PRE-DRAFT: Unlabeled graph classification

cep,djm,jmc,jv2

September 20, 2010

Abstract

We consider classification of unlabeled graphs. We wish to assess the performance degradation due to the application of assignment methods.

This is carey's version – simplified (perhaps too much?); Joshua's connectome (i.e., "brain graph"?) interpretation will have to come from Joshua.

NB: alas, note that "labeled" is used two different ways herein; graphs may be labeled in the sense that the between-graph vertex identification is available, and graphs may be labeled in the sense that their class label is available. i'm sure you're clever enough to negotiate this issue, in the sequel, for the nonce. i solicit constructive approaches to addressing this issue.

NB: i switch willy-nilly twixt graph G and adjacency matrix A. complain if you must. but to what end?

NB: i use "NB" a lot, too. too much, maybe …

1 Introduction

Consider $(G, Y), \{(G_i, Y_i)\}_{i=1}^{s} \overset{iid}{\sim} F_{GY}$, with class labels $Y : \Omega \to \{0, 1\}$ and (labeled) graphs $G : \Omega \to \mathcal{G}_n$, where $\mathcal{G}_n$ denotes the collection of simple (labeled) graphs on $V = [n]$.

NB: We consider simple graphs – unweighted, undirected, with no loops, so the adjacency matrices are binary, symmetric, and hollow; our connectome applications may involve directed, loopy, weighted, attributed, multi graphs … (And I use directed loopy graphs in my example Section 4 … perhaps we should use directed loopy throughout? or undirected unloopy? what say you, John???)

The collection $\mathcal{T} \equiv \{(G_i, Y_i)\}_{i=1}^{s}$ is the training sample and s is the training sample size; G is the graph to be classified, and Y is the true but unobserved class label. For simplicity, we will assume that the prior probability of class membership $\pi \equiv P[Y = 1]$ is known to be $1/2$, and the class-conditional sample sizes $S_y \equiv \sum_{i=1}^{s} I\{Y = y\}$ are fixed (s_y) rather than random variables (S_y) with s even and $s_0 = s_1 = s/2$.

We consider the independent edge model (IE), so that for $y \in \{0, 1\}$ the class-conditional distribution $F_{G|Y=y}$ is parameterized by a (symmetric, hollow) $n \times n$ matrix P_y with entries $p_{y;u,v} \in [0, 1]$.

(NB: We will eventually *generalize* the independent edge model to RDPG … and beyond! But first, herein, we will *simplify* to independent edge block model (IEBM), for mathematical expediency.)

For this IE model, the Bayes optimal classifier for observed graph G (equivalently, for observed (symmetric, hollow) $n \times n$ adjacency matrix $A = A(G) = [a_{u,v}]$) is given by

$$g^*(G) \quad = \quad \arg\max_y \prod_{(u,v) \in \binom{V}{2}} f(a_{u,v}; p_{y;u,v}), \tag{1}$$

where the Bernoulli probability $f(a; p)$ is given by $f(a; p) = p^a(1 - p)^{1-a}$.

Alas, the graph G is not observed; rather, we observe the *unlabeled* version. (right. we don't observe the class label. but here i mean "unlabeled" in the "assignment problem" sense. too.) That is, rather than observing the adjacency matrix A, we observe $\widetilde{A} \equiv QAQ^T$ for some unknown permutation matrix Q.

(NB: sorry John. because of all the Ps, i'm gonna use Q for permutation matrices. deal with it?)

2 An Assignment Problem Application

(This is Assignment Problem Application #1. E.g., voxels = vertices, and voxels have been morphed to anatomical regions as per JV's description. If we assume for #1 that this region assignment is perfect, we have the within-region vertex assignment problem. This is this ... we disussed adding distance $d(u, u')$ and assuming that true assignment is more likely for smaller distances; we discussed relaxing the perfect region assignment to $P[u \in \text{correct region}]$; we discussed other generalizations?)

Consider the observed $\widetilde{A} \equiv QAQ^T$. For $i \in [s]$, let

$$\widehat{Q}_i = \arg\min_{Q'} ||Q'^T \widetilde{A} Q' - A_i||_F. \tag{2}$$

For each pair (u, v), let $\sigma_i(u, v)$ be the reassignment through Q and $\widehat{Q}_i$. That is, entries $a_{u,v}$ in A are out of place due to unlabeledness in $\widetilde{A}$, and the assignment minimization attempts to put them back into place; $\sigma_i(u, v)$ is the result of this attempt – the assignment provided by $\widehat{Q}_i QAQ^T \widehat{Q}_i^T$.

Definition 1. *The* naive assignment classifier *for the observed $\widetilde{A}$ is given by*

$$g(G; \mathcal{T}) = \arg\max_y \max_{i:Y_i=y} \prod_{(u,v)\in\binom{V}{2}} f(a_{\sigma_i(u,v)}; p_{y;u,v}). \tag{3}$$

Note 1: The classifier $g(G; \mathcal{T})$ presented in Definition 1 assumes that the class-conditional edge probabilities $p_{y;u,v}$ are known; however, these probabilities are not used in the assignment equation 2. (They could be? But they're not!)

Note 2: We could employ a plug-in classifier, using estimates in both the classifier and the assignment; e.g., $\widehat{p}_{y;u,v} = (s_y)^{-1} \sum_{i:Y_i=y} a_{i;u,v}$ and $\overline{A}_y = (s_y)^{-1} \sum_{i:Y_i=y} A_i$. However, we would want to use *smoothed* estimates in the classifier (to avoid degeneracy when $\widehat{p}_{y;u,v}$ equals 0 or 1) and *unsmoothed* estimates in the assignment. This complicates the evaluation analysis; we leave this more complicated (and more realistic) investigation for the future.

Note 3: The classifier $g(G; \mathcal{T})$ presented in Definition 1 uses only the single probability-maximizing training assignment for each class. This could be generalized either in the assignment (using $\overline{A}_y$) or by processing the collection $\{\prod_{(u,v)\in\binom{V}{2}} f(a_{\sigma_i(u,v)}; p_{y;u,v})\}_{i:Y_i=y}$ with methods more elaborate than the simple maximum.

Note 4: We could also use nearest neighbor classifier ... but i think this would be less tractable.

The advantage of the classifier $g(G; \mathcal{T})$ presented in Definition 1 is that the assignment methodology and only the assignment methodology is on trial! Better classifiers could be considered, but I'm trying to design a tractable investigation of assignment methodologies ...

Under IE, the difference between $F_{G|Y=1}$ and $F_{G|Y=0}$ is wholly captured by the collection of marginal "signal edges" probabilities

$$\mathcal{E} \equiv \{(u, v) \in \binom{V}{2} : p_{0;u,v} \neq p_{1;u,v}\}.$$

This collection $\mathcal{E}$ might be all of $\binom{V}{2}$. We hereby simplify IE to independent edge block model (IEBM), for mathematical expediency. Let $\mathcal{E} = \binom{V'}{2}$ for a collection V' of $1 \leq m \leq n$ vertices, and define IEBM(n, p, m) to be the model F_{GY} defined by class-conditional probabilities $p_{0;u,v} = p \neq 1/2$ and $p_{1;u,v} = 1 - p$ for all $(u, v) \in \mathcal{E}$ and $p_{0;u,v} = p_{1;u,v} = 1/2$ for all $(u, v) \in \binom{V}{2} \setminus \mathcal{E}$. (In this case, all signal edges are created equally and all noise edges are created equally.) Notice that IEBM(n, p, m) requires that $\mathcal{E}$ is a block – the signal edges consist of precisely the potential interconnections between a set of m vertices.

Let $s = 2$ (one single training observation from each class). In this case, since all signal edges are created equally and all noise edges are created equally, the performance of the classifier $g(G; \mathcal{T})$ is monotonic in the number of signal edges recovered by σ_1 and σ_2.

Let the random variable $L(g) \equiv P[g(G; \mathcal{T}) \neq Y|\mathcal{T}]$ be the probability of misclassification for classifier g conditioned on the training sample [see DGL 1996]. Under IEBM(n, p, m) with $s = 2$, we have that $L(g)$ depends on only n, p, m. Define $L^* \equiv L(g^*)$.

Theorem 1. *For $F_{GY} \in$ IEBM(n, p, m), $L(g|T = t) < L(g|T = t - 1)$ for all $t \in [2m - 1]$, where*

$$T_i \equiv |\{(u, v) \in \mathcal{E} : \sigma_i(u, v) \in \mathcal{E}\}| \tag{4}$$

and

$$T \equiv T_1 + T_2.$$

Proof: (Proof of this (alleged) monotonicity requires but a simple modification to Henry's proof?)

So, for this simple case, we need concern ourselves with only the performance of the assignment algorithm in terms of T_i.

Theorem 2. *$T_1 =^{\mathcal{L}} T_2$ for this simple case.*

Proof: (By construction? it's suppose to … that's why i set up the class-conditional edge probabilities $p_{y;u,v}$ to be reflective about $1/2$ in y.)

3 Performance Degradation

What is the performance degradation due to unlabeled-ness?

Case I:

Theorem 3. *$P[\widehat{Q}_i = Q] = 1$ for all i implies $L(g) = L^*$.*

Proof: If $P[\widehat{Q}_i = Q] = 1$ for all i (that is, if the assignment algorithm gets the *right* answer – not to be confused with the *best* answer in terms of the optimization) then $T_1 = T_2 = |\mathcal{E}|$ and hence $L(g) = L^*$.

Case II:

How about when the assignment algorithm gets the *best* answer in terms of the optimization? Perhaps we can work this out – a theorem/proof?

Theorem 4. *$L_{n,p,\mathcal{E}}(g) =$ some *identifiable* function of $n, p, \mathcal{E}$?*

According to my calculations, the only tricky bit is $I\{T_i(g_1, g_2) = t\}$ given two graphs $g_1, g_2 \in \mathcal{G}_n$. That is, given two graphs (no randomness), T_i either equals t or it doesn't (after (arbitrarily?) accounting for non-uniqueness of assignment solution $\widehat{Q}$). This looks like i could do it for $n = 3$. But then the combinatorics get silly. BUT: maybe this is one of those things for which generating function folks could derive a *generating function* …? at least for my simple stochastic block model class-conditional distributions?

After Case I (the assignment algorithm gets the *right* answer – not to be confused with the *best* answer in terms of the optimization) and Case II (the assignment algorithm gets the *best* answer in terms of the optimization), we will investigate approximation assignment algorithms based on the trade-off between (1) computational complexity and (2) classification performance (either $L(g)$ directly, or in terms of the distribution of T_i).

The Monte Carlo example presented below (Section 4) demonstrates significant but not complete performance degradation due to a particular algorithm (**lp.assign** in **R** package **lpSolve**).

4 Example

A Monte Carlo experiment (20000 paired replications) demonstrates that, (using **directed loopy** graphs for simplicity) with $n = 10, p = 0.25$, and $|\mathcal{E}| = 9$ (where $\mathcal{E}$ is in fact a 3×3 block) the performance degradation due to the application of **lp.assign** in **R** package **lpSolve** (with $Q = I$ specifying starting point) is from $\widehat{L} = 0.0476 \approx L^* = 1 - F_{Binomial(9,0.25)}(4) = 0.04892731\ldots$ to $\widehat{L} = 0.28855$. So better-than-chance classification is achieved for our unlabeled scenario using this assignment algorithm, but performance is significantly degraded.

NB: should also report performance in terms of T_i. and objective value at solution?

Code for this example is provided in the Appendix.

NB: LAP is not QAP; i'd be happy to have QAP R code …

5 Proposal

I propose that we investigate, via theory, simulation, and experiment, the trade-off between computational complexity and performance, and also identify the relationship between the explicit assignment objective function and the exploitation task (classification) objective function.

Except for Cases I (the assignment algorithm gets the *right* answer – not to be confused with the *best* answer in terms of the optimization) and II (the assignment algorithm gets the *best* answer in terms of the optimization), I'm not sure what we'll be able to prove. Perhaps

Theorem 5. *LAP is as good as QAP $\iff$ model $\in$ IEBM?*

But we can do simulation analysis: first, for my simple scenaro; then, generalizing to (perhaps) approach experimental settings?

NB: perhaps we should be doing hypothesis testing instead???

6 20 statements

```
allow me to try to state some things that it seems we agree on, and some things that seem t
with the hope of clarifying at least where we are, and where we'd like to be.  if anything

(1) The "quadratic assignment problem" (QAP) is the following:
give two matrices, A and B,
find a Q and P such that
min ||QAP' - B||
where both Q and P are permutation matrices.


JMC: Actually, this is the bi-linear assignment problem for the QAP P=Q.
The bilinear problem has the pleasant property that the relaxed version
(replacing the permutation constraint with the doubly stochastic constraint)
has its solutions on the vertices.  That is when you solve the
bilinear assignment
your are guaranteed to find a vertex.  The solution may be a local optimum and
multiple starts may still be necessary.

JoVo: modified first two points
(1) The "bilinear assignment problem" (BAP) is the following:
give two matrices, A and B,
find a Q and P such that
min ||QAP' - B||
where both Q and P are permutation matrices.

(2) The Quadratic Assignment Problem (QAP) is the following:

given two adjacency matrices, A and B,
find a P such that
min || PAP' - B||
where P is a permutation matrix.
thus, QAP is a constrained BAP.

(3) the graph isomorphism problem is a QAP


(2) QAP is NP-hard

(3) The Graph Isomorphism Problem (GIP) is the following:
given two adjacency matrices, A and B,
find a P such that
min || PAP' - B||
where P is a permutation matrix.
thus, GIP is a constrained QAP.
```

(4) GIP has weird complexity, somewhere between NP-complete and P

(4) The Linear Assignment Problem (LAP) is the following:
given two adjacency matrices, A and B,
find a P such that
min || PA - B||
where P is a permutation matrix.

(5) LAP can be solved in O(n^3) by the hungarian algorithm

(6) The Frank-Wolfe (FW) algorithm is an algorithm designed to *solve* quadratic problems w
min f(x) = 0.5* x' E x + h' x
where x is in some feasible region defined by a set of linear constraints, that is Ax \leq

(7) FW is an iterative algorithm that works as follows

(i) initialize x with x_0
(ii) compute the first order taylor expansion around f(x_k)
(iii) compute the gradient of f(x_k), call that g(x_k)
(iv) solve the following subproblem:
min f(x_k) + g(x_k) xbar_k, where xbar_k is in the feasible region
(v) compute the step size, lambda that solves
min f(x_k + lambda (xbar_l - x_k)) subject to lambda is in (0,1)
(vi) let x_{k+1} = x_k + lambda*(xbar_k - x_k), and let k=k+1

we stop if ever g(x_k)=0 or lambda=0.

(8) FW can be quite slow. although the first few iterations are often fast.

(9) doubly stochastic matrices have the following properties:

(i) each row sums to 1
(ii) each column sums to 1
(iii) each element is a non-negative real number

these can each be written as linear constraints (linear equalities, in fact).

(10) the set of doubly stochastic matrices is a convex polytope, and is the convex hull of

(11) permutation matrices are square matrices with exactly one 1 in each row and one 1 in e

(12) thus, one can solve the following relaxation of GIP, called the Doubly Stochastic Appr
Yhat = min || YAY' - B||
where Y is the set of doubly stochastic matrices.
in particular, FW can solve this problem in linear time.

(13) it is possible that the solution to DSA is in fact also a solution to GIP, because the

(14) if Yhat is not a permutation matrix, then one can find the closest permutation matrix
min || Yhat - X ||
which can be solved in polynomial time using the hungarian algorithm.

(15) we have found a case for which we can show in simulo:

```
L* ~ E[L_{labeled}] << E[L_{LAP}] << 1/2

(16) question 1: where does E[L_{QAP}] fit in for this simulation.  this is a question we c

(17) question 2: where does E[L_{DSA}] fit in, where L_{DSA} is the misclassification rate

(18) question 3: since DSA takes a long time, let DSA_k indicate the solution to DSA from t

(19) question 4: is it the case that the solution to DSA_1always equals the solution to LAP

(20) question 5: how does conroy's algorithm fit into all of this? it seems that FW will ev

i hope this missive at least clarifies what i believe and don't understand.  i also hope it

shabbat shalom to all,
jovo
```

7 email from jmc

```
John Conroy <conroyjohnm@gmail.com> Sat, Sep 18, 2010 at 11:15 PM
To: Carey Priebe <cep@jhu.edu>, joshuav <joshuav@jhu.edu>
Hi Carey and Joshua,
 Here's a rough draft description of the FW method for QAP.  I adapted from the
original late 90's unpublished paper by Lou, Steve, and I.  I suspect that
once this is merged into the paper much can be cut out and some may be appropriate
for an appendix.   I think it would be good to include the performance of the FW method
on the QAP test set to justify the method.

Here's a couple of points relative to our discussion on Thursday:

 1.  Indeed, the subproblem
FW solves is a linearization of the quadratic function, a Taylor series about $X^{(k)}$.
This sub-problem's solution then produces a vertex which give a search direction for the "e
2.  Furthermore, I am convinced that the
relaxed QAP is NOT in general a convex optimization problem.  The Hessian is
kron(A,B)+kron(A',B') and in general this will not be a positive definite matrix.
(A sufficient condition is for A and B to be symmetric positive definite or for them
both to be symmetric negative definite.)

I sent a copy of this draft to Lou and Steve as well with some background where we intend t
go next.

Best regards,
Johnny
```

Appendix

Code producing the example results presented in Section 4.

```
library(lpSolve)
cuc = function(thisclass,myseed=1)
# Classification of Unlabeled Connectomes
{
set.seed(myseed)
n=10
nmc=10000
# directed, with loops
P1 = P2 = matrix(0.5,n,n) # class-conditional distribution specification
P1[1:3,1:3] = 0.25
P2[1:3,1:3] = 0.75
label = labeltilde = tie = tietilde = rep(0,nmc)
Qhat1lpobjval = Qhat2lpobjval = NULL
for(mc in 1:nmc)
{
G1 = matrix(rbinom(n^2,1,P1),n,n)
G2 = matrix(rbinom(n^2,1,P2),n,n)
if(thisclass == 1 )
 G  = matrix(rbinom(n^2,1,P1),n,n)
if(thisclass == 2 )
 G  = matrix(rbinom(n^2,1,P2),n,n)
Q = matrix(0,n,n)
diag(Q) = 1 # Q == I
Gtilde = Q %*% G %*% t(Q)
C1 = C2 = matrix(0,n,n) # cost
for(i in 1:n) for(j in 1:n)
 {
 C1[i,j] = sum(abs(Gtilde[i,]-G1[j,]))
 C2[i,j] = sum(abs(Gtilde[i,]-G2[j,]))
 }
Qhat1lp = lp.assign(C1)
Qhat2lp = lp.assign(C2)
Qhat1 = Qhat1lp$solution
Qhat2 = Qhat2lp$solution
Qhat1lpobjval[mc] = Qhat1lp$objval
Qhat2lpobjval[mc] = Qhat2lp$objval
sigma1 = t(Qhat1) %*% Gtilde
sigma2 = t(Qhat2) %*% Gtilde
# now ... classify G and Gtilde
p1 = prod( (P1^G) * ((1-P1)^(1-G)) )
p2 = prod( (P2^G) * ((1-P2)^(1-G)) )
p1tilde = prod( (P1^sigma1) * ((1-P1)^(1-sigma1)) )
p2tilde = prod( (P2^sigma2) * ((1-P2)^(1-sigma2)) )
if(p1>p2) label[mc]=1
if(p1==p2) tie[mc]=1
if(p1tilde>p2tilde) labeltilde[mc]=1
if(p1tilde==p2tilde) tietilde[mc]=1
}
return(list(label,tie,labeltilde,tietilde))
}

cuc1 = cuc(1)
cuc2 = cuc(2)
```

```
sum(cuc1[[1]])
sum(cuc1[[2]])
sum(cuc1[[3]])
sum(cuc1[[4]])
# [1] 9524
# [1] 0
# [1] 6986
# [1] 121

sum(cuc2[[1]])
sum(cuc2[[2]])
sum(cuc2[[3]])
sum(cuc2[[4]])
# [1] 476
# [1] 0
# [1] 2759
# [1] 117

(10000 - sum(cuc1[[3]]) - .5*sum(cuc1[[4]]) + sum(cuc2[[3]]) + .5*sum(cuc2[[4]]))/20000
# [1] 0.28855

# L*:
# > 1-pbinom(4,9,.25)
# [1] 0.04892731
```

7.1 move this ...

Under IE, the difference between $F_{G|Y=1}$ and $F_{G|Y=0}$ is wholly captured by the collection of marginal "signal edges" probabilities

$$\mathcal{E} \equiv \{(u,v) \in \binom{V}{2} : p_{0;u,v} \neq p_{1;u,v}\}.$$

$$\begin{aligned} g^*(G) &= \arg\max_y \prod_{(u,v)\in\binom{V}{2}} f(a_{u,v}; p_{y;u,v}) && (5) \\ &= \arg\max_y \prod_{(u,v)\in\mathcal{E}} f(a_{u,v}; p_{y;u,v}), && (6) \end{aligned}$$

If we estimate $p_{y;u,v}$ from the training data, we may consider classifiers

$$g_{NB}(G;\mathcal{T}) = \arg\max_y \prod_{(u,v)\in\binom{V}{2}} f(a_{u,v}; \widehat{p}_{y;u,v}) \qquad (7)$$

and

$$g_{\mathcal{E}}(G;\mathcal{T}) = \arg\max_y \prod_{(u,v)\in\mathcal{E}} f(a_{u,v}; \widehat{p}_{y;u,v}). \qquad (8)$$

NB: requires *smoothed* estimates $\widehat{p}_{y;u,v}$, to avoid degeneracy when $\widehat{p}_{y;u,v}$ equals 0 or 1.

The latter classifier, $g_{\mathcal{E}}(G;\mathcal{T})$, is the best we can hope for – it considers the signal edges and only the signal edges; the former can be swamped by noise from non-signal edges.

If the estimates $\widehat{p}_{y;u,v}$ are consistent (converge to $\widehat{p}_{y;u,v}$ as $s \to \infty$), then both of these classifiers are consistent (converge to Bayes optimal); that is, $L(g) \to L(g^*) \equiv L^*$, where the random variable $L(g) \equiv P[g(G) \neq Y|\{(G_i,Y_i)\}_{i=1}^s]$ is the probability of misclassification for classifier g conditioned on the training sample [see DGL 1996]. Note that $g_{\mathcal{E}}(G;\mathcal{T})$ should dominate $g_{NB}(G;\mathcal{T})$.

8 Simulations

```
n = 10; p = 0.5; m = 5; q0 = 0.2; q1 = 0.8; s(test) = 100; s(train) = 2;
```

For each of the 100 test samples, we first tried to solve GIP with respect to each of the two training samples. Then, given the solution, we tried to classify. Note that because the number of training samples is 2, the parameter estimates are degenerate, we therefore add or subtract epsilon to ensure they are not 0 or 1. The MAP estimates would likely perform better. I will have that code working soon.

9 Results

10 Some misc thoughts

Classifying labeled graphs has already been solved (VP10a, VLP10b).

Two broadly different approaches to classifying unlabeled graphs: i) first (approximately) solve graph isomorphism and then use standard graph classification problems, and ii) classify using graph invariants.

The first approach can be universally consistent, interpretable, etc. Yet, it might take too long. The second approach might have good performance, but lacks consistency results.

Given two unlabeled graphs with n vertices, A, and B, the graph isomorphism problem can be written as:

$$\hat{P} = \operatorname*{argmin}_{P\in\pi} \left\|PAP^{\mathsf{T}} - B\right\| \qquad (9)$$

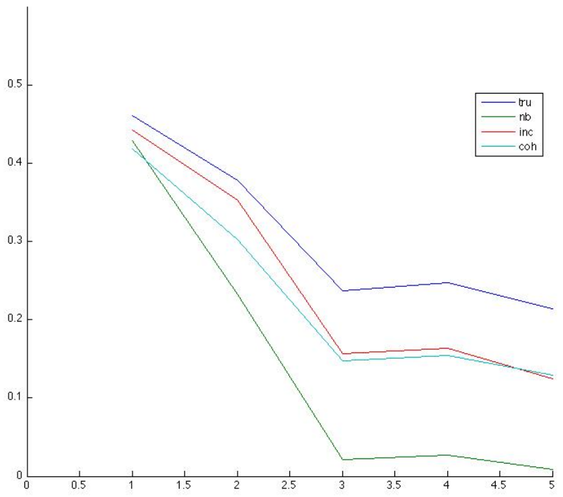

Figure 1: caption

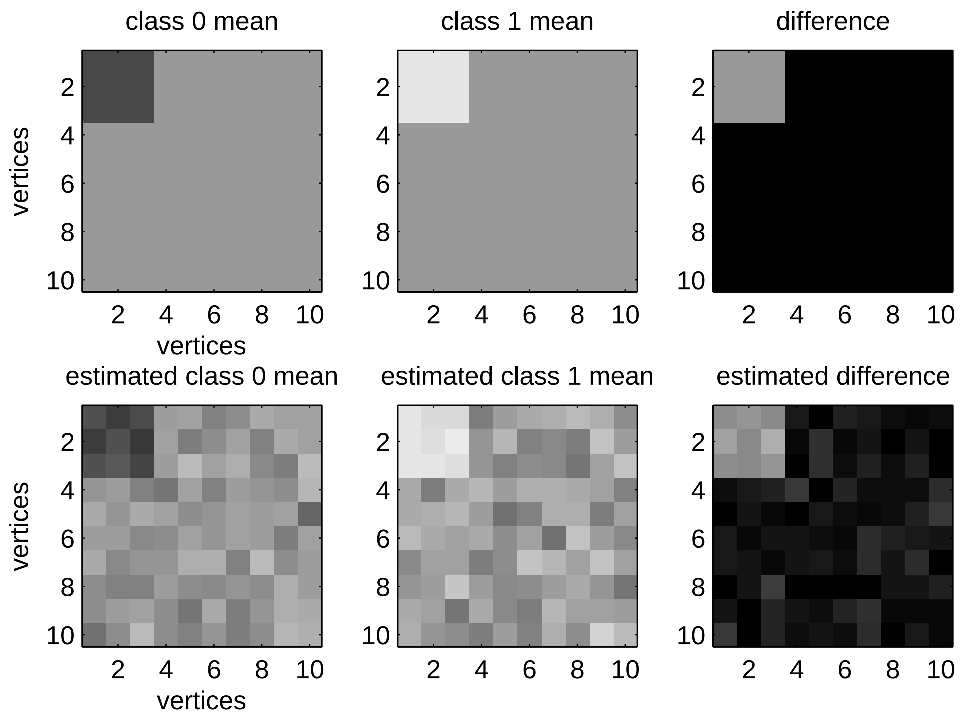

Figure 2: Top row: true parameters for simulation 1. Bottom row: estimates using all training data (assuming vertices are labeled)

where π is the set of permutation matrices ($0-1$ matrices with a single 1 in each row and column). Some people know (ref?) that this problem is nearly NP, and the fastest known algorithms to solve this exactly scale exponentially

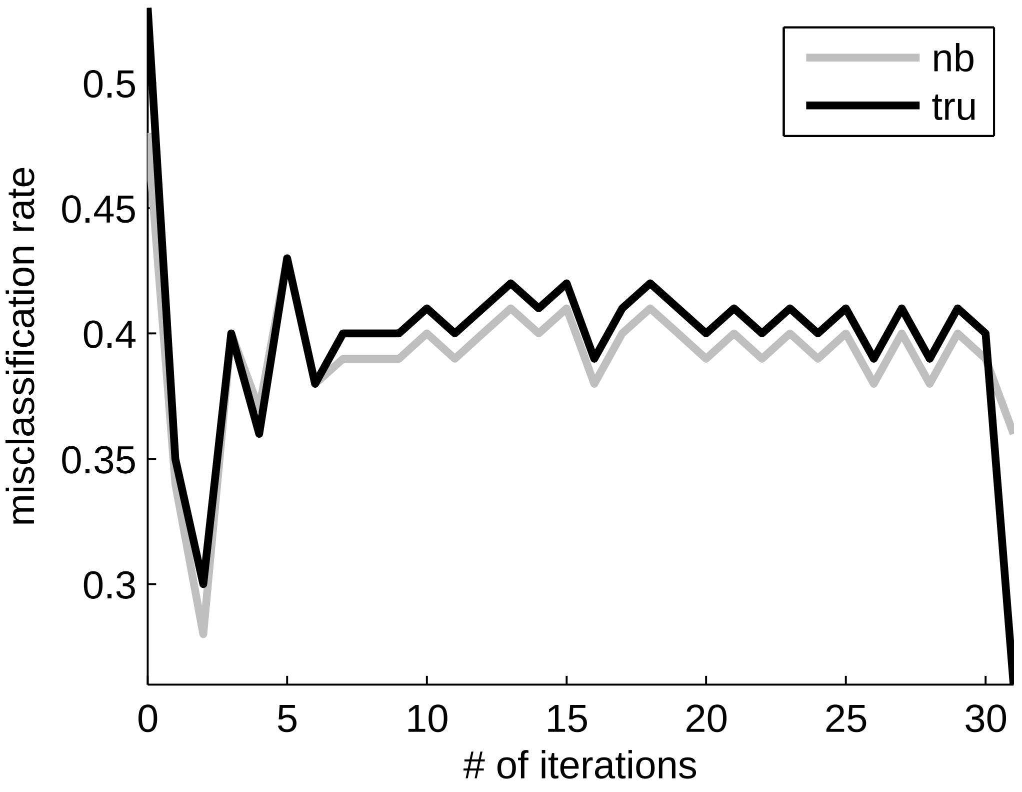

Figure 3: Misclassification rate as a function of max iteration. Note that iteration 0 means did not try to solve GIP. Iteration 31 is assumes that GIP was solved exactly (that is, the algorithm used vertex labels). Lhat between those should be between the bounds, and it is. "Tru" means only classify using the signal subgraph, "nb" means classify using the entire graph. I do not fully understand why "tru" behaves as it does. Note that running this code multiple times gives quantitatively quite different results, but the bounds are always basically respected, suggesting it is bug free.

with n. One could instead solve:

$$\hat{X} = \operatorname*{argmin}_{X \in \mathcal{D}} \left\| X A X^{\mathsf{T}} - B \right\| \tag{10}$$

where $\mathcal{D}$ is the set of doubly stochastic matrices (matrices with non-negative elements whose rows and columns sum to one), which is a superset of permutation matrices. In fact, the permutation matrices are the extreme points of the set of doubly stochastic matrices. Thus, sometimes solving (10) gives you the solution to (9). So, one way to approximately solve (9) is to first solve (10), and then project $\hat{X}$ onto the set of permutation matrices:

$$\hat{P} = \operatorname*{argmin}_{P \in \pi} \left\| P - \hat{X} \right\| \tag{11}$$

Both (10) and (11) can be solved in $\mathcal{O}(n^3)$ using the Hungarian algorithm. However, sometimes $\hat{X}$ is far from any extreme point in π, thus the projection might end up somewhere bad.

According to JCM, (10) is not log-concave, so the initial starting point matters significantly. Thus, multiple restarts, each time starting with a doubly stochastic matrix, can improve results.

JCM fills in here how his algorithm works, as I don't yet understand it.

For classification, we wonder: for what models do multiple restarts help?

In the kidney-egg classification problem, $n - m$ of the vertices can be arbitrarily permuted without loss of classification accuracy (as they are exchangeable). However, in the more general independent edge model, no vertices may be exchangeable with one another, and in such cases, multiple restarts may help significantly.