随着计算机理论、技术和应用的快速发展,视频图像处理和计算能力得到了极大的提高,使得计算机视觉成为了计算机领域与人工智能领域中最热门的研究课题之一。计算机视觉（CV），通俗地说，是一门研究如何使[机器](http://baike.baidu.com/item/%E6%9C%BA%E5%99%A8)“看”的科学，更进一步的说，是指用摄影机和电脑代替人眼对目标进行识别、跟踪和测量等[机器视觉](http://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%A7%86%E8%A7%89)，并进一步做图形处理，使电脑处理成为更适合人眼观察或传送给仪器检测的图像。

本小组研究的课题是基于深度学习的图像识别，最终实现的是对海量图片数据的学习和准确的识别，不仅如此，我们测试了几种不同的分类模型，并比较预测结果，计算预测准确率，对预测方法进行优化，希望得到一种最高效的预测方法，从而实现真正的机器智能化识别。

本小组课设主要基于python开发环境下的scikit-learn标准库以及PIL图像处理库，并采用matplotlib实现最终结果的比对，PIL库用于图像的特征值批量读取，scikit-learn标准库用于分类模型的构建，matplotlib则用于显示最终结果。

这是这两个基本库的homepage以及参考网页：

PIL : <http://www.pythonware.com/products/pil/>

<http://blog.csdn.net/u013467442/article/details/41827085>

<http://blog.csdn.net/passball/article/details/5204132>

scikit-learn ： <http://scikit-learn.org/stable/index.html>

下面是本小组的方案流程以及说明：



**1图像特征向量的提取：**

特征提取是计算机视觉和图像处理中的一个概念。它指的是使用计算机提取图像信息，决定每个图像的点是否属于一个图像特征。特征提取的结果是把图像上的点分为不同的子集，这些子集往往属于孤立的点、连续的曲线或者连续的区域。

特征的精确定义往往由问题或者应用类型决定。特征是一个数字图像中“有趣”的部分，它是许多计算机图像分析算法的起点。因此一个算法是否成功往往由它使用和定义的特征决定。因此特征提取最重要的一个特性是“可重复性”：同一场景的不同图像所提取的特征应该是相同的。

常用的特征提取方法有方向梯度直方图（Histogram of Oriented Gradient, HOG）特征，主要思想是在一副图像中，局部目标的表象和形状（appearance and shape）能够被梯度或边缘的方向密度分布很好地描述；LBP（Local Binary Pattern，局部二值模式）特征，一种用来描述图像局部纹理特征的算子；Haar-like特征，最早由Papageorgiou等应用于人脸表示。

我们采用HOG方法进行图像特征的提取，特别的，HOG+SVM方法进行图像识别已经被广泛用于图像识别领域中。

通过调用PIL库中的Image模块可以实现上述功能。

img = Image.open(j)

j为图片所在路径，采用open方法描述图片。进行灰度化:

Gray = (R^2.2 \* 0.2973 + G^2.2 \* 0.6274 + B^2.2 \* 0.0753)^(1/2.2)

通过相关算法对图片的特征进行处理，最终可得到想要的特征向量。此外，需要保存每张图像对应的目标值以及图像的三原色信息。（详细的容器说明见代码）

**2 训练集、测试集的分离**

调用sklearn中的train\_test\_split方法：

self.x\_train0, self.x\_test0, self.y\_train0, self.y\_test0 = train\_test\_split(

self.pic\_data\_gray, self.target, test\_size=0.25, random\_state=42)

**3 图像主成分的析取及灰度化**

通过第一步图片的特征向量提取，我们得到一组特征向量的列表集，但由于特征向量的维度过高，一来提高了计算的复杂度，占用内存资源，不利于大规模开发，二来特征描述过分详细，相关性不大的维度会对分类器的分类造成负面影响，所以接下来对图片的特征向量进行降维处理。

这里采用PCA（Principal Component Analysis）降维方法，PCA又称主成分分析法，是一种常用的数据分析方法。PCA通过线性变换将原始数据变换为一组各维度线性无关的表示，可用于提取数据的主要特征分量，常用于高维数据的降维。

对于一个k维的特征来说，相当于它的每一维特征与其他维都是正交的（相当于在多维坐标系中，坐标轴都是垂直的），那么我们可以变化这些维的坐标系，从而使这个特征在某些维上方差大，而在某些维上方差很小。例如，一个45度倾斜的椭圆，在第一坐标系，如果按照x,y坐标来投影，这些点的x和y的属性很难用于区分他们，因为他们在x,y轴上坐标变化的方差都差不多，我们无法根据这个点的某个x属性来判断这个点是哪个，而如果将坐标轴旋转，以椭圆长轴为x轴，则椭圆在长轴上的分布比较长，方差大，而在短轴上的分布短，方差小，所以可以考虑只保留这些点的长轴属性，来区分椭圆上的点，这样，区分性比x,y轴的方法要好！

所以我们的做法就是求得一个k维特征的投影矩阵，这个投影矩阵可以将特征从高维降到低维。投影矩阵也可以叫做变换矩阵。新的低维特征必须每个维都正交，特征向量都是正交的。通过求样本矩阵的协方差矩阵，然后求出协方差矩阵的特征向量，这些特征向量就可以构成这个投影矩阵了。特征向量的选择取决于协方差矩阵的特征值的大小。

举一个例子：

对于一个训练集，100个对象模板，特征是10维，那么它可以建立一个100\*10的矩阵，作为样本。求这个样本的协方差矩阵，得到一个10\*10的协方差矩阵，然后求出这个协方差矩阵的特征值和特征向量，应该有10个特征值和特征向量，我们根据特征值的大小，取前四个特征值所对应的特征向量，构成一个10\*4的矩阵，这个矩阵就是我们要求的特征矩阵，100\*10的样本矩阵乘以这个10\*4的特征矩阵，就得到了一个100\*4的新的降维之后的样本矩阵，每个特征的维数下降了。

总结一下PCA的算法步骤：

设有m条n维数据。

1）将原始数据按列组成n行m列矩阵X

2）将X的每一行（代表一个属性字段）进行零均值化，即减去这一行的均值

3）求出协方差矩阵

4）求出协方差矩阵的特征值及对应的特征向量

5）将特征向量按对应特征值大小从上到下按行排列成矩阵，取前k行组成矩阵P

6）即为降维到k维后的数据

具体的原理：<http://blog.csdn.net/xiaojidan2011/article/details/11595869>

<http://www.cnblogs.com/549294286/archive/2013/11/11/3417702.html>

算法实现为：

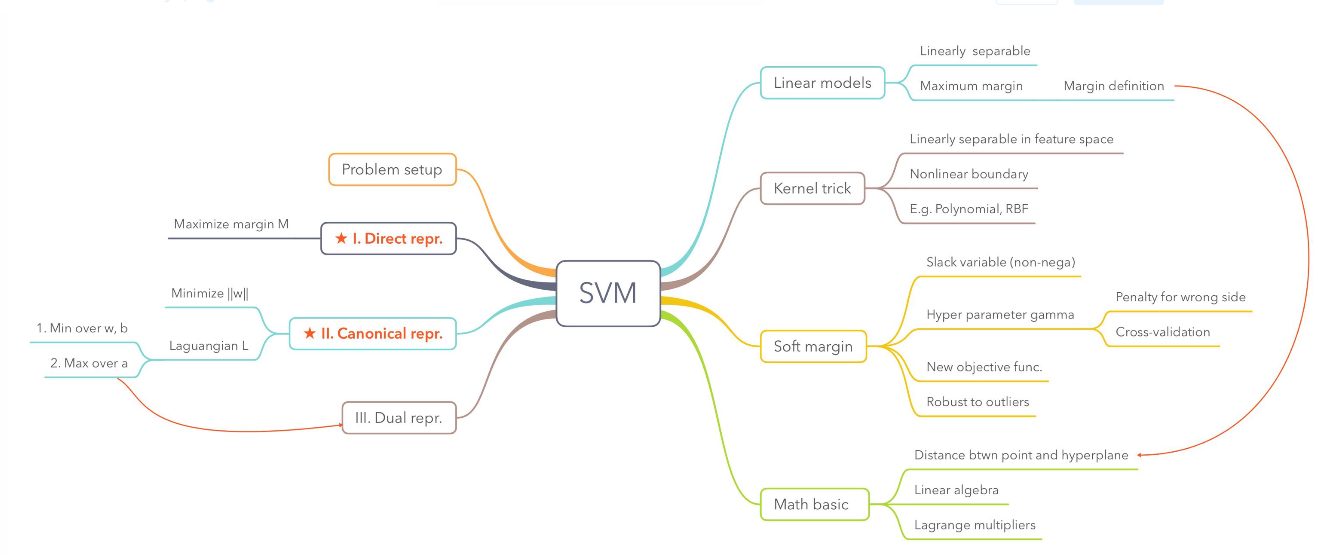
pca = PCA(n\_components = n\_components, svd\_solver='auto',

whiten=True).fit(self.x\_train0)

x\_train\_pca = pca.transform(self.x\_train0)

**4 分类器的构造与训练学习**

这是整个课设的核心部分，即构造建模合理的分类器。我们采用支持向量机(Support Vector Machine)进行分类。



SVM是Cortes和Vapnik于1995年首先提出的，它在解决小样本、非线性及高维模式识别中表现出许多特有的优势，并能够推广应用到函数拟合等其他机器学习问题中。

机器学习本质上就是一种对问题真实模型的逼近（我们选择一个我们认为比较好的近似模型，这个近似模型就叫做一个假设），但毫无疑问，真实模型一定是不知道的（如果知道了，我们干吗还要机器学习？直接用真实模型解决问题不就可以了？对吧，哈哈）既然真实模型不知道，那么我们选择的假设与问题真实解之间究竟有多大差距，我们就没法得知。比如说我们认为宇宙诞生于150亿年前的一场大爆炸，这个假设能够描述很多我们观察到的现象，但它与真实的宇宙模型之间还相差多少？谁也说不清，因为我们压根就不知道真实的宇宙模型到底是什么。

这个与问题真实解之间的误差，就叫做风险（更严格的说，误差的累积叫做风险）。我们选择了一个假设之后（更直观点说，我们得到了一个分类器以后），真实误差无从得知，但我们可以用某些可以掌握的量来逼近它。最直观的想法就是使用分类器在样本数据上的分类的结果与真实结果（因为样本是已经标注过的数据，是准确的数据）之间的差值来表示。这个差值叫做经验风险Remp(w)。以前的机器学习方法都把经验风险最小化作为努力的目标，但后来发现很多分类函数能够在样本集上轻易达到100%的正确率，在真实分类时却一塌糊涂（即所谓的推广能力差，或泛化能力差）。此时的情况便是选择了一个足够复杂的分类函数（它的VC维很高），能够精确的记住每一个样本，但对样本之外的数据一律分类错误。回头看看经验风险最小化原则我们就会发现，此原则适用的大前提是经验风险要确实能够逼近真实风险才行（行话叫一致），但实际上能逼近么？答案是不能，因为样本数相对于现实世界要分类的文本数来说简直九牛一毛，经验风险最小化原则只在这占很小比例的样本上做到没有误差，当然不能保证在更大比例的真实文本上也没有误差。

统计学习因此而引入了泛化误差界的概念，就是指真实风险应该由两部分内容刻画，一是经验风险，代表了分类器在给定样本上的误差；二是置信风险，代表了我们在多大程度上可以信任分类器在未知文本上分类的结果。很显然，第二部分是没有办法精确计算的，因此只能给出一个估计的区间，也使得整个误差只能计算上界，而无法计算准确的值（所以叫做泛化误差界，而不叫泛化误差）。

置信风险与两个量有关，一是样本数量，显然给定的样本数量越大，我们的学习结果越有可能正确，此时置信风险越小；二是分类函数的VC维，显然VC维越大，推广能力越差，置信风险会变大。

泛化误差界的公式为：

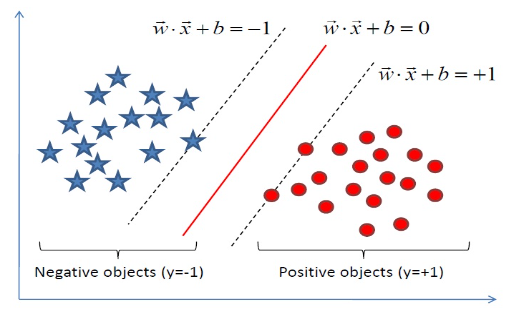
R(w)≤Remp(w)+Ф(n/h)

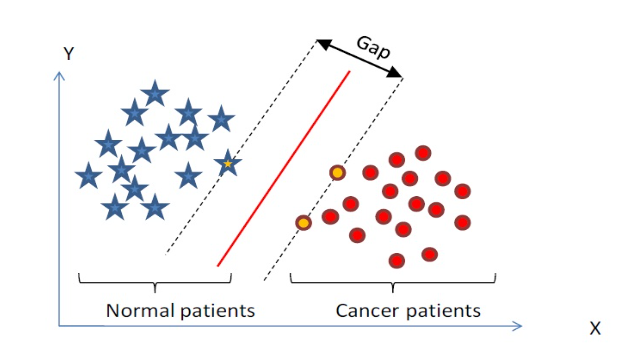
公式中R(w)就是真实风险，Remp(w)就是经验风险，Ф(n/h)就是置信风险。统计学习的目标从经验风险最小化变为了寻求经验风险与置信风险的和最小，即结构风险最小。

SVM正是这样一种努力最小化结构风险的算法。

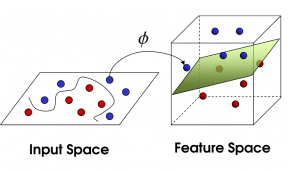
下面举例简述基本原理：

首先是线性SVM向量机，如图所示二维平面有两类点，我们要做的便是找到一条分类线将整个平面分为两部分，一部分属于蓝色五角星，一部分属于红色点，如过新加入的点落在五角星一侧便判定他为五角星类。

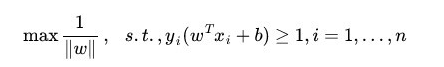


这些分类线可能有很多，现需要找到最优的分类线。

通俗地理解，只需要使得如图所示的Gap达到最大即可。同样的方法适用于多维多类的问题，此时分类线即分类超平面。

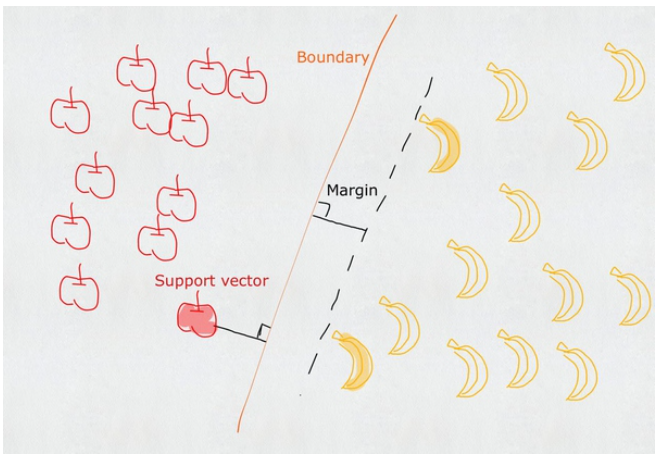


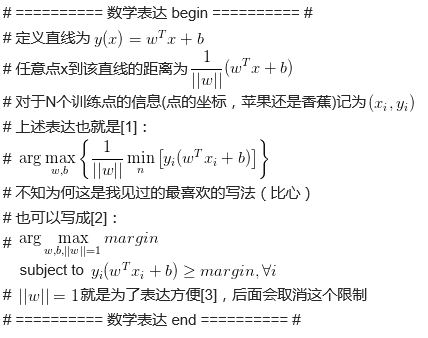
设w为这个超平面的法向量，中间具体的数学推导过程不再详述，总之我们要实现的优化是：



yI为正确的分类结果，xi为特征向量。

一个更简单的例子，要对苹果和香蕉进行分类：





SVM支持向量机则引入了非线性的神经网络，从而更好地解决非线性问题:

我们将使用一种叫做核函数（kernel）的工具，将数据从一个空间转换到另一个空间，使其变成易于分类器理解的形式。但要确保核函数可以很快速的进行计算，否则影响效率。

常用的核函数有：

1. 多项式核函数（Polynomial Kernel），其形式如下：

Kernel(X,X′)=(ξ+γXTX′)Q

2. 径向基函数（Radial Basis Function，RBF），其形式如下：

Kernel(X,X′)=e(−||X−X′||22σ2)

上述高斯核函数将数据从特征空间映射到更高维的空间，具体来说这里是映射到一个无穷维的空间。

将核函数（kernel）引入到 SVM 中也就是替换 SVM 中的 XTY 为 Kernel(X,Y)，也就是将第3部份第2步的二次规划规划问题中的 XTX 为 Kernel(X,X)，例如二次项核函数的 (1+XT∗X)2，将

b=ys−wTxs=ys−(∑Nn=1αnynxn)xs

替换为：

b=ys−∑Nn=1αnynKernel(xn,xs)

其中 s 表示某个支持向量。

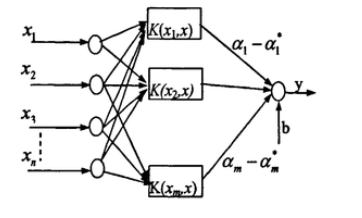
最后，将：

gsvm(x)=sign(wTx+b)=sign((∑Nn=1αnynxn)x+b)

替换为：

gsvm(x)=sign(∑Nn=1αnynKernel(xn,x)+b)

之后进行神经网络的建模，其中K(x1，x)为上述核函数：



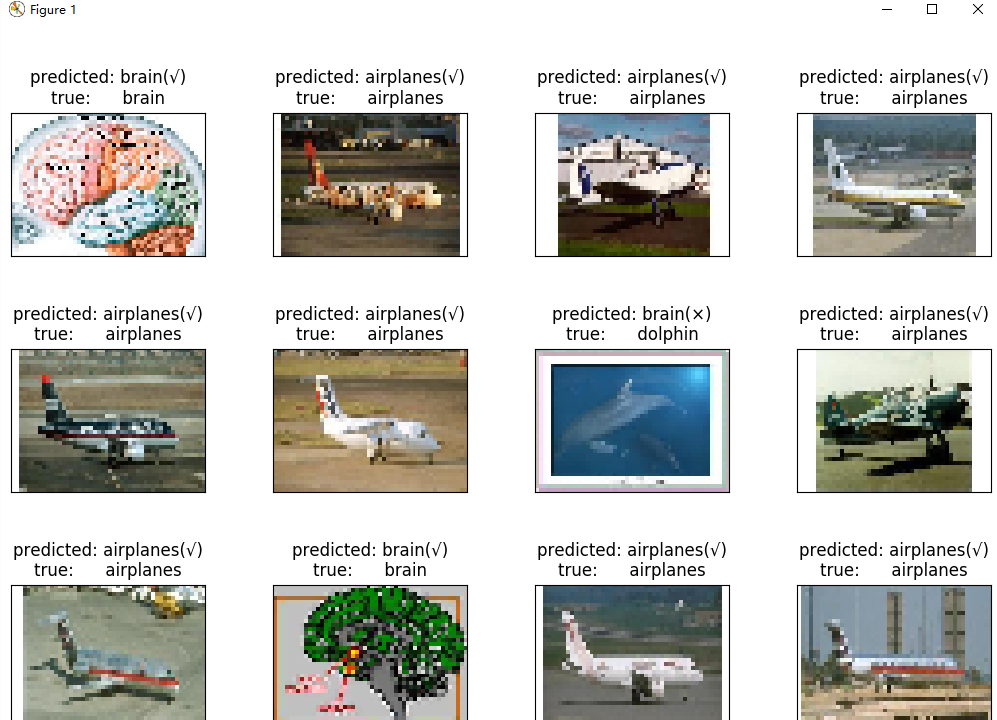
详细：<http://blog.csdn.net/passball/article/details/7661887/>

<http://blog.csdn.net/macyang/article/details/38782399/>

<https://www.zhihu.com/question/21094489>

**5 预测测试图片集，分析结果**

首先，对预测结果，采用matplotlib简单地绘制了预测结果和真实图片的比较：



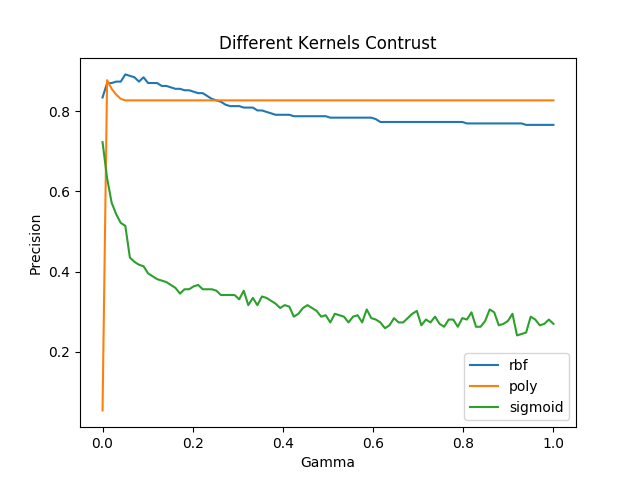
这里比较了12组预测结果，其中只有第7组预测有误，其他均预测正确！

下面是程序在沙盒下的运行结果：

|  |
| --- |
| Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32  Type "copyright", "credits" or "license()" for more information.  >>>  ====== RESTART: C:\Users\76947\Desktop\machine learning\plot\_gallary.py ======  正在采集图像数据...  图片信息采集完毕！花费了9.126秒  ------------------------------------------  总数据库的规模如下:  样本数: 1110  特征数: 80000  分类数: 5  ------------------------------------------  正在进行降维处理...  完成！花费了 28.129秒  数据分类降维完毕！  ------------------------------------------  正在进行训练...  完成，共计 8.308秒  最优分类向量机找到:  SVC(C=1000.0, cache\_size=200, class\_weight='balanced', coef0=0.0,  decision\_function\_shape=None, degree=3, gamma=0.1, kernel='rbf',  max\_iter=-1, probability=False, random\_state=None, shrinking=True,  tol=0.001, verbose=False)  ------------------------------------------  正在预测图片...  完成预测，共计 0.016秒  ------------------------------------------  预测结果如下：  precision recall f1-score support  airplanes 0.93 0.99 0.95 213  brain 0.77 0.81 0.79 21  chair 0.18 0.20 0.19 10  dolphin 0.80 0.42 0.55 19  sunflower 0.75 0.40 0.52 15  avg / total 0.87 0.87 0.86 278  [[210 1 1 1 0]  [ 2 17 2 0 0]  [ 7 0 2 0 1]  [ 4 1 5 8 1]  [ 4 3 1 1 6]]  ------------------------------------------  预测的准确率为： 0.8741007194244604  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'chair'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'sunflower', 'predict:': 'brain'}  {'target:': 'dolphin', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'sunflower'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'sunflower', 'predict:': 'sunflower'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'sunflower', 'predict:': 'sunflower'}  {'target:': 'dolphin', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'chair', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'dolphin'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'sunflower', 'predict:': 'sunflower'}  {'target:': 'airplanes', 'predict:': 'chair'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'dolphin', 'predict:': 'dolphin'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'chair', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'chair', 'predict:': 'sunflower'}  {'target:': 'dolphin', 'predict:': 'chair'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'sunflower', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'airplanes', 'predict:': 'airplanes'}  {'target:': 'brain', 'predict:': 'brain'}  >>> |

预测准确率有可观的0.8741007194244604，对飞机的预测更是达到了0.93的准确率，这里的预测准确度和学习的样本数有关，如果样本数足够多，相信我们会得到更可观的结果。

下面是采用不同核函数的预测准确率比较：



可以看到采用rbf径向基函数可达到很高的预测准确度。

敲黑板---------------

Readme.txt：

python version:python 3.5.3

需要安装的package：

numpy,scipy,matplotlib,sklearn,Image

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

**show\_precise.py ，plot\_gallary.py均为测试结果，分别运行这两个脚本即可**

**(将所有代码和数据库放在同一文件夹下，然后记得改路径)**

前者用于比较不同核函数下SVM向量机的性能

后者用于直观比较预测结果和正确的结果

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

face\_recognition.py 和 renlianshebie.py为参考文件

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

get\_eigenvalue.py 用于得到图片的特征向量，并进行PCA主要特征提取

get\_svm.py 用于生成支持向量机

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

图片库在dataset文件夹，只添加了5种分类，可按格式手动添加，添加路径时记住写对应路径

额外的图片库见两个压缩包

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

你正在阅读我想你的声音。