CS 15-859: Algorithms for Big Data Fall 2019

Lecture 1.1 — 9/5/2019

Prof. David Woodruff *Scribe: Sahil Hasan*

We have massive data sets in today's world. Some notable examples are internet traffic data and financial data. Each of these have their own issues: routers cannot store all internet traffic that goes through them, but they can try to calculate interesting and useful statistics. Financial data is far too large to be utilized with polynomial time algorithms.

The extension of that is that algorithms should be linear time or less, and we usually make this trade-off by using a randomized approximation.

1 Regression

Definition. Statistical method to study dependencies between variables in the presence of noise

We will be focusing on linear regression for the purposes of this lecture:

Definition. Statistical method to study **linear** dependencies between variables in the presence of noise

A good example of linear regression is Ohm's law: $V = R \cdot I$, where we don't know R and are trying to estimate it. This in essence translates to finding the linear function that best fits the data as we see it. The definition of best fit varies and we will explore multiple definitions over this course.

We also define the standard setting of a linear regression:

- A measured variable: b
- A set of predictor variables: $a_1 \dots a_d$
- Error: ϵ (we make no assumptions about error to maintain generality)
- Model parameters we want to learn: $x_0 \dots x_d$, where we can assume $x_0 = 0$ without loss of generality.

This culminates on the assumption that $b = x_0 + a_1x_1 + \cdots + a_dx_d + \epsilon$, where we are trying to solve for the various x values. By extension, if we have n observations, we define A as an $n \times d$ matrix, and b as an n dimensional vector, where n is the number of observations and d is the number of predictor variables. Then we can define linear regression as follows:

Definition. Find x^* such that $Ax^* = b$ or as close as possible

In particular, we will focus on an over constrained case where $n >> d$. In that case as A has many more rows than it does columns, we aren't necessarily guaranteed a solution. Thus, we will settle for a reasonably close answer using least squares.

Definition. Find x^* that minimizes Euclidean distance. Or in other words minimizes $|Ax - b|_2^2 = \Sigma(b_i - < A_{i^*}, x >)^2$, where A_{i^*} is the i-th row of A

Least Squares also has multiple desirable statistical properties, such as resiliency to adding Gaussian noise. We can also view least squares geometrically. We see that we can decompose Ax to $A_{*1}x_1 + A_{*2}x_2 + ... + A_{*d}x_d$. This becomes a d-dimensional subspace of $\mathcal{R}^n$. Then geometrically this problem becomes basically the same as determining the projection of the column space of A closest to b.

Now we try to solve off the normal equations:

Definition. $b = Ax' + b'$ where b' is orthogonal to the column span of A

We plug in $b = Ax' + b'$ to $\min_x |Ax - b|_2^2$ which then results in

$$|Ax - Ax' - b'|_2^2$$

From there we know that b' is orthogonal, and thus can simplify this to become

$$|Ax - Ax'|_2^2 + |b'|_2^2$$

We can also have $x = x'$ which will then render $|Ax - Ax'| = 0$. From there we claim that it is optimal if and only if:

$$A^T(Ax - b) = A^T(Ax - Ax') = 0$$

This is because b' is orthogonal to A/A^T, which means that we can get rid of b' as multiplying orthogonal vectors/matrices results in 0. From there we can say if it is optimal we know that

$$A^T A(x - x') = 0$$

Which implies

$$(x - x')^T A^T A(x - x') = 0$$
$$= |A(x - x')|_2^2$$

The final simplification is because $|W|_2^2 = W^T W$ for any vector W. Thus we get the normal equation:

Definition. $A^T Ax = A^T b$ for any optimal x From here we see that $A^T A$ is $d \times d$ and invertible, which is nice when $n >> d$ Thus:

Definition. $x = (A^T A)^{-1} A^T b$

If the columns of A are not linearly independent, the Moore-Penrose pseudoinverse gives a minimum norm solution x. In other words, we want the minimum length x.

2 Moore-Penrose pseudoinverse

We define singular value decomposition (SVD) as the following:

Definition. $A = U \cdot \Sigma \cdot V^T$

These have the following properties:

- U has orthonormal columns
- Σ is diagonal with non-increasing non-negative entries down the diagonal
- V^T has orthonormal rows

The runtime of SVD is $O(nd^2)$. There are tricks we can do to make it faster, as will be discussed later. It is important to note that $n >> d^2$ so really it is the n value that we don't like. We then proceed to define the pseudoinverse as follows

Definition. $A^- = V\Sigma^{-1}U^T$ where Σ^{-1} is a diagonal matrix where the i-th diagonal entry is either the inverse of the i-th diagonal entry in Σ (i.e. $\Sigma_{ii} \to \frac{1}{\Sigma_{ii}}$) or 0 if the entry is 0. We claim that $\min_x |Ax - b|_2^2$ is not unique when columns of A are linearly independent, but $x = A^-b$ has a minimum norm. We can show this by doing the following. First we establish $x = A^-b$ and $A^T = V\Sigma^T U^T = V\Sigma U^T$. From there we can plug this in to the normal equation:

$$A^A x = A^T b$$
$$(V\Sigma U^T)(U\Sigma V^T)(V\Sigma^{-1}U^T)b = A^T A(A^- b)$$
$$(V\Sigma^2\Sigma^{-1}U^T)b = A^T A(A^- b)$$
$$(V\Sigma U^T)b = A^T b$$

After this we claim:

Claim 1. Any optimal solution x has the form $A^-b + (I - V'(V')^T)z$ for any arbitrary d-dimensional vector z where $(V')^T$ corresponds to the rows i of V^T for which $\Sigma_{i,i} > 0$

We see that $(V')(V')^T$ is what we call a projection matrix, where it takes a vector and projects it onto the column span of V'. Thus, by extension $(I - V'(V')^T)$ projects onto the complement of V'. Thus we can continue on-wards and claim

Claim 2. For any z, $A(I - V'(V')^T)z = 0$

This is because the first k (where k is the rank) of $/Sigma$ are multiplied by the 0 in V', and the final $d - k$ are multiplied by 0 in Σ. Geometrically this means as you go through all possible z you get a subspace of dimension $d - k$ that you shift by A^-b. We know that A^-b is a vector in the column span of V', as $A^- = V\Sigma^{-1}U^T$ which then implies $A^-b = V\Sigma^{-1}U^Tb$. We know that $V\Sigma^{-1}$ forces b to only be in the top k rows of V^T. Beyond that, by definition $(I - V'(V')^T)$ projects onto the complement of V', which makes the vector orthogonal, and as such is the minimum norm.

With regards to time complexity, we see that $x = A^-b$ is the biggest step computationally. This is naively $O(nd^2)$ but can be sped up to $O(nd^{1.376})$. But this is not good enough, and we want faster running time!

3 Sketching and Subspace Embeddings

We want to find an approximate solution x to $\min_x|Ax - b|_2$. We do that by finding x' for which $|Ax' - b|_2 \leq (1 + \epsilon)\min_x |Ax - b|_2$ with high probability. It is important to note that the error we allow is relative error rather than an additive error.

We define the algorithm into the following steps:

- Draw random matrix S from $k \times n$ random family of matrices where $k >> n$
- Compute SA and Sb (note that computing SA is $O(nkd)$)
- Solve $\min_x' |(SA)x - Sb|_x$ where $x' = (SA)^{-}Sb$

We see that lots of random families of matrices work. One example is S such that $\frac{d}{\epsilon^2} \times n$ of i.i.d Normal random variables. We do this via subspace embedding. Let $k = O(\frac{d}{\epsilon^2})$, and let $S = k \times n$ matrix of i.i.d normal $N(0, \frac{1}{k})$ random variables. We want to show that with high probability, $|SAx|_2 = (1 \pm \epsilon)|Ax|_2$. It is important to note that this translates to preserving the length of an infinite number of vectors not just one particular x. The next set of notes from part two of the lecture will show the proof regarding this.