September 15, 2017 Lecture 2, Sep. 11, 2017

Some Broad Concepts in Machine Learning

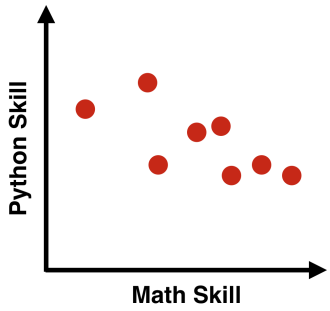
Lecturer: Justin Domke *Scribe: Annamalai Natarajan*

1 Summary

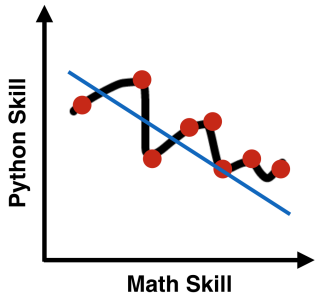
In the previous lecture we covered the five major models in the course with some illustrative examples. In this lecture we cover some broad concepts in Machine Learning. These concepts are widely used in various models of machine learning. Following this we start with regression which is a type of supervised machine learning.

2 Bias/Variance Tradeoff

We introduce bias vs. variance tradeoff in the context of overfitting and underfitting. Bias vs. variance tradeoff also appears under cross validation and loss functions. Consider the example of curve fitting where we given pairs of points *e.g., python skill vs. math skill* as plotted in Figure 2.1a.



(a) Scatterplot of data



(b) Two curves fitted to this data

Figure 2.1

The goal is find a curve that fits the data well. In Figure 2.1b we fit two curves to the data. The first curve (blue) is given by,

$$y = \beta_0 + \beta_1 \times x \tag{2.1}$$

where, x represents math skill and y represents python skill. The second curve (black) is given by,

$$y = \beta_0 + \beta_1 \times x + \beta_2 \times x^2 + ... + \beta_8 \times x^8 \tag{2.2}$$

where, x_1 to x_8 represent the eight data points. The question is which curve will generalize better to new pairs of points? The crucial observation is that the black curve is fitting to noise which is referred to as overfitting. In general fitting to 'big' set of functions results in,

- Low bias
- High variance. In ML, commonly referred to as "overfitting" (but called variance in ESLII)

On the other hand fitting to 'small' set of functions results in,

- High bias. In ML, commonly called "underfitting" (but called bias in ESLII)
- Low variance

3 Basis Expansion/Feature Engineering

Going back to the math vs. python skill data assume our goal is to fit a M^{th} order polynomial function. Given math skill we would like to use this M^{th} order polynomial function to predict python skill. There is lot of overhead involved in fitting a M^{th} order polynomial but with the advantage of reducing bias. Another issue with fitting a M^{th} order polynomial is that one will need to implement a polynomial solver as only linear solvers are readily available in most programming languages. Hence a standard approach in Machine Learning is to first perform basis expansion and then fit a linear model to the expanded set of features. Mathematically,

$$x \in \mathbf{R} \tag{3.1}$$

where, x is a scalar in the real number space. Its basis expansion would like,

$$x' = [x, x^2, x^3, ..., x^M] \tag{3.2}$$

where x' is the expended feature set. We then fit a linear function to x'. In practice we rely on domain knowledge to determine M.

Demo: why distances are not good features in high dimensions. Refer to high_dims.py file

4 Computational/Statistical Tradeoffs

Often it is the case that predictions models are not tractable and we are left with trading off algorithms that we have with algorithms that we want. Given the data in Figure 2.1a it is unlikely for us to precisely know which model will fit the data very well. In practice we simplify this process of model selection by choosing a lower order polynomial which minimizes our chances of overfitting. This simplification leads to model misspecification. To a large extent most if not all models in Machine Learning are misspecified. Another factor that determines model choice is the number of data samples available to fit the model in first place.

5 "That's user specified"

Using Machine Learning in practice involves many different choices like loss functions, hyper-parameter ranges, choice of prediction models, train/test data partitions, feature selection, etc. These choices are determined by the user and are largely influenced by domain knowledge and prior research in the respective fields of study. We refer to these choices as 'that's user specified'.

6 Supervised Learning - Regression

We begin with supervised learning in regression settings. We first present some notation.

- Let $X = [X_1, X_2, ..., X_p]^T$ be a input vector in $\mathbf{R}^p$ dimensional space
- We refer to the p entries as either inputs, features, attributes, predictors
- Let Y be the output variable which is a real number in regression *i.e.* $Y \in \mathbf{R}$
- We are typically given a dataset, D, of X, Y pairs of the form $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ of n examples
- Note that each x_* is of the form $[X_1, X_2, ..., X_p]$ and each $y \in \mathbf{R}$
- In this notation x refers to a specific data case and X refers to generic aspect of an observation. Similarly for y and Y
- To make things concrete assume you are given python and math skill and would like to predict IQ. In this example $p = 2$ (python and math skill) and y is real number (IQ score). Given 100 examples $n = 100$
- We can compactly represent all x's in a $n \times p$ matrix, $\mathbf{X}$, with rows corresponding to examples and columns corresponding to features

6.1 kNN Regression

In kNN regression given an example x, we pick the k nearest neighbors, computes its mean output as the predicted value for the given example. Mathematically this is represented as,

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_{k(x)}} y_i \tag{6.1}$$

where, $N_{k(x)}$ is a set containing the k nearest neighbors for x and y_i is the corresponding output value for each of the neighbors. We typically define 'nearest' by Euclidean distance.