B-Matching for Embedding

Tony Jebara, Blake Shaw, and Vlad Shchogolev
Columbia University
New York, NY 10027
jebara@cs.columbia.edu

When learning from a dataset of samples, many algorithms begin by forming a graph that captures pairwise affinities between all pairs of points. For example, spectral clustering forms a weighted graph between data-points and then approximates a normalized cut on this graph [4, 2]. Nonlinear manifold learning and embedding methods form a graph from data-points and then proceed to reconstruct that graph's weight matrix using a low rank approximation [5, 6]. Even classifiers can be viewed as separating labels on a weighted graph [1]. Often, however, forming a graph from data is a crucial first step yet might be sub-optimal: either keeping the graph fully connected or greedily building a sparse graph via, for example, k-nearest-neighbor. We propose using b-matching [3], an interesting polynomial-time and optimal algorithm for finding the maximum weight subgraph from a larger graph such that the subgraph has in-degree *and* out-degree equal to b for each node. Meanwhile, k-nearest neighbor only finds the maximum weight subgraph from an original graph that only has out-degree (*or* in-degree) equal to k for each node (a less constrained optimization).

Assume we are given an $N \times N$ matrix $A \in \Re^{N \times N}$ which captures pairwise similarity for each pair of points in an N-point dataset. For instance, we may use a radial basis function kernel between all pairs of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ in the dataset. In that case, we have $A_{ij} = \exp(-\frac{1}{2\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. Typically, this is the first step in forming a graph for learning embedding, spectral clustering or even certain types of classifiers. Subsequently, one can compute k-nearest neighbors which is equivalent to finding a binary matrix $P \in \mathcal{B}$ where $\mathcal{B}$ is the space of binary matrices $\{0,1\}^{N \times N}$ that represents the connectivity of the k-nearest neighbors. This matrix is initialized to zeros and then updated by going through the nodes for $i = 1 \ldots N$ and setting $P_{ij} = P_{ji} = 1$ if A_{ij} is one of the top k values for $j = 1 \ldots N$. Often, k-nearest neighbors is solved using a $\mathcal{O}(kN^2)$ time implementation (faster variants are sometimes possible using ball trees, KD trees and VPN trees). Meanwhile, b-matching (a generalization of 1-matching and the classic Kuhn-Munkres or Hungarian algorithm) is optimally solvable in $\mathcal{O}(bN^3)$ time. However, it also enforces that each point has b-nearest neighbors and that only b (or k) other points can use it as their own neighbors:

$$\max_{P \in \mathcal{B}} \sum_{ij} A_{ij} P_{ij} \;\; s.t. \;\; \sum_i P_{ij} = \sum_j P_{ij} = b$$

Fig. 1. Noisy Tea Pot Images

In Figure 1 we show a preliminary dataset for the b-matching algorithm as it learns a slightly different graph from the tea pot data set with added noise. This dataset contains images of a teapot that is being rotated 360 degrees. We downsampled the dataset to 100 images and added noise to make the problem harder. In Figure 2 we see the graph recovered by both the k nearest neighbors approach and the b-matching approach after we have embedded the graphs in two dimensions using the semi-definite embedding (SDE) approach of [6]. Note how b-matching, unlike k-nearest neighbors, recovers

an embedding which is a circle thus capturing that proper nonlinear manifold that corresponds to a rotation. Varying k for the k-nearest neighbors across $k = 2$, $k = 3$, $k = 4$ and beyond did not improve the situation for k-nearest neighbors.

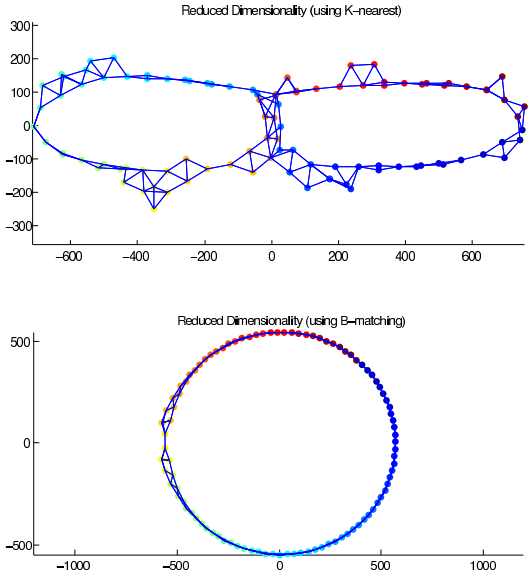

Fig. 2. K-Nearest-Neighbors vs. B-Matching in Semidefinite Embedding

Furthermore, we can now subscribe to the view that learning the graph is part of the optimization of a cost function as opposed to a preprocessing. The equations above suggest that graph structure and connectivity can be represented as a binary $N \times N$ matrix. We can thus interleave graph structure learning with embedding estimation. We perform alternating minimization over the embedding and the graph structure learning (via b-matching) to minimize the overall cost function of interest in SDE. For instance, we iteratively reduce the cost:

$$\min_{K,P} \max_{\beta} tr(K) + \sum_{ij} \beta_{ij} P_{ij} (K_{ii} - 2K_{ij} + K_{jj} - A_{ii} + 2A_{ij} - A_{jj})$$

where P is a binary matrix summing to b row and column-wise, K is the centered positive semidefinite embedding $K \geq 0$ and $\sum_{ij} K_{ij} = 0$ and β is a matrix of Lagrange multipliers forcing the embedding to preserve distances. Finally, it is interesting to view b-matching as a general alternative to k-nearest neighbor in other learning settings. This even includes using b-matching as classifier instead of kNN where points are merely classified by taking votes from the labels of their k nearest neighbors. Despite b-matching's extra computational work, it may still be useful in these settings as well.

References

1. M. Belkin, I. Matveeva, and P. Niyogi. Regularization and Semi-supervised Learning on Large Graphs. COLT 2004.
2. M. Meila, and J. Shi. Learning Segmentation by Random Walks, NIPS, 2000.
3. M. Muller-Hannemann and A. Schwartz, Implementing weighted b-matching algorithms: insights from a computational study, J. Exp. Algorithmics, vol. 5, 2000.
4. A. Ng, M. Jordan and Y. Weiss. On spectral clustering: Analysis and an algorithm, NIPS, 2001.
5. S. Roweis and L. Saul (2000). Nonlinear Dimensionality Reduction by Locally Linear Embedding, *Science*, vol. 290, no. 5500.
6. K.Q. Weinberger, F. Sha, and L.K. Saul, Learning a kernel matrix for nonlinear dimensionality reduction, International Conference on Machine Learning, 2004.