Experiments in Local Graph Learning

Rajmonda Caceres*[1] and Jeremy Kun†[2]

[1]MIT Lincoln Laboratory
[2]Deparment of Mathematics, Statistics, and Computer Science, University of Illinois at Chicago

Abstract

We present a model for metric learning which learns suitable global metric representations from local

1 Introduction and related work

2 Data

2.1 Synthetic Data Model

Our primary synthetic data model is the Erdos-Renyi stochastic block model, which we presently describe. Given a list of block sizes $\mathbf{n} = (n_1, \dots, n_k)$, and a symmetric k-by-k matrix A with entries in $[0, 1]$, we construct a probability distribution over graphs $G(\mathbf{n}, A)$ on $n = \sum_i n_i$ vertices. The vertices are partitioned into k blocks $B_1, \dots, B_k$, where block B_i has size n_i. We declare that the probability of an edge occuring between a vertex in block B_i and block B_j is given by the (i, j) entry of A.

We use multiple instances of the stochastic block model with varying parameters to capture the notion that one graph representation may accurately describe the connections among entities in one community, while another representation may fare better for another community. Specifically, if we have $k = 2$ blocks, then we can use the following two matrices (with p significantly larger than q) to generate a graph representation which is good for each block.

$$\begin{pmatrix} p & q \\ q & q \end{pmatrix} \qquad \begin{pmatrix} q & q \\ q & p \end{pmatrix}$$

As a sanity check, we will also use instances of this model in which the graph representations describe all communities equally well, as well as representations describing communities poorly (which for us is a standard Erdos-Renyi random graph).

2.2 DBLP: Computer Science Research Database

Our primary real-world data set is a subset of DBLP, a comprehensive online database documenting research in computer science. We extracted a subset of the DBLP database corresponding to researchers who have published at two prominent conferences: the Symposium on the Teory of Computing (STOC) and the Symposium on Foundations of Computer Science (FOCS).

To give a quick summary of this data set: it consists of 3153 researchers and 5234 papers. From this we construct two graph representations of the research community social graph. The first is a *coauthorship*

*rajmonda.caceres@ll.mit.edu
†jkun2@uic.edu

graph, in which the weight of an edge between two researchers is the number of papers they have authored together. The second is a *title similarity graph*, in which the weight of an edge between two researchers is the number of papers they have authored with similar titles. We consider two titles to be similar if, after stripping out non alphanumeric characters and removing case, they contain at least three tokens in common which are not stop words. Our tokenization methods and stop word corpus come from the Natural Language Tool Kit, and we include some additional words that show up often in theoretical computer science titles but bear no information as to the content of the paper (one ubiquitous example of this is the phrase "extended abstract").

3 Experiments

3.1 Experimental setup

The general learning procedure can be described as the following game played by our algorithm. As input, the algorithm is given a set of weighted graphs $H_1, \dots, H_m$ on the same vertex set V, which should be thought of as expert advice. Then in each round t the player algorithm produces a graph $G_t = (V, E_t)$, A randomized event $A(G_t)$ occurs which depends on the chosen graph, and a payoff $Q(G_t, A(G)) \in [0, 1]$ is determined which in general depends both on the graph and the resulting event.

In our experiments we fix the event A as a clustering algorithm, and a payoff function Q as a clustering quality metric. These are both understood to be imperfect proxies for idealized clustering algorithms and clustering quality metrics, but the results of the learning algorithm are nonetheless intriguing. In this paper we fix A to be walktrap clustering [cite], and use modularity [cite] and conductance [cite] as global quality metrics Q.

We specialize this framework to two types of algorithms by: *global learning algorithms* and *local learning algorithms*. In the global setting, the algorithm seeks to learn which input graph H_i or linear combination of the input graphs[1] maximizes the expected quality Q. In the algorithm we present for global learning, each H_i is assigned a weight w_i which is updated multiplicatively, and then edges are drawn i.i.d. from the induced distribution on the H_i.

In the local setting, we still seek the highest possible quality under a global quality metric Q, but here we learn the quality of each edge independently of the quality of the suggestions of the H_i for other, possibly distant edges. We can imagine modifying the game to get a better picture of how this would work. In addition to having a global Q we wish to optimize, we have a second metric Q' which gives an appraisal of the quality of each edge of G_t. In notation, we have a local quality metric $Q'(G_t, A(G_t), e)$ which is evaluated for each edge e in G_t. In total, we can think of Q' as having output in $[0, 1]^{|E_t|}$, or else that there is a more generalized global metric encapsulating both measurements (Q, Q'). Our algorithm maintains a set of weights for each edge e which are updated multiplicatively by the outputs of Q'. When the next G_t is produced, edges are drawn according to the probability distributions induced by the local weights.

The particular Q' we use in this paper is a local clustering coefficient. Specifically, given an edge $e = (v, w)$, we compute the clustering coefficient of the induced subgraph on the vertices in the neighborhood of v and w.

Sampling graphs edge by edge, and evaluating a quality function for each edge, brings up some obvious scaling issues which we discuss in Section [ref]. The main purpose of this section is to show that in the stochastic block model our local learning algorithm outperforms the similar global learning algorithm, both in rate of convergence and quality of the final output. The benefit of the synthetic model is that we have ground truth: the partitions B_i used to construct the graphs.

Interestingly, even though above I said the local learning algorithm wants to optimize Q, our local learning algorithm only uses the local clustering coefficient Q'. I imagine there might be questions as to why this is the case.

[1]Given weighted graphs $H_1, \dots, H_m$ on the same vertex set V and coefficients $\alpha_1, \dots, \alpha_m$, we construct the linear combination $H = \sum_i \alpha_i H_i$ edgewise, so that the weight of an edge e is the linear combination of the weights of e in each H_i, where edges not present are ignored.

Another note: we say the graphs are weighted, but in the synthetic experiments all the graphs used are unweighted.

3.2 Algorithms

```
Data: Weighted graphs $H_1, \dots, H_m$ on the same vertex set $V$, a clustering algorithm $A$, a quality
        metric $Q$, a parameter $\varepsilon$
Result: A graph $G$
Initialize $w_1 = \cdots = w_m = 1$;
while the process has not converged do
    Let $D$ be the distribution over $(H_i)_{i=1}^m$ induced by the $w_i$;
    Let $G$ be the empty graph on $V$;
    for $(v, w) \in V \times V$ do
        Draw some $H_j$ according to $D$;
        Include $e = (v, w)$ as an edge of $G$ if $e$ is an edge of $H_j$, and use the same weight as $H_j$;
    end
    Let $s_i$ be the number of times $H_i$ was used to sample an edge of $G$;
    Cluster $G$ using $A$;
    Set $p = -Q(G, A(G))$;
    for each $i = 1, \dots, m$ do
        Set $w_i = w_i(1 - \varepsilon p s_i/|V|)$
    end
end
```

Algorithm 1: Our global learning algorithm

```
Data: Weighted graphs $H_1, \dots, H_m$ on the same vertex set $V$, a clustering algorithm $A$, a local
        quality metric $Q'$, a parameter $\varepsilon$
Result: A graph $G$
Initialize a vector $\mathbf{w}_{u,v} = \mathbf{1}$ for all $u \neq v \in V$;
while the process has not converged do
    Let $G = (V, E)$ be the empty graph;
    for $u, v \in V, u \neq v$ do
        Draw some $H_j$ according to the distribution given by $\mathbf{w}_{u,v}$;
        Include $e = (v, w)$ as an edge of $G$ if $e$ is an edge of $H_j$, and use the same weight as $H_j$;
    end
    Cluster $G$ using $A$;
    for each edge $e \in E$ do
        Set $p = -Q'(G, A(G), e)$;
        Set $w_{u,v,i} = w_{u,v,i}(1 - \varepsilon p)$
    end
end
```

Algorithm 2: Our local learning algorithm

We present one global learning algorithm and one local learning algorithm. Our local learning algorithm is given by Algorithm **??**. The local learning algorithm is given by Algorithm **??**.

3.3 Synthetic experiments

We present two experiments performed on the stochastic block model and describe the results of the final graph produced by our algorithms.

4 Future directions

Acknowledgments