DS-GA 1003: Homework 5
Trees and Boosting

Due on Monday, April 4, 2016

Professor David Ronsenberg

See complete code at: *git@github.com:cryanzpj/1003.git*

Yuhao Zhao
Yz3085

2 Decision Trees

• 2.1 Trees on the Banana Dataset.

– 2.1.1.

```
import numpy as np
import scipy as sp
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier

file_train = open('data/banana_train.csv')
file_test = open('data/banana_test.csv')

train = np.array(map(lambda x: x[:2] + [x[-1].strip()],
                         [i.split(',') for i in file_train]), dtype='float')
test = np.array(map(lambda x: x[:2] + [x[-1].strip()],
                         [i.split(',') for i in file_test]), dtype='float')

y_train = np.array([0 if i == -1 else 1 for i in train[:, 0]])
y_test = np.array([0 if i == -1 else 1 for i in test[:, 0]])
X_train = train[:, 1:]
X_test = test[:, 1:]
```

– 2.1.2.

```
n_classes = 2
plot_colors = "bry"
plot_step = 0.02

error = np.zeros((2, 10))

for i in xrange(1, 11):
        idx = np.arange(X_train.shape[0])
        np.random.seed(1)
        np.random.shuffle(idx)
        X = X_train[idx]
        y = y_train[idx]

        mean = X.mean(axis=0)
        std = X.std(axis=0)
        X = (X - mean) / std

        clf = DecisionTreeClassifier(max_depth=i).fit(X, y)
        plt.subplot(2, 5, i)
        x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
        y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
        xx, yy = np.meshgrid(np.arange(x_min, x_max, plot_step),
        np.arange(y_min, y_max, plot_step))

        Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
        Z = Z.reshape(xx.shape)

        training_error = np.sum(np.equal(clf.predict(X_train),
                                         1 - y_train)) / float(y_train.shape[0])
        testing_error = np.sum(np.equal(clf.predict(X_test),
                                                1 - y_test)) / float(y_test.shape[0])
        error[:, i - 1] = np.array([training_error, testing_error])

        cs = plt.contourf(xx, yy, Z, cmap=plt.cm.Paired)

        plt.axis("tight")

        for i, color in zip(range(n_classes), plot_colors):
                idx = np.where(y == i)
                plt.scatter(X[idx, 0], X[idx, 1], c=color, label=str(i),
                            cmap=plt.cm.Paired)

        plt.axis("tight")
        plt.legend(fontsize=10)
plt.suptitle('Decision surface for different depth')
plt.show()
```