CS 4340 - Logistic Regression

Austin Hester

November 5, 2017

Introduction

⇒ The cell radius is 1 km.

The Code

```
# Austin Hester
# 09/13/2017
# PLA Python Implementation
# Trains with 50 linearly seperable points
# Tests against 30
import numpy as np
import random
import matplotlib.pyplot as plt

class Perceptron:
    def __init__(self, N):
        x1, y1, x2, y2 = [random.uniform(-1, 1) for i in range(4)]
        # for generating linearly seperable data (V)
        self.V = np.array([x2*y1-x1*y2, y2-y1, x1-x2])
        self.X = self.generatePoints(N)
        self.iterations = 0

    def generatePoints(self, N):
        X = []
        for i in range(N):
            x1, x2 = [random.uniform(-1, 1) for i in range(2)]
            x_ = np.array([1, x1, x2])
            # classify based on V, our PLA does not know this line
            s = int(np.sign(self.V.T.dot(x_)))
            x_ = np.append(x_, [s])
            X.append(x_)
        return np.array(X)

    def plot(self, testPts=None, w_=None, save=False):
        fig = plt.figure(figsize=(6,6))
        plt.xlim(-1,1)
        plt.ylim(-1,1)
        plt.title('N␣=␣%s,␣Iteration␣%s\n' % (str(len(self.X)),str(self.iterations)))
        # draw line pla is searching for
        V = self.V
        a, b = -V[1]/V[2], -V[0]/V[2]
        l = np.linspace(-1,1)
        plt.plot(l, a*l+b, 'k-')
        ax = fig.add_subplot(1,1,1)
    ax.scatter(self.X[:,1:2], self.X[:,2:3], c=self.X
        [:,3:4], cmap='prism')
    if (w_ is not None and w_[2] != 0):
        # draw training line
        aa, bb = -w_[1]/w_[2], -w_[0]/w_[2]
        plt.plot(l, aa*l+bb, 'g-', lw=2)
    if (testPts is not None):
        # draw test points
        ax.scatter(testPts[:,1:2], testPts[:,2:3], c=
            testPts[:,3:4], cmap='cool')
    if save:
        plt.savefig('.\gifs\p_N%s' % (str(len(self.X)
            )), dpi=100, bbox_inches='tight')
    else:
        plt.show()

# returns percentage of missed points
def classifyError(self, w_, pts=None):
    if pts is None:
        pts = self.X[:,:3]
        S = self.X[:,3:4]
    else:
        S = pts[:,3:4]
        pts = pts[:,:3]
    M = len(pts)
    n_mispts = 0
    for x_, s in zip(pts, S):
        if int(np.sign(w_.T.dot(x_))) != s:
            n_mispts += 1
    print("Missed␣points:␣%d" % (n_mispts))
    print("w_:␣", w_)
    err = n_mispts / float(M)
    return err

# Pick a random misclassified pt (according to given
    w_)
def pickMisclPoint(self, w_):
    pts = self.X[:,:3]
    S = self.X[:,3:4]
    mispts = []
    for x,d in zip(pts, S):
        if int(np.sign(w_.T.dot(x))) != d:
            mispts.append((x, d))
    return mispts[random.randrange(0, len(mispts))]

# Run PLA on data contained in self.X
```

```
    def pla(self, c=0.01, save=False):
        X = self.X[:,:3]
        N = len(X)
        w_ = np.zeros(len(X[0]))
        it = 0
        print("Iteration %d: " % (it))
        # Run while there are misclassified points
        # or we get to 1,000 iterations
        while self.classifyError(w_) != 0 or it > 1000:
            it += 1
            print("Iteration %d: " % (it))
            # pick mispicked pt
            x, d = self.pickMisclPoint(w_)
            w_ += c*d*x
            if save:
                self.plot(vec=w_, save=True)
                plt.title('N = %s, Iteration %s\n' % (str
                    (N),str(it)))
                plt.savefig('.\gifs\p_N%s_it%s' % (str(N)
                    ,str(it)), dpi=100, bbox_inches='tight
                    ')
        self.w = w_
        self.iterations = it

    # Test our test points using classifyError
    def checkTestPoints(self, testPts, w_):
        print("------------------------------")
        print("Test Info")
        print("------------------------------")
        return self.classifyError(w_, pts=testPts)


def testPLA(p, M):
    testPts = p.generatePoints(M)
    print("------------------------------")
    print("Testing data:\n", testPts)
    print("------------------------------")
    testError = p.checkTestPoints(testPts, p.w)
    print("Test Error:", round(testError * 100, 3), "%")
    print("------------------------------")
    return testPts, testError

# initialize perceptron with 50 training points
train = 50
test = 30
save = False
```

```
p = Perceptron(train)
print("————————————————")
print("Training data:\n", p.X)
print("————————————————")
p.pla(save=save) # run pla, save=True to generate gif

testPts, testErr = testPLA(p, test)

if not save:
    p.plot(testPts=testPts, w_=p.w)
```