

Problem 4

```
In [28]: import load_helper

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

import importlib

from torch import tensor

import seaborn as sns

In [74]: fname = '4.daphne'
graph = load_helper.graph_helper(fname)
test 4.daphne

(let [weight-prior (normal 0 1)
      W_0 (foreach 10 [])
          (foreach 1 [] (sample weight-prior)))
      W_1 (foreach 10 [])
          (foreach 10 [] (sample weight-prior)))
      W_2 (foreach 1 [])
          (foreach 10 [] (sample weight-prior)))
      b_0 (foreach 10 [])
          (foreach 1 [] (sample weight-prior)))
      b_1 (foreach 10 [])
          (foreach 1 [] (sample weight-prior)))
      b_2 (foreach 1 [])
          (foreach 1 [] (sample weight-prior)))
      x (mat-transpose [(1) (2) (3) (4) (5)])
      y [(1) (4) (9) (16) (25)]
      h_0 (mat-tanh (mat-add (mat-mul W_0 x)
                             (mat-rgmat b_0 1 5)))
      h_1 (mat-tanh (mat-add (mat-mul W_1 h_0)
                             (mat-rgmat b_1 1 5)))
      mu (mat-transpose
           (mat-tanh (mat-add (mat-mul W_2 h_1)
                              (mat-rgmat b_2 1 5)))))
      (foreach 5 [y_r y
                  mu_r mu]
        (foreach 1 [y_rc y_r
                    mu_rc mu_r]
          (observe (normal mu_rc 1) y_rc)))
      [W_0 b_0 W_1 b_1])

In [4]: import bbvi
importlib.reload(bbvi)

from bbvi import graph_bbvi_algo12

In [62]: %time
T=1000
L=20
lr=0.05
r, logW, sigma = bbvi.graph_bbvi_algo12(graph,T=T,L=L,lr=lr,
                                         do_log=False)
```







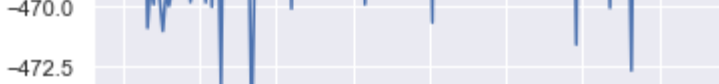
[illegible]

```

s = plot()
plt.xlabel('t')
plt.ylabel('ELBO')
plt.title('') \n Best ELBO {:.2f} \n T={:} | L={:} | Adam, lr={:} '.format(fname,elbo.max(),T,L,lr))

```

Out[73]: Text(0.5, 1.0, '4.daphne \n Best ELBO -458.58 \n T=1000 | L=20 | Adam, lr=0.05 ')



```
In [66]: r_W_0 = np.zeros((T,L,np.array(r[0][0][1]).size))
         r_b_0 = np.zeros((T,L,np.array(r[0][0][1]).size))
         W_1_rows, W_1_cols = np.array(r[0][0][2]).shape
         r_W_1 = np.zeros((T,W_1_rows, W_1_cols))
         r_b_1 = np.zeros((T,L,np.array(r[0][0][3]).size))

         for t in range(T):
             for l in range(L):
```

```

r_w_0[:,1] = np.array(r[t][:1][1]).flatten()
r_b_0[:,1] = np.array(r[t][:1][1][1]).flatten()
r_w_1[:,1] = np.array(r[t][:1][1][2]).flatten()
r_b_1[:,1] = np.array(r[t][:1][1][3]).flatten()

In [67]: probs = np.exp(logM)
         /= probs.sum()

In [68]: posterior_r_w_0 = (probs.reshape(T,L,1) * r_w_0).sum(axis=0,1)
         posterior_r_b_0 = (probs.reshape(T,L,1) * r_b_0).sum(axis=0,1)
         posterior_r_w_1 = (probs.reshape(T,L,1,1) * r_w_1).sum(axis=0,1)
         posterior_r_b_1 = (probs.reshape(T,L,1,1) * r_b_1).sum(axis=0,1)

         posterior_r2_w_0 = (probs.reshape(T,L,1) * r_w_0**2).sum(axis=0,1)
         posterior_r2_b_0 = (probs.reshape(T,L,1) * r_b_0**2).sum(axis=0,1)

```

```

posterior_r2_M_1 = (probs.reshape(T,L,1) * r_M_1**2).sum(axis=(0,1))
posterior_r2_b_1 = (probs.reshape(T,L,1) * r_b_1**2).sum(axis=(0,1))

posterior_std_M_0 = np.sqrt(posterior_r2_M_0 - posterior_r_M_0**2)
posterior_std_b_0 = np.sqrt(posterior_r2_b_0 - posterior_r_b_0**2)
posterior_std_M_1 = np.sqrt(posterior_r2_M_1 - posterior_r_M_1**2)
posterior_std_b_1 = np.sqrt(posterior_r2_b_1 - posterior_r_b_1**2)

In [69]: sns.set_theme()

f, ax = plt.subplots(figsize=(12,11))
ax = sns.heatmap(posterior_r_M_0.reshape(1,1), annot=posterior_std_M_0.reshape(1,1))
plt.title('posterior SW1S [expected SW1 [i]s in color, std overlaid in digits]')

Out[69]: Text(0.5, 1.0, 'posterior SW1S [expected SW1 [i]s in color, std overlaid in digits]')
```

posterior W1 (expected W1, in color, std overlaid in digits)

	0	1	2	3	4	5	6	7	8	9
0	0.53	0.25	0.42	0.44	0.7	0.83	0.47	0.32	1.1	0.29

```
In [70]: f, ax = plt.subplots(figsize=(12,11))
ax = sns.heatmap(posterior_f_b_0.reshape(1,-1),annot=posterior_std_b_0.reshape(1,-1))
plt.title('posterior $b0_0$ (expected $b0_0[i]$, in color, std overlaid in digits)')
```

Out[70]: Text(0.5, 1.0, 'posterior \$b0\_0\$ (expected \$b0\_0[i]\$, in color, std overlaid in digits)')

posterior b0 (expected b0, in color, std overlaid in digits)

	0	1	2	3	4	5	6	7	8	9
0	0.51	0.63	0.42	0.6	0.41	0.57	0.39	0.64	0.47	0.85

In [7]:

```
f, ax = plt.subplots(figsize=(12,12))  
ax = sns heatmap(posterior_F_M_1,anno=posterio_std_W_1  
plt.title('posterior $W_1$ (expected $W_1[i]$\text{) in color, and overlaid in digits'))
```

Out[7]:

Text(0.5, 1.0, 'posterior \$W\_1\$ (expected \$W\_1[i]\$\text{) in color, std overlaid in digits'))

	0	1	2	3	4	5	6	7	8	9
0	0.4	0.31	0.28	0.94	0.4	0.45	0.73	0.58	0.5	0.65
1	0.55	0.5	0.45	0.4	0.35	0.3	0.25	0.2	0.15	0.1
2	0.6	0.55	0.5	0.45	0.4	0.35	0.3	0.25	0.2	0.15
3	0.65	0.6	0.55	0.5	0.45	0.4	0.35	0.3	0.25	0.2
4	0.7	0.65	0.6	0.55	0.5	0.45	0.4	0.35	0.3	0.25
5	0.75	0.7	0.65	0.6	0.55	0.5	0.45	0.4	0.35	0.3
6	0.8	0.75	0.7	0.65	0.6	0.55	0.5	0.45	0.4	0.35
7	0.85	0.8	0.75	0.7	0.65	0.6	0.55	0.5	0.45	0.4
8	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	0.5	0.45
9	0.95	0.9	0.85	0.8	0.75	0.7	0.65	0.6	0.55	0.5

[illegible]

Heatmap showing the correlation matrix of the 10 variables. The color scale ranges from -2.0 (dark purple) to 1.0 (dark red).

	0	1	2	3	4	5	6	7	8	9	
0	1.0	0.3	0.56	0.24	0.27	0.73	0.49	0.07	0.04	0.49	0.27
1	0.32	1.0	0.4	0.56	0.5	0.47	0.29	0.43	0.54	0.59	0.29
2	0.42	1.1	1.0	0.34	0.66	0.32	0.38	0.42	0.5	0.44	0.38
3	0.27	0.56	0.34	1.0	0.63	0.36	0.45	0.38	0.31	0.47	0.27
4	0.73	0.47	0.32	0.63	1.0	0.36	0.45	0.38	0.31	0.47	0.27
5	0.49	0.29	0.38	0.36	0.36	1.0	0.36	0.45	0.38	0.31	0.47
6	0.07	0.43	0.42	0.45	0.38	0.36	1.0	0.36	0.45	0.38	0.31
7	0.04	0.54	0.5	0.38	0.31	0.45	0.38	1.0	0.36	0.45	0.38
8	0.49	0.59	0.44	0.47	0.31	0.38	0.31	0.36	1.0	0.36	0.45
9	0.27	0.29	0.38	0.27	0.47	0.47	0.31	0.45	0.38	1.0	0.36

```
In [72]: f, ax = plt.subplots(figsize=(12,11))
ax = sns.heatmap(posterior_b1.reshape(1,-1).annot=posterior_std_b1.reshape(1,-1))
text('posterior b1$ expected b1.$ in color, std overlaid in digits')

Out[72]: Text(0.5, 1.0, 'posterior b1$ expected b1.$ in color, std overlaid in digits')
```

posterior b1\$ expected b1.\$ in color, std overlaid in digits

Category	Value
0	0.51
1	0.63
2	0.42
3	0.6
4	0.41
5	0.57
6	0.39
7	0.64
8	0.47
9	0.85

0

1

0

We show the heatmaps for brevity, but we have a distribution for each element of the return tensors. In fact, we have samples from the whole posterior and could bin things to get the multidimensional plot of  $p(\{W_{11}\}, \{b_0\}, \{W_{10}\}, \{b_1\})$ , with has  $10 \times 10 \times$

Write a paragraph or two comparing and contrasting mean-field black-box variational inference to parameter estimation via gradient descent.

Mean-field black-box VI desadvantages the parameters of each proposal distribution from the parameters in other distributions, in the gradients. This has the advantage that the gradients don't increase with the complexity of the proposals (their number and dependencies). On the other hand, e.g., p. 132 of the course textbook ([van de Meulen et al., 2027](#)) explains, the variance of the gradient estimator tends to be high, and so many samples per mini batch  $L$  and many iterations  $T$  are needed, more so if the gradient (i.e. program for our forward model) is more complex.

In contrast other methods like variational autoencoders (see Kingma and Welling, 2014; Rezende et al 2014 cited on p. 132 of the course textbook) can have low  $L$ , and as low as  $L=1$ .

An advantage of BBVI is that we don't need to know the analytical forms of the gradients w.r.t the ELBO (i.e. the gradient outside the

expectation over the proposal). Instead, we only need to know the gradient of the distribution parameters being learned/optimized w.r.t. the respective proposal distribution at sampled data points. We know this because we specify the proposals. At the same time, if other methods that make restrictions on the proposals, can leverage the possibility of taking gradients w.r.t. the ELBO and avoid high variance of the gradients. See the [2018 review by Zhang et al](#) for more discussion.