به نام خدا

تمرین دوم یادگیری عمیق

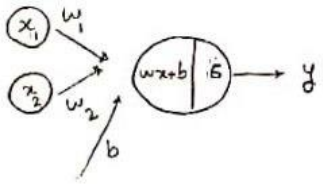
غزل زمانی‌نژاد

۹۷۵۲۲۱۶۶

1.

- **الف)**

۱)



بهتر است در ابتدا داده‌های ورودی را normalized کنیم:

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

$$x_1' = \frac{22-22}{\underbrace{60-22}_{38}} = 0 \qquad x_2' = \frac{25-22}{38} = 0.078 \qquad x_3' = \frac{47-22}{38} = 0.657$$

$$x_4' = \frac{52-22}{38} = 0.789 \qquad x_5' = \frac{46-22}{38} = 0.631 \qquad x_6' = \frac{56-22}{38} = 0.894$$

$$x_7' = \frac{55-22}{38} = 0.868 \qquad x_8' = \frac{60-22}{38} = 1$$

data →	x_1	x_2	y
①	0	1	0
②	0.078	0	0
③	0.657	1	1
④	0.789	0	0
⑤	0.631	1	1
⑥	0.894	1	1
⑦	0.868	0	0
⑧	1	0	1

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_1} = \left(\frac{-y}{a} + \frac{1-y}{1-a}\right) \times a(1-a) x_1$$

$$= (a-y)\, x_1$$

$$\partial L / \partial w_2 = (a-y)\, x_2$$ (مشابه بالا)

$$\partial L / \partial b = a - y$$

epoch ①

data ① $\quad z = w_1 x_1 + w_2 x_2 + b = 0 + 1 + 1 = 2 \qquad a = G(z) = \frac{1}{1+e^{-2}} = 0.88$

$$loss = -\left(0 \log a + \log(\underbrace{1-0.88}_{0.12})\right) = +0.92$$

data ② $\quad z = 0.078 + 0 + 1 = 1.078 \qquad a = G(1.078) = 0.746$

$$loss = -\left(0 \log(0.746) + \log(0.254)\right) = 0.595$$

$$loss = \tfrac{1}{2}(0.92 + 0.595) = 0.75$$

back propagate:

$$\partial L / \partial w_1 = \tfrac{1}{2}\left[(0.88-0)0 + (0.746-0)\,0.078\right] = 0.029$$

$$\partial L / \partial w_2 = \tfrac{1}{2}\left[(0.88-0)1 + (0.746-0)0\right] = 0.44$$

$$\partial L / \partial b = \tfrac{1}{2}\left[0.88 + 0.746\right] = 0.813$$

$w_1 = 1 - 0.05(0.029) = \boxed{0.99}$

$w_2 = 1 - 0.05(0.44) = \boxed{0.97}$

$b = 1 - 0.05(0.813) = \boxed{0.95}$

data ③ $z = 0.99 \times 0.657 + 0.97 \times 1 + 0.95 = 2.57$ $\quad a = \sigma(z) = 0.928$

$loss = -(\log(0.928) + 0 \log(1-a)) = 0.032$

data ④ $z = 0.99 \times 0.789 + 0.97 \times 0 + 0.95 = 1.73$ $\quad a = \sigma(z) = 0.849$

$loss = -(0 \log a + \underbrace{\log(1-0.849)}_{0.151}) = 0.82$

$loss = \frac{1}{2}[0.032 + 0.82] = 0.426$

backprop $\frac{\partial L}{\partial w_1} = \frac{1}{2}[\underbrace{(0.928-1)}_{-0.047}\ 0.657 + \underbrace{(0.849-0)}_{0.669}\ 0.789)] = 0.311$

$\frac{\partial L}{\partial w_2} = \frac{1}{2}[(0.928-1)\ 1 + (a-y)\ 0] = -0.036$

$\frac{\partial L}{\partial b} = \frac{1}{2}[(0.928-1) + (0.849-0)] = 0.388$

$w_1 = 0.99 - 0.05(0.311) = \boxed{0.97}$

$w_2 = 0.97 - 0.05(-0.036) = \boxed{0.97}$

$b = 0.95 - 0.05(0.388) = \boxed{0.93}$

data ⑤ $z = 0.97 \times 0.631 + 0.97 \times 1 + 0.93 = 2.512$ $\quad a = \sigma(z) = 0.924$

$loss = -(\log(0.924) + 0 \log(1-a)) = 0.034$

data ⑥ $z = 0.97 \times 0.894 + 0.97 \times 1 + 0.93 = 2.767$ $\quad a = \sigma(z) = 0.94$

$loss = -(\log(0.94) + 0 \log(1-a)) = 0.026$

$loss = \frac{1}{2}[0.034 + 0.026] = 0.03$

backprop $\partial L/\partial w_1 = \frac{1}{2}\left[\underbrace{(0.924-1)\,0.631}_{-0.047} + \underbrace{(0.94-1)\,0.894}_{-0.053}\right] = -0.05$

$\partial L/\partial w_2 = \frac{1}{2}\left[\underbrace{(0.924-1)^{x1}}_{-0.076} + \underbrace{(0.94-1)^{x1}}_{-0.06}\right] = -0.068$

$\partial L/\partial b = \frac{1}{2}\left[(0.924-1) + (0.94-1)\right] = -0.068$

$w_1 = 0.97 - 0.05(-0.05) = \boxed{0.972}$

$w_2 = 0.97 - 0.05(-0.068) = \boxed{0.973}$

$b = 0.93 - 0.05(-0.068) = \boxed{0.933}$

data ⑦ $z = 0.972 \times 0.868 + 0.973 \times 0 + 0.933 = 1.776 \quad a = 0.85$

$\text{loss} = -(0 \log a + 1 \log(1-0.85)) = 0.82$

data ⑧ $z = 0.972 \times 1 + 0.973 \times 0 + 0.933 = 1.905 \quad a = \sigma(z) = 0.87$

$\text{loss} = -(\log(0.87) + 0 \log(1-a)) = 0.06$

$\text{loss} = \frac{1}{2}[0.82 + 0.06] = 0.44$

backprop $\partial L/\partial w_1 = \frac{1}{2}\left[(0.85-0)\,0.868 + \underbrace{(0.87-1)}_{-0.13}\,1\right] = 0.303$

$\partial L/\partial w_2 = \frac{1}{2}\left[(0.85-0)\,0 + (0.87-1)\,0\right] = 0$

$\partial L/\partial b = \frac{1}{2}\left[(0.85-0) + (0.87-1)\right] = 0.36$

$w_1 = 0.972 - 0.05(0.303) = \boxed{0.956}$

$w_2 = 0.973 - 0.05(0) = \boxed{0.973}$

$b = 0.933 - 0.05(0.36) = \boxed{0.915}$

epoch ②

data ① $z = 0.956 \times 0 + 0.973 \times 1 + 0.915 = 1.88$ $\quad a = \sigma(z) = 0.867$

$loss = -(0 \log a + \underbrace{\log(1-0.867)}_{0.133}) = 0.87$

data ② $z = 0.956 \times 0.078 + 0.973 \times 0 + 0.915 = 0.989$ $\quad a = \sigma(z) = 0.524$

$loss = -(0 \log a + \underbrace{\log(1-0.524)}_{0.476}) = 0.322$

$loss = \frac{1}{2}[0.87 + 0.322]$

$= 0.596$

backprop $\partial L/\partial w_1 = \frac{1}{2}[(0.867-0)0 + (0.524-0)0.078] = 0.204$

$\partial L/\partial w_2 = \frac{1}{2}[(0.867-0)1 + (0.524-0)0] = 0.433$

$\partial L/\partial b = \frac{1}{2}[(0.867-0) + (0.524-0)] = 0.695$

$w_1 = 0.956 - 0.05(0.204) = \boxed{0.945}$

$w_2 = 0.973 - 0.05(0.433) = \boxed{0.951}$

$b = 0.915 - 0.05(0.695) = \boxed{0.88}$

data ③ $z = 0.945 \times 0.657 + 0.951 \times 1 + 0.88 = 2.451$ $\quad a = \sigma(z) = 0.92$

$loss = -(\log(0.92) + 0 \log(1-a)) = 0.036$

data ④ $z = 0.945 \times 0.789 + 0.951 \times 0 + 0.88 = 1.625$ $\quad a = \sigma(z) = 0.835$

$loss = -(0 \log a + \underbrace{\log(1-0.835)}_{0.165}) = 0.782$

$loss = \frac{1}{2}(0.036 + 0.782)$

$= 0.409$

back prop $\partial L/\partial w_1 = \frac{1}{2}[(\underbrace{0.92-1}_{-0.08})0.657 + (0.835-0)0.789] = 0.303$

$\partial L/\partial w_2 = \frac{1}{2}[(\underbrace{0.92-1}_{-0.08})1 + (0.835-0)0] = -0.04$

$\partial L/\partial b = \frac{1}{2}[(\underbrace{0.92-1}_{-0.08}) + (0.835-0)] = 0.377$

$w_1 = 0.945 - 0.05(0.303) = \boxed{0.929}$

$w_2 = 0.951 - 0.05(-0.04) = \boxed{0.953}$

$b = 0.88 - 0.05(0.377) = \boxed{0.861}$

data ⑤ $z = 0.929 \times 0.631 + 0.953 \times 1 + 0.861 = 2.74$ $\quad a = \sigma(z) = 0.916$

$\text{loss} = -(\log(0.916) + 0\log(1-a)) = 0.038$

data ⑥ $z = 0.929 \times 0.894 + 0.953 \times 1 + 0.861 = 2.644$ $\quad a = \sigma(z) = 0.933$

$\text{loss} = -(\log(0.933) + 0\log(1-a)) = 0.030$

$\text{loss} = 0.034$

backprop $\partial L/\partial w_1 = \frac{1}{2}[(0.916-1)\,0.631 + (0.933-1)\,0.894] = 0.056$

$\partial L/\partial w_2 = \frac{1}{2}[\underbrace{(0.916-1)}_{-0.084} + \underbrace{(0.933-1)}_{-0.067}] = -0.07$

$\partial L/\partial b = \frac{1}{2}[(0.916-1) + (0.933-1)] = -0.07$

$w_1 = 0.929 - 0.05(0.056) = \boxed{0.926}$

$w_2 = 0.953 - 0.05(-0.07) = \boxed{0.956}$

$b = 0.861 - 0.05(-0.07) = \boxed{0.864}$

data ⑦ $z = 0.926 \times 0.868 + 0.956 \times 0 + 0.864 = 1.667$ $\quad a = \sigma(z) = 0.841$

$\text{loss} = -(0\log a + \underbrace{\log(1-0.841)}_{0.159}) = 0.798$

data ⑧ $z = 0.926 \times 1 + 0.956 \times 0 + 0.864 = 1.79$ $\quad a = \sigma(z) = 0.856$

$\text{loss} = -(\log(0.856) + 0\log(1-a)) = 0.067$

backprop $\partial L/\partial w_1 = \frac{1}{2}[(0.841 - 0)0.868 + \underbrace{(0.856 - 1)}_{-0.144}] = 0.292$

$\partial L/\partial w_2 = \frac{1}{2}[(0.841 - 0)0 + (0.856 - 1)0] = 0$

$\partial L/\partial b = \frac{1}{2}[(0.841 - 0) + \underbrace{(0.856 - 1)}_{-0.144}] = 0.348$

$w_1 = 0.926 - 0.05\,(0.292) = 0.911$

$w_2 = 0.956 - 0.05\,(0) = 0.956$

$b = 0.864 - 0.05(0.348) = 0.846$

- **ب) هر دو روش GD و SGD برای مینیمم کردن میزان ارور در شبکه‌های عصبی به کار می‌روند. در روش GD در هر epoch بر مدل بر روی تمامی داده‌ها iterate می‌کند و برای آپدیت کردن وزن‌های شبکه از تمامی داده‌ها استفاده می‌کند. اما در SGD بر خلاف روش قبل تنها بر روی زیرمجوعه‌ای از داده‌ها iterate می‌کند. (در SGD بر روی یک داده، و در minibatch GD بر روی زیرمجموعه‌ای از داده‌ها مثلا به طول ۳۲)**

اگر تعداد نمونه‌های داده بسیار زیاد باشد، روشGD بسیار کند می‌شود چرا که باید در هر حلقه از تمامی داده‌ها برای آپدیت کردن وزن‌ها استفاده کند. اما SGD بسیار سریعتر عمل می‌کند.

SGD معمولا سریعتر از GD همگرا می‌شود اما تابع ارور در آن به خوبی GD کمینه نمی‌شود.

2. ابتدا نقاط را به صورت numpy array درمی‌آوریم. شکل آنها را به گونه‌ای تغییر می‌دهیم که در توابع کتابخانه sklearn قابل استفاده باشند.

```
x = np.array([2.3, 1.4, 2.6, 3.1, 1.8, 2.8, 5.4, 6.3, 5.8, 6.7, 4.9, 45.2])
y = np.array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1])

n_samples = 12
x = x.reshape(n_samples, 1)
y = y.ravel()
print("shape of x", x.shape)
print("shape of y", y.shape)
```

```
shape of x (12, 1)
shape of y (12,)
```

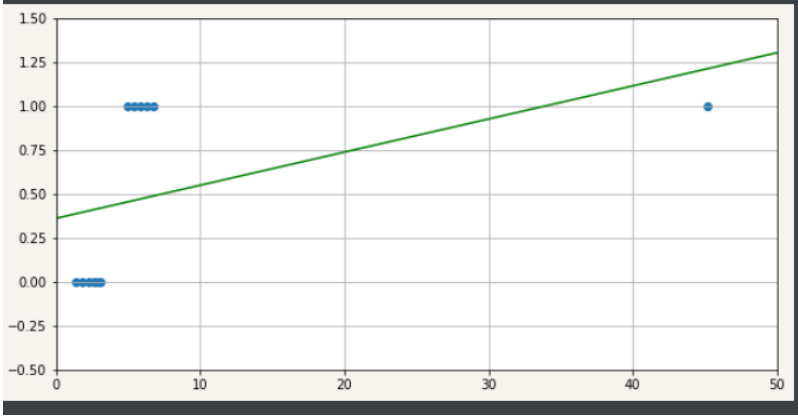
سپس به کمک linear_model.LinearRegression موجود در کتابخانه sklearn یک مدل پیاده‌سازی می‌کنیم. آن را بر روی داده‌ها آموزش (fit) می‌دهیم. و سپس وزن و بایاس محاسبه شده را چاپ می‌کنیم.

```
linear = linear_model.LinearRegression()
linear.fit(x, y)
W = linear.coef_
b = linear.intercept_
print("trained W with linear regression", W)
print("trained bias with linear regression", b)
```

```
trained W with linear regression [0.01885095]
trained bias with linear regression 0.36128845022604983
```

بعد تابع پیش‌بینی شده توسط این مدل که یک خط با شیب W و عرض از مبدا b است را رسم می‌کنیم.

```
plt.pyplot.figure(1, figsize=(10, 5))
xx = np.linspace(0, 50)
yy = xx * W + b
plt.pyplot.plot(xx, yy, 'g')

plt.pyplot.scatter(x, y)
plt.pyplot.xlim(0, 50)
plt.pyplot.ylim(-0.5, 1.5)
plt.pyplot.grid()
plt.pyplot.show()
```



y = Wx + b → if (y = 0.5) then x = 7.35

مرز تصمیم:

(7.35, 0.5)

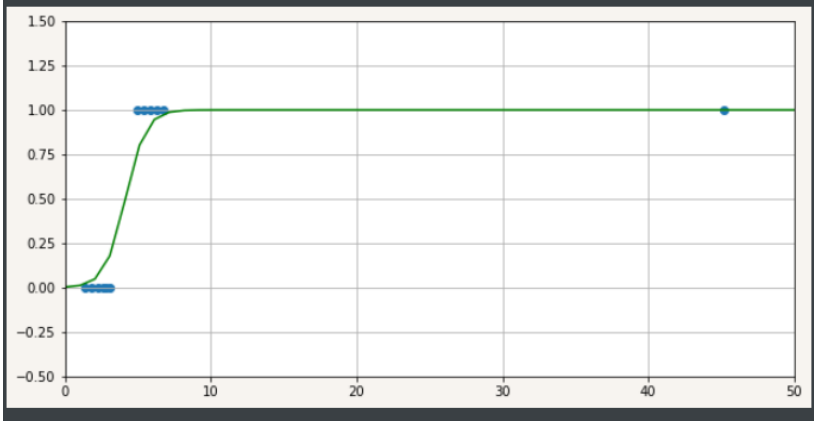
سپس همین مراحل را با استفاده از linear_model.LogisticRegression **تکرار می‌کنیم. یک مدل می‌سازیم و آن را آموزش می‌دهیم.**

```
logistic = linear_model.LogisticRegression()
logistic.fit(x, y)
W_log = logistic.coef_
b_log = logistic.intercept_
print("trained W with logistic regression", W_log)
print("trained biad with logistic regression", b_log)
```

```
trained W with logistic regression [[1.43516077]]
trained biad with logistic regression [-5.9247354]
```

تابع پیش‌بینی شده از مدل (خطی است که از یک تابع sigmoid **عبور کرده.) را رسم می‌کنیم.**

```python
def sigmoid(x):
  return 1 / (1 + np.exp(-1 * x))

plt.pyplot.figure(1, figsize=(10, 5))
xx = np.linspace(0, 50)
yy = sigmoid(xx * W_log[0][0] + b_log[0])
plt.pyplot.plot(xx, yy, 'g')

plt.pyplot.scatter(x, y)
plt.pyplot.xlim(0, 50)
plt.pyplot.ylim(-0.5, 1.5)
plt.pyplot.grid()
plt.pyplot.show()
```



y = sigmoid(W x + b) → if (y = 0.5) then x = 4.12

مرز تصمیم:

(4.12, 0.5)

در linear regression مدل یک خط را پیش‌بینی می‌کند و یک خط برای مسائل classification ممکن است مناسب نباشد و دارای بایاس زیادی باشد. اما در logistic regression مدل یک curve را پیش‌بینی می‌کند و منحنی می‌تواند برای جداسازی کلاس‌های گوناگون بهتر عمل کند. linear regression بیشتر برای مسائل regression به کار می‌رود (و نه مسائل دسته‌بندی) در linear regression خروجی پیوسته است اما در logistic regression خروجی تابع سیگموید بین ۰ و ۱ است که پس از مقایسه با threshold خروجی به یک مقدار باینری assign می‌شود.

3. **در این سوال می‌خواهیم الگوریتم Logistic Regression را بر روی دیتاست Iris پیاده‌سازی کنیم.**
ابتدا کتابخانه‌های مورد نیاز را داخل کد import می‌کنیم. به کمک کتابخانه sklearn دیتاست را لود میکنیم.
سپس با کمک تابع train_test_split داده‌ها را به دو دسته آموزش و تست با نسبت ۷ و ۳ تقسیم می‌کنیم.
بعد شکل داده‌های آموزش و تست را چاپ می‌کنیم. بعد نمونه‌ای از کلاس LogisticRegression می‌سازیم.

پارامتر multi_class آن را بر روی multinomial ست می‌کنیم چون این مسئله ۳ کلاسه است. مدل را بر روی داده‌های آموزشی با استفاده از fit آموزش می‌دهیم.

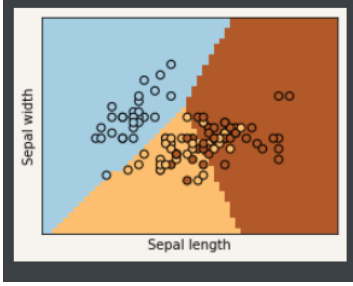
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn import metrics
import seaborn as sns

iris = datasets.load_iris()
#need first two features from flowers
X = iris.data[:, :2]
Y = iris.target

x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, shuffle=True)
print("x train shape:", x_train.shape)
print("y train shape", y_train.shape)
print("x test shape:", x_test.shape)
print("y test shape:", y_test.shape)


regression = LogisticRegression(multi_class='multinomial')
# train the model on train dataset
regression.fit(x_train, y_train)

```

```
x train shape: (105, 2)
y train shape (105,)
x test shape: (45, 2)
y test shape: (45,)
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='multinomial', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

الف) دیتاست Iris برای طبقه‌بندی ۳ نوع گل setosa، versicolor و virginica طراحی شده‌است. شامل ۱۵۰ داده است که برای هر داده ۴ ویژگی استخراج شده‌است. برای سادگی کار تنها از دو ویژگی اول استفاده می‌کنیم.

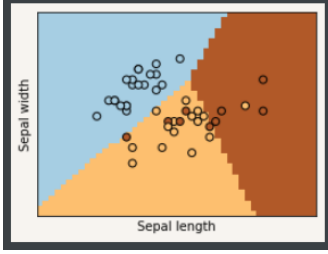
ب) برای رسم داده‌های آموزشی ابتدا decision boundaries را رسم می‌کنیم. سپس نقاط مربوط به هر کلاس را رسم می‌کنیم.

```python
x_min, x_max = x_train[:, 0].min()-1, x_train[:, 0].max()+1 # x axis
y_min, y_max = x_train[:, 1].min()-1, x_train[:, 1].max()+1 # y axis

# plot the boundaries
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1), np.arange(y_min, y_max, 0.1))
Z = regression.predict(np.c_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

# Plot the training points
plt.scatter(x_train[:, 0], x_train[:, 1], c=y_train, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.xticks(())
plt.yticks(())

plt.show()
```



ج) برای رسم داده‌های تست مشابه داده‌های آموزشی عمل می‌کنیم.

```python
x_min2, x_max2 = x_test[:, 0].min()-1, x_test[:, 0].max()+1 # x axis
y_min2, y_max2 = x_test[:, 1].min()-1, x_test[:, 1].max()+1 # y axis

# plot the boundaries
xx2, yy2 = np.meshgrid(np.arange(x_min2, x_max2, 0.1), np.arange(y_min2, y_max2, 0.1))
Z2 = regression.predict(np.c_[xx2.ravel(), yy2.ravel()])

Z2 = Z2.reshape(xx2.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx2, yy2, Z2, cmap=plt.cm.Paired)

# Plot the training points
plt.scatter(x_test[:, 0], x_test[:, 1], c=y_test, edgecolors='k', cmap=plt.cm.Paired)
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

plt.xlim(xx2.min(), xx2.max())
plt.ylim(yy2.min(), yy2.max())
plt.xticks(())
plt.yticks(())

plt.show()
```



د) برای محاسبه دقت از فرمول زیر استفاده می‌کنیم:

$$\text{Accuracy} = \frac{\text{True}}{\text{Total}}$$

یعنی پس از پیش‌بینی نتیجه داده آموزش و تست (به کمک مدل) آن را با لیبل اصلی مقایسه می‌کنیم. هرچه تعداد بیشتری از پیش‌بینی و لیبل باهم برابر باشند دقت بیشتر است.

در اینجا دقت آموزش و تست تقریبا باهم برابر هستند. چون LogisticRegression که در sklearn پیاده‌سازی شده مقداری از روش‌های Regularization استفاده می‌کند، مدل overfit نشده و توانسته به خوبی generalize شود و با دقت خوبی بر روی داده تست عمل کند.

```python
pred = regression.predict(x_train)
score = np.equal(pred, y_train).sum() / y_train.size
print("train set accuracy:", score)
```

```
train set accuracy: 0.8285714285714286
```

```python
pred_test = regression.predict(x_test)
score_test = np.equal(pred_test, y_test).sum() / y_test.size
print("test set accuracy:", score_test)
```

```
test set accuracy: 0.8222222222222222
```

ه) در این سوال با استفاده از metrics.confusion_matrix (موجود در کتابخانه sklearn) به رسم ماتریس confusion می‌پردازیم.

در ماتریس confusion هرچه اعداد روی قطر اصلی ماتریس بزرگتر باشند بهتر است.

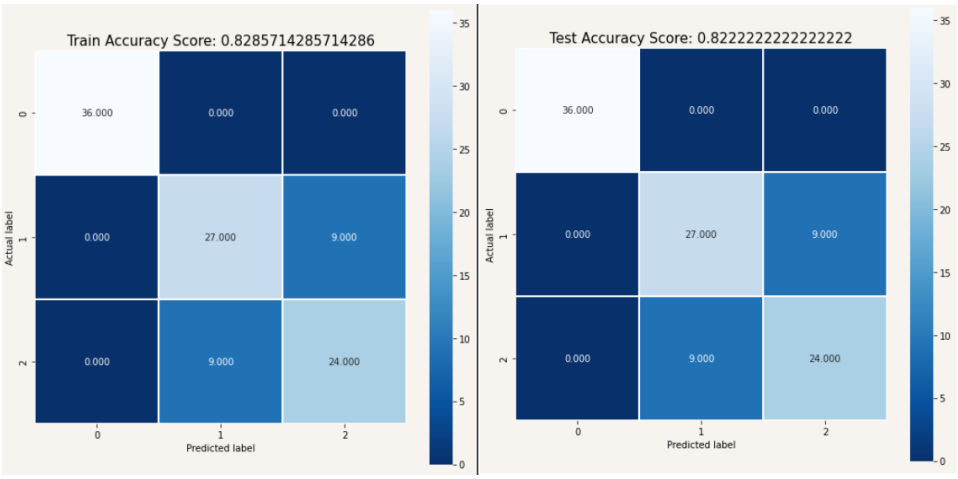
```
cm = metrics.confusion_matrix(y_train, pred)
print("confustion matrix of train set \n", cm)

cm_test = metrics.confusion_matrix(y_test, pred_test)
print("confustion matrix of test set \n", cm_test)

plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Train Accuracy Score: {0}'.format(score)
plt.title(all_sample_title, size = 15);


plt.figure(figsize=(9,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square = True, cmap = 'Blues_r');
plt.ylabel('Actual label');
plt.xlabel('Predicted label');
all_sample_title = 'Test Accuracy Score: {0}'.format(score_test)
plt.title(all_sample_title, size = 15);
```

```
confustion matrix of train set
 [[36  0  0]
 [ 0 27  9]
 [ 0  9 24]]
confustion matrix of test set
 [[14  0  0]
 [ 0 11  3]
 [ 0  5 12]]
```

به طور مثال به بررسی سطر سوم ماتریس داده آموزشی می‌پردازیم. در سطر سوم لیبل اصلی داده‌ها از نوع کلاس ۳ است. طبق پیش‌بینی مدل، ۰ تا از داده‌ها متعلق به کلاس ۱، ۹ تا از داده‌ها متعلق به کلاس ۲ و ۲۴تا از داده‌ها به کلاس ۳ تعلق دارند. یعنی مدل ۲۴ داده را به صورت درست پیش‌بینی کرده است.

با توجه به این ماتریس، در موارد زیادی مدل لیبل ۲ و ۳ را به صورت اشتباه پیش‌بینی کرده است. به طور مثال می‌توانیم از تعداد داده‌های بیشتر برای برطرف کردن این مشکل استفاده کنیم.

برای نمایش بهتر این ماتریس‌ها از کتابخانه seaborn استفاده می‌کنیم.



منابع استفاده‌شده:

https://datascience.stackexchange.com/questions/36450/what-is-the-difference-between-gradient-descent-and-stochastic-gradient-descent

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning

https://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html

https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a