EE559 Homework 3 (week 4)

Jingquan Yan

USC ID: 1071912676

Email: jingquan@usc.edu

EE559 repository: Github

Modified plotDecBoundaries.py :

```
################################################
## EE559 HW Wk2, Prof. Jenkins, Spring 2018
## Created by Arindam Jati, TA
## Tested in Python 3.6.3, OSX El Captain
################################################

import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.distance import cdist


def plotDecBoundaries(training, label_train, mean):
    nclass = 2

    # Set the feature range for ploting

    xrange = (-3, 5)
    yrange = (-7, 3)

    # step size for how finely you want to visualize the decision boundary.
    inc = 0.01

    # generate grid coordinates. this will be the basis of the decision
    # boundary visualization.
    (x, y) = np.meshgrid(np.arange(xrange[0], xrange[1] + inc / 100, inc),
                         np.arange(yrange[0], yrange[1] + inc / 100, inc))

    # size of the (x, y) image, which will also be the size of the
    # decision boundary image that is used as the plot background.
    image_size = x.shape
    xy = np.hstack((x.reshape(x.shape[0] * x.shape[1], 1, order='F'),
                    y.reshape(y.shape[0] * y.shape[1], 1, order='F')))  # make
(x,y) pairs as a bunch of row vectors.
    #  meshgrid 一开始x和y是分开的，这里将x和y先reshape展平，然后压扁（每个x和y一组）
    # distance measure evaluations for each (x,y) pair.
    xy = xy.tolist()
    # xy = [1 if (mean[1]/mean[0])*m[0] < m[1] else 0 for m in xy]
    xy = [1 if ((-training[0][0] / training[0][1]) * m[0] > m[1])
              and (-training[1][0] / training[1][1]) * m[0] > m[1]
              and (-training[2][0] / training[2][1]) * m[0] > m[1]
              and (-training[3][0] / training[3][1]) * m[0] < m[1]
```

```
            else 0 for m in xy]
    # xy = [1 if -(training[0][0]/training[0][1])*m[0] < m[1] else 0 for m in
xy]
    pred_label = np.array(xy)
    # reshape the idx (which contains the class label) into an image.
    decisionmap = pred_label.reshape(image_size, order='F')

    # show the image, give each coordinate a color according to its class label
    plt.imshow(decisionmap, extent=[xrange[0], xrange[1], yrange[0], yrange[1]],
origin='lower',
               alpha=0.5)  # 将[0,0]放在左上角还是左下角

    # plot the class training data.
    plt.plot(training[label_train == 1, 0], training[label_train == 1, 1], 'rx')
    plt.plot(training[label_train == 2, 0], training[label_train == 2, 1], 'go')

    l = plt.legend(('Class 1', 'Class 2'), loc=2)

    x_bar = plt.gca()
    x_bar.add_artist(l)
    x_bar.spines['right'].set_color('none')
    x_bar.spines['top'].set_color('none')
    x_bar.spines['bottom'].set_position(('data', 0))
    x_bar.spines['left'].set_position(('data', 0))

    plt.grid(True, color='black', linewidth=0.6)

    plt.annotate(s='', xy=training[0], xytext=(0, 0),
arrowprops=dict(facecolor='black', width=2, headwidth=8))

    plt.show()
```

Problem 1:

```
from plotDecBoundaries import plotDecBoundaries
import numpy as np

train_data = [[1, -3], [1, -5], [1, 1], [1, -1]]
train_label = [1, 1, 2, 2]

mean = [1, 0.25 * sum(x[1] for x in train_data)]

plotDecBoundaries(np.array(train_data), np.array(train_label), mean)

reflected_data = [[-x for x in train_data[i]] if train_label[i] != 1 else
train_data[i] for i in range(len(train_data))]

plotDecBoundaries(np.array(reflected_data), np.array(train_label), mean)
```