Assignment 4

Due: December 13th at 8pm EST

Instructions: This is an individual assignment, not group work. Though you may discuss the problems with your classmates, you must solve the problems and write the solutions independently. As stated in the syllabus, copying code from a classmate or the internet (even with minor changes) constitutes plagiarism. You are required to submit your answers in pdf form (use LaTeX) in a file called `<your-UNI>-hw4.pdf` to courseworks. The code for the programming assignment should be appended at the end of this pdf. Late submissions will be penalized, except in extenuating circumstances such as medical or family emergency. Submissions submitted 0-24 hours late will be penalized 10%, 24-48 hours late by 20%, 48-72 hours late by 30%, and later than 72 hours by 100% (i.e., zero credit). Questions 1 and 2 are worth 15 points each, for a total of 30 points.

Problem 1

On Courseworks you'll find a file ("hw4data_missing.txt") with some simulated data, generated from a Gaussian graphical model with 20 variables and including missing values. This is basically very similar to the data you analyzed with glasso and PC on your last assignment, but with systematically missing observations. The missingness mechanism is unknown to you. Again, use PC and glasso to analyze this data. Fix a choice for the values of the tuning parameters based on your work from last time. First, analyze the "complete cases," i.e., only rows that have no missing values. Report the results. Then, try imputing the missing values using a multiple imputation procedure, e.g., MICE with the R package `mice`. Generate several imputed data sets (e.g., 10), learn the graphs using PC and glasso (for a fixed value of the tuning parameters), and then "average" the estimated graphs together by reporting a final estimate which includes a specific edge if it appears in the majority of your estimates. Also, considering other ways of averaging or combining the imputed data sets in your analysis. Report your results. How are these results different from the ones where you had no missing data? There are multiple was of doing this reasonably: be sure to explain what you did exactly.

Problem 2

This problem ecourages you to explore approximate inference methods (including MCMC and some variational inference) implemented in a popular probabilistic programming platform called Stan: `www.mc-stan.org`. Stan is written in C++ mostly, but interfaces with number of languages/platforms including R, Python, Matlab, etc. Here, let's use the interfance with R called `rstan`: `https://mc-stan.org/users/interfaces/rstan`.

Consider some latent variable Z with three states or configurations, distributed according to some unknown model with location parameters $\mu \equiv \mu_1, \mu_2, \mu_3$ and variance parameters $\sigma^2 \equiv \sigma_1^2, \sigma_2^2, \sigma_2^3$. You have available only noisy data on a few sensors, which measure the hidden states according to something that looks like a kind of Gaussian mixture model: $Y_i \sim N(y_i; \theta, \mu_{z_i}, \sigma^2_{z_i})$. Here θ is a vector of parameters that quantifies how sensitive each sensor is to the different hidden configurations. Along with this homework you'll find a data set with 1000 observations of 4 variables ("hw4data_stan.txt"), which were generated by a process unknown to you, but roughly of this form. If you examine the data with some quick histograms, you'll see that each variable looks like some mixture of 2 or 3 hidden states. Your assignment is to learn how to use `rstan` and analyze this data using approximate inference methods. I want to know: 1) where are the hidden states located? 2) with what variances? and 3) which "sensors" are mixing which hidden states? To answer this question I'd like to get some point estimates of the various parameters involved (e.g., give me some numbers for $\hat{\mu}_1$, etc.), as well as some representation of your uncertainty about the answers to these questions, in the form of confidence/credal intervals or posterior distributions. These are all relatively straightfoward

things to extract from Stan, once you get a sampling procedure which seems to behave well. What you give me exactly is up to you: make some tables or some plots, however you want. You will have to spend some time with the Stan documentation or instruction manuals on the above website. This example may also be helpful (for binary instead of continuous observations): `https://mc-stan.org/users/documentation/case-studies/Latent_class_case_study.html`

The file "stanhw.R" has a small R script which includes a Stan model specification, also reproduced below. You can execute the commands in the script to see how you might, for example, run Stan's MCMC method with some basic settings. However, the model as I've specified it may not perform very well. The estimates you get out of running the model as-is may look bad and unreliable, and it won't look like the sampling has converged to a nice stationary point. If you look at the output values for `Rhat` (should be near 1.0) and `n_eff` (should be higher, $>$100) indicate that something is wrong, and if you take a look at the traceplots they will be troubling. So, your task is to refine the model and try various options to get a better result. Try changing the priors and the settings of the sampling method. Try mean-field variational inference (and also Stan's "full rank" VI method) using the `vb` function. Explore the data, and write up an analysis with an explanation of some of the things you've tried and what you've learned. (You don't have to report *every single* configuration you tried or command you wrote! Just the highlights.) Note: it is not entirely easy to get good results with this data, so don't dispair if nothing you try seems to work perfectly well. It is more important that you try and learn than find the "correct" values.

The model:

```
// Stan model

data {
int N; // sample size
int D; // dimension of observed vars
int K; // number of latent groups
vector[D] y[N]; // data
}

parameters {
ordered[K] mu; // locations of hidden states
vector<lower = 0>[K] sigma; // variances of hidden states
simplex[K] theta[D]; // mixture components
}

model {
vector[K] obs[D];

// priors
for(k in 1:K){
mu[k] ~ normal(0,10);
sigma[k] ~ inv_gamma(1,1);
}

for(d in 1:D){
theta[d] ~ dirichlet(rep_vector(2.0, K));
}


// likelihood
for(d in 1:D){
for(i in 1:N) {
for(k in 1:K) {
obs[d][k] = log(theta[d][k]) + normal_lpdf(y[i][d] | mu[k], sigma[k]);
```

```
}
target += log_sum_exp(obs[d]);
}
}
}
```