Unlabeled graph classification

Anonymous Author(s)
Affiliation
Address
email

Abstract

Because graphs can encode more information in their structure than vectors, they are becoming increasingly popular data structures for representing information. While the last century has witnessed the development of a plethora of statistical tools for the analysis of data, the vast majority of these tools natively operate in vector spaces, not graph spaces. Thus, algorithms for even relatively simple statistical inference tasks, such as two-class classification, are essentially absent for graph data. In this work, we propose a number of complementary algorithms to classify graphs, with special attention to the possibility of unknown vertex labels. Since exactly solving the graph-matching problem is currently computational intractable, we consider several approximate approaches. We introduce a multiple-restart Frank-Wolfe approach to solving the graph matching problem by formulating it as a quadratic assignment problem. Although this approach has superior performance than previous state-of-the-art approaches to the graph matching problem, even when it "should" do well in classification problems, it is outperformed by a graph invariant strategy. This is just the beginning.

1 Introduction

The statistical analysis of collections of graphs is becoming an increasingly popular desideratum [1]. Specifically, we consider the following idealized and simplified scenario. Let $\mathbb{G} : \Omega \mapsto \mathcal{G}$ be a graph-valued random variable taking values $G \in \mathcal{G}$. Each graph is a 4-tuple: $G = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$, where $\mathcal{V}$ is a set of $|\mathcal{V}| = n_V$ vertices, $\mathcal{E}$ is a set of $|\mathcal{E}| = E$ edges, $\mathcal{L} = \{1, \ldots, n_V\} = [n_V]$ is a set of vertex labels (one per vertex), and $\mathcal{A}$ is a set of edge attributes. Elements of $\mathcal{A}$ could be binary matrices (for unweighted graphs) or higher-order tensors (for mult-graphs). Let Y be a Bernoulli random variable, $Y : \Omega \mapsto \{0, 1\}$, such that each graph has an associated class. Given a collection of graphs and classes, we assume they were jointly sampled independently and identically from some true but unknown distribution, $\{(\mathbb{G}_i, Y_i)\}_{i\in[n]} \overset{iid}{\sim} F_{\mathbb{G},Y}(\cdot; \theta)$. Note that $F_{\mathbb{G},Y}(\cdot; \boldsymbol{\theta})$ is but one of a (possibly infinite) set of distributions, collectively comprising the model: $\mathcal{F}_{\mathbb{G},\mathcal{Y}} = \{F_{\mathbb{G},Y}(\cdot; \boldsymbol{\theta}) : \theta \in \boldsymbol{\Theta}\}$, where $\boldsymbol{\Theta}$ is the set of feasible parameters. The goal of such an analysis is to learn about the relationship between $\mathbb{G}$ and Y. Standard classification techniques fail in this domain as they typically require classifying objects that live in finite dimensional Euclidean space, whereas the object of interest here are graphs (even finite ones do not natively live in Euclidean space). In this work, therefore, we propose novel extensions of several classification algorithms appropriate for the graph domain.

2 Graph Classification

The graph classification problem may be stated thusly: given training data $\mathcal{T}_n = \{(\mathbb{G}_i, Y_i)\}_{i\in[n]}$, and a new graph, $\mathbb{G}$, estimate the new graph's corresponding class, Y, assuming each graph/class pair was sampled identically and independently from some true but unknown distribution, $\mathcal{T}_n, (\mathbb{G}, Y) \overset{iid}{\sim}$

$F_{\mathbb{G},Y}(\cdot;\theta)$. Given an appropriately defined loss-function, such as misclassification rate: $L_h = \mathbb{P}[h(\mathbb{G}) \neq Y]$, one can then search for the function $h^* \in \mathcal{H}$ that minimizes the loss function of interest:

$$h^* = \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, \mathbb{P}[h(\mathbb{G}) \neq Y]. \tag{1}$$

In general, h^* is unavailable and dependent on the model, $\mathcal{F}_{\mathbb{G},Y}$ (which includes the vertex labels). When h^* is unavailable, one can utilize training data, $\mathcal{T}_n$, to obtain $\hat{h}$, an approximation to h^*:

$$\hat{h} \approx \underset{h\in\mathcal{H}}{\operatorname{argmin}}\, \mathbb{P}[h(\mathbb{G}) \neq Y|\mathcal{T}_n], \tag{2}$$

where $\approx$ indicates that in general, we will not be able to find the actual minimum in the set $\mathcal{H}$. Regardless, any approach necessarily estimates a decision boundary in the space of graphs separating them into two classes.

In certain graph classification problems the vertex labels are unknown. In other words, instead of observing $(\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{A})$ for each graph, we only observe the triple, $(\mathcal{V}, \mathcal{E}, \mathcal{A})$, which we refer to as an *unlabeled graph*. In such scenarios, one is essentially faced with a graph isomorphism problem, which can be approached in at least two ways. First, one could try to *graph match*, that is, find a common set of labels for the vertices of the graphs, so that the graphs have the same adjacency matrix. Second, one can project the graphs into a quotient space that is invariant to the labels. This can be a negative check for isomorphisms: if a set of graphs have different representations in the quotient space, then they are not isomorphic to one another. We consider both approaches as possible subroutines as part of a classification function.

2.1 Graph Matching

A graph can be represented by its adjacency matrix, A, assuming the edge attributes are univariate. Unlabeled graphs, on the other hand, can be represented by a set of adjacency matrices, $\{QAQ^\mathsf{T} : Q \in \mathcal{Q}\}$, where Q is any permutation matrix. Given a pair of unlabeled graphs, determining whether they are isomorphic with respect to one another is equivalent to determining whether one can find an adjacency matrix of one graph that is identical to the other's. This problem can be cast as a *quadratic assignment problem* (QAP):

$$Q_{QAP} \triangleq Q_{QAP}(A, B) = \underset{Q\in\mathcal{Q}}{\operatorname{argmin}} \left\|QAQ^\mathsf{T} - B\right\|_F^2, \tag{3}$$

where A and B are adjacency matrix representations of two different graphs. A bit of linear algebra simplifies Eq. (3):

$$\underset{Q\in\mathcal{Q}}{\operatorname{argmin}} \left\|QAQ^\mathsf{T} - B\right\|_F^2 = \underset{Q\in\mathcal{Q}}{\operatorname{argmin}} -tr(B^\mathsf{T}QAQ^\mathsf{T}) - tr(QAQ^\mathsf{T}B), \tag{4}$$

which is equivalent to the standard representation of the quadratic assignment problem [2]:

$$\hat{\sigma} = \underset{\sigma}{\operatorname{argmin}}\, a_{\sigma(i),\sigma(j)}b_{ij} = \underset{q\in\mathcal{Q}}{\operatorname{argmin}}\, q_{ij}a_{ij}, q_{ji}b_{ij} \tag{5}$$

where σ is a permutation function, that is, $\sigma : [n] \mapsto [n]$. Unfortunately, Eq. (3) is an NP-complete problem [3]. The primary difficulty in solving Eq. (3) is the discrete non-convex constraint set. Thus, it is natural to consider an approximation with the constraints relaxed. Since the convex hull of permutation matrices is the set of doubly stochastic matrices, we define the approximate quadratic assignment problem:

$$Q_{AQAP} \triangleq Q_{AQAP}(A, B) = \underset{Q\in\mathcal{D}}{\operatorname{argmin}} \left\|QAQ^\mathsf{T} - B\right\|_F^2, \tag{6}$$

where $\mathcal{D}$ is the set of doubly stochastic matrices. When the permutation matrix constraint is relaxed, the equivalence relation shown in Eq. (4) no longer holds. Nonetheless, we proceed by attempting to solve:

$$\hat{Q}_{AQAP} \approx \underset{Q\in\mathcal{D}}{\operatorname{argmin}} -tr(B^\mathsf{T}QAQ^\mathsf{T}) - tr(QAQ^\mathsf{T}B), \tag{7}$$

considering it an auxiliary function for which we can compute gradients and ascend a likelihood, unlike the permutation constrained case.

The Frank-Wolfe (FW) algorithm is a successive linear programming algorithm for nonlinear programming problems; specifically, for quadratic problems with linear (equality and/or inequality) constraints. Let $f(Q) = -tr(B^\mathsf{T} Q A Q^\mathsf{T}) - tr(Q A Q^\mathsf{T} B)$. With each iteration j, the FW algorithm takes the following steps:

Step 1: Compute the gradient The gradient of f with respect to Q is given by:

$$\nabla_Q^{(j)} = \partial f / \partial Q^{(j)} = A Q^{(j)} B^\mathsf{T} + A^\mathsf{T} Q^{(j)} B. \tag{8}$$

Step 2: Find the closest doubly stochastic matrix Instead of directly descending this gradient, we search for the direction of the doubly stochastic matrix closest to this gradient. Noting that that direction may be computed by the dot-product operator, we have:

$$W^{(j)} = \underset{W^{(j)} \in \mathcal{D}}{\operatorname{argmin}} \langle \nabla_Q^{(j)}, W^{(j)} \rangle. \tag{9}$$

Eq. (9) can be solved as a Linear Assignment Problem (LAP). More specifically, a LAP can be written as:

$$Q_{\text{LAP}} \triangleq Q_{\text{LAP}}(A, B) = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| Q A - B \right\|_F^2, \tag{10}$$

which, when $B = I$, can be simplified:

$$\begin{aligned} Q_{\text{LAP}}(A, I) \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| Q A - I \right\|_F^2 &= \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} (Q A - I)^\mathsf{T} (Q A^\mathsf{T} - I) \\ &= \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} A^\mathsf{T} Q^\mathsf{T} Q A - 2 Q A - I I = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} - \langle Q, A \rangle. \end{aligned} \tag{11}$$

In other words, letting $B = I$, the projection of a matrix onto its nearest doubly stochastic matrix is a LAP problem. While Eq. (11) cannot be solved directly, as above, we can relax the permutation matrix constraint to the doubly stochastic matrix constraint:

$$Q_{\text{LAP}}(A, I) = \underset{Q \in \mathcal{D}}{\operatorname{argmin}} - \langle Q, A \rangle. \tag{12}$$

Since the permutation matrices are the vertices of the set of doubly stochastic matrices, finding the minimum of Eq. (12) is guaranteed to yield a permutation matrix (as minima are necessarily at the vertices). Thus, letting $A = \nabla_Q^{(j)}$, solving Eq. (12)—which is a quadratic problem with linear constraints—is equivalent to solving Eq. (9). Fortunately, the Hungarian algorithm solves any LAP in $\mathcal{O}(n^3)$ [4], thus this projection is relatively efficient.[1]

Step 3: Update the direction Given $W^{(j)}$, the new direction is given by:

$$d^{(j)} = W^{(j)} - Q^{(j)}. \tag{13}$$

Step 4: Line search Given this direction, one can then perform a line search to find the doubly stochastic matrix that minimizes the objective function along that direction:

$$\alpha^{(j)} = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f(Q^{(j)} + \alpha^{(j)} d^{(j)}). \tag{14}$$

This can be performed exactly, because f is a quadratic function.

Step 5: Update Q Finally, the new estimated doubly stochastic matrix is given by:

$$Q^{(j+1)} = Q^{(j)} + \alpha^{(j)} d^{(j)}. \tag{15}$$

[1] More efficient algorithms are available for certain special cases, that is, whenever the matrix-vector multiplication operation is fast (for example, when both A and B are sparse).

The grand finale Steps 1–5 are iterated until convergence, computational budget limits, or some other stopping criterion is met. These 5 steps collective comprise the FW algorithm. Note that while $Q^{(j+1)}$ will generally not be a permutation matrix, we do not project $Q^{(j+1)}$ back onto the set of permutation matrices between each iteration, as that projection requires $\mathcal{O}(n^3)$ time. After the final iteration, however, we have $\hat{Q}_{AQAP}$, which we project onto the set of permutation matrices:

$$\hat{Q}_{QAP} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \langle \hat{Q}_{AQAP}, Q \rangle, \tag{16}$$

which is a LAP, and yields approximate solution to QAP. Let `QAP` indicate this algorithm: FW appended with a projection onto the permutation matrices.

Multiple restarts Note that `QAP` will not generally achieve the global optimum even of Eq. (6), because f is not necessarily positive definite. This is clear upon computing the Hessian of f with respect to Q:

$$\nabla^2_Q = B \otimes A + B^\mathsf{T} \otimes A^\mathsf{T}, \tag{17}$$

where $\otimes$ indicates the Kronecker product. This means that the initialization, $Q^{(0)}$, will be important. While any doubly stochastic matrix would be a feasible initial point, two choices seem natural: (i) the "flat doubly stochastic matrix," $J = \mathbf{1}^\mathsf{T}\mathbf{1}/n_V$, which is the middle of the feasible region, and (ii) the identity matrix, which is a permutation matrix. Therefore, if we run the FW algorithm once, we always start with one of those two. If we use multiple restarts, each initial point is "near" the flat matrix. Specifically, we sample J', a random doubly stochastic matrix using 10 iterations of Sinkhorn balancing [5], and let $Q^{(0)} = (J + J')/2$. We refer to multiple re-starts of `QAP` with subscripts, that is, the performance of $\texttt{QAP}_n$ is the best result of n pseudo-random re-starts of `QAP`. Note that `QAP` natively operates on matrices, which could correspond to either weighted or unweighted graphs.

2.2 Graph Invariants

A graph invariant (GI) is any function that maps a graph to a scalar whose value is independent of the vertex labels, $T : (\mathcal{V}, \mathcal{E}, \mathcal{A}) \mapsto \mathbb{R}$ (note that often graph invariants are defined to operate only on unweighted graphs, that is, $T : (\mathcal{V}, \mathcal{E}) \mapsto \mathbb{R}$). By defining a set of GIs, one can embed a collection of graphs into a quotient space invariant to vertex labels. Whenever this quotient space is a subset of finite dimensional Euclidean space, all standard machine learning classifiers may be implemented to solve the classification problem.

3 Unlabeled Graph Classification Algorithms

Below we provide an example, and briefly discuss, three complementary approaches to solving unlabeled graph classification problems, that is, graph classification problems in which the vertex labels are unknown. Each strategy uses one of the two above approaches as a subroutine.

3.1 A graph dissimilarity approach: k`NN` $\circ$ `QAP`

Define a dissimilarity on graph spaces: $d : \mathcal{G} \times \mathcal{G} \mapsto \mathbb{R}_+$, in which one can compute the dissimilarity between any pair of graphs. Given an adjacency matrix representation, many such dissimilarities are possible (e.g., graph edit distance, Hamming distance, etc.).[2] Given such a dissimilarity, one can apply at least two different kinds of classifiers.

First, one could implement a nearest neighbor style classifiers, such as the k_n nearest neighbor (kNN) classifier. In addition to being universally consistent[3], kNN classifiers are relatively efficient, in that they only require $n + 1$ graph dissimilarity computations (assuming a single test graph).

[2] It is becoming increasingly popular to use a *graph kernel*, $\kappa(G, G') = \langle \phi(G), \phi(G') \rangle$, as the dissimilarity [1] ($\langle \cdot, \cdot \rangle$ indicates a dot-product, and $\phi(\cdot)$ is defined in the main text). Graph kernels have a number of desirable properties, perhaps most notably, that one can then use standard *kernel machines* to classify [6]

[3] A sequence of kNN classifiers is guaranteed to converge to the Bayes optimal classifier if as $n \to \infty$, $k \to \infty$ but $k/n \to 0$ [7].

Alternately, one could implement an interpoint-dissimilarity matrix based algorithm [8]. This strategy has the advantage of using all available information to generate a class prediction, but a disadvantage that it requires $(n+1)^2$ graph dissimilarity computations. Moreover, it may be more sensitive to outliers.

For simplicity, we only consider the kNN strategy here. Specifically, given a test adjacency matrix, A, find $\hat{Q}_i^A = \hat{Q}_{QAP}(A, B_i)$ for all n training adjacency matrices, $\{B_i\}_{i\in[n]}$. Given these solutions, let $\widetilde{A}_i = \hat{Q}_i^A A \hat{Q}_i^{A\mathsf{T}}$ for all i. Given a suitable dissimilarity $d : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_+$, one can compute $d(\widetilde{A}_i, B_i)$ for all $i \in [n]$, and sort them: $d_{(1)} \leq d_{(2)} \leq \cdots \leq d_{(n)}$. Let the k_n nearest neighbors of A be the graphs with the k_n smallest distances, $\{d_{(1)}, \ldots, d_{(j)}\}$. The estimated class of the training sample A is then the plurality class of the k_n nearest neighbors: $\hat{y} = \operatorname{argmax}_y \mathbb{I}\{\sum_{i\in[k_n]} y_{(i)} = y\}$. Call this algorithm $k\texttt{NN} \circ \texttt{QAP}$.

3.2 A graph model approach: BPI∘QAP

The joint distribution on graphs and classes may be factorized into (i) a class-conditional random graph model: $\mathcal{F}_{\mathbb{G}|Y} = \{F_{\mathbb{G}|Y}[\cdot; \boldsymbol{\theta}_Y] : \boldsymbol{\theta}_Y \in \boldsymbol{\Theta}\}$, and (ii) a class prior, $F_Y[\cdot; \pi_Y] \in \mathcal{F}_Y$. Given such a factorization, one could then, for instance, estimate $\{\boldsymbol{\theta}, \pi\}$ and then use standard model-based classifiers, such as the Bayes plugin (BPI) classifier.

To utilize a BPI classifier with QAP, one must first assume a model, $\mathcal{F}_{\mathbb{G},Y}$. Given the model, one could then use training data to estimate the model parameters. However, if all the training graphs are unlabeled, unless the class-conditional signal is independent of the vertex labels, one must somehow deal with the absent labels. Sometimes a "canonical" labeling is somehow natural, and can be searched for. Otherwise, could QAP all the training graphs to the test graph. This approach has the unfortunate consequence of making all the training graphs more similar to one another, disregarding the class labels. Another option would be to select a "prototype" from each class. Although several prototype selection strategies have been studied, it is still problematic [1]. Finally, one could generate prototype for each class, using all available information. This approach is intriguing but complicated. Each of these alternatives essentially defines a canonical labeling per class.

Regardless of how a canonical labeling is chosen per class, given this canonical labeling, one then aligns each graph in each class appropriately using $\texttt{QAP}_1$. Once aligned, the likelihood parameters, $\{p_{uv|y}\}$, may be estimated, yielding a likelihood matrix for each class, P_y. Because QAP natively handles weighted graphs, one can then simply $\texttt{QAP}_1$ the test graph to each P_y. Then, compute the likelihood of A coming from each class, using the estimated parameters. Because class-prior probabilities are $1/2$, the likelihood is equal to the posterior, so $\hat{y} = \operatorname{argmax}_y \mathbb{P}[Y = y|\widetilde{A}; P_y]$. Call this algorithm $\texttt{BPI} \circ \texttt{QAP}$.

3.3 A graph invariant approach: CW∘GI

Define an embedding of graphs into finite dimensional Euclidean space: $T : (\mathcal{V}, \mathcal{E}, \mathcal{A}) \mapsto \mathbb{E}^d$. Unfortunately, there is no known set of graph invariants that collectively solve the graph isomorphism problem [2]. Fortunately, some recent theoretical work shows that certain graph invariants have greater discriminability with regard to certain graph inference tasks [9]. With that in mind, in this work, we use a standard set of unweighted graph invariant measures, described fully in [**?**] and [] [Borges11]. Importantly, this means that we discard all weighting information and calculate these invariants on the simple, unweighted graph with no self loops. For brevity, we only present the intuition behind each invariant, leaving the specific formulas for each to their respective papers.

- T_{size} is the number of edges in the graph.
- $T_{maxdegree}$ is the $\max_v d(v)$ for all $v \in \mathcal{V}$, where $d(v)$ is the degree of vertex v.
- T_{MAD_g} is a greedy approximation of the maximum average degree.
- T_{MAD_e} is an eigenvalue approximation of the same.
- T_{S_1} is the maximum locality statistic, as described in [**?**].
- $T_{triangles}$ is the number of triangles (paths of length 3) in the graph.
- T_{cc} is the average clustering coefficient.

- $T_{averagepathlength}$, $T_{closeness}$, and $T_{betweenness}$ are all measures of the path lengths required to traverse between arbitrary vertices or edges.

Let T_{ij} indicate the i^{th} graph invariant of the j^{th} graph. We normalize each value T_{ij} according to $T_{ij} \leftarrow \frac{T_{ij} - \min(T_i)}{\max(T_i) - \min(T_i)}$, yielding a value between 0 and 1.

For each graph G_i in the training set, we compute a graph invariant vector: $\boldsymbol{T}_i : \mathcal{G} \mapsto \mathbb{R}^d$. We stack these n d-dimensional vectors to form a matrix $\boldsymbol{T} \in \mathbb{R}^{n \times d}$. We then whiten this matrix to control for the divergence means and scales of the various graph invariants, $\boldsymbol{T} \rightarrow \boldsymbol{E}\boldsymbol{\Lambda}^{-1/2}\boldsymbol{E}^\mathsf{T}\boldsymbol{T}$, where $\boldsymbol{E}\boldsymbol{\Lambda}\boldsymbol{E}^\mathsf{T}$ is the eigenvalue decomposition of the covariance matrix,$\mathbb{E}[\boldsymbol{T}\boldsymbol{T}^\mathsf{T}]$ [**?**]. Now, to estimate the class of a test graph, we first compute its invariant vector, $\boldsymbol{t}$, and normalize it appropriately. We then applied a variety of machine learning algorithms, including kNN, linear classifiers, and SVMs. For the below connectome data, the best performing algorithm is the confidence weighted classifier. Call this algorithm `CWoGI`.

4 Results

4.1 QAP benchmarks vs. PATH algorithm

We first compare the performance of $\texttt{QAP}_n$ with recent state-of-the-art approaches on the QAP benchmark library [10]. Specifically, [11] reported improved performance in all but two cases, in which the QPB method of Cremers et al. [12] achieved a lower minimum. We compare $\texttt{QAP}_n$ with the previous best performing algorithm. In *all* cases, $\texttt{QAP}_3$ outperforms the previous best result, often by orders of magnitude in terms of relative error. In three cases, $\texttt{QAP}_{100}$ achieves the minimum. In 12 out of 16 cases 75%, the simple $\texttt{QAP}_1$ algorithm outperforms the others (starting with the flat doubly stochastic matrix). See Figure 3 for quantitative comparisons.

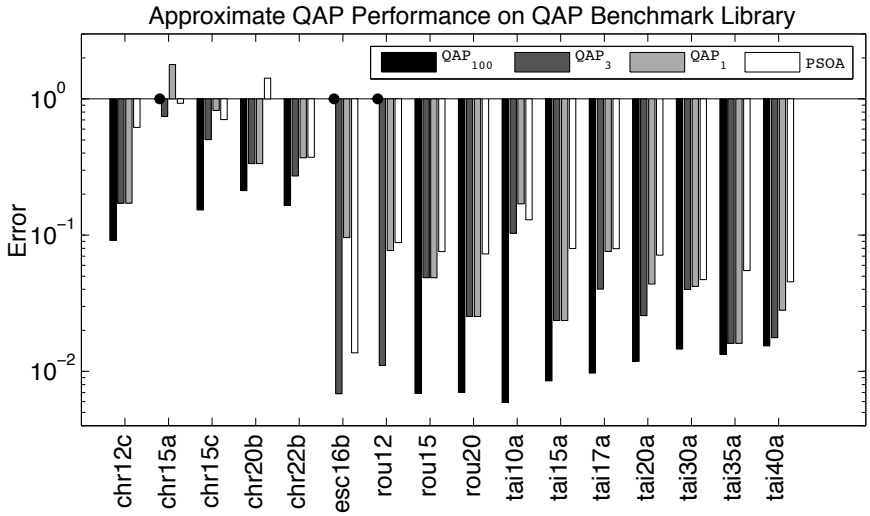

Figure 1: $\texttt{QAP}_3$ outperforms PSOA on all 16 benchmark graph matching problems. Moreoever, $\texttt{QAP}_1$ outperforms PSOA on 12 of 16 tests. For 3 of 16 tests, $\texttt{QAP}_{100}$ achieves the minimum (none of the other algorithms ever find the absolute minimum), as indicated by a black dot. Let f_* be the minimum and $\hat{f}_x$ be the minimum achieved by algorithm x. Error is $f_*/\hat{f}_x - 1$.

4.2 Simulations

$\texttt{QAP}_n$'s near perfect performance gave us hope for using `QAP` as part of an unlabeled graph classifier. To investigate further, we generated some simulations using the following assumptions. First, assume an independent edge random graph model for each class: $F_y = \prod_{(u,v)\in\mathcal{E}} p_{uv|y}^{a_{uv}}(1 - p_{uv|y})^{(1-a_{uv|y})}$, where $\{p_{uv|y}\}$ are the likelihood parameters. Then, assume class prior probabilities

are equal, $\mathbb{P}[Y = 1] = \mathbb{P}[Y = 0] = 1/2$. For simplicity, we sample one graph from each class, meaning $n = 2$, and a single training graph sampled according the class priors. Thus, each simulation is defined by $\mathcal{M} = (P_0, P_1, n_V)$, where P_0 and P_1 are the class-conditional likelihoods, and n_V is the number of vertices per graph. Given a model, $\mathcal{M}$, we generate n_{MC} Monte Carlo trials. For each model, $L_{chance} = 0.5$. We estimate Bayes error, L_*, by using the true parameters and a Bayes plugin classifier. We then implement $\texttt{1NN} \circ \texttt{QAP}_1$, and plot both the misclassification rate and objective function $f(Q^{(j)})$ as a function of of iteration number (not number of restarts, which we hold fixed at one). Figure **??** shows the result of one such simulation.

While Fig. **??** shows the QAP objective function decreasing with each iteration, Fig. **??** shows that classification performance does not decrease. We found similar numerical results in two additional simulations: a heterogeneous-kidney-egg model (Figure **??**) and a fully heterogeneous model (Figure **??**). Note that in all simulations $L_{chance} \geq \hat{L}_{QAP} \geq L_*$, as it must be. Multiple iterations not improving classification performance led us to investigate the relationship between QAP and the Linear Assignment Problem (LAP).

4.3 LAP vs. QAP

The LAP approximation for this class problem is given by the following equation:

$$\hat{Q}_{\text{LAP}} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| QA^\mathsf{T} - B \right\|_F^2, \tag{18}$$

which is quite similar to QAP, except A is only post-multiplied by Q, instead of also being pre-multiplied by Q. LAP can be solved exactly as a quadratic problem with linear constraints (as above, one can relax the constraint to the set of doubly stochastic matrices, and still be guaranteed to obtain a permuation matrix). Thus, one can use gradient ascent to to solve a LAP. The gradient of $f'(Q) = \left\| AQ^\mathsf{T} - B \right\|_F^2$ is $\partial f'/\partial Q = 2AB^\mathsf{T}$. Comparing this gradient to that of QAP (Eq. (8)), one can see that when $Q^{(j)}$ is the identity matrix, the two gradients are identical. Thus, if QAP is initialized at the identity matrix, the first permutation matrix (output of Eq. (9)) is be identical to $\hat{Q}_{LAP}$; although the line search will make $Q^{(1)} \neq \hat{Q}_{LAP}$, in general. Moreoever, projecting a matrix onto the closest permutation matrix can be written as a LAP because of the following relationship:

$$\begin{aligned} \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| QA^\mathsf{T} - I \right\|_F^2 &= \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} (QA^\mathsf{T} - I)^\mathsf{T} (QA^\mathsf{T} - I) \\ &= \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} AQ^\mathsf{T} QA^\mathsf{T} - 2QA^\mathsf{T} - II = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} -\langle Q, A^\mathsf{T} \rangle \end{aligned} \tag{19}$$

This suggests that for certain problems, LAP is both an efficient and useful approximation to graph matching. We confirm this intuition by substituting QAP with LAP in the above simulations (black line). As depicted in the above figures, this intuition is consistent with the numerical results. In other words, while naively one might implement an algorithm with exponential time complexity, LAP, which is only quadratic time complexity, will often suffice.

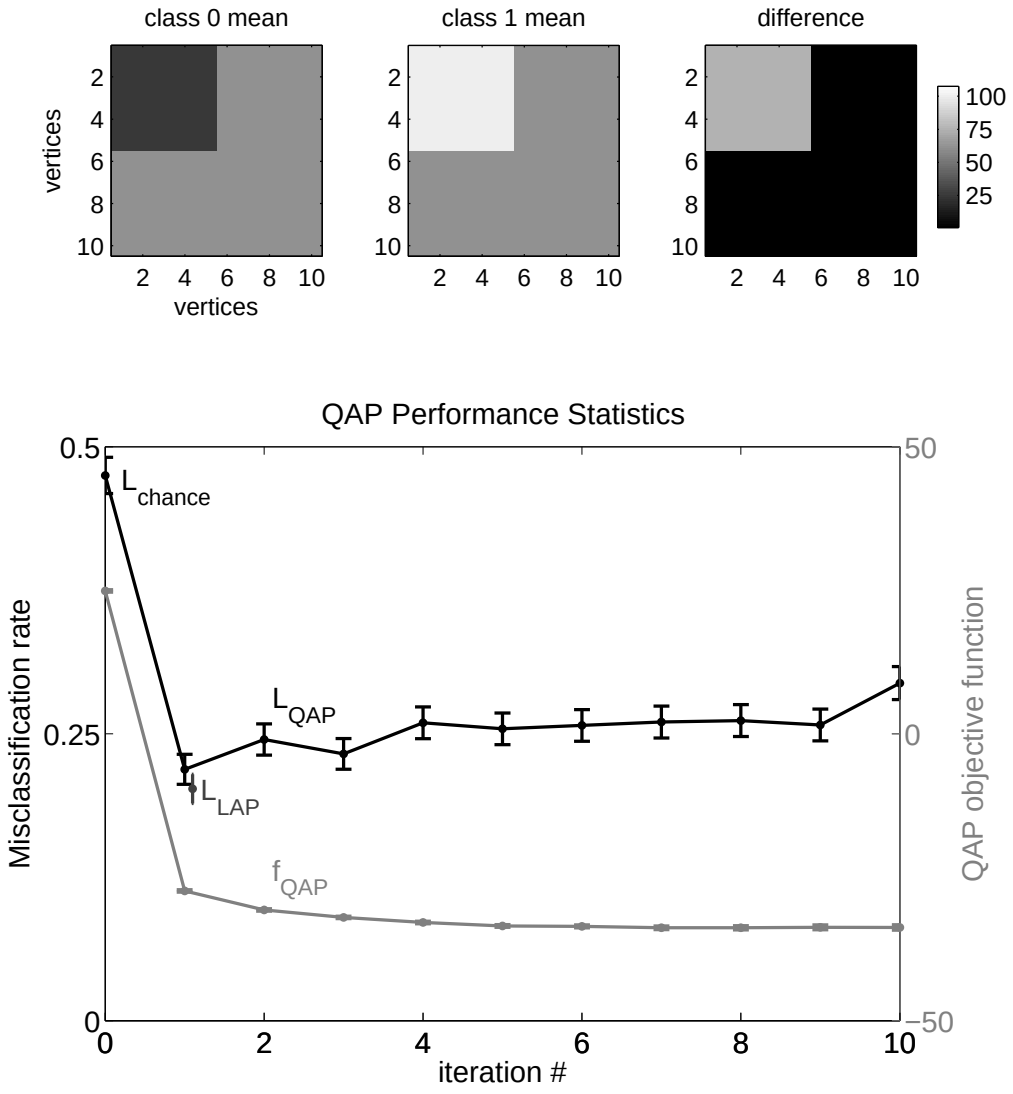

Figure 2: Homogeneous-kidney-egg model simulation results. The top panel shows the model: each edge in the "kidney" in both classes has probability 0.5; in the egg, class 0 edges are sampled with probability 0.25, and class 1 edges are sampled with probability 0.75. The bottom panel shows the QAP objective function (gray) and misclassification rate (black) as a function of iteration number.

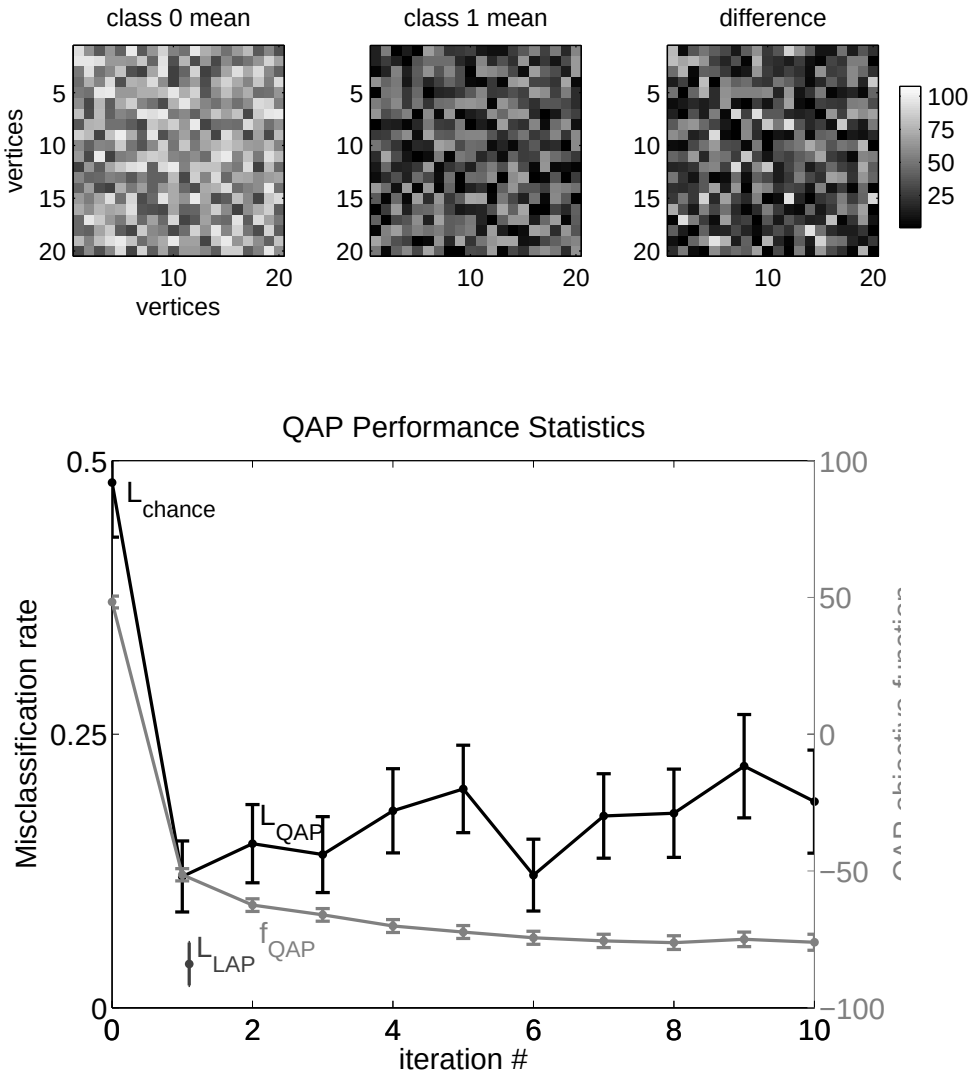

Figure 3: Heterogeneous model simulation results. The top panel shows the model: each edge in the "kidney" in both classes has probability 0.5; in the egg, class 0 edges are sampled with probability 0.25, and class 1 edges are sampled with probability 0.75. The bottom panel shows the QAP objective function (gray) and misclassification rate (black) as a function of iteration number.

4.4 Connectome Classification

A "connectome" is a graph in which vertices correspond to biological neural units, and edges correspond to connections between the units. Diffusion Magnetic Resonance (MR) Imaging and related technologies are making the acquisition of MR connectomes routine [13]. We use 49 subjects from the Baltimore Longitudinal Study on Aging, with acquisition and connectome inference details as reported in [14]. For each connectome, we obtain a 70×70 element adjacency matrix, where each element of the matrix encodes the number of streamlines between a pair of regions, ranging between 0 and about 65,000. Associated with each graph is class label based on the gender of the individual (24 males, 25 females). Because the vertices are labeled, we can compare the results of having the labels and not having the labels. As such, we implement the following classification strategies. In each case, we use a leave-one-out strategy to evaluate performance:

N/A-QAP Using the vertex labels, implement a standard 1NN classifier, where distance is the norm of the difference between any pair of adjacency matrices.

1-QAP Permute only the vertex labels of the test graph, and then implement $\texttt{1NN}\circ\texttt{QAP}_1$.

48-QAP Permuting the vertex labels, then implement $\texttt{1NN}\circ\texttt{QAP}_1$.

AVG-QAP Permuting the vertex labels, $\texttt{QAP}_1$ each of the 48 training graphs to the test graph. Then, given those permuted adjacency matrices, compute the average, and then implement a standard 1NN classifier.

1NN-GI Use the graph invariant approach as described above. We provide the normalized graph invariants as inputs into a number of standard classifiers, including kNN, linear classifiers, support vector machines, random forests, and CW. On this data, the CW classifier performed best; we therefore only report its results.

Table 1 shows leave-one-out misclassification rates for the various strategies.

Table 1: MR Connectome Leave-One-Out Misclassification Rates

N/A-QAP	1-QAP	48-QAP	AVG-QAP	1NN-GI
20%	31%	45%	??	25%

5 Discussion

In this work, we have presented a number of approaches one could take to classifier graphs. Importantly, when the vertex labeling function is unavailable, one must deal with this uncertainty somehow. We compare a number of approaches on both simulated and connectome data. A multiple-restart Frank-Wolfe approach to approximating QAP outperforms previous state-of-the-art approaches in terms of approximating the graph matching problem. Simulations demonstrate that only the first iteration of such an iterative algorithm, starting from the identity matrix, yields classification performance better than chance. Moreover, the first iteration is identical to LAP, which is a quadratic problem with linear constraints, and therefore can be solved quite easily.

On a connectome dataset, we compare the performance of various QAP classification algorithms with several graph invariant (GI) based strategies. Of the algorithms that we tried, a graph invariant approach was most effective, even though, in theory, a QAP based approach could have done better (compare the first and last columns of Table 1).

These analyses leave many open questions. Perhaps most interestingly, when might one expect a QAP-based approach to outperform a GI-based approach? Resorting to a generative model, it should be clear that if the class conditional difference is independent of the vertex labels, then there is no reason to even try to implement graph matching. However, if one believes that the labeling function might convey some class-conditional signal (as in the connectome data), then QAP-based approaches could outperform any approach that ignores the labeling function. Which QAP-based approach to use in such a scenario, however, will depend on many factors, including the assumed model and computational resources.

References

[1] Horst Bunke and Kaspar Riesen. Towards the Unification of Structural and Statistical Pattern Recognition. *Pattern Recognition Letters*, in press, 2011.

[2] D Conte, P Foggia, C Sansone, and M Vento. THIRTY YEARS OF GRAPH MATCHING. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.

[3] M R Garey and D S Johnson. *Computer and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[4] Rainer E Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment Problems*. SIAM, 2009.

[5] R Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

[6] V N Vapnik. *Statistical Learning Theory*, volume 2 of *Wiley Series on Adaptive and Learning Systems for Signal Processing, Communications and Control*. Wiley, 1998.

[7] Luc Devroye, Laszlo Györfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition (Stochastic Modelling and Applied Probability)*. Springer, 1997.

[8] Robert P.W. Duin and Elbieta Pkalskab. The dissimilarity space: bridging structural and statistical pattern recognition. *Pattern Recognition Letters*, in press, 2011.

[9] Andrey Rukhin and Carey E Priebe. A Comparative Power Analysis of the Maximum Degree and Size Invariants for Random Graph Inference. *Journal of Statistical Planning and Inference*, 141(2):1041–1046, 2011.

[10] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. QAPLIB A Quadratic Assignment Problem Library. *Journal of Global Optimization*, 10(4):391–403, 1997.

[11] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, 2009.

[12] Christian Schellewald, Stefan Roth, and Christoph Schnörr. Evaluation of Convex Optimization Techniques for the Weighted Graph-Matching Problem in Computer Vision. In *Proceedings of the 23rd DAGMSymposium on Pattern Recognition*, pages 361–368. Springer-Verlag, 2001.

[13] P Hagmann, L Cammoun, X Gigandet, S Gerhard, P Ellen Grant, V Wedeen, R Meuli, J P Thiran, C J Honey, and O Sporns. MR connectomics: Principles and challenges. *J Neurosci Methods*, 194(1):34–45, 2010.

[14] Joshua T Vogelstein, William R Gray, Jerry L Prince, Luigi Ferrucci, Susan M Resnick, Carey E Priebe, and R Jacob Vogelstein. Graph-Theoretical Methods for Statistical Inference on MR Connectome Data. *Organization Human Brain Mapping*, 2010.