CSE 291-D: Homework 2

Hao-en Sung [A53204772] (wrangle1005@gmail.com)
Department of Computer Science
University of California, San Diego
San Diego, CA 92092

Problem 1

(a)

According to Metropolis-Hastings update rule, a proposal will be accepted only if

$$\alpha = \frac{P(\boldsymbol{X}') \cdot Q(\boldsymbol{X}^{(t)}|\boldsymbol{X}')}{P(\boldsymbol{X}^{(t)}) \cdot Q(\boldsymbol{X}'|\boldsymbol{X}^{(t)})} >= 1.$$

If it is applied to Gibbs sampling, I have

$$\begin{aligned}
\frac{P(\boldsymbol{X}') \cdot Q(\boldsymbol{X}^{(t)}|\boldsymbol{X}')}{P(\boldsymbol{X}^{(t)}) \cdot Q(\boldsymbol{X}'|\boldsymbol{X}^{(t)})} &= \frac{P(X_i'|X_{-i}')P(X_{-i}') \cdot Q(\boldsymbol{X}^{(t)}|\boldsymbol{X}')}{P(X_i^{(t)}|X_{-i}^{(t)})P(X_{-i}^{(t)}) \cdot Q(\boldsymbol{X}'|\boldsymbol{X}^{(t)})} \\
&= \frac{P(X_i'|X_{-i}')P(X_{-i}') \cdot P(X_i^{(t)}|X_{-i}')}{P(X_i^{(t)}|X_{-i}^{(t)})P(X_{-i}^{(t)}) \cdot P(X_i'|X_{-i}^{(t)})} \\
&= \frac{P(X_i'|X_{-i}')P(X_{-i}') \cdot P(X_i^{(t)}|X_{-i}')}{P(X_i^{(t)}|X_{-i}')P(X_{-i}') \cdot P(X_i'|X_{-i}')} \\
&= 1,
\end{aligned}$$

and thus, I prove proposal by Gibbs sampling will always be accepted.

(b)

If I integral X_a at both sides of detailed balance formula, I can derive

$$\begin{aligned}
& \int_{X_a} T(X_a|X_b)P(X_b)\, d(X_a) = \int_{X_a} T(X_b|X_a)P(X_a)\, d(X_a) \\
\Rightarrow\quad & \int_{X_a} T(X_a, X_b)\, d(X_a) = \int_{X_a} T(X_b|X_a)P(X_a)\, d(X_a) \\
\Rightarrow\quad & X_b = \int_{X_a} T(X_b|X_a)P(X_a)\, d(X_a).
\end{aligned}$$

Thus, I prove the stationary distribution requirement.

Problem 2

(a): update of μ

Assume $\mu \sim \text{Gaussian}(\mu_0, \tau_0)$, the update of μ can be derived as follows.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

$$
\begin{aligned}
P(\mu|\boldsymbol{x},\tau) &= \frac{P(\mu,\boldsymbol{x},\tau)}{\sum_\mu P(\mu,\boldsymbol{x},\tau)} \\
&= \frac{P(\boldsymbol{x}|\mu,\tau)\cdot P(\mu|\tau)\cdot P(\tau)}{\sum_\mu P(\boldsymbol{x}|\mu,\tau)\cdot P(\mu|\tau)\cdot P(\tau)} \\
&= \frac{P(\boldsymbol{x}|\mu,\tau)\cdot P(\mu)}{\sum_\mu P(\boldsymbol{x}|\mu,\tau)\cdot P(\mu)} \\
&\propto P(\boldsymbol{x}|\mu,\tau)\cdot P(\mu) \\
&= P(\mu)\cdot \prod_i P(x_i|\mu,\tau) \\
&= \sqrt{\frac{\tau_0}{2\pi}} e^{-\frac{1}{2}\tau_0(\mu-\mu_0)^2} \cdot \prod_i \left(\sqrt{\frac{\tau}{2\pi}} e^{-\frac{1}{2}\tau(x_i-\mu)^2} \right) \\
&= \sqrt{\frac{\tau_0\tau^n}{2\pi}} e^{-\frac{1}{2}\left[\tau_0(\mu-\mu_0)^2+\tau\cdot\left(\sum_i(x_i-\bar{x})^2+n(\bar{x}-\mu)^2\right)\right]} \\
&= \sqrt{\frac{\tau_0\tau^n}{2\pi}} e^{-\frac{1}{2}\left[\tau\cdot\sum_i(x_i-\bar{x})^2+(\tau_0+\tau n)\left(\mu-\frac{\tau_0\mu_0+\tau n\bar{x}}{\tau_0+\tau n}\right)^2+\frac{\tau_0\tau n}{\tau_0+\tau n}(\mu_0-\bar{x})^2\right]} \\
&\propto \sqrt{\frac{\tau_0+\tau n}{2\pi}} e^{-\frac{1}{2}(\tau_0+\tau n)\left(\mu-\frac{\tau_0\mu_0+\tau n\bar{x}}{\tau_0+\tau n}\right)^2} \\
&= \text{Gaussian}\left(\frac{\tau_0\mu_0+\tau n\bar{x}}{\tau_0+\tau n}, \frac{1}{\tau_0+\tau n}\right)
\end{aligned}
$$

(b): update of τ

Assume $\tau \sim \text{Gamma}(\alpha,\beta)$, the update of τ can be derived as follows.

$$
\begin{aligned}
P(\tau|\boldsymbol{x},\mu) &= \frac{P(\mu,\boldsymbol{x},\tau)}{\sum_\tau P(\mu,\boldsymbol{x},\tau)} \\
&= \frac{P(\boldsymbol{x}|\mu,\tau)\cdot P(\tau|\mu)\cdot P(\mu)}{\sum_\tau P(\boldsymbol{x}|\mu,\tau)\cdot P(\tau|\mu)\cdot P(\mu)} \\
&= \frac{P(\boldsymbol{x}|\mu,\tau)\cdot P(\tau)}{\sum_\tau P(\boldsymbol{x}|\mu,\tau)\cdot P(\tau)} \\
&\propto P(\boldsymbol{x}|\mu,\tau)\cdot P(\tau) \\
&= P(\tau)\cdot \prod_i P(x_i|\mu,\tau) \\
&= \frac{\beta^\alpha}{\Gamma(\alpha)}\tau^{\alpha-1}e^{-\beta\tau} \cdot \prod_i \left(\sqrt{\frac{\tau}{2\pi}} e^{-\frac{1}{2}\tau(x_i-\mu)^2} \right) \\
&= \frac{1}{\sqrt{2\pi}}\tau^{\alpha+\frac{n}{2}-1}e^{-\tau\cdot\left(\beta+\frac{1}{2}\sum_i(x_i-\mu)^2\right)} \\
&\approx \text{Gamma}\left(\alpha+\frac{n}{2}, \beta+\frac{1}{2}\sum_i(x_i-\mu)^2\right)
\end{aligned}
$$

Problem 3

(a)

The figure is shown as follows.

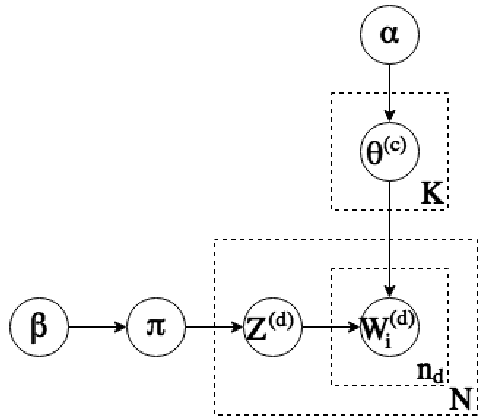

Figure 1: Directed Graphical Model Diagram

(b)

The joint probability of model can be written as follows.

$$P(\boldsymbol{\pi}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{w}|\alpha = 0.1, \beta = 1) = P(\boldsymbol{\pi}|\beta = 1) \cdot \prod_d P(Z^{(d)}|\boldsymbol{\pi}) \cdot \prod_c P(\theta^{(c)}|\alpha = 0.1) \cdot \prod_d \prod_i P(w_i^{(d)}|Z^{(d)}, \boldsymbol{\theta})$$

(c)

From Fig. 1, one can easy tell the follows.

- Markov blanket for π includes
  - β,
  - $Z^{(d)},\ \forall d$.
- Markov blanket for $\theta^{(c)}$ includes
  - α,
  - $w_i^{(d)},\ \forall i,\ \forall Z^{(d)} = c$,
  - $Z^{(d)},\ \forall Z^{(d)} = c$.
- Markov blanket for $Z^{(d)}$ includes
  - $\boldsymbol{\pi}$,
  - $w_i^{(d)}, \forall i$,
  - $\theta^{Z^{(d)}}$.

(d)

The update for $\boldsymbol{\pi}$ is derived as follows. It it noticeable that $Z_i^{(d)}$ is a boolean notation to indicate the latent class for document d, i.e. $\sum_i Z_i^{(d)} = 1$.

$$
\begin{aligned}
P(\boldsymbol{\pi}|\beta=1,\boldsymbol{Z}) &= \frac{P(\boldsymbol{\pi},\beta=1,\boldsymbol{Z})}{\sum_{\boldsymbol{\pi}} P(\boldsymbol{\pi},\beta=1,\boldsymbol{Z})} \\
&\approx P(\boldsymbol{\pi}|\beta=1)\cdot\prod_d P(Z^{(d)}|\boldsymbol{\pi}) \\
&= \frac{\Gamma(K)}{\Gamma(1)^K}\cdot\prod_{i=1}^K \pi_i^0 \cdot \prod_d \left(\frac{\Gamma(2)}{\prod_{i=1}^K \Gamma(Z_i^{(d)}+1)} \prod_{i=1}^K \pi_i^{Z_i^{(d)}} \right) \\
&\approx \prod_d \left(\prod_{i=1}^K \pi_i^{Z_i^{(d)}} \right) \\
&= \prod_{i=1}^K \prod_d \pi_i^{Z_i^{(d)}} \\
&= \prod_{i=1}^K \pi_i^{\sum_d Z_i^{(d)}} \\
&\approx \text{Dirichlet}\left(\pi_1,\cdots,\pi_K \,\middle|\, \sum_d Z_1^{(d)}+1,\cdots,\sum_d Z_K^{(d)}+1 \right)
\end{aligned}
$$

The update for $\theta^{(c)}$ is derived as follows. Notice that I use $'$ to indicate every variable with document class c for simplicity, for example: Z', w', and d'. On top of that, $Z^{(d)}$ is again boolean notation here; while $w_i^{(d)}$ indicates the count of words i in document d.

$$
\begin{aligned}
P(\theta^{(c)}|\alpha=0.1,\boldsymbol{w},\boldsymbol{Z}) &= P(\theta^{(c)}|\alpha=0.1,\boldsymbol{w}',\boldsymbol{Z}') \\
&= \frac{P(\theta^{(c)},\alpha=0.1,\boldsymbol{w}',\boldsymbol{Z}')}{\sum_{\theta^c} P(\theta^{(c)},\alpha=0.1,\boldsymbol{w}',\boldsymbol{Z}')} \\
&\approx P(\theta^{(c)}|\alpha=0.1)\cdot\prod_{d'} P(w^{(d')}|\theta^{(c)}) \\
&= \frac{\Gamma(0.1V)}{\Gamma(0.1)^V}\cdot\prod_{i=1}^V \left(\theta_i^{(c)}\right)^{-0.9}\cdot\prod_{d'} \frac{\Gamma(n'_d+1)}{\prod_{i=1}^V \Gamma\left(w_i^{(d')}+1\right)} \prod_{i=1}^V \left(\theta_i^{(c)}\right)^{w_i^{(d')}} \\
&\approx \prod_{i=1}^V \left(\theta_i^{(c)}\right)^{-0.9}\cdot\prod_{d'}\prod_{i=1}^V \left(\theta_i^{(c)}\right)^{w_i^{(d')}} \\
&= \prod_{i=1}^V \left(\theta_i^{(c)}\right)^{-0.9+\sum_{d'} w_i^{(d')}} \\
&\approx \text{Dirichlet}\left(\theta_1^{(c)},\cdots,\theta_V^{(c)} \,\middle|\, \sum_{d'} w_1^{(d')}+0.1,\cdots,\sum_{d'} w_V^{(d')}+0.1 \right)
\end{aligned}
$$

The update for $\boldsymbol{Z}^d$ is derived as follows. Again, $Z^{(d)}$ and $w_i^{(d)}$ follow the definition in previous update formulas.

$$
\begin{aligned}
P(\boldsymbol{Z}^{(d)}|\boldsymbol{\pi}, w^{(d)}, \boldsymbol{\theta}) &= \frac{P(\boldsymbol{Z}^{(d)}, \boldsymbol{\pi}, w^{(d)}, \boldsymbol{\theta})}{\sum_{\boldsymbol{Z}^{(d)}} P(\boldsymbol{Z}^{(d)}, \boldsymbol{\pi}, w^{(d)}, \boldsymbol{\theta})} \\
&= \frac{\Gamma(2)}{\prod_{i=1}^{K} \Gamma\left(Z_i^{(d)}+1\right)} \prod_{i=1}^{K} \pi_i^{Z_i^{(d)}} \cdot \frac{\Gamma(n_d+1)}{\prod_{i=1}^{V} \Gamma\left(w_i^{(d)}+1\right)} \prod_{i=1}^{V} \left(\theta_i^{Z^{(d)}}\right)^{w_i^{(d)}} \\
&\approx \frac{\prod_{i=1}^{K} \pi_i^{Z_i^{(d)}} \cdot \prod_{i=1}^{V} \left(\theta_i^{Z^{(d)}}\right)^{w_i^{(d)}}}{\prod_{i=1}^{K} \Gamma\left(Z_i^{(d)}+1\right)} \\
&= \pi_{Z_i^{(d)}} \cdot \prod_{i=1}^{V} \left(\theta_i^{Z^{(d)}}\right)^{w_i^{(d)}}
\end{aligned}
$$

(e)

According to my implementation, shown as Code 1, the top ten words for each latent class can be listed as follows. It is noticeable that I filter out most common English stop words through one of the *CountVectorizer* parameters.

- $P(\pi = 0) = 0.35005771$: ['edu', u'lines', u'subject', u'organization', u'com', u'space', u'writes', u'article', u'posting', u'just']
- $P(\pi = 1) = 0.25799032$: [u'edu', u'image', u'jpeg', u'com', u'subject', u'organization', u'lines', u'space', u'graphics', u'software']
- $P(\pi = 2) = 0.39195198$: [u'edu', u'lines', u'subject', u'organization', u'com', u'space', u'writes', u'article', u'like', u'posting']

As one can tell, there are many stop words among them, since these words account for a great portion in each document.

Code Listing 1: Code for Homework 2

```
# coding: utf-8

# In[1]:

import numpy as np
from collections import defaultdict
import itertools

from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score

seed = 514
np.random.seed(seed)


# In[2]:

# select targets
categories = [ 'talk.religion.misc', 'comp.graphics', 'sci.space']
K = len(categories)


# In[3]:

# load data
```

```python
newsgroups = fetch_20newsgroups(subset='all', categories=categories,
                                   shuffle=True, random_state=seed)
N = len(newsgroups.data)


# In[4]:

# create vectorizer
vectorizer = CountVectorizer(stop_words='english')
tdata = vectorizer.fit_transform(newsgroups.data, )
V = len(vectorizer.vocabulary_)


# In[5]:

# prepare words
wordLst = defaultdict(list)

idxLst = zip(*tdata.nonzero())
for idx in idxLst:
    wordLst[idx[0]].append((idx[1], tdata[idx]))


# In[6]:

# given a, b
a = 0.1
b = 1.0

# initialize pZ
pZ = np.zeros((N, K))
for d in range(N):
    for k in range(K):
        pZ[d,k] = 1.0 / K

# initialize Z
np.random.seed(514)
Z = np.zeros((N,), dtype=np.int)
for d in range(N):
    c = np.random.choice(range(K), 1, p=pZ[d])[0]
    Z[d] = c


# In[7]:

# main process
for i in xrange(200):
    if i % 10 == 0:
        print 'Now is working on Iteration #%d...' % i

    np.random.seed(514)

    # update pP
    sD = np.zeros((K,))
    for d in range(N):
        sD[Z[d]] += 1

    pP = np.random.dirichlet(sD + b, 1)[0]

    # update pT
    sW = np.zeros((K, V))
    for d, vcLst in wordLst.iteritems():
        for v, c in vcLst:
            sW[Z[d], v] += c
```

```python
    pT = np.zeros((K, V))
    for k in range(K):
        pT[k] = np.random.dirichlet(sW[k] + a, 1)[0]

    # update pZ
    for d in range(N):
        for k in range(K):
            pZ[d,k] = np.log(pP[k])
            for v, c in wordLst[d]:
                pZ[d,k] += c * np.log(pT[k,v])

        # trick
        maxv = max(pZ[d])
        pZ[d] = np.exp(pZ[d] - maxv)
        pZ[d] /= sum(pZ[d])

    # update Z
    for d in range(N):
        c = np.random.choice(range(K), 1, p=pZ[d])[0]
        Z[d] = c


# In[8]:

# maximize prediction
print pP
argm = np.argmax(pZ, axis=1)

def calACC(perm):
    pred = []
    for v in argm:
        pred.append(perm[v])
    return accuracy_score(newsgroups.target, pred)

maxc = 0
for perm in itertools.permutations(range(K), K):
    pred = []
    for v in argm:
        pred.append(perm[v])
    acc = accuracy_score(newsgroups.target, pred)
    if acc > maxc:
        maxc = acc
        maxp = pred


# In[9]:

# ten most frequent words
words = vectorizer.get_feature_names()
for k in range(K):
    sortLst = sorted(list(enumerate(pT[k])), key=lambda x: x[1],
                                          reverse=True)
    print [words[p[0]] for p in sortLst[:10]]
```