logistic-regression

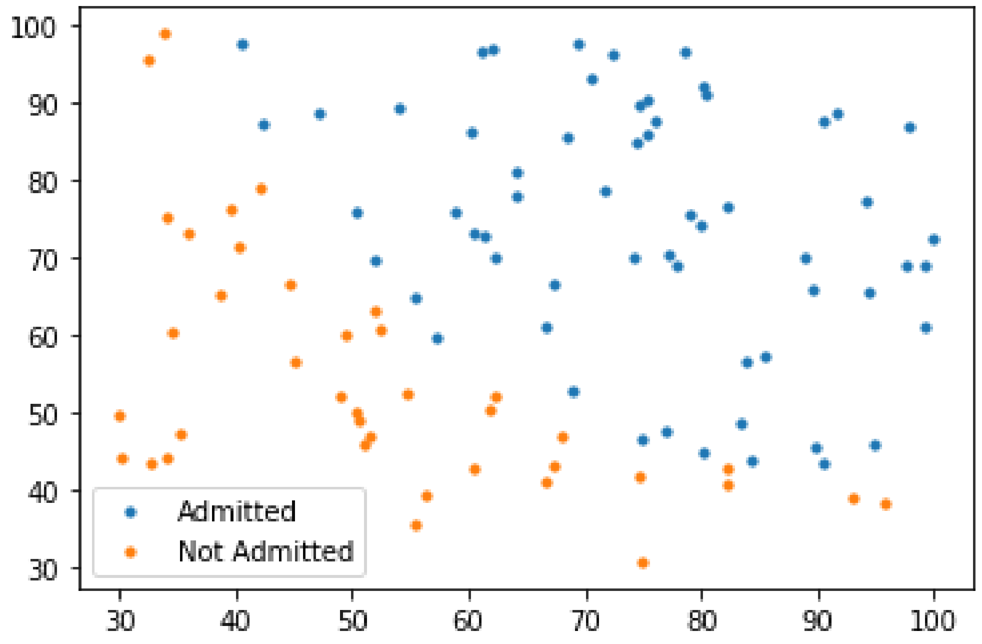
September 29, 2021

[1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import os

from scipy.optimize import fmin_tnc
from numpy import log
```

[2]:
```python
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def prob(theta, x):
    return sigmoid(np.dot(x, theta))

def objective(theta, x, y):
    # Computes the (negative of the) objective function, for all the training␣
 ↪samples
    p = prob(theta, x)
    return -np.sum(y * log(p) + (1 - y) * log(1 - p))

def gradient(theta, x, y):
    # Computes the gradient of the cost function at the point theta
    return np.dot(x.T, sigmoid(np.dot(x, theta)) - y)

def fit(x, y, theta):
    return fmin_tnc(func=objective, x0=theta, fprime=gradient, args=(x, y))[0]
```

[3]:
```python
data = pd.read_csv("".join([os.environ['HOME'], "/marks.txt"]))

 # X = feature values, all the columns except the last column
X = data.iloc[:, :-1]
X = np.c_[np.ones((X.shape[0], 1)), X] ## augment with column of ones

# y = target values, last column of the data frame
y = data.iloc[:, -1].to_numpy()

# select the applicants that got admitted vs. not admitted
```

```
admitted = data.loc[y == 1]
not_admitted = data.loc[y == 0]

plt.scatter(admitted.iloc[:, 0], admitted.iloc[:, 1], s=10, label='Admitted')
plt.scatter(not_admitted.iloc[:, 0], not_admitted.iloc[:, 1], s=10, label='Not␣
 ↪Admitted')
plt.legend()
plt.show()
```



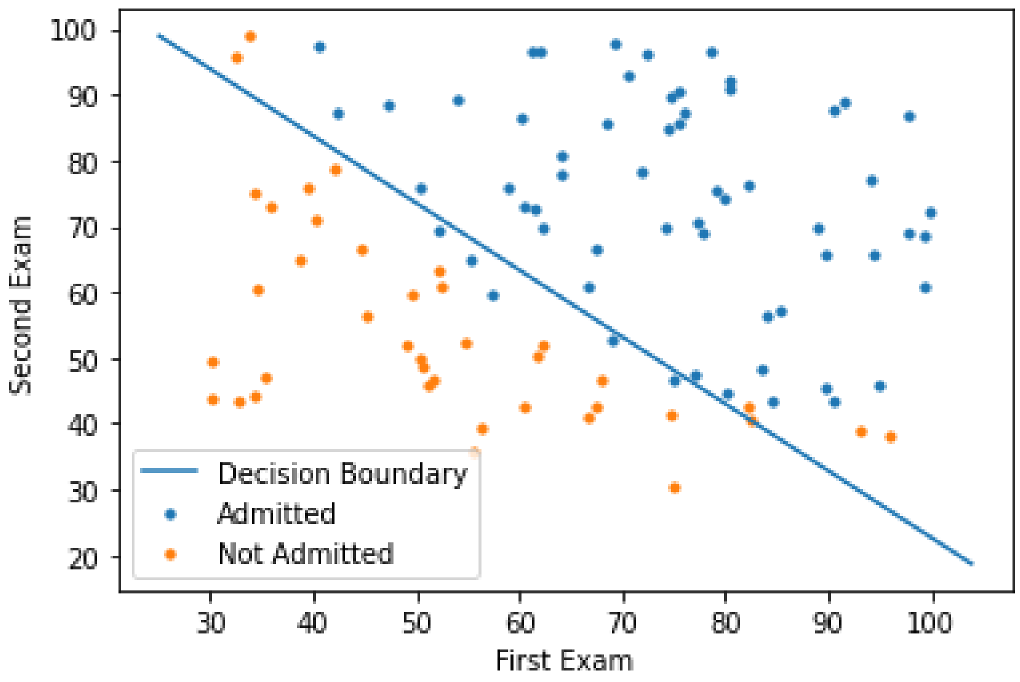
[4]:
```
theta_star = fit(X, y, np.zeros((X.shape[1], 1)))
print(theta_star)
```

```
[-24.86874868   0.2033721    0.19987258]
```

[5]:
```
def accuracy(x, actual_classes):
    predicted_classes = (prob(theta_star, x) >= 0.5).astype(int).flatten()
    return 100 * np.mean(predicted_classes == actual_classes)

print(accuracy(X, y))
```

```
88.88888888888889
```

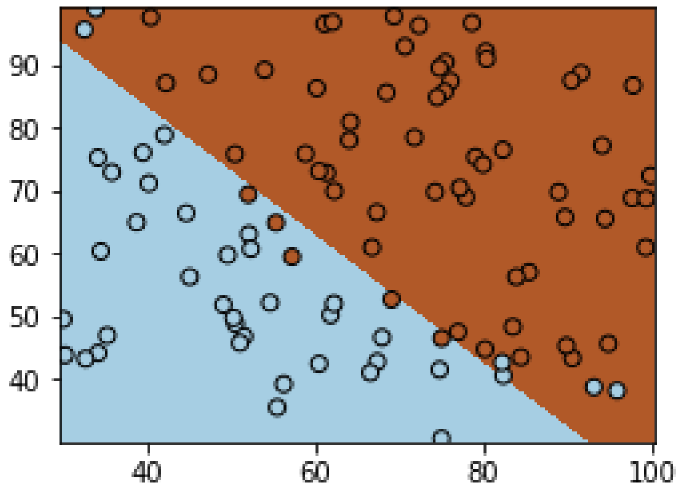
[6]:
```
def plot_decision_boundary(x, par):
    x_values = [np.min(x[:, 1] - 5), np.max(x[:, 2] + 5)]
    y_values = - (par[0] + np.dot(par[1], x_values)) / par[2]
    plt.plot(x_values, y_values, label='Decision Boundary')
    plt.scatter(admitted.iloc[:, 0], admitted.iloc[:, 1], s=10,␣
 ↪label='Admitted')
    plt.scatter(not_admitted.iloc[:, 0], not_admitted.iloc[:, 1], s=10,␣
 ↪label='Not Admitted')
    plt.xlabel('First Exam')
    plt.ylabel('Second Exam')
    plt.legend()
    plt.show()

plot_decision_boundary(X, theta_star)
```



[7]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression(penalty='none',solver='lbfgs')

X = data.iloc[:, :-1].to_numpy()

model.fit(X, y)
print(100 * accuracy_score(y, model.predict(X)))


# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max], x[y_min, y_max].
```

```
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
h = .02  # step size in the mesh
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1, figsize=(4, 3))
plt.pcolormesh(xx, yy, Z, cmap=plt.cm.Paired)

# Plot also the training points
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', cmap=plt.cm.Paired)

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

plt.show()
```

```
88.88888888888889
```



90
80
70
60
50
40
40
60
80
100