Classifying random graphs with independent edges

Joshua T. Vogelstein, Henry Pao R. Jacob Vogelstein, and Carey E. Priebe

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 websites, 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 populations of graphs. Our interest here is to develop classifiers that operate directly on graphs, without requiring embedding the graphs into vector spaces, or extracting features. Therefore, our approach is to develop a joint model, $\mathbb{P}[G, Y]$, characterizing the distribution of random graphs, G and targets, Y. We study some simple special cases by assuming edges are conditionally independent, yielding stochastic block models. We develop two classifiers, and prove that they are consistent. Moreover, we prove that performance depends on the model, the size of the graph, and the number of samples. 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.

Index Terms—blah, blah, blah.

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 techniques. Supervised learning algorithms have largely focused on problems that loosely satisfy the following assumptions: data has been exchangeably from some model: $(x_s, y_s) \overset{exch.}{\sim} \mathbb{P}[X, Y]$, where each $x_s \in \mathcal{X} \subseteq \mathbb{R}^p$ is a "feature vector," $y_s \in \mathcal{Y} \subseteq \mathbb{R}^q$ is a (set of) target 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 exemplars.

Here we are interested in a slightly different setting. In particular, rather than assuming that features collectively form a vector, we assume that the features form a graph, where a graph is defined as a set of n vertices and an $n \times n$ element adjacency matrix characterizing the connectivity between vertices. The space of graphs, $\mathcal{G}$ is not in $\mathbb{R}^p$, and therefore, most supervised learning algorithms can not be naïvely applied directly to problems of this form. Importantly, many arising and existing data sets are more naturally represented as graphs than vectors, including telecommunications grids, social networks, the internet, and brains. Thus, tools designed specifically to operate on graph spaces would potentially 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.

A somewhat different approach is considered here, based on the statistical theory of pattern recognition [1]. Specifically, data is assumed to be sampled exchangeably from some joint distribution, $\mathbb{P}[G, Y]$, where G is a *random-graph*, not a *random-vector*. Given this assumption, one can build a classifier that takes as input training data (in the form of graphs and their targets) and a new graph, g, and predicts the most likely y. 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. Therefore, one can describe a number of models, each with distinct assumptions. For each model, a classifier is designed specifically to be consistent for that model, and to converge quickly to Bayes optimal perfor-

- *JTV is blah. CEP is blah. RJV is blah. E-mail: joshuav@jhu.edu*
- *J. Doe and J. Doe are with Anonymous University.*

mance. Simulations confirm that the classifiers behave as they should. These classifiers are then applied to "connectome" data, where each graph corresponds to the macroanatomical structure of a human brain. These classifiers can differentiate gender with better accuracy then other previously proposed approaches, in less time, with more interpretability.

2 THEORY

2.1 Notation and terminology

Both vectors and matrices will be indicated by bold notation, $\mathbf{x}$. The real number line will be $\mathbb{R}$, the dimensionality of $\mathbf{x}$ will be indicated by d, e.g., $x \in \mathcal{X} \subseteq \mathbb{R}^d$, where $\subseteq$ indicates a subset (possibly equal), where script upper case latin letters denote sets, with cardinality indicated by $|\mathcal{X}|$.

2.1.1 Random variables

Throughout this text, we use the following notation and terminology. Upper case latin letters are random objects, $X : \Omega \mapsto \mathcal{X}$ mapping from the universal sample space Ω to the sample space, $\mathcal{X}$. A sample will be indicated by lower case latin letters, e.g., $x \in \mathcal{X}$. The probability distribution of X will be $\mathbb{P}[X]$, and the probability mass function of X taking value x will be written $\mathbb{P}[X = x]$, or just the notationally abusive shorthand, $\mathbb{P}[x]$.

2.1.2 Random graphs

A random graph, G, takes values $g \in \mathcal{G}$, defined by a set of n vertices $\mathcal{V} = \{V_i\} = (V_1, \ldots, V_n)$ and edges (or arcs) between them. The value of each edge, a_{ij}, is encoded in a $n \times n$ element array called an adjacency matrix, $\mathbf{a} = \{a_{ij}\}$. When the set of vertices is fixed, the graph is called a *labeled graph*. When comparing multiple graphs, if they are all labeled graphs, then all of the information about the graph is within the adjacency matrix, so one can simply refer to the random adjacency matrix $\mathbf{A}$. Below, we assume all edges are binary, thus $a_{ij} \in \{0, 1\}$. A hollow graph forbids self-loops, so $a_{ii} = 0 \quad \forall i \in [n]$. Undirected graphs require that $a_{ij} = a_{ji} \quad \forall i, j \in [n]$ (note that $a_{ij} = 1$ indicates the presence of an edge from V_j to V_i). Directed graphs impose no such requirements. The number of possible labeled graphs for a set of vertices is $|\mathcal{G}| = 2^d$, where d is the dimensionality of the graphs (and this is also the number of distinct adjacency matrices). A *simple graph* has a hollow, symmetric, and binary adjacency matrix, so $d = \binom{n}{2}$. Directed graphs with self-loops have $d = n^2$. We denote the probability distribution of a random graph, $\mathbb{P}[G]$. Below, we elaborate on various probability distributions on graphs.

2.1.3 Random targets

Let Y be a random target, taking values $y \in \mathcal{Y}$. We are particularly interested in scenarios in which $\mathcal{Y} = \{0, 1\}$, in which case y is called a class.

2.1.4 Model

Given these definitions, a joint distribution, $\mathbb{P}[G, Y]$, specifies the probability of observing any graph $g \in \mathcal{G}$ (to be defined below) and any target $y \in \mathcal{Y}$. The model is the collection of all possible joint distributions under consideration, $\mathcal{P} = \{\mathbb{P}[G, Y]\}$.

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. T

2.1.5 Parametric models

A model is said to be parametric if the distribution can be characterized entirely by a finite set of parameters, $\theta \in \Theta \subseteq \mathbb{R}^p$, where $p < \infty$ is the dimensionality of the parameter. Strictly speaking, the parameter of the model must be *identifiable*. Formally, $\theta : \Theta \mapsto \mathcal{P}$ is the inverse map from Θ to $\mathcal{P}$ if and only if the latter map is $1-1$, that is $\mathbb{P}_{\theta_1} = \mathbb{P}_{\theta_2} \Rightarrow \theta_1 = \theta_2$ (and $\mathbb{P}_{\theta_i}$ indicates a distribution characterized by θ_i).

2.1.6 Data

Throughout, data is assumed to be sampled exchangeably from some true (but typically unknown) distribution, $x \overset{exch.}{\sim} \mathbb{P}_\theta[X]$. A collection of S data samples is denoted by $\mathcal{D}_S = \{x_s\}$.

2.2 Parameter estimates

A parameter estimate uses some data to obtain an estimate of the true (but typically known) parameter θ^*, $\hat{\theta}_S : \mathcal{X}^S \mapsto \Theta$. An unbiased estimator is one for which its expectation equals the true parameter value: $\mathbb{E}[\hat{\theta}_S] = \theta^*$. An asymptotically unbiased estimator is one for which $\mathbb{E}[\hat{\theta}_S] \to \theta^*$ 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 θ^*. Formally, $\lim_{S \to \infty} \mathbb{P}[\hat{\theta}_S = \theta^*] = 1$.

2.3 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]:

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

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

where $\mathcal{F}$ is the space of all possible classifiers. The misclassification rate, or simply error, 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. In particular, we will assume a corpus of S data points have been sampled exchangeably from the joint distribution, $(g, y), \{(g_s, y_s)\} \overset{exch.}{\sim} \mathbb{P}[G, Y]$, for $s \in [S]$, where $[S] = \{1, 2, \ldots, S\}$ and $\mathcal{D}_S = \{(g_s, y_s)\}$ denotes the set of S samples. Then, we can construct an classifier estimate: $\hat{f}(\cdot; \mathcal{D}_S) : \mathcal{G} \times (\mathcal{G} \times \mathcal{Y})^S \mapsto \mathcal{Y}$. 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:

$$\hat{y} = \operatorname*{argmax}_{y \in \{0,1\}} \hat{\mathbb{P}}[g|y]\hat{\mathbb{P}}[y]. \tag{4}$$

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

$$\hat{L}_{\hat{f}(\cdot;\mathcal{D}_S)} = \sum_{c=1}^{C} P[\hat{f}(g; \mathcal{D}_{S_c}) \neq y] P[\mathcal{D}_{S_c}], \tag{5}$$

noting that (5) generalizes the ideas of "leave-one-out" and related approaches by allowing any sampling strategy, any size subsets, and any number of subsamples. Below we describe several different types of classifiers with different properties.

2.4 Models

Here, we describe a few different independent edge models, each with different constraints on the parameters.

2.4.1 Identical and independent edge model

Perhaps the simplest random graph model one could assume is the Erdös-Rényi (ER) random graph, which asserts that each edge is independent and identically distributed (iid): $\mathbb{P}[A_{ij} = p], \forall i, j \in [n]$. To use this assumption for graph classification, we would assume that

$$\mathbb{P}_\theta[G] = \mathbb{P}_\theta[\mathbf{A}] = \prod_{i,j \in [n]} \mathbb{P}_\theta[A_{ij}]$$

$$= \prod_{i,j \in [n]} \text{Bernoulli}(a_{ij}; p) = \prod_{i,j \in [n]} a_{ij}^p (1 - a_{ij})^{1-p} \tag{6}$$

where $\theta = p$. The distribution of these ER graphs is therefore determined entirely by n and p (and assumptions of whether the graph is directed and/or hollow). This iid model can be easily generalized by relaxing the second 'i', namely, letting edges by independent, but not identically distributed.

Below, we elaborate on two special cases of this generalization. In each case, some edges are Bernoulli(p), and others are Bernoulli(q), where $q > p$; the two models differ in which edges have probability q. In each case, the *signal subgraph* is the graph defined by the edges with probability q. Figure 1 shows examples of all three models.

2.4.2 Incoherent model

Above, the distribution of the entire graph is given by n and p, here we allow for certain edges to have probability q. In particular, we consider the set of m edges, $\mathcal{M}_i = \{A_{ij} | (i, j) \in \mathcal{M}\}$, where $|\mathcal{M}| = m$, each of which has probability q, yielding the following model:

$$\mathbb{P}_\theta[G] = \prod_{(i,j) \notin \mathcal{M}_i} \text{Bernoulli}(a_{ij}; p) \prod_{(i,j) \in \mathcal{M}_i} \text{Bernoulli}(a_{ij}; q). \tag{7}$$

The parameters of this model are therefore: $\theta_{INC} = (p, q, \mathcal{M}_i)$.

2.4.3 Star$_1$ model

In the above model, the edges with probability q could be uniformly scattered across all vertices. Here, we assume that all the edges with probability q share a common end-point. More formally, $\mathcal{M}_* = \{A_{ij} | V_i = V_+ \text{ or } V_j = V_+\}$. This leads to a model identical to (7), except replace the $\mathcal{M}_i$ with $\mathcal{M}_*$. The parameters of the star$_1$ models are therefore: $\theta_{STAR_1}(p, q, \mathcal{M}_*)$.

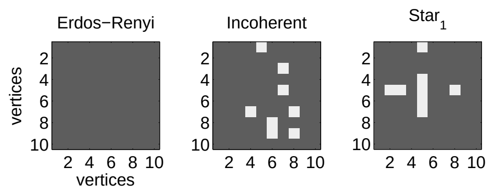

Fig. 1. Schematic depicting the probability of each edge for the three models

2.5 Model estimates

Each of the above models is characterized by a set of parameters, either p, $\{p, q, \mathcal{M}_i\}$, or $\{p, q, \mathcal{M}_*\}$. For a Bernoulli random variable, $a \sim \text{Bernoulli}(a; p)$ the maximum likelihood estimator (MLE) is consistent:

$$\hat{p}_S = \frac{1}{S} \sum_{s=1}^{S} a_s \tag{8}$$

Each edge in each model is Bernoulli, so each edge can be estimated independently using Eq. (8). Therefore, the plugin estimator for any independent edge random graph model could be:

$$\hat{\mathbb{P}}[A=a] = \prod_{i,j\in[n]} a_{ij}^{\hat{p}_{ij}}(1-a_{ij})^{1-\hat{p}_{ij}} \tag{9}$$

where we have dropped the subscript S for brevity. Note however, that if any $\hat{p}_{ij}=0$, then $\mathbb{P}[A=a]=0$. Therefore, we smooth the MLE by using a different contrast function:

$$\hat{p}_M = \frac{\sum_{s=1}^{S} a_s + 1/(2S)}{S+1/(2S)} \tag{10}$$

so that the parameter estimate is never actually zero. The above estimator is in fact an M-estimator, where an M-estimator for θ is any estimator that satisfies:

$$\theta_M = \operatorname*{argmin}_{\theta\in\Theta} \sum_{s=1}^{S} \rho(a_s,\theta) \tag{11}$$

where $\rho(a_s,\theta)$ is called a contrast function. Eq. (10) implicitly defines the following contrast function

$$\rho(a_s,\theta) = -\frac{a_s+1/(2S)}{S+1/(2S)}, \tag{12}$$

which also happens to be a robust estimator, that is, an estimator robust to various model misspecifications (for instance, edge independence is likely to be inaccurate often). Given the above, we have the following theorem:

Theorem 1: If a is Bernoulli distributed with probability p, then $\hat{p}_M$ is a consistent estimator for p, where $\hat{p}_M$ is defined by Eq. (10).

Proof: To prove that an estimator is consistent, it is sufficient to show that it converges to another estimator known to be consistent.

$$\mathbb{E}\left[\frac{\sum_{s\in[S]} a_s + 1/(2S)}{S+1/(2S)}\right] = \frac{\mathbb{E}[\sum_{s\in[S]} a_s] + 1/(2S)}{S+1/(2S)}$$
$$= \frac{\mathbb{E}[\sum_{s\in[S]} a_s]}{S+1/(2S)} + \frac{1/(2S)}{S+1/(2S)} \tag{13}$$

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

Note that for these simple models, better parameter estimates are readily available. For instance, averaging over $\hat{p}_{ij}$ in the ER model would give an improved estimate for p (bias is not introduced, and variance is reduced, so the estimate is better from a bias-variance trade-off perspective). However, we abstain for such averaging so that the theory and simulations generalize to more heterogeneous models (where each a_{ij} might be distributed according to its own p_{ij}).

2.6 Consistent classifiers

For each of the above three models, we desire to have estimators that are consistent. Under certain conditions, consistent estimators can be plugged into (2) to obtain consistent classifiers.[1]

The Naïve Bayes graph classifier, which assumes all edges are independent, is given by:

$$\begin{aligned} f(g) &= \operatorname*{argmax}_y \mathbb{P}[g,y] = \operatorname*{argmax}_y \mathbb{P}[g|y]\mathbb{P}[y] \\ &= \operatorname*{argmax}_y \prod_{i,j\in[n]} \mathbb{P}[a_{ij}|p_{ij}^y]\mathbb{P}[y] \\ &= \operatorname*{argmax}_y \mathbb{P}[y] \prod_{i,j\in[n]} \text{Bernoulli}(a_{ij};p_{ij}^y) \\ &= \operatorname*{argmax}_y \pi^y \prod_{i,j\in[n]} a_{ij}^{p_{ij}^y}(1-a_{ij})^{1-p_{ij}^y}, \end{aligned} \tag{14}$$

where $\pi^Y=\mathbb{P}[Y]$, and p_{ij}^Y is the probability of edge (i,j) existing in class Y.

Upon presuming that a signal subgraph exists, 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, $\mathcal{M}=\{E_{ij}|p_{ij}^0\neq p_{ij}^1\}$, then one can use this information to obtain a better classifier, by only looking at the signal subgraph:

$$f(g) = \operatorname*{argmax}_y \pi^y \prod_{(i,j)\in\mathcal{M}} a_{ij}^{p_{ij}^y}(1-a_{ij})^{1-p_{ij}^y}. \tag{15}$$

This approach does not depend on homogeneity of edges, each edge a_{ij} could be sampled according to its own potentially unique distribution p_{ij}.

Because the parameters will be unknown, one must first estimate them. Given the estimates, they can be plugged in to either (14) or (15). Estimating p and q is quite trivial given $\mathcal{M}$, which could potentially be the complete graph (in the ER case). Specifically, $\hat{p}$ is the mean of $\hat{p}_{ij}$ for $(i,j)\notin\mathcal{M}$, and $\hat{q}$ is the mean of $\hat{q}_{ij}\in\mathcal{M}$. The more difficult task is estimating $\mathcal{M}$, the signal subgraph. Below, we provide some options.

2.6.1 Exhaustive search for signal subgraphs

The number of signal subgraphs is equal to the number of graphs in the random graph family, $p=|\mathcal{G}|=2^d$, where d is around n^2 depending on assumed constraints (see section **??** for details). Thus, one could enumerate all possible signal subgraphs, $\{\mathcal{E}_1,\ldots,\mathcal{E}_p\}$, and compute $\hat{L}_{f_{\mathcal{E}_c}(\cdot;\mathcal{D}_S)}$ for each $c\in[p]$. Finally, let $\hat{c}=\operatorname{argmin}_c \hat{L}_{f_{\mathcal{E}_c}(\cdot;\mathcal{D}_S)}$. Unfortunately, even when n is relatively small (e.g., ≈ 10), p 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.

1. which conditions?

2.6.2 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^0_{ij} and p^1_{ij} for all $i, j \in [n]$, one can compute the distance, $\delta_{ij} = d(p^1_{ij}, p^0_{ij})$, which conveys the difference in position between the two classes. δ_{ij} is thus an uncorrected test-statistic that conveys the relative significance of each edge, so edges can be ranked by ordering these test statistics, $\delta_{(1)}, \ldots, \delta_{(d)}$. Under certain conditions, the best classifier using only m dimensions is the one that uses $\{\delta_{(1)}, \ldots, \delta_{(m)}\}$, where "best" means classifier lowest misclassification rate. [1] shows that when variables are Gaussian and independent, ranking variables by z-score yields the optimal ranking. When the independent edge probabilities are Bernoulli, the estimates of p^y_{ij} are distributed according to a Binomial, $\hat{p}^y_{ij} \sim$ Binomial$(s_y; p^y_{ij})$, which has mean p^y_{ij} and variance $p^y_{ij}(1-p^y_{ij})$. Therefore, plugging the estimate in for the mean and variance, we can estimate δ_{ij} using

$$\hat{\delta}_{ij} = \left| \frac{\hat{p}^1_{ij}}{\hat{p}^1_{ij}(1-\hat{p}^1_{ij})} - \frac{\hat{p}^0_{ij}}{\hat{p}^0_{ij}(1-\hat{p}^0_{ij})} \right|, \qquad (16)$$

which corresponds to the uncorrected p-values, and is the most powerful test statistic [**?**].[2]

If m, the number of edges in the signal subgraph, is known, then the optimal selection of m edges is simply $\delta_{(1)}, \ldots, \delta_{(m)}$, the m edges with the lowest δ's. When m is not known a priori, m must also be estimated. Cross-validation can then be used to choose the optimal m, given the data $\mathcal{D}_S$. More specifically, one obtains a sequence of classifiers, $\hat{f}_1, \ldots, \hat{f}_{m'}$, each one including an additional dimension, and then uses the one with the lowest empirical risk, $\hat{L}_{\hat{f}_{m'}(\cdot;\mathcal{D}_S)}$, that is, let $\hat{m} = \operatorname{argmin}_{m'} \hat{L}_{f_{\hat{\mathcal{E}}_{m'}}(\cdot;\mathcal{D}_S)}$. This approach is hereafter referred to as the *incoherent signal subgraph* search method. Given $\mathcal{E}_{\hat{m}}$, one can apply any of the above classifiers to the selected subgraph. Note that this approach assumes that the class-conditional signal is somewhat *sparse*. Figure 2 depicts δ's for different assumptions on the class conditional differences. The left panel shows an example where class conditional differences are dense, and the middle panel shows an example where these differences are sparse. A third option, depicted on the right, shows the class-conditional difference being both sparse and structured.

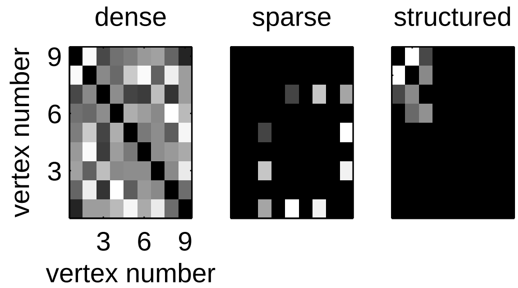

Fig. 2. Various kinds of class-conditional differences suggesting different algorithms to estimate the signal subgraph. The left panel shows a dense signal, meaning that no signal subgraph will contain much information. The middle panel shows a sparse signal subgraph, suggesting using the incoherent signal subgraph search method. The right panel shows a structured sparse signal subgraph, suggesting using the coherent signal subgraph search method.

2.6.3 Star$_1$ 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 the class-conditional differences takes a block structure, yielding a small signal subgraph defined on only a few vertices. In such scenarios, we can use many tools to find the vertices of interest, including community detection algorithms [13] and latent stochastic block model approaches [14]. We propose a different approach here, which is provably asymptotically consistent and quite simple. Specifically, we are searching for vertices that contain much of the class-conditional signal. Each vertex can be described by its associated connection probabilities. For instance, vertex j is characterized by $a_{\cdot j}$ and $a_{j \cdot}$, the j-th row and column, respectively, of the adjacency matrix. Let $p^y_j = (p^y_{\cdot j}, p^{y\,\mathsf{T}}_{j \cdot})$, that is, the concatenation of the j-th row and column (note that p^y_{jj} is implicitly only included once, not twice as would be suggested by the notation). Given p^y_j's, we can compute a distance vector: $\delta_j = d(p^1_j, p^1_0)$. Given the above notions of distance defined for edges, we define difference between vertices as:

$$\delta_j = \sum_i \delta_{ij} + \delta_{ji} - \delta_{jj} \qquad (17)$$

While other definitions of δ_j are certainly possible, this notion worked well in practice in various simulations, and is guaranteed to converge to the correct signal subgraph given a sufficiently large data corpus. This approach to finding a signal subgraph is hereafter referred to as the *coherent signal subgraph* search method.

2. Likelihood-ratio test is UMP for single parameter exponential family distributions, like binomial, but only for simple hypothesis tests. The hypothesis test here is: $H_0 : \left|p^0_{ij} - p^0_{ij} - \right| = 0$ vs. $H_1 : \left|p^0_{ij} - p^0_{ij} - \right| > 0$, which is a composite test. Does this mean that the above measure is not UMP?

3 Simulations

4 Results

4.1 Monotonicity of error given T

4.1.1 $k = 1$

4.1.2 $k > 1$

4.2 Approximate Asymptotica distribution of T

4.2.1 Incoherent search

4.2.2 Star$_1$ search

4.3 Relative Efficiency

4.4 Simulated classification results

4.5 Connectome classification results

ER vs IE Lhat vs. s

sim 1: num of edges sim 2: IE model

4.6 Finding signal subgraphs

algs: ie vs incoherent vs coherent

3 sims: ie, a coherent and incoherent sim

fig 1: example of finding subgraphs fig 2: error vs s, num correct vs s

4.7 Real data

Lhat vs. s

algs: all possible

5 Discussion

5.1 summary

ensemble of approaches to classifying graphs

which algorithm has best Lhat is a function of $\mathbb{P}[G, Y]$, n, and s

comparing performance of algs that are designed for different models provides a way of doing "model selection" with exploitation task in mind

model checking

ind edge subgraph finding is robust

5.2 extensions

LSRGM

Bayesian algorithms

ind edge is M-estimate

ACKNOWLEDGMENTS

The authors would like to thank...

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*. Citeseer, 2004, pp. 729–736.
[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*. Springer, 2008, p. 1007.
[9] K. Riesen and H. Bunke, "Graph classification by means of lipschitz embedding." *IEEE Trans Syst Man Cybern B Cybern*, vol. 39, no. 6, pp. 1472–1483, Dec 2009. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2009.2019264
[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. [Online]. Available: http://www.sciencedirect.com/science/article/B6V10-4X24VM3-2/2/6e8b21d0a93056b57e3aaa573c645c15
[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.
[13] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003. [Online]. Available: http://link.aip.org/link/?SIR/45/167/1
[14] T. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

REFERENCES

[1] H. Kopka and P.W. Daly, *A Guide to LaTeX*, third ed. Harlow, U.K.: Addison-Wesley, 1999.

PLACE PHOTO HERE

Joshua T. Vogelstein Biography text here.

R. Jacob Vogelstein Biography text here.

Carey E. Priebe Biography text here.