Fast Approximate Graph Matching via Spectral Methods

Geoffrey Iyer, Jocelyn Chanussot, Andrea Bertozzi

February 14, 2018

1 The Method

Given two datasets X, Y, we will define the corresponding undirected, weighted graphs $G = (X, W_X)$, $H = (Y, W_Y)$. Here the nodes of G (resp. H) correspond to the points in X (resp. Y). The weight matrix W_X has size $|X| \times |X|$, and the value $W_X(i, j)$ represents the similarity between nodes $X(i), X(j)$ (and similar for W_Y). There are many possible definitions of W_X in the literature, but the most common involve a RBF kernel, as we use in our experiments (section 2, equation 10).

1.1 The Weighted Graph Matching Problem (WGMP)

Let $G = (X, W_X)$, $H = (Y, W_Y)$ be undirected, weighted graphs. Here X, Y represent the nodes of G, H (respectively), and W_X, W_Y are the corresponding matrix of edge weights. For convenience of notation in the statement of the problem, we will assume $|X| = |Y| = N$. The extension to the general case is quite straightforward, and will be explained in section 1.2. The goal of the Weighted Graph Matching Problem is to find a bijection $\rho : X \to Y$ that minimizes the squared difference of edge weights:

$$\text{argmin}_\rho \sum_{i=1}^{N} \sum_{j=1}^{N} \left(W_X(i, j) - W_Y(\rho(i), \rho(j))\right)^2. \tag{1}$$

It is often easier to view the map ρ as a permutation of the indices $\{1, 2, \ldots, N\}$. Let P be the corresponding $N \times N$ permutation matrix. Then we look to minimize

$$\text{argmin}_P \left\|PW_XP^T - W_Y\right\|_F^2. \tag{2}$$

Finding an exact solution to this problem is NP-Hard [1]. Instead, we look for an approximate solution via the methods presented in [2, 3]. In particular, we create a feature space in which our two graphs can be more directly compared, and perform the matching there.

1.2 Solving the relaxed WGMP via the graph Laplacian

As the WGMP is too difficult to solve exactly, we instead introduce a relaxed version of the problem which is much more reasonable. Specifically, instead of choosing P a permutation matrix as in 2, we look for an orthogonal matrix

$$Q^* = \text{argmin}_{QQ^T=I} \left\|QW_XQ^T - W_Y\right\|_F^2. \tag{3}$$

This problem was solved theoretically in [2] using eigenvectors of the graph Laplacian. Specifically, for each graph G_k ($j \in \{X, Y\}$) we define the graph Laplacian

$$L_k = D_k - W_k, \tag{4}$$

where D_k is the diagonal matrix of degrees,

$$D_k(i, i) = \sum_{j=1}^{n} W_k(i, j). \tag{5}$$

Note that L_k is symmetric, and is in fact positive semidefinite [4]. Let $L_k = U_k \Lambda_k U_k^t$, be the eigendecomposition of the Laplacian. Then the spectral graph matching theorem from [2] states that if each L_x, L_y has distinct eigenvalues, the optimal Q from 3 is given by

$$Q^* = U_Y^T S U_X, \tag{6}$$

where S is a diagonal matrix with values ± 1 to account for the sign ambiguity in eigenvectors. The calculation of S can be rather frustrating, as there are 2^N possibilities for this matrix. In [2] the authors omit S entirely by replacing U_k with $|U_k|$ in equation 6. In [3] the authors determine S by looking at the histograms of the columns of U_k (which are invariant under permutation) and adding $\pm$ according to the best fit. In the current state of our project, we are using a semi-supervised method to determine S. If we know of even one preexisting match between the nodes in X, Y, we can use this to compare eigenvectors U_X, U_Y and add the appropriate signs.

To complete our approximate solution to the WGMP, we use the matrix Q^* to find a matching

$$\rho : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, \}. \tag{7}$$

In the original formulation of the WGMP 2, an entry of 1 at position (i, j) in the permutation matrix P represents a match between node i of Y with node j of X. In the relaxed version of the problem, $Q^*(i, j)$ gives the similarity between node i of X and node j of Y rather than an exact matching. But the task of converting Q^* to a permutation ρ is easily solved using the Hungarian algorithm, which in $O(N^3)$ time finds the matching ρ that maximizes

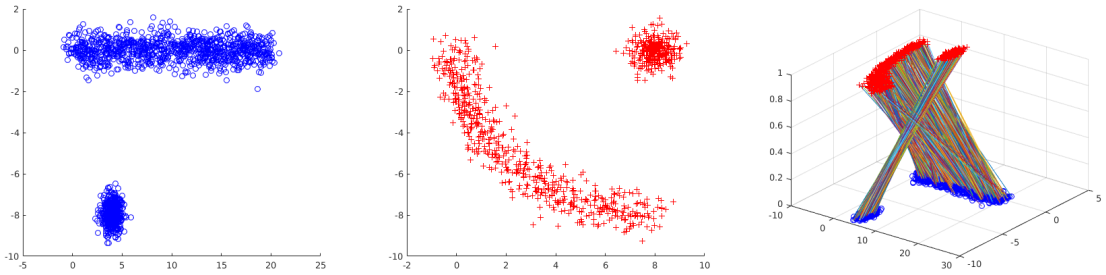
$$\sum_{i=1}^{N} Q^*(i, \rho(i)). \tag{8}$$

In practice, one often does not use every eigenvector of L_X, L_Y in the calculation of Q. Rather, we choose a number $K \ll N$, and let U_k be the $N \times K$ matrix where the columns are the eigenvectors corresponding to the K smallest eigenvalues. Heuristically, the columns of U_k represent features extracted from the original graph, and the size of the corresponding eigenvalues represents the strength of each particular feature. If we then assume that the sign ambiguity has been dealt with (so that $S = I$), then $Q^* = U_Y U_X^t$. That is, $Q^*(i, j)$ is the dot product of row i of U_Y with row j of U_X. So we are determining match strength by comparing the images of our data in this new feature space. In doing this, we can also handle the case where $|X| \neq |Y|$. If we chose $K \leq \min(|X|, |Y|)$, then formula 6 will still hold, and Q^* will be an $|Y| \times |X|$ matrix representing the node similarities.

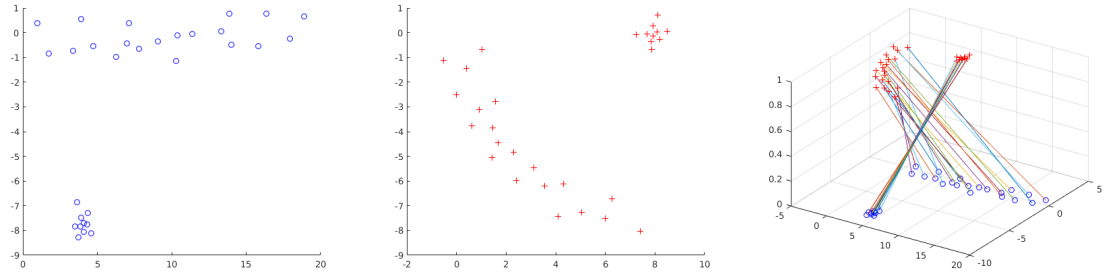
1.3 Hierarchical graph matching

The major benefit of using the Hungarian algorithm in 1.2 is that it results in the optimal one-to-one matching based on the input data, but the $O(N^3)$ runtime presents a major problem when dealing with larger datasets. To address this, we introduce a hierarchical algorithm for graph matching that solves the problem by making many smaller matches, thereby circumventing the N^3 issue. The key step in the hierarchical matching algorithm is the creation of smaller "coarsified" graphs $\tilde{G}, \tilde{H}$ of size $M \ll N$. The goal is for $\tilde{G}, \tilde{H}$ to represent the same geometrical structure as G, H, with significantly reduced size. In the current state of the project we are still investigating different methods for creating $\tilde{G}, \tilde{H}$, but the overarching idea is to choose a few nodes of high degree in the original graphs G, H to serve as representatives for larger clusters. These representatives will then form the $\tilde{G}, \tilde{H}$ that we desire.

Once we have created the coarsified graphs $\tilde{G}, \tilde{H}$, the hierarchical matching algorithm is as follows. We first run the standard graph matching algorithm on $\tilde{G}, \tilde{H}$. Then, for each match $i \to j$ in the coarse graphs, we run our graph matching algorithm between the corresponding clusters in the original graph. So in total, we create 1 match of size M, and M matches of size $\frac{N}{M}$, which gives a new runtime of $O\left(M^3 + \frac{N^3}{M^2}\right)$. This is significantly faster than the original $O(N^3)$ algorithm, and allows us to work with much larger datasets. In practice, the code takes approximately 15 seconds to run when $N = 10,000$ and $M = 50$. In theory, we could achieve an even better runtime for the matching process by performing multiple coarsification steps, and iteratively matching the graphs on increasingly fine levels. However, this would not significantly affect the runtime of the full algorithm, as finding the eigenvectors of L_X, L_Y would then become the bottleneck for the problem.



(a) Dataset X (b) Dataset Y (c) Hierarchical match result



(d) Coarsified data $\tilde{X}$ (e) Coarsified data $\tilde{Y}$ (f) Graph match on coarse data

Figure 1: Example hierarchical matching on synthetic data

2 Experiment

2.1 Graph matching

Here we show the results of our graph matching algorithm on a synthetic dataset, which is pictured in figure 1. Here the nodes of the graphs G, H are represented by 2-dimensional datasets X, Y, and the weight matrices W_X, W_Y are determined via an RBF kernel applied to the 2-norm.

$$E_X(i,j) = \|X(i) - X(j)\| \tag{9}$$

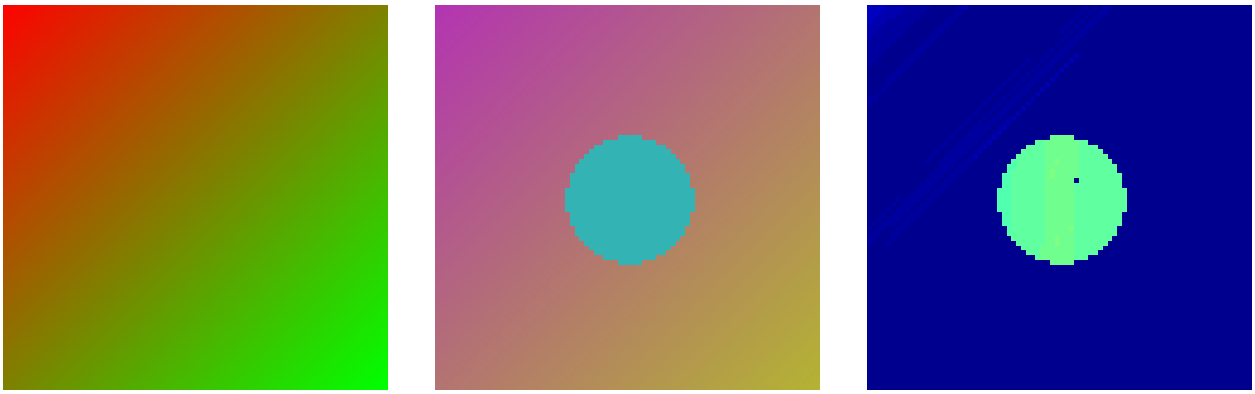
$$W_X(i,j) = -\exp\left(\frac{E_X(i,j)}{std(E_X)}\right), \tag{10}$$

and similar for W_Y. The matching is then calculated using the hierarchical algorithm in 1.3, with the match on the coarse level begin shown in figures 1d, 1e, 1f. This example has datasets of size $N = 1500$, with $M = 50$, so that the coarsified data has size $\left|\tilde{X}\right| = \left|\tilde{Y}\right| = 30$.

The purpose of this example is to show that the matching algorithm can recognize similar shapes in data that has been altered in a smooth-enough manner. Both sets X, Y contain a tight cluster of points, as well as longer line segment. In figures 1c, 1f we see the final result of the algorithm, where each match is represented by a line connecting the two points in question. As we can see in the figure, the algorithm successfully matches the objects based on their shape, as we desired.

2.2 Change detection using graph matching

One possible application of graph matching is in change detection. Suppose that the sets X, Y represent two images of the same scene, taken at different times. A direct comparison of X to Y is often not useful, as it is possible for individual pixels to change drastically while keeping the overall structure of the image the same. For example, if the images X, Y were captured in different lighting, then the set $X - Y$ would show



(a) Image X (b) Image Y (c) $\|Y(\rho^{-1}(i)) - Y(i)\|$

Figure 2: Example change detection on synthetic data

many "false" changes. Instead, we turn to graph matching as a way to compare pixels between X, Y in the context of the larger image.

We will track the changes as follows: apply the matching algorithm to X, Y, and let $\rho : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, N\}$ be the permutation of the indices corresponding to the match $X \to Y$. Then we measure the degree of change in the scene at pixel i via the quantities

$$\text{Change in } X \text{ at pixel } i = \|X(i) - X(\rho(i))\| \tag{11}$$

$$\text{Change in } Y \text{ at pixel } i = \left\|Y(\rho^{-1}(i)) - Y(i)\right\| \tag{12}$$

As a first example, we apply the algorithm to a synthetic dataset, with the results shown in figure 2. This data was constructed to mimic the above discussion, showing both a change in overall hue between images X, Y, as well as an actual structural change with the addition of a blue dot in Y. In this example, we consider the dot in Y to be the only true "change" between the images. The change in hue represents some incidental affect in the data capture process, and should be ignored by the algorithm. Here, both images are size 80×80, for a total of $N = 6400$ pixels. The algorithm runs in roughly 10 seconds, and successfully highlights the dot in Y, as desired.

At this point in the project, we began working with real-world datasets. In figure 3 we show a preliminary result of our algorithm on data from the 2010 Data Fusion Contest [5]. The dataset consists of remote sensing image of Gloucester, UK, taken both before (3a) and after (3b) a flood in the year 2000. The final result in figure 3c is difficult to fully interpret, but it is encouraging to see the algorithm highlighting noteworthy shapes from the individual images. As we continue to improve our methods, we expect to see even more obvious results.

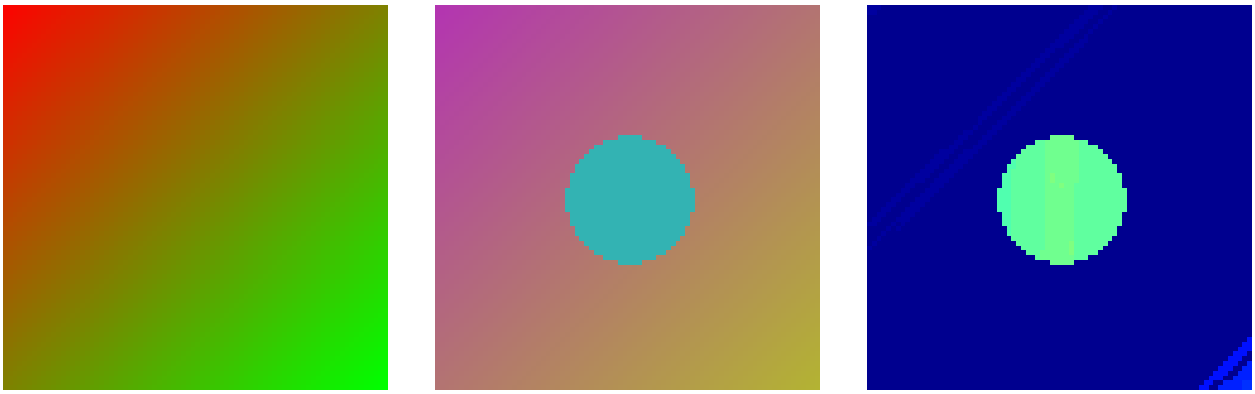
References

[1] Vikraman Arvind, Johannes Köbler, Sebastian Kuhnert, and Yadu Vasudev. *Approximate Graph Isomorphism*, pages 100–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. 1.1

[2] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, September 1988. 1.1, 1.2, 1.2, 1.2

[3] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. *Inexact Matching of Large and Sparse Graphs Using Laplacian Eigenvectors*, pages 144–153. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. 1.1, 1.2

[4] Bojan Mohar. The laplacian spectrum of graphs. *Graph Theory, Combinatorics, and Applications*, 2:871–898, 1991. 1.2



(a) Image X (before flood) (b) Image Y (after flood) (c) $\|X(i) - X(\rho(i))\|$

Figure 3: Example change detection on DFC 2010 data

[5] Nathan Longbotham, Fabio Pacifici, Taylor Glenn, Alina Zare, Michele Volpi, Devis Tuia, Emmanuel Christophe, Julien Michel, Jordi Inglada, Jocelyn Chanussot, and Qian Du. Multi-modal change detection, application to the detection of flooded areas: outcome of the 2009-2010 data fusion contest. *IEEE J. Sel. Topics Appl. Earth Observ.*, 5(1):331–342, Feb. 2012. 2.2