PRE-DRAFT: Graph Classification under an independent edge model

Joshua T. Vogelstein[1], Henry Pao[1] R. Jacob Vogelstein[1,2], and Carey E. Priebe[1]
[1] Johns Hopkins University, Department of Applied Mathematics & Statistics
[2] Johns Hopkins University Applied Physics Laboratory, National Security Technology Department

February 22, 2011

Abstract

The statistical analysis of data that are well represented by networks, or graphs, is a rapidly developing field. In particular, many aspects of the world, including economics, telecommunications, social networks, and transportation grids, to name a few, are well characterized by graphs. While much work has been devoted to studying the statistics of individual graphs, less attention has been given to the analysis of collections of graphs. Our interest here is to develop classifiers that operate directly on graphs, without requiring embedding the graphs into vector spaces. Therefore, our approach is to develop a joint model, $\mathbb{P}[G, Y]$, characterizing the possible distributions of random graphs, G and classes, Y. We study some simple special cases by assuming edges are independent, but not identically, distributed. Based on this model, we develop classifiers that are consistent and more efficient than more naïve classifiers, given sufficient data. In our motivating example, the graphs correspond to brain connectivity, i.e. connectomes, of individuals. These results suggest several avenues for the development of classification algorithms for graphs.

1 Introduction

Current technology facilitates acquiring large swaths of data in myriad diverse fields, ranging from telecommunications to neuroinformatics. As data *collection* technologies become increasingly sophisticated, they beckon an analogous development of data *analysis* technologies. Statistical theory, and in particular, pattern recognition, has therefore received widespread attention and devotion in the recent decades, including an explosion in so-called "machine learning" techniques, including both supervised and unsupervised learning. Supervised learning algorithms have largely focused on problems that loosely satisfy the following assumptions: data has been exchangeably sampled from some distribution: $(x_s, y_s) \overset{exch.}{\sim} \mathbb{P}[X, Y]$, where each $x_s \in \mathcal{X} \subseteq \mathbb{R}^d$ is a "feature vector," $y_s \in \mathcal{Y} \subseteq \mathbb{R}^{d'}$ is a (set of) class variable(s), and $\mathbb{P}[X, Y]$ is some joint distribution [1]. Given these assumptions, one then desires to build a function that utilizes training data to make a prediction of y given a new x, $f : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^S \mapsto \mathcal{Y}$, where S is the number of training samples.

Here we are interested in a slightly different setting. In particular, rather than $x \in \mathbb{R}^d$, we assume that the features form a graph. A graph is a double composed of a set of n vertices, and up to n^2 edges between its vertices. The space of graphs, $\mathcal{G}$, is not Euclidean, because edges share vertices, and therefore, the graphs have structure. Therefore, while one could naïvely apply standard supervised learning algorithms to graph classification, they will discard structural information. Tools designed specifically to operate on graph spaces could perhaps facilitate extracting more information from these data.

To date, most work on these kinds of problems has utilized "graph kernels" [2]. More specifically, the investigator first defines a set of graph kernels, projects each graph into the graph kernel space, and then utilizes standard machine learning techniques [3], typically some kind of boosting algorithm [4] (for example, [5, 6, 7]). [8] and [9] defined various embeddings and then built classifiers based on distance between embedded graphs). [10] and [11] assume edges are independent, and then use standard tools to perform classification. While effective, these methods lack a certain desirable interpretability: for many applications, we desire to understand which edges/vertices convey the signal.

A somewhat different approach is considered here, both explicitly utilizing graph structure and admitting interpretable results. Of primary interest here is the development of consistent classifiers, that is, classifiers guaranteed to

converge to the Bayes optimal classifier with enough data. Moreover, the preference is that these classifiers converge quickly, as data is often limited. Analytics and simulations demonstrate the utility of this framework for classifying graphs. We then apply these classifiers to a real-world application: that of classifying gender based only on "conntectome" data, where each graph corresponds to the macroanatomical structure of a human brain. These classifiers can differentiate gender with better accuracy then more naïve classifiers, in less time, with more interpretability.

2 Methods

2.1 Background

2.1.1 Preliminaries

Upper case latin letters are random variables, $X : \Omega \mapsto \mathcal{X}$, whose samples x take values in the sample space, $\mathcal{X}$, with cardinality $|\mathcal{X}|$. The probability distribution of X will be $\mathbb{P}[X]$, and the probability mass function of X taking value x will be written $\mathbb{P}[x]$. Both vectors and matrices will be indicated by bold notation, $\boldsymbol{x} \in \mathbb{R}^{d \times d'}$.

A random graph, G, takes values $g \in \mathcal{G}$, defined by a tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of $|\mathcal{V}| = n$ vertices, and $\mathcal{E}$ is the set of edges (or arcs) between them. Let Y be a random class, taking values $y \in \mathcal{Y}$. We are particularly interested in scenarios in which $\mathcal{Y} = \{0, 1\}$. Given these definitions, a joint distribution, $\mathbb{P}_{\boldsymbol{\theta}}[G, Y]$, specifies the probability of observing any graph $g \in \mathcal{G}$ (to be defined below) and any class $y \in \mathcal{Y}$, with parameter $\boldsymbol{\theta} \in \boldsymbol{\Theta}$. The model is the collection of all possible joint distributions under consideration, $\mathcal{P} = \{\mathbb{P}_{\boldsymbol{\theta}}[G, Y] | \boldsymbol{\theta} \in \boldsymbol{\Theta}\}$. Let $\mathbb{P}[g|y]$ indicate the *likelihood* of observing g given y, $\mathbb{P}[y]$ denote the *prior* probability of observing y, and $\mathbb{P}[y|g]$ be the *posterior* probability of observing y given g (note that we sometimes drop the $\boldsymbol{\theta}$ subscript for brevity).

Throughout, data is assumed to be sampled exchangeably from some true (but typically unknown) distribution, $x \overset{exch.}{\sim} \mathbb{P}_{\boldsymbol{\theta}}[G, Y]$. A collection of S training samples is denoted by $\mathcal{T}_S = \{(g_s, y_s)\}$. Let $\mathcal{S}_y$ indicate the set of data points in class y, and $|\mathcal{S}_y| = S_y$.

A parameter estimate uses some data to obtain an estimate of the true (but typically known) parameter $\boldsymbol{\theta}^*$, $\hat{\boldsymbol{\theta}}_S : \mathcal{X}^S \mapsto \boldsymbol{\Theta}$. An unbiased estimator is one for which its expectation equals the true parameter value: $\mathbb{E}[\hat{\boldsymbol{\theta}}_S] = \boldsymbol{\theta}^*$. An asymptotically unbiased estimator is one for which $\mathbb{E}[\hat{\boldsymbol{\theta}}_S] \to \boldsymbol{\theta}^*$ as $S \to \infty$. Technically, this is a *sequence* of estimators, as each estimator is a function of S data points, so they have different domain spaces, and are therefore different functions. A consistent estimator (sometimes called an asymptotically consistent estimator) is a sequence of estimators that converges in probability to $\boldsymbol{\theta}^*$. Formally, an estimator is consistent if and only if $\lim_{S \to \infty} \mathbb{P}[\hat{\boldsymbol{\theta}}_S = \boldsymbol{\theta}^*] = 1$.

2.1.2 Basic classification theory

In the graph classification setting, we define a graph classifier as any function that takes as input a graph g and outputs an expected class, $f : \mathcal{G} \mapsto \mathcal{Y}$, when $\mathcal{Y}$ is discrete. Graph classification quality is assessed by misclassification rate:

$$L_f = \mathbb{P}[f(G) \neq Y] = \int_{g \in \mathcal{G}} \mathbb{P}[f(g) \neq y] \mathbb{P}[g] dg. \tag{1}$$

We would like to find a graph classifier, f^*, with minimum misclassification rate, also called the Bayes optimal graph classifier. It can be shown that selecting the class that maximizes the class-conditional posterior is Bayes optimal [1]:

$$\hat{y} = f^*(g) = \underset{f \in \mathcal{F}}{\operatorname{argmin}} L_f(g) = \underset{y \in \{0,1\}}{\operatorname{argmax}} \mathbb{P}[y|g] = \underset{y \in \{0,1\}}{\operatorname{argmax}} \mathbb{P}[g|y] \mathbb{P}[y] \tag{2}$$

where $\mathcal{F}$ is the space of all possible classifiers. The misclassification rate of the Bayes optimal graph classifier is called the *Bayes error* (or *Bayes risk*). Because f^* is typical unknown, one can approximate f^* by utilizing training data to construct an classifier estimate: $\hat{f}(\cdot; \mathcal{T}_S) : \mathcal{G} \times (\mathcal{G} \times \mathcal{Y})^S \mapsto \mathcal{Y}$. A *Bayes plug-in* classifier first estimates the likelihood $\mathbb{P}[G|Y]$ and prior, $\mathbb{P}[Y]$, and then plugs them in to (2) to obtain:

$$\hat{y} = \underset{y \in \{0,1\}}{\operatorname{argmax}} \hat{\mathbb{P}}[g|y] \hat{\mathbb{P}}[y]. \tag{3}$$

Assessing the quality of an estimated classifier is a sticky wicket, as the integral in (1) is typically intractable without an infinite amount of data. Instead, we typically approximate this integral using a (sub-)sampling procedure. In

particular, select subsets of the data: $\{\mathcal{T}_{s_1}, \ldots, \mathcal{T}_{S_C}\}$, where each $\mathcal{T}_{S_c} \subseteq \mathcal{T}_S$, and compute the *cross-validated error*, an estimate of the misclassification rate for an estimated classifier:

$$\hat{L}_{\hat{f}(\cdot;\mathcal{T}_S)} = \sum_{c=1}^{C} P[\hat{f}(g;\mathcal{T}_{S_c}) \neq y] P[\mathcal{T}_{S_c}], \tag{4}$$

noting that (4) generalizes the ideas of "leave-one-out" and related approaches by allowing any sampling strategy, any size subsets, and any number of subsamples. A consistent classifier is one that converges to Bayes classifier, that is: $\lim_{S\to\infty} \mathbb{P}[L_{\hat{f}} = L_{f^*}] = 1$.

3 Models

Let $\mathbb{P}_{\boldsymbol{\theta}}[G]$ indicate the probability distribution over graphs. We assume that our collection of graphs shares the same set of *labeled* vertices, $\mathcal{V}$, so that all the variability is in the edge list. We then make the following assumptions about the edges. First, edges are binary random variables, so $a_{uv} = 1$ whenever there is an edge from vertex u to vertex v, and $a_{uv} = 0$ otherwise. Second, loops are forbidden, so $a_{uu} = 0\ \forall u \in \mathcal{V}$. Third, the graphs are undirected, so $a_{uv} = a_{vu}$ $\forall u, v \in \mathcal{V}$. Collectively, these three assumptions means that the graphs under investigation are *simple graphs*, although the 2^{nd} and 3^{rd} assumption can be relaxed in everything that follows without any substantive changes.

Because both the possible values of each edge and the number of vertices are finite, the number of possible graphs is also finite. More specifically, $|\mathcal{E}| = d_n = \binom{n}{2}$ for simple graphs (if an edge is possibly in the edge list, then we write $u \sim v \in \mathcal{E}$). One can therefore define a probability mass function, $\mathbb{P}_{\boldsymbol{\theta}}[G]$, characterizing the distribution over graphs. The most general model for these random graphs is therefore a categorical random variable, which assigns a probability to each possible graph. Because there are 2^{d_n} possible simple graphs with n vertices, this categorical model would have a parameter with $2^{d_n} - 1$ dimensions.

Clearly, estimating such a parameter from data will be intractable in all but the smallest graphs, as 2^{d_n} is superexponential in n (for example, when $n = 10$, $2^{d_n} > 10^13$). Thus, we search for simpler models, for which we hope to be able to achieve parameter estimates sufficiently accurate to enable good classification accuracy. There is a delicate balance here, we desire a model that is sufficiently complex to facilitate accurate classification, but sufficiently simple to enable both estimation given the amount of data we expect to collect, as well as interpretability. Below, we describe several such models, each with increasing levels of complexity.

3.1 Identical and independent edge model

Perhaps the simplest and most well-known random graph model one could assume is the Erdös-Rényi (ER) random graph model, which asserts that each edge is independent and identically distributed (i.i.d.): $\mathbb{P}[A_{uv} = p]$, $\forall u \sim v \in \mathcal{E}$ [**?**, **?**]:

$$\mathbb{P}_{\boldsymbol{\theta}}[G] = \mathbb{P}_{\boldsymbol{\theta}}[\boldsymbol{A}] = \prod_{u\sim v\in\mathcal{E}} \mathbb{P}_{\boldsymbol{\theta}}[A_{uv}] = \prod_{u\sim v\in\mathcal{E}} \text{Bern}(a_{uv}; p) = \prod_{u\sim v\in\mathcal{E}} a_{uv}^{p}(1-a_{uv})^{1-p}, \tag{5}$$

where $\boldsymbol{\theta} = p$. Thus, we have reduced the dimensionality of the parameter from 2^{d_n} to 1. It seems unlikely that this simple model will suffice for all but the simplest classifications problems. We therefore generalize this i.i.d. model by relaxing the second 'i', namely, letting edges by independent, but not identically distributed.

3.2 Incoherent edge model

Perhaps the simplest generalization of the above totally "homogeneous" model is to allow edges to be independent, but some have probability p and others have probability q. Moreover, if the edges with probability q are scattered "incoherently" around the adjacency matrix, we call this an *incoherent edge model*. More formally, we consider the set of m edges, $\mathcal{E}_{inc} = \{A_{uv} | u \sim v \in \mathcal{E}_{inc}\}$, where $|\mathcal{E}_{inc}| = m$, each of which has probability q, and the rest have probability p. Thus, we have the following model:

$$\mathbb{P}_{\boldsymbol{\theta}}[G] = \prod_{u\sim v\notin\mathcal{E}_{inc}} \text{Bern}(a_{uv}; p) \prod_{u\sim v\in\mathcal{E}_{inc}} \text{Bern}(a_{uv}; q). \tag{6}$$

The parameters of this model are therefore: $\boldsymbol{\theta}_{inc} = (p, q, \mathcal{E}_{inc})$. Note that $\mathcal{E}_{inc}$ could be thought of as a binary matrix, indicating for each edge whether its parameter was p or q.

3.3 Coherent edge model

In the above model, the edges with probability q were incoherently scattered across all edges. Here, we assume that all the edges with probability q share γ common vertices. More formally, $\mathcal{E}_* = \{A_{uv} | u \in \mathcal{V}_* \text{ or } v \in \mathcal{V}_*\}$, where $\mathcal{V}_*$ is the set of "star" vertices, and $|\mathcal{V}_*| = \gamma$. This leads to a model identical to (6), except replace the $\mathcal{E}_{inc}$ with $\mathcal{E}_*$. The parameter of the coherent model is therefore: $\boldsymbol{\theta}_* = (p, q, \mathcal{E}_*)$. The top 3 panels of Figure 1 shows examples of all three models.

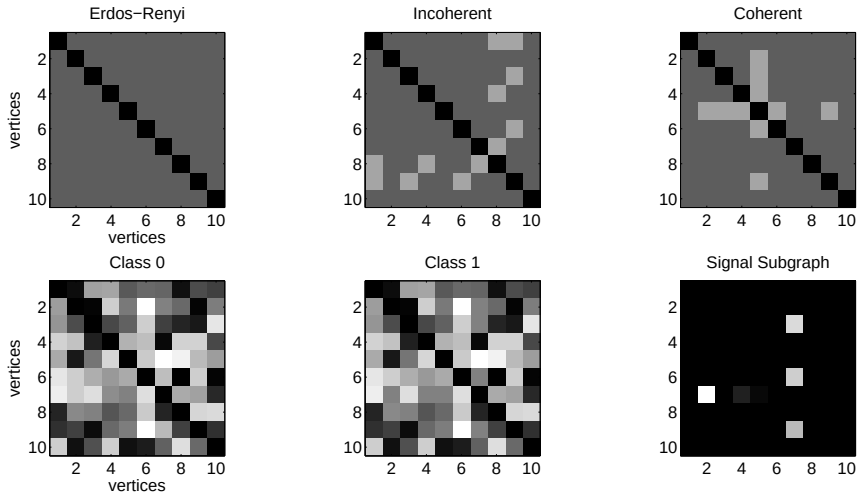

Figure 1: Schematic depicting the probability of each edge for the various models. Top panels show the various homogeneous models. The bottom panels show two heterogeneous class-conditional distributions, as well as their difference.

3.4 Heterogeneous model

In the above models, while each edge was independent, the each edge's probability was either completely homogeneous (in the identical and independent model) or largely homogeneous (in both the incoherent and coherent edge models). Here we also consider the heterogeneous model, in which each edge is sampled according to some probability, yielding:

$$\mathbb{P}_{\boldsymbol{\theta}}[G] = \mathbb{P}_{\boldsymbol{\theta}}[\boldsymbol{A}] = \prod_{u \sim v \in \mathcal{E}} \mathbb{P}_{\boldsymbol{\theta}}[A_{uv}] = \prod_{u \sim v \in \mathcal{E}} \text{Bern}(a_{uv}; p_{uv}) = \prod_{u \sim v \in \mathcal{E}} a_{uv}^{p_{uv}} (1 - a_{uv})^{1 - p_{uv}}. \tag{7}$$

In some sense, this is the most general model described so far. In another sense, it has less structure, is harder to estimate, and is a model selection problem (finding which edges have probability q, and how many such edges are there).

3.5 Joint models

The above models only characterize the distribution of graphs, not the distribution of graphs and classes, $\mathbb{P}[G, Y]$. For Bayes risk to be non-zero, the two classes must have some difference: $\mathbb{P}[G|Y = 0] \neq \mathbb{P}[G|Y = 1]$. In independent edge models, this means that for at least one edge, $\mathbb{P}[A_{uv}|Y = 0] \neq \mathbb{P}[A_{uv}|Y = 1]$. Moreover, the above models can be considered models of the class-conditional differences, that is: (i) all, (ii) an incoherent collection, or (iii) a coherent collection of edges could have different distributions (or some combination thereof). Let the *signal subgraph*, denoted by $\mathcal{E}_s$, be the collection of edges with class-conditional differences, that is: $\mathcal{E}_s = \{u \sim v | p_{uv|0} \neq p_{uv|1}\}$. In such a scenario, the edge probabilities could be totally heterogeneous. The bottom panel of Figure 1 shows an example of two class-conditional models with a coherent signal subgraph.

4 Bayes Plug-in Classifiers

The Bayes optimal graph classifier for all the above models (which assume all edges are independent), is given by:

$$\begin{aligned} f(g) &= \underset{y}{\operatorname{argmax}}\, \mathbb{P}[g, y] = \underset{y}{\operatorname{argmax}}\, \mathbb{P}[g|y]\mathbb{P}[y] \\ &= \underset{y}{\operatorname{argmax}} \prod_{u\sim v\in\mathcal{E}} \mathbb{P}[a_{uv}|p_{uv|y}]\mathbb{P}[y] = \underset{y}{\operatorname{argmax}}\, \mathbb{P}[y] \prod_{u\sim v\in\mathcal{E}} \text{Bern}(a_{uv}; p_{uv|y}) \\ &= \underset{y}{\operatorname{argmax}}\, \pi^y \prod_{u\sim v\in\mathcal{E}} a_{uv}^{p_{uv|y}}(1-a_{uv})^{1-p_{uv|y}}, \end{aligned} \tag{8}$$

where $\pi^Y = \mathbb{P}[Y]$, and $p_{uv|y}$ is the probability $a_{uv} = 1$ in class y. In practice, neither π_y nor $\{p_{uv|y}\}$ are known. The Bayes plugin estimator therefore estimates both, and plugs them in to Eq. (8). Below, we provide details for estimating both.

4.1 Parameter estimates

The maximum likelihood estimate of π_y is simply the sum of graphs in class y, dividing by the total number of graphs: $\hat{\pi}_y = S_y/S$. The maximum likelihood estimate (MLE) of $\hat{p}_{uv|y}$ is the mean of a_{uv}, averaged over all i in class y:

$$\hat{p}_{uv|y}^{MLE} = \frac{1}{S_y} \sum_{i\in\mathcal{S}_y} a_{uv}^{(i)}, \tag{9}$$

where $a_{uv}^{(i)}$ indicates the value of edge $u \sim v$ in graph i. When S is relatively small, $\hat{p}_{uv|y}$ can be zero, which is problematic for a plug-in classifier. Therefore, we use an estimator that is both robust to model misspecifications and smoothes the estimate away from zero without adding much bias. An M-estimator is any estimator that maximizes a certain contrast function:

$$\hat{p}_{uv|y} = \underset{p}{\operatorname{argmin}} \sum_{i\in\mathcal{S}_y} \rho(a_{uv}^{(i)}, p) \tag{10}$$

where $\rho(a_{uv}^{(i)}, p)$ is called a contrast function. For instance, the MLE uses $\rho(\cdot, p) = -\log \mathbb{P}[\cdot]$. Here, we propose a slightly modified estimator:

$$\hat{p}_{uv|y} = \frac{\sum_{i\mathcal{S}_y} a_{uv}^{(i)} + 1/(2S)}{S + 1/(2S)}, \tag{11}$$

so that the parameter estimate is never actually zero. Eq. (11) implicitly defines the following contrast function

$$\rho(a_{uv}^{(i)}, \boldsymbol{\theta}) = -\frac{a_{uv}^{(i)} + 1/(2S)}{S + 1/(2S)}. \tag{12}$$

Other estimators with smoothing and robustness, include the *maximum a posteriori* estimators, which we do not consider here, other than to acknowledge their existence and potential great utility.

4.2 Signal Subgraph Search

The naïve Bayes classifier takes $\mathcal{E}$ to be all $\binom{n}{2}$ edges. If a signal subgraph exists, however, one can outperform the naïve Bayes classifier. In particular, if one assumes that edges are independent in both classes, but that only a small subset of edges differ between the two classes, $\mathcal{E}_s = \{u \sim v | p_{uv|0} \neq p_{uv|1}\}$, then one can use this information to obtain a better classifier, by only looking at the signal subgraph:

$$f(g) = \underset{y}{\operatorname{argmax}}\, \pi^y \prod_{u\sim v\in\mathcal{E}_s} a_{uv}^{p_{uv|y}}(1-a_{uv})^{1-p_{uv|y}}. \tag{13}$$

This approach does not depend on homogeneity of edges, nor the coherency, as each edge $a_{uv}^{(i)}$ could be sampled according to its own potentially unique distribution $p_{uv|y}$. Below we provide several approaches to searching for the signal subgraph.

4.2.1 Exhaustive search for signal subgraphs

The number of signal subgraphs is equal to the number of graphs in the random graph family, $|\mathcal{G}| = 2^d$, where d is around n^2 depending on assumed constraints (see Section 2.1.1 for details). Thus, one could enumerate all possible signal subgraphs, $\{\mathcal{E}_1, \ldots, \mathcal{E}_{2^d}\}$, and compute $\hat{L}_{f_{\mathcal{E}_c}(\cdot;\mathcal{T}_S)}$ for each $c \in [2^d]$. Finally, let $\hat{c} = \operatorname{argmin}_c \hat{L}_{f_{\mathcal{E}_c}(\cdot;\mathcal{T}_S)}$. Unfortunately, even when V is relatively small (e.g., ≈ 10), 2^d is quite large ($\approx 10^{30}$), making this approach computationally intractable. Also, this approach depends on the particular classification algorithm. It is therefore often desirable to be able to search more efficiently for signal subgraphs independent of the classifier.

4.2.2 Incoherent signal subgraph search

In the face of such a large subspace, many algorithms have been developed to find approximately optimal subspaces, including most prominently so-called forward search and backwards prune strategies [12]. In general, these (greedy) strategies have no guarantees of consistency even though they can be quite computationally intensive.

However, given the independent edge assumption, we can compute the significance of each edge independently, to obtain a rank ordering of edges. More specifically, given $p_{uv|0}$ and $p_{uv|1}$ for all $u \sim v \in \mathcal{E}$, one can compute the dissimilarity, $\delta_{uv} = \delta(p_{uv|1}, p_{uv|0})$, which conveys the difference in position between the two classes. δ_{uv} is thus an uncorrected test-statistic. Because δ is a dissimilarity, it satisfies: (i) $\delta(p_{uv|1}, p_{uv|0}) \geq 0$, (ii) with equality if and only if $p_{uv|1} = p_{uv|0}$), and (iii) $\delta(p_{uv|1}, p_{uv|0}) = \delta(p_{uv|0}, p_{uv|1})$.

Given a suitably defined dissimilarity, it should be non-zero for any edge in the signal subgraph, and identically zero otherwise, that is: $\delta_{uv} > 0 \, \forall u \sim v \in \mathcal{E}_s$ and $\delta_{uv} = 0 \, \forall u \sim v \notin \mathcal{E}_s$. Thus, if one had the true δ_{uv}'s, finding the signal subgraph would be trivial: all edges with non-zero δ_{uv} are in the signal subgraph. Unfortunately, because p_{uv} is unobserved, δ_{uv} must be estimated.

We consider three dissimilarity metrics: (i) absolute magnitude, (ii) z-score, and (iii) a Fisher's exact test. The absolute magnitude estimator for δ_{uv} is simply $\hat{\delta}^a_{uv} = |\hat{p}^1_{uv} - \hat{p}^1_{uv}|$, where $|\cdot|$ indicates the absolute value.

The absolute magnitude dissimilarity does not consider the variance of the estimators. In particular, the variance of the estimators $\hat{p}$ is a function of the true p, because it has a binomial distribution: $\hat{p}_{uv|y} \sim \text{Binomial}(S_y, p_{uv|y})$. Therefore, it would be desirable to scale the confidence of the difference between the two classes by the uncertainty around each estimate. One option is to normalize each estimate by its variance:

$$\hat{\delta}^z_{uv} = \left| \frac{\hat{p}_{uv|1}}{\hat{p}_{uv|1}(1 - \hat{p}_{uv|1})} - \frac{\hat{p}_{uv|0}}{\hat{p}_{uv|0}(1 - \hat{p}_{uv|0})} \right|, \tag{14}$$

which is akin to a z-score when estimators have a Gaussian distribution, which is the most powerful test statistic in that domain.

Finally, Fisher's exact test computes an exact test statistic comparing the distribution of categorical variables in two classes against the null distribution that the two classes have the same distribution [**?**]. Specifically, Fisher showed that the probability of obtaining a particular distribution of categories in two classes is given by the hypergeometric distribution: $\hat{\delta}^F_{uv} = \binom{S_0}{\#_{uv|0}} \binom{S_1}{\#_{uv|1}} / \binom{S}{\#_{uv}}$, where $\#_{uv|y} = \sum_{i \in \mathcal{S}_y} a^{(i)}_{uv}$ and $\#_{uv} = \sum_{i \in \mathcal{S}} a^{(i)}_{uv}$.

Regardless of which dissimilarity metric we use, one can then rank them, $\delta_{(1)} \geq \ldots \geq \delta_{(|\mathcal{E}_s|)}$. If the number of edges in the true signal subgraph, m, were known, then one could choose the m most significant dissimilarities, $\hat{\delta}_{(1)}, \ldots, \hat{\delta}_{(m)}$. Under the independent edge model, using Fisher's exact test with m edges is the optimal estimate of $\mathcal{E}_s$. Note that this approach is a strict generalization of the naïve Bayes classifier, in that by letting $m = V$, one recovers the naïve Bayes classifier.

4.2.3 Coherent signal subgraph search

When the signal subgraph is expected to have some structure, we can utilize this prior information to improve our search. Specifically, assume that class-conditional difference forms a coherent model. In this case, instead of looking for *edges* to define the signal subgraph, one can look for anomalous *vertices*. In particular, the vertices that compose $\mathcal{V}_*$ should have a larger number of significant edges than the vertices not in $\mathcal{V}_*$. We therefore devise Algorithm 1.

Note that by letting $\gamma = V$, one recovers the incoherent signal subgraph search algorithm, so this approach is a generalization of that one. Moreover, this strategy is robust to a vertex having some important edges, and others wholly unimportant, as only the important ones form the signal subgraph.

Algorithm 1 Coherent signal subgraph search

Input: γ, m

- Compute $\hat{\delta}_{uv} \, \forall \, u \sim v \in \mathcal{E}$
- Compute the number of edges incident to vertex v with significance less than w: $Z_v(w) = \sum_v \mathbb{I}\{\hat{\delta}_{uv} \leq w\} + \sum_v \mathbb{I}\{\hat{\delta}_{vu} \leq w\}$
- Find the minimum w such that there are m total edges incident to γ vertices with significance maximally w: $w_\gamma = \min_w$ s.t. $\sum_{v \in [\gamma]} Z_v(w) = m$
- Let the signal subgraph contain those edges.

Output: $\mathcal{E}_s(\gamma, m)$

4.3 Model selection

When γ and m are known, the above algorithms may be used as described above. However, in general these hyper-parameters will be unknown. To estimate them from the data, we build classifiers for all possible choices of γ and m, and choose the best one.

5 Results

5.1 Consistency

We have three different questions of consistency here, consistency of (i) the estimator, $p_{uv|y}$, (ii) the signal subgraph $\mathcal{E}_s$, and (iii) the classifier, $f(\cdot, \mathcal{T}_S)$.

First, we prove consistency of our estimator, as defined by Eq. (11):

Theorem 1. *If a is Bernoulli distributed with probability p, then $\hat{p}$ is a consistent estimator for p, where $\hat{p}$ is defined by Eq.* (11)*.*

Proof. To prove that an estimator is consistent, it is sufficient to show that it converges to another estimator known to be consistent. The maximum likelihood estimator (MLE) is consistent for a Bernoulli random variable:

$$\hat{p}_S = \frac{1}{S} \sum_{s=1}^{S} a_s. \tag{15}$$

Moreover, the estimator defined by Eq. (11) converges to the MLE:

$$\begin{aligned} \mathbb{E}\left[\frac{\sum_{s \in [S]} a_s + 1/(2S)}{S + 1/(2S)}\right] &= \frac{\mathbb{E}[\sum_{s \in [S]} a_s] + 1/(2S)}{S + 1/(2S)} \\ &= \frac{\mathbb{E}[\sum_{s \in [S]} a_s]}{S + 1/(2S)} + \frac{1/(2S)}{S + 1/(2S)} \end{aligned} \tag{16}$$

As $S \to \infty$, the second term converges to zero, and $S + 1/(2S)$ converges to S, yielding the MLE. Thus, this estimator is consistent. □

Second, both the incoherent and coherent signal subgraph algorithms are consistent estimators of the signal subgraph. To see this, first consider the absolute magnitude dissimilarity measure, $\delta_{uv} = \left|p_{uv|0} - p_{uv|1}\right|$. When using the true (but unknown) values of $p_{uv|y}$, $\delta_{uv} > 0$ for all $u \sim v \in \mathcal{E}_s$, and $\delta_{uv} = 0$ otherwise. When estimating $p_{uv|y}$'s, as long as the estimates are consistent, then the estimate of δ_{uv} is consistent, by virtue of consistency being closed under addition. Therefore, when using the absolute value distance, the incoherent signal subgraph search (assuming the size of the signal subgraph is known), is a consistent estimator of the signal subgraph. The same argument applies to the coherent signal subgraph search. Finally, this is true also when using the z-score estimator and Fisher's exact test, by a similar argument.

Third, a Bayes plugin classifier is consistent as long as the parameter estimates used to plug-in are consistent, therefore, the Bayes plugin classifiers that we employ are all consistent classifiers.

Given that each classifier is consistent, a natural question to ask is which is more efficient, that is, which classifier converges more quickly to the truth (as $S \to \infty$). More specifically, that classification performance scales with signal subgraph estimation performance. Then, we ask under what conditions will the coherent signal subgraph search be more efficient than the incoherent signal subgraph search.

5.2 Monotonicity Proofs

In IE1, using k canonical dimensions recovered from the training data ($|\hat{\mathcal{E}}| = k$), the probability of misclassification is monotonically decreasing as a function of $T = |\mathcal{E} \cap \hat{\mathcal{E}}|$ that is

$$t_1 > t_2 \Rightarrow E[L(g_{\hat{\mathcal{E}}})|T = t_1] < E[L(g_{\hat{\mathcal{E}}})|T = t_2].$$

5.3 Approximate Asymptotic distribution of T

Note that $n\hat{p}_{0;i,j}$ and $n\hat{p}_{1;i,j}$ are binomals, thus they can be approximated with the following normal approximations for n sufficiently large.

$$\hat{p}_{0;i,j} \approx N_{0;i,j} \sim \text{Normal}(p_{0;i,j}, p_{0;i,j}(1 - p_{0;i,j})s/2)$$
$$\hat{p}_{1;i,j} \approx N_{1;i,j} \sim \text{Normal}(p_{1;i,j}, p_{1;i,j}(1 - p_{1;i,j})s/2)$$

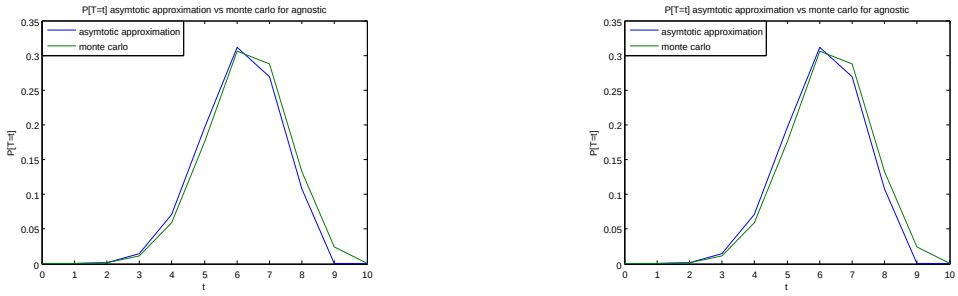

Figure 2: Left: A comparison of a monte carlo simulation and asymtotic distribution of the agnostic method with $n = 10$, $m = 10$, $k = 10$, $p = 0.45$, $q = 0.55$, $s_0 = s_1 = 100$. Right: A comparison of a monte carlo simulation and asymtotic distribution of the max degree method with $n = 20$, $m = 10$, $k = 10$, $p = 0.45$, $q = 0.55$, $s_0 = s_1 = 100$.

5.4 Relative Efficiency

$$T_x(k, s, F_{GY}) = \left|\mathcal{E} \cap \hat{\mathcal{E}}_s\right| \tag{17}$$

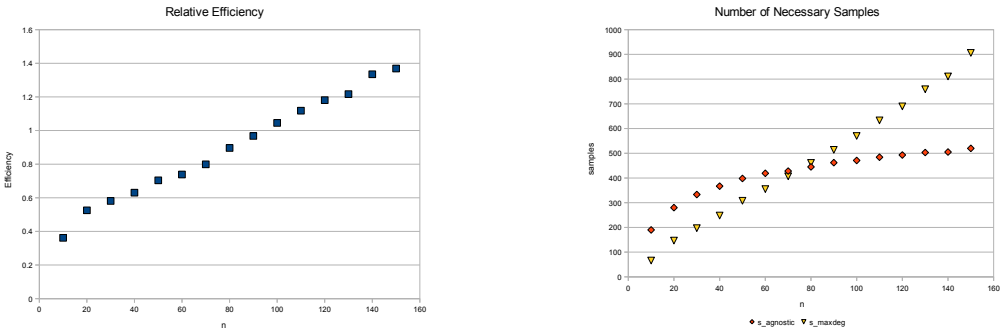

Figure 3: Left: Relative efficiency simulation with $p = 0.1$, $q = 0.2$. Right: s values of the agnostic and max degree models with $p = 0.45$, $q = 0.55$.

6 Connectome classification results

We apply the above classification procedure to connectome data. Briefly, diffusion MRI (dMRI) data was collected from 50 subjects as part of the Baltimore Longitudinal Study on Aging as described in [**?**]. The dMRI data was then processed using the JIST/CATNAP framework [**?**] to obtain 70×70 element binary, symmetric, and hollow adjacency matrices. Of the 50 subjects, 21 were male, and 29 were female. We estimated the coherent signal subgraph for all possible values of γ, m as described using Algorithm 1, using δ_{uv}^F as the dissimilarity measure, and estimated $\{p_{uv|y}\}$ using Eq. (11). We then plugged these estimates into Eq. (13). The top panel of Figure 4 shows the performance of each classifier as a function of γ and m. Note that performance is relatively robust to small changes in either hyper-parameter. The bottom left panel shows the best obtained $\hat{L}$ for each value of γ. If the signal subgraph was completely incoherent, than the best performance would be expected at $\gamma = n$, that is, the incoherent classifier, but clearly that is not the case: the best performance occurs at $\gamma = 7$. The right panel shows the full incoherent classifier performance as a function of m. Note that this is identical to the bottom line of the top panel. If the naïve Bayes classifier were best, than the smallest $\hat{L}$ would be obtained at $m = \binom{n}{2} = d = 2415$. However, of the d edges, only around 1500 were different across the two classes, so the naïve Bayes classifier performance clearly cannot improve over some incoherent signal subgraphs, and the best performance occurs with $m = 2$, which seems like noise, and is still not as small as the best values obtained using the coherent signal subgraph search method.

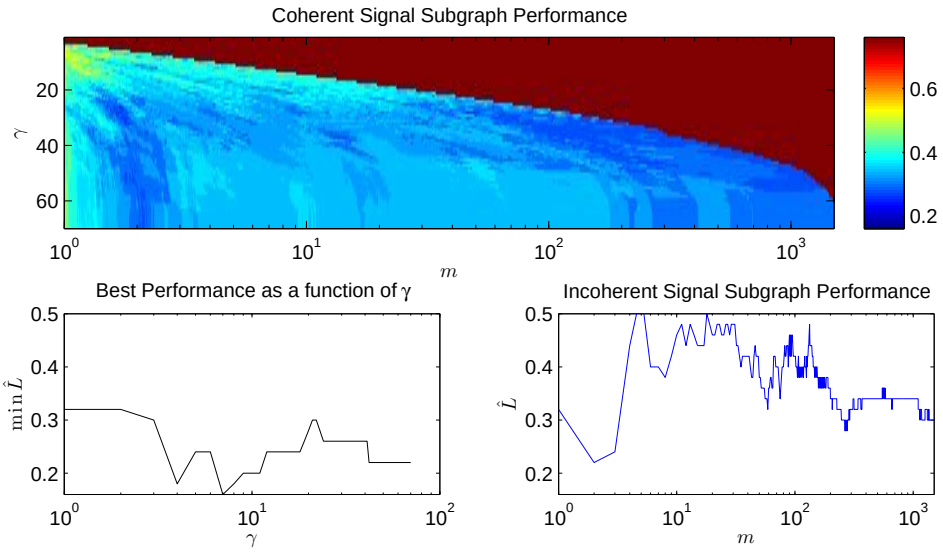

Figure 4: Classifier performance

7 Discussion

This work contributes novel, model-based graph classification algorithms. The algorithms are proven to be consistent, and approach Bayes optimal performance faster than the naïve Bayes approach when the model assumptions are correct, given enough data. However, when the number of data points is relatively small, a misspecified model is *expected* to outperform the correctly specified model, because it wins the bias-variance trade-off. We then show that this approach indeed improves upon the naïve Bayes performance for a real application: gender classification using dMRI data.

The improved performance of the coherent signal subgraph classifier over both the naïve Bayes classifier and the incoherent classifier leads us to wonder how coherent the gender signal subgraph is. To visualize coherency, we generate a coherogram, which shows the number of edges incident to each vertex as w is increased from its minimum to its maximum value. Figure 5 shows that the gender signal subgraph is not that coherent. Nonetheless, the coherent signal subgraph classifier outperforms the naïve Bayes classifier.

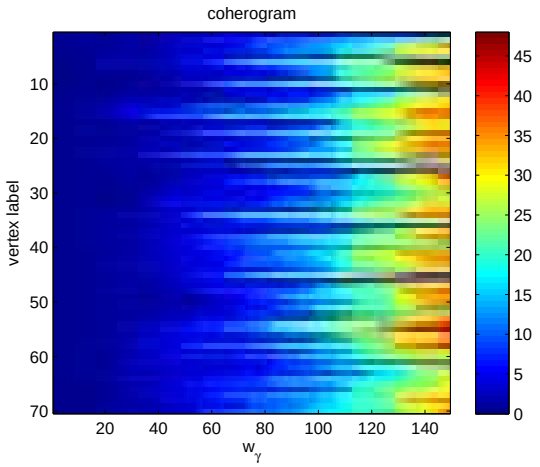

Figure 5: coherogram

These results may be significant for a number of reasons. First, as graphs are becoming an increasingly dominant representation of data, random graph models and classifiers based on those models will perhaps become increasingly important tools in the data analysts toolbox. Second, the performance of each classifier is generally believed to be a function of the true distribution from which the data was sampled. However, this work shows that the best performance may be achieved by a misspecified model, whenever the amount of data available is insufficient to fit the true model. More specifically, to predict which classifier will perform best, it is not sufficient to simply assume independence or not, but rather, both the size of the data and the amount data available are key determinants of prediction which classifiers will be best.

Finally, while our signal subgraph searches are not exhaustive, when the edges truly are independent, our searches are optimal. Even when they are not, our search methods are in fact robust estimators (M-estimators) of the true signal subgraph. Regardless, the model assumptions made in this work can be straightforwardly generalized in a couple ways. First, one can imagine a signal subgraph that is partially coherent and partially incoherent, such as perhaps the gender signal subgraph. An algorithm to find the signal subgraph in this scenario would merely add some number of incoherent edges to the coherent signal subgraph. Second, the signal subgraph could be thought of as a latent feature, that is, each vertex or edge could have a binary latent feature, indicating whether or not it is in the signal subgraph. Estimating the signal subgraph could therefore be cast as a latent feature inference problem. Recent works analyzing estimating latent feature vectors for random graph models include [**?**].

The coherent signal subgraph model is closely related to conditionally independent edge models, which specify that the probability of an edge between any pair of vertices is a function of each vertex's feature vector. In the coherent signal subgraph, one could assume that the feature vector for the $\mathcal{V}_*$ vertices are identical across classes, whereas the feature vectors for the other vertices differ across classes. Therefore, future work will investigate more directly estimating feature vectors for the purpose of classification.

Acknowledgments

The authors would like to acknowledge helpful discussions with ...

References

[1] L. Devroye, L. Györfi, and G. Lugosi, *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

[2] T. Gartner, "A survey of kernels for structured data," *ACM SIGKDD Explorations Newsletter*, vol. 5, no. 1, p. 58, 2003.

[3] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[4] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," pp. 23–37, 1995.

[5] H. Kashima and A. Inokuchi, "Kernels for graph classification," vol. 2002, 2002.

[6] A. I. Hisashia Kashima, Koji Tsuda, "Marginalized kernels between labeled graphs," *International Conference on Machine Learning*, 2003.

[7] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification," in *Proc. of NIPS*, pp. 729–736, Citeseer, 2004.

[8] H. Bunke and K. Riesen, "Graph Classification Based on Dissimilarity Space Embedding," in *Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, p. 1007, Springer, 2008.

[9] K. Riesen and H. Bunke, "Graph classification by means of lipschitz embedding.," *IEEE Trans Syst Man Cybern B Cybern*, vol. 39, pp. 1472–1483, Dec 2009.

[10] P. Flach and N. Lachiche, "Naive Bayesian classification of structured data," *Machine Learning*, vol. 57, no. 3, pp. 233–269, 2004.

[11] E. Trentin and E. D. Iorio, "Classification of graphical data made easy," *Neurocomputing*, vol. In Press, Corrected Proof, pp. –, 2009.

[12] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on knowledge and data engineering*, pp. 491–502, 2005.