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, even relatively simple statistical inference tasks, such as two-way 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 an unknown vertex labeling function. 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

$\texttt{QAP}_1$ and $\texttt{QAP}_n$ and $\texttt{QAP}_{100}$. The statistical analysis of collections of graphs is becoming an increasingly popular desideratum [cite]. 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: $G = (\mathcal{V}, \mathcal{E}, \alpha_V, \alpha_E)$, where $\mathcal{V}$ is a set of $|\mathcal{V}| = V$ vertices, $\mathcal{E}$ is a set of $|\mathcal{E}| = E$ edges, $\alpha_V : \mathcal{V} \mapsto \mathcal{A}_V$ is a vertex labeling function, and $\alpha_E : \mathcal{E} \mapsto \mathcal{A}_E$ is an edge attributing function (for example, edge weights). Given a graph, if $\mathcal{A}_E$ is univariate, one can construct an adjacency matrix representation, $A \in \mathcal{A}_E^{V \times V}$ (generalizations to adjacency tensors are straightforward). Let Y be a Bernoulli covariate: $Y : \Omega \mapsto \{0, 1\}$, yielding graph two-way *classification* problem. Given a collection of graphs and associated covariates, we assume they were jointly sampled independently and identically from some true but unknown distribution, $\{(\mathbb{G}_i, Y_i)\}_{i \in [n]} \sim F_{\mathbb{G},Y}(\cdot; \theta)$. 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 finite dimensional Euclidean objects ($G \in \mathbb{R}^d$), whereas the object of interest here are graphs ($G \in \mathcal{G} \nsubseteq \mathbb{R}^d$). In this work, therefore, we propose novel extension of classification algorithms appropriate for the graph domain.

2 Graph Classification

The graph classification problem may be stated thusly: given $\mathcal{T}_n = \{(\mathbb{G}_i, Y_i)\}_{i \in [n]} \sim F_{\mathbb{G},Y}(\cdot; \theta)$ and a new graph, $\mathbb{G}$, estimate the new graph's corresponding class, Y. 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 algorithm $h^* \in \mathcal{H}$ that minimizes the loss function of interest:

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

In general, h^* is unavailable and dependent on the model, $\mathcal{F}_{\mathbb{G},Y}$. Instead, one can therefore utilize training data, $\mathcal{T}_n$, to obtain $\widetilde{h}$, an approximation to h^*:

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

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. We consider a few distinct such approaches to constructing such a decision boundary:

Graph dissimilarity approach 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 [cite]. Given an adjacency matrix representation, many such dissimilarities are possible (e.g., graph edit distance, Hamming distance, etc.). It is becoming increasingly popular to use a *graph kernel*, $\kappa(G, G') = \langle \phi(G), \phi(G') \rangle$, as the dissimilarity [cite] ($\langle \cdot, \cdot \rangle$ indicates a dot-product). Graph kernels have a number of desirable properties, perhaps most notably, that one can then use standard *kernel machines* to classify. Whether or not one uses a graph kernel, given such a dissimilarity, standard classification algorithms, including k_n nearest neighbor (kNN) algorithms [cite] and interpoint-dissimilarity matrix based algorithms [cite], can be straightforwardly applied.

Graph model approach Factorize the joint distribution on graphs and classes into a (i) 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 (for example, the Bayes plugin classifier) [VogelsteinPriebe11].

Graph embedding approach Define an embedding of graphs into finite dimensional Euclidean space: $\phi : \mathcal{G} \mapsto \mathbb{R}^d$. Once in $\mathbb{R}^d$, one can apply one of many possible standard machine learning or other such approaches [Bunke].

3 Unlabeled sticky wicket

In certain graph classification problems the vertex labeling function, α_V, is unknown. In such scenarios, one must (either implicitly or explicitly) deal with the *graph matching* (GM) problem. In words, graph matching is the operation of finding a set of labels for a collection of vertices on multiple graphs. For the graph classification setting, we require an *approximate* GM approach [Conte04], as we do not expect, in general, for graphs to be isomorphic to one another; rather, we expect graphs within the same class to be "similar" to one another. We consider two complimentary approaches to approximate graph matching.

Adjacency matrix space approach In this representation, GM can be considered a special case of a quadratic assignment problem (QAP) [cite]. Unfortunately, no polynomial time algorithm is known to solve QAP, although much work has been devoted to this problem [cite]. Fortunately, efficient and approximate QAP solvers, such as the Frank-Wolfe (FW) algorithm [cite], are readily available. Given an approximate assignment of each graph, one can use any of the above classification approaches.

Graph invariant (GI) approach A graph invariant is any function that maps a graph to a scalar whose value is independent of the vertex labeling, $T : (\mathcal{V}, \mathcal{E}) \mapsto \mathbb{R}$. By defining a set of GIs, one can embed a collection of graphs into an invariant space. This could be considered a special case of the "Graph embedding approach" to classification described above.

4 Methods

4.1 QAP Approach

In the adjacency matrix approach, we first use an approximate QAP approach to approximate the GM problem. Then, given the estimated labels, we can implement a kNN classifier.

The QAP is defined by the following, where we seek to find a permutation matrix, $\hat{Q}$, such that

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

where A and B are adjacency matrix representations of two different graphs. A bit of linear algebra [HornJohnson] shows that Eq (3) can be simplified:

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

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

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

where σ is a permutation, that is, $\sigma : [n] \mapsto [n]$. As hinted at above, solving Eq. (4) is NP-Incomplete (not known to belong either to P or NP). Because the primary difficulty is the discrete, non-convex constraint set, it is natural to consider an approximate solution with the constraints relaxed. And because the set of permutation matrices is a subset of the doubly stochastic matrices, we define the approximate quadratic assignment problem:

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

where $\mathcal{D}$ is the set of doubly stochastic matrices. Note that when the permutation matrix constraint it relaxed, the equivalence relation shown in Eq. (4) no longer holds, that is:

$$\underset{Q \in \mathcal{D}}{\operatorname{argmin}} \left\| QAQ^{\mathsf{T}} - B \right\|_F^2 \neq \underset{Q \in \mathcal{D}}{\operatorname{argmin}} -tr(B^{\mathsf{T}} QAQ^{\mathsf{T}}) - tr(QAQ^{\mathsf{T}} B). \tag{7}$$

Nonetheless, we proceed by trying to solve Eq. (6), 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 (SLP) [] algorithm for non-linear programming problems, specifically, for quadratic problems with linear (equality and/or inequality) constraints. Let $f(Q) = \left\| QAQ^{\mathsf{T}} - B \right\|_F^2$. With each step k, the gradient of f with respect to Q is given by:

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

see [**?**] pg. 168 for details. Instead of directly ascending this gradient, we traverse 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:

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

Although $W^{(k)}$ is constrained only to be a doubly stochastic matrix, it is guaranteed to be a permutation, because the permutation matrices are the vertices of the set of doubly stochastic matrices (and minima are necessarily at the vertices). Note that Eq. (14) is a linear assignment problem (LAP) []. The Hungarian algorithm is an efficient algorithm for finding the global optimum of any LAP in $\mathcal{O}(V^3)$ [].[1] Given this direction, one can then perform a line search to find the doubly stochastic matrix that minimizes the objective function along that direction:

$$\alpha^{(k)} = \underset{\alpha \in [0,1]}{\operatorname{argmin}} f(Q^{(k)} + \alpha^{(k)} W^{(k)}) \tag{10}$$

[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).

This can be performed exactly, because f is a quadratic function. Finally, the new estimated doubly stochastic matrix is given by:

$$Q^{(k+1)} = Q^{(k)} + \alpha^{(k)} W^{(k)} \tag{11}$$

Eqs. (8)–(11) are iterated until convergence, computational budget limits, or some other stopping criterion is met. Note that while $Q^{(k+1)}$ is not a permutation matrix, we do not project $Q^{(k+1)}$ back onto the set of permutation matrices between each iteration, as that projection is a LAP, and requires $\mathcal{O}(n^3)$ time. At convergence, however, we have $\hat{Q}_{AQAP}$, which we project onto the set of permutation matrices:

$$\hat{Q}_{QAP} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \langle \hat{Q}_{AQAP}, Q \rangle, \tag{12}$$

which is our approximate solution to QAP. Note that FW will not generally achieve the global optimal even of Eq. (6), because f is not necessarily positive definite. This is clear upon computing the Hessian of f with respect to Q:

$$\nabla^2_Q = B \otimes A + B^\mathsf{T} \otimes A^\mathsf{T}, \tag{13}$$

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," $J = \mathbf{1}^\mathsf{T}\mathbf{1}/V$, which is the middle of the feasible region, and (ii) the identity matrix, which is actually 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 rebalancing, 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 QAP of n pseudo-random re-starts.

4.1.1 kNN ∘ QAP

We use the following algorithm to utilize the above approach within a kNN classification framework. Given a test adjacency matrix, A, find $\hat{Q}^A_i = \hat{Q}_{QAP}(A, B_i)$ for all n training adjacency matrices, $\{B_i\}_{i \in [n]}$. Given these solutions, let $\widetilde{A}_i = \hat{Q}^A_i A \hat{Q}^{A\mathsf{T}}_i$ for all i. Given a suitable dissimilarity $d : \mathcal{A}_E^{V \times V} \times \mathcal{A}_E^{V \times V} \mapsto \mathbb{R}_+$, one can compute $d(\widetilde{A}_i, B_i)$ for all $i \in [n]$, and sort them: $d_{(1)} \leq d_{(2)} \leq \cdot \leq d_{(n)}$. Let the k_n nearest neighbors of A be the graphs with the k_n smallest distances, $\{d_{(1)}, \ldots, d_{(k)}\}$.[2] The estimated class of the training sample A is then the plurality class of the k_n nearest neighbors: $\hat{y} = \operatorname{argmax}_y \mathbb{I}\{\sum_{i \in [k_n]} y_{(i)} = y\}$.

4.1.2 BPI∘QAP

To utilize a Bayes plugin (BPI) classifier with `QAP` , we take the following strategy. First, we 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. For simplicity, we assume class prior probabilities are equal, $\mathbb{P}[Y = 1] = \mathbb{P}[Y = 0] = 1/2$. Given a test adjacency matrix, A, do QAP_1 with respect to a single training graph from each class: $\hat{Q}^A_i = \hat{Q}_{QAP}(A, B_i)$ for $i = 0, 1$. Then, compute the likelihood of A coming from each class, using the true parameters. Because class-prior probabilities are $1/2$, the likelihood is equal to the posterior, so $\hat{y} = \operatorname{argmax}_y \mathbb{P}[Y = y | A; \{p_{uv|y}\}]$. This procedure fundamentally only uses *two* training samples, one from each class.

4.2 Graph invariant approach

In the graph invariant approach, we first define a set of d graph invariants: (i) , (ii) , (iii).... 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, $\widetilde{\boldsymbol{T}} = (\boldsymbol{T} - \boldsymbol{\mu})\boldsymbol{\Sigma}^{-1}$, where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and variance of $\boldsymbol{T}$, respectively. Now, to estimate the class of a test graph, we first compute its invariant vector, $\boldsymbol{t}$, and normalize it appropriately. Then, we can apply a standard kNN algorithm, given a suitably defined dissimilarity (and rule for k_n).

[2] Note that k_n is a function of n, typically chosen so that as $n \to \infty$, $k \to \infty$ but $k/n \to 0$, to satisfy the universal consistency proofs [cite].

5 Results

5.1 QAP benchmarks vs. PATH algorithm

Before comparing the QAP and GI approach, we first compare the performance of $\texttt{QAP}_n$ with recent state-of-the-art approaches on the QAP benchmark library [Cela07 # 38 in PATH paper]. Specifically, [PATH] reported improved performance in all but two cases, in which the QPB method of Cremers et al. [18 in PATH] achieved a lower minimum. We compare $\texttt{QAP}_n$ with the previous best performing algorithm. In *all* cases, $\texttt{QAP}_{100}$ outperforms the previous best result. In 12 out of 16 cases 75%, the simple $\texttt{QAP}_1$ algorithm outperforms the others (starting with the flat doubly stochastic matrix). See Table 1 and Figure 1 for quantitative results.

Table 1: Comparison of Frank-Wolfe with Minimum Solution and Previous State-of-the-Art (PSOA)

#	Problem	Min	$\texttt{QAP}_{100}$	$\texttt{QAP}_3$	$\texttt{QAP}_2$	$\texttt{QAP}_1$	PSOA
1	chr12c	11156	12176	13072	13072	13072	18048
2	chr15a	9896	9896	17272	17272	27584	19086
3	chr15c	9504	10960	14274	14274	17324	16206
4	chr20b	2298	2786	3068	3068	3068	5560
5	chr22b	6194	7218	7876	7876	8482	8500
6	esc16b	292	292	294	294	320	296
7	rou12	235528	235528	238134	253684	253684	256320
8	rou15	354210	356654	371458	371458	371458	381016
9	rou20	725522	730614	743884	743884	743884	778284
10	tai10a	135028	135828	148970	157954	157954	152534
11	tai15a	388214	391522	397376	397376	397376	419224
12	tai17a	491812	496598	511574	511574	529134	530978
13	tai20a	703482	711840	721540	721540	734276	753712
14	tai30a	1818146	1844636	1890738	1894640	1894640	1903872
15	tai35a	2422002	2454292	2460940	2460940	2460940	2555110
16	tai40a	3139370	3187738	3194826	3194826	3227612	3281830

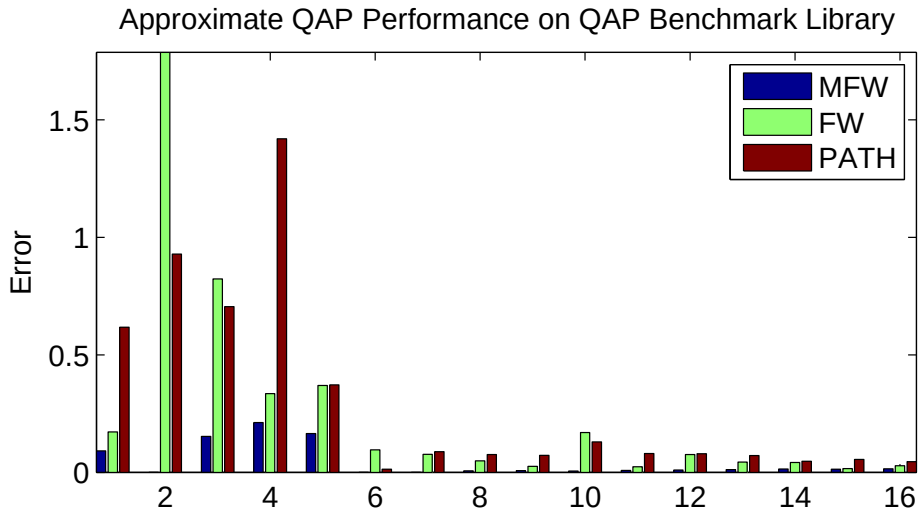
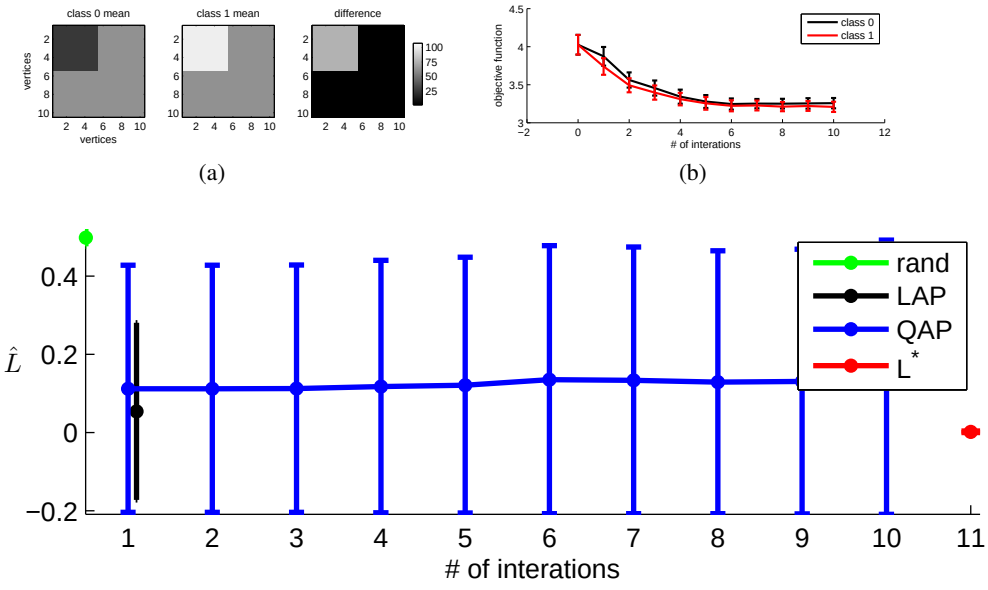


Figure 1: $\texttt{QAP}_{100}$ outperforms PSOA on all 16 benchmark graph matching problems. Moreoever, $\texttt{QAP}_1$ outperforms PSOA on 12-of-16 tests. Note that for many tests, $\texttt{QAP}_1$ has a negligible error. Error is the raito of the true minimum is the lowest achieved minimum of each algorithm .

5.2 Simulations

That $\texttt{QAP}_n$ performs so well, nearly perfectly, on all the benchmarks, inspired us investigate classifying graphs after implementing $\texttt{QAP}_1$. In all simulations we generate data according to the independent edge model described above with $V = 10$ vertices and $n = 1000$ total samples, 500 from each class. First, we generated a simulated homogeneous-kidney-egg problem, as depicted in Figure 2(a). Then, we run the Bayes Plugin $\texttt{QAP}_1$ algorithm described above. Note that $\texttt{QAP}_1$ is an *iterative* algorithm, so we can evaluate the performance of each iteration (not restart). While $\texttt{QAP}_1$ shows the QAP objective function decreasing with each iteration (Figure 2(b)), the classification performance does not decrease (Figure 2(c)). We found similar numerical results in two additional simulations: a heterogeneous-kidney-egg model (Figure 5.2) and a fully heterogeneous model (Figure 5.2). 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).



(a) (b)

(c) Lhat +/- Lstd

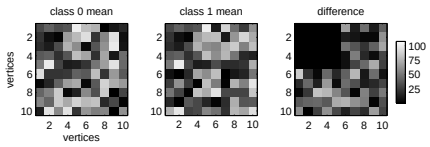
Figure 2: Homogeneous-kidney-egg model simulation results. (a) Model: Each edge in the "kidney" in both classes has probability 0.5; in the edge, class 0 edges are sampled with probability 0.25, and class 1 edges are sampled with probability 0.75. (b) QAP objective function improves with each iteration of $\texttt{QAP}_1$. (c) But classification performance does not improve.

5.3 LAP vs. QAP

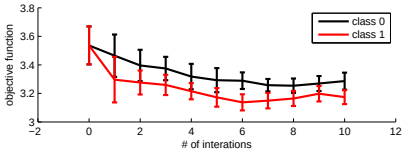
The LAP approximation for this class problem is given by the following equation:

$$\hat{Q}_{\text{LAP}} = \underset{Q \in \mathcal{Q}}{\operatorname{argmin}} \left\| AQ^{\mathsf{T}} - B \right\|_F^2, \tag{14}$$

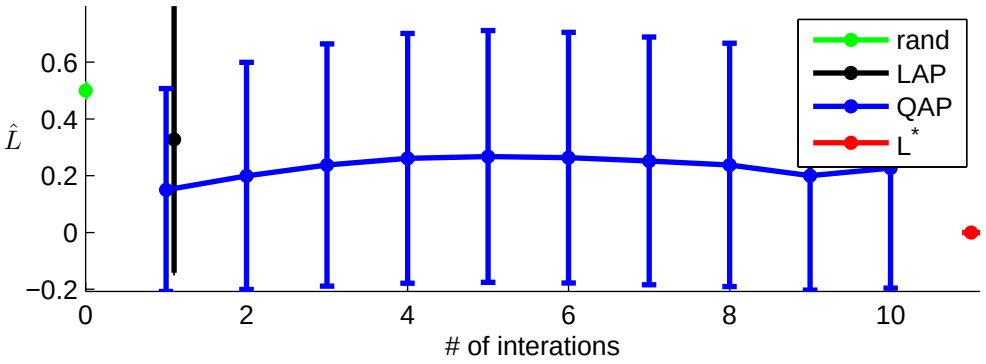
which is quite similar to QAP, except A is only post-multiplied by Q, instead of also being pre-multiplied by Q. LAP can be solved exactly as a quadratic problem with linear constraints (as above, one can relax the constraint to the set of doubly stochastic matrices, and still be guaranteed to obtain 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 $Q^{(k)}$ is the identical matrix, the two gradients are identical. Thus, if QAP is initialized at the identity matrix, the first iteration is identical to LAP. 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



(a) model

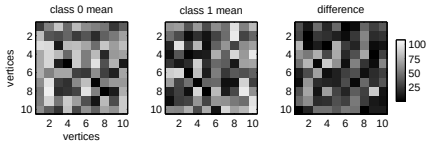


(b) objective function

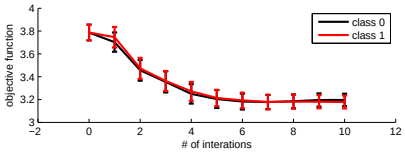


(c) Lhat +/- Lstd

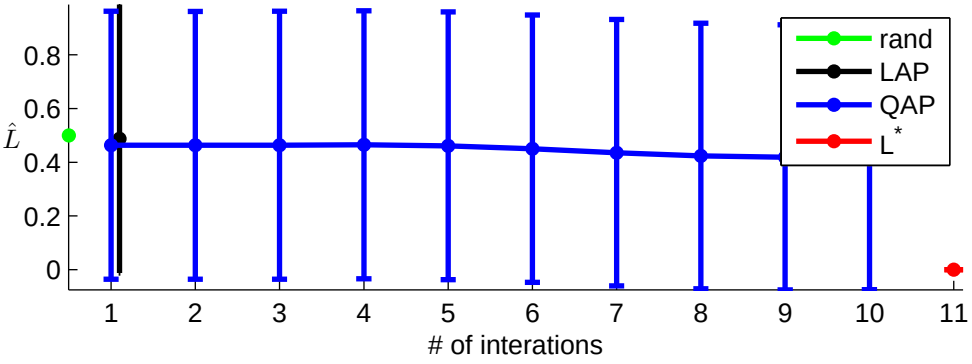
Figure 3: hetero kidney egg model



(a) model



(b) objective function



(c) Lhat +/- Lstd

Figure 4: hetero model

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.

5.4 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 [cite]. We use 49 subjects from the Baltimore Longitudinal Study on Aging, with acquisition and connectome inference details as reported in [cite]. 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 [FACT cite], 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 2 shows leave-one-out misclassification rates for the various strategies.

Table 2: MR Connectome Leave-One-Out Misclassification Rates

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

6 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 2).

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 labeling function, α_V, 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.