INCRES with hard constraint on the size of the clusters

May 23, 2015

We present an algorithm for the optimization problem

$$\boxed{\begin{aligned}&\text{Minimize } \sum_r \mathrm{Cut}(A_r, A_r^c) \qquad (1)\\ &\text{such that } S_r \leq |A_r| \leq T_r \qquad (2)\end{aligned}}$$

where S_r and T_r are given size constraints on cluster r. If the similarity matrix of the graph is positive definite, then the algorithm is monotonic. We also present a stochastic gradient descent version of this algorithm which reduces to the incremental reseeding strategy.

If the similarity matrix of the graph is not positive definite (and in practice it is never), then we smooth the graph using a random walk: for example, if L is the graph laplacian of the original graph, then the smoothed graph

$$W = \left(I + \frac{\alpha}{1-\alpha} L\right)$$

is symmetric positive definite.

0.1 Maximizing a convex function with simple explicit gradient ascent

Suppose $E(f)$ is a smooth convex functional and we want to maximize it over some constraint set $\mathcal{C}$. Note that:

$$E(f^{k+1}) \geq E(f^k) + \left\langle \nabla E(f^k), f^{k+1} - f^k \right\rangle$$

So as long as $f^{k+1} = f^k + v$ where v is an ascent direction, that is $\left\langle \nabla E(f^k), v \right\rangle \geq 0$, then we have $E(f^{k+1}) \geq E(f^k)$. So the following algorithm is monotonic:

$$\text{Given } f^k \in \mathcal{C}, \text{ find } f^{k+1} \in \mathcal{C} \text{ such that } \left\langle \nabla E(f^k), f^{k+1} - f^k \right\rangle \geq 0$$

Obviously, in the case where

$$E(f_1, \ldots, f_r) = \sum_{r=1}^{R} E_r(f_r) \tag{3}$$

the algorithm becomes

$$\text{Given } (f_1^k, \ldots, f_R^k) \in \mathcal{C}, \text{ find } (f_1^{k+1}, \ldots, f_R^{k+1}) \in \mathcal{C} \text{ such that } \sum_r \left\langle \nabla E_r(f^k), f_r^{k+1} - f^k \right\rangle \geq 0 \tag{4}$$

0.2 Minimizing Cut under size constraints

We use the above algorithm on the problem:

$$\boxed{\begin{array}{l}\text{Minimize } \displaystyle\sum_r \mathrm{Cut}(A_r, A_r^c) \\ \text{such that } S_r \le |A_r| \le T_r\end{array}} \qquad \begin{array}{r}(5)\\(6)\end{array}$$

Let $F = (f_1, \dots, f_k)$ be the indicator matrix of a partition

$$f_{ir} = 1 \text{ if vertex i belongs to class r, zero otherwise}$$

Then the sum of the cut can be written as

$$\sum_r \mathrm{Cut}(A_r, A_r^c) = \sum_r \sum_{ij} w_{ij} f_{ir}(1 - f_{jr}) = \sum_r \sum_{ij} w_{ij} f_{ir} - \sum_r \sum_{ij} w_{ij} f_{ir} f_{jr} = \sum_{ij} w_{ij} - \sum_r \sum_{ij} w_{ij} f_{ir} f_{jr}$$

so problem (5)-(6) is equivalent to:

$$\text{Maximize } \sum_r \langle f_r, W f_r \rangle \tag{7}$$

$$\text{such that } S_r \le \sum_i f_{ir} \le T_r \tag{8}$$

$$\text{and } F = (f_1, \dots, f_k) \text{ is the indicator matrix of a partition} \tag{9}$$

This can also be written in "Trace notation"

$$\text{Maximize } \mathrm{Tr}(F^T W F^k) \tag{10}$$

$$\text{such that } S_r \le \sum_i F_{ir} \le T_r \tag{11}$$

$$\text{and } F \text{ is the indicator matrix of a partition} \tag{12}$$

Now let's denote by $\mathcal{C}$ the set of constraints (8)-(9). Then (4) becomes

$$\boxed{\begin{array}{c}\text{Given } (f_1^k, \dots, f_R^k) \in \mathcal{C}, \text{ find } (f_1^{k+1}, \dots, f_R^{k+1}) \in \mathcal{C} \text{ such that} \\ \displaystyle\sum_r \left\langle W f_r^k, f_r^{k+1} \right\rangle \ge \sum_r \left\langle W f_r^k, f_r^k \right\rangle\end{array}} \qquad \begin{array}{r}(13)\\(14)\end{array}$$

Let $H^k = W F^k$ be the "heat bump" n-by-R matrix and let $\pi^k : \{1, \dots, n\} \to \{1, \dots, R\}$ be the membership function associated with the partition F^k (that is $\pi^k(i) = r$ means vertex i belongs to class r). Then equation (14) can be written

$$\sum_i H^k_{i,\pi^{k+1}(i)} \ge \sum_i H^k_{i,\pi^k(i)} \tag{15}$$

Let $\Delta \subset V$ be the vertices who want to switch class, that is:

$$\Delta = \{i \in V : \pi^k(i) \notin \arg\max_r H^k_{ir}\}$$

Note that given any nonempty subset of $\tilde{\Delta} \subset \Delta$, the partition

$$\pi^{k+1}(i) = \begin{cases} \arg\max_r H^k_{ir} & \text{if } i \in \tilde{\Delta} \\ \pi^k(i) & \text{otherwise} \end{cases}$$

satisfies (15) with strict inequality. The question now is to find a subset $\tilde{\Delta} \subset \Delta$ such that the new partition still satisfies the size constraints. We also want this subset $\tilde{\Delta}$ to be as big as possible, otherwise the algorithm is going to stagnate.

Let me summarize: Δ is the subset of vertices which would change class under plain thresholding. The problem is that if all vertices change classes typically we will break the balance constraint. The nice thing is that we don't need all the vertices in Δ to change class: any subset of Δ, even if it consist on only one vertex, will lead to a decrease in the cut value. The goal is then to look for the subset $\tilde{\Delta} \subset \Delta$ that leads to the biggest gain in equation (15) while not breaking the size constraint. So this lead to the optimization problem:

$$\text{Find } \tilde{\Delta} \subset \Delta \text{ that maximize } \sum_i H^k_{i,\pi^{k+1}(i)} \tag{16}$$

$$\text{while not breaking the size constraint} \tag{17}$$

And this problem can be solved with a small linear program which has $R(R-1)$ unknown (the unknowns are the number of vertices that are allowed to move from class k to class r). This is what they do in the facebook paper.

0.3 The algorithm in short

So the full algo, without reseeding is exactly a MBO scheme with size constraints enforced by the facebook paper strategy. At each step we do

- Compute $H^k = WF^k$ where F^k is the 0-1 indicator function of the current partition (and not the normalized one as in our current algorithm).
- To get F^{k+1} we threshold the matrix H^k, but before to change the class of the vertices who are asking to change, we pause and solve a small linear program. This linear program determines which vertex will be allowed to change class. We will accept only the most beneficial change and we will guarantees that the size constraints are satisfied.

If W is positive definite, then this gives a monotonic algorithm for the minimization problem (5)-(6).

0.4 Incremental reseeding as a stochastic gradient ascent

Note first that the matrix $H^k = WF^k$ is just the gradient of the energy $E(F) = \sum_r \langle f_r, Wf_r \rangle$:

$$\nabla E(F^k) = H^k = WF^k$$

And that the algorithm is simply looking for a feasible F^{k+1} that satisfies

$$\left\langle \nabla E(F^k) \, , \, F^{k+1} - F^k \right\rangle > 0$$

and that if it finds one, then we have made the cut smaller while maintaining the constraints.

We now propose the following stochastic gradient algorithm. At each step we will look for a feasible F^{k+1} that satisfies

$$\left\langle H(\omega) \, , \, F^{k+1} - F^k \right\rangle > 0$$

where $H(\omega)$ is a random n-by-R matrix satisfying

$$\text{Expectation of } H(\omega) = \nabla E(F^k) \tag{18}$$

We will also have a "cooling schedule" such that the variance shrink to 0 as the algorithm progresses. That is, after a large number of iterations, we will have $H(\omega) = \nabla E(F^k)$ for all ω.

Let $\nabla E_r(F^k)$ denotes the r^{th} column of $\nabla E(F^k)$. Note that

$$\nabla E_r(F^k) = WF_r = \sum_{i \in V} F_{i,r} \vec{W}_i \tag{19}$$

where F_r and W_i denotes the r^{th} column of F and the i^{th} column (or row) of W. Now if we replace the vector $\nabla E_r(F^k)$ with the random vector

$$H_r(\omega) = \frac{|V|}{|S(\omega)|} \sum_{i \in S(\omega)} F_{i,r} \vec{W}_i \tag{20}$$

where $S(\omega) \subset V$ is a subset of the vertices chosen uniformly at random with replacement, and that we will call the "seeds". With this definition of $H_r(\omega)$ we have that (18) hold. Then from this that would be nice if we could say something like: "the value of the cut is expected to descend" or "with high probability the value of the cut will descend".