CPSC 532W Assignment 1

Ali Seyfi - 97446637

1 Question 1

Show that the Gamma distribution is conjugate to the Poisson distribution.

We have:

$$y \sim Poisson(\lambda), \lambda \sim Gamma(\alpha, \beta)$$

which means:

$$p(y|\lambda) = \frac{\lambda^y e^{-\lambda}}{y!}$$

and:

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

We know from the Bayes' rule that:

$$p(\lambda|y, \alpha, \beta) = \frac{p(y|\lambda, \alpha, \beta)p(\lambda|\alpha, \beta)}{p(y|\alpha, \beta)}$$

So:

$$p(\lambda|y, \alpha, \beta) \propto p(y|\lambda, \alpha, \beta)p(\lambda|\alpha, \beta)$$

We know from the DAG of the model and conditional independence, that:

$$P(y|\lambda, \alpha, \beta) = P(y|\lambda)$$

So we will have:

$$p(\lambda|y, \alpha, \beta) \propto p(y|\lambda)p(\lambda|\alpha, \beta)$$

By putting the distributions in the above formula we will get:

$$p(\lambda|y, \alpha, \beta) \propto \frac{\lambda^y e^{-\lambda}}{y!} \times \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda}$$

Which is equal to:

$$p(\lambda|y, \alpha, \beta) \propto \lambda^y e^{-\lambda} \times \lambda^{\alpha-1} e^{-\beta\lambda}$$

so we will have:

$$p(\lambda|y, \alpha, \beta) \propto \lambda^{y+\alpha-1} e^{-(\beta+1)\lambda}$$

Now lets define these new variables:

$$\alpha^+ = y + \alpha, \beta^+ = \beta + 1$$

Then we will have:

$$p(\lambda|y, \alpha, \beta) \propto \lambda^{\alpha^+-1} e^{-\beta^+\lambda}$$

Which is also a Gamma distribution: $Gamma(y + \alpha, \beta + 1)$.

2 Question 2

Show that the Gibbs transition operator satisfies the detailed balance equation and as such can be interpreted as an MH transition operator that always accepts.

2.1 Gibbs transition operator satisfies the detailed balance equation

Suppose that x_i is the variable that we want to sample conditioned on the value of all other variables, x_{-i}. Based on the notations on the course slides:

$$\begin{aligned} p^*(x_i, x_{-i})T((x_i, x_{-i}), (y_i, x_{-i})) &= p^*(x_i, x_{-i})P(y_i, x_{-i}) \\ &= p^*(x_i, x_{-i})\frac{p^*(y_i, x_{-i})}{\int p^*(z, x_{-i})dz} \\ &= p^*(y_i, x_{-i})\frac{p^*(x_i, x_{-i})}{\int p^*(z, x_{-i})dz} \\ &= p^*(y_i, x_{-i})P(x_i, x_{-i}) \\ &= p^*(y_i, x_{-i})T((y_i, x_{-i}), (x_i, x_{-i})). \end{aligned}$$

2.2 Interpretation as an MH transition operator that always accepts.

For MH transition, we always accept the probability of the move with:

$$A((y_i, x_{-i}), (x_i, x_{-i})) = min(1, \frac{p(y_i, x_{-i})}{p(x_i, x_{-i})}\frac{q((x_i, x_{-i})|(y_i, x_{-i}))}{q((y_i, x_{-i})|(x_i, x_{-i}))})$$

Then as we have computed earlier:

$$\frac{q((x_i, x_{-i})|(y_i, x_{-i}))}{q((y_i, x_{-i})|(x_i, x_{-i}))} = \frac{\frac{p^*(x_i, x_{-i})}{\int p^*(z, x_{-i})dz}}{\frac{p^*(y_i, x_{-i})}{\int p^*(z, x_{-i})dz}} = \frac{p(x_i, x_{-i})}{p(y_i, x_{-i})}$$

So, $A((y_i, x_{-i}), (x_i, x_{-i}))$ will be:

$$A((y_i, x_{-i}), (x_i, x_{-i})) = min(1, \frac{p(y_i, x_{-i})}{p(x_i, x_{-i})}\frac{p(x_i, x_{-i})}{p(y_i, x_{-i})}) = 1.$$

So Gibbs transition operator could be interpret as an MH transition operator that always accepts.

3 Question 3

Write code to compute the probability three ways that it is cloudy given that we observe that the grass is wet using this Bayes net model.

3.1 Part 1

By enumerating all possible world states and conditioning by counting which proportion are cloudy given the observed world characteristic, namely, that the grass is wet.

```
##1. enumeration and conditioning:

## compute joint:
p = np.zeros((2,2,2,2)) #c,s,r,w
for c in range(2):
    for s in range(2):
        for r in range(2):
            for w in range(2):
                p[c,s,r,w] = p_C(c)*p_S_given_C(s,c)*p_R_given_C(r,c)*p_W_given_S_R(w,s,r)

## condition and marginalize:
p_cw = np.zeros((2,2)) #c,w
for c in range(2):
    for w in range(2):
        p_cw[c,w] = np.sum(p[c,:,:,w])

p_w = np.sum(p_cw,0)

p_C_given_W = p_cw / p_w

p_C_given_W_True = p_C_given_W[:,1]

print('There is a {:.2f}% chance it is cloudy given the grass is wet'.format(
    p_C_given_W_True[1]*100))
```

Listing 1: Q3.py - Enumerating and conditioning

There is a 57.58% chance it is cloudy given the grass is wet.

3.2 Part 2

Using ancestral sampling and rejection.

```
##2. ancestral sampling and rejection:

num_samples = 10000
samples = np.zeros(num_samples)
rejections = 0
i = 0
while i < num_samples:
    C = np.random.choice(np.arange(2), p=[p_C(0), p_C(1)])
    S = np.random.choice(np.arange(2), p=[p_S_given_C(0,C), p_S_given_C(1,C)])
    R = np.random.choice(np.arange(2), p=[p_R_given_C(0,C), p_R_given_C(1,C)])
    W = np.random.choice(np.arange(2), p=[p_W_given_S_R(0,S,R), p_W_given_S_R(1,S,R)])

    if W == 0:
        rejections += 1
    else:
        samples[i] = C
        i += 1

```

```
print('The chance of it being cloudy given the grass is wet is {:.2f}%'.format(samples.mean()*100))
print('{:.2f}% of the total samples were rejected'.format(100*rejections/(samples.shape[0]+rejections)))
```

Listing 2: Q3.py - Ancestral Sampling and Rejection

The results vary in each run, here is one of the results:
The chance of it being cloudy given the grass is wet is 57.91%
35.30% of the total samples were rejected

3.3 Part 3

Using Gibbs sampling.

```
##gibbs sampling
num_samples = 100000
samples = np.zeros(num_samples)
state = np.zeros(4,dtype='int')
#c,s,r,w, set w = True
state[3] = 1

for i in range(100000):
    change_index = np.random.randint(0,3)

    if change_index == 0:
        change_value = np.random.choice(np.arange(2), p=[p_C_given_S_R[0,state[1], state[2]], p_C_given_S_R[1,state[1], state[2]]])
    elif change_index == 1:
        change_value = np.random.choice(np.arange(2), p=[p_S_given_C_R_W[state[0],0, state[2], state[3]], p_S_given_C_R_W[state[0],1, state[2], state[3]]])
    elif change_index == 2:
        change_value = np.random.choice(np.arange(2), p=[p_R_given_C_S_W[state[0],state[1], 0, state[3]],p_R_given_C_S_W[state[0],state[1], 1, state[3]]])
    state[change_index] = change_value
    samples[i] = state[0]


print('The chance of it being cloudy given the grass is wet is {:.2f}%'.format(samples.mean()*100))
```

Listing 3: Q3.py - Gibbs Sampling

The results vary in each run, here is one of the results:
The chance of it being cloudy given the grass is wet is 57.36%

4 Question 4

Show and derive the updates required to

4.1 Perform MH within Gibbs on the blocks w and $\hat{t}$.

4.1.1 Updating $\hat{t}$

Based on Gibbs sampling, as we want to update $\hat{t}$, so we want to go from $(W^k, \hat{t}^k)$ to $(W^k, \hat{t}^{k+1})$. To preform this update, we assume that we have the value of W^k, and so that is an observed variable.
Now it is the time to perform MH algorithm. We first propose a new value for $\hat{t}$, $\hat{t}^{k+1}$, where:

$$\hat{t}^{k+1} \sim q(\hat{t}^{k+1}|\hat{t}^k)$$

We will accept this new $\hat{t}^{k+1}$, with the probability $A(\hat{t}^{k+1}, \hat{t}^k)$, where:

$$\begin{aligned}
A(\hat{t}^{k+1}, \hat{t}^k) &= min\{1, \frac{p(\hat{t}^{k+1}|W^k, ...)}{p(\hat{t}^k|W^k, ...)} \frac{q(\hat{t}^k|\hat{t}^{k+1})}{q(\hat{t}^{k+1}|\hat{t}^k)}\} \\
&= min\{1, \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(\hat{t}^{k+1} - W^k\hat{x})^2}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(\hat{t}^k - W^k\hat{x})^2}} \frac{q(\hat{t}^k|\hat{t}^{k+1})}{q(\hat{t}^{k+1}|\hat{t}^k)}\} \\
&= min\{1, e^{-\frac{1}{2\sigma^2}[(\hat{t}^{k+1} - W^k\hat{x})^2 - (\hat{t}^k - W^k\hat{x})^2]} \times \frac{q(\hat{t}^k|\hat{t}^{k+1})}{q(\hat{t}^{k+1}|\hat{t}^k)}\} \\
&= min\{1, e^{-\frac{1}{2\sigma^2}[(\hat{t}^{k+1} - \hat{t}^k)(\hat{t}^{k+1} + \hat{t}^k - 2W^k\hat{x})]} \times \frac{q(\hat{t}^k|\hat{t}^{k+1})}{q(\hat{t}^{k+1}|\hat{t}^k)}\}
\end{aligned}$$

which if we have a symmetric proposal for q, we will end up with:

$$A(\hat{t}^{k+1}, \hat{t}^k) = min\{1, e^{-\frac{1}{2\sigma^2}[(\hat{t}^{k+1} - \hat{t}^k)(\hat{t}^{k+1} + \hat{t}^k - 2W^k\hat{x})]}\}$$

4.1.2 Updating W

Based on Gibbs sampling, as to update W, so we want to go from $(W^k, \hat{t}^{k+1})$ to $(W^{k+1}, \hat{t}^{k+1})$. To preform this update, we assume that we have the value of $\hat{t}^{k+1}$, and so that is an observed variable.
Now it is the time to perform MH algorithm. We first propose a new value for W, W^{k+1}, where:

$$W^{k+1} \sim q(W^{k+1}|W^k)$$

Also, let us define $t^{k+1} = [t; \hat{t}^{k+1}]$, and $X = [x; \hat{x}]$.
We will accept this new W^{k+1}, with the probability $A(W^{k+1}, W^k)$, where:

$$\begin{aligned}
A(W^{k+1}, W^k) &= min\{1, \frac{p(W^{k+1}|\hat{t}^{k+1}, ...)}{p(W^k|\hat{t}^{k+1}, ...)} \frac{q(W^k|W^{k+1})}{q(W^{k+1}|W^k)}\} \\
&= min\{1, \frac{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\|t^{k+1} - W^{k+1}X\|^2} e^{-\frac{1}{2\alpha}\|W^{k+1}\|^2}}{\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}\|t^{k+1} - W^kX\|^2} e^{-\frac{1}{2\alpha}\|W^k\|^2}} \frac{q(W^k|W^{k+1})}{q(W^{k+1}|W^k)}\} \\
&= min\{1, e^{-\frac{1}{2\sigma^2}[\|t^{k+1} - W^{k+1}X\|^2 - \|t^{k+1} - W^kX\|^2] - \frac{1}{2\alpha}[\|W^{k+1}\|^2 - \|W^k\|^2]} \times \frac{q(W^k|W^{k+1})}{q(W^{k+1}|W^k)}\}
\end{aligned}$$

which if we have a symmetric proposal for q, we will end up with:

$$A(W^{k+1}, W^k) = min\{1, e^{-\frac{1}{2\sigma^2}[\|t^{k+1} - W^{k+1}X\|^2 - \|t^{k+1} - W^kX\|^2] - \frac{1}{2\alpha}[\|W^{k+1}\|^2 - \|W^k\|^2]}\}$$

4.2 Perform pure Gibbs on both W and $\hat{t}$.

4.2.1 Updating $\hat{t}$

Based on Gibbs sampling, as we want to update $\hat{t}$, so we want to go from $(W^k, \hat{t}^k)$ to $(W^k, \hat{t}^{k+1})$. To preform this update, we assume that we have the value of W^k, and so that is an observed variable.
Then in order to update $\hat{t}$, we should first find $p(\hat{t}|W^k, ...)$. Based on the last part, we know that:

$$p(\hat{t}|W^k, ...) = \mathcal{N}(W^k\hat{x}, \sigma^2)$$

So in order to update the $\hat{t}$, we should sample from the mentioned Gaussian distribution:

$$\hat{t}^{k+1} \sim \mathcal{N}(W^k\hat{x}, \sigma^2)$$

4.2.2 Updating W

Based on Gibbs sampling, as to update W, so we want to go from $(W^k, \hat{t}^{k+1})$ to $(W^{k+1}, \hat{t}^{k+1})$. To preform this update, we assume that we have the value of $\hat{t}^{k+1}$, and so that is an observed variable.
Then in order to update W, we should first find $p(W|\hat{t}^{k+1}, ...)$. Let's first define $t^{k+1} = [t; \hat{t}^{k+1}]$, and $X = [x; \hat{x}]$.
Now we know that:

$$\begin{aligned} p(W|t, X, \sigma, \alpha) &\propto p(W, t, X, \sigma, \alpha) \\ &\propto p(t|W, X, \sigma)p(W|\alpha) \\ &= \prod_{i=1}^{N+1} p(t_i|W, X_i, \sigma^2)p(W|\alpha) \\ &\propto e^{-\frac{1}{2\sigma^2}\|t-WX\|^2} \times e^{-\frac{1}{2\alpha}\|W\|^2} \end{aligned}$$

And we need to complete the square, using matrix cookbook, we will reach again a Gaussian distribution:

$$p(W|t, X, \sigma, \alpha) = \mathcal{N}(\frac{1}{\sigma^2}\Sigma^{-1}Xt, \Sigma^{-1})$$

Where $\Sigma = \frac{1}{\alpha}I + \frac{1}{\sigma^2}X^TX$.

4.3 Produce the analytic form of the posterior predictive

From Bishop's book, Appendix B, page 689, we know that if we have:

$$\begin{aligned} p(x) &= \mathcal{N}(\mu, \Sigma) \\ p(y|x) &= \mathcal{N}(Ax + b, L^{-1}) \end{aligned}$$

Then we will have:

$$p(y) = \mathcal{N}(A\mu + b, L^{-1} + A\Sigma A^T)$$

Now, from last part, we have:

$$p(W|t, X, \sigma, \alpha) = \mathcal{N}(\frac{1}{\sigma^2}(\frac{1}{\alpha}I + \frac{1}{\sigma^2}X^TX)^{-1}Xt, (\frac{1}{\alpha}I + \frac{1}{\sigma^2}X^TX)^{-1})$$

And we know that:

$$p(\hat{t}|W, \hat{x}) = \mathcal{N}(W^T\hat{x}, \sigma^2)$$

Then, the posterior predictive, will be:

$$p(\hat{t}|\hat{x}, X, t, \sigma, \alpha) = \mathcal{N}(\frac{1}{\sigma^2}(\frac{1}{\alpha}I + \frac{1}{\sigma^2}X^TX)^{-1}Xt\hat{x}, \sigma^2 + \hat{x}^T(\frac{1}{\alpha}I + \frac{1}{\sigma^2}X^TX)^{-1}\hat{x})$$

5 Question 5

Implement a sampler for the LDA model on a corpus consisting of abstracts from NeurIPS in years past.

5.1 Joint Log Likelihood Function

```python
import numpy as np
from scipy.special import loggamma

def joint_log_lik(doc_counts, topic_counts, alpha, gamma):
    """
    Calculate the joint log likelihood of the model

    Args:
        doc_counts: n_docs x n_topics array of counts per document of unique topics
        topic_counts: n_topics x alphabet_size array of counts per topic of unique words
        alpha: prior dirichlet parameter on document specific distributions over topics
        gamma: prior dirichlet parameter on topic specific distribuitons over words.
    Returns:
        ll: the joint log likelihood of the model
    """
    n_docs = doc_counts.shape[0]
    n_topics = doc_counts.shape[1]
    alphabet_size = topic_counts.shape[1]

    #From wikipedia:

    first_term = n_docs*(loggamma(n_topics * alpha)-n_topics*loggamma(alpha))
    posterier_doc_counts = doc_counts + alpha
    second_term = np.sum(np.sum(loggamma(posterier_doc_counts))) - np.sum(loggamma(np.sum(
    posterier_doc_counts, axis=1)))

    third_term = n_topics*(loggamma(alphabet_size*gamma) - alphabet_size*loggamma(gamma))
    posterier_topic_counts = topic_counts + gamma
    fourth_term = np.sum(np.sum(loggamma(posterier_topic_counts))) - np.sum(loggamma(np.sum(
    posterier_topic_counts, axis=1)))

    log_like = first_term + second_term + third_term + fourth_term

    return log_like
```

Listing 4: joint_log_lik.py - Joint Log Likelihood

5.2 Sample Topic Assignment Function

```python
import numpy as np

def sample_topic_assignment(topic_assignment,
                            topic_counts,
                            doc_counts,
                            topic_N,
                            doc_N,
                            alpha,
                            gamma,
                            words,
                            document_assignment):
    """
    Sample the topic assignment for each word in the corpus, one at a time.

    Args:
        topic_assignment: size n array of topic assignments
        topic_counts: n_topics x alphabet_size array of counts per topic of unique words
        doc_counts: n_docs x n_topics array of counts per document of unique topics

        topic_N: array of size n_topics count of total words assigned to each topic
        doc_N: array of size n_docs count of total words in each document, minus 1

        alpha: prior dirichlet parameter on document specific distributions over topics
        gamma: prior dirichlet parameter on topic specific distribuitons over words.

        words: size n array of wors
        document_assignment: size n array of assignments of words to documents
    Returns:
        topic_assignment: updated topic_assignment array
        topic_counts: updated topic counts array
        doc_counts: updated doc_counts array
        topic_N: updated count of words assigned to each topic
    """

    # I get helped from this article to understand the concept and implement this part:
    # https://www.ics.uci.edu/~asuncion/pubs/KDD_08.pdf

    n_topics, vocabulary_size = topic_counts.shape

    for w, word in enumerate(words):
        topic = topic_assignment[w]
        document = document_assignment[w]

        topic_counts[topic, word] -= 1 #phi
        doc_counts[document,topic] -= 1 #theta
        topic_N[topic] -= 1

        P_topic_word = (topic_counts[:,word] + gamma)/(topic_N + vocabulary_size*gamma)
        P_document_topic = (doc_counts[document] + alpha)/(doc_N[document] + n_topics*alpha)

        P_Z = P_document_topic*P_topic_word
        P_Z /= np.sum(P_Z)

        new_topic = np.random.multinomial(1, P_Z).argmax()

        topic_assignment[w] = new_topic
        topic_counts[new_topic, word] += 1
        doc_counts[document, new_topic] += 1
        topic_N[new_topic] += 1

    return topic_assignment, topic_counts, doc_counts, topic_N
```

Listing 5: sample_topic_assignment.py - Sample Topic Assignment

5.3 Joint Log Likelihood Plot

```
alpha = 1/n_topics
gamma = 1/n_topics
iters = 1000
```

Listing 6: LDA.py - parameters

I ran the script over full dataset for 1000 iterations, with the choice of α and γ both equal to $\frac{1}{\#topics} = 0.05$, in the following sections, I have provided the results of plotting joint log likelihood value in each iteration.

5.3.1 All 1000 iterations

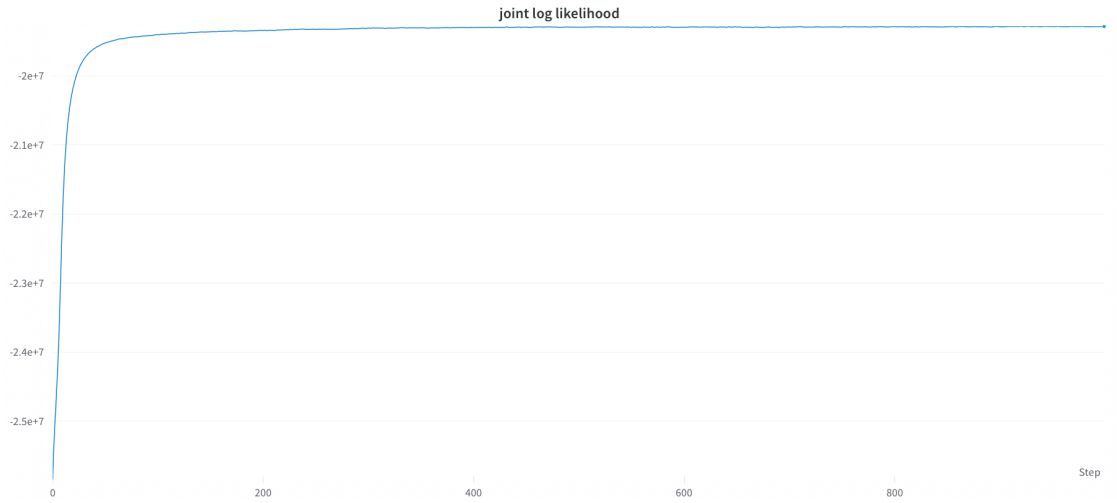

Figure 1: All 100 iterations

5.3.2 First 100 iterations

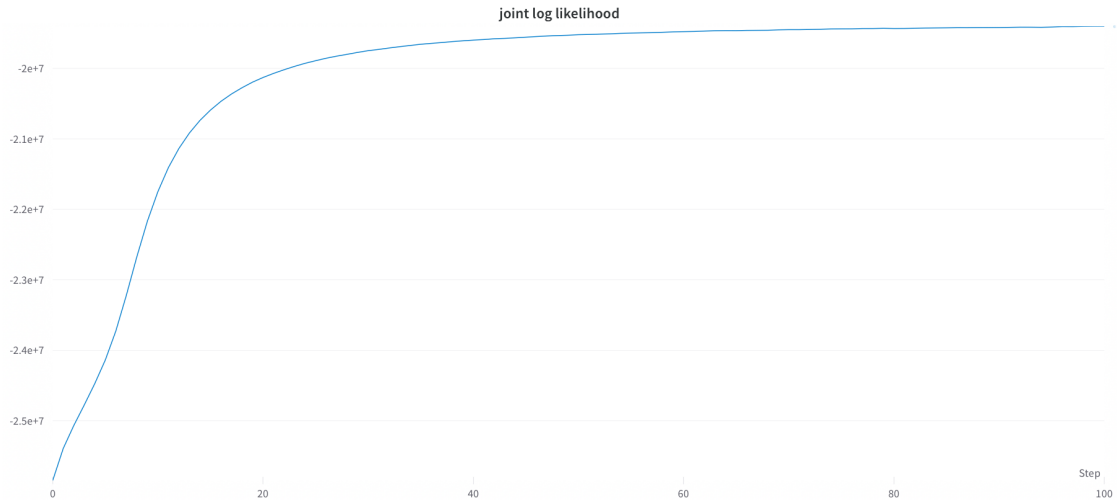

Figure 2: First 100 iterations

5.3.3 After 100 iterations

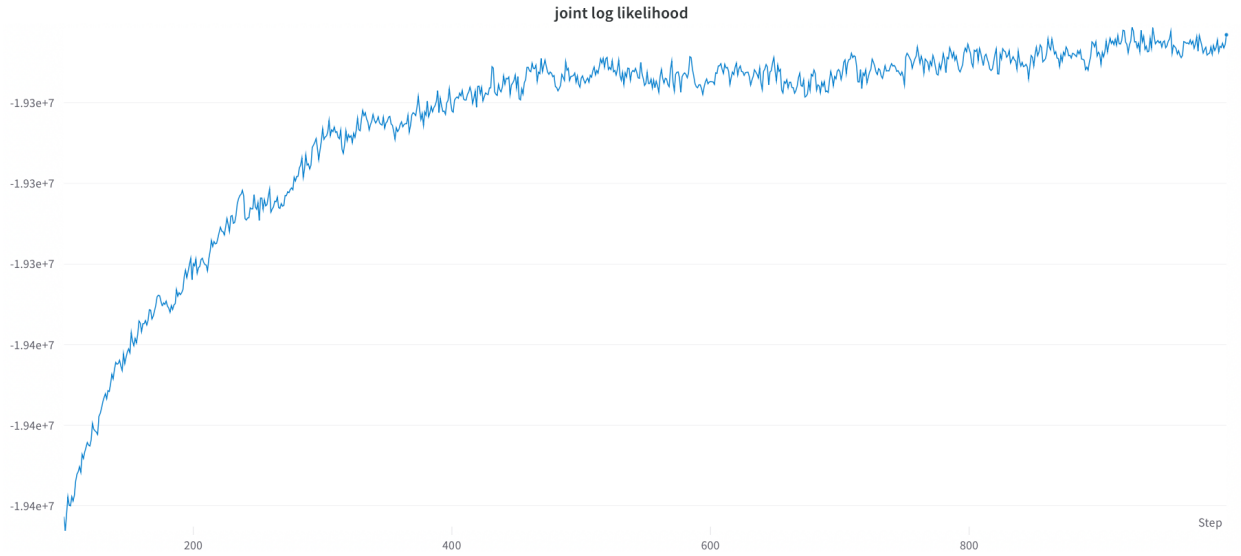

Figure 3: After 100 iterations

5.3.4 After 400 iterations

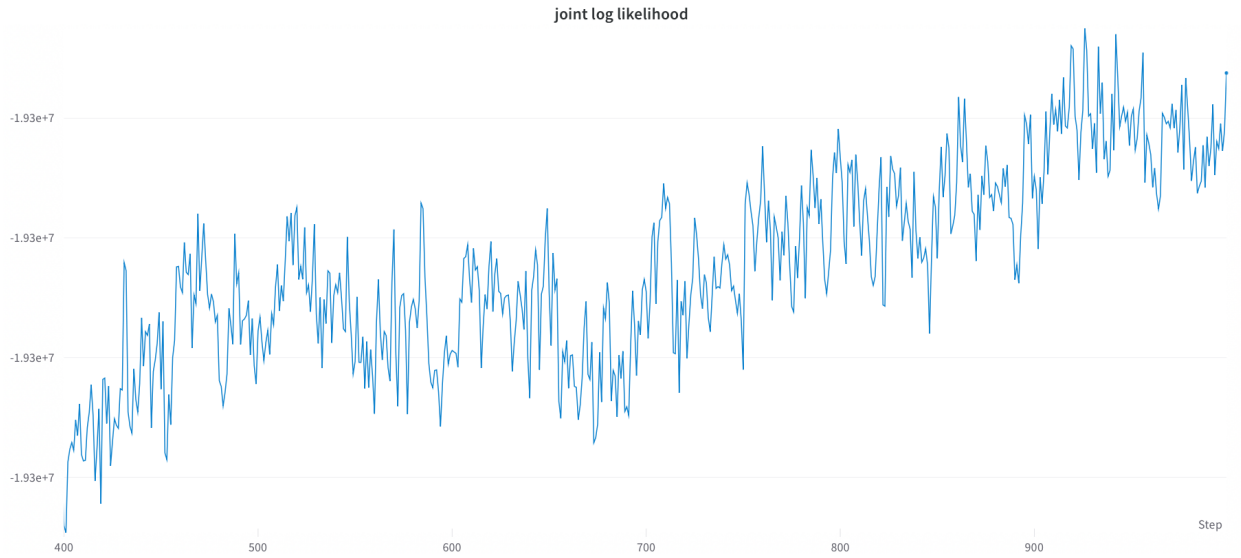

Figure 4: After 400 iterations

5.4 What are the top ten most probable words for each of the 20 topics?

```
### find the 10 most probable words of the 20 topics:
fstr = ''

for t in range(n_topics):
    most_probable = topic_counts[t,:].argsort()[-10:]
    for i in range(10):
        fstr = fstr + str(WO[most_probable[9-i]][0]) + ', '
    fstr += '\n'

with open('most_probable_words_per_topic','w') as f:
    f.write(fstr)
```

Listing 7: LDA.py - ten most probable words of each topic

Topic 1	: learning, error, weight, training, gradient, generalization, noise, function, weights, rate
Topic 2	: classification, training, classifier, class, set, recognition, pattern, feature, classifiers, error
Topic 3	: function, functions, algorithm, theorem, case, bound, set, number, class, linear
Topic 4	: time, neural, memory, system, network, number, performance, parallel, bit, vector
Topic 5	: speech, recognition, word, system, time, hmm, training, context, speaker, state
Topic 6	: node, nodes, tree, state, neural, networks, input, threshold, trees, sequence
Topic 7	: neurons, model, time, neuron, firing, cell, spike, synaptic, input, activity
Topic 8	: learning, state, time, reinforcement, action, policy, function, optimal, states, control
Topic 9	: space, algorithm, data, problem, points, local, point, function, distance, figure
Topic 10	: visual, cells, model, motion, spatial, receptive, orientation, field, cortex, figure
Topic 11	: information, matrix, signal, noise, linear, analysis, component, components, data, independent
Topic 12	: rules, rule, model, representation, representations, structure, memory, connectionist, figure, learning
Topic 13	: control, model, system, motor, position, head, figure, eye, controller, movement
Topic 14	: network, input, units, networks, output, hidden, learning, layer, neural, unit
Topic 15	: data, training, set, performance, learning, test, examples, results, number, based
Topic 16	: circuit, chip, analog, figure, output, input, voltage, current, vlsi, neural
Topic 17	: image, images, object, recognition, objects, features, visual, figure, feature, face
Topic 18	: model, data, function, neural, linear, error, regression, prediction, models, training
Topic 19	: model, data, distribution, probability, models, gaussian, parameters, likelihood, mixture, bayesian
Topic 20	: network, neural, system, state, networks, time, dynamics, neurons, neuron, model

Results of words in some topics are really interesting and show that our model has found the proper topics. For example topic 2 seems to be about pattern recognition, topic 3 about algorithms, topic 5 about NLP, topic 6 about decision trees, topic 7 about Neuroscience - cell level, topic 8 about Reinforcement Learning, topic 10 about Neuroscience - visual system, topic 11 about linear algebra, topic 13 about control systems, topic 14 about neural networks, topic 16 about Electrical Engineering, topic 19 about probability, etc.

5.5 What are the ten most similar documents to the first document?

```
#most similar documents to document 0 by cosine similarity over topic distribution:
#normalize topics per document and dot product:

doc_counts_row_sums = doc_counts.sum(axis=1)
normalized_doc_counts = doc_counts/doc_counts_row_sums[:,np.newaxis]

most_similar = 0
score = np.ones(normalized_doc_counts.shape[0])

for d in range(1,doc_counts.shape[0]):
    score[d] = np.dot(normalized_doc_counts[0,:], normalized_doc_counts[d,:])

best_scores = score.argsort()[-10:]
fstr = ''
for i in range(2,11):
    fstr += str(best_scores[10-i]) + ' with score of: ' + str(score[best_scores[10-i]]) +'\n
    '

with open('most_similar_titles_to_0','w') as f:
    f.write(fstr)
```

Listing 8: LDA.py - ten most similar documents to the first document

Notice that document indices start with 0 and we are comparing the other documents with document 0.

Ten most similar documents to the first document	
Document index	Score
539	0.322397
649	0.298438
870	0.282900
267	0.281705
1642	0.272937
1036	0.269094
1470	0.267486
1154	0.259408
57	0.258253