PRE-DRAFT: On unlabeled graph classification

jv1,jv2,djm,jmc,cep

May 22, 2011

1 Introduction

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, one can construct an adjacency matrix representation, $A \in \mathcal{A}_E^{V \times V}$. When the edge attributing function is binary, $\mathcal{A}_E = \{0, 1\}$, resulting in a binary adjacency matrix (generalizations are straightforward). Let Y be a Bernoulli covariate: $Y : \Omega \mapsto \{0, 1\}$, yielding graph *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 $\mathcal{F}_{\mathbb{G},\mathcal{Y}} = \{F_{\mathbb{G},Y}(\cdot; \boldsymbol{\theta}) : \theta \in \boldsymbol{\Theta}\}$. 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 ($G \in \mathbb{E}^d$), whereas the object of interest here are graphs ($G \in \mathcal{G}$). In this work, therefore, we propose a 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: $L_h = F[h(\mathbb{G}) \neq Y]$, one can then search for the algorithm $h^* \in \mathcal{H}$ that minimizes the misclassification rate:

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

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^*:

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

where $\approx$ indicates that in general, we might not be able to find the actual minimum in the set $\mathcal{H}$. Regardless, any approach requires estimating 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]. 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 ($k$nn) algorithms [cite] and interpoint-dissimilarity matrix based algorithms [cite], can be straightforwardly applied.

Graph model approach Define a random graph model: $\mathcal{F}_{\mathbb{G}} = \{F_{\mathbb{G}}[\cdot;\boldsymbol{\theta}] : \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$. Given such a model, one could then, for instance, estimate $\boldsymbol{\theta}$ 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: $\phi : \mathcal{G} \mapsto \mathbb{E}^d$. Once in $\mathbb{E}^d$, one can apply one of many possible standard machine learning or other such approaches [Bunke]. Note that using a graph kernel is closely related to graph embedding.

3 Unlabeled sticky wicket

In certain graph classification problems the vertex labels, α_V, are unobserved. 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, in some sense. We consider two complimentary approaches to 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 $k$nn classifier.

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

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

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:

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

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

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

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:

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

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 nonlinear 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:

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

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. Note that Eq. (9) 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:

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

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:

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

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, the "flat doubly stochastic matrix," $J = \mathbf{1}^{\mathsf{T}}\mathbf{1}/V$, is the middle of the feasible region. Therefore, if we run the FW algorithm once, we always start with the flat matrix. 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 call this approach the Multiple Frank-Wolfe (MFW) approach to approximating QAP.

We use the following algorithm to utilize the above approach within a $k$nn classification framework. Given a test adjacency matrix, A, find $\hat{Q}_i^A = \hat{Q}_{QAP}(A, A_i)$ for all n training samples, $\{A_i\}_{i \in [n]}$. Given these solutions, let $\widetilde{A}_i = \hat{Q}_i^A A \hat{Q}_i^{A\mathsf{T}}$ for all i. Now, define a suitable dissimilarity $d : \mathcal{A} \times \mathcal{A} \mapsto \mathbb{R}_+$, one can compute $d(\widetilde{A}_i, A_i)$ for all $i \in [n]$, and then one can sort the distances, $d_{(1)} \leq d_{(2)} \leq \cdot \leq d_{(n)}$. 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. Formally, $\hat{y} = \operatorname{argmax}_y \mathbb{I}\{\sum_{i \in [k_n]} y_{(i)} = y\}$.

[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).

[2]Note that k_n is a function of n, typically chosen so that as $n \to \infty$, $k \to infinity$ but $k/n \to 0$.

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 $k$nn algorithm, given a suitably defined dissimilarity (and rule for k_n).

5 Results

5.1 QAP benchmarks vs. PATH algorithm

Before comparing the QAP and GI approach, we first compare the performance of MFW-QAP with recent state-of-the-art approaches on the QAP benchmark library [Cela07 # 38 in PATH paper]. Specifically, in [PATH] this benchmark was used to compare the PATH algorithm against previous state-of-the-art approaches. In all but two cases the PATH algorithm achieved a lower minimum. For those two, the QPB method of Cremers et al. [18 in PATH] achieved a lower minimum. We compare MFW-QAP with the previous best performing algorithm. In *all* cases, MFW-QAP outperforms the previous best result. In 12 out of 16 cases 75%, the simple FW algorithm outperforms the others. See Table 1 and Figure 1 for quantitative results.

Table 1: Comparison of Frank-Wolfe with Minimum Solution and Path Algorithm

#	Problem	Min	FW$_{100}$	FW$_3$	FW$_2$	FW$_1$	min(PATH,QPB)
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

5.2 Misclassification rate and computational complexity for different model/classifier combo's

model 1: symmetric model in which LAP=1 iter of QAP fig 1: model params

proof that LAP=QAP.

fig 2: numerical confirmation of proof: Lhat vs. # iters of FW

possible simulation details: Homogeneous kidney-egg simulation: For the homogeneous kidney-egg simulations, we assume $n = 10$, so $\eta^y \in (0,1)^{n \times n}$. We further assume a stochastic block model, with two blocks, for both classes. For class 0, $\eta^0_{ij} = q_0 = 0.25$ for $i, j \in [3]$, and for class 1, $\eta^1_{ij} = q_1 = 0.75$ for $i, j \in [3]$. In both classes, $\eta_{ij} = p = 0.5$ for $i, j \in \{4, \ldots 10\}$. See Figure **??** (top row) for a graphical depiction of this model.

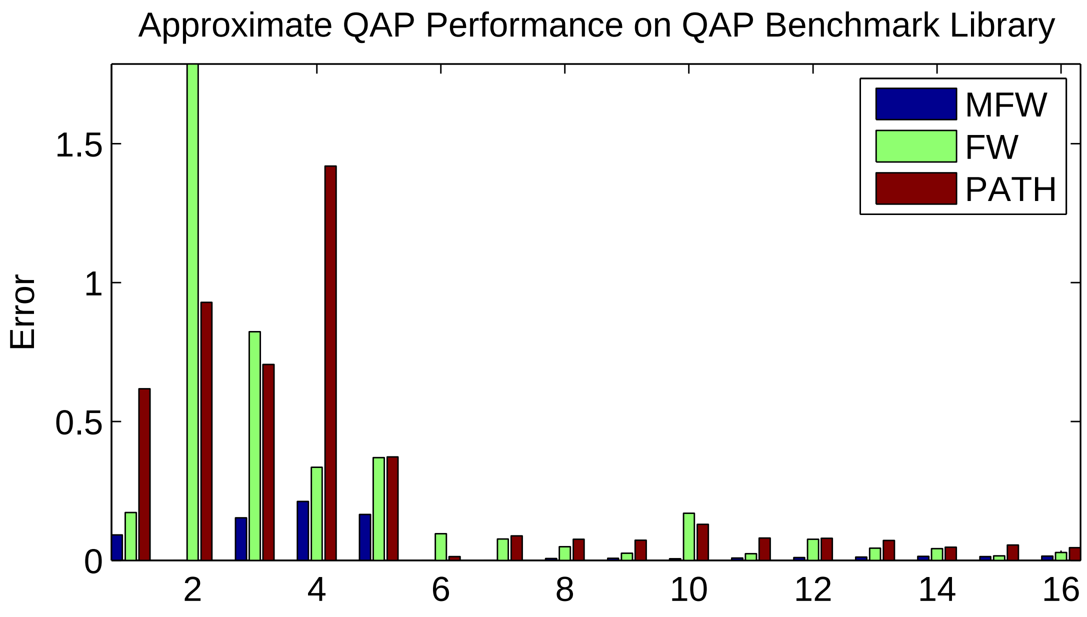

Figure 1: relative performance plot. XXX: would a log yscale be better?

perhaps: model 2: assymetric model in which LAP $<$ QAP (fig 2: show model params) (this one we do depending on the results from the below synthetic data analysis (SDA). if the SDA results are much different than the above results, this warrants a "linking" simulated result.)

perhaps: Heterogeneous simulation: For the heterogeneous simulations, we assume $n = 10$, so $\eta^y \in (0,1)^{n \times n}$. We further assume that each $\eta_{ij}^y \overset{iid}{\sim}$Uniform$(0,1)$. See Figure **??** (bottom) row for a graphical depiction of this model.

same figures as above.

synthetic data 49 subjects: 25 male, 24 female. each with 70x70 symmetric binary matrix.

figure 3: plot mean male, mean female, significant difference matrix.

labeled results: naive bayes performance around 70%. finding signal subgraph performance around 85%.

fig 4: unlabeled results: Lhat vs. # iters of FW (we scrabble labels).

NB: we might want to add a table (like the benchmark results) showing average performance of QAP for these guys

6 Discussion

References