Homework 1

Kevin Yang - 50244152

September 30, 2021

Exercise 1

Let $X = (x_1, x_2, \dots, \boldsymbol{x}_n)$
$x_i \sim Pois(\lambda)$ $\qquad\qquad\qquad$ $\lambda \sim \Gamma(\alpha, \beta)$

We have that:

$$\begin{aligned} P(\lambda \mid X) &= \frac{P(X \mid \lambda)P(\lambda)}{P(X)} \\ &\propto P(X \mid \lambda)P(\lambda) \\ &\propto \lambda^{n\bar{x}} e^{-n\lambda} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &= \lambda^{n\bar{x}+\alpha-1} e^{-(n+\beta)\lambda} \end{aligned}$$

Therefore, $P(\lambda \mid X) \sim \Gamma(n\bar{x} + \alpha, \beta + n)$ and the Gamma distribution is conjugate to the Poisson distribution.

Exercise 2

Let $s = (s_1, s_2, \dots, s_n)$, $s' = (s'_1, s'_2, \dots, s'_n)$ and $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_n)$
We will show that Gibbs sampling satisfies the detailed balance equation: $\pi(s)P(s, s') = \pi(s')P(s', s)$

w.l.g consider an update for s_1

Case 1: $s_{-1} \neq s'_{-1}$

$$\pi(s)P(s, s') = \pi(s')P(s', s) = 0$$

Case 2: $s_{-1} = s'_{-1}$

$$\begin{aligned}
\pi(s)P(s, s') &= \pi(s)P(s'_1 \mid s'_{-1}) \\
&= \pi(s)\frac{\pi(s')}{\sum_z \pi(z, s'_{-1})} \\
&= \pi(s')\frac{\pi(s)}{\sum_z \pi(z, s_{-1})} \\
&= \pi(s')P(s_1 \mid s_{-1}) \\
&= \pi(s')P(s', s)
\end{aligned}$$

Consider the move from s to s'. The acceptance probability for MH will be

$$\frac{\pi(s')P(s', s)}{\pi(s)P(s, s')} = 1$$

Exercise 3

a)

```
##1. enumeration and conditioning:


## condition and marginalize:
## 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)


p_C_given_W = np.zeros(2)
for c in range(2):
    for s in range(2):
        for r in range(2):
            p_C_given_W[c] += p[c, s, r, 1]

p_C_given_W /= np.sum(p_C_given_W)

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

b)

```
##2. ancestral sampling and rejection:
# https://www.cs.ubc.ca/~fwood/CS532W-539W/lectures/mcmc.pdf

num_samples = 10000
samples = np.zeros(num_samples)
rejections = 0
i = 0
while i < num_samples:
    c = np.argmax(np.random.multinomial(1, [p_C(0), p_C(1)]))
    s = np.argmax(np.random.multinomial(1, [p_S_given_C(0, c), p_S_given_C(1, c)]))
    r = np.argmax(np.random.multinomial(1, [p_R_given_C(0, c), p_R_given_C(1, c)]))
    w = np.argmax(np.random.multinomial(1, [p_W_given_S_R(0, s, r), p_W_given_S_R(1, s, r)]))
    if w != 1:
        rejections += 1
        continue
    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)))
```

c)

```python
##gibbs sampling
num_samples = 10000
samples = np.zeros(num_samples)
state = np.zeros(4, dtype='int')
# c,s,r,w, set w = True

c, s, r, w = 0, 1, 2, 3
i = 0
state[w] = 1
while i < num_samples:
    state[c] = np.argmax(np.random.multinomial(1, p_C_given_S_R[:, state[s], state[r]]))
    state[s] = np.argmax(np.random.multinomial(1, p_S_given_C_R_W[state[c], :, state[r], state[w]]))
    state[r] = np.argmax(np.random.multinomial(1, p_R_given_C_S_W[state[c], state[s], :, state[w]]))

    samples[i] = state[c]
    i += 1


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

Results:

```
There is a 57.58% chance it is cloudy given the grass is wet
The chance of it being cloudy given the grass is wet is 58.05%
34.73% of the total samples were rejected
The chance of it being cloudy given the grass is wet is 58.91%

Process finished with exit code 0
```

Exercise 4

MH within Gibbs on blocks w and $\hat{t}$:

$\boldsymbol{w}$ block

Let $q(\boldsymbol{w}, \boldsymbol{w}')$ be the proposal distribution. We also have that,

$$\begin{aligned}\pi(\boldsymbol{w}) &\propto p(\boldsymbol{t}, \boldsymbol{x}, \sigma^2, \boldsymbol{w}, \alpha) \\ &= \prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t_n - \boldsymbol{w}^T x_n}{\sigma})^2} \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\alpha}} e^{-\frac{1}{2}(\frac{w_d}{\sqrt{\alpha}})^2}\end{aligned}$$

So the update probability is,

$$\begin{aligned}r &= \frac{\pi(\boldsymbol{w}')q(\boldsymbol{w}', \boldsymbol{w})}{\pi(\boldsymbol{w})q(\boldsymbol{w}, \boldsymbol{w}')} \\ &= \frac{\prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t_n - \boldsymbol{w}'^T x_n}{\sigma})^2} \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\alpha}} e^{-\frac{1}{2}(\frac{w'_d}{\sqrt{\alpha}})^2} q(\boldsymbol{w}', \boldsymbol{w})}{\prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{t_n - \boldsymbol{w}^T x_n}{\sigma})^2} \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi\alpha}} e^{-\frac{1}{2}(\frac{w_d}{\sqrt{\alpha}})^2} q(\boldsymbol{w}, \boldsymbol{w}')} \\ &\propto \frac{\prod_{n=1}^{N} e^{-\frac{1}{2}(\frac{t_n - \boldsymbol{w}'^T x_n}{\sigma})^2} e^{-\frac{1}{2}(\boldsymbol{w}'^T(\alpha I)^{-1}\boldsymbol{w}')} q(\boldsymbol{w}', \boldsymbol{w})}{\prod_{n=1}^{N} e^{-\frac{1}{2}(\frac{t_n - \boldsymbol{w}^T x_n}{\sigma})^2} e^{-\frac{1}{2}(\boldsymbol{w}^T(\alpha I)^{-1}\boldsymbol{w})} q(\boldsymbol{w}, \boldsymbol{w}')} \\ &= \frac{e^{-\frac{1}{2}(\boldsymbol{t} - \boldsymbol{x}\boldsymbol{w}')^T(\sigma^2 I)^{-1}(\boldsymbol{t} - \boldsymbol{x}\boldsymbol{w}') - \boldsymbol{w}'^T(\alpha I)^{-1}\boldsymbol{w}'} q(\boldsymbol{w} \mid \boldsymbol{w}')}{e^{-\frac{1}{2}(\boldsymbol{t} - \boldsymbol{x}\boldsymbol{w})^T(\alpha I)^{-1}(\boldsymbol{t} - \boldsymbol{x}\boldsymbol{w}) - \boldsymbol{w}^T(\alpha I)^{-1}\boldsymbol{w}} q(\boldsymbol{w}' \mid \boldsymbol{w})}\end{aligned}$$

$\hat{\boldsymbol{t}}$ block

Let $q(\boldsymbol{w}, \boldsymbol{w}')$ be the proposal distribution.

$$\pi(\hat{\boldsymbol{t}}) \propto N(\boldsymbol{w}^T\hat{\boldsymbol{x}}, \sigma^2)$$

The update probability is,

$$\begin{aligned}r &= \frac{\pi(\hat{\boldsymbol{t}}')q(\hat{\boldsymbol{t}}', \hat{\boldsymbol{t}})}{\pi(\hat{\boldsymbol{t}})q(\hat{\boldsymbol{t}}, \hat{\boldsymbol{t}}')} \\ &\propto \frac{e^{-\frac{1}{2}(\frac{\hat{\boldsymbol{t}}' - \boldsymbol{w}^T\hat{\boldsymbol{x}}}{\sigma})^2} q(\hat{\boldsymbol{t}}', \hat{\boldsymbol{t}})}{e^{-\frac{1}{2}(\frac{\hat{\boldsymbol{t}} - \boldsymbol{w}^T\hat{\boldsymbol{x}}}{\sigma})^2} q(\hat{\boldsymbol{t}}, \hat{\boldsymbol{t}}')}\end{aligned}$$

Pure Gibbs on blocks w and $\hat{t}$:

<u>w block</u>

$$\begin{aligned}
\log p(\boldsymbol{w} \mid \boldsymbol{t}, \boldsymbol{x}, \sigma^2 \alpha) &\propto \log p(\boldsymbol{w}, \boldsymbol{t}, \boldsymbol{x}, \sigma^2, \alpha) \\
&= \log \prod_{n=1}^{N} p(\boldsymbol{t}_n \mid \boldsymbol{w}, \boldsymbol{x}_n, \sigma^2) p(\boldsymbol{w} \mid 0, \alpha \boldsymbol{I}) \\
&\propto \frac{-\sum_{n=1}^{N} (\boldsymbol{t}_n - \boldsymbol{w}^T \boldsymbol{x}_n)^2}{2\sigma^2} - \frac{1}{2} \boldsymbol{w}^T (\alpha \boldsymbol{I})^{-1} \boldsymbol{w} \\
&\propto -\frac{1}{2\sigma^2} (\|\boldsymbol{t} - \boldsymbol{x}^T \boldsymbol{w}\|^2) - \frac{1}{2} \boldsymbol{w}^T (\alpha \boldsymbol{I})^{-1} \boldsymbol{w} \\
&= -\frac{1}{2\sigma^2} (\boldsymbol{t}^T \boldsymbol{t} - 2\boldsymbol{t}^T \boldsymbol{x}^T \boldsymbol{w} + \boldsymbol{w}^T \boldsymbol{x} \boldsymbol{x}^T \boldsymbol{w}) - \frac{1}{2} \boldsymbol{w}^T (\alpha \boldsymbol{I})^{-1} \boldsymbol{w} \\
&= -\frac{1}{2} (\boldsymbol{w} - \boldsymbol{\mu}) \Sigma^{-1} (\boldsymbol{w} - \boldsymbol{\mu}) \sim N(\boldsymbol{\mu}, \Sigma)
\end{aligned}$$

$$\boldsymbol{\mu} = \sigma^{-2} \Sigma \boldsymbol{x}^T \boldsymbol{t} \qquad\qquad \Sigma^{-1} = \sigma^{-2} \boldsymbol{x} \boldsymbol{x}^T + (\alpha \boldsymbol{I})^{-1}$$

<u>$\hat{\boldsymbol{t}}$ block</u>

$$p(\hat{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{x}, \sigma^2, \boldsymbol{w}, \alpha) \propto e^{-\frac{1}{2}\left(\frac{\hat{t} - \boldsymbol{w}^T \hat{\boldsymbol{x}}}{\sigma}\right)^2} \sim Normal(\boldsymbol{w}^T \hat{\boldsymbol{x}}, \sigma^2)$$

Posterior Predictive:

$$\begin{aligned}
p(\hat{\boldsymbol{t}} \mid \hat{\boldsymbol{x}}, \boldsymbol{t}) &= \int p(\hat{\boldsymbol{t}} \mid \hat{\boldsymbol{x}}, \boldsymbol{w}) p(\boldsymbol{w} \mid \boldsymbol{t}) dw \\
&= \int N(\boldsymbol{w}^T \hat{\boldsymbol{x}}, \sigma^2) N(\boldsymbol{w} \mid \boldsymbol{\mu}, \Sigma) dw
\end{aligned}$$

Bye the linear combination rules for Gaussian random variables, we have that,

$$p(\hat{\boldsymbol{t}} \mid \hat{\boldsymbol{x}}, \boldsymbol{t}) \sim N(\boldsymbol{\mu}', \sigma'^2)$$

$$\begin{aligned}
\boldsymbol{\mu}' &= \boldsymbol{\mu}^T \hat{\boldsymbol{x}} \\
\sigma'^2 &= \hat{\boldsymbol{x}}^T \Sigma \hat{\boldsymbol{x}} + \sigma^2
\end{aligned}$$

Exercise 5

M = number of documents, K = number of topics, V = vocabulary size
N_{wi} = number of times word w is assigned to topic i
N_i = number of words assigned to topic i
N_{ji} = number of words in document j assigned to topic i
x_{lj} = l^{th} word in document j (observed)
z_{lj} = topic assignment for the l^{th} word in document j
$\hat{\phi}_i$ = distribution of words for topic i
$\hat{\theta}_j$ = distribution of topics for document j

Joint log likelihood:

$$\begin{aligned}\log p(z, w \mid \alpha, \beta) \propto \sum_{j=1}^{M}\Big(\sum_{i=1}^{K} \log \Gamma(N_{ji} + \alpha_i)\Big) - \log \Gamma\Big(\sum_{i=1}^{K} N_{ji} + \alpha_i\Big) \\ + \sum_{i=1}^{K}\Big(\sum_{w=1}^{V} \log \Gamma(N_{wi} + \beta_w)\Big) - \log \Gamma\Big(\sum_{w=1}^{V} N_{wi} + \beta_w\Big)\end{aligned}$$

Conditional:

$$p(z_{l,j} = k \mid z^{-lj}, x, \alpha, \beta) = \frac{1}{Z} a_{ji} b_{wi}$$

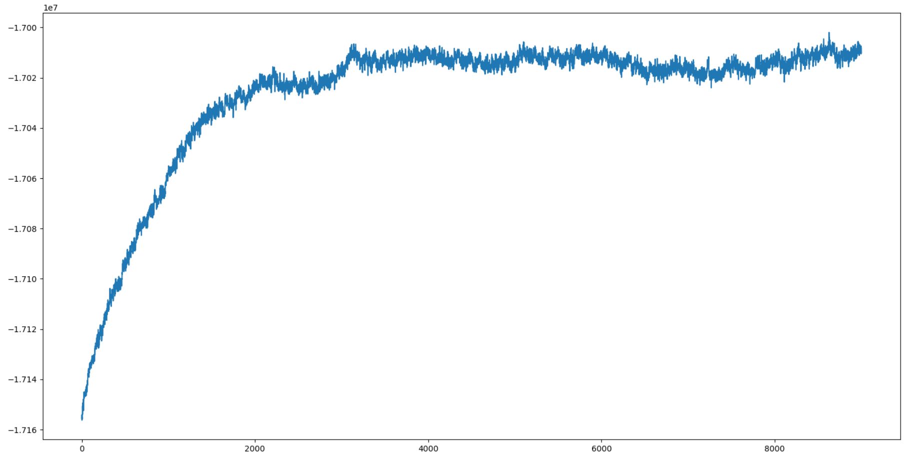
$$a_{ji} = N_{ji}^{-lj} + \alpha \qquad b_{wi} = \frac{N_{wi}^{-lj} + \beta}{N_i^{-lj} + V\beta} \qquad Z = \sum_{i}^{K} a_{ji} b_{wi}$$

Estimates:

$$\hat{\phi}_{wi} = \frac{N_{wi} + \beta}{N_i + V \cdot \beta}$$

$$\hat{\theta}_{ji} = \frac{N_{ji} + \alpha}{N_j + K \cdot \alpha}$$

The joint log likelihood plot,



The 10 most probable words per topic (rows correspond topics)

```
['robot'] ['trajectory'] ['head'] ['eye'] ['position'] ['control'] ['motor'] ['system'] ['model'] ['controller']
['neural'] ['weights'] ['networks'] ['training'] ['layer'] ['output'] ['input'] ['units'] ['hidden'] ['network']
['temporal'] ['neurons'] ['neuron'] ['rate'] ['frequency'] ['time'] ['spike'] ['information'] ['firing'] ['signal']
['feature'] ['vision'] ['figure'] ['objects'] ['motion'] ['features'] ['object'] ['images'] ['visual'] ['image']
['actions'] ['reinforcement'] ['time'] ['state'] ['function'] ['action'] ['states'] ['policy'] ['optimal'] ['learning']
['case'] ['networks'] ['class'] ['set'] ['threshold'] ['number'] ['bound'] ['theorem'] ['functions'] ['function']
['matrix'] ['approximation'] ['method'] ['function'] ['functions'] ['optimal'] ['linear'] ['error'] ['vector'] ['noise']
['speaker'] ['state'] ['training'] ['context'] ['hmm'] ['time'] ['speech'] ['word'] ['recognition'] ['system']
['classifier'] ['class'] ['neural'] ['performance'] ['error'] ['data'] ['test'] ['training'] ['set'] ['classification']
['model'] ['recurrent'] ['neuron'] ['system'] ['neurons'] ['state'] ['networks'] ['neural'] ['time'] ['network']
['values'] ['work'] ['figure'] ['data'] ['set'] ['problem'] ['point'] ['space'] ['approach'] ['local']
['bayesian'] ['probability'] ['mixture'] ['distribution'] ['parameters'] ['data'] ['models'] ['gaussian'] ['model'] ['likelihood']
['space'] ['linear'] ['vectors'] ['component'] ['matrix'] ['data'] ['analysis'] ['components'] ['information'] ['vector']
['random'] ['rate'] ['error'] ['examples'] ['gradient'] ['convergence'] ['algorithms'] ['time'] ['algorithm'] ['learning']
['language'] ['sequence'] ['nodes'] ['tree'] ['representations'] ['structure'] ['representation'] ['rules'] ['node'] ['rule']
['computer'] ['results'] ['level'] ['vector'] ['information'] ['number'] ['system'] ['memory'] ['parallel'] ['performance']
['neural'] ['vlsi'] ['voltage'] ['current'] ['output'] ['figure'] ['input'] ['chip'] ['analog'] ['circuit']
['cortical'] ['figure'] ['neurons'] ['cortex'] ['activity'] ['visual'] ['input'] ['cell'] ['model'] ['cells']
['tangent'] ['matching'] ['vectors'] ['feature'] ['set'] ['pattern'] ['character'] ['distance'] ['recognition'] ['image']
['theory'] ['generalization'] ['large'] ['field'] ['case'] ['learning'] ['energy'] ['function'] ['order'] ['noise']
```

The most similar titles to document 0 are,

```
['Observability of Neural Network Behavior ']
['Noisy Neural Networks and Generalizations,']
['A Precise Characterization of the Class of Languages Recognized by Neural Nets under Gaussian and Other Common Noise Di
['Analog Neural Networks of Limited Precision I: Computing with Multilinear Threshold Functions ']
['The Hopfield Model with Multi-Level Neurons ']
['On Properties of Networks of Neuron-Like Elements ']
['Complexity of Finite Precision Neural Network Classifier ']
['On the Effect of Analog Noise in Discrete-Time Analog Computations, ']
['Are Hopfield Networks Faster than Conventional Computers ?, ']
['On the Power of Neural Networks for Solving Hard Problems ']
```

Joint log likelihood code,

```python
from scipy.special import loggamma
import numba_scipy
from numba import jit

@jit(nopython=True)
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]

    ll = 0

    for j in range(n_docs):
        term_1 = 0
        for i in range(n_topics):
            term_1 += loggamma(doc_counts[j][i] + alpha)
```

```python
        term_2 = 0
        for i in range(n_topics):
            term_2 += doc_counts[j][i] + alpha
        term_2 = loggamma(term_2)
        ll += (term_1 - term_2)

    for i in range(n_topics):
        term_1 = 0
        for r in range(alphabet_size):
            term_1 += loggamma(topic_counts[i][r] + gamma)

        term_2 = 0
        for r in range(alphabet_size):
            term_2 += topic_counts[i][r] + gamma
        term_2 = loggamma(term_2)
        ll += (term_1 - term_2)

    return ll
```

Sampler code,

```python
import numpy as np
import numba as nb


@nb.jit(nopython=True)
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 distributions 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_words = len(words)
    n_topics = len(topic_N)
    alphabet_size = topic_counts.shape[1]
```

```python
    alphabet_size = topic_counts.shape[1]

    topic_assignment_updated = topic_assignment
    topic_counts_updated = topic_counts
    doc_counts_updated = doc_counts
    topic_N_updated = topic_N

    for i in range(n_words):

        # get relevant indices for the current word
        doc_idx = document_assignment[i]
        top_idx = topic_assignment[i]
        word_idx = words[i]

        # remove the current word from counts
        topic_counts_updated[top_idx][word_idx] -= 1
        doc_counts_updated[doc_idx][top_idx] -= 1
        topic_N_updated[top_idx] -= 1


        a = doc_counts_updated[doc_idx] + alpha
        b = (topic_counts_updated[:, word_idx] + gamma) / (topic_N_updated + alphabet_size * gamma)
        probabilities = a * b
        probabilities /= np.sum(probabilities)

        # sample new topic assignment
        new_top_idx = np.argmax(np.random.multinomial(1, probabilities))

        # update counts
        topic_assignment[i] = new_top_idx
        topic_counts_updated[new_top_idx][word_idx] += 1
        doc_counts_updated[doc_idx][new_top_idx] += 1
        topic_N_updated[new_top_idx] += 1

    return topic_assignment_updated, \
           topic_counts_updated, \
           doc_counts_updated, \
           topic_N_updated
```