دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس یادگیری ماشین

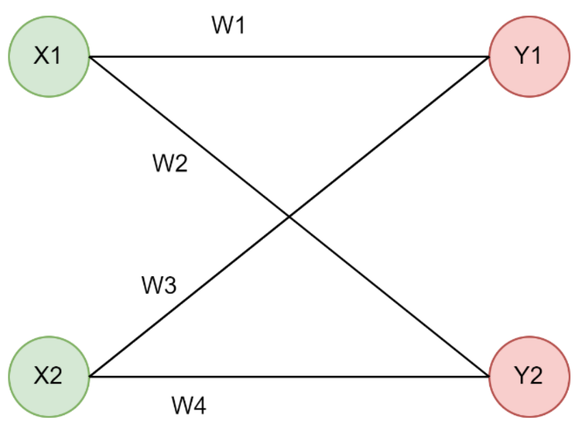
تمرین سوم

نام و نام خانوادگی	**میلاد محمدی**
شماره دانشجویی	**810100462**
تاریخ ارسال گزارش	**1401/09/11**

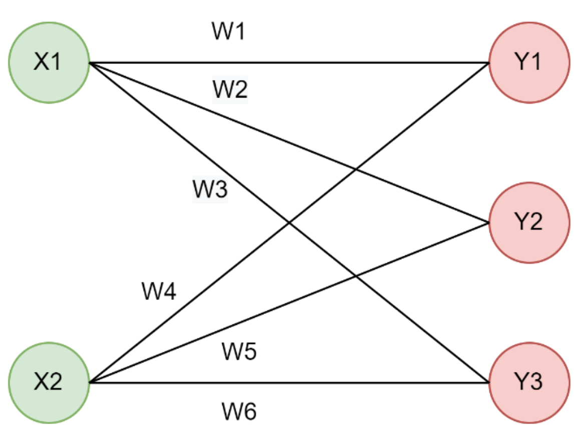
سوال اول

قسمت الف)

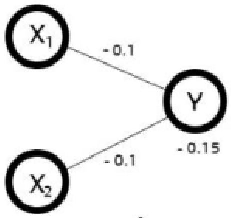
در شکل A دو کلاس داریم و لایه خروجی دارای دو نورون خواهد بود. با توجه به اینکه داده‌ها در دو بعد هستند، لایه ورودی نیز دارای دو نورون خواهد بود. نیازی به لایه پنهان نیست چون با استفاده از دو خط متقاطع کلاس‌ها قابل جداسازی می‌باشند. در صورتی لایه پنهان نیاز بود که بخواهیم با استفاده از یک خط خمیده کلاس‌ها را از هم جدا کنیم ولی ساده‌ترین مدل مدنظر ماست.



در شکل B سه کلاس داریم و لایه خروجی دارای سه نورون خواهد بود. با توجه به اینکه داده‌ها در دو بعد هستند، لایه ورودی نیز دارای دو نورون خواهد بود. نیازی به لایه پنهان نیست چون با استفاده از دو خط کلاس‌ها قابل جداسازی می‌باشند و ساده‌ترین مدل مدنظر ماست.



قسمت ب)



A

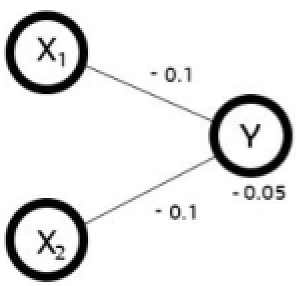
$(0\,,0)$: $(0 \,*\, -0.1) + (0 * -0.1) = 0 > \,-0.15 \;\rightarrow\; Y = 1$

$(0\,,1)$: $(0 \,*\, -0.1) + (1 * -0.1) = -0.1 > \,-0.15 \;\rightarrow\; Y = 1$

$(1\,,0)$: $(1 \,*\, -0.1) + (0 * -0.1) = -0.1 > \,-0.15 \;\rightarrow\; Y = 1$

$(1\,,1)$: $(1 \,*\, -0.1) + (1 * -0.1) = -0.2 < \,-0.15 \;\rightarrow\; Y = 0$

با توجه به خروجی‌ها، گیت NAND می‌باشد.



B

$(0\,,0)$: $(0 \,*\, -0.1) + (0 * -0.1) = 0 > \,-0.05 \;\rightarrow\; Y = 1$

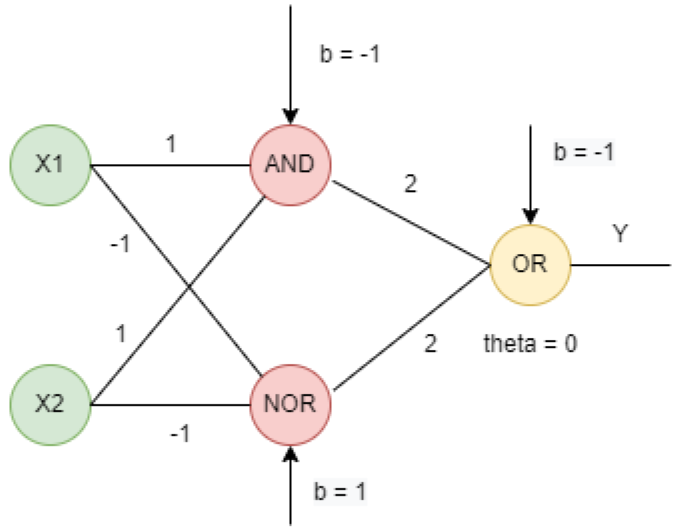
$(0\,,1)$: $(0 \,*\, -0.1) + (1 * -0.1) = -0.1 < \,-0.05 \;\rightarrow\; Y = 0$

$(1\,,0)$: $(1 \,*\, -0.1) + (0 * -0.1) = -0.1 < \,-0.05 \;\rightarrow\; Y = 0$

$(1\,,1)$: $(1 \,*\, -0.1) + (1 * -0.1) = -0.2 < \,-0.05 \;\rightarrow\; Y = 0$

با توجه به خروجی‌ها، گیت NOR می‌باشد.

قسمت ج) گیت XNOR با استفاده از ترکیب سه گیت AND، NOR و OR پیاده‌سازی می‌شود. در لایه‌های پنهان و لایه خروجی theta = 0 است.



$(0\,,0)$: $\quad AND$: $(0 * 1) + (0 * 1) - 1 = -1 < 0 \rightarrow 0$

NOR: $(0 * -1) + (0 * -1) + 1 = 1 > 0 \rightarrow 1$

OR: $(0 * 2) + (1 * 2) - 1 = 1 > 0 \rightarrow Y = 1$

$(0\,,1), (1,0)$: $\quad AND$: $(0 * 1) + (1 * 1) - 1 = 0 \leq 0 \rightarrow 0$

NOR: $(0 * -1) + (1 * -1) + 1 = 0 \leq 0 \rightarrow 0$

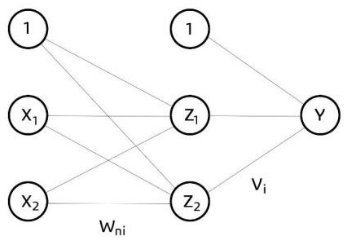
OR: $(0 * 2) + (0 * 2) - 1 = -1 < 0 \rightarrow Y = 0$

$(1\,,1)$: $\quad AND$: $(1 * 1) + (1 * 1) - 1 = 1 > 0 \rightarrow 1$

NOR: $(1 * -1) + (1 * -1) + 1 = -1 < 0 \rightarrow 0$

OR: $(1 * 2) + (0 * 2) - 1 = 1 > 0 \rightarrow Y = 1$

سوال دوم



$$W_{01} = 0.4, W_{02} = 0.6, W_{11} = 0.7,$$
$$W_{12} = -0.4, W_{21} = -0.2, W_{22} = 0.3$$
$$V_0 = -0.3, V_1 = 0.5, V_2 = 0.1$$

ابتدا خروجی را حساب می‌کنیم. حساب خطا را به دست آورده و پس انتشار انجام می‌دهیم تا وزن‌ها به‌روز شوند.

$$f(Z_1): W_{01} * 1 + W_{11} * X_1 + W_{21} * X_2 = (0.4 * 1) + (0.7 * 1) + 0 = f(1.1) = 0.75$$

$$f(Z_2): W_{02} * 1 + W_{12} * X_1 + W_{22} * X_2 = (0.6 * 1) + (-0.4 * 1) + 0 = f(0.2) = 0.53$$

$$f(Y): V_0 * 1 + V_1 * X_1 + V_2 * X_2 = (-0.3 * 1) + (0.75 * 0.5) + (0.1 * 0.53) = f(0.2) = 0.128$$

$$error = 0.5 * (0.128 - 0)^2 = 0.008$$

$$\sigma_Y = 1 * error = 1 * 0.008 = 0.008$$

$$\sigma_{Z_1} = f(z_1)\big(1 - f(z_1)\big) * V_1 * \sigma_Y = 0.75 * 0.25 * 0.5 * 0.008 = 0.00075$$

$$\sigma_{Z_2} = f(z_2)\big(1 - f(z_2)\big) * V_2 * \sigma_Y = 0.53 * 0.47 * 0.1 * 0.008 = 0.0001$$

$$W_{ji}(new) = \alpha \delta_j f(z_j), \qquad W_{ji}(new) = W_{ji}(old) + W_{ji}(new)$$

$$\Delta V_0 = 0.2 * 0.008 * 1 = 0.0016 \rightarrow \quad V_0(new) = -0.3 + 0.0016 = -0.2984$$

$$\Delta V_1 = 0.2 * 0.008 * 0.75 = 0.0012 \rightarrow \quad V_1(new) = 0.5 + 0.0012 = 0.5012$$

$$\Delta V_2 = 0.2 * 0.008 * 0.53 = 0.0008 \rightarrow \quad V_2(new) = 0.1 + 0.0008 = 0.1008$$

$$\Delta W_{11} = 0.2 * 0.00075 * 1 = 0.0001 \rightarrow \quad W_{11}(new) = 0.7 + 0.0001 = 0.7001$$

$$\Delta W_{21} = 0.2 * 0.00075 * 0 = 0 \rightarrow \quad W_{21}(new) = -0.2$$

$$\Delta W_{12} = 0.2 * 0.0001 * 1 = 0.00002 \rightarrow \quad W_{12}(new) = -0.4 + 0.00002 = -0.39$$

$$\Delta W_{22} = 0.2 * 0.0001 * 0 = 0 \rightarrow \quad W_{22}(new) = 0.3$$

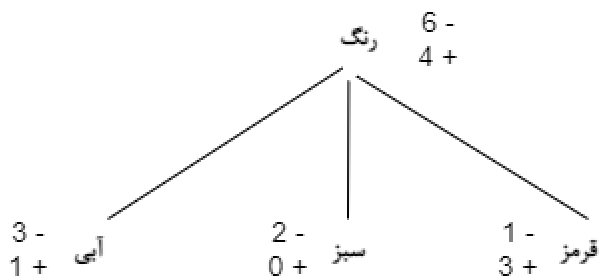
$$\Delta W_{01} = 0.2 * 0.00075 * 1 = 0.0001 \rightarrow \quad W_{01}(new) = 0.4 + 0.0001 = 0.4001$$

$$\Delta W_{02} = 0.2 * 0.0001 * 1 = 0.00002 \rightarrow \quad W_{02}(new) = 0.6 + 0.00002 = 0.60002$$

سوال سوم

$$H(y) = -\frac{4}{10} * \log\frac{4}{10} - \frac{6}{10} * \log\frac{6}{10} = 0.29$$

ویژگی رنگ:

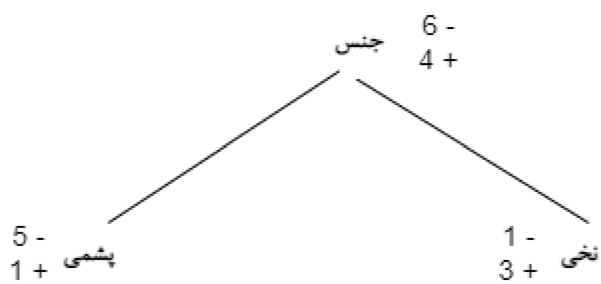


$P(y = +|\ \text{آبی}) = 1/4$ $\qquad P(y = +|\ \text{سبز}) = 0$ $\qquad P(y = +|\ \text{قرمز}) = 3/4$

$$H\left(y \middle| \text{رنگ}\right) = \frac{4}{10} H\left(\frac{1}{4}\right) + \frac{2}{10} H(0) + \frac{4}{10} H\left(\frac{3}{4}\right) = \frac{8}{10}\left(-\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}\right)$$

$$I(y; \text{رنگ}) = H(y) - H\left(y \middle| \text{رنگ}\right) = 0.29 - 0.19 = 0.1$$

ویژگی جنس:

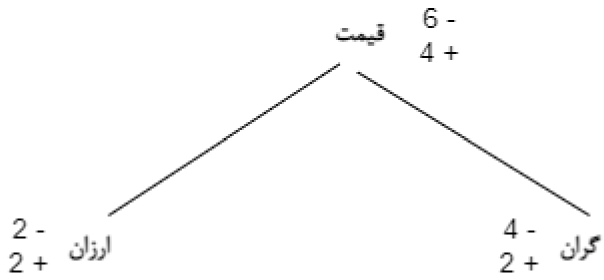


$P\left(y = + \middle| \text{پشمی}\right) = 1/6$ $\qquad P(y = +|\ \text{نخی}) = 3/4$

$$H\left(y \middle| \text{جنس}\right) = \frac{6}{10} H\left(\frac{1}{6}\right) + \frac{4}{10} H\left(\frac{3}{4}\right) = \frac{4}{10}\left(-\frac{1}{4}\log\frac{1}{4} - \frac{3}{4}\log\frac{3}{4}\right) + \frac{6}{10}\left(-\frac{1}{6}\log\frac{1}{6} - \frac{5}{6}\log\frac{5}{6}\right)$$

$$I(y; \text{جنس}) = H(y) - H\left(y \middle| \text{جنس}\right) = 0.29 - 0.20 = 0.09$$

ویژگی قیمت:

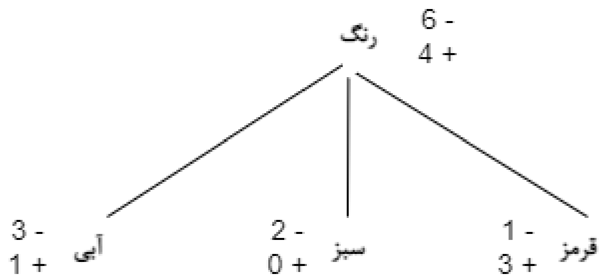


$P\left(y=+\middle|\text{ارزان}\right)=1/2$ $\qquad$ $P(y=+|\text{گران})=1/3$

$$H\left(y\middle|\text{قیمت}\right)=\frac{6}{10}H\left(\frac{1}{3}\right)+\frac{4}{10}H\left(\frac{1}{2}\right)=\frac{4}{10}\left(-\frac{1}{2}\log\frac{1}{2}-\frac{1}{2}\log\frac{1}{2}\right)+\frac{6}{10}\left(-\frac{1}{3}\log\frac{1}{3}-\frac{2}{3}\log\frac{2}{3}\right)$$

$$I\left(y;\text{قیمت}\right)=H(y)-H\left(y\middle|\text{قیمت}\right)=\ 0.28-0.28=0$$

با استفاده از درخت نیز بدیهی بود که ویژگی رنگ بهترین است.



با استفاده از درخت بالا پیش‌بینی‌ها به این صورت انجام می‌شود که برچسب رنگ آبی منفی(1 غلط)، رنگ سبز منفی و رنگ قرمز مثبت(1 غلط) خواهد بود. جدول زیر با توجه به ویژگی رنگ ساخته شده ولی صحت و دقت با استفاده از جنس نیز همین مقادیر خواهد بود.

Real \ Pred	+	–
+	3	1
–	1	5

$$Recall=\frac{TP}{P}=\frac{3}{4}\ ,Precision=\frac{TP}{TP+FP}=\frac{3}{4}\ ,Accuracy=\frac{TP+TN}{n}=\frac{3+5}{10}=0.8$$

سوال چهارم

قسمت الف)

مورد اول: زمانی که x کوچکتر از 0 است : $x - x_i < 0$ و در نتیجه $P_n(x)$ برابر صفر خواهد بود. پس میانگین نیز برابر صفر خواهد بود.

مورد دوم: زمانی که $0 \leq x \leq a$:

$$P_n^-(x) = E[P_n(x)] = \frac{1}{n}\sum_{i=1}^{n} E[\frac{1}{h_n} * e^{-\frac{x-x_i}{h_n}}] = \int_0^x \frac{1}{ah_n}\, e^{-\frac{x-t}{h_n}}\, dt$$

$$= \frac{1}{a}\, e^{-\frac{x}{h_n}} * \left(e^{\frac{x}{h_n}} - 1\right) = \frac{1}{a}\, (1 - \; e^{\left(-\frac{x}{h_n}\right)})$$

مورد سوم: با توجه به اینکه $p(x) \sim U(0, a)$، تمامی x_i ها در بازه 0 تا a قرار خواهند داشت.

$$V_n = \int e^{\frac{x-x_i}{h_n}} = h_n$$

$$x \geq a \rightarrow \;\; x - x_i \geq 0$$

$$P_n(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{v_n} * \varphi(\frac{x-x_i}{h_n}) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h_n} * e^{-\frac{x-x_i}{h_n}}$$

$$P_n^-(x) = E[P_n(x)] = \frac{1}{n}\sum_{i=1}^{n} E[\frac{1}{h_n} * e^{-\frac{x-x_i}{h_n}}] = \int_0^a \frac{1}{ah_n}\, e^{-\frac{x-t}{h_n}}\, dt$$

$$= \frac{1}{a}\, e^{-\frac{x}{h_n}} * \left(e^{\frac{a}{h_n}} - 1\right)$$

قسمت ب) با توجه به $p(x) \sim U(0, a)$:

$$p(x) = \begin{cases} \frac{1}{a}, & 0 \leq x \leq a \\ 0, & otherwise \end{cases}$$

بایاس از رابطه زیر به دست می‌آید:

$$p(x) - P_n^-(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{a} * e^{\left(-\frac{x}{h_n}\right)} & 0 \leq x \leq a \\ -\frac{1}{a} e^{-\frac{x}{h_n}} * \left(e^{\frac{a}{h_n}} - 1\right) & x > a \end{cases}$$

برای اینکه اندازه‌ی بایاس تخمین پارزن، در نود و نه درصد از طول بازه‌ی 0 تا a کمتر یک درصد باشد:

$$\frac{p(x) - P_n^-(x)}{p(x)} \leq 0.01 \rightarrow \frac{\frac{1}{a} * e^{\left(-\frac{x}{h_n}\right)}}{\frac{1}{a}} \leq 0.01 \rightarrow \mathrm{h_n} \leq \frac{(0.01 * \mathrm{a})}{\ln 100}$$

سوال پنجم

قسمت الف)

احتمال پیشین دو کلاس برابر است. احتمال متوسط خطا برای n نقطه به صورت زیر است:

$$p_n(e) = p(x \,\epsilon\, \omega_1, y \rightarrow \ \omega_2) + p(x \,\epsilon\, \omega_2, y \rightarrow \ \omega_1) = 2p(x \,\epsilon\, \omega_1, y \rightarrow \ \omega_2)$$

$$= \ 2p(\omega_1) * p_r(label\ of\ \omega_1\ for\ fewer\ than \frac{k-1}{2}\ points\ and\ rest \rightarrow \ \omega_2)$$

$$= 2 * \frac{1}{2} \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j} \frac{1}{2^j} \frac{1}{2^{n-j}} = \frac{1}{2^n} \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j}$$

قسمت ب) مقدار خطا برای حالت $k = 1$ برابر $\frac{1}{2^n}$ است در حالی که خطا برای حالت $k > 1$ ، $\frac{1}{2^n} \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j}$ می‌باشد که بدیهی است برای حالت $k = 1$ کمتر است.

قسمت ج)

$$p_n(e) = \frac{1}{2^n} \sum_{j=0}^{\frac{k-1}{2}} \binom{n}{j} = p\left(B\left(n, \frac{1}{2}\right) \leq \frac{k-1}{2}\right) = p(Y_1 + Y_2 + \cdots Y_n \leq \ \frac{k-1}{2})$$

تمامی Y_i ها مستقل از هم هستند و B توزیع $Binomial$ می‌باشد و در این سوال $p(Y_1) = p(Y_2) = \frac{1}{2}$

$$p_n(e) = p\left(Y_1 + Y_2 + \cdots Y_n \leq \frac{\frac{a}{\sqrt{n}} - 1}{2}\right) = p\ (Y_1 + Y_2 + \cdots Y_n \leq 0)$$

بنابراین با افزایش n به سمت بی‌نهایت، میزان خطا به صفر نزدیک خواهد شد.

1 Question 6

[1]:
```
from keras.datasets import cifar10
from sklearn.model_selection import train_test_split
import random
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.metrics import plot_confusion_matrix
from keras.layers import BatchNormalization
import pandas as pd
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support
import keras.utils.vis_utils
from importlib import reload
import pydot
reload(keras.utils.vis_utils)
keras.utils.vis_utils.pydot = pydot
from tensorflow.keras.utils import to_categorical
from tabulate import tabulate
```

[2]:
```
(X_train_full, y_train_full), (X_test, y_test) = cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 14s 0us/step
```

[3]:
```
X_train_normal  = X_train_full/255
X_test_normal  = X_test/255
```

[4]:
```
X_train, X_val, y_train, y_val = train_test_split(X_train_normal, y_train_full,
 ↪test_size=0.15, random_state = 20)
```

[5]:
```
y_train_onehot = to_categorical(y_train, 10)
y_test_onehot = to_categorical(y_test, 10)
y_val_onehot = to_categorical(y_val, 10)
```

[6]:
```
np.random.seed(0)
tf.random.set_seed(0)
```

[7]:
```
def plot(x):

    loss = model_history.history.copy()
    loss.pop('accuracy')
    loss.pop('val_accuracy')
    acc = model_history.history.copy()
    acc.pop('loss')
    acc.pop('val_loss')

    pd.DataFrame(loss).plot(figsize=(8,5))
    plt.grid(True)
    plt.show()

    pd.DataFrame(acc).plot(figsize=(8,5))
    plt.grid(True)
    plt.ylim(0,1)
    plt.show()
```

[8]:

```
model = keras.models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(layers.Conv2D(32, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.2))
model.add(layers.Conv2D(64, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(layers.Conv2D(64, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.3))
model.add(layers.Conv2D(128, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(layers.Conv2D(128, (3, 3), activation='relu',␣
↪kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.4))
model.add(layers.Flatten())
model.add(layers.Dense(128, activation='relu', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(10, activation='softmax'))
opt = tf.keras.optimizers.SGD(lr=0.001, momentum=0.9)
```

```
/usr/local/lib/python3.8/dist-
packages/keras/optimizers/optimizer_v2/gradient_descent.py:108: UserWarning: The
`lr` argument is deprecated, use `learning_rate` instead.
  super(SGD, self).__init__(name, **kwargs)
```

2 Part A

i. Train data: 70%, Test data: 15%, Validation data: 15%

ii. Table 1. Architecture of model

[17]:
```
data = [["conv2d", 32,(3,3),"relu"],["batch_normalization","","",""], ["conv2d_1",␣
↪32,(3,3),"relu"], ["batch_normalization_1","","",""],␣
↪["max_pooling2d","",(2,2),""],
        ["dropout",0.2,"",""],␣
↪["conv2d_2",64,(3,3),"relu"],["batch_normalization_2","","",""],␣
↪["conv2d_3",64,(3,3),"relu"],["batch_normalization_3","","",""],␣
↪["max_pooling2d_1","",(2,2),""],
        ["dropout_1 ",0.3,"",""],␣
↪["conv2d_4",128,(3,3),"relu"],["batch_normalization_4","","",""],["conv2d_5",128,(3,3),"relu"],["
↪","","",""],["max_pooling2d_2 ","",(2,2),""],
        ["dropout_2",0.4,"",""],["flatten ","","",""], ["dense␣
↪",128,"","relu"],["batch_normalization_6 ","","",""],["dropout_3 ",0.
↪5,"",""],["dense_1 ",10,"","softmax"]]

col_names = ["layers", "number of neurons", "kernel","activation function"]
print(tabulate(data, headers=col_names))
```

```
layers                 number of neurons    kernel    activation function
---------------------  -------------------  --------  ---------------------
conv2d                 32                   (3, 3)    relu
batch_normalization
conv2d_1               32                   (3, 3)    relu
batch_normalization_1
max_pooling2d                               (2, 2)
dropout                0.2
conv2d_2               64                   (3, 3)    relu
batch_normalization_2
conv2d_3               64                   (3, 3)    relu
batch_normalization_3
max_pooling2d_1                             (2, 2)
dropout_1              0.3
conv2d_4               128                  (3, 3)    relu
batch_normalization_4
conv2d_5               128                  (3, 3)    relu
batch_normalization_5
max_pooling2d_2                             (2, 2)
dropout_2              0.4
flatten
dense                  128                            relu
batch_normalization_6
dropout_3              0.5
dense_1                10                             softmax
```

a: Output layer:

Activation function: Softmax. Because SoftMax outputs its probability for each class, and it is the most appropriate function of the activation function.

Number of neurons: 10. Because there are 10 classes.

Cost function: categorical_crossentropy. categorical_crossentropy is a cost function used for classification and other functions do not perform well.

Learning rate:0.001.

Optimizer: SGD.

3 Part B

[11]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_accuracy', patience=3,␣
↪restore_best_weights=True)
```

[12]:
```
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```

[13]:
```
model_history = model.fit(X_train, y_train_onehot, epochs=100, batch_size = 32,␣
↪validation_data=(X_val, y_val_onehot), callbacks = [callback])
```

4 Part C

[14]:
```
plot(model_history.history)
```

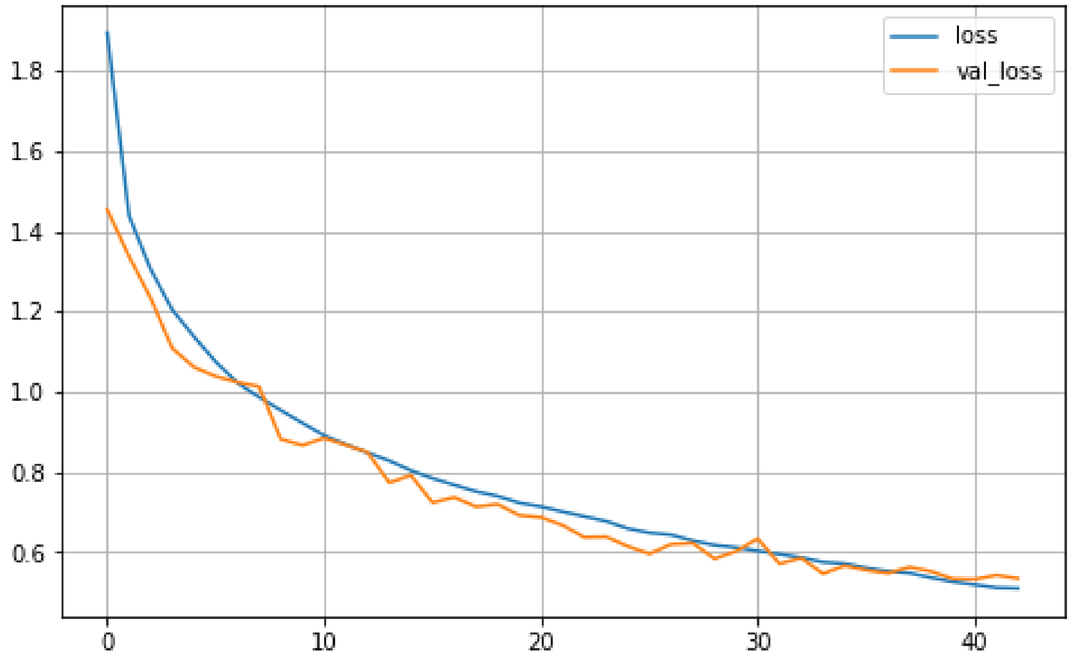


Figure 1: Loss plot for training and validation data in each epoch

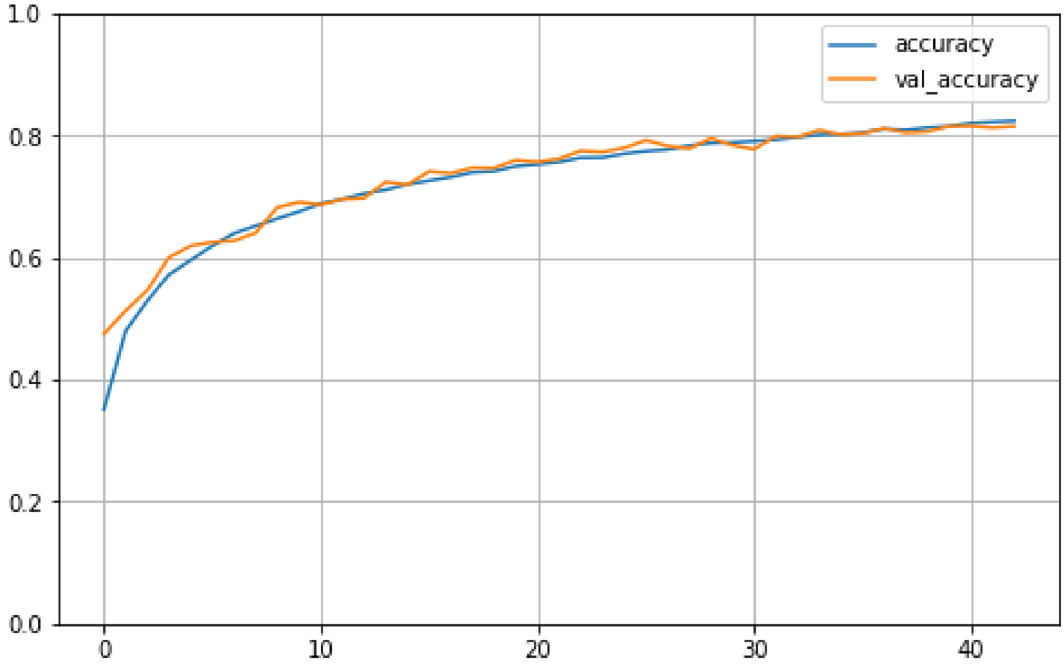


Figure 2: Accuracy plot for training and validation data in each epoch

[15]:

```
import seaborn as sns
predict_x=model.predict(X_test_normal)
predictions=np.argmax(predict_x,axis=1)

plt.figure(figsize=(10,8))
conf_mat = confusion_matrix(predictions.tolist(), y_test.tolist())
sns.heatmap(conf_mat, annot=True, cmap='crest')
plt.title('Confusion Matrix')
```

```
313/313 [==============================] - 1s 3ms/step
```

[15]:

```
Text(0.5, 1.0, 'Confusion Matrix')
```

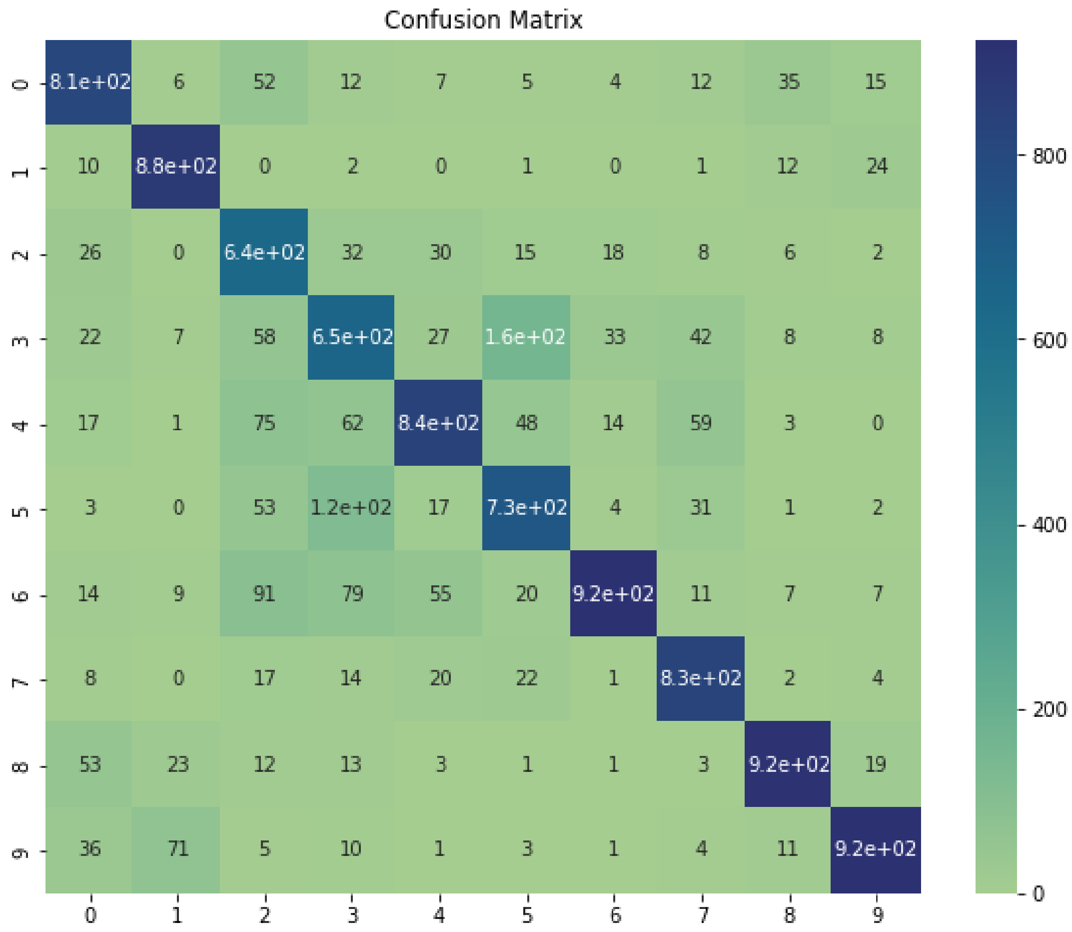


Figure 3: Confusion matrix

5 Question 8

[7]:
```
import numpy as np
import pandas as pd
import scipy.stats as st
import matplotlib.pyplot as plt
import seaborn as sns
```

[12]:
```
N = 1000
np.random.seed(1)
X = np.concatenate((np.random.normal(0, 1, int(0.3 * N)), np.random.normal(5, 1,␣
↪int(0.7 * N))))[:, np.newaxis]
```

i. Real distribution

[13]:
```
sns.distplot(X, bins = 100)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619:
FutureWarning: `distplot` is a deprecated function and will be removed in a
future version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `histplot` (an axes-level function for
histograms).
  warnings.warn(msg, FutureWarning)
```

[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f6d4ad8d640>

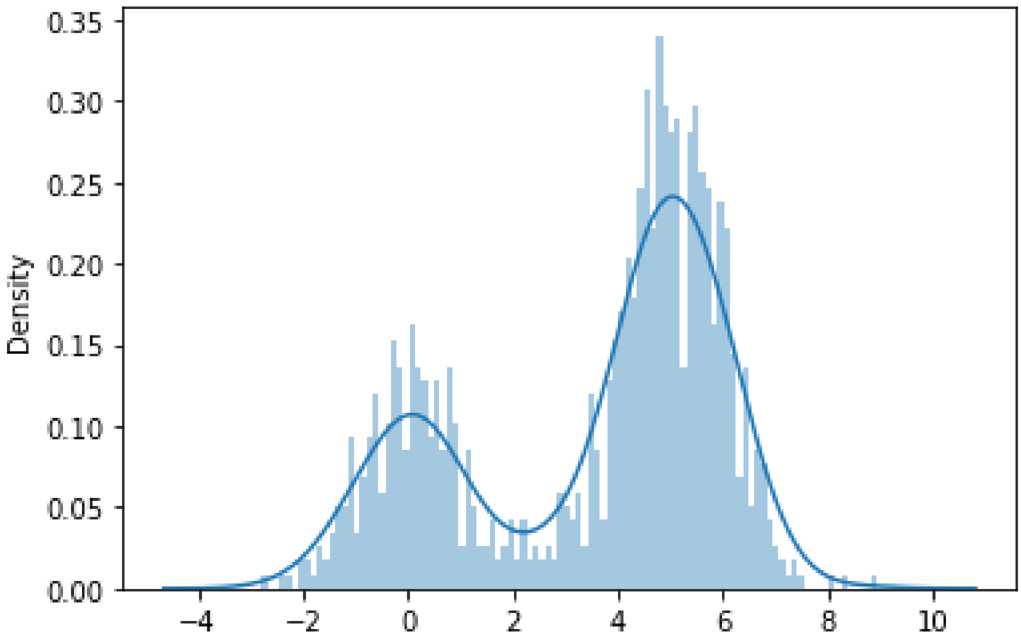


Figure 4: Real distribution

[4]:
```
class Parzen:
    data = []
    h = 1
    def __init__(self, data: list = []):
        self.data = data

    def set_window_size(self, h):
        self.h = h

    def get_dist(self, x):
        after_kernel = lambda i: st.norm.pdf((x-i)/self.h)
        kernelized = map(after_kernel,self.data)
        return sum(kernelized)/len(self.data)
```

Window = 10

[ ]:
```
parzen_model = Parzen(X)
x = np.linspace(-1,max(X),1000)
y = parzen_model.get_dist(x)
parzen_model.set_window_size(10)
plt.plot(x,parzen_model.get_dist(x))
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f5aacfb64f0>]
```

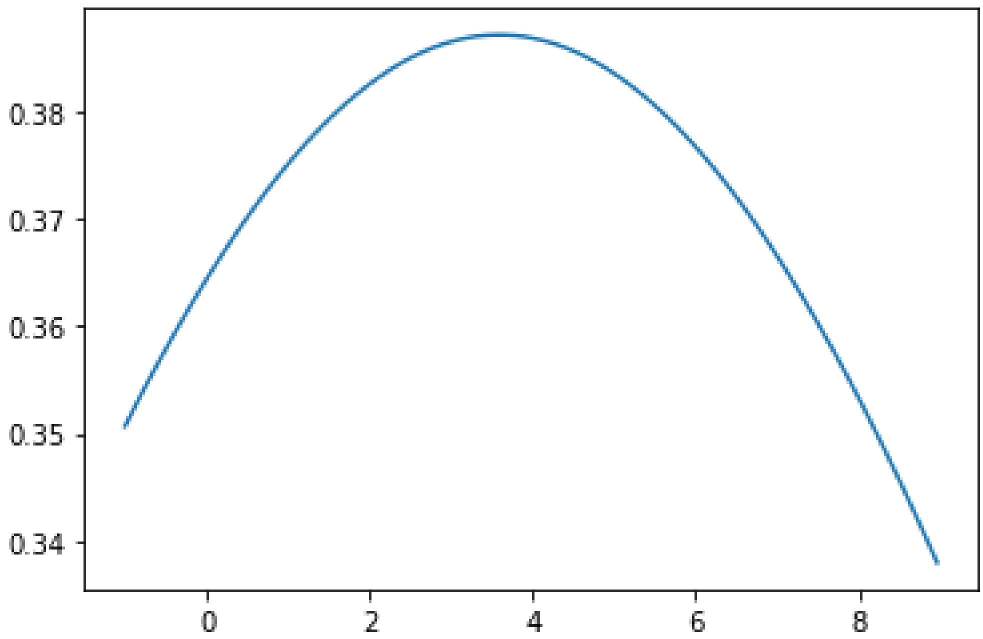


Figure 5: Data distribution plot with window = 10

Window = 1

[ ]:
```
parzen_model.set_window_size(1)
plt.plot(x,parzen_model.get_dist(x))
```

[ ]:
```
[<matplotlib.lines.Line2D at 0x7f5aacf08310>]
```

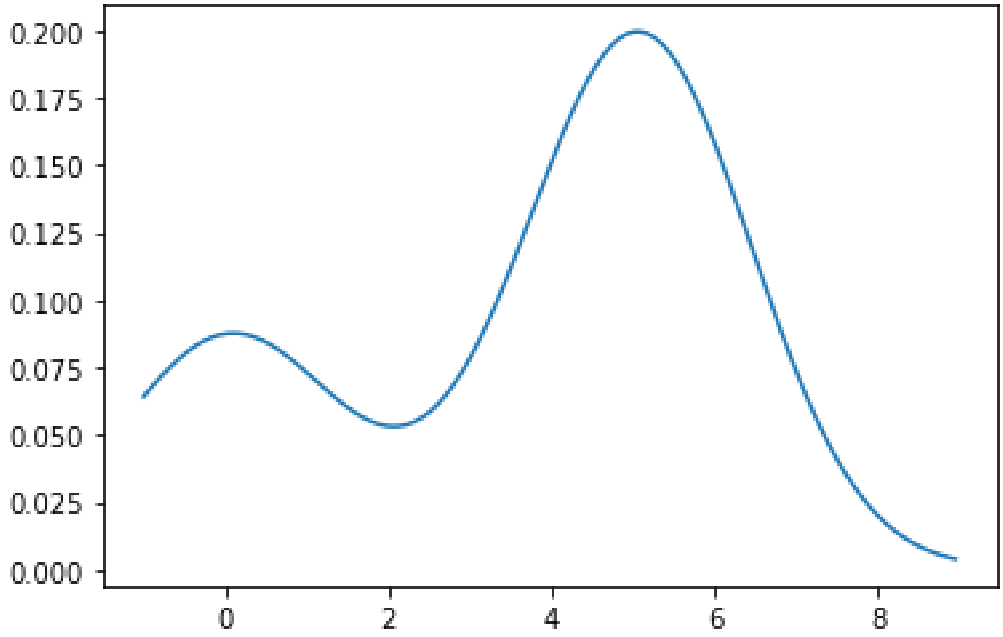


Figure 6: Data distribution plot with window = 1

Window = 0.1

```
[ ]: parzen_model.set_window_size(0.1)
     plt.plot(x,parzen_model.get_dist(x))
```

```
[ ]: [<matplotlib.lines.Line2D at 0x7f5aacedbcd0>]
```

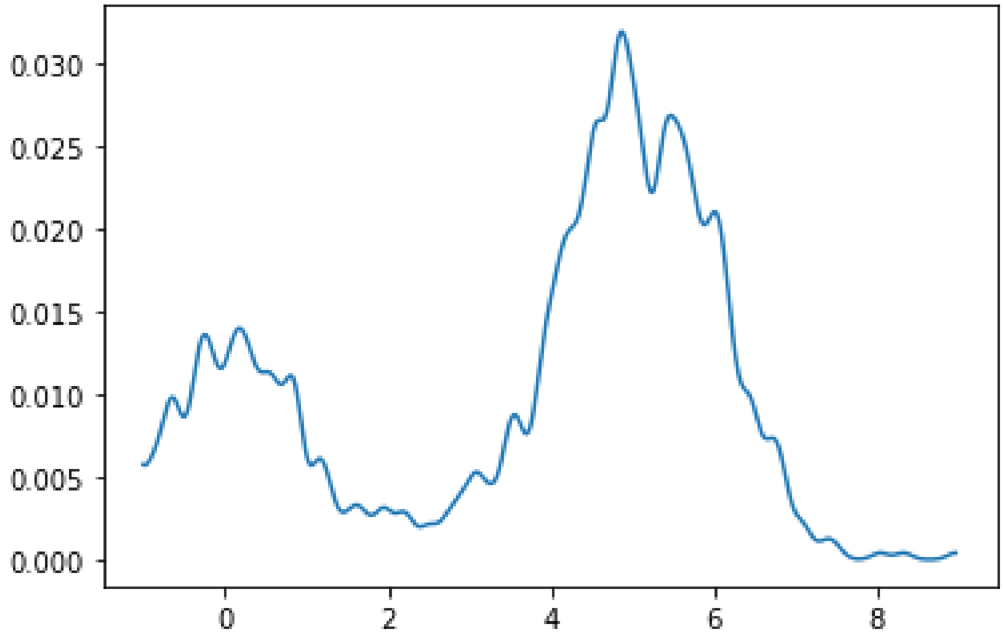


Figure 7: Data distribution plot with window = 0.1

6 Conclusion

When the window size is equal to 10, the model suffers from underfitting, and if it is equal to 0.1, it also pays attention to the data noise and suffers from overfitting. It performs the best fit when the window size is equal to one.