6.867: Exercises (Week 9)

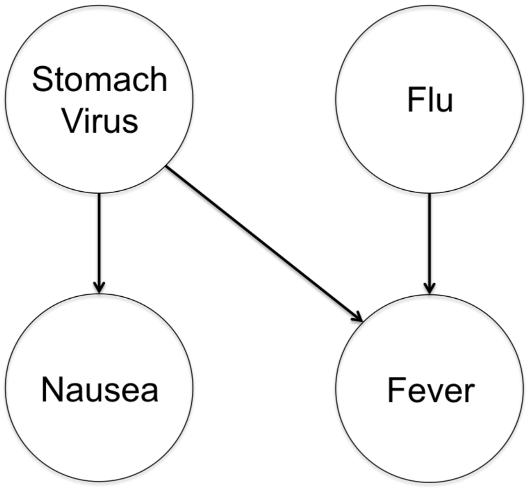
November 10, 2017

Contents

Solution: Don't look at the solutions until you have tried your absolute hardest to solve the problems.

1 Bayesian Network Parameter Estimation

You are a doctor working at a clinic and want to model the relationship between common diseases and symptoms. Drawing on your vast knowledge of the human body you decide to use the following Bayesian network:



Given data on previous patients, you now want to determine the parameters of this network. Let X_1, X_2, X_3, and X_4 be binary random variables representing the event of a given patient having a stomach virus, having the flu, having nausea and having a fever, respectively. In other words, these variables denote the nodes in the network.

Your dataset D consists of the patient profiles of n past patients. The dataset is complete, meaning that you know the values of $X_1^{(i)}$, $X_2^{(i)}$, $X_3^{(i)}$, and $X_4^{(i)}$ for all patients i. We will first find the maximum likelihood parameter estimates for the network.

(a) Provide an expression for the log-likelihood of the data given this network structure in terms of parameterized conditional probability distributions that shows that each conditional probability distribution can be optimized independently.

Solution: Define the following conditional probabilities:

$$p_1(i) = p(X_1^{(i)}; \theta_1)$$
$$p_2(i) = p(X_2^{(i)}; \theta_2)$$
$$p_3(i) = p(X_3^{(i)} | X_1^{(i)}; \theta_3)$$
$$p_4(i) = p(X_4^{(i)} | X_1^{(i)}, X_2^{(i)}; \theta_4)$$

We get the following log-likelihood for the data:

$$L(D;\theta) = \sum_{i=1}^{n}\sum_{j=1}^{4} \log(p_j(i))$$
$$= \sum_{j=1}^{4}\sum_{i=1}^{n} \log(p_j(i))$$

From here it is easy to see that each of θ_j can be optimized independently.

(b) Assume that the conditional distributions are fully parameterized (ie. in $p(y_1|y_2)$ the distribution over y_1 can be set independently for each value of y_2). Provide expressions for the maximum likelihood parameters in terms of patient frequency counts (eg. $N(X_1 = 1)$ is the number of patients with a stomach virus).

Solution: In general, the conditional probability distributions can be written as:

$$p(x_i|x_{pa(i)}; \theta_{x_i|x_{pa(i)}})$$

Where $x_{pa(i)}$ is the set of parent nodes of node x_i. We can then define the probability for every realization of parent node values independently as:

$$\theta_{ML, x_i=x|x_{pa(i)}=y} = \frac{N(x_i = x, x_{pa(i)} = y)}{N(x_{pa(i)} = y)}$$

Where x is the value of node x_i and y is the vector of values for all parent nodes.

(c) We have been assuming that the conditional probability distributions are fully parameterized. Under this assumption, how many parameters must be learned for the conditional probability distribution of X_4?

Solution: The distribution will have $2^2 = 4$ parameters which is exponential in the number of parents. In general, requiring a number of parameters exponential in the number of parents may make it easy for the model to overfit in the case of limited data.

To decrease the number of parameters we can make some assumptions about the structure of our conditional probability distributions. In practice, for discrete variables a Noisy-OR distribution is commonly used. Here, any individual parent being true can independently cause the child to be true but with some error rate p_i for parent i. This means that for a node x_0 with n parents, where all variables take values in $\{0, 1\}$ we get:

$$p(x_0 = 1|x_1, ..., x_n) = 1 - \prod_{i=1}^{n} p_i^{x_i}$$

(d) In our disease example, if we know $p(X_4 = 1|X_1 = 0, X_2 = 1) = 0.9$ and $p(X_4 = 1|X_1 = 1, X_2 = 0) = 0.7$, what is $p(X_4 = 1|X_1 = 1, X_2 = 1)$?

Solution:

$$p(X_4 = 1|X_1 = 1, X_2 = 1) = 1 - (1 - 0.9)(1 - 0.7) = 0.97$$

(e) Using a Noisy-OR distribution, how many parameters must be learned for the conditional probability distribution of X_4? Why might this generalize better than a fully parameterized distribution?

Solution: We now only need to learn 2 parameters which is linear in the number of parents.

This setup will be especially helpful in generalizing for rare cases without much data (eg. patients who have both a stomach virus and the flu). In this new setting we gain information about this case from other examples (only stomach virus or only flu) whereas in the fully parameterized case we only learn from the few rare examples.

2 K-Means Clustering

This question will explore some cases in which the k-means clustering algorithm may or may not give ideal results. All parts of this question will use squared Euclidean distance as a distance metric.

(a) First provide an expression for the objective function being minimized by a k-means algorithm using Euclidean distance squared.

Solution: For n data points and k clusters whose centroids are denoted by the set K:

$$C = \sum_{i=1}^{n} \min_{k \in K} \|x^{(i)} - k\|_2^2$$

(b) Consider a 1D dataset generated from the two Gaussian distributions $N_1(\mu_1, \sigma_1^2)$ and $N_2(\mu_2, \sigma_2^2)$. Assume the dataset contains n points from each distribution for some arbitrarily large n.

Let's define the optimal classifier as the classifier that assigns each point to the most likely cluster given knowledge of the generating distribution. Consider the case where $\sigma_1^2 = \sigma_2^2$. Would you expect a 2-means clustering algorithm to approximate the optimal classifier for large n? Explain.

Solution: Yes. With identical variances, the optimal classifer has a decision boundary at the midpoint between μ_1 and μ_2. Given large enough n, the two k-means centroids will be near μ_1 and μ_2 respectively and the assignment of points to clusters will be a similar split around the midpoint.

(c) Now consider the case where $\sigma_1^2 \gg \sigma_2^2$. Would you expect a 2-means clustering algorithm to approximate the optimal classifier for large n? Explain.

Solution: No. If the variances are different, then the optimal decision boundary is closer to one centroid than the other. Since k-means clustering will always have an equidistant split between the two centroids, this behavior cannot be reproduced and thus k-means clustering will erroneoously assign more points to the cluster with a smaller variance.

(d) What might be a better way to cluster the data in the case where $\sigma_1^2 \gg \sigma_2^2$?

Solution: Fitting a mixture of k-Gaussians with unknown variances will likely have better results because it does not implicitly assume the clusters have equal variance.

3 EM for word counts

You want to create a language model that models text. Instead of using a simple bi-gram model, you observe that the grammatical word class of a word can usually be predicted by the previous word and decide to explicitly model this. Consider the probabilistic model of the following form:

$$p(w_2, c \mid w_1) = q(c \mid w_1)q(w_2 \mid c)$$

Here w_1 and w_2 are words, drawn from some set of possible words $\mathcal{V}$. c is a word class, which can take any value in the set $\{1, 2, \ldots, k\}$ for some integer k. $q(c \mid w_1)$ and $q(w_2 \mid c)$ are the parameters of the model. We can interpret this as a model where: (1) a class c is generated by word w_1; (2) w_2 is then generated by the class c chosen in step 1.

Under this model, we can derive

$$p(w_2 \mid w_1) = \sum_{c=1}^{k} q(c \mid w_1)q(w_2 \mid c)$$

This will be a model of the conditional probability of seeing the word w_2 given that the previous word in a sentence was w_1. We want to derive a method to compute the q parameters for this model.

(a) Write down an expression for $p(c \mid w_1, w_2)$ as a function of the q parameters.

Solution:

The posterior probability (our "belief") in class c, after observing w_1 and w_2, is found using Bayes' theorem.

$$\begin{aligned} p(c \mid w_1, w_2) &= \frac{p(c, w_1, w_2)}{\sum_{c'} p(c', w_1, w_2)} \\ &= \frac{p(w_1)p(w_2, c \mid w_1)}{\sum_{c'} p(w_1)p(w_2, c' \mid w_1)} \\ &= \frac{q(c \mid w_1)q(w_2 \mid c)}{\sum_{c'} q(c' \mid w_1)q(w_2 \mid c)} \end{aligned}$$

(b) Say for each pair of words w_1, w_2, count(w_1, w_2) is the number of times w_1 is followed by w_2 in our training data. We are going to derive an EM algorithm for optimization of the following log likelihood function:

$$L(\theta) = \sum_{w_1, w_2} \text{count}(w_1, w_2) \log p(w_2 \mid w_1)$$

For given parameter values q, define count$(c \mid w_1)$ to be expected number of times that w_1 generates class c, and define count$(w_2 \mid c)$ to be the expected number of times w_2 is generated by class c. (Here expectation is taken with respect to the distribution defined by the q parameters.) State how count$(c \mid w_1)$ and count$(w_2 \mid c)$ can be calculated as a function of the q parameters:

Solution:

To calculate count$(c \mid w_1)$ based on our beliefs $p(c \mid w_1, w_2)$ from the previous question, we estimate: of all occurrences of word w_1, how many do we believe generated class c? This involves marginalizing w_2.

$$\begin{aligned} \text{count}(c \mid w_1) &= \sum_{w_2} \text{count}(w_1, w_2) p(c \mid w_1, w_2) \\ &= \sum_{w_2} \text{count}(w_1, w_2) \frac{q(c \mid w_1)q(w_2 \mid c)}{\sum_{c'} q(c' \mid w_1)q(w_2 \mid c')} \end{aligned}$$

Similarly, we marginalize w_1 to estimate: of all believed occurrences of class c, how many were associated with word w_2?

$$\begin{aligned} \text{count}(w_2 \mid c) &= \sum_{w_1} \text{count}(w_1, w_2) p(c \mid w_1, w_2) \\ &= \sum_{w_1} \text{count}(w_1, w_2) \frac{q(c \mid w_1)q(w_2 \mid c)}{\sum_{c'} q(c' \mid w_1)q(w_2 \mid c')} \end{aligned}$$

(c) Now describe how the q parameters are recalculated in the EM algorithm, based on the $\text{count}(c \mid w_1)$ and $\text{count}(w_2 \mid c)$ counts derived in the previous part.

Solution: In several instances of EM, the M-step update rules for frequency parameters involving the hidden data (e.g., Gaussian mixture proportions) work out to the weighted average frequency in the training data, where the weighting is by our beliefs about the hidden data for each training example.

For example, $\text{count}(c \mid w_1)$ from the last part tells us how many times we believe w_1 generated class c. We get a weighted frequency estimate of $q(c \mid w_1)$ by dividing this by the total number of observations of w_1.

$$q(c \mid w_1) \leftarrow \frac{\text{count}(c \mid w_1)}{\sum_{c'} \text{count}(c' \mid w_1)}$$

The denominator could be written in several equivalent ways. Similarly we divide $\text{count}(w_2 \mid c)$, the number of times we believe class c generated w_2, by the total number of times we believe class c occurred.

$$q(w_2 \mid c) \leftarrow \frac{\text{count}(w_2 \mid c)}{\sum_{w_2'} \text{count}(w_2' \mid c)}$$

(d) Say we initialize the parameters to be $q(c \mid w_1) = 1/k$ for all w_1,c, and $q(w_2 \mid c) = 1/ \mid \mathcal{V} \mid$ for all w_2,c. Given these initial parameter values, what parameter values will the EM algorithm converge to?

Solution: $q(c \mid w_1)$ will not change, simply because there is nothing to distinguish c from any other class – $\text{count}(c \mid w_1)$ is the same for all c. However, $\text{count}(w_2 \mid c)$ will vary based on how many times we observe w_2 in the training data. Therefore, we will set $q(w_2 \mid c)$ to the frequency of w_2 in the training data:

$$q(w_2 \mid c) \leftarrow \frac{\sum_{w_1} \text{count}(w_1, w_2)}{\sum_{w_1, w_2} \text{count}(w_1, w_2)}$$

("Of all training examples, what fraction had w_2?" Again, this could be written in several equivalent ways.) After this update, there is still nothing distinguishing any class from any other, so we'll be stuck after one iteration.

4 Missing data

In most real world datasets we do not have the privilege of complete data. Certain observations may be incorrect or never taken in the first place. For example, in survey data you may not have answers to every question for every person because they chose not to answer some questions. Due to this being able to make predictions despite missing data is very important in practice.

To gain intuition about how to handle missing data we'll start with a very simple problem, in which a single attribute of a single data set is missing. There are two attributes, A and B, and this is our data set, $\mathcal{D}$:

```
i    A  B
1    1  1
2    1  1
3    0  0
4    0  0
5    0  0
6    0  H ***missing **
7    0  1
8    1  0
```

Assume the data is *missing completely at random* (MCAR): that is, that the fact that it is missing is independent of its value.

Our goal is to estimate $\Pr(A, B)$ from this data. We'd really like to find the maximum-likelihood parameter values, if we can. The likelihood is

$$\mathcal{L}(\theta) = \log \Pr(\mathcal{D}; \theta) = \log\left(\Pr(\mathcal{D}, H = 0; \theta) + \Pr(\mathcal{D}, H = 1; \theta)\right) \quad .$$

(a) Kim is lazy and decides to ignore $x^{(6)}$ all together, and estimate the parameters:

$$\widehat{\theta}^1 = \begin{pmatrix} 3/7 & 1/7 \\ 1/7 & 2/7 \end{pmatrix} = \begin{pmatrix} .429 & .143 \\ .143 & .285 \end{pmatrix}$$

What is $\mathcal{L}(\widehat{\theta}^1)$?

Solution: If we do that, then

$$\begin{aligned} \mathcal{L}(\widehat{\theta}^1) &= \log\left(\Pr(00\ ;\ \widehat{\theta}^1)\prod_{i\neq 6}\Pr(x^{(i)}\ ;\ \widehat{\theta}^1) + \Pr(01\ ;\ \widehat{\theta}^1)\prod_{i\neq 6}\Pr(x^{(i)}\ ;\ \widehat{\theta}^1)\right) \\ &= 3\log 0.429 + 2\log 0.143 + 2\log 0.285 + \log(0.429 + 0.143) \\ &= -9.498 \end{aligned}$$

(b) Jan thinks we should let H be the 'best' value it could have, that is to make the log likelihood as large as possible, and so tries setting $H = 0$ and then $H = 1$ and computes the log likelihood of the complete data in both cases. What value gives the highest complete-data log likelihood? What is the likelihood value?

Solution: That value is 0. So, then we'd have

$$\widehat{\theta}^2 = \begin{pmatrix} .5 & .125 \\ .125 & .25 \end{pmatrix}$$

and

$$\mathcal{L}(\widehat{\theta}^2) = -9.481 \quad .$$

That's a little better!

(c) Evelyn thinks this is all unprincipled messing around and says we should optimize the thing we want to optimize! That is,

$$\widehat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta) \ .$$

Evelyn also thinks we can just use the code for gradient descent that we already built in 6.867 to do this job. Is Evelyn right?

Solution: Evelyn is absolutely right about (if at all possible!) optimizing the thing we want to optimize.

However, we cannot do this with basic gradient descent because of the constraint that $\widehat{\theta}$ be a valid probability distribution; that constraint is *not* maintained by our basic gradient descent code. So, we'd have to investigate constrained optimization algorithms, or try to formulate the problem using Lagrange multipliers.

(d) Ariel was paying close attention in lecture and thinks EM is good to use for estimation with missing data because the missing data is effectively just a latent variable.

Let's start with the guess

$$\theta_0 = \begin{pmatrix} .25 & .25 \\ .25 & .25 \end{pmatrix}$$

What is the formula for the E step in this problem? What is the numerical result in this particular case?

Solution:

$$\tilde{P}(H = 1) = \Pr(H = 1 \mid \mathcal{D} \ ; \ \theta_0) = \Pr(H = 1 \mid x^{(6)} \ ; \ \theta_0) = \Pr(B = 1 \mid A = 0 \ ; \ \theta_0) = 0.5$$

(e) Ariel's roommate Angel joins in the EM game and computes the M step, to get θ_1. What is the numerical value in this case, and why?

Solution:

$$\begin{aligned} \theta_1 &= \arg\max \theta \left(0.5 \log \Pr(\mathcal{D}, H = 0 \ ; \ \theta) + 0.5 \log \Pr(\mathcal{D}, H = 1 \ ; \ \theta)\right) \\ &= \begin{pmatrix} 7/16 & 3/16 \\ 2/16 & 4/16 \end{pmatrix} \end{aligned}$$

This step is not immediately obvious: to derive it, we need to take the derivative with respect to each of the parameters, set to 0, and solve for θ. We find that we can treat the estimation problem as one in which we have a data item for each possible value of H, weighted by the probability that H has that value. We get such a decomposition because, for each particular value of H, only one of the parameter estimates is affected.

We get the same result by doing estimation as usual, but treating $\Pr(H)$ as giving us fractional counts on both data cases:

```
i    A  B   count = P~(H)
1    1  1
2    1  1
3    0  0
4    0  0
5    0  0
6a   0  0    0.5
6b   0  1    0.5
7    0  1
8    1  0
```

On subsequent EM iterations, we have $\mathcal{L}(\theta) = -10.39, -9.47, -9.4524, -9.4514, \ldots$.

(f) Will EM always find a solution that maximizes $\mathcal{L}$?

Solution: No. It will converge monotonically to a *local optimum* but it may not be a global optimum, and will depend, in general, on your initial guess.

5 Factor Analysis

You are given a dataset $X \in \mathbb{R}^{n\times k}$ of gene expression data where k is the number of genes and n is the number of observations. But $n < k$, posing a challenge for many modelling approaches (eg. fitting a Gaussian). You believe patterns in gene expression data can be explained by less variables than the number of genes so instead you decide to use the following factor analysis model where $z \in \mathbb{R}$ is your latent variable:

$$z \sim N(0,\ 1)$$
$$x|z \sim N(wz,\ I)$$

where $w \in \mathbb{R}^{k\times 1}$ and I is the $k \times k$ identity matrix. We are modelling the variations in the k-dimensional variable x as a function of variations in the lower dimensional variable z. Here we have chosen z to be 1D to simplify the calculations, but in general this method can be used with multivariate z.

In this question we will derive the EM update steps for determining the maximum likelihood estimate of w. For the following questions it will be useful to note that:

$$p(z, x; w) \sim N\left(\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1 & w^T\\ w & ww^T + I\end{bmatrix}\right)$$

And if y_1 and y_2 are Gaussian random variables, we have:

$$\mu_{y_1|y_2} = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_2 - \mu_2)$$
$$\Sigma_{y_1|y_2} = \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}$$

where $\Sigma_{ij} = cov(y_i, y_j)$.

(a) First we will compute the E-step. Provide an expression for $p(z|x; w^{(t)})$.

Solution:

$$z|x \sim N(w^{(t)T}(w^{(t)}w^{(t)T} + I)^{-1}x,\ 1 - w^{(t)T}(w^{(t)}w^{(t)T} + I)^{-1}w^{(t)})$$

(b) In the M-step we want to find $w^{(t+1)}$ that maximizes the expected complete log likelihood $\sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}, z^{(i)}; w^{(t+1)})]$ where the expectation is taken with respect to $p(z|x; w^{(t)})$. Show that this is equivalent to maximizing $L = \sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}|z^{(i)}; w^{(t+1)})]$.

Solution:

$$\sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}, z^{(i)}; w^{(t+1)})] = \sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}|z^{(i)}; w^{(t+1)})] + \mathbb{E}[\log p(z^{(i)})]$$

Since $\mathbb{E}[\log p(z^{(i)})]$ isn't a function of $w^{(t+1)}$, we can ignore it for the purpose of maximization and we are left with L.

(c) Provide an expression for L.

Solution:

$$\sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}|z^{(i)}; w^{(t+1)})] = \sum_{i=1}^{n} \mathbb{E}[-\frac{1}{2}\log|2\pi I| - \frac{1}{2}(x^{(i)} - w^{(t+1)}z^{(i)})^T(x^{(i)} - w^{(t+1)}z^{(i)})]$$

(d) Find the update step by solving for the maximizing $w^{(t+1)}$ in terms of $\mu_{z|x}$ and $\Sigma_{z|x}$.

Solution: Taking the gradient with respect to $w^{(t+1)}$ and setting to 0 we get:

$$
\begin{aligned}
0 &= \nabla_w \sum_{i=1}^{n} \mathbb{E}[\log p(x^{(i)}|z^{(i)}; w^{(t+1)})] \\
&= \sum_{i=1}^{n} \mathbb{E}[\nabla_w \log p(x^{(i)}|z^{(i)}; w^{(t+1)})] \\
&= \sum_{i=1}^{n} \mathbb{E}[\nabla_w (-\frac{1}{2}(x^{(i)} - w^{(t+1)}z^{(i)})^T(x^{(i)} - w^{(t+1)}z^{(i)}))] \\
&= \sum_{i=1}^{n} \mathbb{E}[x^{(i)}z^{(i)} - z^{(i)2}w^{(t+1)}] \\
&= \sum_{i=1}^{n} x^{(i)}\mathbb{E}[z] - \sum_{i=1}^{n} w^{(t+1)}\mathbb{E}[z^2] \\
\Rightarrow w^{(t+1)} &= \frac{\sum_{i=1}^{n} x^{(i)}\mu_{z|x^{(i)}}}{\sum_{i=1}^{n} \mu_{z|x^{(i)}}^2 + \Sigma_{z|x^{(i)}}}
\end{aligned}
$$

Where $\mu_{z|x}$ and $\Sigma_{z|x}$ are functions of $w^{(t)}$ (see part a).

6 Time Varying Topic Model

In recent years, social media outlets represent a wealth of data related to people's opinions and world events. As such, tools for understanding this data have grown increasingly important. Intrigued by this trend, you decide you want to perform topic modelling on a corpus of tweets collected over some time interval. Your dataset consists of M tweets (or documents) at each time step t, with tweet i at time t denoted by $D_{i,t}$, for T total time steps. Tweet $D_{i,t}$ is composed of $L_{i,t}$ words where the j^{th} word is denoted by $w_{i,t,j}$.

You believe that any given tweet can be decomposed as a mixture of K topics. However, as is the nature of Twitter, you believe the topic proportions and the nature of the topics themselves are changing over time. Because of this you need to figure out a way to handle the time varying nature of the topics in your topic modelling. Here is the generative process for a static topic model you are using as a starting point:

1. For each document i choose $\theta_i \sim \mathrm{Dir}(\alpha)$
2. For each topic k choose $\phi_k \sim \mathrm{Dir}(\beta)$
3. For each word i, j in document i and position j:
   (a) Choose a topic $z_{i,j} \sim \mathrm{Multi}(\theta_i)$
   (b) Choose a word $w_{i,j} \sim \mathrm{Multi}(\phi_{z_{i,j}})$

where $\mathtt{Dir}()$ is a Dirichlet distribution and $\mathtt{Multi}()$ is a multinomial distribution with 1 trial.

To gain intuition about time varying topics, let's first consider a simplified case. We want to see which distributional assumptions are good for modelling our time varying latent variables. To do this assume we know the true value of the topic proportions for the documents at each of T time steps (assuming there is a single topic proportion at each time step). For further simplicity we only care about modelling the topic proportions (step 1 above). We will see how different modelling approaches perform on this data.

(a) First you consider only using the data at a specific time step t for training. What issues might this approach have?

Solution: Since the topic proportions at step t are known, you can fit this arbitrarily well. However, since the topic proportions vary with time, the resulting model won't be very useful at later time steps. Furthermore, in reality the topic proportions will not be known and only considering data at a single time point severely reduces the amount of data for training.

(b) Next you consider ignoring the time altogether and using the static topic model shown above on the pooled data. What issues might this approach have?

Solution: Here we are modelling a range of topic proportions with only a single distribution so the distribution won't be able to assign high likelihood to the topic proportion at each time step.

(c) Now let's assume a topic proportion at time t is randomly selected from the vicinity of the topic proportion at time $t-1$. What distribution could be used to model this? Does this seem like a better approach?

Solution: If θ_t is the topic proportion parameter at time t and we assume θ_{t+1} is sampled from a Gaussian distribution centered at θ_t we get:

$$\theta_{t+1}|\theta_t \sim N(\theta_t, \sigma^2)$$

Since this method explicitly models the time dependence of topic proportions it seems like a better approach.

(d) Moving back to the original problem without our simplifying assumptions, provide a generative process for tweets analogous to that of the static topic model for time varying topics. Recall we are assuming both topic proportions and the topics themselves (defined as their corresponding distributions over words) are changing with time. In place of a Dirichlet distribution you can use a normal distribution where the samples are then passed through a softmax function $\pi()$ in order to achieve probability distributions.

Solution: One possible solution is:

1. Choose topic proportions at time t $\alpha_t | \alpha_{t-1} \sim N(\alpha_{t-1}, \sigma^2 I)$
2. For each topic k choose topic distribution $\beta_{k,t} | \beta_{k,t-1} \sim N(\beta_{k,t-1}, \delta^2 I)$
3. For each tweet $D_{i,t}$ at time t:
   (a) Choose tweet topic proportions $\theta_{i,t} \sim N(\alpha_{t-1}, a^2 I)$
   (b) For each word $w_{i,t,j}$ in tweet $D_{i,j}$:
      i. Choose a topic $z_{i,t,j} \sim \mathrm{Multi}(\pi(\theta_{i,t}))$
      ii. Choose a word $w_{i,t,j} \sim \mathrm{Multi}(\pi(\beta_{z_{i,t,j},t}))$

This process models all time varying variables using our intuition from part c. This process assumes the mean topic proportions α_t are varying with time. When generating documents at time t, first document proportions $\theta_{i,t}$ are sampled from a Gaussian distribution centered at that time's mean topic proportions. Words are then generated in the same way as with the static topic model, however, the distributions over words, $\beta_{k,t}$, for each topic model is also time varying.