H4-LogisticModel

April 2, 2021

1 Implement SGD for binary classification by using logistic model

- Your Name:
- Your ID:

2 import necessary modules

- numpy for matrix calculation, `pip install numpy`
- matplotlib for plot figures, `pip install matplotlib`

[ ]:
```
import numpy as np
from matplotlib import pyplot as plt
```

3 functions for plot

[ ]:
```
from matplotlib import pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D


def plot_fig(w, x, y):
    w = w.tolist()[0]
    fig = plt.figure()
    ax = Axes3D(fig, azim=-80)

    x1 = np.linspace(0,1,30)
    x2 = np.linspace(0,1,30)
    x1, x2 = np.meshgrid(x1, x2)
    a = sigmoid(w[0]*x1 + w[1]*x2 + w[2])

    ax.plot_surface(x1, x2, a, rstride=1, cstride=1, cmap='rainbow')
    ax.scatter(x[0,:],x[1,:],y,)

    ax.set_xlabel('x1')
    ax.set_ylabel('x2')
    ax.set_zlabel('y')
```

```
    plt.title("model")
    #plt.savefig("data-scatter.jpg")

def plot_decision_line(w, x, y):
    print(y)
    fig = plt.figure()
    w = w.tolist()[0]
    xx = np.linspace(0,1,30)
    yy = -(w[0]*xx + w[-1])/w[1]
    plt.plot(xx, yy,label="decision line")
    pos_x = np.array([x[:2,i] for i in range(len(y[0])) if y[0][i] == 1]).
 ↪transpose()
    neg_x = np.array([x[:2,i] for i in range(len(y[0])) if y[0][i] == 0]).
 ↪transpose()
    plt.plot(pos_x[0], pos_x[1], 'rd', label="pos")
    plt.plot(neg_x[0], neg_x[1], 'b*', label="neg")
    plt.legend()
    plt.title("decision line")
```

4 prepare dataset D

1. load data `x` and `y` from the file `x.npy` and `y.npy`, respectively
2. adding bias term `1` to `x`

[ ]:
```
x = np.load("x4.npy")
print("original shape of x:{}".format(x.shape))
x = np.concatenate([x, np.ones((1,x.shape[1]))])
print("new shape of x:{}".format(x.shape))
y = np.load("y4.npy")
```

5 implement the function below for logistic model

$a = \sigma(wx)$, where

- $x \in R^{3\times n}$
- $w \in R^{1\times 3}$
- $\sigma(z) = \frac{1}{1+e^{-z}}$

[ ]:
```
def sigmoid(z):
    #### Your code here ####
    a = z
    ########################
    return a
def logisticModel(w, x):
    #### Your code here ####
    a = np.ones_like(x)
    ########################
```

```python
    return a
```

6 define cost function and accuracy

$J = \frac{1}{m} \sum_{i=1}^{m} \left[y^i \cdot \log(a^i) + (1 - y^i) \cdot \log(1 - a^i) \right]$

$\frac{\partial J}{\partial w} = (a - y)x^T$

[ ]:
```python
def cost(a, y):
    #### Your code here ####
    J = 0.5
    ########################
    return J

def grad(a, x, y):
    #### Your code here ####
    grad_w = np.random.rand(1,3)
    ########################
    return grad_w

def accuarcy(a, y, tao=0.5):
    #### Your code here ####
    acc = 0.5
    ########################
    return acc
```

7 gradient descent algorithm

[ ]:
```python
w = np.random.rand(1,3)

alpha = 0.01
Js = []
accs = []

for i in range(2000):
    a = logisticModel(w, x)
    grad_w = grad(a, x, y)
    w = w - alpha * grad_w
    Js.append(cost(a, y))
    accs.append(accuarcy(a, y))

plt.plot(Js,'r-*',label="cost")
plt.plot(accs,'b-',label="accuracy")
plt.legend()
plt.savefig("cost.jpg")
print("-"*40)
```

```python
print("gradient descent method: w={}".format(w))
print("-"*40)

plot_fig(w, x, y)

plot_decision_line(w, x, y)
```

[ ]:

```python

```