CPSC 532W - Homework 1

Xiaoxuan Liang – 48131163

1. Assume the random variables $\mathbf{X} = \{X_1, X_2, \cdots, X_n\}$ follow the Poisson distribution, with the probability mass function:

$$f(\mathbf{X} \mid \lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$$

Let the prior be the Gamma distribution with parameters (α, β) :

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

Then the posterior is

$$\begin{aligned} p(\lambda \mid \mathbf{X}, \alpha, \beta) &\propto f(\mathbf{X} \mid \lambda) p(\lambda \mid \alpha, \beta) \\ &= \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!} \frac{\beta^{\alpha}}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &\propto \lambda^{(\sum_{i=1}^{n} x_i + \alpha - 1)} e^{(-(\beta+n)\lambda)} \end{aligned}$$

Therefore, the posterior follows Gamma distribution with parameters $(\sum_{i=1}^{n} x_i + \alpha, \beta + n)$, so the Gamma distribution is conjugate to the Poisson distribution.

2. First, we show the Gibbs transition operator satisfies the detailed balance equation. Assume we want to obtain a sample $\mathbf{x}'$ from $\mathbf{x} = (X_1 = x_1, \cdots, X_i = c, \cdots, X_n = x_n)$ from the joint distribution $p(\mathbf{x}) = p(x_1, \cdots, x_n)$ using Gibbs sampling. Suppose we randomly select the ith variable with probability π_i, then obtain $\mathbf{x}'$ by replacing value x_i with conditional probability

$$\Pr[\, X_i = c' \mid \mathbf{x}_{-i} \,] = \Pr[\, X_i = c' \mid x_1, x_2, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n \,]$$

Therefore, the transition probability is

$$\Pr[\,\mathbf{x}' \mid \mathbf{x}\,] = \pi_i \Pr[\,X_i = c' \mid \mathbf{x}_{-i}\,]$$

and

$$\begin{aligned}
\Pr[\,\mathbf{x}' \mid \mathbf{x}\,]\, p(\mathbf{x}) &= \pi_i \Pr[\,X_i = c' \mid \mathbf{x}_{-i}\,]\, p(\mathbf{x}) \\
&= \pi_i \Pr[\,X_i = c' \mid \mathbf{x}_{-i}\,] \Pr[\,X_i = c \mid \mathbf{x}_{-i}\,]\, p(\mathbf{x}_{-i}) \\
&= \pi_i \Pr[\,X_i = c \mid \mathbf{x}_{-i}\,] \Pr[\,X_i = c' \mid \mathbf{x}_{-i}\,]\, p(\mathbf{x}_{-i}) \\
&= \pi_i \Pr[\,X_i = c \mid \mathbf{x}_{-i}\,]\, p(\mathbf{x}') \\
&= \Pr[\,\mathbf{x} \mid \mathbf{x}'\,]\, p(\mathbf{x}') \\
\Rightarrow \Pr[\,\mathbf{x}' \mid \mathbf{x}\,]\, p(\mathbf{x}) &= \Pr[\,\mathbf{x} \mid \mathbf{x}'\,]\, p(\mathbf{x}')
\end{aligned}$$

Here, we showed that the Gibbs transition operator satisfies the detailed balance equation.

To show this can be interpreted as an MH transition operator that always accepts, let $q(\mathbf{x}' \mid \mathbf{x})$ be the proposal distribution giving the probability of proposing $\mathbf{x}'$ to $\mathbf{x}$ and $\mathbf{x}'$ is differed by only one component, which is the transition probability $p(\mathbf{x}' \mid \mathbf{x})$ exactly. Since Metropolis-Hastings algorithm always accepts a proposed $\mathbf{x}'$ if

$$u \leq \frac{p(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})}$$

where u is any random number uniformly generated between 0 and 1.
We have proved that the detailed balance equation is satisfied in this case by applying Gibbs sampling, we know that

$$p(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}') = p(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x}) \Rightarrow \frac{p(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})} = 1 \Rightarrow u \leq \frac{p(\mathbf{x}')q(\mathbf{x} \mid \mathbf{x}')}{p(\mathbf{x})q(\mathbf{x}' \mid \mathbf{x})} \quad \forall u$$

Therefore, for any generated number u between 0 and 1, MH algorithm will always accept the proposed $\mathbf{x}'$.

3. (a) This part exactly performs

$$\Pr[c=1 \mid w=1] = \frac{\Pr[c=1, w=1]}{\Pr[w=1]} = \frac{\sum_s \sum_r \Pr[c=1, s, r, w=1]}{\sum_c \sum_s \sum_r \Pr[c, s, r, w=1]}$$

The code and result are:

```
## condition and marginalize:
p_C_given_W = np.ones(2)
num = 0
denom = 0

for s in range(2):
    for r in range(2):
        num += p[1, s, r, 1]

for c in range(2):
    for s in range(2):
        for r in range(2):
            denom += p[c, s, r, 1]

p_C_given_W[1] = num/denom
p_C_given_W[0] = 1 - p_C_given_W[1]

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

Figure 1: Code for enumerating all possible states

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

Figure 2: Result for enumerating all possible states

(b) The idea for coding part is every time when we sample for a state, we need to check if c is 1 or not, e.g, grass is wet or not. If $c = 0$, we add the counter for rejection case by one and keep sampling, otherwise, this sampling is effective the add the counter for number of sampling by one. The probability is approximate to

$$\Pr[c = 1 \mid w = 1] \approx \frac{\text{number of times seeing both c} = 1 \text{ and w} = 1}{\text{number of times seeing w} = 1}$$

The code and result is:

```
num_samples = 10000
samples = np.zeros(num_samples)
rejections = 0
i = 0
while i < num_samples:
    u1 = uniform(0,1)
    if (u1 >= p_C(0)) :
        c = 1
    else :
        c = 0
    u2 = uniform(0, 1)
    u3 = uniform(0, 1)
    if (u2 >= p_S_given_C(0, c)) :
        s = 1
    else :
        s = 0
    if (u3 >= p_R_given_C(0, c)) :
        r = 1
    else :
        r = 0

    u4 = uniform(0, 1)
    if (u4 >= p_W_given_S_R(0, s, r)) :
        w = 1
    else :
        w = 0
    if w == 1:
        if c == 1:
            samples[i] = 1
        i += 1
    else :
        rejections += 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)))
```

Figure 3: Code for ancestral sampling and rejection

```
The chance of it being cloudy given the grass is wet is 57.72%
34.84% of the total samples were rejected
```

Figure 4: Result for ancestral sampling and rejection

(c) The idea for coding this part is, I fixed w = 1 as grass is wet, then I randomized the order for which component I should sample for based on the conditional probability on the rest of components. The probability is approximate to how many times c = 1 is seen from the whole program.

The code and result is :

```
##gibbs sampling
num_samples = 10000
samples = np.zeros(num_samples)
state = np.zeros(4,dtype='int')
#c,s,r,w, set w = True
state[3] = 1
i = 0
while i < num_samples:
    u = randint(3, size = 1)
    if u == 0:
        u1 = uniform(0, 1)
        if u1 >= p_C_given_S_R[0, state[1], state[2]]:
            state[0] = 1
            samples[i] = 1

    elif u == 1:
        u1 = uniform(0, 1)
        if u1 >= p_S_given_C_R_W[0, state[0], state[2], state[3]]:
            state[1] = 1
            if state[0] == 1:
                samples[i] = 1

    elif u == 2:
        u1 = uniform(0, 1)
        if u1 >= p_R_given_C_S_W[0, state[0], state[1], state[3]]:
            state[2] = 1
            if state[0] == 1:
                samples[i] = 1

    i += 1

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

Figure 5: Code for Gibbs sampling

```
The chance of it being cloudy given the grass is wet is 55.09%
```

Figure 6: Result for Gibbs sampling

4. (a)

- Perform MH within Gibbs on the blocks $\mathbf{w}$: Let the proposal distribution be $q(\cdot)$, then if any generated number u from uniform distribution Unif$(0,1)$ satifies

$$u \leq \frac{p(\mathbf{w}' \mid \mathbf{x}, \mathbf{t}, \sigma^2, \alpha) q(\mathbf{w} \mid \mathbf{w}')}{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}, \sigma^2, \alpha) q(\mathbf{w}' \mid \mathbf{w})}$$

we accept the proposed $\mathbf{w}'$ going from $\mathbf{w}$, and

$$\begin{aligned}
r &= \frac{p(\mathbf{w}' \mid \mathbf{x}, \mathbf{t}, \sigma^2, \alpha) q(\mathbf{w} \mid \mathbf{w}')}{p(\mathbf{w} \mid \mathbf{x}, \mathbf{t}, \sigma^2, \alpha) q(\mathbf{w}' \mid \mathbf{w})} \\
&= \frac{p(\mathbf{t}, \mathbf{x}, \sigma^2, \mathbf{w}', \alpha) q(\mathbf{w} \mid \mathbf{w}')}{p(\mathbf{t}, \mathbf{x}, \sigma^2, \mathbf{w}, \alpha) q(\mathbf{w}' \mid \mathbf{w})} \\
&= \frac{\prod_{n=1}^{N} p(t_n \mid x_n, \sigma^2, \mathbf{w}') p(\mathbf{w}' \mid \alpha) q(\mathbf{w} \mid \mathbf{w}')}{\prod_{n=1}^{N} p(t_n \mid x_n, \sigma^2, \mathbf{w}) p(\mathbf{w} \mid \alpha) q(\mathbf{w}' \mid \mathbf{w})} \\
&= \frac{\prod_{n=1}^{N} \exp\left(-\frac{(t_n - \mathbf{w}'^{\top} x_n)^2}{2\sigma^2}\right) \exp\left(-\frac{1}{2} \mathbf{w}'^{\top} \left(\alpha \mathbf{I}_{d \times d}\right)^{-1} \mathbf{w}'\right) q(\mathbf{w} \mid \mathbf{w}')}{\prod_{n=1}^{N} \exp\left(-\frac{(t_n - \mathbf{w}^{\top} x_n)^2}{2\sigma^2}\right) \exp\left(-\frac{1}{2} \mathbf{w}^{\top} \left(\alpha \mathbf{I}_{d \times d}\right)^{-1} \mathbf{w}\right) q(\mathbf{w}' \mid \mathbf{w})} \\
&= \frac{\exp\left(-\frac{1}{2} (\mathbf{t} - \mathbf{x}\mathbf{w}')^{\top} (\sigma^2 \mathbf{I}_{N \times N})^{-1} (\mathbf{t} - \mathbf{x}\mathbf{w}') - \frac{1}{2} \mathbf{w}'^{\top} \left(\alpha \mathbf{I}_{d \times d}\right)^{-1} \mathbf{w}'\right) q(\mathbf{w} \mid \mathbf{w}')}{\exp\left(-\frac{1}{2} (\mathbf{t} - \mathbf{x}\mathbf{w})^{\top} \left(\alpha \mathbf{I}_{N \times N}\right)^{-1} (\mathbf{t} - \mathbf{x}\mathbf{w}) - \frac{1}{2} \mathbf{w}^{\top} \left(\alpha \mathbf{I}_{d \times d}\right)^{-1} \mathbf{w}\right) q(\mathbf{w}' \mid \mathbf{w})}
\end{aligned}$$

If q is normally-distributed, then $q(\cdot \mid \cdot)$ can be eliminated from both numerator and denominator.

- Perform MH within Gibbs on t: Let the proposal distribution be $q(\cdot)$, the if any generated number u from the uniform distribution Unif$(0,1)$ satisfies

$$u \leq \frac{p(\hat{t}' \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t} \mid \hat{t}')}{p(\hat{t} \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t}' \mid \hat{t})}$$

we accept the proposed $\hat{\mathbf{t}}'$ going from $\hat{\mathbf{t}}$, and

$$\begin{aligned}
r &= \frac{p(\hat{t}' \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t} \mid \hat{t}')}{p(\hat{t} \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t}' \mid \hat{t})} \\
&= \frac{p(\hat{t}', \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t} \mid \hat{t}')}{p(\hat{t}, \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t}' \mid \hat{t})} \\
&= \frac{p(\hat{t}' \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t} \mid \hat{t}')}{p(\hat{t} \mid \mathbf{w}, \hat{x}, \sigma^2, \alpha) q(\hat{t}' \mid \hat{t})} \\
&= \frac{\exp\left(-\frac{1}{2\sigma^2} (\hat{t}' - \mathbf{w}^{\top} \hat{x})^2\right) q(t \mid \hat{t}')}{\exp\left(-\frac{1}{2\sigma^2} (\hat{t} - \mathbf{w}^{\top} \hat{x})^2\right) q(\hat{t}' \mid \hat{t})}
\end{aligned}$$

If q is normally-distributed, then $q(\cdot \mid \cdot)$ can be eliminated from both numerator and denominator.

(b) Perform pure Gibbs on $\mathbf{w}$: we sample $\mathbf{w}'$ by sampling for the kth component in $\mathbf{w}$:

$$\begin{aligned}
p(\mathbf{w}' \mid \mathbf{w}) &= p(w'_k \mid \mathbf{w}_{-k}) \\
&\propto p(w'_k, \mathbf{w}_{-k}) \\
&= p(\mathbf{w}') \\
&= p(\mathbf{w}' \mid \mathbf{t}, \mathbf{x}, \sigma^2, \alpha) \\
&\propto \prod_{n=1}^{N} p(t_n \mid \mathbf{w}', x_n, \sigma^2) p(\mathbf{w}' \mid \alpha) \\
&\propto \exp\left(-\frac{1}{2}(\mathbf{t} - \mathbf{x}\mathbf{w}')^\top (\sigma^2 \mathbf{I}_{N\times N})^{-1}(\mathbf{t} - \mathbf{x}\mathbf{w}')\right) \exp\left(-\frac{1}{2}\mathbf{w}'^\top (\alpha \mathbf{I}_{d\times d})^{-1}\mathbf{w}'\right)
\end{aligned}$$

By 4.4.1 theorem from Murphy's book, this follows multivariate normal distribution with parameters

$$\boldsymbol{\Sigma}^{-1} = (\alpha \mathbf{I}_{d\times d})^{-1} + \mathbf{x}^\top (\sigma^2 \mathbf{I}_{N\times N})^{-1}\mathbf{x}, \qquad \boldsymbol{\mu} = \boldsymbol{\Sigma}(\mathbf{x}^\top (\sigma^2 \mathbf{I}_{N\times N})^{-1}\mathbf{t})$$

Perform pure Gibbs on $\hat{t}$: we sample $\hat{\mathbf{t}}'$ by sampling of the kth component in $\hat{t}$:

$$\begin{aligned}
p(\hat{t}' \mid \hat{t}) &= p(\hat{t}'_k \mid \hat{t}) \\
&\propto p(\hat{t}') \\
&= p(\hat{t}' \mid \mathbf{w}, \hat{x}, \sigma^2) \\
&\propto \exp\left(-\frac{1}{2\sigma^2}(\hat{t}' - \mathbf{w}^\top \hat{x})^2\right)
\end{aligned}$$

This also follows normal distribution with parameters $N \sim (\mathbf{w}^\top \hat{x}, \sigma^2)$.

(c) The posterior distribution is

$$\begin{aligned}
p(\hat{t} \mid \mathbf{t}, \hat{x}, \mathbf{x}, \sigma^2, \alpha) &= \int p(\mathbf{w} \mid \mathbf{t}, \mathbf{x}, \sigma^2, \alpha)\, p(\hat{t} \mid \hat{x}, \mathbf{w}, \sigma^2)\, d\mathbf{w} \\
&\propto \int \exp\left(-\frac{1}{2}(\mathbf{w} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{w} - \boldsymbol{\mu})\right) \exp\left(-\frac{1}{2\sigma^2}(\hat{t}' - \mathbf{w}^\top \hat{x})^2\right) d\mathbf{w}
\end{aligned}$$

This is a convolution of two normal distributions, which still follows the normal distribution.

5. First, the notations should be introduced:

$$
\begin{aligned}
K &: \text{number of topics} \\
N &: \text{number of words in the vocabulary} \\
M &: \text{number of documents} \\
\alpha, \gamma &: \text{parameters for prior} \\
\boldsymbol{\theta}_{d=1,\cdots,D} &: \text{distribution over K topics of document d,} \quad \theta_d \sim Dir_K(\alpha \mathbf{1}) \\
\boldsymbol{\beta}_{k=1,\cdots,K} &: \text{word distribution for topic k,} \quad \beta_k \sim Dir_M(\gamma \mathbf{1}) \\
z_i^d &: \text{topic assigned for ith word in document d} \\
w_i^d &: \text{observed ith word in document d}
\end{aligned}
$$

Then the joint likelihood is

$$
\Pr\left[\, \mathbf{W}, \mathbf{Z}, \boldsymbol{\theta}, \boldsymbol{\beta} \mid \alpha, \gamma \,\right] \propto \prod_{d=1}^{D} \Pr\left[\, \theta_d \mid \alpha \,\right] \prod_{k=1}^{K} \Pr\left[\, \beta_k \mid \gamma \,\right] \prod_{i=1}^{N} \Pr\left[\, Z_i^d \mid \theta_d \,\right] \Pr\left[\, W_i^d \mid \beta_k, Z_i^d \,\right]
$$

Then we can integrate out $\boldsymbol{\theta}, \boldsymbol{\beta}$ with

$$
\Pr\left[\, \mathbf{Z}, \mathbf{W} \mid \alpha, \gamma \,\right] = \prod_{d=1}^{D} \frac{\Gamma(\sum_{k=1}^{K} \alpha)}{\prod_{k=1}^{K} \Gamma(\alpha)} \frac{\prod_{k=1}^{K} \Gamma(n_{d,(\cdot)}^{k} + \alpha)}{\Gamma(\sum_{k=1}^{K} n_{d,(\cdot)}^{k} + \alpha)} \times \prod_{k=1}^{K} \frac{\Gamma(\sum_{i=1}^{N} \gamma)}{\prod_{i=1}^{N} \Gamma(\gamma)} \frac{\prod_{i=1}^{N} \Gamma(n_{(\cdot),i}^{k} + \gamma)}{\Gamma(\sum_{i=1}^{N} n_{(\cdot),i}^{k} + \gamma)}
$$

where $n_{d,i}^{k}$ is the number of word token in the dth document with the same word symbol (the ith word in the vocabulary) assigned to kth topic, so $n_{d,(\cdot)}^{k}$ is the number of word token in the dth document assigned to kth topic, and $n_{(\cdot),i}^{k}$ is the number of word token with the same word symbol (the ith word in the vocabulary) assigned to the kth topic. Then the joint log-likelihood is

$$
\begin{aligned}
\log \Pr\left[\, \mathbf{Z}, \mathbf{W} \mid \alpha, \gamma \,\right] = & \sum_{d=1}^{D} \left(\log \Gamma(\sum_{k=1}^{K} \alpha) + \sum_{k=1}^{K} \log \Gamma(n_{d,(\cdot)}^{k} + \alpha) - \sum_{k=1}^{K} \log \Gamma(\alpha) - \log \Gamma\left(\sum_{k=1}^{K} n_{d,(\cdot)}^{k} + \alpha \right) \right) \\
& + \sum_{k=1}^{K} \left(\log \Gamma(\sum_{i=1}^{N} \gamma) + \sum_{i=1}^{N} \log \Gamma(n_{(\cdot),i}^{k} + \gamma). - \sum_{i=1}^{N} \log \Gamma(\gamma) - \log \Gamma\left(\sum_{i=1}^{N} n_{(\cdot),i}^{k} + \gamma \right) \right)
\end{aligned}
$$

For topic assignment sampling, we have

$$
\Pr\left[\, z_i^d = k \mid \mathbf{z}_{-i}^{-d}, \mathbf{W}, \alpha, \gamma \,\right] \propto \frac{n_{d,(\cdot)}^{k} + \alpha}{\sum_{k=1}^{K} n_{d,(\cdot)}^{k} + \alpha} \times \frac{n_{(\cdot),i}^{k} + \gamma}{\sum_{i=1}^{N} n_{(\cdot),i}^{k} + \gamma}
$$

The normalization term Z is the summation of the above probability over $k = 1, 2, \cdots, K$

$$
Z = \sum_{k=1}^{K} \frac{n_{d,(\cdot)}^{k} + \alpha}{\sum_{k=1}^{K} n_{d,(\cdot)}^{k} + \alpha} \times \frac{n_{(\cdot),i}^{k} + \gamma}{\sum_{i=1}^{N} n_{(\cdot),i}^{k} + \gamma}
$$

Codes:

```
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 distribution over words.
    Returns:
        ll: the joint log likelihood of the model
    """
    n_topics = np.shape(doc_counts)[1]
    n_docs = np.shape(doc_counts)[0]
    alphabet_size = np.shape(topic_counts)[1]

    ll = 0

    for k in range(n_topics):
        ll = ll + loggamma(alphabet_size * gamma) + np.sum(loggamma(topic_counts[k] + gamma))
        ll = ll - alphabet_size * loggamma(gamma) - loggamma(np.sum(topic_counts[k] + gamma))

    for d in range(n_docs):
        ll = ll + loggamma(n_topics * alpha) + np.sum(loggamma(doc_counts[d] + alpha))
        ll = ll - n_topics * loggamma(alpha) - loggamma(np.sum(doc_counts[d] + alpha))

    return ll
```

Figure 7: Code for joint log lik

```
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 distribuitions over words.
        words: size n array of words
        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
    """
```

```
n_docs = np.shape(doc_counts)[0]

for d in range(n_docs):
    z_d = topic_assignment[document_assignment == d]
    w_d = words[document_assignment == d]

    for i in range(int(doc_N[d])):
        z_di = z_d[i]
        topic_counts[z_di][w_d[i]] -= 1
        topic_N[z_di] -= 1
        doc_counts[d][z_di] -= 1

        p = (doc_counts[d] + alpha) / np.sum(doc_counts[d] + alpha) * (topic_counts[:, w_d[i]] + gamma) / (
                    np.sum(topic_counts, axis=1) + gamma)
        p = p / np.sum(p)
        topic = np.argmax(np.random.multinomial(1, p))
        z_d[i] = topic

        topic_counts[topic][w_d[i]] += 1
        topic_N[topic] += 1
        doc_counts[d][topic] += 1

    topic_assignment[document_assignment == d] = z_d

return topic_assignment, topic_counts, doc_counts, topic_N
```

Figure 8: Code for sample˙topic˙assignment

```
fstr1 = ''
for k in range(np.shape(topic_counts)[0]):
    ind = np.argsort(topic_counts[k])[-10:]
    ind = np.flip(ind)
    fstr1 += 'topic %s :' %k
    for i in ind:
        fstr1 += '%s, ' %WO[i][0]


with open('most_probable_words_per_topic','w') as f:
    f.write(fstr1)



#most similar documents to document 0 by cosine similarity over topic distribution:
#normalize topics per document and dot product:
sim = doc_counts @ doc_counts[0].T / (norm(doc_counts[0]) * norm(doc_counts, axis = 1))
ind = np.argsort(sim)[-10:]
ind = np.flip(ind)
fstr2 = ''
for i in ind:
    fstr2 += '%s, ' %titles[i][0]

with open('most_similar_titles_to_0','w') as f:
    f.write(fstr2)
```

Figure 9: Code for LDA

I iterate all of the words 5000 times, and the first 1000 iterations are burning-out.

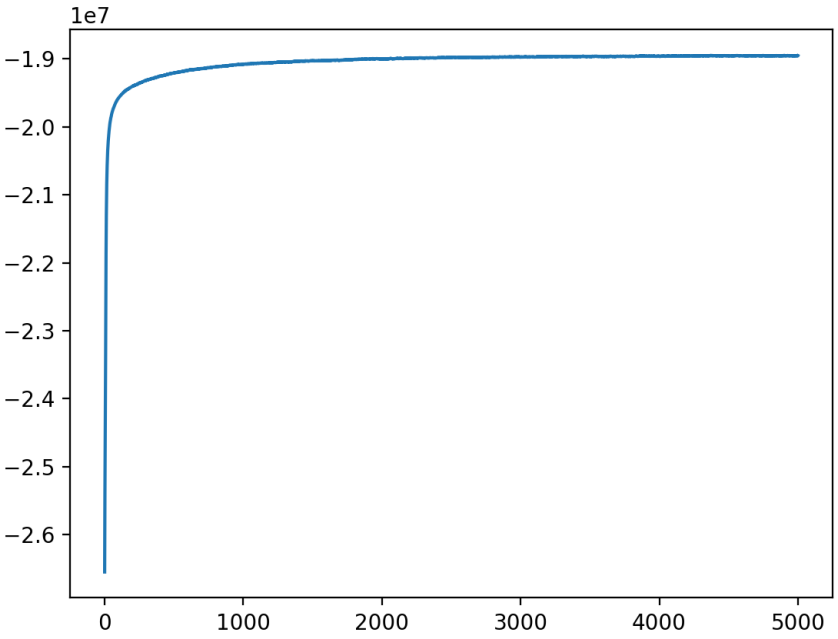

Figure 10: Without burning

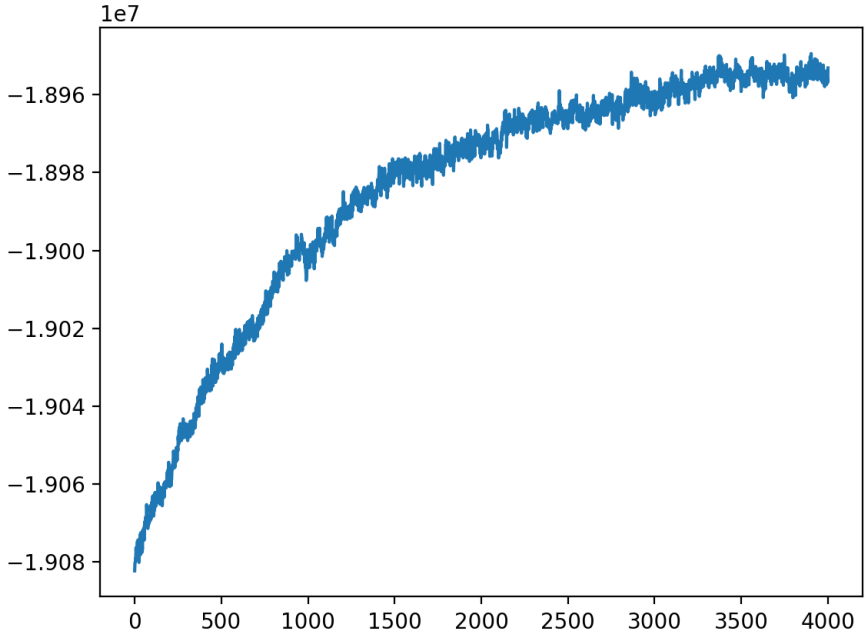

Figure 11: with burning

The top ten most probable words for each of the 20 topics:

```
topic 0 :neurons, neuron, cell, model, synaptic, firing, spike, cells, input, activity,
topic 1 :analog, circuit, chip, neural, figure, input, output, current, voltage, vlsi,
topic 2 :image, images, object, objects, recognition, figure, features, feature, visual, model,
topic 3 :learning, algorithm, time, algorithms, problem, search, problems, performance, function, step,
topic 4 :performance, human, task, data, system, face, target, subjects, detection, similarity,
topic 5 :number, results, values, order, high, large, single, set, small, level,
topic 6 :vector, space, data, vectors, matrix, distance, points, dimensional, linear, algorithm,
topic 7 :visual, model, cells, motion, field, direction, figure, eye, orientation, receptive,
topic 8 :speech, recognition, word, system, training, hmm, time, words, context, neural,
topic 9 :network, input, units, networks, output, training, hidden, neural, layer, weights,
topic 10 :learning, function, error, gradient, order, weight, equation, eq, optimal, energy,
topic 11 :function, functions, theorem, bound, number, case, threshold, neural, networks, loss,
topic 12 :data, model, function, error, prediction, training, neural, regression, linear, set,
topic 13 :information, figure, work, paper, point, process, research, general, note, tion,
topic 14 :signal, noise, time, frequency, filter, signals, information, auditory, source, channel,
topic 15 :training, set, classification, class, data, error, classifier, test, performance, examples,
topic 16 :network, time, neural, state, memory, system, networks, model, neurons, dynamics,
topic 17 :model, data, distribution, probability, models, gaussian, parameters, likelihood, mixture, log,
topic 18 :state, control, learning, model, time, reinforcement, action, policy, system, optimal,
topic 19 :node, tree, rules, structure, nodes, rule, set, state, sequence, representation,
```

Figure 12: top ten most probable words for each topic

The top ten similar titles with document 0:

```
'Connectivity Versus Entropy ,
On Properties of Networks of Neuron-Like Elements ,
On the Effect of Analog Noise in Discrete-Time Analog Computations, ,
Observability of Neural Network Behavior ,
Complexity of Finite Precision Neural Network Classifier ,
Rational Parameterizations of Neural Networks ,
Noisy Neural Networks and Generalizations,,
On The Circuit Complexity of Neural Networks . ,
Remarks on Interpolation and Recognition Using Neural Nets . ,
On Neural Networks with Minimal Weights , '
```

Figure 13: top ten most similar titles