Learning

Samuel Schlesinger

March 15, 2017

1 Problems

1.1 Linear Regression

In linear regression, we wish to relate two sets of data by a linear estimator. Specifically, we have some set of vectors $\{(x_i \in \mathbb{R}^D, y_i \in \mathbb{R})\}_{i \in [N]}$ and we want to find some $\beta \in \mathbb{R}^{D+1}$ such that we minimize the sum of squared errors. From now on, for notational parsimony, I'll assume that $x_i \in \mathbb{R}^{D+1}$ and say, the last dimension, is reserved for a 1, to multiplicatively interact with the extra place in β, which will perform the role of a "bias" in our estimate. We're learning an affine transformation rather than simply a linear one this way. We relate our variables in the following way, where $\epsilon \in \mathbb{R}$ is the error term.

$$y = x\beta + \epsilon$$
$$SSE = \epsilon^T \epsilon$$

We'll go through a quick derivation of the closed formula for β which minimizes SSE. First, we recognize:

$$\epsilon = y - x\beta$$

Thus, we can write SSE as:

$$SSE = (y - x\beta)^T (y - x\beta)$$

We can distribute the transposition:

$$SSE = (y^T - \beta^T x^T)(y - x\beta)$$

Finally, we expand:

$$SSE = y^T y - y^T x\beta - \beta^T x^T y + \beta^T x^T x\beta$$

And by transposing one of the inner terms (not changing its value, as this is the error which is a scalar):

$$SSE = y^T y - 2\beta^T x^T y + \beta^T x^T x\beta$$

So we have a nice formula for the error, but we have to minimize!

$$\frac{\partial SSE}{\partial \beta} = -2x^T y + 2x^T x\beta = 0$$

$$\beta = (x^T x)^{-1} x^T y$$

We have found our β in closed form, so then my Haskell code initially reflected this. That being said, once one observes a property of β its clear that something better can be done computationally. Critically, $x^T x$ is always going to be positive definite, so we can use the Cholesky factorization and get a matrix $L^* L = x^T x$ and write:

$$\beta = (L^* L)^{-1} x^T y = (L^{-1})(L^{-1})^* x^T y$$

The property of L which is important here is that it is triangular, making the inversion much more numerically stable than it otherwise could be. This becomes incredibly important and because of this implementation change I went from being able to deal with thousands of data points of dimension 3, now I can handle millions of dimension 100.

1.2 KMeans Clustering

When we kmeans-cluster data, we are given a set of data points and an integer:

$$X = \{x_i \in \mathbb{R}^D\}_{i \in [N]}, k \in \mathbb{N}, k \leq N$$

Our objective is to produce $S = \{S_i \subset X\}_{i \in [k]}$ a partition of X with means $\mu_i = \frac{\sum_{x \in S_i} x}{|S_i|}$ such that it minimizes the kmeans error function:

$$KME(S) = \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2$$

Solving this problem is not easy, in fact its NP-hard if you squint at it long enough, so we'll use a simple greedy heuristic. Lets assume that we have some candidate set of μ_i, we can use these to generate a clustering by taking $S_i = \{x \in X \mid argmin_j d(x, \mu_j) = i\}$. That being said, we must then check if these μ_i actually are the means for these clusters, and thus we must check that μ_i truly is the average of S_i. Thus we compute the new mean of S_i, μ_i', and we check if $\mu_i' = \mu_i$. If this is the case for all i, we are finished, but if it is not, we must undergo the hassle of reconstructing S, as a different partition will result for the new means we've found. This process repeats and repeats and repeats until it reaches a stable point, which we say is the kmeans clustering of the dataset according to our heuristic. It turns out this process works well if you choose the points right, and there are a variety of ways to do this, so I've left this choice up to the user of the algorithm. In the test, I use a simple uniform choice selection, which works decently well in practice but lacks some perks for theory. This algorithm performs much more poorly than linear regression

simply because of the nature of the problem, and because I also wasn't able to delegate enough of this work to the fast processes in hmatrix. I really want to give this another whack sometime and make it much more efficient.