به نام خدا

دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

درس سیستم‌های هوشمند

تمرین شماره 3

نام و نام خانوادگی : محمدرضا بختیاری

شماره دانشجویی : 810197468

آذر 1400

فهرست سوالات

سوال 1 : کاربرد شبکه های عصبی پیچشی در طبقه بندی

ابتدا با استفاده از قطعه کد زیر داده های Cifar10 را لود می کنیم :

```
from keras.datasets import cifar10
(X_train , y_train), (X_test , y_test) = cifar10.load_data()
```

در ادامه نیاز داریم تا داده های ورودی را نرمال کنیم و سپس از هر طبقه یک عکس را نشان می دهیم و با طبقه های مختلف این دادگان آشنا می شویم . با استفاده از قطعه کد زیر خواهیم داشت :

```
# Checkout the Data
print('Training data shape :', X_train.shape, y_train.shape)
print('Testing data shape :', X_test.shape, y_test.shape)

# Find the unique numbers from the train labels
import numpy as np
classes = np.unique(y_train)
nClasses = len(classes)

import matplotlib.pyplot as plt

plt.figure(figsize=[15,10])

# Display 10 images in training data
plt.subplot(191)
plt.imshow(X_train[0,:,:], cmap='gray')

plt.subplot(192)
plt.imshow(X_train[1,:,:], cmap='gray')

plt.subplot(193)
plt.imshow(X_train[9,:,:], cmap='gray')

plt.subplot(194)
plt.imshow(X_train[3,:,:], cmap='gray')

plt.subplot(195)
plt.imshow(X_train[13,:,:], cmap='gray')

plt.subplot(196)
plt.imshow(X_train[5,:,:], cmap='gray')

plt.subplot(197)
plt.imshow(X_train[220,:,:], cmap='gray')

plt.subplot(198)
```

```
plt.imshow(X_train[7,:,:], cmap='gray')

plt.subplot(199)
plt.imshow(X_train[8,:,:], cmap='gray')


# Flatten the data
# Change from matrix to array of dimension 32*32 to array of dimension 1024
X_train = X_train.reshape((len(X_train), np.prod(X_train.shape[1:]))) #(60000, 1024)
X_test = X_test.reshape((len(X_test), np.prod(X_test.shape[1:]))) #(10000, 1024)

# Change to float datatype
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# Normalization from [0;255] to [0;1], Scale the data to lie between 0 to 1
X_train /= 255
X_test /= 255

# convert labels to one-hot  vectors
from keras.utils import np_utils
Y_train = np_utils.to_categorical(y_train)
Y_test = np_utils.to_categorical(y_test)
```

حال نمونه تصاویر برای هر طبقه را مشاهده می کنیم :

شکل 1-1 : نمونه ای تصادفی از تصاویر یک طبقه

حال نوبت به آن می رسد که مدل را پیاده سازی کنیم , با استفاده از قطعه کد زیر خواهیم داشت :

```
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers.core import Dense

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(None,32,3072)))  #Hidden Layer 1
model.add(Dense(512, activation='relu'))  #Hidden Layer 2
model.add(Dense(10, activation='softmax'))  #Last layer with one output per class
model.summary()

# Configure the Network
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

که ساختار مدل نیز به این شکل خواهد بود :

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, None, 32, 512)     1573376

 dense_1 (Dense)             (None, None, 32, 512)     262656

 dense_2 (Dense)             (None, None, 32, 10)      5130

=================================================================
Total params: 1,841,162
Trainable params: 1,841,162
Non-trainable params: 0
```

شکل 2-1 : ساختار کلی مدل طراحی شده

اکنون نوبت به آموزش مدل و بررسی دقت و خطا بر روی دادگان ارزیابی و آموزش می باشد . در اینجا 20 درصد دادگان آموزش را به دادگان ارزیابی اختصاص می دهیم . در انتها از روی نتایج به دست آمده از دادگان ارزیابی , فرا پارامترهای[1] مورد نیاز (در اینجا تعداد دوره[2] مناسب) را به دست می آوریم .

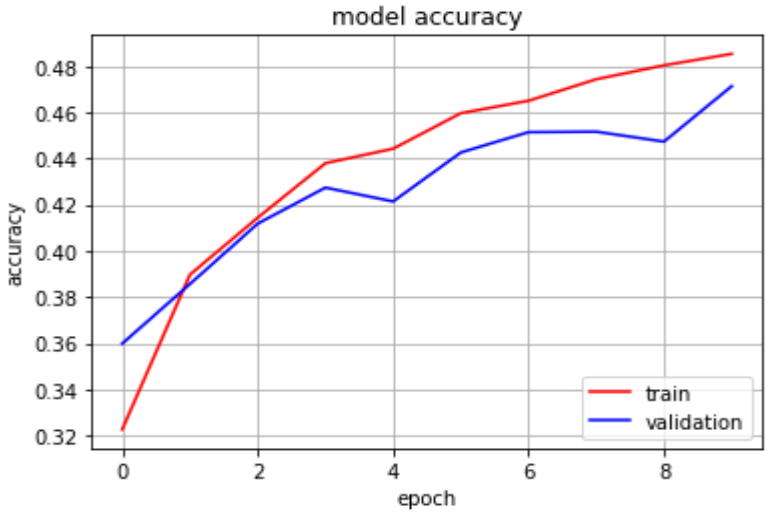
با استفاده از قطعه کد زیر خواهیم داشت :

```
history = model.fit(X_train, Y_train, epochs=10, batch_size=32, validation_split=0.2)

# Plotting Metrics
# Plot the Accuracy Curves
fig = plt.figure()
plt.plot(history.history['accuracy'],'r')
plt.plot(history.history['val_accuracy'],'b')
```

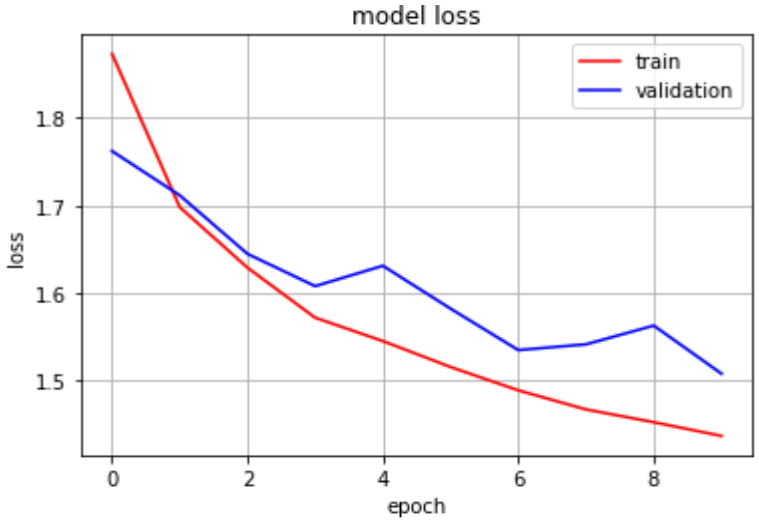
[1] Hyperparameters
[2] Epoch

```
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='lower right')
plt.grid()

# Plot the Loss Curves
fig = plt.figure()
plt.plot(history.history['loss'],'r')
plt.plot(history.history['val_loss'],'b')
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
plt.grid()
```

بر این اساس نمودار دقت و خطا برحسب تعداد دوره به این ترتیب خواهد بود :



شکل 3-1 : نمودار دقت بر حسب تعداد دوره برای دادگان آموزش و ارزیابی



شکل 4-1 : نمودار خطا بر حسب تعداد دوره برای دادگان آموزش و ارزیابی

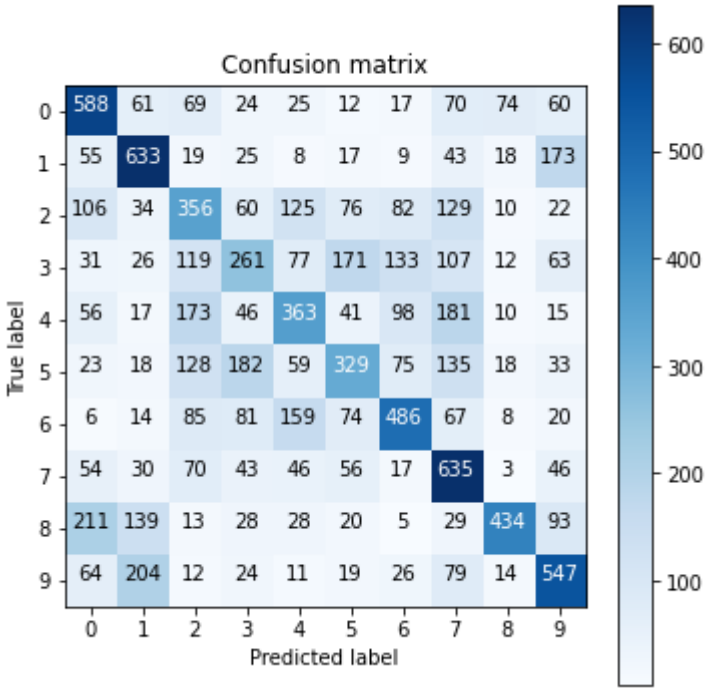
در انتها نوبت به ارزیابی مدل توسط داده های آزمون می رسد . با استفاده از قطعه کد زیر دقت و ماتریس آشفتگی[1] را گزارش می کنیم :

```
# Prediction Labels
Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)

# Evaluate the trained model
[test_loss, test_acc] = model.evaluate(X_test, Y_test)
print("Test Loss", test_loss)
print("Test Accuracy", test_acc)

correct_indices = np.nonzero(y_pred == y_test)[0]
incorrect_indices = np.nonzero(y_pred != y_test)[0]

print(" classified correctly", len(correct_indices))
print(" classified incorrectly", len(incorrect_indices))


from sklearn.metrics import confusion_matrix , classification_report
# Confusion Matrix
class_names=[0,1,2,3,4,5,6,7,8,9]
confusion_mtx = confusion_matrix(y_test, y_pred)
print("confusion matrix=\n",confusion_mtx)

import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
 """
 This function prints and plots the confusion matrix.
 Normalization can be applied by setting "normalize=True" .
 """
 plt.figure(figsize = (5,5))
 plt.imshow(cm, interpolation='nearest', cmap=cmap)
 plt.title(title)
 plt.colorbar()
 tick_marks = np.arange(len(classes))
 plt.xticks(tick_marks, classes)
 plt.yticks(tick_marks, classes)
 if normalize:
   cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

 thresh = cm.max() / 2.
 for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
```

[1] Confusion matrix

```
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j] > thresh else "black")
  plt.tight_layout()
  plt.ylabel('True label')
  plt.xlabel('Predicted label')

plot_confusion_matrix(confusion_mtx, class_names)

print(classification_report(y_test,y_pred))
```

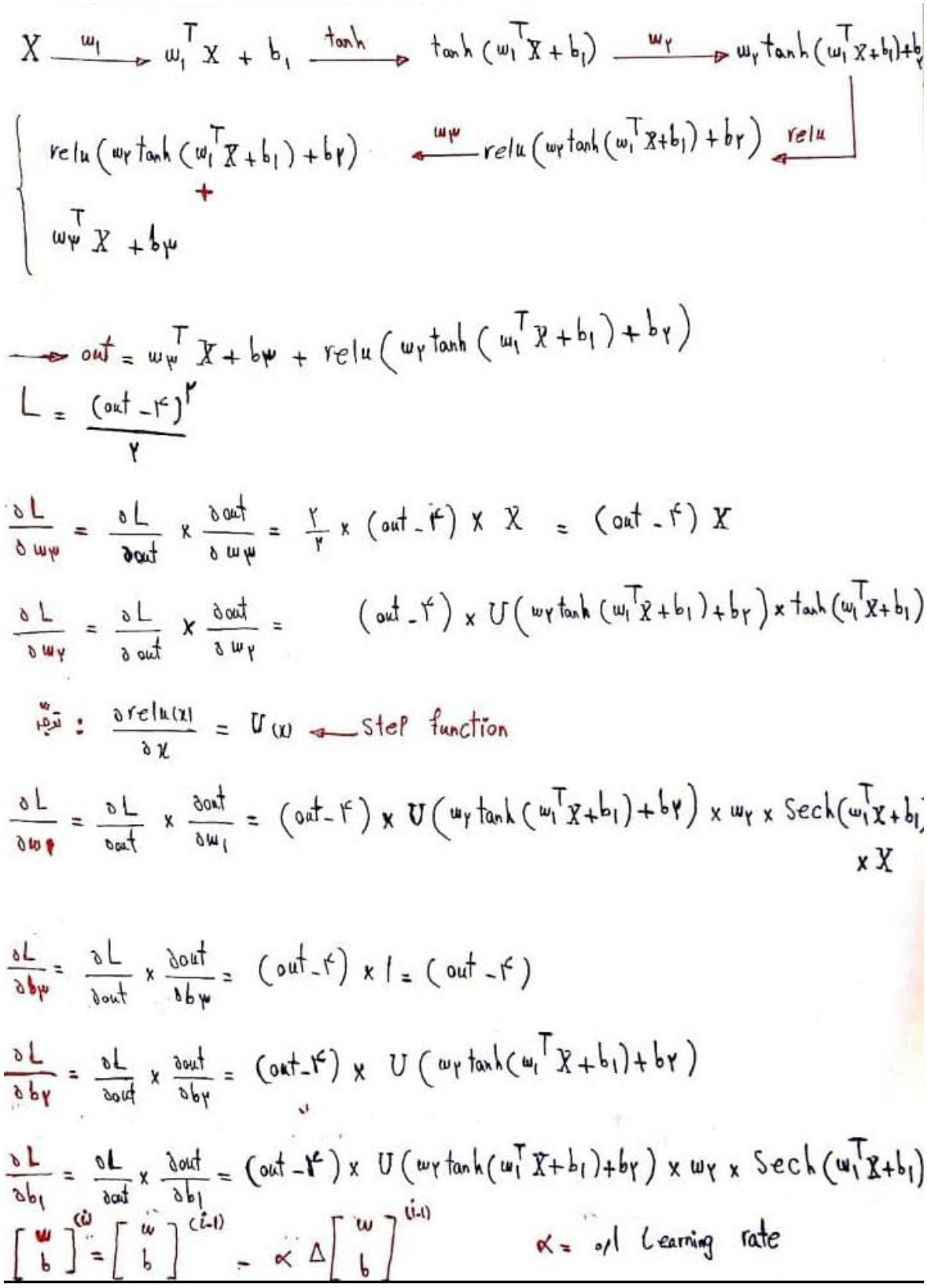
و ماتریس آشفتگی به این صورت خواهد بود :



شکل 5-1 : ماتریس آشفتگی داده های آزمون

سوال 2 : شبکه عصبی (پرسپترون با چند لایه مخفی)

الف: تحلیلی



$$\hat{out}^{(0)} = w_3^T X_0 + b_3 + relu(w_2 \tanh(w_1^T X_0 + b_1) + b_2)$$

$$X_0 = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \quad w_1 = \begin{bmatrix} 1.84 & -0.18 & 0.4 \\ 0.24 & -1.8 & 0.43 \end{bmatrix} \quad b_1 = \begin{bmatrix} 0 \\ 0.21 \\ 1.8 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1.850 & -0.504 & 0.48 \end{bmatrix} \quad b_2 = 0.1$$

$$w_3 = \begin{bmatrix} -0.8 \\ 1.4 \end{bmatrix} \quad b_3 = 0.1$$

اولین تکرار:

$$\hat{out}^{(0)} = 6.4894 \longrightarrow w_1 = \begin{bmatrix} 1.84784 & -0.20232 & 0.55484 \\ 0.24502 & 0.59602 & -1.84348 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1.40112 & -0.31108 & 0.43109 \end{bmatrix} \quad w_3 = \begin{bmatrix} -1.29788 \\ 0.58318 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} -0.01114 \\ 0.59884 \\ 1.78884 \end{bmatrix} \quad b_2 = -0.14894 \quad b_3 = -0.14894$$

$$\hat{out}^{(1)} = 2.0084$$

دومین تکرار:

$$w_1 = \begin{bmatrix} 1.85864 & -0.18394 & 0.59404 \\ 0.24504 & -1.80594 & 0.44604 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 1.80019 & -0.50121 & 0.4302 \end{bmatrix} \quad w_3 = \begin{bmatrix} -0.1994 \\ 1.4506 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} 0.00198 \\ 0.40802 \\ 1.79802 \end{bmatrix} \quad b_2 = 0.502 \quad b_3 = 0.502$$

$$\hat{out}^{(2)} = 5.594$$

ب: تحقیق

تابع هزینه رگرسیون

تابع هزینه نرم اول بیانگر میانگین , قدر مطلق اختلاف نتیجه به دست آمده با نتیجه واقعی است و به صورت زیر آن را نمایش می دهیم :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

مشکل این تابع هزینه این است که در ادامه بزرگی گرادیان به اندازه خطا وابسته نیست و فقط به علامت $y - \hat{y}$ بستگی دارد و می تواند منجر به این شود که بزرگی گرادیان حتی زمانی که خطا کوچک باشد , بزرگ خواهد بود که می تواند منجر به مشکلات همگرایی شود .

تابع هزینه نرم دوم نمایانگر میانگین مجذور اختلاف نتیجه به دست آمده با نتیجه واقعی است و به صورت زیر آن را نمایش می دهیم :

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

ضعف این روش هنگامی مشخص می شود که در بین دادگان , داده ای فاصله ی زیادی با مقدار واقعی خود داشته باشد و به اصطلاح داده ی پرت باشد که منجر به افزایش قابل ملاحظه خطا می شود .

تابع هزینه هوبر که ترکیبی از دو تابع هزینه ذکر شده می باشد , به ازای اختلاف های کوچک به صورت غیر خطی (همانند نرم دوم) عمل کرده و به ازای اختلاف های بزرگ به صورت خطی (همانند نرم اول) عمل کرده و به صورت زیر قابل نمایش است :

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}(y_i - \hat{y}_i)^2 \qquad |y_i - \hat{y}_i| \le \delta$$

$$Huber = \frac{1}{n}\sum_{i=1}^{n}\delta\left(|y_i - \hat{y}_i| - \frac{1}{2}\delta\right) \qquad |y_i - \hat{y}_i| > \delta$$

تابع هزینه هوبر در واقع نسخه اصلاح شده نرم دوم است هنگامی که خطا زیاد است و نسخه اصلاح شده نرم اول است هنگامی که خطا کوچک است .

استفاده از داده ارزیابی

هنگامی که خطا برای داده های آموزش و ارزیابی بسیار به هم نزدیک باشند دو حالت وجود دارد , اول این که مقدار خطا کم بوده که در این صورت به احتمال زیاد مدل درسته آموزش داده شده است و مشکلی نداریم .

حالت دوم زمانی رخ می دهد که خطا در ابتدا زیاد بوده (هم برای داده های آموزش و هم داده های ارزیابی) و با افزایش تعداد تکرار نیز خطا کاهش پیدا نمی کند که در این صورت با پدیده ی Underfitting مواجه هستیم . علت نیز این است که مدل به درستی آموزش داده نشده است و بایاس مدل زیاد می باشد. همانگونه که در تصویر زیر مشاهده می کنیم تابع هزینه (یا در اینجا خطا) از ابتدا برای دادگان آموزش و ارزیابی بالا بوده و با افزایش تعداد تکرار نیز به صورت قابل ملاحظه ای کاهش پیدا نمی کند .

شکل 1-2 : تابع هزینه بر حسب تعداد تکرار برای حالتی که بایاس مدل بالا است .

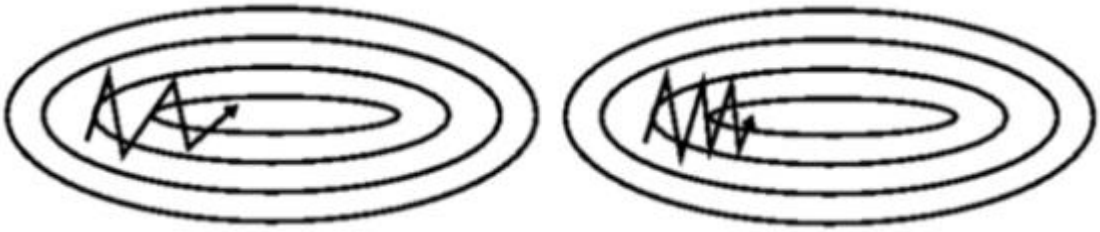
گرادیان نزولی به همراه تکانه

مشکل اصلی گرادیان نزولی[1] این است که وقتی می خواهد به نقطه ی کمینه نزدیک شود به علت نوسانات زیاد نمی توان نرخ یادگیری[2] را افزایش داد در نتیجه سرعت همگرایی کم بوده .

این مشکل در گرادیان نزولی به همراه تکانه[3] رفع می شود . به این صورت که در تابع گرادیان یک تکانه اضافه می کنیم به این معنی که مقدار گرادیان در نقطه فعلی به مقدار گرادیان در نقاط قبل وابسته است. که منجر به این می شود که در گرادیان نزولی به همراه تکانه سرعت همگرایی افزایش یافته و همچنین نوسانات نیز کاهش می یابد .

---

[1] SGD
[2] Learning rate
[3] SGD with momentum



شکل 2-2 : به ترتیب از راست به چپ هنگامی که از تکانه استفاده نمی کنیم و هنگامی که از تکانه استفاده می کنیم .

پ: پیاده سازی شبکه پرسپترون در کاربرد رگرسیون

تولید دادگان

ابتدا با استفاده از قطعه کد زیر دو مجموعه 10000 تایی از X , Y ها در بازه ی گفته شده تولید می کنیم و برچسب[1] متناظر با هر مجموعه را محاسبه می کنیم .

در انتها 20 درصد از کل داده ها را به داده های آزمون[2] , 64 درصد از کل داده ها را به داده های آموزش[3] و 16 درصد از کل داده ها را به داده های ارزیابی[4] اختصاص می دهیم .

```
import numpy as np
import math
from Function import Random_Sampling
X = np.random.uniform(0,2*math.pi,10000)                    # 0 < X < 2*PI
Y = np.random.uniform(0,2*math.pi,10000)                    # 0 < Y < 2*PI
np.random.shuffle(X)
np.random.shuffle(Y)
f_Label = np.random.uniform(0,0,10000)
for Counter in range(0,10000):
    f_Label[Counter] = math.sin(X[Counter]+Y[Counter])          # f(X,Y) = sin(X+Y)
x_Train,y_Train,label_Train,x_Test,y_Test,label_Test,x_validation,y_validation,label_validation =
Random_Sampling(f_Label , X , Y)
```

پیش پردازش

با توجه به موارد ذکر شده در کلاس , لازم است مجموعه داده های X , Y نرمال شوند .

برای نرمال کردن داده ها با توجه به رابطه ی زیر لازم است ابتدا میانگین داده ها را از هر داده کم کنیم و سپس به انحراف معیار[5] آن تقسیم کنیم .

---

[1] Label
[2] Test dataset
[3] Train dataset
[4] Validation dataset
[5] Standard deviation

$$Z = \frac{x - \mu}{\sigma}$$

شکل 3 -2 : رابطه ی نرمال کردن داده ها

لازم به ذکر است میانگین و انحراف معیار ذکر شده مربوط به دادگان آموزش می باشند و از همین مقادیر برای نرمال کردن دادگان تست و ارزیابی استفاده می کنیم .

حال با استفاده از تابع زیر ورودی را نرمال می کنیم :

```
def Normalization(data , Mean , Standard_deviation):
    for Counter in range(0,len(data)):
        data[Counter] = (data[Counter] - Mean) / Standard_deviation
    return data
```

توجه شود نرمال کردن فقط برای دادگان ورودی صورت می گیرد (مجموعه داده های X , Y) و برچسب ها نیازی به نرمال شدن ندارند .

پیوست:

در سوال 2 , پیش از اجرای فایل اصلی , لازم است ابتدا یک بار فایل توابع (Function.py) اجرا شود .