CPSC532W: Probabilistic Programming, Homework 1

Namrata Deka

1. Let a random vector $\mathbf{y}$ be sampled from a Poisson distribution, $\mathbf{y} \sim \text{Poisson}(\lambda)$, and the parameter λ be distributed as per a Gamma distribution $\lambda \sim \text{Gamma}(\alpha, \beta)$. Then, from Bayes rule we have:

$$\begin{aligned}
P(\lambda|\mathbf{y};\alpha,\beta) &\propto P(\mathbf{y}|\lambda)P(\lambda;\alpha,\beta) \\
&= \prod_{i=1}^{n} P(y_i|\lambda)P(\lambda;\alpha,\beta) \\
&\propto \left(\prod_{i+1}^{n} \lambda^{y_i} e^{-\lambda}\right) \lambda^{\alpha-1} e^{-\beta\lambda} \\
&= \lambda^{\sum y_i} e^{-n\lambda} \lambda^{\alpha-1} e^{-\beta\lambda} \\
&= \lambda^{\alpha+\sum y_i - 1} e^{-(\beta+n)\lambda} \\
&= \lambda^{\alpha+|\mathbf{y}|-1} e^{-(\beta+n)\lambda}
\end{aligned}$$

Therefore, the posterior also belongs to the family of Gamma distributions: $\lambda|\mathbf{y} \sim \text{Gamma}(\alpha + |\mathbf{y}|, \beta + n)$.

2. Let $\mathcal{X}$ and $\mathcal{Y}$ be two sets of random variables in a graphical model such that, without loss of generality, the joint probability density factorizes as follows:

$$P(\mathcal{X}, \mathcal{Y}) = P(\mathcal{Y}|\mathcal{X})P(\mathcal{X})$$

Let x_i and y_i represent the state of $\mathcal{X}$ and $\mathcal{Y}$ respectively during Gibbs sampling. Consider the transition from (x_1, y_1) to (x_1, y_2). To show that the transition T satisfies the detailed balance equation we need to prove that:

$$P(x_1, y_1)T((x_1, y_1) \to (x_1, y_2)) = P(x_1, y_2)T((x_1, y_2) \to (x_1, y_1))$$

Consider the L.H.S.:

$$\begin{aligned}
P(x_1, y_1)T((x_1, y_1) \to (x_1, y_2)) &= P(x_1, y_1)P(y_2|x_1) \\
&= P(x_1, y_1)\frac{P(x_1, y_2)}{P(x_1)} \\
&= P(x_1, y_2)\frac{P(x_1, y_1)}{P(x_1)} \\
&= P(x_1, y_2)P(y_1|x_1) \\
&= P(x_1, y_2)T((x_1, y_2) \to (x_1, y_1))
\end{aligned}$$

Therefore, the Gibbs transition operator $P(\mathcal{Y}|\mathcal{X})$ satisfies the detailed balance equation and this can be generalized to increased number of sets.

In the case of MH, the acceptance probability of the above transition would be calculated as:

$$\begin{aligned}
A((x_1, y_2), (x_1, y_1)) &= \min\left(1, \frac{P(x_1, y_2)P(y_1|x_1)}{P(x_1, y_1)P(y_2|x_1)}\right) \\
&= \min\left(1, \frac{P(x_1, y_2)P(x_1, y_1)}{P(x_1, y_1)P(x_1, y_2)}\right) \\
&= \min(1, 1) = 1
\end{aligned}$$

Therefore, the Gibbs transition is a special case of a MH transition that always accepts the proposed state.

4a. MH within Gibbs on $\mathbf{w}$: In a Gibbs iteration we would like to sample from $P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)$. Therefore, a MH transition from $\mathbf{w}$ to $\mathbf{w}'$ would have the following acceptance probability:

$$\begin{aligned} A(\mathbf{w}', \mathbf{w}) &= \min\left(1, \frac{P(\mathbf{w}'|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)P(\mathbf{w}|\mathbf{w}')}{P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)P(\mathbf{w}'|\mathbf{w})}\right) \\ &= \min\left(1, \frac{P(\mathbf{w}'|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)}{P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)}\right) \end{aligned}$$

Lets r be the ratio of the conditionals. Then:

$$A(\mathbf{w}', \mathbf{w}) = \min(1, r)$$

where,

$$\begin{aligned} r &= \frac{P(\mathbf{w}'|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)}{P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)} \\ &= \frac{P(\mathbf{t, w', x}, \sigma^2, \alpha)}{P(\mathbf{t, w, x}, \sigma^2, \alpha)} \\ &= \frac{\prod_{n=1}^{N} P(t_n|x_n, \mathbf{w}', \sigma^2)P(\mathbf{w}'|\alpha)}{\prod_{n=1}^{N} P(t_n|x_n, \mathbf{w}, \sigma^2)P(\mathbf{w}|\alpha)} \\ &= \frac{\prod_{n=1}^{N} \exp\left(\frac{(t_n - \mathbf{w}'^T x_n)^2}{2\sigma^2}\right) \prod_{d=1}^{D} \exp\left(\frac{(\mathbf{w}' - \mathbf{0})^2}{2\alpha}\right)}{\prod_{n=1}^{N} \exp\left(\frac{(t_n - \mathbf{w}^T x_n)^2}{2\sigma^2}\right) \prod_{d=1}^{D} \exp\left(\frac{(\mathbf{w} - \mathbf{0})^2}{2\alpha}\right)} \\ &= \frac{\exp\left(\frac{(\mathbf{t} - \mathbf{w}'\hat{}\mathbf{Tx})^2}{2\sigma^2\mathcal{I}_n}\right) \exp\left(\frac{(\mathbf{w}'^T\mathbf{w}')}{2\alpha\mathcal{I}_D}\right)}{\exp\left(\frac{(\mathbf{t} - \mathbf{w}\hat{}\mathbf{Tx})^2}{2\sigma^2\mathcal{I}_n}\right) \exp\left(\frac{(\mathbf{w}^T\mathbf{w})}{2\alpha\mathcal{I}_D}\right)} \end{aligned}$$

4b. Pure Gibbs on $\mathbf{w}$: In a pure Gibbs iteration we would need the closed form expression of the conditional $P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha)$.

$$\begin{aligned} P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha) &= \frac{P(\mathbf{t}, \mathbf{w}, \mathbf{x}, \sigma^2, \alpha)}{\int_{\mathbf{w}} P(\mathbf{t}, \mathbf{w}, \mathbf{x}, \sigma^2, \alpha) d\mathbf{w}} \\ &= \frac{\prod_{n=1}^{N} P(t_n|x_n, \mathbf{w}, \sigma^2)P(\mathbf{w}|\alpha)}{\int_{\mathbf{w}} \prod_{n=1}^{N} P(t_n|x_n, \mathbf{w}, \sigma^2)P(\mathbf{w}|\alpha) d\mathbf{w}} \\ &= \frac{P(\mathbf{t|x, w}, \sigma^2)P(\mathbf{w}|\alpha)}{P(\mathbf{t, x}, \sigma^2, \alpha)} \\ &= \frac{\mathcal{N}(\mathbf{w}^T\mathbf{x}, \sigma^2\mathcal{I}_N)\mathcal{N}(\mathbf{w}, \alpha\mathcal{I}_D)}{P(\mathbf{t, x}, \sigma^2, \alpha)} \end{aligned}$$

Since, both the likelihood and the prior are Gaussians, we have (from Bishop 2.116):

$$P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha) = \mathcal{N}\left(\Sigma\{\mathbf{x}^T(\sigma^2\mathcal{I}_N)^{-1}\mathbf{t}\}, \Sigma\right)$$

where,

$$\Sigma = \left(\alpha\mathcal{I}_D + \mathbf{x}^T(\sigma^2\mathcal{I}_N)\mathbf{x}\right)^{-1}$$

4c. Analytic form of the posterior predictive:

$$\begin{aligned} P(\hat{\mathbf{t}}|\mathbf{t}) &= \int_{\mathbf{w}} P(\hat{\mathbf{t}}|\mathbf{w}, \hat{\mathbf{x}}, \sigma^2)P(\mathbf{w}|\mathbf{t}, \mathbf{x}, \sigma^2, \alpha) d\mathbf{w} \\ &= \int_{\mathbf{w}} \mathcal{N}\left(\mathbf{w}^T\hat{x}, \sigma^2\right) \mathcal{N}\left(\Sigma\{\mathbf{x}^T(\sigma^2\mathcal{I}_N)^{-1}\mathbf{t}\}, \Sigma\right) d\mathbf{w} \end{aligned}$$

The posterior predictive is a convolution of two Gaussian distributions which has a closed form solution also in the form of a Gaussian.

3a. Probability that it is cloudy given that the grass is wet = 57.58%.

```
## condition and marginalize:
#TODO
p_C_given_W = np.zeros((2, 2))

p_C_W = np.zeros((2, 2))
for c in range(2):
    for w in range(2):
        for r in range(2):
            for s in range(2):
                p_C_W[c, w] += p[c, s, r, w]
p_C_W /= p_C_W.sum()

p_W = np.zeros((2))
for w in range(2):
    for c in range(2):
        for r in range(2):
            for s in range(2):
                p_W[w] += p[c, s, r, w]
p_W /= p_W.sum()

for c in range(2):
    for w in range(2):
        p_C_given_W[c, w] = p_C_W[c, w] / p_W[w]
print('There is a {:.2f}% chance it is cloudy given the grass is wet'.format(p_C_given_W[1, 1]*100))
```

Figure 1: 3a: Solution by enumerating all possibilities.

3b. Probability that it is cloudy given that the grass is wet = 64.20%.

```
##2. ancestral sampling and rejection:
def sample_from_dist(dist):
    flat = dist.flatten()
    idx = np.random.choice(a=flat.size, p=flat)
    return np.unravel_index(idx, dist.shape)

num_samples = 10000
samples = np.zeros(num_samples)
rejections = 0
i = 0
while i < num_samples:
    #TODO
    c, w = sample_from_dist(p_C_W)
    u = np.random.uniform(low=0, high=p_C_W[c, w])
    if u <= p_C_W[c, w] * int(w==1):
        samples[i] = 1
    else:
        rejections += 1
    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)))
```

Figure 2: 3b. Using ancestral sampling and rejection.

3c. Probability that it is cloudy given that the grass is wet = 57.32%, and 26.36% of the samples were rejected.

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

#TODO
# set w = True
w = 1
# begin loop
i = 0
while i < num_samples:

    r = np.random.choice(a=p_R_given_C_S_W[c, s, :, w].size, p=p_R_given_C_S_W[c, s, :, w])

    s = np.random.choice(a=p_S_given_C_R_W[c, :, r, w].size, p=p_S_given_C_R_W[c, :, r, w])

    c = np.random.choice(a=p_C_given_S_R[:, s, r].size, p=p_C_given_S_R[:, s, r])
    if c == 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 3: 3c. Gibbs Sampling.

```
    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
    """
    #TODO
    for d in range(doc_counts.shape[0]):
        z_d = topic_assignment[document_assignment==d]
        words_d = words[document_assignment==d]
        for n in range(int(doc_N[d])):
            # get current topic of nth word in dth doc.
            z_d_n = z_d[n]

            # decrement counts for topic z_d_n.
            doc_counts[d][z_d_n] -= 1
            topic_counts[z_d_n][words_d[n]] -= 1
            topic_N[z_d_n] -= 1

            # sample a new topic for nth word in dth doc.
            p_T_D = (alpha + doc_counts[d]) / np.sum(alpha + doc_counts[d])
            p_W_T = (gamma + topic_counts[:, words_d[n]]) / (np.sum(topic_counts, axis=1) + gamma)
            p_z = p_W_T * p_T_D
            p_z /= p_z.sum()
            z = np.random.choice(topic_N.size, p=p_z)

            # set new assignment.
            doc_counts[d][z] += 1
            topic_counts[z][words_d[n]] += 1
            topic_N[z] += 1
            z_d[n] = z

        topic_assignment[document_assignment==d] = z_d

    return topic_assignment, topic_counts, doc_counts, topic_N
```

Figure 4: Q5: LDA Sampler

joint_log_lik.py | sample_topic_assignment.py | lda.py | most_similar_titles_to_0

```python
import numpy as np
from scipy.special import gamma as gamma_fn

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
    """
    #TODO

    jll = 0
    for k in range(topic_counts.shape[0]):
        jll += np.sum(np.log(gamma_fn(topic_counts[k, :] + gamma))) - np.log(gamma_fn(np.sum(topic_counts[k, :] + gamma)))
        jll -= topic_counts.shape[1] * np.log(gamma_fn(gamma)) - np.log(gamma_fn(topic_counts.shape[1] * gamma))

    for d in range(doc_counts.shape[0]):
        jll += np.sum(np.log(gamma_fn(doc_counts[d, :] + alpha))) - np.log(gamma_fn(np.sum(doc_counts[d, :] + alpha)))
        jll -= doc_counts.shape[1] * np.log(gamma_fn(alpha)) - np.log(gamma_fn(doc_counts.shape[1] * alpha))

    return jll
```

Figure 5: Q5: Joint Log-likelihood

```python
### find the 10 most probable words of the 20 topics:
#TODO:
topic_counts_norm = topic_counts / np.expand_dims(topic_counts.sum(axis=1), 1)
top_10_idx = np.argsort(topic_counts_norm, axis=1)[:, -1:-11:-1]

fstr = ''
for k in range(topic_N.shape[0]):
    fstr += 'topic {}: '.format(k)
    for w in top_10_idx[k]:
        fstr += '{}, '.format(WO[w][0])
    fstr += '\n'

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

Figure 6: 5a.: Top 10 most probable words per topic

joint_log_lik.py | sample_topic_assignment.py | lda.py | most_probable_words_per_topic

```
topic 0: firing, cells, activity, cell, input, cortex, cortical, synaptic, olfactory, neurons,
topic 1: input, vector, code, error, output, matrix, memory, performance, associative, receptive,
topic 2: networks, case, time, learning, system, order, model, network, pattern, fig,
topic 3: hidden, units, system, unit, inputs, problem, figure, net, learning, output,
topic 4: state, model, space, orientation, information, states, noise, input, map, principle,
topic 5: time, network, number, memory, capacity, output, patterns, set, nat, input,
topic 6: layer, representation, nodes, units, role, fig, decision, classifiers, patterns, representations,
topic 7: synaptic, neural, connection, parallel, chip, model, connections, analog, processing, neurons,
topic 8: neural, networks, convergence, neurons, matrix, network, output, figure, equation, state,
topic 9: approach, network, activation, joint, units, unit, number, weight, networks, performance,
topic 10: input, neural, function, systems, output, functions, values, hidden, network, form,
topic 11: time, connections, system, neural, connected, set, probability, number, effects, neurons,
topic 12: network, neural, problem, functions, neurons, output, inputs, fig, threshold, classification,
topic 13: stimulus, response, model, information, spike, threshold, potential, stimuli, figure, axon,
topic 14: network, system, number, state, order, initial, networks, matrix, fixed, delay,
topic 15: training, time, net, output, input, signal, dynamic, error, units, figure,
topic 16: memory, fig, hopfield, image, vectors, input, vector, optical, energy, neural,
topic 17: learning, units, algorithm, weight, weights, output, networks, set, network, error,
topic 18: cells, cell, pattern, layer, patterns, input, control, fig, motor, probability,
topic 19: networks, method, neural, results, network, time, function, neuron, model, points,
```

Figure 7: 5a.: List of top 10 most probable words per topic

```python
#most similar documents to document 0 by cosine similarity over topic distribution:
#normalize topics per document and dot product:
#TODO:
d0 = doc_counts[0].reshape(1, -1)
sim = np.matmul(d0, doc_counts.T) / (np.linalg.norm(d0) * np.linalg.norm(doc_counts, axis=1))
top_10_idx = np.argsort(sim)[0][-1:-12:-1]
fstr = 'Titles most similar to {} are:\n'.format(titles[0][0])
for i, idx in enumerate(top_10_idx):
    fstr += '{}. {}\n'.format(i, titles[idx][0])

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

Figure 8: 5b.: Top 10 most similar docs to Doc 0

joint_log_lik.py | sample_topic_assignment.py | lda.py | most_similar_titles_to_0

```
Titles most similar to Connectivity Versus Entropy  are:
0. Connectivity Versus Entropy
1. Network Generality, Training Required, and Precision Required
2. Computing Motion Using Resistive Networks
3. On Properties of Networks of Neuron-Like Elements
4. Capacity for Patterns and Sequences in Kanerva's SDM as Compared to Other Associative Memory Models
5. Temporal Patterns of Activity in Neural Networks
6. A Dynamical Approach to Temporal Pattern Processing
7. An Optimization Network for Matrix Inversion
8. Spatial Organization of Neural Networks: A Probabilistic Modeling Approach
9. Self-Organization of Associative Database and Its Applications
10. Probabilistic Characterization of Neural Model Computations
```

Figure 9: 5b.: List of top 10 most similar docs to Doc 0