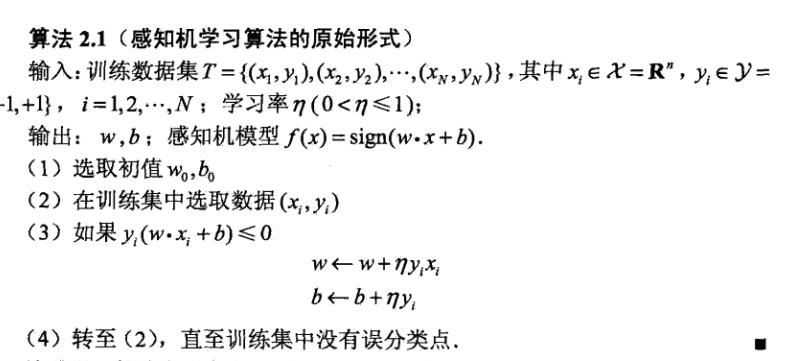
【感知机模型 】



【HOG特征提取】

Python提取特征需要调用opencv2，代码如下所示

hog = cv2.HOGDescsriptor(‘../hog.xml’)

img = np.reshape(img,(28,28))

cv\_img = img.astype(np.uint8)

hog\_fearture = hog.compute(cv\_img)

【代码】

1）sklearn.model\_selection.train\_test\_split随机划分训练集和测试集

* 一般形式

train\_test\_split是交叉验证中常用的函数，功能是从样本中随机的按比例选取train data和testdata，形式为：

X\_train,X\_test, y\_train, y\_test =

cross\_validation.train\_test\_split(**train\_data**,**train\_target**,**test\_size**=0.4, **random\_state**=0)

* 参数解释

**train\_data**：所要划分的样本特征集

**train\_target**：所要划分的样本结果

**test\_size**：样本占比，如果是整数的话就是样本的数量

**random\_state**：是随机数的种子。

随机数种子：其实就是该组随机数的编号，在需要重复试验的时候，保证得到一组一样的随机数。比如你每次都填1，其他参数一样的情况下你得到的随机数组是一样的。但填0或不填，每次都会不一样。

随机数的产生取决于种子，随机数和种子之间的关系遵从以下两个规则：种子不同，产生不同的随机数；种子相同，即使实例不同也产生相同的随机数。

2）

>>> from sklearn.metrics import accuracy\_score

accuracy\_score(y\_true, y\_pred[, normalize, …])

http://scikit-learn.org/stable/_images/math/cd4bea15b385d15cceb8e24f68976da7d8510290.png

# normalize=False 则返回总数

【感知机代码】

import pandas as pd

import numpy as np

import cv2

import random

import time

from sklearn.cross\_validation import train\_test\_split

from sklearn.metrics import accuracy\_score

# 利用opencv获取图像hog特征

def get\_hog\_features(trainset):

features = []

hog = cv2.HOGDescriptor('../hog.xml')

for img in trainset:

img = np.reshape(img,(28,28))

cv\_img = img.astype(np.uint8)

hog\_feature = hog.compute(cv\_img)

# hog\_feature = np.transpose(hog\_feature)

features.append(hog\_feature)

features = np.array(features)

features = np.reshape(features,(-1,324))

return features

def Train(trainset,train\_labels):

# 获取参数

trainset\_size = len(train\_labels)

# 初始化 w,b

w = np.zeros((feature\_length,1))

b = 0

study\_count = 0 # 学习次数记录，只有当分类错误时才会增加

nochange\_count = 0 # 统计连续分类正确数，当分类错误时归为0

nochange\_upper\_limit = 100000 # 连续分类正确上界，当连续分类超过上界时，认为已训练好，退出训练

while True:

nochange\_count += 1

if nochange\_count > nochange\_upper\_limit:

break

# 随机选的数据

index = random.randint(0,trainset\_size-1)

img = trainset[index]

label = train\_labels[index]

# 计算yi(w\*xi+b)

yi = int(label != object\_num) \* 2 - 1 # 如果等于object\_num, yi=-1, 否则yi=1

result = yi \* (np.dot(img,w) + b)

# 如果yi(w\*xi+b) <= 0 则更新 w 与 b 的值

if result <= 0:

img = np.reshape(trainset[index],(feature\_length,1)) # 为了维数统一，需重新设定一下维度

w += img\*yi\*study\_step # 按算法步骤3更新参数

b += yi\*study\_step

study\_count += 1

if study\_count > study\_total:

break

nochange\_count = 0

return w,b

def Predict(testset,w,b ):

predict = []

for img in testset:

result = np.dot(img,w) + b

result = result > 0

predict.append(result)

return np.array(predict)

study\_step = 0.0001 # 学习步长

study\_total = 10000 # 学习次数

feature\_length = 324 # hog特征维度

object\_num = 0 # 分类的数字

if \_\_name\_\_ == '\_\_main\_\_':

print('Start read data')

time\_1 = time.time()

raw\_data = pd.read\_csv('../data/train\_binary.csv',header=0)

data = raw\_data.values

imgs = data[0::,1::]

labels = data[::,0]

features = get\_hog\_features(imgs)

# 选取 2/3 数据作为训练集， 1/3 数据作为测试集

train\_features, test\_features, train\_labels, test\_labels = train\_test\_split(features, labels, test\_size=0.33, random\_state=23323)

# print(train\_features.shape)

# print(test\_features.shape)

time\_2 = time.time()

print('read data cost ', '{} '.format(time\_2 - time\_1) , ' second','\n')

print('Start training')

**w,b = Train(train\_features,train\_labels)**

time\_3 = time.time()

print 'training cost ',time\_3 - time\_2,' second','\n'

print('Start predicting ')

**test\_predict = Predict(test\_features,w,b)**

time\_4 = time.time()

print 'predicting cost ',time\_4 - time\_3,' second','\n'

**score = accuracy\_score(test\_labels,test\_predict)**

print "The accruacy socre is ", score