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

1. Name(s): Ricky Ma, Patricia Ye
2. Student ID(s): 82943424, 18139162

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?
   Students 3, 6, 8, and 9 did not study. State 1 corresponds to getting the question right.

2. What is the optimal decoding? Hand in your function to compute the optimal decoding.
   Optimal decoding: [1, 1, 1, 1, 1, 1, 1, 2, 2, 1]
   Probability of optimal decoding: 0.094930

```
function decode(phi1,phi2,E,n,s)
    (d,k) = size(phi1)
    optimalDecoding = ones(d)
    bestP = 0
    for x in DiscreteStates(d,k)
        # Calculate optimal decoding
        if n == s == 0
            pTilde = unnormalizedProb(x,phi1,phi2,E)
            if pTilde > bestP
                bestP = pTilde
                optimalDecoding = x
            end
        # Calculate optimal decoding conditioned on node n being in state s
        elseif x[n] == s
            pTilde = unnormalizedProb(x,phi1,phi2,E)
            if pTilde > bestP
                bestP = pTilde
                optimalDecoding = x
            end
        end
    end
    return optimalDecoding
end
```

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.
   $p(x_j = 1) = [0.134, 0.13, 0.073, 0.129, 0.129, 0.069, 0.117, 0.045, 0.047, 0.128]$

```
function marginals(phi1,phi2,E)
    (d,k) = size(phi1)
    marginals = zeros(d)
    for x in DiscreteStates(d,k)
        for (idx, s) in enumerate(x)
            if s == 1
                marginals[idx] += unnormalizedProb(x,phi1,phi2,E)
            end
        end
    end
    marginals = marginals ./ sum(marginals)
    return marginals
end
```

4. How does the decoding change if you condition on student 5 gets the question wrong?
   Optimal decoding: [1, 1, 1, 1, 2, 2, 1, 2, 2, 1]
   Probability of optimal decoding: 0.007911

5. How does the decoding change if you condition on student 8 getting the question right?
   Optimal decoding: [1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
   Probability of optimal decoding: 0.042191

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,. . . .
ICM converges to [1, 1, 1, 1, 1, 1, 1, 1, 1, 1] if it is initialized to "all right", and the optimal decoding [1, 1, 1, 1, 1, 1, 1, 2, 2, 1] if it is initialized to "all wrong".

```
function icm(phi1,phi2,E)
    (d,k) = size(phi1)
    xstar = rand([1, 2], d)  # random initialization
    nodes = 1:d

    t1 = time()
    iters = 0
    # Continuously loop through nodes in sequence
    for j in Iterators.cycle(nodes)
        if j == 1  # MB of "first" node
            jl,jr = d,2
        elseif j == d  # MB of "last" node
            jl,jr = d-1,1
        else  # MB of all other nodes
            jl,jr = j-1,j+1
        end
        # Conditional probability of xj given its Markov blanket
        condProb = zeros(2)
        for x in DiscreteStates(d,k)
            if (x[j] == 1) && (x[jl] == xstar[jl]) && (x[jr] == xstar[jr])
                condProb[1] += unnormalizedProb(x,phi1,phi2,E)
            elseif (x[j] == 2) && (x[jl] == xstar[jl]) && (x[jr] == xstar[jr])
                condProb[2] += unnormalizedProb(x,phi1,phi2,E)
            end
        end
        # Update xstar
        _, xstar[j] = findmax(condProb)
        iters += 1
        if xstar == [1, 1, 1, 1, 1, 1, 1, 2, 2, 1]
            break
        end
        if time() - t1 > 5
            println("Optimal decoding not found")
            break
        end
    end
    return xstar,iters
end    > icm
```

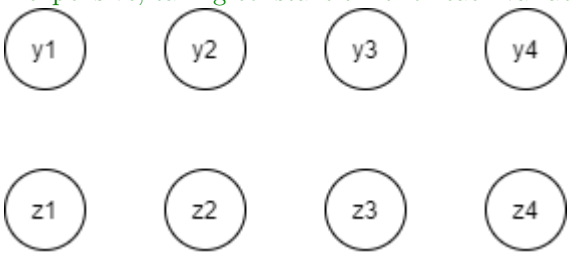
7. Suppose you want to make the dependency between students stronger, so you multiply every $\phi_e(s, s')$ by 10 for all edges e and states s and s' ("*phi2 = 10*phi2*" in the code). How would this change the answers to all of the above questions, and why? (Feel free to try it out, but think about why you get the answers that you do.)
The things we calculated, like optimal decoding, marginal and conditional probabilities, are all dependent on the relative potentials between the edges. Multiplying each edge potential by the same constant does not change the relationships between the nodes. Hence, multiplying every $\phi_e(s, s')$ does not change any of our previous answers.

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.)

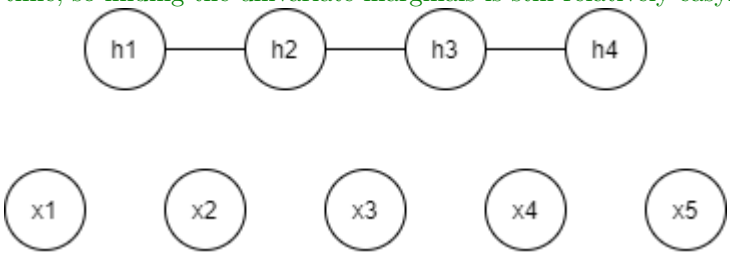
1.2 Conditional UGMs

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.

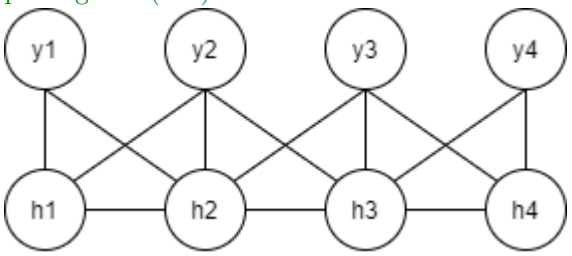
1. Conditioning on all the x and h values.
   By conditioning on all the x and h values, the y and z values become conditionally independent, meaning the graph is completely disconnected. Hence, computing the univariate marginals would be inexpensive, taking constant time for each variable.



2. Conditioning on all the z and y values.
   By conditioning on all the z and y values, the x values become conditionally independent and the h values form a chain. Chains can be easily computed using the forward-backward algorithm in $O(dk^2)$ time, so finding the univariate marginals is still relatively easy.

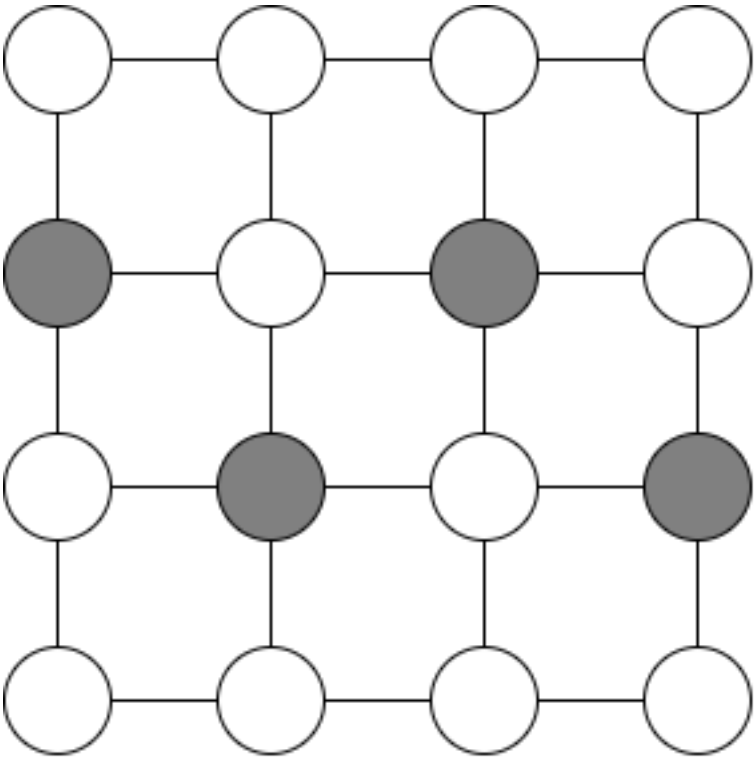


3. Conditioning on all the x and z values.
   Conditioning on all the x and z values leaves cycles in the UGM, meaning finding the univariate marginals will be NP-hard. However, we can simplify the problem by combining the y and h nodes into 4 group nodes to form a chain. The univariate marginals can then be computed using message passing in $O(dk^3)$ time.



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 $\beta > 0$,

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

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 quantities:

1. The posterior distribution, $p(\lambda \mid y, \alpha, \beta)$.

$$\begin{aligned} p(\lambda|y, \alpha, \beta) &\propto p(y|\lambda, \alpha, \beta)p(\lambda|\alpha, \beta) \\ &\propto p(y|\lambda)p(\lambda|\alpha, \beta) \\ &\propto \frac{\lambda^y \exp(-\lambda)}{y!} * \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda) \\ &\propto \frac{1}{y!} * \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1+y} \exp(-\lambda(\beta+1)) \end{aligned}$$

   This is proportional to the Gamma distribution, $\lambda \sim \text{Gamma}(\alpha + y, \beta + 1)$.
   Hence, $p(\lambda \mid y, \alpha, \beta) = \frac{(\beta+1)^{\alpha+y}}{\Gamma(\alpha+y)} \lambda^{\alpha-1+y} \exp(-\lambda(\beta+1))$.

2. The marginal likelihood of y given the hyper-parameters α and β, $p(y \mid \alpha, \beta) = \int p(y, \lambda \mid \alpha, \beta) d\lambda$.

$$\begin{aligned} p(y \mid \alpha, \beta) &= \int p(y, \lambda \mid \alpha, \beta) d\lambda. \\ &= \int p(y \mid \lambda, \alpha, \beta) p(\lambda \mid \alpha, \beta) d\lambda \\ &= \int \frac{\lambda^y \exp(-\lambda)}{y!} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} \exp(-\beta\lambda) d\lambda \\ &= \frac{\beta^\alpha}{y!\Gamma(\alpha)} \int \lambda^{\alpha+y-1} \exp(-(\beta+1)\lambda) d\lambda \end{aligned}$$

Now set $t = (\beta+1)\lambda, \lambda = \frac{t}{\beta+1}, d\lambda = \frac{1}{\beta+1} dt$

$$\begin{aligned} p(y \mid \alpha, \beta) &= \frac{\beta^\alpha}{y!\Gamma(\alpha)} \int \left(\frac{t}{\beta+1}\right)^{\alpha+y-1} e^{-t} \frac{1}{\beta+1} dt \\ &= \frac{\beta^\alpha}{y!\Gamma(\alpha)} \left(\frac{1}{\beta+1}\right)^{\alpha+y} \int t^{\alpha+y-1} e^{-t} dt \\ &= \frac{\beta^\alpha}{y!\Gamma(\alpha)} \left(\frac{1}{\beta+1}\right)^{\alpha+y} \Gamma(\alpha+y) \\ &= \frac{\Gamma(\alpha+y)}{y!\Gamma(\alpha)} \left(\frac{1}{\beta+1}\right)^{y} \left(\frac{\beta}{\beta+1}\right)^{\alpha} \\ &= \frac{\Gamma(\alpha+y)}{y!\Gamma(\alpha)} \left(\frac{1}{\beta+1}\right)^{y} \left(1 - \frac{1}{\beta+1}\right)^{\alpha} \end{aligned}$$

So $y \mid \alpha, \beta \sim \text{NB}(\alpha, \frac{1}{\beta+1})$

3. The posterior mean estimate for λ,

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

Since $\lambda \mid y, \alpha, \beta \sim \text{Gamma}(y+\alpha, \beta+1)$, $\mathbb{E}_{\lambda \mid y,\alpha,\beta}[\lambda] = \frac{\alpha+y}{\beta+1}$

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

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

$$\begin{aligned}
p(\tilde{y} \mid y, \alpha, \beta) &= \int p(\tilde{y}, \lambda \mid y, \alpha, \beta) d\lambda \\
&= \int p(\tilde{y} \mid \lambda, y, \alpha, \beta) p(\lambda \mid y, \alpha, \beta) d\lambda \\
&= \int p(\tilde{y} \mid \lambda, \alpha, \beta) p(\lambda \mid y, \alpha, \beta) d\lambda \\
&= \int \frac{\lambda^{\tilde{y}} \exp(-\lambda)}{\tilde{y}!} \frac{(\beta+1)^{\alpha+y}}{\Gamma(\alpha+y)} \lambda^{\alpha+y-1} \exp(-(\beta+1)\lambda) d\lambda \\
&= \frac{(\beta+1)^{\alpha+y}}{\tilde{y}!\Gamma(\alpha+y)} \int \lambda^{\alpha+y+\tilde{y}-1} \exp(-(\beta+2)\lambda) d\lambda
\end{aligned}$$

Now set $t = (\beta+2)\lambda, \lambda = \frac{t}{\beta+2}, d\lambda = \frac{1}{\beta+2} dt$

$$\begin{aligned}
p(\tilde{y} \mid y, \alpha, \beta) &= \frac{(\beta+1)^{\alpha+y}}{\tilde{y}!\Gamma(\alpha+y)} \int \left(\frac{t}{\beta+2}\right)^{\alpha+y+\tilde{y}-1} e^{-t} \frac{1}{\beta+2} dt \\
&= \frac{(\beta+1)^{\alpha+y}}{\tilde{y}!\Gamma(\alpha+y)} \left(\frac{1}{\beta+2}\right)^{\alpha+y+\tilde{y}} \int t^{\alpha+y+\tilde{y}-1} e^{-t} dt \\
&= \frac{(\beta+1)^{\alpha+y}}{\tilde{y}!\Gamma(\alpha+y)} \left(\frac{1}{\beta+2}\right)^{\alpha+y+\tilde{y}} \Gamma(\alpha+y+\tilde{y}) \\
&= \frac{\Gamma(\alpha+y+\tilde{y})}{\tilde{y}!\Gamma(\alpha+y)} \left(\frac{1}{\beta+2}\right)^{\tilde{y}} \left(\frac{\beta+1}{\beta+2}\right)^{\alpha+y} \\
&= \frac{\Gamma(\alpha+y+\tilde{y})}{\tilde{y}!\Gamma(\alpha+y)} \left(\frac{1}{\beta+2}\right)^{\tilde{y}} \left(1 - \frac{1}{\beta+2}\right)^{\alpha+y}
\end{aligned}$$

So $\tilde{y} \mid y, \alpha, \beta \sim \text{NB}(\alpha+y, \frac{1}{\beta+2})$

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

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

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.

degree=3, $\sigma = 1.0$, $\lambda = 0.02$

```
function leastSquaresEmpiricalBasis(x,y)
    λs = [0.01, 0.02, 0.03, 0.04, 0.05]
    σs = [0.5, 0.75, 1, 1.25, 1.5]
    ps = [1, 2, 3, 4, 5]

    maxlikelihood = -Inf
    best = [0, 0, 0]
    for p in ps
        Z = polyBasis(x,p)
        n,k = size(Z)
        for λ in λs
            for σ in σs
                Θ = (1/σ^2) * (Z'*Z) + λ*I  # posterior precision
                w = (1/σ^2) * inv(Θ) * (Z'*y)  # posterior mean
                # marginal likelihood
                c = (λ^(.5k)) / (((σ*sqrt(2*pi))^n)*det(Θ)^(.5))
                l = log(c) - (1/(2σ^2))*norm(Z*w-y) - .5λ*norm(w)
                # update hyperparameters
                if l > maxlikelihood
                    best = [p,σ,λ]
                    maxlikelihood = l
                end
            end
        end
    end

    println(best)
    p,σ,λ = best
    p = Int(p)
    Z = polyBasis(x,p)
    Θ = (1/σ^2) * (Z'*Z) + λ*I  # posterior precision
    w = (1/σ^2) * inv(Θ) * (Z'*y)  # posterior mean
    predict(xhat) = polyBasis(xhat,p)*w
    return GenericModel(predict)
end
```

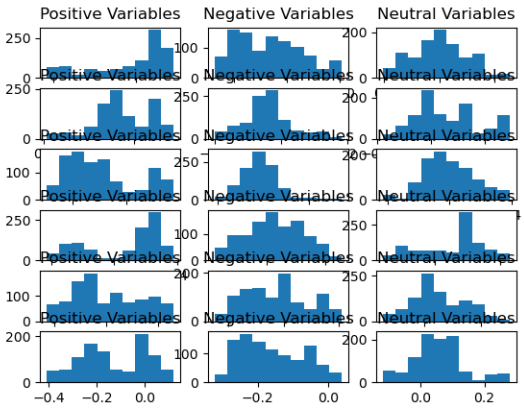
2.3 Markov Chain Monte Carlo

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?
   The Metropolis algorithm would not give a good approximation of the posterior because it does not necessarily converge to the optimal value and it needs extra assumptions to ensure the solution is unique.

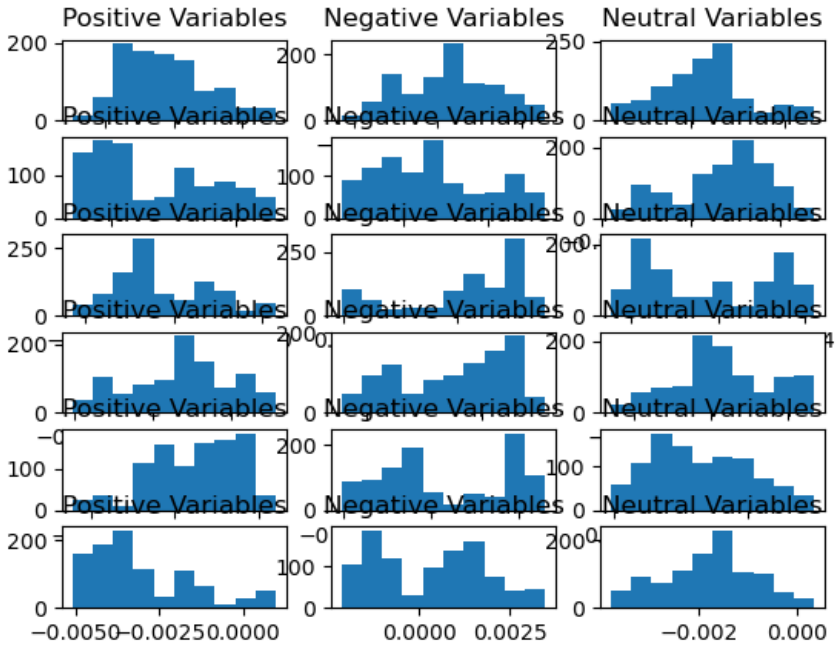
2. Modify the proposal used by the demo to $\hat{w} \sim \mathcal{N}(w, (1/100)I)$ instead of $\hat{w} \sim \mathcal{N}(w, I)$. Hand in your code and the update histogram plot.

```
function blogreg(X,y,lambda,nSamples,sd)
    (n,d) = size(X);
    samples = zeros(nSamples,d)

    # Initialize and compute negative Log-posterior (up to constant)
    w = zeros(d,1);
    log_p = logisticObj(w,X,y,lambda)

    nAccept = 0
    for s in 1:nSamples
        # Propose candidate
        dist = Normal(0,sd)
        wHat = w + rand(dist,d)
        log_phat = logisticObj(wHat,X,y,lambda)

        # Metropolis-Hastings accept/reject step (in log-domain)
        logR = log_phat - log_p
        if log(rand()) < logR
            w = wHat
            log_p = log_phat
            nAccept += 1
            @printf("Accepted sample %d, acceptance rate = %f\n",s,nAccept/s)
        end
        samples[s,:] = w
    end
    return samples
end
```



[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. 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.)
This performs worse than the previous choice because it results in accepting an unrepresentative proportion of the proposed samples. Approximately 99% of the proposed samples are accepted compared to $\sim 81\%$ from before.



3 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?
   We can use alpha-beta swap moves when each node is assigned to some state α or β, and we assume that for every edge between a pair of nodes, $log\phi_{ij}(\alpha,\alpha) + log\phi_{ij}(\beta,\beta) \geq log\phi_{ij}(\alpha,\beta) + log\phi_{ij}(\beta,\alpha)$.

2. Why do we use the parameterization $\phi_j(s) = \exp(w_{j,s})$ in UGMs?
   By setting $\phi_j(s)$ equal to $\exp(w_{j,s})$, the negative log-likelihood of the UGM becomes convex, which makes optimization easier.

3. What is the key difference between Younes' algorithm and our usual SGD setting of using an unbiased approximation of the gradient?
   Younes' algorithm combines SGD with Gibbs sampling using 1 Gibbs sample per iteration. This gives an approximation of the gradient by using a Monte Carlo approximation of the next expected value.

4. What is the key advantage of the graph structure in restricted Boltzmann machines?
   The graph structure lets us do block Gibbs sampling, where a block consists of either hidden variables or observed variables. My conditioning on one block, we eliminate all dependencies, making inference easy.

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 $p(y, x)$).
   CRF doesn't need to model features x like it does in density estimation (CRF is discriminative).

6. Why can fully-convolutional networks segment images of different sizes?
   By performing parameter tieing within convolutions, FCNs can segment images of different sizes.

7. What is the key feature of a "sequence to sequence" RNN?
   The inputs and outputs to seq2seq RNNs are both sequences.

8. What are two advantages of the Bayesian approach to learning?
   The Bayesian approach to learning considers the possibilities of all possible models weighted using the posterior distribution, instead of just the most likely model. It also allows models with an unknown/infinite number of parameters.

9. What is the difference between the posterior and posterior predictive distributions?
   The posterior distribution is the distribution over the parameters of a model given hyper-parameters and observed data. The posterior predictive distribution is the distribution over new data given existing data.

10. What is a hyper-hyper-parameter?
    A hyper-hyper-parameter is a prior we put on a hyper-parameter. For example, a parameter θ of a Bernoulli distribution follows the Beta distribution with hyper-parameters α and β. We can add another prior γ to act as a hyper-hyper-parameter and create dependencies between different θs

11. What is the key property of a conjugate prior?
    The conjugate prior must come from the same family of distribution as the posterior.

12. In what setting is it unnecessary to include the q function in the Metropolis-Hastings acceptance probability?
    It is unnecessary to include the q function when the Markov chain is symmetric ($q(x^t|\hat{x}^t) = q(\hat{x}^t|x^t)$)