



#实现简单的神经网络

# coding=utf-8

import numpy as np

#分类器代码

class Perceptron(object):

    """

    eta:学习率

    n\_iter:权重向量的训练次数

    w\_:神经分叉权重向量

    errors\_:用于记录神经元判断出错次数

    """

    def \_\_init\_\_(self,eta = 0.01,n\_iter = 10):

        self.eta = eta;

        self.n\_iter = n\_iter

        pass

    def fit(self,X,y):

        """

        输入训练数据，培训神经元

        :param X: 输入样本向量

        :param y: 对应样本分类

        X:shape[n\_samples, n\_features]

        X:[[1,2,3],[4,5,6]]

        n\_samples :2

        n\_features:3

        y:[1,-1]

        """

        """

        初始化向量为0

        加一是因为步调函数阈值

        """

        self.w\_ =np.zeros(1+X.shape[1]);

        self.errors\_=[]

        for \_ in range(self.n\_iter):

            errors = 0

            for xi, target in zip(X,y):

                update = self.eta\*(target - self.predict(xi))

                self.w\_[1:]+= update \* xi

                self.w\_[0] += update;

                errors += int(update!= 0)

                self.errors\_.append(errors)

                pass

            pass

    def net\_input(self,X):

        return np.dot(X,self.w\_[1:]+self.w\_[0])

        pass

    def predict(self,X):

        return np.where(self.net\_input(X)>=0.0,1,-1)

        pass

 ----------------------------------------------------------------------------

 #数据获取

import pandas as pd

file = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

df = pd.read\_csv(file,header=None)

import matplotlib.pyplot as plt

y = df.loc[0:100,4].values

y=np.where(y=='Iris-setosa',-1,1)

#根据整数位置选取单列或单行数据

X = df.loc[0:100,[0,2]].values

plt.scatter(X[:50,0],X[:50,1],color='red',marker='o',label="setosa")

plt.scatter(X[50:100,0],X[50:100,1],color='blue',marker='x',label="versicolor")

plt.xlabel('huabanchangdu')

plt.ylabel('huajingchangdu')

plt.legend(loc='upper left')

 -----------------------------------------------------------------

 #训练函数的调用

ppn =Perceptron(eta=0.1,n\_iter=10)

ppn.fit(X,y)

from matplotlib.colors import ListedColormap

def plot\_decision\_region(X,y,classifier,resolution=0.02):

    markers=('s','x','o','v')

    colors=('red','blue','lightgreen','gray','cyan')

    cmap = ListedColormap(colors[:len(np.unique(y))])

    x1\_min ,x1\_max = X[:,0].min()-1,X[:,0].max()

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max()

#扩展矩阵

    xx1,xx2 = np.meshgrid(np.arange(x1\_min,x1\_max,resolution),

                      np.arange(x2\_min, x2\_max, resolution))

    Z =classifier.predict(np.array([xx1.ravel(),xx2.ravel()]).T)

    print (xx1.ravel())

    print(xx2.ravel())

    print Z

Z=Z.reshape(xx1.shape)

#画图

    plt.contourf(xx1,xx2,Z,alpha =0.4, cmap=cmap)

    plt.xlim(xx1.min(),xx1.max())

    plt.ylim(xx2.min(),xx2.max())

    for idx,cl in enumerate(np.unique(y)):

        plt.scatter(x=X[y==cl,0],y=X[y==cl,1],alpha=0.8,c=cmap(idx),

                    marker=markers[idx],label=cl)

plot\_decision\_region(X,y,ppn,resolution=0.02)

plt.xlabel('huajingchang')

plt.ylabel('huabanchang')

plt.legend(loc='upper left')

plt.show()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# coding=utf-8

import numpy as np

import matplotlib.pyplot as plt

#ada代码

class AdalineGd(object):

'''

eta: float

学习效率，处于0和1之间

n\_iter：int

对训练数据进行学习，改进次数

w\_：一维向量

存储权重数值

error\_：

一维向量

存储每次迭代改进时，神经网络对数据进行错误判断的次数

'''

def \_\_init\_\_(self,eta = 0.01,n\_iter=50):

self.eta=eta

self.n\_iter =n\_iter

def fit(self,X,y):

'''

:param X: 二维数组[n\_samples, n\_features]

n\_samples 表示X中含有训练数据条目数

n\_features含有4个数据的一维向量，用于表示一条训练条目

:param y: 一维向量

用于存储每一训练条目对应的正确分类

:return:

'''

self.w\_ =np.zeros(1+X.shape[1])#权重初始化为零

self.cost\_ =[]

for i in range(self.n\_iter):

output = self.net\_input(X)

errors = (y - output)#向量

self.w\_[1:]+=self.eta \*X.T.dot(errors)

self.w\_[0] +=self.eta \*errors.sum()

cost =(errors \*\*2).sum()/2

self.cost\_.append(cost)

return self

def net\_input(self,X):

return np.dot(X,self.w\_[1:]+self.w\_[0])

def activation(self,X):

return self.net\_input(X)

def predict(self,X):

return np.where(self.activation(X)>=0,1,-1)

import pandas as pd

file = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'

df = pd.read\_csv(file,header=None)

y = df.loc[0:100,4].values

y=np.where(y=='Iris-setosa',-1,1)

#根据整数位置选取单列或单行数据

X = df.loc[0:100,[0,2]].values

from matplotlib.colors import ListedColormap

def plot\_decision\_region(X,y,classifier,resolution=0.02):

markers=('s','x','o','v')

colors=('red','blue','lightgreen','gray','cyan')

cmap = ListedColormap(colors[:len(np.unique(y))])

x1\_min ,x1\_max = X[:,0].min()-1,X[:,0].max()

x2\_min, x2\_max = X[:, 1].min() - 1, X[:, 1].max()

xx1,xx2 = np.meshgrid(np.arange(x1\_min,x1\_max,resolution),

np.arange(x2\_min, x2\_max, resolution))

Z =classifier.predict(np.array([xx1.ravel(),xx2.ravel()]).T)

print (xx1.ravel())

print(xx2.ravel())

Z=Z.reshape(xx1.shape)

plt.contourf(xx1,xx2,Z,alpha =0.4, cmap=cmap)

plt.xlim(xx1.min(),xx1.max())

plt.ylim(xx2.min(),xx2.max())

for idx,cl in enumerate(np.unique(y)):

plt.scatter(x=X[y==cl,0],y=X[y==cl,1],alpha=0.8,c=cmap(idx),

marker=markers[idx],label=cl)

ada = AdalineGd(eta=0.0001,n\_iter=100)

ada.fit(X,y)

plot\_decision\_region(X,y,classifier=ada)

plt.xlabel('huajingchang')

plt.ylabel('huabanchang')

plt.legend(loc='upper left')

plt.show()

plt.plot(range(1,len(ada.cost\_)+1),ada.cost\_,marker ='o')

plt.xlabel('epochs')

plt.ylabel('sum-squard-error')

plt.show()