CPSC 440 Assignment 4 (due Friday April 9 at midnight)

1. Name(s):
2. Student ID(s):

1 Undirected Graphical Models

1.1 Inference in UGMs

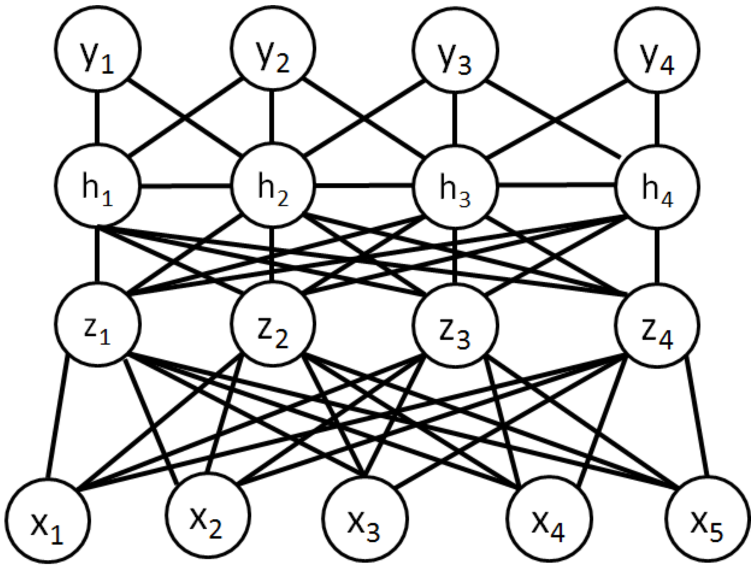
Consider a set of 10 students taking an exam. 6 of the students studied hard so will get 90% of the questions right, while 4 students did not study at all so will get 25% of the questions right. However, when it comes time to take the exam the students are asked to sit in a circle, where they can clearly see the answers of the students next to them. The code *example_UGM.jl* models this scenario as a UGM, assuming that the potential of neighbouring students to get the same answer is twice the potential of neighbouring students to get different answers. Look through this code, as well the "unnormalizedProb" and "computeZ" functions in *UGM.jl*, and answer the following questions:

1. Which node numbers correspond to the students that did not study? And which state corresponds to getting the question right?
2. What is the optimal decoding? Hand in your function to compute the optimal decoding.
3. What is the probability that each student gets the question right, $p(x_j = 1)$ for all j? Hand in your function to compute the marginals.
4. How does the decoding change if you condition on student 5 gets the question wrong?
5. How does the decoding change if you condition on student 8 getting the question right?
6. Hand in code implementing the ICM method for this problem. What solution does ICM converge to if you initialize the algorithm "all wrong"? What solution does ICM converge to if you initialize the algorithm to "all right"? Assume that we chose the variable to update by cycling through the variables in order: 1,2,3,4,5,6,7,8,9,10,1,2,3,. . . .
7. Suppose you want to make the dependency between students stronger, so you multiply every $\phi_e(s, s')$ by 10 for all edges e and states s and s' ("*phi2 = 10*phi2*" in the code). How would this change the answers to all of the above questions, and why? (Feel free to try it out, but think about why you get the answers that you do.)

Hint: feel free to use a "brute force" approach in the code, as in the "computeZ" function. (You don't need to use dynamic programming or message passing, and for the conditioning questions you don't need to form the conditional UGM.)

1.2 Conditional UGMs

Consider modeling the dependencies between sets of binary variables x_j and y_j with the following UGM which is a variation on a stacked RBM:

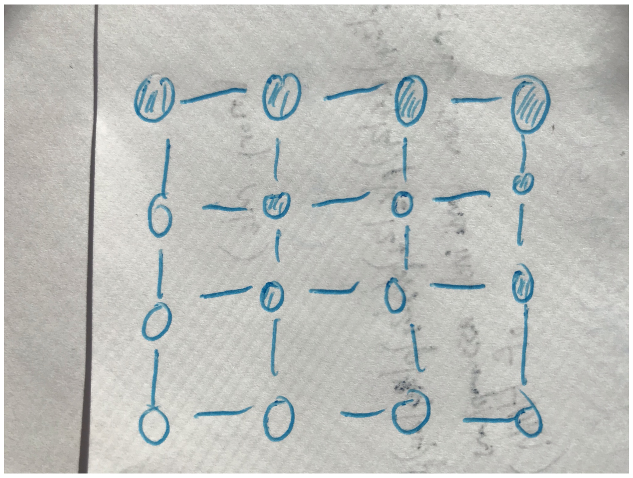


Computing univariate marginals in this model will be NP-hard in general, but the graph structure allows efficient block updates by conditioning on suitable subsets of the variables (this could be useful for designing approximate inference methods). For each of the conditioning scenarios below, draw the conditional UGM and informally comment on how expensive it would be to compute univariate marginals (for all variables) in the conditional UGM.

1. Conditioning on all the x and h values.
2. Conditioning on all the z and y values.
3. Conditioning on all the x and z values.

1.3 Lattice-Structured Conditional UGM

Suppose we need to do approximate inference in a 4-by-4 lattice structured UGM. Rather than updating one node at a time, we could condition on the following 8 nodes and cheaply update the remaining nodes (since they form a tree structure):



But there are "tree-structured" updates that require conditioning on fewer nodes. For example, we would still have a tree-structured graph if we did not condition on the node in the upper-left (so we would condition on 7 nodes and still have a tree-structured conditional UGM). Draw a conditioning scenario that would minimize the number of nodes you condition on, subject to the conditional UGM being a tree (or forest).

2 Bayesian Inference

2.1 Conjugate Priors

Consider counts $y \in \{0, 1, 2, 3, \dots\}$ following a Poisson distribution with rate parameter $\lambda > 0$,

$$p(y \mid \lambda) = \frac{\lambda^y \exp(-\lambda)}{y!}.$$

We'll assume that λ follows a Gamma distribution (the conjugate prior to the Poisson) with shape parameter $\alpha > 0$ and rate parameter $\beta > 0$,

$$\lambda \sim \mathrm{Gamma}(\alpha, \beta),$$

or equivalently that

$$p(\lambda \mid \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda),$$

where Γ is the gamma function.

Compute the following quantites:

1. The posterior distribution,

$$p(\lambda \mid y, \alpha, \beta).$$

2. The marginal likelihood of y given the hyper-parameters α and β,

$$p(y \mid \alpha, \beta) = \int p(y, \lambda \mid \alpha, \beta) d\lambda.$$

3. The posterior mean estimate for λ,

$$\mathbb{E}_{\lambda \mid y, \alpha, \beta}[\lambda] = \int \lambda p(\lambda \mid y, \alpha, \beta) d\lambda.$$

4. The posterior predictive distribution for a new independent observation $\tilde{y}$ given y,

$$p(\tilde{y} \mid y, \alpha, \beta) = \int p(\tilde{y}, \lambda \mid y, \alpha, \beta) d\lambda.$$

Hint: You should be able to use the form of the gamma distribution to solve all the integrals that show up in this question. You can use $Z(\alpha, \beta) = \frac{\Gamma(\alpha)}{\beta^\alpha}$ to represent the normalizing constant of the gamma distribution, and use α^+ and β^+ as the updated parameters of the gamma distribution in the posterior.

2.2 Empirical Bayes

Consider the model

$$y^i \sim \mathcal{N}(w^T z^i, \sigma^2), \quad w_j \sim \mathcal{N}(0, \lambda^{-1}),$$

where z^i is a length-k non-linear transformation of the features x^i (like a polynomial basis or RBFs). The posterior distribution in this model as the form

$$y^i \sim \mathcal{N}(w^+, \Theta^{-1}),$$

where the posterior precision and mean are given by

$$\begin{aligned}\Theta &= \frac{1}{\sigma^2} Z^T Z + \lambda I,\\ w^+ &= \frac{1}{\sigma^2}\Theta^{-1} Z^T y,\end{aligned}$$

and Z contains the z^i vectors in the rows. The marginal likelihood in this model is given by

$$p(y \mid X, \sigma^2, \lambda) = \frac{(\lambda)^{k/2}}{(\sigma\sqrt{2\pi})^n |\Theta|^{1/2}} \exp\left(-\frac{1}{2\sigma^2}\|Zw^+ - y\|^2 - \frac{\lambda}{2}\|w^+\|^2\right).$$

As discussed in class, the marginal likelihood can be used to optimize hyper-parameters like σ, λ, and even the basis Z.

The demo *example_basis* loads a dataset and fits a degree-2 polynomial to it. Normally we would use a test set to choose the degree of the polynomial but here we'll use the marginal likelihood of the training set. Write a function, *leastSquaresEmpiricalBaysis*, that uses the marginal likelihood to choose the degree of the polynomial as well as the parameters λ and σ (you can restrict your search for λ and σ to powers of 2). Hand in your code and report the marginally most likely values of the degree, σ, and λ.

2.3 Markov Chain Monte Carlo

If you run *example_MH.jl*, it loads a set of images of '2' and '3' digits. It then runs the Metropolis MCMC algorithm to try to generate samples from the posterior over w, in a logistic regression model with a Gaussian prior. Once the samples are generated, it makes a histogram of the samples for several of the variables.[1]

1. Why would the samples coming from the Metropolis algorithm not give a good approximation to the posterior?
2. Modify the proposal used by the demo to $\hat{w} \sim \mathcal{N}(w, (1/100)I)$ instead of $\hat{w} \sim \mathcal{N}(w, I)$. Hand in your code and the update histogram plot.
3. Modify the proposal to use $\hat{w} \sim \mathcal{N}(w, (1/10000)I)$. Do you think this performs better or worse than the previous choice? (Briefly explain.)

[1]The "positive" variables are some of the positive weights when you fit an L2-regularized logistic regression model to the this data. The "negative" variables are some of the negative regression weights in that model, and the "neutral" ones are set to 0 in that model.

3 Deep Structured/Generative Models

Cancelled. (We will not get to this material soon enough to leave time for the assignment question. However, if lectures end before I get to the end of my material, I will record/distribute videos to you going over material on the remaining topics. You are definitely still welcome to do your projects on topics related to deep learning.)

4 Very-Short Answer Questions

Give a short and concise 1-sentence answer to the below questions.

1. Under what conditions can we use alpha-beta swap moves for approximate decoding?
2. Why do we use the parameterization $\phi_j(s) = \exp(w_{j,s})$ in UGMs?
3. What is the key difference between Younes' algorithm and our usual SGD setting of using an unbiased approximation of the gradient?
4. What is the key advantage of the graph structure in restricted Boltzmann machines?

5. What is the advantage of using a CRF, modeling $p(y \mid x)$, rather than treating supervised learning as special case of density estimation (modeling $p(y, x)$).
6. Why can fully-convolutional networks segment images of different sizes?
7. What is the key feature of a "sequence to sequence" RNN?
8. What are two advantages of the Bayesian approach to learning?

9. What is the difference between the posterior and posterior predictive distributions?
10. What is a hyper-hyper-parameter?
11. What is the key property of a conjugate prior?
12. In what setting is it unnecessary to include the q function in the Metropolis-Hastings acceptance probability?

CPSC 440 Project Proposal

This section is only to be done by project groups consisting of CPSC 440 students. For groups containing at least one CPSC 540 student, see the next section.

As mentioned at the start of term, I was sick for most of last year so did not have as much time to prepare for this course as I needed. This led to me re-using lectures/assignments that I've perviously used to teach graduate courses for people specifically interested in learning how to read machine learning papers, which is not the target audience for this course. I also found it really helpful seeing what people understood and did not understand in the course, and I am planning to use this to offer a much-better course next year. For example, here are some lessons learned:

- There needs to be a tighter coupling between math and code in lectures, as some people are having trouble going between them (undergrad students who are taking 5 courses do not have the time to fill in all the blanks on assignments like grad students who are taking 2-3 courses can).
- There needs to be a tighter coupling between lectures and assignments (again, undergrad students tend not to have the same time to fill in details and explore concepts like Lagrangians-in-more-than-one-variable on their own).
- The difficulty of the assignment questions needs to be better calibrated, as well as the workload across questions/assignments. (Some of the questions on early assignments required too much work, some of the questions on later assignments don't require any exploration.)
- Remove most of the material for the first 5 lectures, particularly anything that isn't built up to later (some of those topics belong in 340, some belong in more-advanced grad classes).
- Focus less on density estimation and message passing as unifying concepts, but instead focus on individual real-world problems and appropriate solutions for them.
- There needs to be a review of likelihoods and MLE/MAP before diving into things, since these concepts are not well-reinforced in 340 (and possibly big-O notation, at least in the tutorial in that case).
- Deep learning should appear much sooner, probably in the first two weeks. This will make it easier to see how the concepts covered in the course are applicable in modern applications, and will help differentiate the course from CPSC 422.
- "Abstract" lectures probably aren't ideal. For example, rather than having one lecture on Monte Carlo methods, introduce Monte Carlo early and talk about Monte Carlo for each method as they are introduced. Similarly, ideas like Bayesian statistics can be introduced early and people can get practice with these ideas in simple cases, rather than introducing the abstract idea late and saying "now you can make everything Bayesian" (without giving details for simple, moderate, and difficult cases).
- Follow 340's principle of delaying math concepts as long as possible. For example, EM should probably appear much later in the course while D-separation and conjugate priors should appear earlier. Similarly, make sure many of the most-useful and most-easily-used methods are covered early in the course (like CNNs and LSTMs) before people get overwhelmed with information.
- Less time spent, and overall less of a focus, on graphical models. I do not think they give enough "bang for your buck" compared to how much time we spend on them, and reducing this time will give more time for: (a) going into various topics in greater detail, (b) adding more deep learning stuff, and (c) possibly adding some reinforcement learning material.

At this point, I'm planning to re-order a lot of topics and redo most of the course material from scratch. In particular, I want to organize the material "by models" rather than by abstract concept. And for each model, we go through a structured list of topics to motivate the model and show in detail how it is used in various scenarios. In particular, for each model I want to use the following structure (which is roughly following the structure that 340 roughly follows):

1. **Motivating problem**: introduce an application that motivaties the model.
2. **Model definition**: how is it defined and what are the parameters (and their sizes)?
3. **General framework and other applications**: is this solving an abstract problem, and where else would this model be useful?
4. **Inference**: how do you do things like sampling, decoding, marginalization, conditioning, and test-set predictions/evaluations? (Theoretically and with code.)
5. **MLE**: how do you compute the MLE parameters? (Theoretically and with code.)
6. **MAP**: how do you introduce a prior and compute the MAP parameters?
7. **Bayes**: how do you make Bayesian predictions with the model?
8. **Multivarate** (for one-dimensional distributions): is there a multi-variable version of the model?
9. **MNIST**: what do MNIST samples look like if you use it as a density estimator?
10. **Generative classifiers**: how is this used with a generative classifier?
11. **Discriminiative classifier**: how do we add features to the model to use it as a discriminative model?
12. **Deep**: how do we add layers of hidden layers to learn features?

For your project, you are going to help give suggestions for making the course better. In particular, for your project I want you to pick a model and prepare a lecture and an assignment question on the model. For the model you pick, I would suggest either (a) picking a model where you thought I did a particularly-bad job explaining the details, or (b) pick a model that you think *should* be coverd in the course. In either case, you should learn some new things along the way (and preparing material is one of the best ways to learn things). Your lecture should follow the outline above, touching on most of those 12 items (the assignment question would only cover a small subset of them).

The particular deliverables due with this assignment are:

1. Specify which model you will focus on.
2. Give a 1-sentence-maximum description of how you might cover the 12 topics above in the model. Note: not all 12 topics make sense for all models, so it's ok to say that you won't cover some of them.
3. Give an outline for what the assignment related to the model will cover.
4. Send any other suggestions you have for improving the course.

To give you an idea of what this might look like, I am expecting to cover the basic Bernoulli model first (moving it from Lecture 6 to Lecture 2) and this is how I'm planning to handle each item for the basic Bernoulli model:

1. **Motivating problem**: what is the probability that a random person in a population has COVID-19?
2. **Model definition**: introduce θ parameter representing proportion.
3. **General framework and other applications**: introduce density estimation problem, give examples of "passing course", counting number of successes, and stopping times.
4. **Inference**: measuring test likelihood, decoding, binary sampling (with code and runtime).
5. **MLE**: set gradient to zero to give closed-form solution (with code and runtime), brief review of convexity to show that it's a min.
6. **MAP**: beta distribution, $\propto$ notation, and using a validation set for hyper-parameters.
7. **Bayes**: conjugate prior.

8. **Multivarate** (for one-dimensional distributions): product of independent Bernoullis. How to compute marginals/conditionals/samples/decoding.
9. **MNIST**: show the terrible performance with the product of independent Bernoullis model.
10. **Generative classifiers**: naive Bayes.
11. **Discriminiative classifier**: show that adding features leads to logistic regression.
12. **Deep**: add hidden layers to get binary neural networks (introduce backprop and autodiff). Show how to add multiple labels to a neural network and use CNNs for multi-label classification.

Also FYI, below is my current plan for the ordering of the models as well as where I expect to introduce various course topics:

1. Bernoulli: see above for concepts to be introduced.
2. Categorical: diseases grading as motivation, dice and letter grade as other examples, Lagrangians when doing MLE, log-linear when doing MLE?, Dirichlet when doing MAP, type II MLE when doing Bayes, general discrete when doing multivariate, softmax for discriminative, FCNs and RNNs/LSTMs and "encoding/decoding" framework and attention/transformers when doing deep.
3. Gaussian: percent grade as motivation, size and time as other examples, difference bewteen PDF and probability for continuous, introduce Gaussian properties while covering different inference tasks, Wishart and NIW when doing MAP, student and chi-squared when doing Bayes, multivariate normal for multivariate (and independent as diagonal), LDA/GDA for generative, least squares for discriminative, region proposals and end-to-end examples for deep.
4. Student/Laplace: outlier problem as motivation, introduce Monte Carlo and rejection/importance sampling for inference, multivariate versions, robust regression as discriminative.
5. Exponential families: count data as motivation, other distributions in general framework, sufficient statistics for MLE, ordinal/censored for discriminative.
6. Markov models: rain, list of applications and CS grads as other examples, ancestral sampling and CK and Viterbi and variable elimination for inference, conditional parameterization and homogeneous/inhomogenous (and parameter tieing) for learning, putting a Markov chain on top of a neural network.
7. DAGs: better exampe than "wet grass" as motivation, D-separation and treewdith/ICM/Gibbs (easier to explain in a DAG context) and MCMC, more-fancy parameterizations like treating it as supervised learning, structure learning along with regular learning, hierarchical Bayes and LDA here?, Bayes net classifier as discriminative, maybe put graph neural networks here?
8. Mixture models: some multi-modal continuous as example, EM when learning, mixture of experts as discriminative.
9. HMMs: forward-backward for inference (cheaper than using VE repeatedly).
10. UGMs: Ising or denoising or brain tumours as motivating, block approximate inference (and tree version of forward-backward), pseudolikelihood, CRF as discriminative, deep structured models.
11. RBMs.
12. Advanced inference/Bayes here? NP-Bayes/SMC/Variational.
13. VAE/GANs/beyond.

CPSC 540 (or mixed 440/540 groups) Literature Survey

This section is only to be done by project groups that contain at least one person enrolled in CPSC 540.

Reading academic papers is a skill that takes practice. When you first start out reading papers, you may find that you need to re-read things several times before you understand them, or that details will still be very fuzzy even after you've put a great amount of effort into trying to understand a paper. Don't panic, this is normal.

Even if you are used to reading papers from your particular sub-area, it can be challenging to read papers about a completely-different topic. Usually, people in different areas use different language/notation and focus on different issues. Nevertheless, many of the most-successful people in academia and industry are those that are able to understand/adapt ideas from different areas. (There are a ton of smart people in the world working on all sorts of amazing things, it's good to know how to communicate with as many of them as possible.)

A common technique when trying to understand a new topic (or reading scientific papers for the first time) is to read and write notes on 10 papers on the topic. When you read the first paper, you'll often find that it's hard to follow. This can make reading through it take a long time and might still leave you feeling that many things don't make sense; keep reading and trying to take notes. When you get to the second paper, it might still be very hard to follow. But when you start getting to the 8th or 9th paper, things often start making more sense. You'll start to form an impression of what the influential works in the area are, you'll start getting to used to the language and jargon, you'll start to understand what the main issues that people who work on the topic care about, and you'll probably notice some important references that weren't on your initial list of 10 papers. Ideally, you'll also start to notice how the topic has changed over time and you may get ideas of future work that you could do on the topic.

To help you make progress on your project or to give you an excuse to learn about a new topic, for this part you should write a literature survey of at least 10 academic papers on a particular topic. While your personal notes on the papers may be longer, the survey should be at most 4 pages of text (excluding references/tables/figures) in a format similar to the one for this document. Some logical components of a literature survey might be:

- A description of the overall topic, and the key themes/trends across the papers.
- A short high-level description of what was explored in each paper. For example, describe the problem being addressed, the key components of the proposed solution, and how it was evaluated. In addition, it is important to comment on the *why* questions: why is this problem important and why would this particular solution method make progress on it? It's also useful to comment on the strengths and weaknesses of the various works, and it's particularly nice if you can show how some works address the weaknesses of prior works (or introduce new weaknesses).
- One or more logical "groupings" of the papers. This could be in terms of the variant of the topic that they address, in terms of the solution techniques used, or in chronological terms.

Some advice on choosing the topic:

- The most logical/easy topic for your literature survey is a topic related to your course project, given that your final report will need a (shorter) literature survey included.
- If you are an undergrad, or a masters student without a research project yet, you may alternately want to choose a general area (like variance-reduced stochastic gradient, non-Gaussian graphical models, recurrent neural networks, matrix factorization, neural artistic style transfer, Bayesian optimization, transformer networks, etc.) as your topic.
- If you are a masters student that already has a thesis project, it could make sense to do a survey on a topic where ML intersects with your thesis (or where ML *could* intersect your thesis).

- If you are a PhD student, I would recommend using this an excuse to learn about a *completely different* topic than what you normally work on. Choose something hard that you would like to learn about, but previously haven't been able to justify spending the time exploring carefully. This can be invaluable to your future research, because during/after your PhD it often becomes hard to allocate time to learn completely new topics.