CS561 HW13

December 2, 2023

1 Problem 1

The pmf of the multinomial distribution $(X_1, X_2, \dots, X_K) \sim \text{Multi}(\mathbf{p})$ with multinomial parameter $\mathbf{p} = (p_1, p_2, \dots, p_k)$ is given as

$$P(X_i = k_i : i = 1, 2, \dots, K) = \frac{n!}{k_1! k_2! \dots k_K!} p_1^{k_1} p_2^{k_2} \dots p_K^{k_K}$$

We can derive the KL-divergence between $\text{Multi}(\mathbf{p})$ and $\text{Multi}(\mathbf{q})$ as follows:

$$\begin{aligned}
D_{KL}(\text{Multi}(\mathbf{p}) || \text{Multi}(\mathbf{q})) &= \sum_{k_1+k_2+\dots+k_K=n} \frac{n!}{k_1! k_2! \dots k_K!} p_1^{k_1} p_2^{k_2} \dots p_K^{k_K} \log\left(\frac{p_1^{k_1} p_2^{k_2} \dots p_K^{k_K}}{q_1^{k_1} q_2^{k_2} \dots q_K^{k_K}}\right) \\
&= \sum_{k_1+k_2+\dots+k_K=n} \frac{n!}{k_1! k_2! \dots k_K!} p_1^{k_1} p_2^{k_2} \dots p_K^{k_K} \sum_{i=1}^{K} k_i \log\left(\frac{p_i}{q_i}\right) \\
&= \sum_{i=1}^{K} \log\left(\frac{p_i}{q_i}\right) \sum_{k_1+k_2+\dots+k_K=n} k_i \frac{n!}{k_1! k_2! \dots k_K!} p_1^{k_1} p_2^{k_2} \dots p_K^{k_K} \\
&= \sum_{i=1}^{K} \log\left(\frac{p_i}{q_i}\right) \mathbb{E}[X_i : (X_1, X_2, \dots, X_K) \sim \text{Multi}(\mathbf{p})] \\
&= \sum_{i=1}^{K} \log\left(\frac{p_i}{q_i}\right) n p_i \\
&= n \sum_{i=1}^{K} p_i \log\left(\frac{p_i}{q_i}\right) \\
&= n D_{KL}(\mathbf{p} || \mathbf{q})
\end{aligned}$$

2 Problem 2

[3]:
```
import numpy as np
import tensorflow as tf
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
print(np.shape(x_train))
```

```python
def vectorize(_image):
    return np.reshape(_image, (-1,1))

vec_x_train = np.squeeze(np.array([vectorize(m) for m in x_train]))
vec_x_test = np.squeeze(np.array([vectorize(m) for m in x_test]))

# generate an instance of the logistic regression class model with multinomial␣
↪logistic regression
model = LogisticRegression(solver='saga', tol=0.01, multi_class='multinomial')
model.fit(vec_x_train, y_train)
```

```
(60000, 28, 28)
```

[3]:
```
LogisticRegression(multi_class='multinomial', solver='saga', tol=0.01)
```

[5]:
```python
from sklearn.metrics import classification_report

### compute the accuracy and print a classification report
y_train_hat = model.predict(vec_x_train)
y_test_hat = model.predict(vec_x_test)
print("Train")
print(classification_report(y_train_hat, y_train))
print("Test")
print(classification_report(y_test_hat, y_test))
```

```
Train
              precision    recall  f1-score   support

           0       0.98      0.97      0.97      5972
           1       0.98      0.97      0.97      6822
           2       0.92      0.94      0.93      5856
           3       0.91      0.92      0.92      6099
           4       0.94      0.94      0.94      5853
           5       0.89      0.91      0.90      5283
           6       0.97      0.96      0.96      5994
           7       0.94      0.95      0.95      6199
           8       0.91      0.90      0.90      5896
           9       0.92      0.91      0.92      6026

    accuracy                           0.94     60000
   macro avg       0.94      0.94      0.94     60000
weighted avg       0.94      0.94      0.94     60000

Test
              precision    recall  f1-score   support
```

```
           0       0.98      0.95      0.97      1010
           1       0.98      0.96      0.97      1161
           2       0.90      0.93      0.91       995
           3       0.91      0.90      0.91      1024
           4       0.93      0.93      0.93       983
           5       0.86      0.91      0.89       850
           6       0.95      0.95      0.95       963
           7       0.92      0.93      0.93      1021
           8       0.88      0.87      0.88       985
           9       0.91      0.91      0.91      1008

    accuracy                           0.93     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.93      0.93      0.93     10000
```

[6]:

```
from sklearn.preprocessing import OneHotEncoder

# convert the 10 classes to one hot encoding
one_hot = OneHotEncoder()
Y_train = one_hot.fit_transform(y_train.reshape(-1,1)).toarray()
Y_test = one_hot.fit_transform(y_test.reshape(-1,1)).toarray()
print(np.shape(Y_train))
```

```
(60000, 10)
```

[18]:

```
import keras
from keras.models import Sequential
from keras.layers import Dense

model = Sequential()
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
history = model.fit(vec_x_train, Y_train, batch_size=32, epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 2s 1ms/step - loss: 9.9260 -
accuracy: 0.8382
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 6.0376 -
accuracy: 0.8794
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.7018 -
accuracy: 0.8839
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.5749 -
accuracy: 0.8853
```

```
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.3354 -
accuracy: 0.8875
Epoch 6/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.3640 -
accuracy: 0.8880
Epoch 7/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.2025 -
accuracy: 0.8895
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.1955 -
accuracy: 0.8895
Epoch 9/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.1506 -
accuracy: 0.8888
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 5.1100 -
accuracy: 0.8908
```

[19]:
```python
from sklearn.metrics import classification_report

Y_train_hat = model.predict(vec_x_train)
y_train_hat = Y_train_hat.argmax(-1)
Y_test_hat = model.predict(vec_x_test)
y_test_hat = Y_test_hat.argmax(-1)
print("Train")
print(classification_report(y_train_hat, y_train))
print("Test")
print(classification_report(y_test_hat, y_test))
```

```
1875/1875 [==============================] - 2s 1ms/step
313/313 [==============================] - 0s 1ms/step
Train
              precision    recall  f1-score   support

           0       0.98      0.95      0.96      6118
           1       0.94      0.99      0.96      6423
           2       0.88      0.94      0.91      5538
           3       0.89      0.89      0.89      6093
           4       0.93      0.86      0.89      6367
           5       0.85      0.87      0.86      5273
           6       0.96      0.93      0.95      6117
           7       0.95      0.88      0.92      6752
           8       0.90      0.82      0.86      6439
           9       0.75      0.91      0.82      4880

    accuracy                           0.90     60000
   macro avg       0.90      0.90      0.90     60000
```

	precision	recall	f1-score	support
weighted avg	0.91	0.90	0.90	60000

Test

	precision	recall	f1-score	support
0	0.97	0.94	0.96	1014
1	0.95	0.98	0.97	1104
2	0.86	0.94	0.90	945
3	0.88	0.89	0.88	996
4	0.92	0.84	0.88	1076
5	0.83	0.86	0.84	859
6	0.94	0.92	0.93	989
7	0.94	0.87	0.90	1105
8	0.90	0.80	0.84	1101
9	0.74	0.91	0.82	811
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.90	0.89	0.89	10000

2.c) The neural network model is an interative variant to the logistic regression model. Therefore, the neural network takes faster to train, but performs worse overall due to the logistic regression model fitting the parameter with the whole dataset.