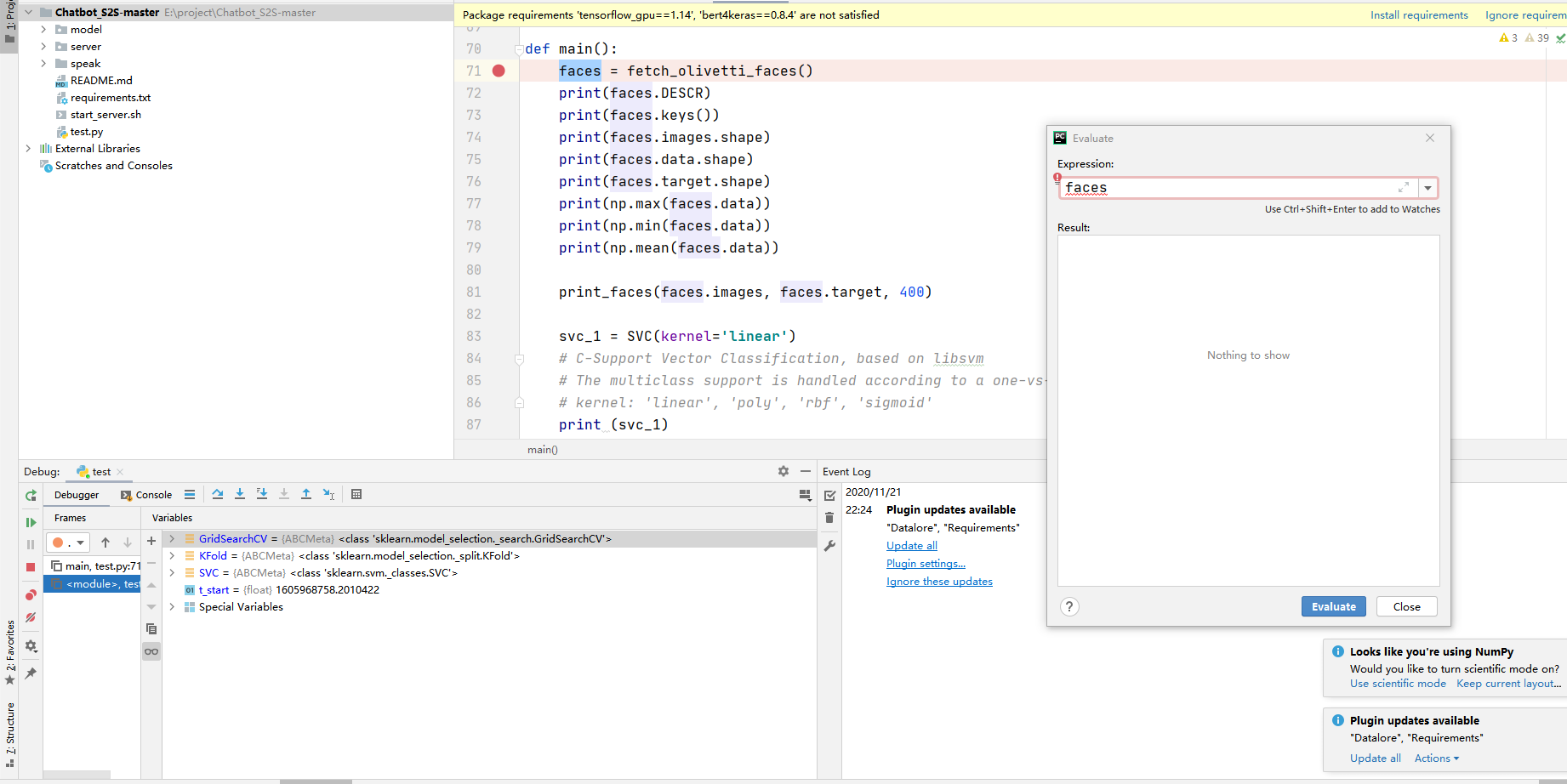
sklearn中的SVM的实现  


使用debug



## SVC的参数

SVC：多分类器

C： C-SVC的惩罚参数C，默认值是1.0

kernel ：核函数，默认是rbf，可为如下：

– 'linear'，线性：u'v

– 'poly'，多项式：(gamma\*u'\*v + coef0)^degree

– 'rbf'，RBF函数：exp(-gamma|u-v|^2)

–'sigmoid'：tanh(gamma\*u'\*v + coef0)

degree ：多项式poly函数的维度，默认是3，其他核函数会被忽略

gamma ： ‘rbf’,‘poly’ 和‘sigmoid’的核函数参数。默认是’auto’，则会选择1/n\_features

coef0 ：核函数的常数项，对于‘poly’和 ‘sigmoid’有用

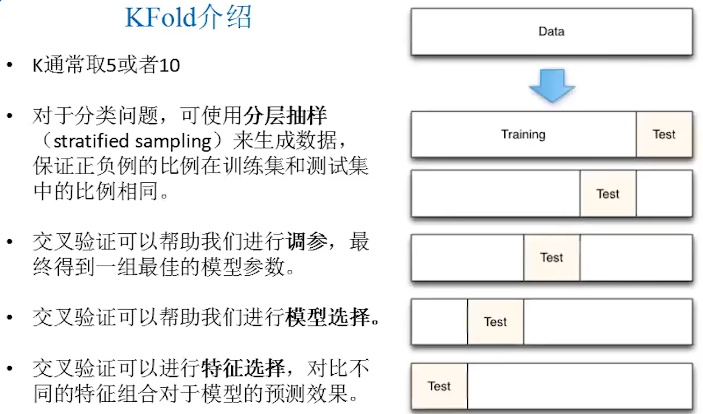
max\_iter ：最大迭代次数，-1为无限制

tol ：停止训练的误差值大小，默认为1e-3

decision\_function\_shape :’ovo‘, ‘ovr’ or None, default=‘ovr’(one vs rest)

random\_state ：数据洗牌时的种子值，int值

主要调节的参数有：C、kernel、degree、gamma、coef0。



|  |
| --- |
| import numpy as np  import time  from scipy.stats import sem  # from sklearn.cross\_validation import train\_test\_split  # cross\_validation has been deprecated in >=sklearn 0.18  from sklearn.model\_selection import train\_test\_split,cross\_val\_score  from sklearn.model\_selection import KFold,GridSearchCV  from sklearn.svm import SVC  from sklearn.metrics import classification\_report  from sklearn.metrics import confusion\_matrix  from sklearn.datasets import fetch\_olivetti\_faces  import ssl  ssl.\_create\_default\_https\_context = ssl.\_create\_unverified\_context  # fetch\_olivetti\_faces(): certificate verify failed, above line can avoid such err  import matplotlib  matplotlib.use('TkAgg') # 'TkAgg' can show GUI in imshow()  # matplotlib.use('Agg') # 'Agg' will not show GUI  import matplotlib.pyplot as plt  def evaluate\_cross\_validation(clf, X, y, K):  # create a k-fold cross validation iterator  cv = KFold(K, shuffle=True, random\_state=0)  # score method of the estimator (accuracy)  scores = cross\_val\_score(clf, X, y, cv=cv)  print (scores)  print ("Mean score: {0:.3f} (+/-{1:.3f})".format(  np.mean(scores), sem(scores)))  def train\_and\_evaluate(clf, X\_train, X\_test, y\_train, y\_test):  clf.fit(X\_train, y\_train)  print ("Accuracy on training set:")  print (clf.score(X\_train, y\_train))  print ("Accuracy on testing set:")  print (clf.score(X\_test, y\_test))  y\_pred = clf.predict(X\_test)  print ("Classification Report:")  print (classification\_report(y\_test, y\_pred))  print ("Confusion Matrix:")  print (confusion\_matrix(y\_test, y\_pred))  def create\_target(num\_sample,segments):  # create a new y array of target size initialized with zeros  y = np.zeros(num\_sample)  # put 1 in the specified segments  for (start, end) in segments:  y[start:end + 1] = 1  return y  def print\_faces(images, target, top\_n):  # set up figure size in inches  fig = plt.figure(figsize=(12, 12))  fig.subplots\_adjust(left=0, right=1, bottom=0, top=1, hspace=0.05, wspace=0.05)  for i in range(top\_n):  # we will print images in matrix 20x20  p = fig.add\_subplot(20, 20, i + 1, xticks=[], yticks=[])  p.imshow(images[i], cmap=plt.cm.bone)  # label the image with target value  p.text(0, 14, str(target[i]))  p.text(0, 60, str(i))  def main():  faces = fetch\_olivetti\_faces()  print(faces.DESCR)  print(faces.keys())  print(faces.images.shape)  print(faces.data.shape)  print(faces.target.shape)  print(np.max(faces.data))  print(np.min(faces.data))  print(np.mean(faces.data))  print\_faces(faces.images, faces.target, 400)  svc\_1 = SVC(kernel='linear')  # C-Support Vector Classification, based on libsvm  # The multiclass support is handled according to a one-vs-one scheme.  # kernel: 'linear', 'poly', 'rbf', 'sigmoid'  print (svc\_1)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  faces.data, faces.target, test\_size=0.25, random\_state=0)  evaluate\_cross\_validation(svc\_1, X\_train, y\_train, 5)  train\_and\_evaluate(svc\_1, X\_train, X\_test, y\_train, y\_test)  # the index ranges of images of people with glasses  glasses = [  (10, 19), (30, 32), (37, 38), (50, 59), (63, 64),  (69, 69), (120, 121), (124, 129), (130, 139), (160, 161),  (164, 169), (180, 182), (185, 185), (189, 189), (190, 192),  (194, 194), (196, 199), (260, 269), (270, 279), (300, 309),  (330, 339), (358, 359), (360, 369)  ]  num\_samples = faces.target.shape[0]  target\_glasses = create\_target(num\_samples, glasses)  svc\_2 = SVC(kernel='linear')  X\_train, X\_test, y\_train, y\_test = train\_test\_split(  faces.data, target\_glasses, test\_size=0.25, random\_state=0)  evaluate\_cross\_validation(svc\_2, X\_train, y\_train, 5)  train\_and\_evaluate(svc\_2, X\_train, X\_test, y\_train, y\_test)  X\_test = faces.data[30:40]  y\_test = target\_glasses[30:40]  print (y\_test.shape[0])  select = np.ones(target\_glasses.shape[0])  select[30:40] = 0  X\_train = faces.data[select == 1]  y\_train = target\_glasses[select == 1]  print (y\_train.shape[0])  svc\_3 = SVC(kernel='linear')  train\_and\_evaluate(svc\_3, X\_train, X\_test, y\_train, y\_test)  y\_pred = svc\_3.predict(X\_test)  eval\_faces = [np.reshape(a, (64, 64)) for a in X\_test]  print\_faces(eval\_faces, y\_pred, 10)  if \_\_name\_\_ == "\_\_main\_\_":  t\_start = time.time()  main()  t\_end = time.time()  t\_all = t\_end - t\_start  print('FR\_SVM.py: whole time: {:.2f} min'.format(t\_all / 60.)) |