SA22011021 Yi-Xiang Hu
School of Computer Science and Technology
University of Science and Technology of China

Exercise Sheet 2 for

Applied Mathematic for Computer Science and Technology

Autumn 2022

Due 8 Dec 2022 at 23:59

Exercise 1

$$\begin{cases} x_1 - 2x_2 + 3x_3 - x_4 - x_5 = 2 \\ x_1 + x_2 - x_3 + x_4 - 2x_5 = 1 \\ 2x_1 - x_2 + x_3 - 0 - 2x_5 = 2 \\ 2x_1 + 2x_2 - 5x_3 + 2x_4 - x_5 = 5 \end{cases}$$

$$AX = b$$

$$[A|b] = \begin{bmatrix} 1 & -2 & 3 & -1 & -1 & 2 \\ 1 & 1 & -1 & 1 & -2 & 1 \\ 2 & -1 & 1 & 0 & -2 & 2 \\ 2 & 2 & -5 & 2 & -1 & 5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 3 & -1 & -1 & 2 \\ 0 & 3 & -4 & 2 & -1 & -1 \\ 0 & 3 & -5 & 2 & 0 & -2 \\ 0 & 6 & -11 & 4 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 3 & -1 & -1 & 2 \\ 0 & 1 & -\frac{4}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & -1 & 0 & 1 & -1 \\ 0 & 0 & -3 & 0 & 3 & 3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 3 & -1 & -1 & 2 \\ 0 & 1 & -\frac{4}{3} & \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 6 \end{bmatrix}$$

增广矩阵的先导元素出现在了最后一列，故无解。

Exercise 2

$$\begin{cases} x_1+x_2+x_3+x_4+x_5=1 \\ 3x_1+2x_2+x_3+x_4-3x_5=a \\ x_2+2x_3+2x_4+6x_5=3 \\ 5x_1+4x_2+3x_3+3x_4-x_5=b \end{cases}$$

增广矩阵为:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 1 & -3 & a \\ 0 & 1 & 2 & 2 & 6 & 3 \\ 5 & 4 & 3 & 3 & -1 & b \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 1 & 0 & 0 & -4 & a-1 \\ -2 & -1 & 0 & 0 & 4 & 1 \\ 2 & 1 & 0 & 0 & -4 & b-3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & a \\ -2 & -1 & 0 & 0 & 4 & 1 \\ 0 & 0 & 0 & 0 & 0 & b-2 \end{bmatrix}$$

故

$$\begin{cases} a=0 \\ b=2 \end{cases}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 0 & 4 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 6 & 3 \end{bmatrix}$$

$$\begin{cases} x_1 = x_3+x_4+5x_5-2 \\ x_2 = -2x_3-2x_4-6x_5+3 \\ x_3,\ x_4,\ x_5 \in \mathrm{R} \end{cases}$$

Exercise 3

四个观测点，分别为 $(1,1),(2,1),(3,2),(3,3)$，用直线 $y=ax+b$ 拟合。

一般解法:

$$\begin{cases} a = \dfrac{n\sum_{i=1}^n x_iy_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n\sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i\right)^2} \\ b = \dfrac{1}{n}\displaystyle\sum_{i=1}^n y_i - \dfrac{a}{n}\sum_{i=1}^n x_i \end{cases}$$

$$\sum_{i=1}^n x_iy_i = 18$$

$$\sum_{i=1}^{n} x_i = 9$$

$$\sum_{i=1}^{n} y_i = 7$$

$$\sum_{i=1}^{n} x_i^2 = 23$$

$$\begin{cases} a = \dfrac{9}{11} \\ b = -\dfrac{1}{11} \end{cases}$$

矩阵解法*:*

$$X = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 3 & 1 \end{bmatrix}$$

$$y = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{bmatrix}$$

$$\begin{aligned}
c &= \left(X^T X\right)^{-1} X^T y \\
&= \left(\begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 3 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} \\
&= \begin{bmatrix} 23 & 9 \\ 9 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} \\
&= \frac{1}{11} \begin{bmatrix} 4 & -9 \\ -9 & 23 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \frac{1}{11} \begin{bmatrix} 9 \\ -1 \end{bmatrix}
\end{aligned}$$

Exercise 4

现有如下数据（*4* 个维度），用 *PCA* 将数据降到 *2* 维。

A	B	C	D
1	5	3	1
4	2	6	3
1	4	3	2
4	4	1	1
5	5	2	3

样例:

$$X = \begin{bmatrix} 1 & 4 & 1 & 4 & 5 \\ 5 & 2 & 4 & 4 & 5 \\ 3 & 6 & 3 & 1 & 2 \\ 1 & 3 & 2 & 1 & 3 \end{bmatrix}$$

零均值化后:

$$X = \begin{bmatrix} -2 & 1 & -2 & 1 & 2 \\ 1 & -2 & 0 & 0 & 1 \\ 0 & 3 & 0 & -2 & -1 \\ -1 & 1 & 0 & -1 & 1 \end{bmatrix}$$

协方差矩阵:

$$\begin{aligned} C &= \frac{1}{5}XX^T \\ &= \frac{1}{5}\begin{bmatrix} -2 & 1 & -2 & 1 & 2 \\ 1 & -2 & 0 & 0 & 1 \\ 0 & 3 & 0 & -2 & -1 \\ -1 & 1 & 0 & -1 & 1 \end{bmatrix}\begin{bmatrix} -2 & 1 & 0 & -1 \\ 1 & -2 & 3 & 1 \\ -2 & 0 & 0 & 0 \\ 1 & 0 & -2 & -1 \\ 2 & 1 & -1 & 1 \end{bmatrix} \\ &= \frac{1}{5}\begin{bmatrix} 14 & -2 & -1 & 4 \\ -2 & 6 & -7 & -2 \\ -1 & -7 & 14 & 4 \\ 4 & -2 & 4 & 4 \end{bmatrix} \end{aligned}$$

使用 *Python* 中的 *numpy* 库，计算 C 的特征值及特征向量，得到

$$\lambda_1 = 3.9391182,\ \lambda_2 = 3.02968804,\ \lambda_3 = 0.13513162,\ \lambda_4 = 0.49606215$$

$$c_1 = \begin{bmatrix} -0.26087912 & 0.91114239 & -0.30581868 & -0.09075555 \end{bmatrix}^T$$

$$c_2 = \begin{bmatrix} 0.4801165 & 0.0378551 & -0.5085857 & 0.71371964 \end{bmatrix}^T$$

$$c_3 = \begin{bmatrix} -0.77212094 & -0.3639535 & -0.48334366 & 0.19428332 \end{bmatrix}^T$$

$$c_4 = \begin{bmatrix} -0.32443718 & 0.18953731 & 0.64357939 & 0.66679959 \end{bmatrix}^T$$

选取特征值较大的两个作为矩阵 P:

$$P = \begin{bmatrix} -0.26087912 & 0.91114239 & -0.30581868 & -0.09075555 \\ 0.4801165 & 0.0378551 & -0.5085857 & 0.71371964 \end{bmatrix}$$

降维到 2 维后的数据为:

$$Y = PX = \begin{bmatrix} 3.28662124 & -1.32841043 & 2.2847233 & 2.20447885 & 2.36741234 \\ -0.14264546 & 1.08582092 & 0.53321908 & 2.27702034 & 3.71384552 \end{bmatrix}$$

Exercise 5

源代码:

```
from sklearn.preprocessing import OrdinalEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
import pandas as pd
from sklearn.decomposition import PCA
import torch.nn as nn
import matplotlib.pyplot as plt
import torch


class AutoEncoder(nn.Module):
    def __init__(self):
        super(AutoEncoder, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(41, 32),
            nn.ReLU(),
            nn.Linear(32, 8),
            nn.ReLU(),
            nn.Linear(8, n_components)
        )
        self.decoder = nn.Sequential(
            nn.Linear(n_components, 8),
            nn.ReLU(),
            nn.Linear(8, 32),
            nn.ReLU(),
            nn.Linear(32, 41),
            nn.Sigmoid()
        )

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)
        return encoded, decoded
```

```python
# load data
train_file = 'D:\\AMCS_2022\\data\\kddcup99_train.csv'  # training set
df = pd.read_csv(train_file, header=None)
df[41] = df[41].apply(lambda x: 0 if x == "normal." else 1)
X_train = df.iloc[:, :-1]
y_train = df.iloc[:, -1]

test_file = 'D:\\AMCS_2022\\data\\kddcup99_test.csv'  # test set
df = pd.read_csv(test_file, header=None)
df[41] = df[41].apply(lambda x: 0 if x == "normal." else 1)
X_test = df.iloc[:, :-1]
y_test = df.iloc[:, -1]

X = pd.concat([X_test, X_train])

# data preprocessing

OrdEnc = OrdinalEncoder()
X = OrdEnc.fit_transform(X)
X = pd.DataFrame(X)

method = "pca"
# method = "autoencoder"
for n in [2, 3, 4]:
    n_components = n
    print(n_components)
    if method == "pca":
        pca = PCA(n_components=n_components, random_state=42)
        result = pca.fit_transform(X)

        X_test = result[:len(X_test)]
        X_train = result[len(X_test):]

        rf_clf = RandomForestClassifier(criterion="entropy")
        rf_clf.fit(X_train, y_train)

        y_predict = rf_clf.predict(X_test)
        print(accuracy_score(y_test, y_predict))
        X_test_d = pd.DataFrame(X_test)
        X_train_d = pd.DataFrame(X_train)
        if n == 2:
```

```
            fig = plt.figure(figsize=(30, 10))
            ax = fig.add_subplot(131)
            ax.scatter(X_train_d[:][0], X_train_d[:][1], c=y_train[:])
            ax.set_title("training set")
            bx = fig.add_subplot(132)
            bx.scatter(X_test_d[:][0], X_test_d[:][1], c=y_test[:])
            bx.set_title("truth on test set")
            cx = fig.add_subplot(133)
            cx.scatter(X_test_d[:][0], X_test_d[:][1], c=y_predict[:])
            cx.set_title("predict on test set")

            plt.savefig('D:\\AMCS_2022\\result\\pca2.png')

    elif method == "autoencoder":
        X_test = X.iloc[:len(X_test)]
        X_train = X.iloc[len(X_test):]

        Coder = AutoEncoder()
        EPOCH = 2
        BATCH_SIZE = 16
        LR = 0.005
        N_TEST_IMG = 5
        print(Coder)
        optimizer = torch.optim.Adam(Coder.parameters(), lr=LR)
        loss_func = nn.MSELoss()
        for epoch in range(EPOCH):
            for i in range(len(X_train)):
                tensor_t = (torch.from_numpy(X_train.iloc[i, :].values)).t()
                tensor_t = tensor_t.float()
                b_x = tensor_t
                b_y = tensor_t
                encoded, decoded = Coder(b_x)
                loss = loss_func(decoded, b_y)
                optimizer.zero_grad()
                loss.backward()
                optimizer.step()
        torch.save(Coder, 'AutoEncoder'+str(n_components)+'.pkl')
        print('________________________________________')
        print('finish training')

        Coder = torch.load('AutoEncoder'+str(n_components)+'.pkl')
```

```python
X_t = []
for i in range(len(X_train)):
    tensor_t = (torch.from_numpy(X_train.iloc[i, :].values)).t()
    tensor_t = tensor_t.float()
    encoded, _ = Coder(tensor_t)
    X_t.append(encoded.detach().numpy())

X_train_d = pd.DataFrame(X_t)
print("[Start fitting...]")
rf_clf = RandomForestClassifier(criterion="entropy")
rf_clf.fit(X_train_d, y_train)
print("[Finish fitting...]")

X_te = []
for i in range(len(X_test)):
    tensor_t = (torch.from_numpy(X_test.iloc[i, :].values)).t()
    tensor_t = tensor_t.float()
    encoded, _ = Coder(tensor_t)
    X_te.append(encoded.detach().numpy())
    if i % 1000000 == 0:
        print("processing data:", i/(1.0*len(X_test)))
X_test_d = pd.DataFrame(X_te)
y_predict = rf_clf.predict(X_test_d)
print(accuracy_score(y_test, y_predict))

if n == 2:
    fig = plt.figure(figsize=(30, 10))
    ax = fig.add_subplot(131)
    ax.scatter(X_train_d[:][0], X_train_d[:][1], c=y_train[:])
    ax.set_title("training set")
    bx = fig.add_subplot(132)
    bx.scatter(X_test_d[:][0], X_test_d[:][1], c=y_test[:])
    bx.set_title("truth on test set")
    cx = fig.add_subplot(133)
    cx.scatter(X_test_d[:][0], X_test_d[:][1], c=y_predict[:])
    cx.set_title("predict on test set")

    plt.savefig('D:\\AMCS_2022\\result\\auto2.png')
```

结果*:*

本题使用的分类器模型为随机森林，分别使用 *PCA* 和 *AutoEncoder* 进行降维，后进行分类。

	2-D	3-D	4-D
Accuracy(PCA)	0.9993	0.9999	0.9999
Accuracy(AutoEncoder)	0.9958	0.9982	0.9995

表 1: 不同降维方法降低到不同维度对随机森林分类器的性能影响

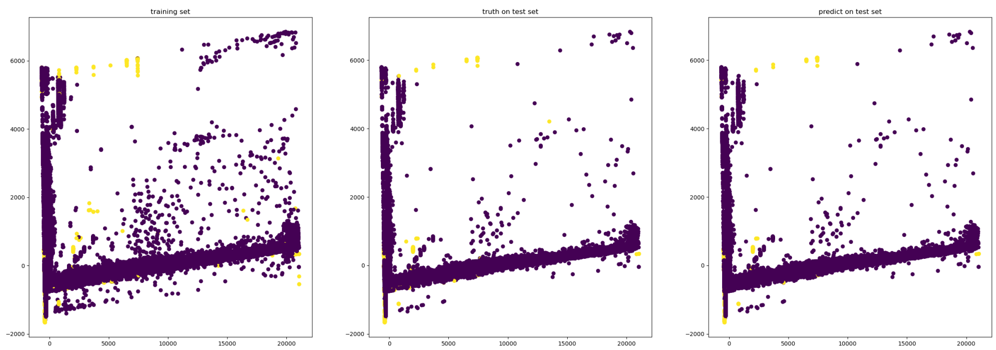

图 1: 用 PCA 降维到二维之后的结果

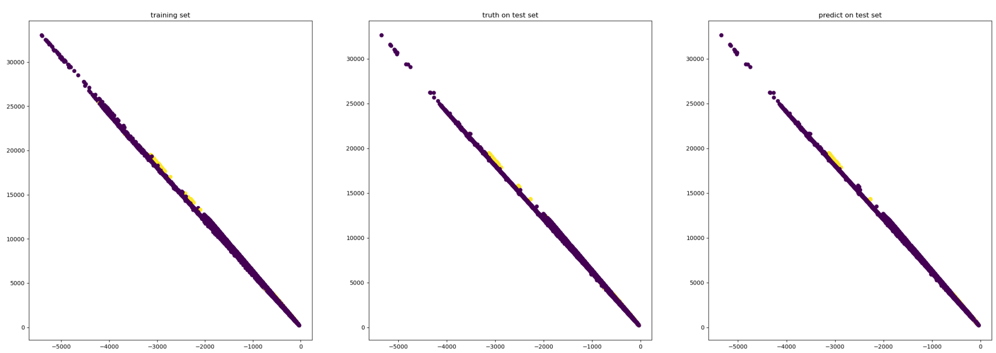

图 2: 用 AutoEncoder 降维到二维之后的结果

Exercise 6

源代码:

```
import tensorflow as tf
import numpy as np
import pandas as pd
from tensorflow import keras
from sklearn.preprocessing import OrdinalEncoder
import pandas as pd
```

```
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# load data
train_file = 'D:\\AMCS_2022\\data\\kddcup99_train.csv'  # training set
df = pd.read_csv(train_file, header=None)
df[41]=df[41].apply(lambda x:0 if x=="normal." else 1)
X_train = df.iloc[:, :-1]
y_train = df.iloc[:, -1]

test_file = 'D:\\AMCS_2022\\data\\kddcup99_test.csv'  # test set
df = pd.read_csv(test_file, header=None)
df[41]=df[41].apply(lambda x:0 if x=="normal." else 1)
X_test = df.iloc[:, :-1]
y_test = df.iloc[:, -1]


X = pd.concat([X_test, X_train])

# data preprocessing

OrdEnc = OrdinalEncoder()
X = OrdEnc.fit_transform(X)
X = pd.DataFrame(X)
X_test = X.iloc[:len(X_test)]
X_train = X.iloc[len(X_test):]
# WIP

# initialization="uniform"
initialization="normal"
optimizer="SGD"
# optimizer="adam"
model = Sequential([
  Dense(41, activation='relu',kernel_initializer=initialization),
  Dense(36, activation='softmax',kernel_initializer=initialization),
  Dense(24, activation='relu',kernel_initializer=initialization),
  Dense(12, activation='relu',kernel_initializer=initialization),
  Dense(6, activation='softmax',kernel_initializer=initialization),
  Dense(1, activation='softmax',kernel_initializer=initialization),
])
```

```
model.compile(
  optimizer=optimizer,
  loss='binary_crossentropy',
  metrics=['binary_accuracy'],
)


history=model.fit(
  X_train, # training data
  y_train, # training targets
  epochs=3,
  batch_size=32,
  validation_data=(X_test,y_test)
)

model.summary()
print("initialization: ",initialization)
print("optimizer :",optimizer)
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.savefig("loss_"+initialization+"_"+optimizer+"_rs2r2s"+".png")
plt.show()
```

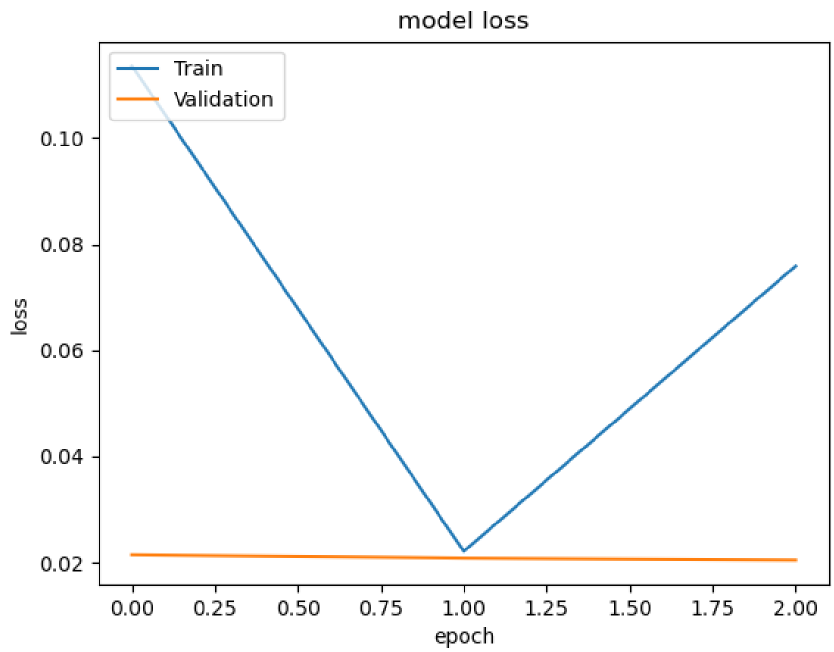

图 3: 层数为 6，41→36→24→12→6→1，前两层为 ReLU，后两层为 softmax，loss 函数为 binary_crossentropy，参数初始化方法为 normal，训练方法为 adam

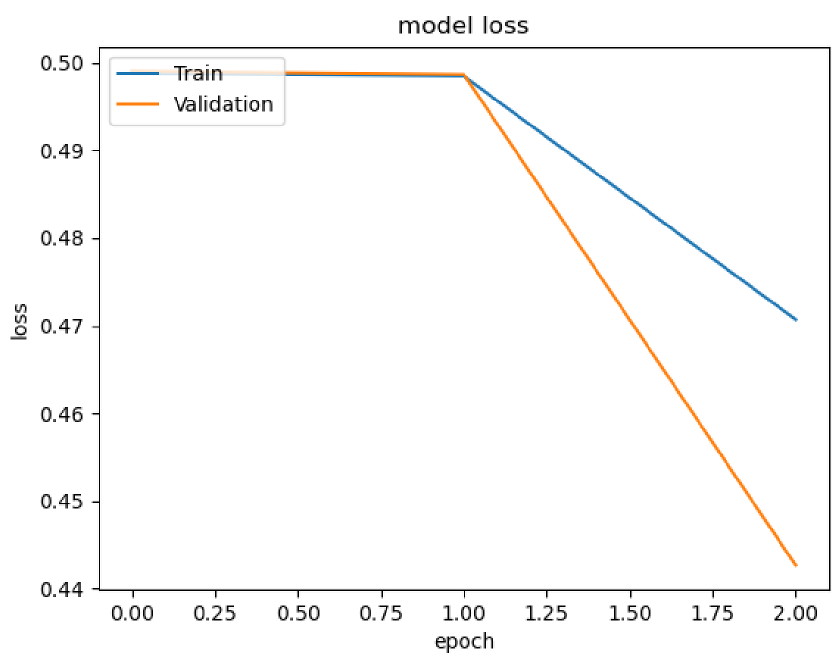

图 4: 层数为 6，41→36→24→12→6→1，1,3,4 层为 ReLU，2.5.6 层为 softmax，loss 函数为 binary_crossentropy，参数初始化方法为 normal，训练方法为 SGD

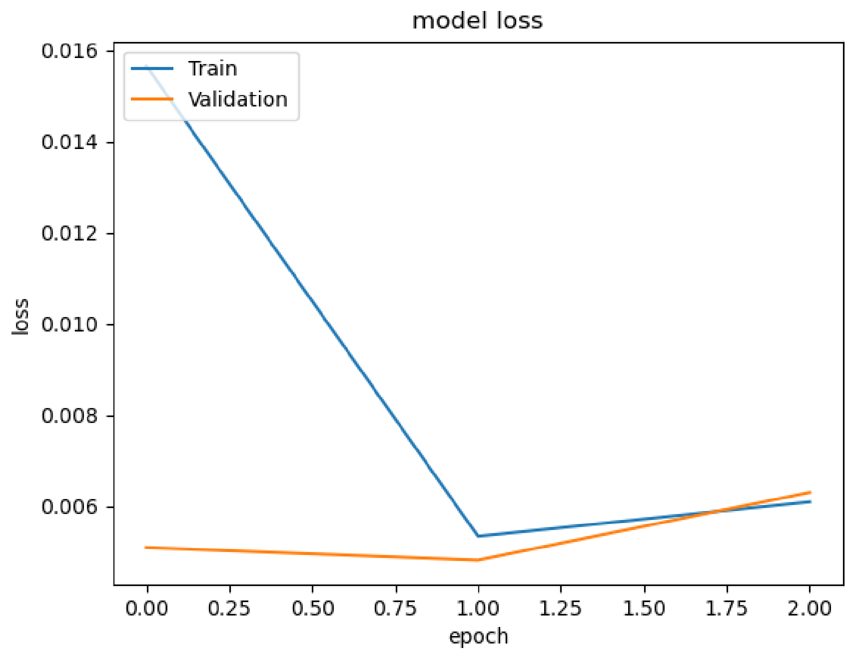

图 5: 层数为 6，41→36→24→12→6→1，前 4 层为 ReLU，5.6 层为 softmax，loss 函数为 binary_crossentropy，参数初始化方法为 uniform，训练方法为 adam

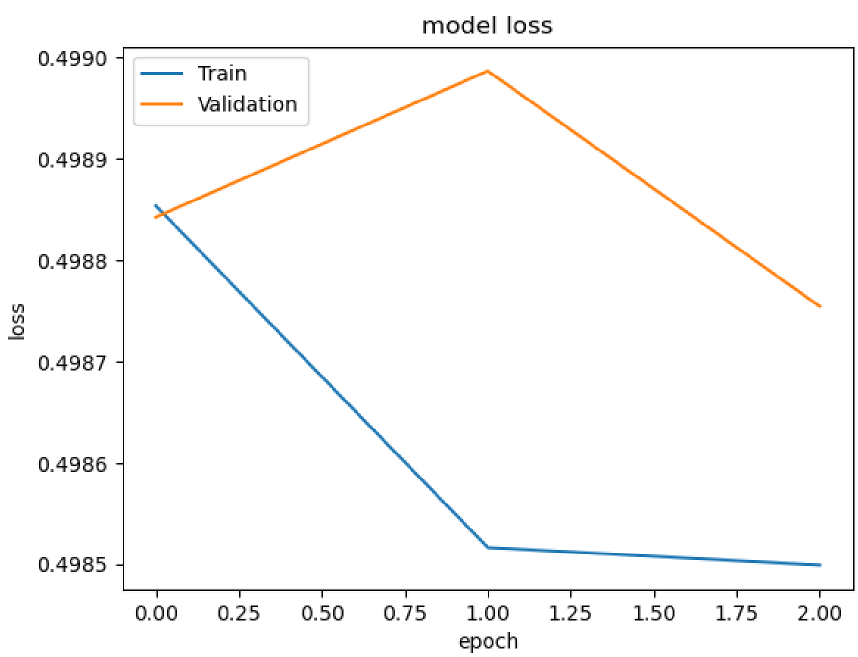

图 6: 层数为 6，41→36→24→12→6→1，前 2 层为 ReLU，后 4 层为 softmax，loss 函数为 binary_crossentropy，参数初始化方法为 uniform，训练方法为 SGD