changed, you now have a set of three absolute errors: 0.3, 0.1, and 1.4. If we average those errors together, we get 0.6, which is the *mean absolute error* for your banana data.

Data Concepts

Leakage

Leakage is the use of data in training your model that would not be typically be available when making predictions. For example, suppose we want to predict property prices in USD but include property prices in Mexican Pesos in our model. If we assume a fixed exchange rate or a nearly constant exchange rate, then our model will have a low error on the training data, but this will not be reflective of its performance on real world data.

Imputation

Datasets are often incomplete or missing entries. If the dataset is large and the missing entries are few, then the missing entries aren't all that important. But sometimes, it might be useful to include data with missing entries by finding a way to **impute** the missing entries in a row or column of a DataFrame. For example, you might use extrapolation when the data points have a pattern, or you might approximate the missing values by mean values.

Generalization

Notice that we tested the model with a dataset that's *different* from the one we used to train the model. Machine learning models are useful if they allow you to make predictions about data other than what you used to train your model. We call this concept **generalization.** By testing your model with different data than you used to train it, you're checking to see if your model can generalize. Most machine learning models do not generalize to all possible types of input data, so they should be used with care. On the other hand, machine learning models that don't generalize to make predictions for at least a restricted set of data aren't very useful.

Model Concepts

Hyperparameters

When we instantiate an estimator, we can pass keyword arguments that will dictate its structure. These arguments are called **hyperparameters**. For example, when we defined our decision tree estimator, we chose how many layers the tree would have using the `max_depth` keyword. This is in contrast to **parameters**, which are the numbers that our model uses to make predictions based on features. Parameters are optimized during the training process based on data and input features. They keep changing during training to fit the data and only the best performed ones were selected. Hyperparameters values are set before training begins and will not be changed during the training process. Pretty much all models have hyperparameters. Even a simple linear regressor has a hyperparameter: `fit_intercept` . Here are some common examples for Hyperparameters:

- The imputation strategy used for missing data.
- The number of trees in a random forest model.
- The number of jobs to run in parallel when fitting and predicting.

References and Further Reading

- Parameters and Hyperparameters in Machine Learning and Deep Learning

---