logistic_regression

March 25, 2019

Gahan Saraiya (18MCEC10)

AIM: Implementation of Logistic Regression

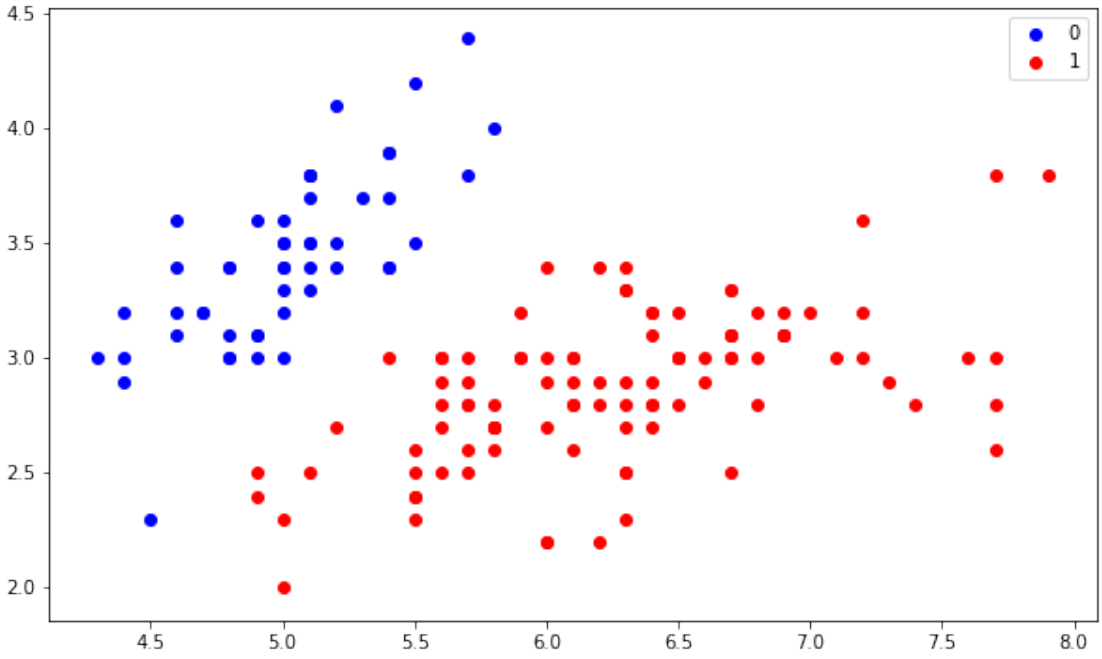
Logistic Regression

In [1]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
```

In [2]:
```
iris = datasets.load_iris()
```

In [3]:
```
X = iris.data[:, :2]
y = (iris.target != 0) * 1
```

In [4]:
```
plt.figure(figsize=(10, 6))
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='b', label='0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='r', label='1')
plt.legend();
```



```
In [5]: class LogisticRegression:
            def __init__(self, lr=0.01, num_iter=100000, fit_intercept=True, verbose=False):
                self.lr = lr
                self.num_iter = num_iter
                self.fit_intercept = fit_intercept
                self.verbose = verbose
        #         self.theta = None

            def __add_intercept(self, X):
                intercept = np.ones((X.shape[0], 1))
                return np.concatenate((intercept, X), axis=1)

            def __sigmoid(self, z):
                return 1 / (1 + np.exp(0-z))

            def __loss(self, h, y):
                return (-y * np.log(h) - (1 - y) * np.log(1 - h)).mean()

            def fit(self, X, y):
                if self.fit_intercept:
                    X = self.__add_intercept(X)

                # weights initialization
                self.theta = np.zeros(X.shape[1])

                for i in range(self.num_iter):
                    z = np.dot(X, self.theta)
                    h = self.__sigmoid(z)
                    gradient = np.dot(X.T, (h - y)) / y.size
                    self.theta -= self.lr * gradient

                    z = np.dot(X, self.theta)
                    h = self.__sigmoid(z)
                    loss = self.__loss(h, y)

                    if(self.verbose ==True and i % 10000 == 0):
                        print('loss: {} \t'.format(loss))

            def predict_prob(self, X):
                if self.fit_intercept:
                    X = self.__add_intercept(X)

                return self.__sigmoid(np.dot(X, self.theta))

            def predict(self, X):
```

```
                return self.predict_prob(X).round()
```

```
In [6]: model = LogisticRegression(lr=0.1, num_iter=3000)
```

```
In [7]: %time model.fit(X, y)
```

```
CPU times: user 149 ms, sys: 185 µs, total: 149 ms
Wall time: 152 ms
```

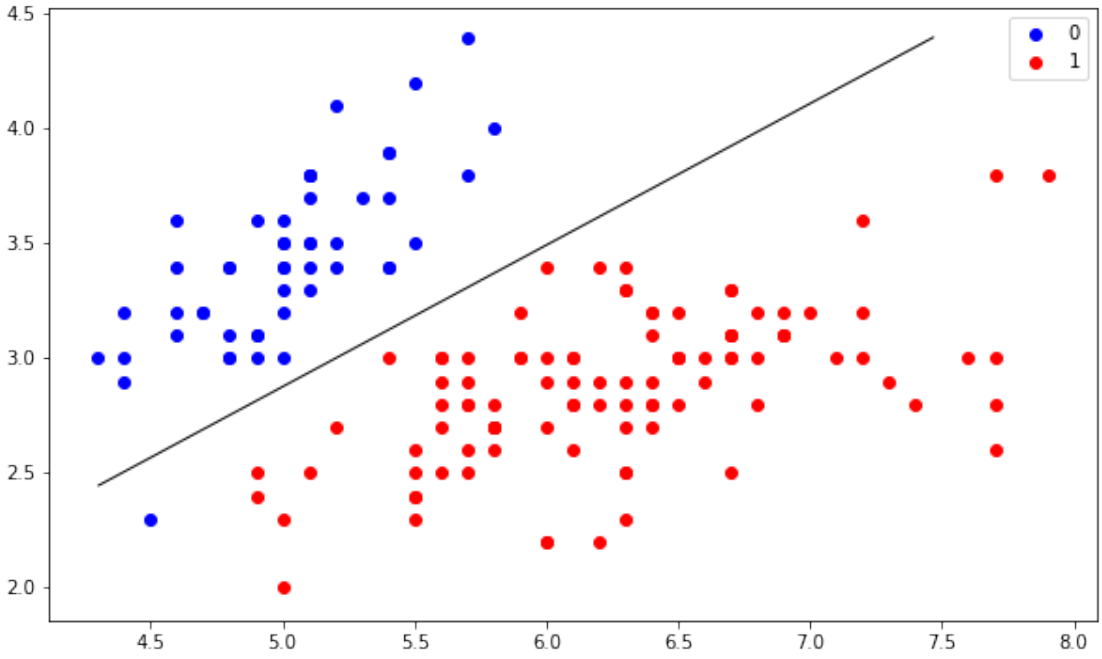
```
In [8]: preds = model.predict(X)
        (preds == y).mean()
```

```
Out[8]: 0.9933333333333333
```

```
In [9]: model.theta
```

```
Out[9]: array([-1.44894305,  4.25546329, -6.89489245])
```

```
In [10]: plt.figure(figsize=(10, 6))
         plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='b', label='0')
         plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='r', label='1')
         plt.legend()
         x1_min, x1_max = X[:,0].min(), X[:,0].max(),
         x2_min, x2_max = X[:,1].min(), X[:,1].max(),
         xx1, xx2 = np.meshgrid(np.linspace(x1_min, x1_max), np.linspace(x2_min, x2_max))
         grid = np.c_[xx1.ravel(), xx2.ravel()]
         probs = model.predict_prob(grid).reshape(xx1.shape)
         plt.contour(xx1, xx2, probs, [0.5], linewidths=1, colors='black');
```



```
In [16]: from sklearn.linear_model import LogisticRegression
         model = LogisticRegression(C=1e20)
         %time model.fit(X, y)

CPU times: user 2.03 ms, sys: 884 µs, total: 2.92 ms
Wall time: 2.46 ms


/usr/local/lib/python3.5/dist-packages/sklearn/linear_model/logistic.py:433: FutureWarning: De
  FutureWarning)


Out[16]: LogisticRegression(C=1e+20, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, max_iter=100, multi_class='warn',
                   n_jobs=None, penalty='l2', random_state=None, solver='warn',
                   tol=0.0001, verbose=0, warm_start=False)

In [17]: preds = model.predict(X)
         (preds == y).mean()

Out[17]: 1.0

In [18]: model.intercept_, model.coef_

Out[18]: (array([-80.54201957]), array([[ 31.5951929 , -28.30153825]]))
```