Binary Classification based on Logistic Regression

September 20, 2019

1 Binary Classification based on Logistic Regression

20166450 YoungminKim

```
In [4]: import matplotlib.pyplot as plt
        from random import *
        import numpy as np
```

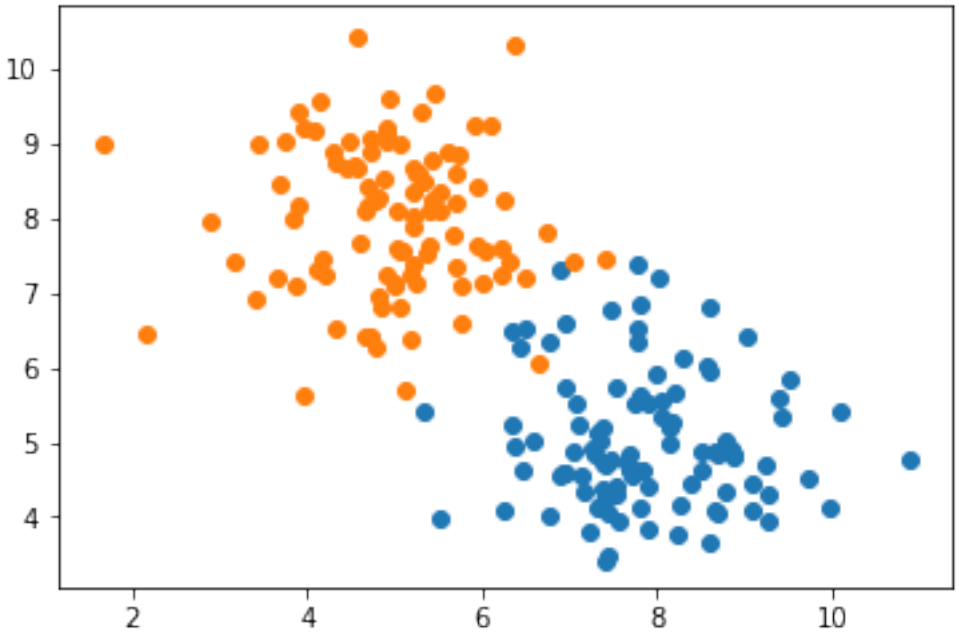
1.1 1. Plot two clusters of points for training dateset

- Generate two sets of separable random point clusters in R^2
- Let ${ x_i }_{i=1}^n $ be a set of points and $\{y_i\}_{i=1}^n$ be their corresponding labels Plot the point clusters in the training dataset using different colors depending on their labels

```
In [9]: s1=8
        t1=5
        s2=5
        t2=8
```

```
In [10]: x1_train=np.random.randn(100)
         y1_train=np.random.randn(100)
         x2_train=np.random.randn(100)
         y2_train=np.random.randn(100)
```

```
In [11]: xy1_train=np.column_stack([s1-x1_train,t1-y1_train])
         xy2_train=np.column_stack([s2-x2_train,t2-y2_train])
         plt.scatter(s1-x1_train,t1-y1_train)
         plt.scatter(s2-x2_train,t2-y2_train)
         plt.show()
```



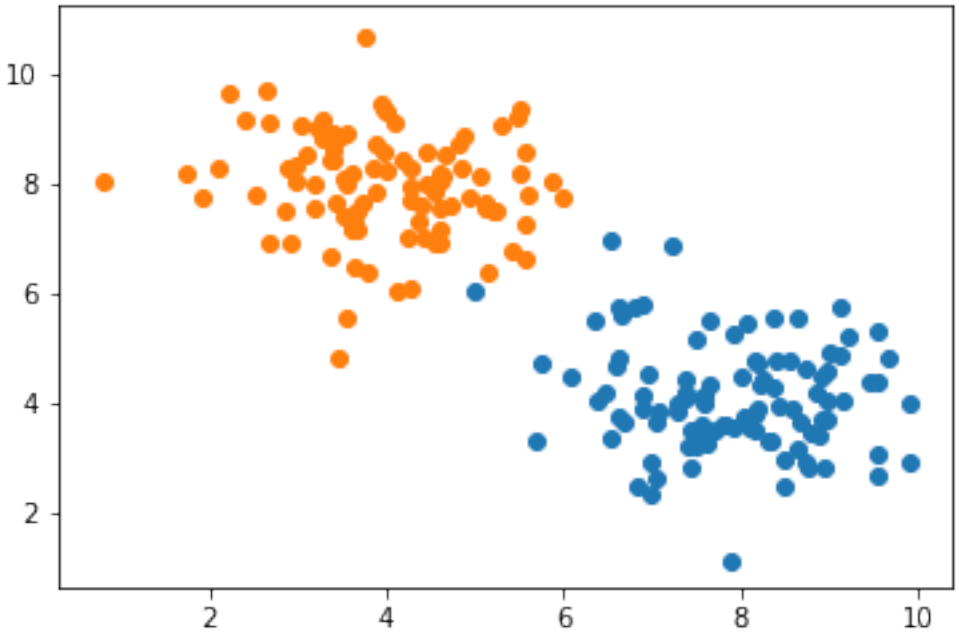
1.2 2. Plot two clusters of points for testing dataset

- Generate two sets of separable random point clusters in R^2 for a testing dataset using the same centroid and the standard deviation of random generator as the training dataset
- Plot the point clusters in the testing dataset using different colors depending on their labels (different colors from the training dataset)

```
In [20]: s1_test=8
         t1_test=4
         s2_test=4
         t2_test=8

In [21]: x1_test=np.random.randn(100)
         y1_test=np.random.randn(100)
         x2_test=np.random.randn(100)
         y2_test=np.random.randn(100)

In [22]: xy1_test=np.column_stack([s1_test-x1_test,t1_test-y1_test])
         xy2_test=np.column_stack([s2_test-x2_test,t2_test-y2_test])
         plt.scatter(s1_test-x1_test,t1_test-y1_test)
         plt.scatter(s2_test-x2_test,t2_test-y2_test)
         plt.show()
```



1.3 3. Plot the learning curves

- Apply the gradient descent algorithm
- Plot the training loss at every iteration
- Plot the testing loss at every iteration
- Plot the training accuracy at every iteration
- Plot the testing accuracy at every iteration

In [ ]: