دانشکده مهندسی کامپیوتر

مباحث ویژه ۱ (یادگیری عمیق)

تمرین ۲

علی صداقی

۹۷۵۲۱۳۷۸

سوال ۱

- Forward Pass:

$$z = x_1 w_1 + x_2 w_2 + b$$

$$\hat{y} = a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$L_{single\ example}(a, y) = -(y log(a) + (1 - y) \log(1 - a))$$

$$J_{batch} = \frac{1}{N} \sum_{i=1}^{N} L(a_i, y_i) = -\frac{1}{N} \sum_{i=1}^{N} (y_i log(a_i) + (1 - y_i) log\ (1 - a_i))$$

- Backward pass for single example:

$$\frac{dL(a, y)}{da} = -\frac{y}{a} + \frac{1 - y}{1 - a}$$

$$\frac{da}{dz} = \frac{-(-e^{-z})}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \times \frac{e^{-z} + 1 - 1}{1 + e^{-z}} = \sigma(z) \times \big(1 - \sigma(z)\big) = a(1 - a)$$

$$\frac{dL}{dz} = \frac{dL}{da} \times \frac{da}{dz} = \left(-\frac{y}{a} + \frac{1 - y}{1 - a}\right) \times a(1 - a) = a - y$$

$$\frac{dL}{dw_1} = \frac{dL}{dz} \times \frac{dz}{dw_1} = (a - y) x_1$$

$$\frac{dL}{dw_2} = \frac{dL}{dz} \times \frac{dz}{dw_2} = (a - y) x_2$$

$$\frac{dL}{db} = \frac{dL}{dz} \times \frac{dz}{db} = a - y$$

- Backward pass for N example:

$$\frac{dJ}{dw_1} = \frac{1}{N} \sum_{i=1}^{N} (a_i - y_i) x_{1-i}$$

$$\frac{dJ}{dw_2} = \frac{1}{N} \sum_{i=1}^{N} (a_i - y_i) x_{2-i}$$

$$\frac{dJ}{db} = \frac{1}{N} \sum_{i=1}^{N} (a_i - y_i)$$

- Update Parameters:

$$w_1 = w_1 - \alpha \frac{dJ}{dw_1}$$

$$w_2 = w_2 - \alpha \frac{dJ}{dw_2}$$

$$b = b - \alpha \frac{dJ}{db}$$

با توجه به اینکه محدوده مقادیر سن بسیار بزرگ است حاصل ضرب آن در وزن مربوطه اعداد بزرگی تولید می‌کند، در نتیجه مقدار z بزرگ می‌شود و در نتیجه آن مقدار a بسیار نزدیک به عدد ۱ می‌شود. این اتفاقات باعث می‌شود دچار Exploding Gradient شویم. پس ابتدا داده‌های مربوط به سن را به گونه‌ای نرمال می‌کنیم که میانگین ۰ و انحراف معیار ۱ داشته باشد.

```
x_train -= np.mean(x_train)
x_train /= np.std(x_train)
```

x_1	x_2	y
-1.749	*1*	*0*
-1.525	*0*	*0*
0.121	*1*	*1*
0.495	*0*	*0*
0.046	*1*	*1*
0.795	*1*	*1*
0.720	*0*	*0*
1.094	*0*	*1*

با توجه به روابط به دست آمده مسئله را حل می‌کنیم.

$$w_1 = 1$$

$$w_2 = 1$$

$$b = 1$$

$$N = 2$$

$$\alpha = 0.005$$

1. Epoch 1

Loss = 5.283563725422388

1.1. Batch 1

x_1	x_2	y
-1.749	*1*	*0*
-1.525	*0*	*0*

$$z_1 = (-1.749) \times 1 + 1 \times 1 + 1 = 0.251 \Longrightarrow a_1 = \frac{1}{1 + e^{-0.251}} = 0.562$$

$$z_2 = (-1.525) \times 1 + 0 \times 1 + 1 = -0.525 \Longrightarrow a_2 = \frac{1}{1 + e^{0.525}} = 0.371$$

$$\frac{dJ}{dw_1} = \frac{1}{2}\big((0.562 - 0) \times (-1.749) + (0.371 - 0) \times (-1.525)\big) = -0.775$$

$$\frac{dJ}{dw_2} = \frac{1}{2}\big((0.562 - 0) \times (1) + (0.371 - 0) \times (0)\big) = 0.281$$

$$\frac{dJ}{db} = \frac{1}{2}\big((0.562 - 0) + (0.371 - 0)\big) = 0.467$$

$$w_1 = 1 - 0.005 \times (-0.775) = 1.003$$

$$w_2 = 1 - 0.005 \times (0.281) = 0.998$$

$$b = 1 - 0.005 \times (0.467) = 0.997$$

1.2. Batch 2

$$w_1 = 1.002$$

$$w_2 = 0.998$$

$$b = 0.995$$

1.3. Batch 3

$$w_1 = 1.003$$

$$w_2 = 0.999$$

$$b = 0.996$$

1.4. Batch 4

$$w_1 = 1.001$$

$$w_2 = 0.999$$

$$b = 0.994$$

Loss = 5.269889269044415

2. Epoch 2

2.1. Batch 1

$$w_1 = 1.005$$

$$w_2 = 0.997$$

$$b = 0.992$$

2.2. Batch 2

$$w_1 = 1.004$$

$$w_2 = 0.998$$

$$b = 0.990$$

2.3. Batch 3

$$w_1 = 1.004$$

$$w_2 = 0.998$$

$$b = 0.990$$

2.4. Batch 4

$$w_1 = 1.003$$

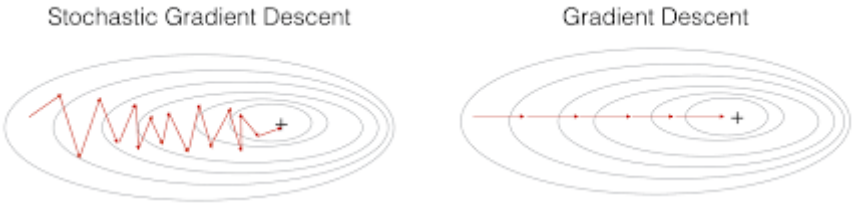
$$w_2 = 0.998$$

$$b = 0.988$$

Loss = 5.25632016967364

قسمت ب) در روش عادی کل دیتا با هم در یک Batch وارد شبکه می‌شود اما در روش تصادفی ابتدا دسته‌هایی کوچکتر (Mini-Batch) ساخته می‌شود سپس آن‌ها به ترتیب وارد شبکه می‌شوند و به ازای هر دسته پارامتر های شبکه آپدیت می‌شود.

Stochastic Gradient Descent	Gradient Descent
داده‌های در بسته‌های کوچکتری (معمولا توان ۲) وارد شبکه می‌شوند.	کل داده‌ها در هر ایپاک همزمان وارد شبکه می‌شوند.
تعداد Epoch کمتری نیاز دارد.	تعداد Epoch زیادی نیاز دارد.
مقدار گرادیان سریع‌تر محاسبه می‌شود.	زمان بیشتری برای محاسبه گرادیان نیاز است.
برای داده‌های حجیم مناسب‌تر است.	برای داده‌هایی با حجم کم مناسب تر است.
حافظه کمتری را اشغال می‌کند.	حافظه زیادی نیاز دارد.
احتمال گذر از نقطه min محلی بیشتر است.	احتمال بیشتری وجود دارد تا در نقطه min محلی گیر کنیم.
مقدار loss نوسان زیادی دارد.	مقدار loss یکنوا تر است.
سریع‌تر converge می‌شود.	دیرتر converge می‌شود.
ممکن است از نقطه min گلوبال عبور کند.	احتمال عبور از نقطه min گلوبال کمتر است.
داده‌ها باید قبل از وارد شدن به شبکه shuffle بشوند. (جلوگیری از bias)	نیازی به shuffle نیست.



مشکلات و مزایا هر یک در جدول بالا اشاره شد. در واقع روش عادی در عمل استفاده نمی‌شود زیرا برای دیتاست‌های کنونی بسیار کند است و نیاز به حافظه بسیار زیادی دارد. جهت کم کردن نوسان‌ها می‌توانیم از بهینه‌ساز هایی نظیر Adam، Momentum، RMSprop استفاده کنیم. این بهینه‌ساز ها احتمال رد کردن از نقطه min global را نیز کم می‌کنند.

سوال ۲) ابتدا داده‌های ورودی را همانند سوال قبلی نرمال می‌کنیم.

یک شبکه طراحی می‌کنیم که با یک آرگومانی بولی به‌نام kind می‌توان "Logistic" یا "Linear" بودن رگرسیون آن را تنظیم کرد.

تفاوت دو حالت خطی و لجستیک در موارد زیر است:

۱. در حالت رگرسیون خطی دیگر از تابع فعالسازی زیگموید استفاده نمی‌کنیم و مقدار نهایی همان z است. در صورت که در حالت رگرسیون لجستیک مقدار نهایی a می‌باشد که برابر زیگموید مقدار z است.

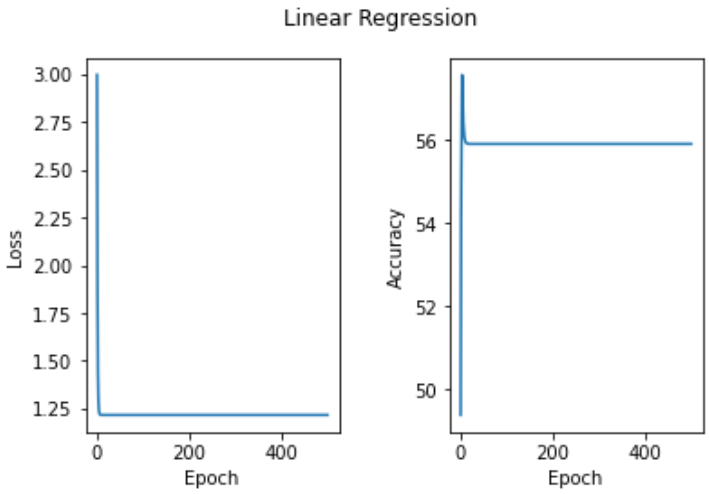
۲. در حالت رگرسیون خطی از تابع ضرر Mean Square Error استفاده می‌کنیم اما در حالت لجستیک از Cross Entropy استفاده می‌کنیم. دلیل این کار مربوط به محدوده خروجی نورون است، چون در حالت لجستیک مقدار خروجی بین ۰ تا ۱ است تابع ضرر کراس‌آنتروپی نتیجه بهتری می‌دهد. (بیشینه کردن likelihood)

با هایپرپارامترهای زیر آموزش را شروع می‌کنیم:

```
LEARNING_RATE = 0.4
EPOCHS = 500
```

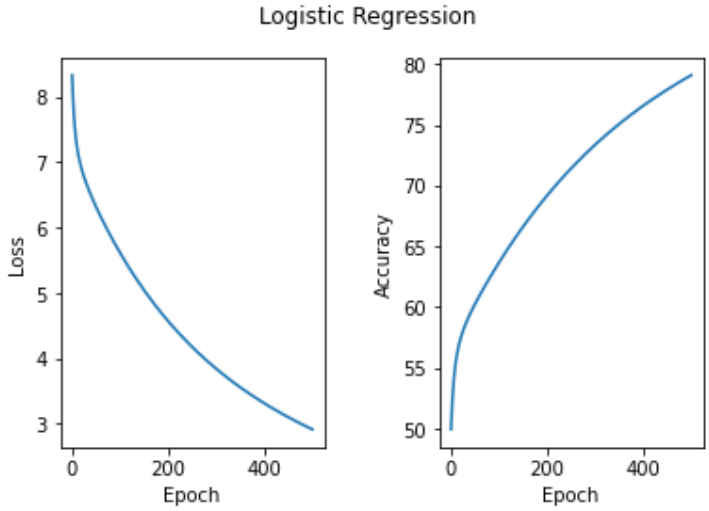
نتایج Linear Regression:

```
Epoch:   0    Loss: 2.996092    Accuracy: 49.388244
Epoch:  20    Loss: 1.215822    Accuracy: 55.917137
Epoch:  40    Loss: 1.215822    Accuracy: 55.916747
Epoch:  60    Loss: 1.215822    Accuracy: 55.916747
Epoch:  80    Loss: 1.215822    Accuracy: 55.916747
Epoch: 100    Loss: 1.215822    Accuracy: 55.916747
Epoch: 120    Loss: 1.215822    Accuracy: 55.916747
Epoch: 140    Loss: 1.215822    Accuracy: 55.916747
Epoch: 160    Loss: 1.215822    Accuracy: 55.916747
Epoch: 180    Loss: 1.215822    Accuracy: 55.916747
Epoch: 200    Loss: 1.215822    Accuracy: 55.916747
Epoch: 220    Loss: 1.215822    Accuracy: 55.916747
Epoch: 240    Loss: 1.215822    Accuracy: 55.916747
Epoch: 260    Loss: 1.215822    Accuracy: 55.916747
Epoch: 280    Loss: 1.215822    Accuracy: 55.916747
Epoch: 300    Loss: 1.215822    Accuracy: 55.916747
Epoch: 320    Loss: 1.215822    Accuracy: 55.916747
Epoch: 340    Loss: 1.215822    Accuracy: 55.916747
Epoch: 360    Loss: 1.215822    Accuracy: 55.916747
Epoch: 380    Loss: 1.215822    Accuracy: 55.916747
Epoch: 400    Loss: 1.215822    Accuracy: 55.916747
Epoch: 420    Loss: 1.215822    Accuracy: 55.916747
Epoch: 440    Loss: 1.215822    Accuracy: 55.916747
Epoch: 460    Loss: 1.215822    Accuracy: 55.916747
Epoch: 480    Loss: 1.215822    Accuracy: 55.916747
Epoch: 499    Loss: 1.215822    Accuracy: 55.916747
```

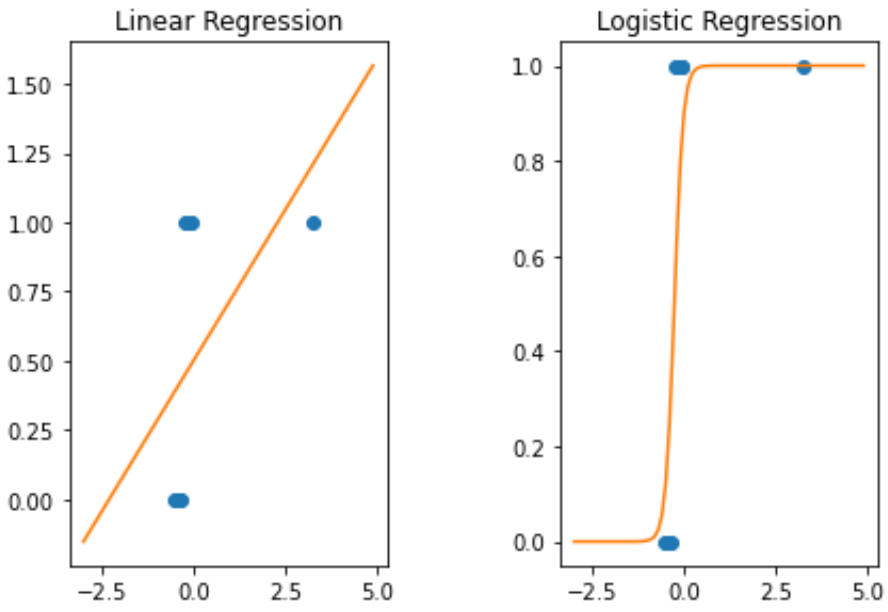


نتایج Logistic Regression:

```
Epoch:   0    Loss: 8.331774    Accuracy: 49.942531
Epoch:  20    Loss: 6.886894    Accuracy: 57.442581
Epoch:  40    Loss: 6.478519    Accuracy: 59.477951
Epoch:  60    Loss: 6.157606    Accuracy: 61.003621
Epoch:  80    Loss: 5.870836    Accuracy: 62.387551
Epoch: 100    Loss: 5.608130    Accuracy: 63.687499
Epoch: 120    Loss: 5.365934    Accuracy: 64.916133
Epoch: 140    Loss: 5.142072    Accuracy: 66.078265
Epoch: 160    Loss: 4.934783    Accuracy: 67.177344
Epoch: 180    Loss: 4.742521    Accuracy: 68.216640
Epoch: 200    Loss: 4.563905    Accuracy: 69.199425
Epoch: 220    Loss: 4.397690    Accuracy: 70.128949
Epoch: 240    Loss: 4.242758    Accuracy: 71.008407
Epoch: 260    Loss: 4.098103    Accuracy: 71.840889
Epoch: 280    Loss: 3.962822    Accuracy: 72.629359
Epoch: 300    Loss: 3.836104    Accuracy: 73.376637
Epoch: 320    Loss: 3.717216    Accuracy: 74.085386
Epoch: 340    Loss: 3.605504    Accuracy: 74.758111
Epoch: 360    Loss: 3.500374    Accuracy: 75.397156
Epoch: 380    Loss: 3.401294    Accuracy: 76.004711
Epoch: 400    Loss: 3.307781    Accuracy: 76.582819
Epoch: 420    Loss: 3.219401    Accuracy: 77.133375
Epoch: 440    Loss: 3.135760    Accuracy: 77.658143
Epoch: 460    Loss: 3.056501    Accuracy: 78.158760
Epoch: 480    Loss: 2.981301    Accuracy: 78.636742
Epoch: 499    Loss: 2.913351    Accuracy: 79.071144
```



مقایسه نتیجه تفکیک و مرزبندی:



هر یک از روش‌های بالا کاربرد مخصوص خود را دارند و نباید در یک مسئله مشخص جایگزین همدیگر شوند. رگرسیون خطی برای مسائلی مناسب است که خروجی مد نظر یک طیف پیوسته و نامحدود می‌باشد، به همین دلیل است که تابع فعالسازی نیز ندارد و از تابع ضرر MSE استفاده می‌کند. از کاربردهای رگرسیون خطی می‌توان به تخمین قیمت خانه، تخمین قد افراد، تخمین سن و ... اشاره کرد.

روش رگرسیون لجستیک برای مسائل دسته‌بندی (Classification) مناسب و با توجه به تعداد کلاس‌ها می‌تواند تابع فعال‌سازی Sigmoid یا Softmax داشته باشد. همچنین بهترین تابع ضرر برای آن Cross Entropy (باینری) یا Categorical Cross Entropy (مولتی کلاس) است. از کاربردهای این روش می‌توان به تشخیص احساسات، تشخیص اعداد، تشخیص جنسیت و ... اشاره کرد.

مسئله‌ای که حل کردیم یک مسئله دسته‌بندی (توانایی پرداخت یا عدم آن) بود پس روش Logistic برای آن بهتر است.

روش رگرسیون خطی همواره فضا را با یک خط تخمین میزند و می‌توان بالا یا پایین آن را به منظره دسته‌ها در نظر گرفت. اما روش لجستیک به صورت Multinomial این کاررا انجام می‌دهد و ناحیه را می‌تواند با شکل‌های غیر خطی دسته‌بندی کند.

مقادیر w و b برای مرز های تصمیم ارائه شده در نمودار ها به صورت زیر است (از مرز تصمیمی که در صورت سوال ذکر شده استفاده نکردیم و پیدا کردن بهترین مرز را به شبکه سپردیم):

Linear: w = 0.217, b = 0.5

```
[[0.21763043]]
[0.5]
```

Logistic: w = 8.363, b = 2.211

```
[[8.36348788]]
[2.21129158]
```

- ✓ تابع دقت پیاده‌سازی شده یک تابع ساده است که مقادیر واقعی و پیشبینی شده را از هم تفریق می‌کند و سپس میانگین می‌گیرد. به‌همین دلیل است که دقت گزارش شده پایین است. می‌توانستیم با argmax گرفتن Label های پیشبینی شده را بررسی کنیم اما چون دقت همواره نزدیک ۱۰۰ می‌شد (کم بودن داده‌ها) این کار را نکردیم.

سوال ۳) در حل این سوال از داکیومنت رسمی کتاب‌خانه scikit استفاده کردیم:

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

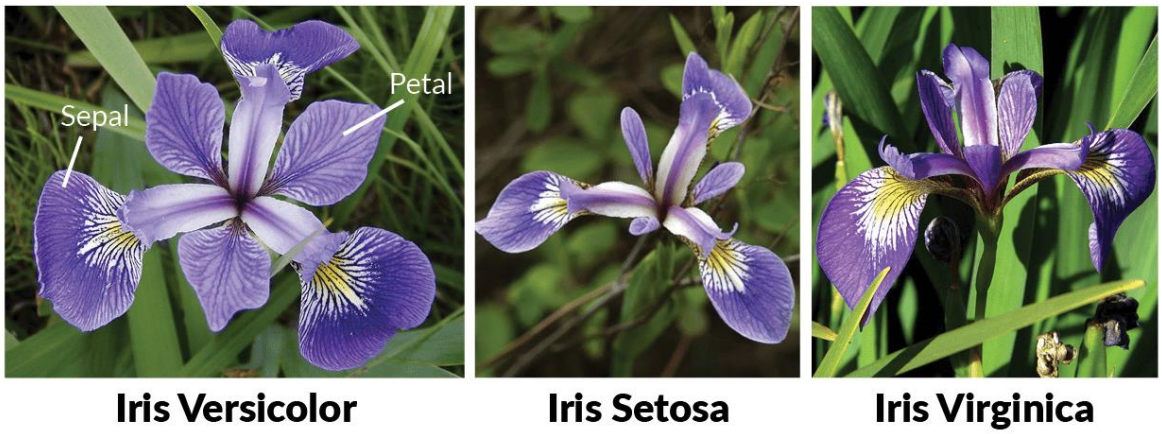
https://scikit-learn.org/stable/auto_examples/linear_model/plot_iris_logistic.html#sphx-glr-download-auto-examples-linear-model-plot-iris-logistic-py

الف) این دیتاست شامل ۱۵۰ داده از زنبق‌ها هست که هر یک دارای ۴ ویژگی زیر است:

1. Sepal Length
2. Sepal Width
3. Petal Length
4. Petal Width

هم‌چنین داده‌های این دیتاست در ۳ کلاس زیر موجود هستند:

1. Versicolor
2. Setosa
3. Virginica



shape دیتاست به صورت زیر است:

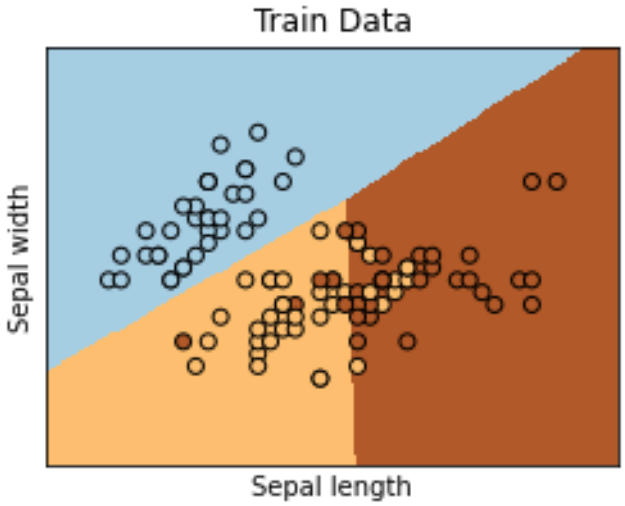
```
(150, 4): 4 features
(150,): 3 classes labeled as 0, 1, 2
```

ما در این سوال تنها با دو ویژگی اول این دیتاست کار داریم.

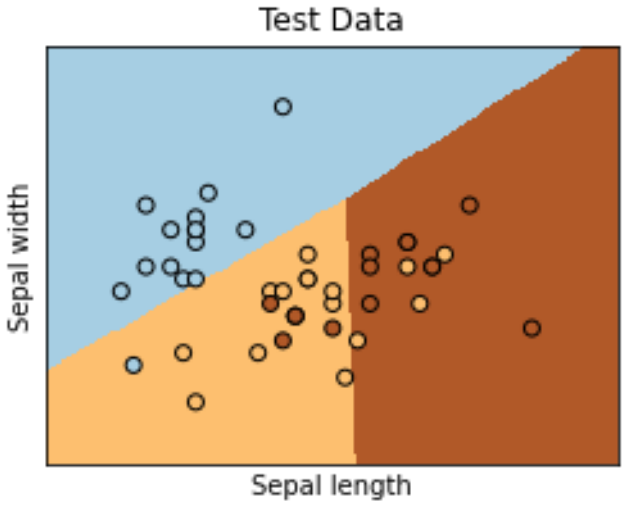
ابتدا دیتاست را shuffle می‌کنیم و ۳۰ درصد آن را به عنوان داده تست در نظر می‌گیریم یعنی:

```
(105, 2)
(105,)
(45, 2)
(45,)
```

ب) پس از آموزش بر روی داده آموزشی Decision Boundary مربوط به آن را رسم می‌کنیم. همانطور که مشاهده می‌کنید کلاس آبی به خوبی تفکیک شده اما در دو کلاس دیگر کمی نویز داریم. در واقع شبکه confuse زیادی در آن دو کلاس داشته است.



ج) پس از آموزش بر روی داده تست Decision Boundary مربوط به آن را رسم می‌کنیم. همانطور که مشاهده می‌کنید کلاس آبی به خوبی تفکیک شده اما در دو کلاس دیگر کمی نویز داریم. در واقع شبکه confuse زیادی در آن دو کلاس داشته است.



د) با استفاده از score نحوه عملکرد شبکه را بر روی داده آموزش و داده تست ارزیابی می‌کنیم.

```
Train Accuracy: 0.8476190476190476
Test Accuracy: 0.7333333333333333
```

با توجه به اینکه شبکه برروی داده train آموزش داده شده و آن را دیده به نوعی یک ذهنیت (bias) درباره آن دارد پس طبیعی است که دقت بیشتری روی داده آموزشی داشته باشیم. عملکرد شبکه برروی داده تست که برای اولین آن را می‌بیند نیز قابل قبول است.

در داده آموزشی از ۱۰۵ مورد ۸۲ مورد را درست حدس زدیم و تنها ۲۳ مورد اشتباه داشتیم. در داده تست از ۴۵ مورد ۳۳ مورد را درست و ۱۲ مورد را فلط حدس زدیم.

(٥

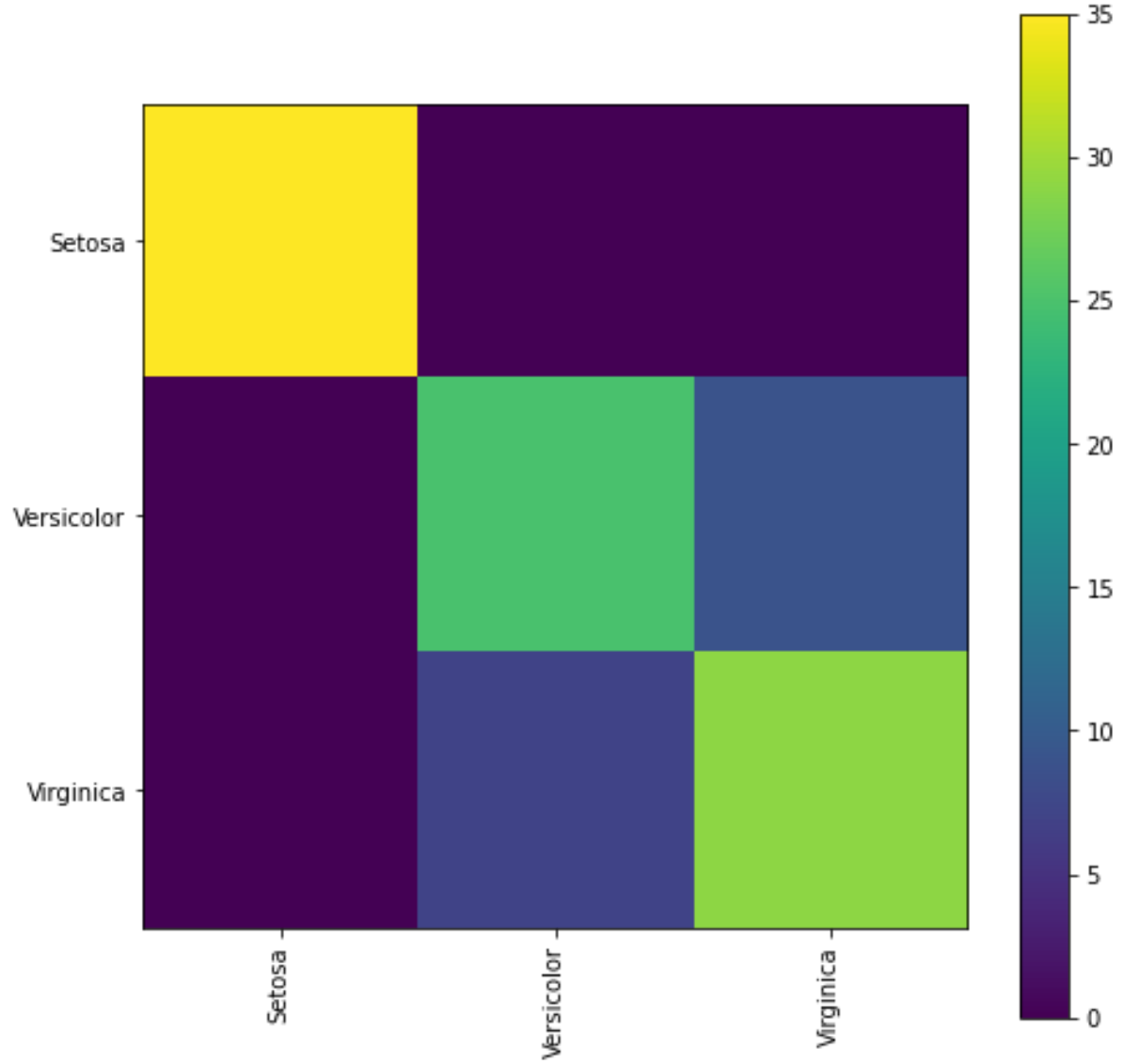
Train Data:

```
Confusion Matrix Trainset
[[35  0  0]
 [ 0 25  9]
 [ 0  7 29]]

Classification Report Trainset
              precision    recall  f1-score   support

      Setosa       1.00      1.00      1.00        35
  Versicolor       0.78      0.74      0.76        34
   Virginica       0.76      0.81      0.78        36

    accuracy                           0.85       105
   macro avg       0.85      0.85      0.85       105
weighted avg       0.85      0.85      0.85       105
```



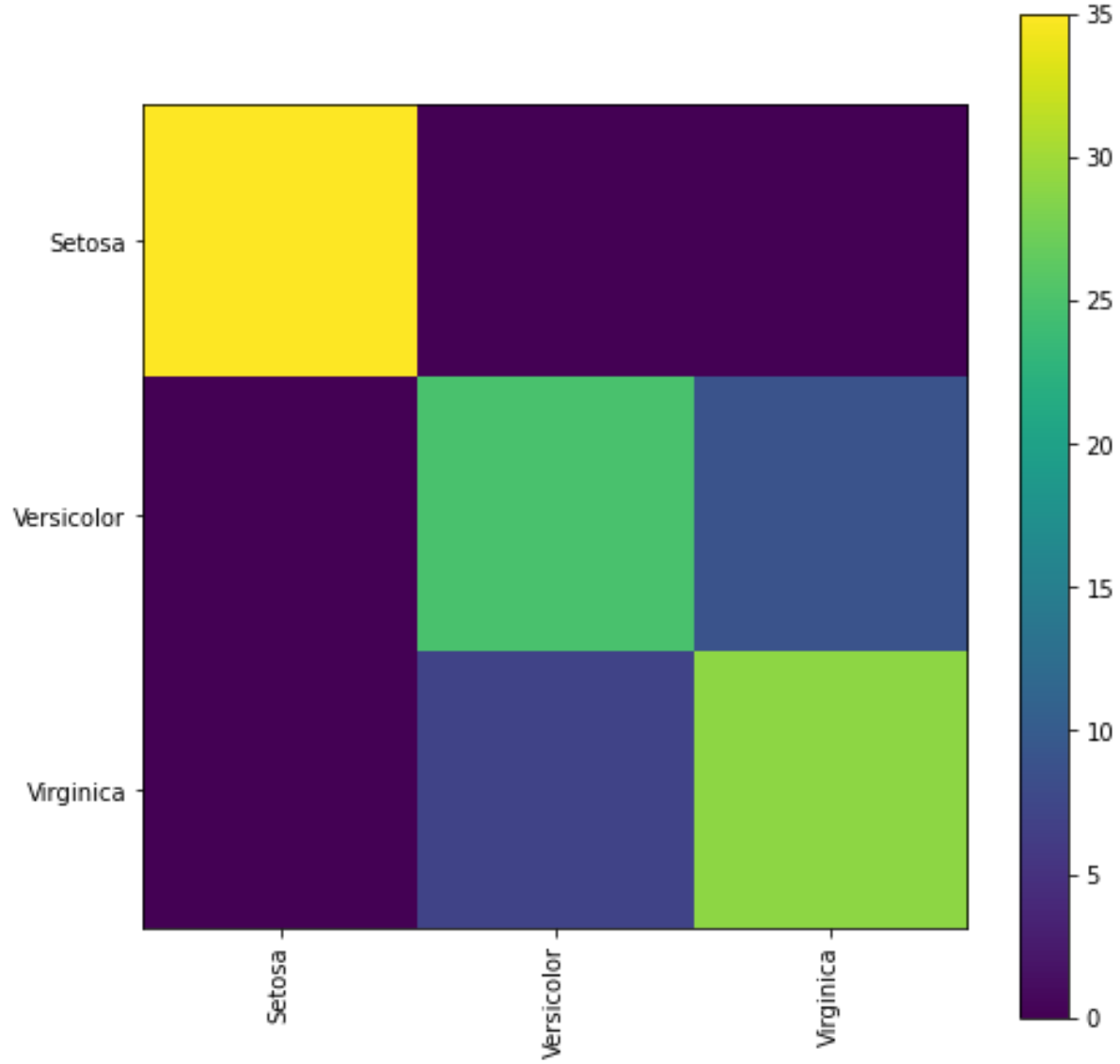
Test Data:

```
Confusion Matrix Trainset
[[14  1  0]
 [ 0 11  5]
 [ 0  6  8]]

Classification Report Trainset
              precision    recall  f1-score   support

      Setosa       1.00      0.93      0.97        15
  Versicolor       0.61      0.69      0.65        16
   Virginica       0.62      0.57      0.59        14

    accuracy                           0.73        45
   macro avg       0.74      0.73      0.74        45
weighted avg       0.74      0.73      0.74        45
```



در هر دو داده آموزش و تست شبکه به خوبی توانسته Setosa را تشخیص دهد. اما همانطور که در Confusion Matrix ها مشاهده می‌کنیم شبکه در تشخیص دو گروه Virginica و Versicolor خوب عمل نکرده و دچار Confusion شده. دلیل این اتفاق را می‌توان در جمله زیر دید:

One class is linearly separable from the other 2; the latter are NOT linearly separable from each other.

در واقع شبکه در تشخیص کلاس‌هایی که تفکیک‌پذیر خطی نبوده‌اند دچار مشکل شده است.