1 Clustering: Mixture of Multinomials(2 points)

1.1 MLE for multinomial(1 point)

Derive the maximum-likelihood estimator for the parameter $\mu = (\mu_i)_{i=1}^d$ of a multinomial distribution:

$$P(x|\mu) = \frac{n!}{\prod_i x_i!} \prod_i \mu_i^{x_i}, \quad i = 1, \cdots, d \tag{1}$$

where $x_i \in \mathbb{N}$, $\sum_i x_i = n$ and $0 < \mu_i < 1$, $\sum_i \mu_i = 1$.

Date. . .

1.1 MLE

$$= \frac{n!}{\Pi_i x_i!} \Pi_i u_i^{x_i}$$

$$= \log\left(\frac{n!}{\Pi_i x_i!} \Pi_i u_i^{x_i}\right)$$

$$= \log(n!) - \log(\Pi_i x_i!) + \log(u_i^{x_i})$$

$$= \log(n!) - \Sigma_i \log(x_i!) + \Sigma_i x_i \log(u_i)$$

S.t $\rightarrow$ $x_i \in N, \Sigma_i x_i = n$

$$= \log(n!) - \Sigma_i \log(x_i!) + n \log(u_i)$$

Lagrange Method $S.T \rightarrow \Sigma_i u_i = 1 \Rightarrow \Sigma_i - 1 = 0$

$$L(u_i, x_i, \lambda) = \log(n!) - \Sigma_i \log(x_i!) + n \log(u_i) - \lambda(\Sigma u_i - 1)$$

$$\frac{dL}{du_i} = 0$$

$$= \frac{d\log(n!)}{du_i} - \frac{d(\Sigma_i \log(x_i!))}{du_i} + \frac{d(n\log(u_i))}{du_i} - d(\lambda(\Sigma u_i - 1))$$

$$= 0 + 0 + \frac{n}{u_i} - \lambda.1 = 0$$

$$= u_i = \frac{n}{\lambda}$$

$S.t \rightarrow \Sigma_i u_i = 1$

$$\frac{n}{\lambda} = 1 \Leftrightarrow n = \lambda$$

$$= u_i = \frac{x_i}{n}$$

$$\boxed{\text{Answer} = u_i = \frac{x_i}{n}}$$

1.2 EM for mixture of multinomials(1 point)

No.

Date.

EM Algorithm

E-Step

For every document(d), use $\lambda_d = p(c_d \mid \pi^{(t)}, \mu_{ik}^{(t)})$

$$p(c_d = k \mid \pi^{(t)}, \mu_{ik}^{(t)}) \; \pi_k \prod_{w=1}^{W} \mu_{k,w}^{n_w(x_d)}$$

M-Step ~~Cluster label distribution:~~

$$\mu_{k,w}^{(t+1)} = \frac{\sum_{d=1}^{D} \lambda_{d,k}\, n_w(x_d)}{\sum_{d=1}^{D} \lambda_{d,k}\, n(x_d)}$$

Cluster Proposition:

$$\pi_k^{(t+1)} = \frac{\sum_{d=1}^{D} \lambda_{d,k}}{D}$$

2 PCA(2 points)

2.1 Minimum Error Formulation(2 points)

No.

Date.

2.1 PCA

$$J = \frac{1}{N}\sum_{n=1}^{N} \| x_n - \sum_{i=1}^{d} z_{ni} u_i + \sum_{i=d+1}^{p} b_i u_i \|^2$$

$$\frac{dJ}{dz_{ni}} = 0 \Leftrightarrow \frac{1}{N}(x_n^T u_i - z_{ni}) = 0$$

Result

$$\boxed{\rightarrow \quad z_{ni} = x_n^T u_i}$$

$$\frac{dJ}{db_i} = 0 \Leftrightarrow \frac{1}{N}\sum_{n=1}^{N}(x_n^T u_i - b_i) = 0 \Leftrightarrow \bar{x}^T u_i - b_i = 0$$

Result

$$\boxed{\rightarrow \quad b_i = \bar{x}^T u_i}$$

3 Reinforcement Learning(1 point)

3.1 Value Iteration(1 point)

R(s,a)	non	light	heavy
non-send	$ 0	$ 20	$ 100
send	$ (-1)	$ 15	$ 80

P(s'|s, a_no-send)	non	light	heavy
non	0.95	0.05	0
light	0.2	0.75	0.05
heavy	0.1	0.2	0.7

P(s'|s, a_send)	non	light	heavy
non	0.1	0.85	0.05
light	0.1	0.2	0.7
heavy	0.05	0.15	0.8

Q Table	non	light	heavy
non-send	-1	10	50
send	-2	35	40

(After 1 Iteration)

Q Table	non	light	heavy
non-send	0.72	45.695	137.71
send	27.935	52.72	120.68

Calculation = 0 + 0.9 x (0.95 x (-1) + 0.05 x (35) + 0 x (50))

R(s,a)	non	light	heavy
non-send	$ 0	$ 20	$ 100
send	$ (-1)	$ 15	$ 80

P(s'|s, a_no-send)	non	light	heavy
non	0.95	0.05	0
light	0.2	0.75	0.05
heavy	0.1	0.2	0.7

P(s'|s, a_send)	non	light	heavy
non	0.1	0.85	0.05
light	0.1	0.2	0.7
heavy	0.05	0.15	0.8

Q Table	non	light	heavy
non-send	-1	10	50
send	-2	35	40

(After 1 Iteration)

Q Table	non	light	heavy
non-send	0.72	45.695	137.71
send	27.935	52.72	120.68

Calculation = 20 + 0.9 x (0.2 x (-1) + 0.75 x (35) + 0.05 x (50))

R(s,a)	non	light	heavy
non-send	$ 0	$ 20	$ 100
send	$ (-1)	$ 15	$ 80

P(s'|s, a_no-send)	non	light	heavy
non	0.95	0.05	0
light	0.2	0.75	0.05
heavy	0.1	0.2	0.7

P(s'|s, a_send)	non	light	heavy
non	0.1	0.85	0.05
light	0.1	0.2	0.7
heavy	0.05	0.15	0.8

Q Table	non	light	heavy
non-send	-1	10	50
send	-2	35	40

(After 1 Iteration)

Q Table	non	light	heavy
non-send	0.72	45.695	137.71
send	27.935	52.72	120.68

Calculation = 15 + 0.9 x (0.1 x (-1) + 0.2 x (35) + 0.7 x (50))

HW2 Machine Learning Student ID: 2019280513 Student Name: Sami Emre Erdogan

R(s,a)	non	light	heavy
non-send	$ 0	$ 20	$ 100
send	$ (-1)	$ 15	$ 80

P(s'|s, a_no-send)	non	light	heavy
non	0.95	0.05	0
light	0.2	0.75	0.05
heavy	0.1	0.2	0.7

P(s'|s, a_send)	non	light	heavy
non	0.1	0.85	0.05
light	0.1	0.2	0.7
heavy	0.05	0.15	0.8

Q Table	non	light	heavy
non-send	-1	10	50
send	-2	35	40

(After 1 Iteration)

Q Table	non	light	heavy
non-send	0.72	45.695	137.71
send	27.935	52.72	120.68

Calculation = 100 + 0.9 x (0.1 x (-1) + 0.2 x (35) + 0.7 x (50))

R(s,a)	non	light	heavy
non-send	$ 0	$ 20	$ 100
send	$ (-1)	$ 15	$ 80

P(s'|s, a_no-send)	non	light	heavy
non	0.95	0.05	0
light	0.2	0.75	0.05
heavy	0.1	0.2	0.7

P(s'|s, a_send)	non	light	heavy
non	0.1	0.85	0.05
light	0.1	0.2	0.7
heavy	0.05	0.15	0.8

Q Table	non	light	heavy
non-send	-1	10	50
send	-2	35	40

(After 1 Iteration)

Q Table	non	light	heavy
non-send	0.72	45.695	137.71
send	27.935	52.72	120.68

Calculation = 80 + 0.9 x (0.05 x (-1) + 0.15 x (35) + 0.8 x (50))

4 Deep Generative Models: Class-conditioned VAE(5 points)

Findings

I have found that after comparing 10 and 20 epochs. 20 was clearer on the edges and more understandable.

Here are the comparison photos:

10

20

output of test lower bound and Log Likelihood:

```
Epoch 1 (14.2s): Lower bound = -173.15237426757812
Epoch 2 (13.1s): Lower bound = -125.95645904541016
Epoch 3 (13.1s): Lower bound = -115.4300537109375
Epoch 4 (13.3s): Lower bound = -111.14623260498047
Epoch 5 (17.0s): Lower bound = -108.410400390625
Epoch 6 (18.3s): Lower bound = -106.51114654541016
Epoch 7 (18.1s): Lower bound = -105.159423828125
Epoch 8 (13.0s): Lower bound = -104.07119750976562
Epoch 9 (13.5s): Lower bound = -103.14167022705078
Epoch 10 (13.5s): Lower bound = -102.26787567138672
>>> TEST (379.8s)
>> Test lower bound = -101.43687438964844
>> Test log likelihood (IS) = -96.09476470947266
Epoch 11 (14.1s): Lower bound = -101.62956237792969
Epoch 12 (13.1s): Lower bound = -100.99659729003906
Epoch 13 (12.9s): Lower bound = -100.55213165283203
Epoch 14 (13.1s): Lower bound = -100.00975036621094
Epoch 15 (12.9s): Lower bound = -99.74927520751953
Epoch 16 (12.9s): Lower bound = -99.3624267578125
Epoch 17 (12.8s): Lower bound = -99.09620666503906
Epoch 18 (12.9s): Lower bound = -98.78494262695312
Epoch 19 (13.0s): Lower bound = -98.54756164550781
Epoch 20 (13.2s): Lower bound = -98.28917694091797
>>> TEST (402.6s)
>> Test lower bound = -98.56088256835938
>> Test log likelihood (IS) = -93.06107330322266
```

Appendix Code

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from __future__ import absolute_import
from __future__ import print_function
from __future__ import division
import os
import time

import tensorflow as tf
from six.moves import range
import numpy as np
import zhusuan as zs

from examples import conf
from examples.utils import dataset, save_image_collections


@zs.meta_bayesian_net(scope="gen", reuse_variables=True)
def build_gen(x_dim, z_dim, n, n_particles=1):
    bn = zs.BayesianNet()
    z_mean = tf.zeros([n, z_dim])
    z = bn.normal("z", z_mean, std=1., group_ndims=1, n_samples=n_particles)
    h = tf.layers.dense(z, 500, activation=tf.nn.relu)
    h = tf.layers.dense(h, 500, activation=tf.nn.relu)
    x_logits = tf.layers.dense(h, x_dim)
    bn.deterministic("x_mean", tf.sigmoid(x_logits))
    bn.bernoulli("x", x_logits, group_ndims=1)
    return bn


@zs.reuse_variables(scope="q_net")
def build_q_net(x, z_dim, n_z_per_x):
    bn = zs.BayesianNet()
    h = tf.layers.dense(tf.cast(x, tf.float32), 500, activation=tf.nn.relu)
    h = tf.layers.dense(h, 500, activation=tf.nn.relu)
    z_mean = tf.layers.dense(h, z_dim)
    z_logstd = tf.layers.dense(h, z_dim)
    bn.normal("z", z_mean, logstd=z_logstd, group_ndims=1, n_samples=n_z_per_x)
    return bn


def main():
    # Load MNIST
    data_path = os.path.join(conf.data_dir, "mnist.pkl.gz")
    x_train, t_train, x_valid, t_valid, x_test, t_test = \
```

```
    dataset.load_mnist_realval(data_path)
x_train = np.vstack([x_train, x_valid])
x_test = np.random.binomial(1, x_test, size=x_test.shape)
x_dim = x_train.shape[1]

# Define model parameters
z_dim = 40

# Build the computation graph
n_particles = tf.placeholder(tf.int32, shape=[], name="n_particles")
x_input = tf.placeholder(tf.float32, shape=[None, x_dim], name="x")
x = tf.cast(tf.less(tf.random_uniform(tf.shape(x_input)), x_input),
        tf.int32)
n = tf.placeholder(tf.int32, shape=[], name="n")

model = build_gen(x_dim, z_dim, n, n_particles)
variational = build_q_net(x, z_dim, n_particles)

lower_bound = zs.variational.elbo(
    model, {"x": x}, variational=variational, axis=0)
cost = tf.reduce_mean(lower_bound.sgvb())
lower_bound = tf.reduce_mean(lower_bound)

# # Importance sampling estimates of marginal log likelihood
is_log_likelihood = tf.reduce_mean(
    zs.is_loglikelihood(model, {"x": x}, proposal=variational, axis=0))

optimizer = tf.train.AdamOptimizer(learning_rate=0.001)
infer_op = optimizer.minimize(cost)

# Random generation
x_gen = tf.reshape(model.observe()["x_mean"], [-1, 28, 28, 1])

# Define training/evaluation parameters
epochs = 20
batch_size = 128
iters = x_train.shape[0] // batch_size
save_freq = 10
test_freq = 10
test_batch_size = 400
test_iters = x_test.shape[0] // test_batch_size
result_path = "results/vae"

# Run the inference
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
```

```
    for epoch in range(1, epochs + 1):
        time_epoch = -time.time()
        np.random.shuffle(x_train)
        lbs = []
        for t in range(iters):
            x_batch = x_train[t * batch_size:(t + 1) * batch_size]
            _, lb = sess.run([infer_op, lower_bound],
                             feed_dict={x_input: x_batch,
                                        n_particles: 1,
                                        n: batch_size})
            lbs.append(lb)
        time_epoch += time.time()
        print("Epoch {} ({:.1f}s): Lower bound = {}".format(
            epoch, time_epoch, np.mean(lbs)))

        if epoch % test_freq == 0:
            time_test = -time.time()
            test_lbs, test_lls = [], []
            for t in range(test_iters):
                test_x_batch = x_test[t * test_batch_size:
                                      (t + 1) * test_batch_size]
                test_lb = sess.run(lower_bound,
                                   feed_dict={x: test_x_batch,
                                              n_particles: 1,
                                              n: test_batch_size})
                test_ll = sess.run(is_log_likelihood,
                                   feed_dict={x: test_x_batch,
                                              n_particles: 1000,
                                              n: test_batch_size})
                test_lbs.append(test_lb)
                test_lls.append(test_ll)
            time_test += time.time()
            print(">>> TEST ({:.1f}s)".format(time_test))
            print(">> Test lower bound = {}".format(np.mean(test_lbs)))
            print('>> Test log likelihood (IS) = {}'.format(
                np.mean(test_lls)))

        if epoch % save_freq == 0:
            images = sess.run(x_gen, feed_dict={n: 100, n_particles: 1})
            name = os.path.join(result_path,
                                "vae.epoch.{}.png".format(epoch))
            save_image_collections(images, name)

if __name__ == "__main__":
    main()
```