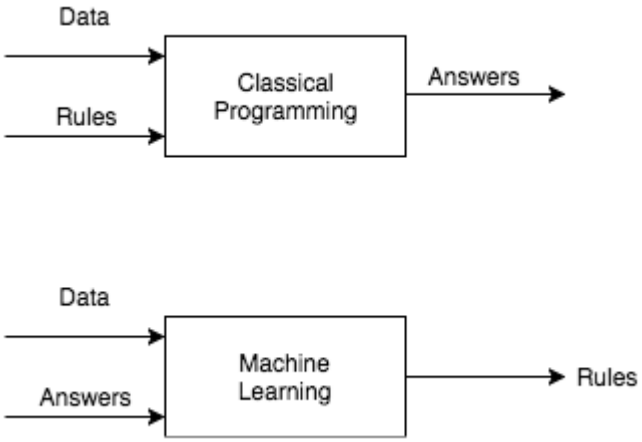
CS 6600: Machine Learning
Overview

Weber State University - Prof. Dylan Zwick



A few years back I ran across the diagram above, which I thought did a marvelous job of capturing the basic idea behind machine learning, and how it differs from traditional computer programming. With traditional computer programming, you tell the computer what to do - what the rules are - and provide the computer with inputs. Based upon these rules, the computer produces outputs - or answers - based on the inputs.

With machine learning, you don't know what the rules are, and you want the machine to figure them out! In other words, you want the machine to "learn" the rules. Now, of course there are a lot of nuances to this formulation that can't be captured by a single diagram, but in terms of something simple but informative this diagram is hard to beat.

What types of rules might you want to learn? Well, suppose you're working for Taboo Pizza here in Ogden, and you've added a new pizza to your menu. You want to get the word out, and you've got a marketing budget. There are many paid marketing channels you can use - Facebook, YouTube, billboards, mailers, radio, etc. You want to allocate your budget to these channels to maximize the sale of your new menu item. Now, you don't know the formula for the relation between marketing spend and pizza purchases, but you do have the spend and

sales data from your previous marketing campaigns. Could you use this data to help you optimize your current budget? Maybe you can, and you could use machine learning techniques to do it.

Let's make this a bit more mathematical. Typically, we denote the variables we can control, in this case marketing spend on various channels, with the symbol $\mathbf{X}$, and distinguish these variables with subscripts. So, X_1 could be Facebook spend, X_2 YouTube spend, and so on. These inputs sometimes go by different names like *predictors*, *independent variables*, *features*, or just *variables*. The output variable - in this case sales - is typically denoted by the symbol Y. It's sometimes called the *response* or *dependent variable*. We can then mathematically model the relationship between marketing spend and sales in the following very general form:

$$Y = f(\mathbf{X}) + \epsilon$$

where Y is sales, $\mathbf{X}$ is a vector whose components are the various marketing channel spends, so $\mathbf{X} = (X_1, X_2, \ldots, X_n)$, and ϵ is a random error term.

Why the error term? Well, because sales is of course a function of more than just marketing spend. Sales could also depend on the date, the weather, word of mouth, and a ton of other things. Marketing spend is only one component that goes into sales, and if it's the only thing that we control, the best that we can hope for in a model is one for which the error is independent of $\mathbf{X}$ and is unbiased, which means it has mean zero.

What we want to do is estimate the function f, because if we know f we can then optimize it to figure out the best allocation of marketing spend. Now, for someplace like Taboo Pizza this might seem like a lot of work. But for a company like, say, Overstock.com that spends $300,000,000 per year on marketing, estimating this function is a *big* deal.

So, how do we estimate this function? Well, *machine learning* is essentially a set of approaches for estimating f. We'll get into some of these approaches in some detail as the semester goes on, but today we'll go over some basic concepts.

Basic Terms and Concepts

One of the first and most important questions to be answered when you're trying to learn a function is *why* you're doing it. There are many reasons why you might want to learn a function, and different approaches are more useful for different reasons.

Prediction vs. Inference

In our Taboo Pizza example above, we're trying to make a *prediction* of sales based on marketing spend. In this scenario, you'll assume the error is zero, and predict Y using

$$\hat{Y} = \hat{f}(\mathbf{X}),$$

where $\hat{f}$ represents our estimate for f, and $\hat{Y}$ represents the resulting prediction for Y. In this sort of setting, $\hat{f}$ is typically treated as a *black box*, in that you're not so concerned with understanding the exact form of $\hat{f}$, as long as it works. We want to determined $\hat{f}$ so that it minimizes the *reducible* error in the prediction, keeping in mind there's also some *irreducible* error.

On the other hand, sometimes you're not so much interested in prediction as in *inference*. Suppose, for example, you're trying to sell billboard advertising to Taboo Pizza. In this case, you don't want a complicated model in which it's hard to understand the relative contributions of the different inputs. You want a model that infers the relationship between billboard marketing spend and sales, so you can use it to argue for more spend on billboards.

Parametric vs. Non-Parametric Methods

For a parametric method, we first make as assumption about the form of the function. For example, in *linear regression* we assume the function is linear, which is the form:

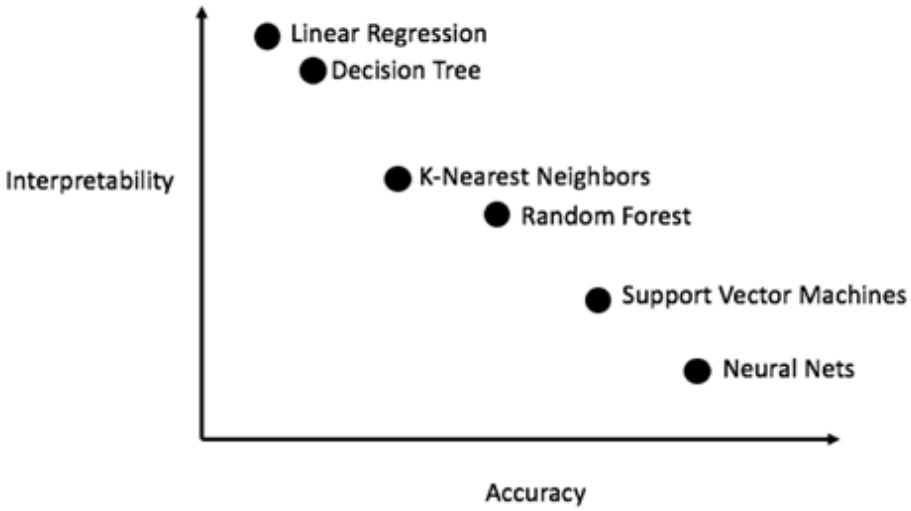
$$f(\mathbf{x}) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

With this assumption, the only thing we need to estimate are the *parameters*, in this case $\beta_0, \beta_1, \ldots, \beta_p$. This typically significantly simplifies the task of estimation, and can be useful when there's not a lot of input data from which to build the model. However, the problem with a parametric model is that the assumption on the form of the function can be wrong - sometimes very wrong.

On the other hand, non-parametric methods do not make explicit assumptions about the functional form of f. Instead these methods seek an estimate of f that gets as close to the data points as possible without being too rough or wiggly. These methods tend to be much more powerful than parametric methods if there's enough data to train them. However, they can sometimes requires *a lot* of data.

Prediction Accuracy vs. Model Interpretability

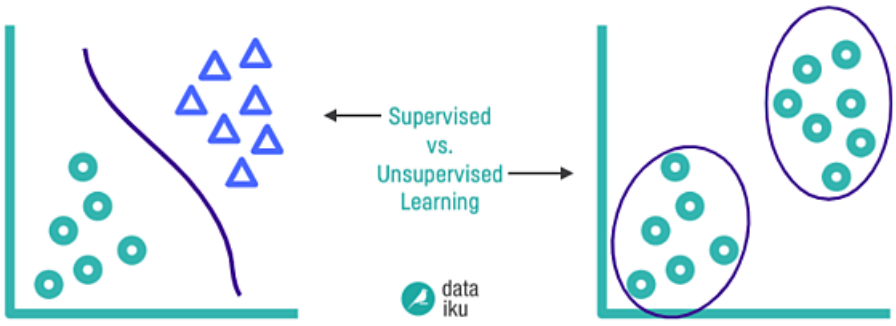
Generally speaking, there's a trade-off between prediction accuracy and model interpretability. Linear regression, for example, is quite easy to interpret, but might not be all that accurate. Neural networks, on the other hand, can be extremely accurate, but are typically impossible to intuitively explain in any detail. The balance between these typically depends on what you need your model to do. If you're working for a hedge fund and you need to predict a stock price, you really want your model to be as accurate as possible. On the other hand, if you're an expert witness who needs to explain an argument to a jury, interpretability may be critical.



Supervised vs. Unsupervised Learning

Supervised learning is learning where for each input in our data there is an associated output. In our Taboo Pizza example above, the inputs would be marketing spend, and the outputs would be sales. Much of what we cover in this class will be problems of this type.

However, sometimes you're given a bunch of data and you want to try to make some sense of it outside of the input / response paradigm. For example, you could have a bunch of data on your customers, and want to try to find customer "personas", or groups within your customers of similar people. For these customers you may have some data, but there's no output from them you're trying to predict. When there's no output you're trying to predict, you're in the realm of *unsupervised* learning. An example of an unsupervised learning technique is *cluster analysis*, or clustering.



Regression vs. Classification

Variables can be classified as either *quantitative* or *categorical*. Quantitative variables have numerical values, like height or weight. Categorical variables non-numeric values like gender or eye-color. A *regression* problem is one in which we're trying to predict a numeric value - like the number of sales from our Taboo Pizza example. A *classification* problem is one in which we're trying to predict a categorical value, like whether somebody orders pizza or chicken wings.

Training Data vs. Test Data

Broadly speaking, the training data is the data you have with which you're building your model, and the test data is the data you want to predict. In our Taboo Pizza model, the training data would be all the historical marketing spend and sales figures, while the test data would be the marketing spend and sales data from the new campaign. The test data isn't used to build the model, but the model's performance on the test data is what really matters.

Sometimes when evaluating multiple models, a subset of the training data (either random or sequential) will be set aside to be used as test data. The rest of the data will be the training data, and the performance of the model on the test data is used as the measure of its success.