Stat260/CS294: Spectral Graph Methods **Lecture 25 - 04/23/2015**

Lecture: Laplacian solvers (1 of 2)

Lecturer: Michael Mahoney *Scribe: Michael Mahoney*

Warning: these notes are still very rough. They provide more details on what we discussed in class, but there may still be some errors, incomplete/imprecise statements, etc. in them.

25 Overview

We have seen problems that can be written in the form of a system of linear equations with Laplacian constraint matrices, i.e.,

$$Lx = b.$$

For example, we saw this with the various semi-supervised learning methods as well as with the MOV weakly-local spectral method. In some cases, this arises in slightly modified form, e.g., as an augmented/modified graph and/or if there are additional projections (e.g., the Zhou et al paper on "Learning with labeled and unlabeled data on a directed graph," that is related to the other semi-supervised methods we discussed, does this explicitly). Today and next time we will discuss how to solve linear equations of this form.

25.1 Basic statement and outline

While perhaps not obvious, solving linear equations of this form is a useful *algorithmic primitive*—like divide-and-conquer and other such primitives—much more generally, and thus there has been a lot of work on it in recent years.

Here is a more precise statement of the use of this problem as a primitive.

Definition 1. *The* Laplacian Primitive *concerns systems of linear equations defined by Laplacian constraint matrices:*

- INPUT*: a Laplacian $L \in \mathbb{R}^{n \times n}$, a vector $b \in \mathbb{R}^n$ such that $\sum_{i=1}^n b_i = 0$, and a number $\epsilon > 0$.*
- OUTPUT*: a vector $\tilde{x}_{opt} \in \mathbb{R}^n$ such that $\|\tilde{x}_{opt} - L^\dagger v\|_L \leq \|L^\dagger b\|_L$, where for a vector $z \in \mathbb{R}^n$ the L-norm is given by $\|z\|_L = \|z^T L z\|_2$.*

While we will focus on linear equations with Laplacian constraint matrices, most of the results in this area hold for a slightly broader class of problems. In particular, they hold for any linear system $Ax = b$, where A is an SDD (symmetric diagonally dominant) matrix (i.e., that the diagonal entry of each row is larger, or not smaller, than the sum of the absolute values of the off-diagonal entries in that row). The reason for this is that SDD systems are linear-time reducible to Laplacian linear systems via a construction that only doubles the number of nonzero entries in the matrix.

As mentioned, the main reason for the interest in this topic is that, given a fast, e.g., nearly linear time algorithm, for the Laplacian Primitive, defined above, one can obtain a fast algorithm for all sorts of other basic graph problems. Here are several examples of such problems.

- Approximate Fiedler vectors.
- Electrical flows.
- Effective resistance computations.
- Semi-supervised learning for labeled data.
- Cover time of random walks.
- Max flow and min cut and other combinatorial problems.

Some of these problems we have discussed. While it might not be surprising that problems like effective resistance computations and semi-supervised learning for labeled data can be solved with this primitive, it should be surprising that max flow and min cut and other combinatorial problems can be solved with this primitive. We won't have time to discuss this in detail, but some of the theoretically fastest algorithms for these problems are based on using this primitive.

Here is a statement of the basic result that led to interest in this area.

Theorem 1 (ST)**.** *There is a randomized algorithm for the Laplacian Primitive that runs in expected time* $O\left(m \log^{O(1)}(n) \log(1/\epsilon)\right)$*, where* n *is the number of nodes in* L*,* m *is the number of nonzero entries in* L*, and* ϵ *is the precision parameter.*

Although the basic algorithm of ST had something like the 50^{th} power in the exponent of the logarithm, it was a substantial theoretical breakthrough, and since then it has been improved by KMP to only a single log, leading to algorithms that are practical or almost practical. Also, although we won't discuss it in detail, many of the local and locally-biased spectral methods we have discussed arose out of this line of work in an effort to develop and/or improve this basic result.

At a high level, the basic algorithm is as follows.

1. Compute a sketch of the input by sparsifying the input graph.
2. Use the sketch to construct a solution, e.g., by solving the subproblem with any black box solver or by using the sketch as a preconditioner for an iterative algorithm on the original problem.

Thus, the basic idea of these methods is very simple; but to get the methods to work in the allotted time, and in particular to work in nearly-linear time, is very complicated.

Today and next time, we will discuss these methods, including a simple but slow method in more detail and a fast but complicated method in less detail.

- **Today.** We will describe a simple, non-iterative, but slow algorithm. This algorithm provides a very simple version of the two steps of the basic algorithm described above; and, while slow, this algorithm highlights several basic ideas of the more sophisticated versions of these methods.

- **Next time.** We will describe a fast algorithm provides a much more sophisticated implementation of the two steps of this basic algorithm. Importantly, it makes nontrivial use of combinatorial ideas and couples the linear algebra with combinatorial preconditioning in interesting ways.

25.2 A simple slow algorithm that highlights the basic ideas

Here, we describe in more detail a very simple algorithm to solve Laplacian-based linear systems. It will be good to understand before we get to the fast but more complicated versions of the algorithm.

Recall that $L = D - W = B^T W B$ is our Laplacian, where B is the $m \times n$ edge-incidence matrix, and where W is an $m \times m$ edge weight matrix. In particular, note that $m > n$ (assume the graph is connected to avoid trivial cases), and so the matrix B is a *tall* matrix.

Here is a restatement of the above problem.

Definition 2. *Given as input a Laplacian matrix $L \in \mathbb{R}^{n \times n}$, a vector $b \in \mathbb{R}^n$, compute*

$$argmin_{x \in \mathbb{R}^n} \|Lx - b\|_2.$$

The minimal ℓ_2 norm x_{opt} is given by $x_{opt} = L^\dagger b$, where $L^\dagger$ is the Moore-Penrose generalized inverse of L.

We have reformulated this as a regression since it makes the proof below, which is based on RLA (Randomized Linear Algebra) methods, cleaner.

The reader familiar with linear algebra might be concerned about the Moore-Penrose generalized inverse since, e.g., it is typically not well-behaved with respect to perturbations in the data matrix. Here, the situation is particularly simple: although L is rank-deficient, (1) it is invertible if we work with vectors $b \perp \vec{1}$, and (2) because this nullspace is particular simple, the pathologies that typically arise with the Moore-Penrose generalized inverse do *not* arise here. So, it isn't too far off to think of this as the inverse.

Here is a simple algorithm to solve this problem. This algorithm takes as input L, b, and ϵ; and it returns as output a vector $\tilde{x}_{opt}$.

1. Form B and W, define $\Phi = W^{1/2} B \in \mathbb{R}^{m \times n}$, let $U_\Phi \in \mathbb{R}^{m \times n}$ be an orthogonal matrix spanning the column space of Φ, and let $(U_\Phi)_{(i)}$ denote the i^{th} row of U_Φ.

2. Let p_i, for $i \in [n]$ such that $\sum_{i=1}^n p_i = 1$ be given by

$$p_i \geq \beta \frac{\| (U_\Phi)_{(i)} \|_2^2}{\|U_\Phi\|_F^2} = \frac{\beta}{n} \| (U_\Phi)_{(i)} \|_2^2 \tag{1}$$

   for some value of $\beta \in (0, 1]$. (Think of $\beta = 1$, which is a legitimate choice, but the additional flexibility of allowing $\beta \in (0, 1)$ will be important in the next class.)

A key aspect of this algorithm is that the sketch is formed by choosing elements of the Laplacian with the probabilities in Eqn. (1); these quantities are known as the statistical leverage scores, and they are of central interest in RLA. Here is a definition of these scores more generally.

Definition 3. *Given a matrix $A \in \mathbb{R}^{m \times n}$, where $m > n$, the i^{th} leverage score is*

$$(P_A)_{ii} = \left(U_A U_A^T\right)_{ii} = \| \left(U_A\right)_{ii} \|_2^2,$$

i.e., it is equal to the diagonal element of the projection matrix onto the column span of A.

Here is a definition of a seemingly-unrelated notion that we talked about before.

Definition 4. *Given $G = (V, E)$, a connected, weighted, undirected graph with n nodes, m edges, and corresponding weights $w_e \geq 0$, for all $e \in E$, let $L = B^T W B$. Then, the effective resistance R_e across edge $e \in E$ are given by the diagonal elements of the matrix $R = BL^{\dagger}B$.*

Here is a lemma relating these two quantities.

Lemma 1. *Let $\Phi = W^{1/2}B$ denote the scaled edge-incidence matrix. If ℓ_i is the leverage score of the i^{th} row of Φ, then $\frac{\ell_i}{w_i}$ is the effective resistance of the i^{th} edge.*

Proof. Consider the matrix

$$P = W^{1/2}B\left(B^T W B\right)^{\dagger} B^T W^{1/2} \in \mathbb{R}^{m \times m},$$

and notice that $P = W^{1/2} R W^{1/2}$ is a rescaled version of $R = BL^{\dagger}B$, whose diagonal elements are the effective resistances. Since $\Phi = W^{1/2}B$, it follows that

$$P = \Phi\left(\Phi^T \Phi\right)^{\dagger} \Phi^T.$$

Let U_Φ be an orthogonal matrix spanning the columns of Φ. Then, $P = U_\Phi U_\Phi^T$, and so

$$P_{ii} = \left(U_\Phi U_\Phi^T\right)_{ii} = \| \left(U_\Phi\right)_{(i)} \|_2^2,$$

which establishes the lemma. □

So, informally, we sparsify the graph by biasing our random sampling toward edges that are "important" or "influential" in the sense that they have large statistical leverage or effective resistance, and then we use the sparsified graph to solve the subproblem.

Here is the main theorem for this algorithm.

Theorem 2. *With constant probability, $\|x_{opt} - \tilde{x}_{opt}\|_L \leq \epsilon \|x_{opt}\|_L$.*

Proof. The main idea of the proof is that we are forming a sketch of the Laplacian by randomly sampling elements, which corresponds to randomly sampling rows of the edge-incidence matrix, and that we need to ensure that the corresponding sketch of the edge-incidence matrix is a so-called subspace-preserving embedding. If that holds, then the eigenvalues of the edge-incidence matrix and it's sketch are close, and thus the eigenvalues of the Laplacian are close, and thus the original Laplacian and the sparsified Laplacian are "close," in the sense that the quadratic form of one is close to the quadratic form of the other.

Here are the details.

By definition,

$$\|x_{opt} - \tilde{x}_{opt}\|_L^2 = \left(x_{opt} - \tilde{x}_{opt}\right)^T L \left(x_{opt} - \tilde{x}_{opt}\right).$$

Recall that $L = B^T W B$, that $x_{opt} = L^\dagger b$, and that $\tilde{x}_{opt} = \tilde{L}^\dagger b$. So,

$$\begin{aligned} \|x_{opt} - \tilde{x}_{opt}\|_L^2 &= (x_{opt} - \tilde{x}_{opt})^T B^T W B (x_{opt} - \tilde{x}_{opt}) \\ &= \|W^{1/2} B (x_{opt} - \tilde{x}_{opt})\|_2^2 \end{aligned}$$

Let $\Phi \in \mathbb{R}^{m \times n}$ be defined as $\Phi = W^{1/2} B$, and let its SVD be $\Phi = U_\Phi \Sigma_\Phi V_\Phi^T$. Then

$$L = \Phi^T \Phi = V_\Phi \Sigma_\Phi^2 V_\Phi^T$$

and

$$x_{opt} = L^\dagger b = V_\Phi \Sigma_\Phi^{-2} V_\Phi^T b.$$

In addition

$$\tilde{L} = \Phi^T S^T S \Phi = (S\Phi)^T (S\Phi)$$

and also

$$\tilde{x}_{opt} = \tilde{L}^\dagger b = (S\Phi)^\dagger (S\Phi)^{T\dagger} b = \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^\dagger \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^{T\dagger} b$$

By combining these expressions, we get that

$$\begin{aligned} \|x_{opt} - \tilde{x}_{opt}\|_L^2 &= \|\Phi (x_{opt} - \tilde{x}_{opt})\|_2^2 \\ &= \|U_\Phi \Sigma_\Phi V_\Phi^T \left(V_\Phi \Sigma_\Phi^{-2} V_\Phi^T - \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^\dagger \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^{T\dagger}\right) b\|_2^2 \\ &= \|\Sigma_\Phi^{-1} V_\Phi^T b - \Sigma_\Phi \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^\dagger \left(SU_\Phi \Sigma_\Phi V_\Phi^T\right)^{T\dagger} V_\Phi b\|_2^2 \end{aligned}$$

Next, we note the following:

$$\mathbb{E}\left[\|U_\Phi^T S^T S U_\Phi - I\|_2\right] \leq \sqrt{\epsilon},$$

where of course the expectation can be removed by standard methods. This follows from a result of Rudelson-Vershynin, and it can also be obtained as a matrix concentration bound. This is a key result in RLA, and it holds since we are sampling $O\left(\frac{n}{\epsilon} \log\left(\frac{n}{\epsilon}\right)\right)$ rows from U according to the leverage score sampling probabilities.

From standard matrix perturbation theory, it thus follows that

$$\left|\sigma_i\left(U_\Phi^T S^T S U_\Phi\right) - 1\right| = \left|\sigma_i^2 (SU_\Phi) - 1\right| \leq \sqrt{\epsilon}.$$

So, in particular, the matrix SU_Φ has the same rank as the matrix U_Φ. (This is a so-called subspace embedding, which is a key result in RLA; next time we will interpret it in terms of graphic inequalities that we discussed before.)

In the rest of the proof, let's condition on this random event being true.

Since SU_Φ is full rank, it follows that

$$(SU_\Phi \Sigma_\Phi)^\dagger = \Sigma_\Phi^{-1} (SU_\Phi)^\dagger.$$

So, we have that

$$\begin{aligned} \|x_{opt} - \tilde{x}_{opt}\|_L^2 &= \|\Sigma_\Phi^{-1} V_\Phi^T b - (SU_\Phi)^\dagger (SU_\Phi)^{T\dagger} \Sigma_\Phi^{-1} V_\Phi^T b\|_2^2 \\ &= \|\Sigma_\Phi^{-1} V_\Phi^T b - V_\Omega \Sigma_\Omega^{-2} V_\Omega^T \Sigma_\Phi^{-1} V_\Phi^T b\|_2^2, \end{aligned}$$

where the second line follows if we define $\Omega = SU_\Phi$ and let its SVD be

$$\Omega = SU_\Phi = U_\Omega \Sigma_\Omega V_\Omega^T.$$

Then, let $\Sigma_\Omega^{-1} = I + E$, for a diagonal error matrix E, and use that $V_\Omega^T V_\Omega = V_\Omega V_\Omega^T = I$ to write

$$\begin{aligned}
\|x_{opt} - \tilde{x}_{opt}\|_L^2 &= \|\Sigma_\Phi^{-1} V_\Phi^T b - V_\Omega (I+E) V_\Omega^T \Sigma_\Phi^{-1} V_\Phi^T b\|_2^2 \\
&= \|V_\Omega E V_\Omega^T \Sigma_\Phi^{-1} V_\Phi^T b\|_2^2 \\
&= \|E V_\Omega^T \Sigma_\Phi^{-1} V_\Phi^T b\|_2^2 \\
&\leq \|E V_\Omega^T\|_2^2 \|\Sigma_\Phi^{-1} V_\Phi^T b\|_2^2 \\
&= \|E\|_2^2 \|\Sigma_\Phi^{-1} V_\Phi^T b\|_2^2
\end{aligned}$$

But, since we want to bound $\|E\|$, note that

$$|E_{ii}| = \left|\sigma_i^{-2}(\Omega) - 1\right| = \left|\sigma_i^{-1}(SU_\Phi) - 1\right|.$$

So,

$$\|E\|_2 = \max_i \left|\sigma_i^{-2}(SU_\Phi) - 1\right| \leq \sqrt{\epsilon}.$$

So,

$$\|x_{opt} - \tilde{x}_{opt}\|_L^2 \leq \epsilon \|\Sigma_\Phi^{-1} V_\Phi^T b\|_2^2.$$

In addition, we can derive that

$$\begin{aligned}
\|x_{opt}\|_L^2 &= x_{opt}^T L x_{opt} \\
&= \left(W^{1/2} B x_{opt}\right)^T \left(W^{1/2} B x_{opt}\right) \\
&= \|\Phi x_{opt}\|_2^2 \\
&= \|U_\Phi \Sigma_\Phi V_\Phi^T V_\Phi \Sigma_\Phi^{-2} V_\Phi^T b\|_2^2 \\
&= \|\Sigma_\Phi^{-1} V_\Phi^T b\|_2^2.
\end{aligned}$$

So, it follows that

$$\|x_{opt} - \tilde{x}_{opt}\|_L^2 \leq \epsilon \|x_{opt}\|_L^2,$$

which establishes the main result. □

Before concluding, here is where we stand. This is a very simple algorithm that highlights the basic ideas of Laplacian-based solvers, but it is not fast. To make it fast, two things need to be done.

- We need to compute or approximate the leverage scores quickly. This step is very nontrivial. The original algorithm of ST (that had the $\log^{50}(n)$ term) involved using local random walks (such as what we discussed before, and in fact the ACL algorithm was developed to improve this step, relative to the original ST result) to construct well-balanced partitions in nearly-linear time. Then, it was shown that one could use effective resistances; this was discovered by SS independently of the RLA-based method outlined above, but it was also noted that one could call the nearly linear time solver to approximate them. Then, it was shown that one could relate it to spanning trees to construct combinatorial preconditioners. If this step was done very carefully, then one obtains an algorithm that runs in nearly linear time. In particular, though, one needs to go beyond the linear algebra to map closely to the combinatorial properties of graphs, and in particular find low-stretch spanning trees.

- Instead of solving the subproblem on the sketch, we need to use the sketch to create a preconditioner for the original problem and then solve a preconditioned version of the original problem. This step is relatively straightforward, although it involves applying an iterative algorithm that is less common than popular CG-based methods.

We will go through both of these in more detail next time.