COGSCI131–Spring 2019 — Homework 9Solutions

Yuchen Zhou, SID 3034489901

1.

- Posterior $P(W|a, n_1, n_2)$:

$$P(W|a, n_1, n_2) \propto P(W) \cdot P(a|W, n_1, n_2) = P(W) \cdot \prod_{i=1}^{t} P(a_i|W, n_{1i}, n_{2i})$$

  In logarithm form:

$$\log P(W|a, n_1, n_2) = \log P(W) + \log \sum_{i=1}^{t} P(a_i|W, n_{1i}, n_{2i})$$

- It takes more time to do multiplication than addition, so by transformation, we save time.

2.

- As Bayesian formula shows:

$$P(W|D) = \frac{P(D|W) \cdot P(W)}{\sum_i P(D|W_i) \cdot P(W_i)}$$

And

$$P(W'|D) = \frac{P(D|W') \cdot P(W')}{\sum_i P(D|W_i) \cdot P(W_i)}$$

By calculating the ratio of the posterior, we simply need to compute:

$$\frac{P(W'|D)}{P(W|D)} = \frac{P(D|W') \cdot P(W')}{P(D|W) \cdot P(W)}$$

where denominators in two fractions, the integral part are cancelled.
Since prior is given and likelihood can be easily computed, it's convenient for us to get this ratio.

3.

- Write function to compute logarithmic prior and logarithmic posterior.
- When $W \leq 0$, $prior(W) = 0$, so $\log prior(W) = -inf$.
- Code as follows:

```
def log_prior(W):
    if W<0:
        return -float('inf')
    else:
        return log(np.exp(-W))

def log_posterior(W):
    return log_prior(W) + log_likelihood(data['n1'], data['n2'], data['correct
                                                    '], W)
```

4.

- The posterior score of W over the first 300 samples as figure 1 shows.

```
fig = plt.figure ()
ax= fig.add_subplot(111)
ax.plot(postscore,color="green",linewidth=1.2)
plt.ylabel('log posterior score')
plt.show()
```

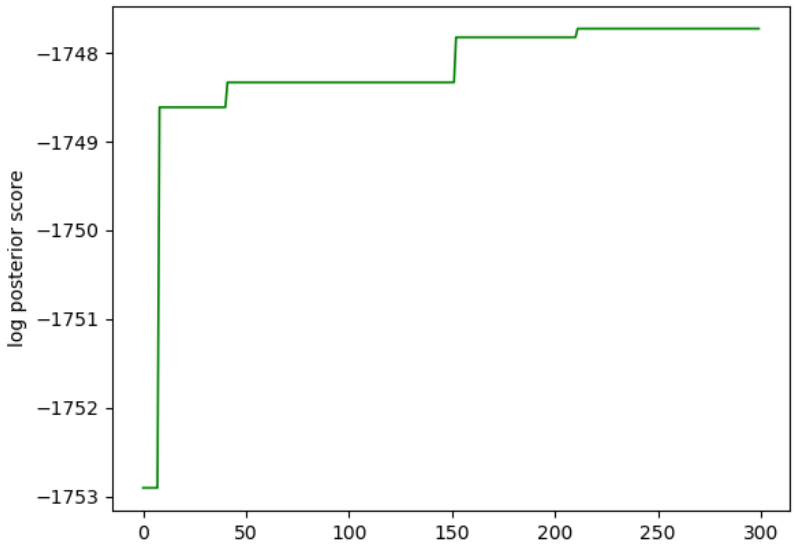

Figure 1: logarithmic posterior score of W

- The value of W over the first 300 samples as figure 2 shows.

```
fig = plt.figure ()
ax= fig.add_subplot(111)
ax.plot(hypothesis_list,color="red",linewidth=1.2)
plt.ylabel('W')
plt.show()
```

- Histogram of the samples of W over the first 10,000 samples after 1000 samples as figure 3 shows.

```
fig = plt.figure ()
ax= fig.add_subplot(111)
ax.hist(postscore[1000:],color="brown")
plt.ylabel('log posterior score')
plt.show()
```

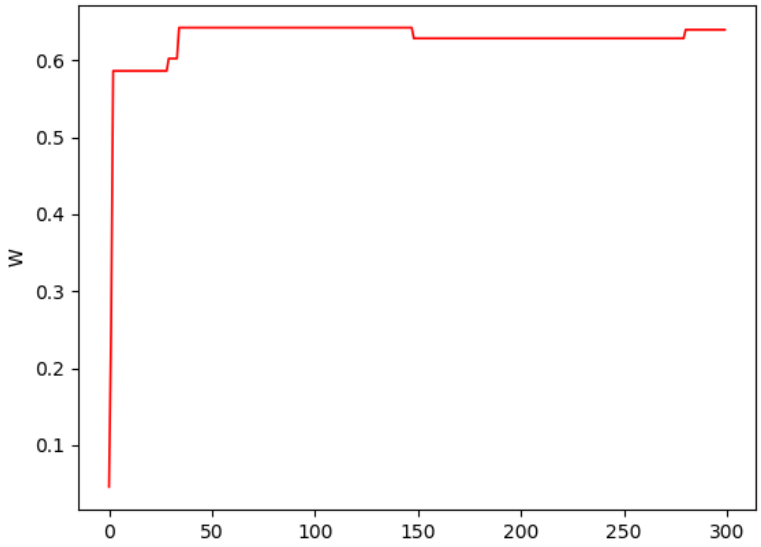

Figure 2: value of W

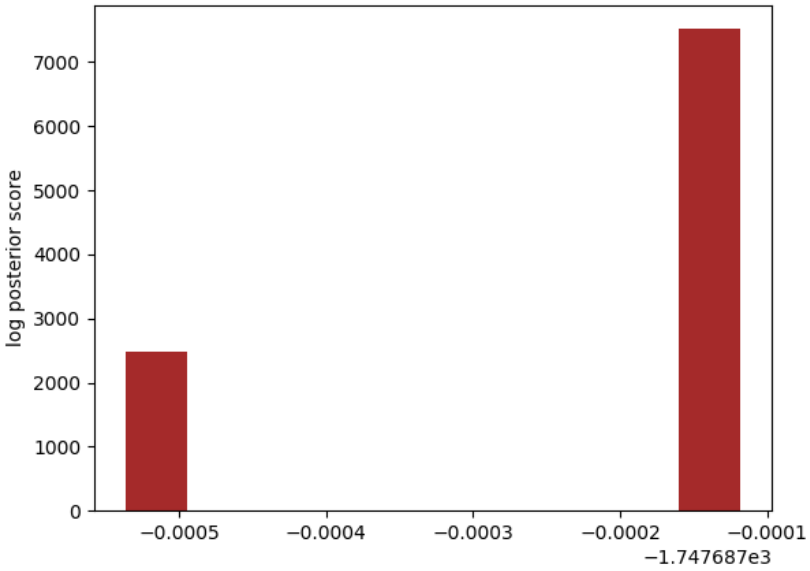

Figure 3: histogram of logarithmic posterior score

- Generalization code as follows:

```
data=pd.read_csv("Assignment9-data.csv")

hypothesis_list=[]
temp=random.random()
hypothesis_list.append(temp)
postscore=[]
postscore.append(log_posterior(temp))
for i in range(11000):
    print("i:",i)
    w = hypothesis_list[-1]
    w_next = w + np.random.normal(0, 1)
    print(w_next,w)
    p1=log_posterior(w_next)
    p2=log_posterior(w)
    if p1>p2:
        hypothesis_list.append(w_next)
        postscore.append(p1)
    else:
        if random.random()*p2<p1:
            hypothesis_list.append(w_next)
            postscore.append(p1)
        else:
            hypothesis_list.append(w)
            postscore.append(p2)
```

5.

- Use the first 10,000 results after 1000 iterations of "burn in" as samples. Compute the normalized posterior.
- Code as follows:

```
data=pd.read_csv("Assignment9-data.csv")

hypothesis_list=[]
temp=random.random()
hypothesis_list.append(temp)
postscore=[]
postscore.append(log_posterior(temp))
for i in range(11000):
    print("i:",i)
    w = hypothesis_list[-1]
    w_next = w + np.random.normal(0, 1)
    print(w_next,w)
    p1=log_posterior(w_next)
    p2=log_posterior(w)
    if p1>p2:
        hypothesis_list.append(w_next)
        postscore.append(p1)
    else:
        if random.random()*p2<p1:
            hypothesis_list.append(w_next)
            postscore.append(p1)
        else:
            hypothesis_list.append(w)
            postscore.append(p2)
pp=0
postsum=0
kk=0
for i in range(10000):
    postsum+=postscore[i+1000]
    if hypothesis_list[i+1000]<=0.3 and hypothesis_list[i+1000]>=0.2:
        kk+=1
        pp+=postscore[i+1000]
print("probability:",pp/postsum)
print("times:",kk)
```

- The result shows that there is no sample in $[0.2, 0.3]$, so the probability is 0.