班 级 1402052
学 号 14020520013

西安电子科技大学

本科课程设计报告

题 目 Fisher 辨别分析用于人脸数据维数约简的实现

学 院 电子工程学院

专 业 智能科学与技术

学生姓名 董龙锐

导师姓名 冯婕

西 安 电 子 科 技 大 学

电 子 工 程 学 院

课 程 设 计（报告）任 务 书

学生姓名 董龙锐 指导教师 冯婕 职称 副教授

学生学号 14020520013 专业 智能科学与技术

题目 Fisher 辨别分析用于人脸数据维数约简的实现

相关专业课程 《模式识别》等

任务与要求

1、利用网络和图书馆查阅 Fisher 辨别分析的相关知识；

2、学习掌握 Python 编程语言；

3、掌握 Fisher 辨别分析的理论知识以及了解数据集的维数约简，投影等相关知识；

4、利用 Python 语言编程实现 Fisher 辨别分析用于人脸数据集的维数约简以及分类的验证，分析实验结果；

5、通过实验练习，让学生了解维数约简的知识并认真撰写课程设计报告。

开始日期 2018 年 1 月 2 日 完成日期 2018 年 1 月 26 日

课程设计所在单位 智能信息处理研究所 2018 年 1 月 19 日

Fisher 辨别分析用于人脸数据维数约简的实现

摘要：Fisher 辨别分析是模式识别领域的经典算法之一，它又称线性判别分析（Linear Discriminant Analysis, LDA）。本文基于 Python 语言，选取 ExtYaleB、AT&T （Olivettti）作为人脸数据集，利用 Fisher 辨别分析算法对两个数据集进行维数约简；然后对降维后的人脸数据进行分类，并对分类结果进行定量分析，最后做出讨论和总结。

关键词：Fisher 辨别分析，维数约简，分类，人脸数据

Abstract: Fisher discriminant analysis is one of the classical algorithms in the field of Pattern Recognition, which is also called Linear Discriminant Analysis (LDA). Based on the Python programing language, this paper selects ExtYaleB, AT&T (Olivettti) as the face data set and implements Fisher discriminant analysis algorithm to reduce the dimensionality of those two data sets; After that, the transformed face data are classified by LDA classifier, and the result of the classification are quantitatively analyzed. Finally, this paper makes a discussion and conclusion.

Keyword: Fisher discriminant analysis, Dimensionality reduction, Classification, Face data

1. 引言

在如今这个互联网+的新时代，可能最不稀有的就是数据了。因为数据的获取变得越来越容易且经济。但随之产生的负面后果是数据呈现出高维度、大规模的特点。举一个简单的例子[1]，在科学研究中，我们常常要对数据进行处理，而这些数据通常位于一个高维空间中，例如当处理一个 256*256 的图像序列时，我们需要将其拉成一个向量，这样，我们就得到了 4096 维的数据，如果直接对这些数据进行处理，会有以下问题：首先，会出现所谓的“维数灾难”问题，巨大的计算量将使我们无法忍受；其次，这些数据通常没有反映出数据的本质特征，如果直接对他们进行处理，不会得到理想的结果。为了高效的分析这些数据，一个主要的途径是维数约简，然后对约简后的数据进行处理。关键是要保证约简后的数据特征能反映甚至更能揭示原数据的本质特征。作为机器学习中的经典问题之一，维数约简主要用于处理维数灾难问题、帮助加速算法的计算效率和提高可解释性以及数据可视化[2]。该方向的研究分为无监督维数约简和有监督维数约简等。

那么维数约简方法都有哪些呢？数据维数约简的方法可以分为线性维数约简和非线性维数约简，而非线性维数约简又分为基于核函数的方法和基于特征值的方法。线性维数约简的方法主要有主成分分析（PCA）、独立成分分析（ICA）、线性判别分析（LDA）、局部特征分析（LFA）等等。

本文用到的就是经典的 Fisher 辨别分析（LDA）。线性判别分析是对费舍尔的线性

鉴别方法的归纳[3]，这种方法使用统计学，模式识别和机器学习方法，试图找到两类物体或事件的特征的一个线性组合，以能够特征化或区分它们。所得的组合可用来作为一个线性分类器，或者，更常见的是，为后续的分类做维数约简。为了展示该算法，本文选择 ExtYaleB、AT&T (Olivettti)两个经典的人脸数据集作为实验数据集。因为人脸数据的维数相对较高，可以检验实际降维效果。我们先用 LDA 算法的降维功能对人脸数据进行维数约简，然后再使用 LDA 的分类器功能对降维后的数据进行分类。最后对分类结果进行定量分析。分类的效果直接反映出维数约简的好坏：若维数约简后的数据仍能被准确的分类，则说明这样的降维揭示了数据的本质特征；反之则降维丢失了原始数据的重要信息。

2. 基础原理

数据的维数约减（又称“降维”）认为，尽管数据中的元素个数巨大，但是数据生成则被数量远小于数据维数的潜在因子控制，这些潜在因子被视为相关现象本质规律的体现。数据降维将数据由观测空间映射至一个低维空间，通过去除数据的无关信息而得到低维形式的表达，或是通过得到的低维空间探知数据的本质性规律[4]。数据维数约简可以描述为：给定观测数据 x，根据某种准则找到 x 的一个低维表示 s，使得 s 最大程度获得原空间的信息。在统计分析中，s 被称为隐藏因子[5]，而在模式识别领域中常用特征表示它[6]。

2.1 Fisher 辨别分析

Fisher 线性辨别分析（LDA/FDA）[7,8]是有监督线性投影的代表方法，基本思想是由 Fisher[9]最早提出的，目的是基于 Fisher 准则函数选择最优解向量作为最佳投影方向，从而在该方向上，同分类样本之间尽可能靠近，异类样本之间远离（如图 1），从而同时实现了分类功能。

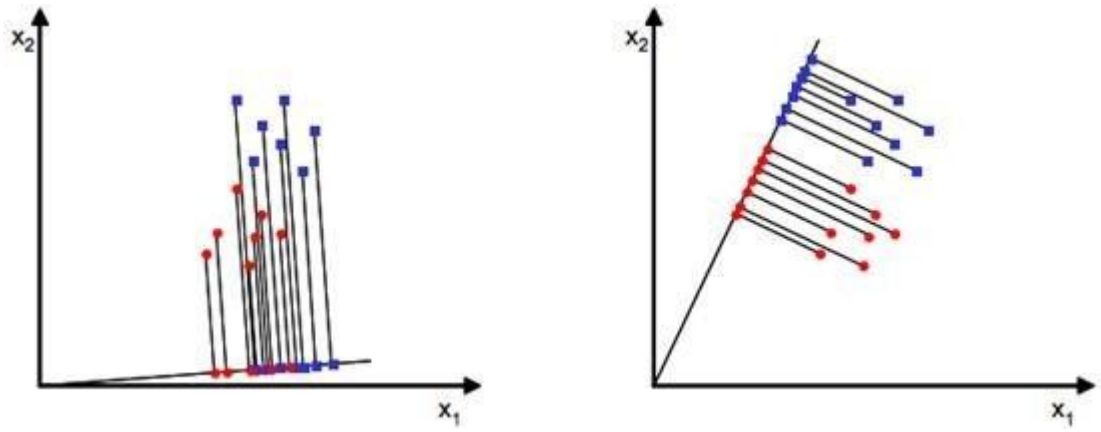

图 1　寻找最佳投影方向

这里以两类问题为例，先定义一些概念。训练样本集是 $\chi=\{x_1,...,x_N\}$，每个样本是一个 d 维向量，其中 w_1 类的样本是 $\chi_1=\{x_1^1,...,x_{N_1}^1\}$， w_2 类的样本是 $\chi_2=\{x_1^2,...,x_{N_2}^2\}$。我们要寻找一个投影方向 w，（ w 也是一个 d 维向量）投影以后的样本变成

$$y_i=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i,\quad i=1,2,....N \tag{1}$$

在原样本空间中，类均值向量为

$$\boldsymbol{m}_i=\frac{1}{N_i}\sum_{x_j\in\chi_i}\boldsymbol{x}_j,\quad i=1,2 \tag{2}$$

定义各类的类内离散度矩阵（within-class scatter matrix）为

$$\boldsymbol{S}_i=\sum_{x_j\in\chi_i}(\boldsymbol{x}_j-\boldsymbol{m}_i)(\boldsymbol{x}_j-\boldsymbol{m}_i)^{\mathrm{T}},\quad i=1,2 \tag{3}$$

总类内离散度矩阵（pooled within-class scatter matrix）为

$$\boldsymbol{S}_w=\boldsymbol{S}_1+\boldsymbol{S}_2 \tag{4}$$

类间离散度矩阵（between-class scatter matrix）定义为

$$\boldsymbol{S}_b=(\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}} \tag{5}$$

在投影以后的一维空间，两类的均值分别为

$$\tilde{m}_i=\frac{1}{N_i}\sum_{y_j\in\zeta_i}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_j=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{m}_i,\quad i=1,2 \tag{6}$$

类内离散度不再是一个矩阵，而是一个值

$$\tilde{S}_i^2=\sum_{y_j\in\zeta_i}(y_i-\tilde{m}_i)^2,\quad i=1,2 \tag{7}$$

总类内离散度为

$$\tilde{S}_w=\tilde{S}_1^2+\tilde{S}_2^2 \tag{8}$$

而类间离散度就成为两类均值之差的平方

$$\tilde{S}_b=(\tilde{m}_1-\tilde{m}_2)^2 \tag{9}$$

前面已经提出，希望寻找的投影方向使投影以后两类尽可能分开，而各类内部又尽可能聚集，这一目标可以表示成如下的准则

$$\max J_F(w)=\frac{\tilde{S}_b}{\tilde{S}_w}=\frac{(\tilde{m}_1-\tilde{m}_2)^2}{\tilde{S}_1^2+\tilde{S}_2^2} \tag{10}$$

这就是 Fisher 准则函数（Fisher's Criterion）。

把式 1 代入式 9 和式 7，得到

$$\tilde{S}_b=(\tilde{m}_1-\tilde{m}_2)^2=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_b\boldsymbol{w} \tag{11}$$

以及

$$\tilde{S}_w=\tilde{S}_1^2+\tilde{S}_2^2=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w} \tag{12}$$

因此，Fisher 判别准则变成

$$\max_{\boldsymbol{w}} J_F(\boldsymbol{w})=\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_b\boldsymbol{w}}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w}} \tag{13}$$

这一表达式在数学物理中被称作广义 Rayleigh 商（generalized Rayleigh quotient）。

应注意到，我们的目的是求使得式 13 最大的投影方向 **w**。由于对 **w** 幅值的调节并不会影响 **w** 的方向，即不会影响 $J_F(\boldsymbol{w})$ 的值，因此，可以设定式（13）的分母为非零常

数而最大化分子部分，即把式（13）的优化问题转化为

$$\begin{aligned} &\max \ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_b\boldsymbol{w} \\ &s.t. \quad \boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w}=c\neq 0 \end{aligned} \tag{14}$$

这是一个等式约束下的极值问题，可以通过引入拉格朗日（Lagrange）乘子转化成以下拉格朗日函数的无约极值问题：

$$L(\boldsymbol{w},\lambda)=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_b\boldsymbol{w}-\lambda(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{S}_w\boldsymbol{w}-c) \tag{15}$$

在式 15 的极值处，应该满足

$$\frac{\partial L(\boldsymbol{w},\lambda)}{\partial \boldsymbol{w}}=0 \tag{16}$$

由此可得，极值解 **w***应满足

$$\boldsymbol{S}_b\boldsymbol{w}^*-\lambda\boldsymbol{S}_w\boldsymbol{w}^*=0 \tag{17}$$

假定 $\boldsymbol{S}_w$ 是非奇异的，可以得到

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b\boldsymbol{w}^*=\lambda\boldsymbol{w}^* \tag{18}$$

也就是说，**w***是矩阵 $\boldsymbol{S}_w^{-1}\boldsymbol{S}_b$ 的本征向量。我们把式 5 代入，式 18 变成

$$\lambda\boldsymbol{w}^*=S_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w}^* \tag{19}$$

应注意到，$(\boldsymbol{m}_1-\boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{w}*$ 是标量，不影响 **w***的方向，因此可以得到 **w***的方向是由 $S_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2)$ 决定的。由于我们只关心 **w***的方向，因此可以取

$$\boldsymbol{w}^*=\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2) \tag{20}$$

这就是 Fisher 判别准则下的最优投影方向[10]。

需要注意的是，Fisher 判别函数最优的解本身只是给出了一个投影方向，并没有给出我们所要的分类面。要得到分类面，需要在投影方向（一维空间）上确定一个分类阈值 w_0，并采取决策规则

$$若\boldsymbol{g}(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0\leqslant 0,则\boldsymbol{x}\in\begin{cases}w_1\\ w_2\end{cases} \tag{21}$$

当样本是正态分布且两类协方差矩阵相同时，最优贝叶斯分类器是线性函数 $\boldsymbol{g}(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+w_0$，且其中

$$\begin{aligned} &\boldsymbol{w}=\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2) \\ &w_0=-\frac{1}{2}(\boldsymbol{\mu}_1+\boldsymbol{\mu}_2)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_2)-\ln\frac{P(w_2)}{P(w_1)} \end{aligned} \tag{22}$$

在样本为正态分布且两类协方差相同的情况下，如果把样本的算术平均作为均值的估计、把样本的协方差矩阵当作是真实协方差矩阵的估计，则 Fisher 线性判别所得到的方向实际就是最优贝叶斯决策的方向，因此可以用式 22 中的 w_0 作为分类阈值，其中，m_i 代替 $\boldsymbol{\mu}_i$，用 $\boldsymbol{S}_w^{-1}$ 代替 $\boldsymbol{\Sigma}^{-1}$，即

$$w_0=-\frac{1}{2}(\boldsymbol{m}_1+\boldsymbol{m}_2)^{\mathrm{T}}\boldsymbol{S}_w^{-1}(\boldsymbol{m}_1-\boldsymbol{m}_2)-\ln\frac{P(w_2)}{P(w_1)} \tag{23}$$

在样本不是正态分布时，这种投影方向和阈值并不能保证是最优的，但通常仍可以取得较好的分类结果。

如果不考虑先验概率的不同，则可以采用阈值

$$w_0 = -\frac{1}{2}(\tilde{m}_1 + \tilde{m}_2) \tag{24}$$

或者

$$w_0 = -\tilde{m} \tag{25}$$

其中，$\tilde{m}$ 是所有样本在投影后的均值。

把式 23 代入式 21 并考虑到式 20，可以把决策规则写成

$$若g(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}(\boldsymbol{x} - \frac{1}{2}(\boldsymbol{m}_1 + \boldsymbol{m}_2) \lessgtr \ln\frac{P(w_2)}{P(w_1)}, 则\boldsymbol{x} \in \begin{cases} w_1 \\ w_2 \end{cases} \tag{26}$$

其直观的解释就是，把待决策的样本投影到 Fisher 判别的方向上，通过与两类均值投影的平分点相比较做出分类决策。在先验概率相同的情况下，以该平分点为两类的分类点；在先验概率不同的时候，分界点向先验概率小的那一侧偏移，如图 2 所示。

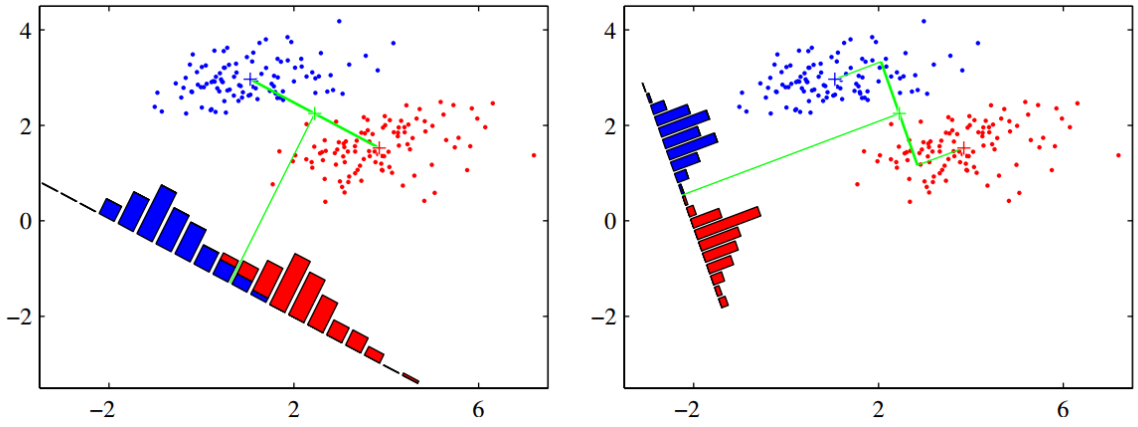

图 2　Fisher 线性判别示意图[11]

2.2 人脸数据集介绍

本文选取了 ExtYaleB[12]、AT&T (Olivettti)[13]两个经典的人脸数据集作为实验数据集，接下来进行详细介绍。

（1） ExtYaleB 数据库

该数据库包含 576 个观看条件下（9 个姿势×64 个照明条件）下看到的 10 个对象的 5760 个单光源图像。对于特定姿势中的每个对象，还捕捉到具有环境（背景）照明的图像。因此，图像的总数实际上是 5760＋90＝5850。压缩数据库的总大小约为 1GB。

一个特定姿势的主体的 65 个（64 个照明+ 1 个环境）图像已经被“涂抹”和“压缩”成单个文件。有 90 个（10 个主题×9 个姿势）'* .tar.gz'文件。每个'* .tar.gz'文件大约 11MB。所有文件名都以基本名称“yaleB”开头，后面跟着一个两位数字，表示主题编号（01 - 10）。'_P'后面的 2 位数表示姿势号（00 - 08）。

关于每个图像的命名的解释：图像的文件名的第一部分遵循与“tar”（和“gzip”）文件的文件名相同的约定。它以基本名称“yaleB”开头，后面跟着两位数字表示主体编号，然后是两位数字表示姿势。文件名的其余部分处理单个光源方向的方位角和仰角。比如：

'yaleB03_P06A+035E+40.pgm'

属于在姿势＃6 中看到的主题＃3，并且相对于相机轴的光源方向在 35 度方位（'A＋035'）和 40 度仰角（'E＋40'）处。（请注意，正方位表示光源位于拍摄对象的右侧，负值表示位于左侧，正视图位于水平线以上，负值指位于地平线以下）。请注意，有 47（超出 5760）相应的闪光灯没有熄灭的图像。这些图像基本上看起来像在特定姿势中的主体的环境图像。另外还有 4 张图片稍有损坏; 在获取过程中，每帧的奇偶场强度存在一个微小的不平衡。

获取的图像是用 Sony XC-75 相机（具有线性响应功能）捕捉的 8 位（灰度）图像，并以 PGM 原始格式存储（如图 3）。

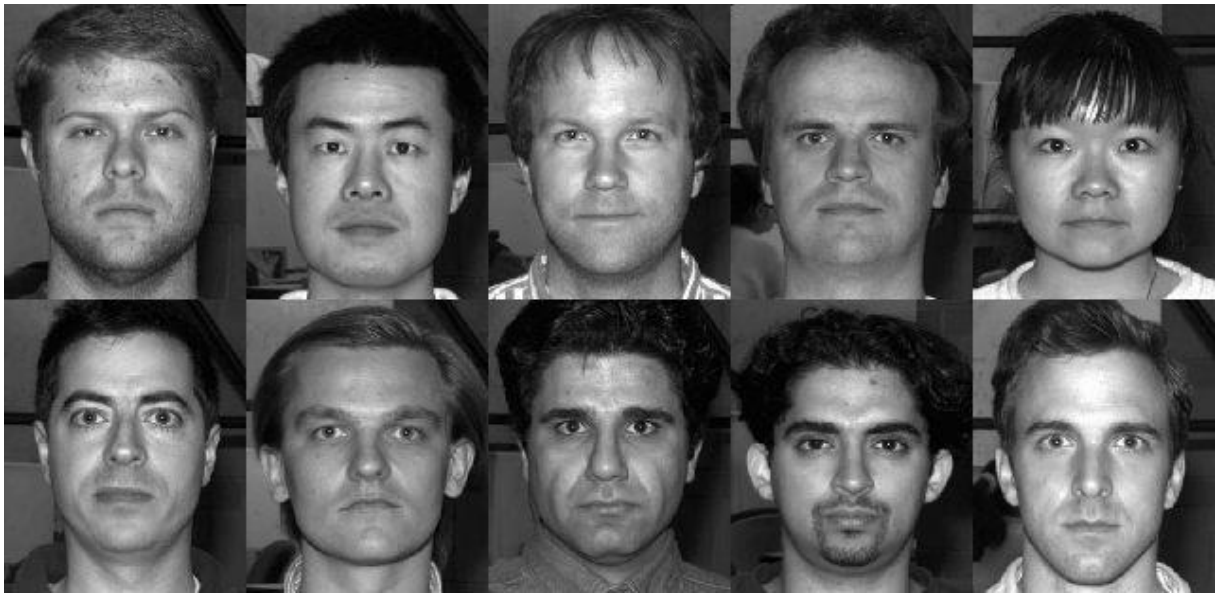

图 3　ExtYaleB 人脸数据集（部分）

扩大的数据库，首次由 Kuang-Chih Lee，Jeffrey Ho 和 David Kriegman 在“Acquiring Linear Subspaces for Face Recognition under Variable Lighting, PAMI, May, 2005”。实验中使用的所有测试图像数据都是手动对齐，剪裁，然后重新调整为 168x192 的图像。 如果要使用裁剪的图像发布实验结果，请参阅 PAMI2005 论文[14]。

（2） AT&T (Olivettti)数据库

该面孔数据库（以前的“The ORL Database of Faces”）包含了 1992 年 4 月至 1994 年 4 月间在实验室拍摄的一组面部照片。该数据库被用于与剑桥大学工程部的言语，视

觉和机器人小组合作的一个面部识别项目。

每个 40 个不同的人物有十个不同的图像（如图 4）。对于一些人物，图像是在不同的时间拍摄的，改变了照明，面部表情（开放/闭眼，微笑/不微笑）和面部细节（眼镜/没有眼镜）。所有图像都是在黑暗的均匀背景下拍摄的，拍摄对象处于直立的正面位置（对一些侧面偏移有容忍）。

这些文件是 PGM 格式的，可以使用'xv'程序方便地在 UNIX（TM）系统上查看。每个图像的大小是 92×112 像素，每个像素有 256 个灰度级。图像被组织在 40 个目录中（每个人物一个），其名称形式为 sX，其中 X 表示主题编号（1 到 40 之间）。在这些目录中，有十个不同的主题图像，名称的格式为 Y.pgm，其中 Y 是该主题的图像编号（介于 1 和 10 之间）。

图 4　AT&T 人脸数据集（部分）

3. 方案设计

本文的方案大体上分为两个模块，模块框图见图 5：

图 5　系统模块框图

程序流程图见图 6：

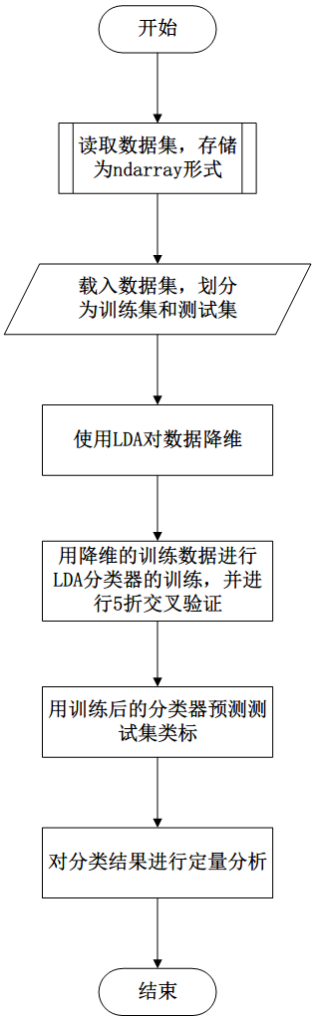

图 6　程序流程图

4. 结果与总结

本次试验中的测试集共有 210 个人脸样本，测试集是总数据集的 20%。其中有 130 个测试样本来自 ExtYaleB，他们是通过从 10 类中每一类随机抽取 13 个样本得到的；余下的 80 个来自 AT&T，同理他们通过 40 类中每一类随机抽取 2 个样本得到。

维数约简前后的特征维数情况如表 1：

表 1　降维前后的特征维数

	ExtYaleB	AT&T
原始数据特征维数	32256	10304
维数约简后的维数	9	39

LDA 分类器对测试集人脸的类标预测结果见图 7，图 8：

图 7　ExtYaleB 部分分类结果

图 8　AT&T 部分分类结果

4.1 定量分析

本文选取了平均分类精度（average accuracy,AA）、总体分类精度(overall accuracy,OA)，Kappa 系数作为定量分析的指标，结果见表 2。

表 2　实验结果分析

数　据　库 指　　标	**ExtYaleB**	**AT&T**
平均分类精度（AA）	0.98 (+/−0.01)	0.99 (+/−0.02)
总体分类精度（OA）	0.9615	0.9625
Kappa 系数	0.9573	0.9615

（注：平均分类精度后的“+/−0.01”表示方差）

除了以上的指标，还绘制了两个数据集的的正则化混淆矩阵可视化结果[15]，如图 9，图 10。

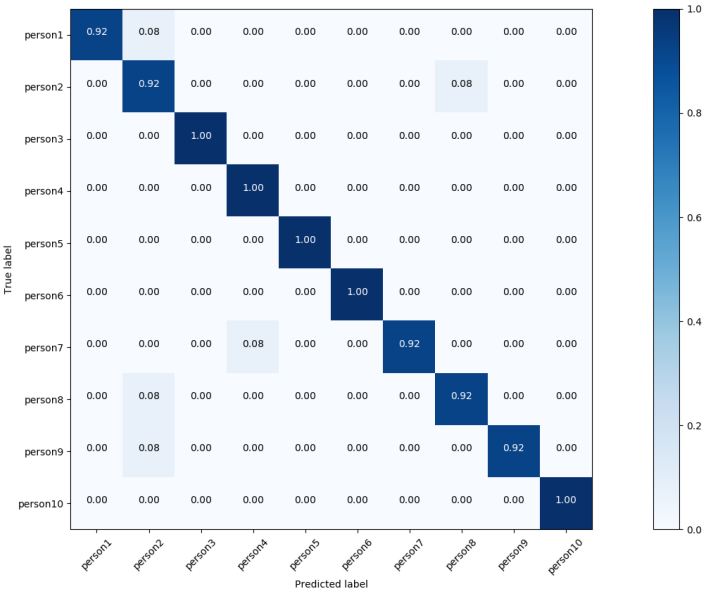

图 9　ExtYaleB 分类结果的混淆矩阵

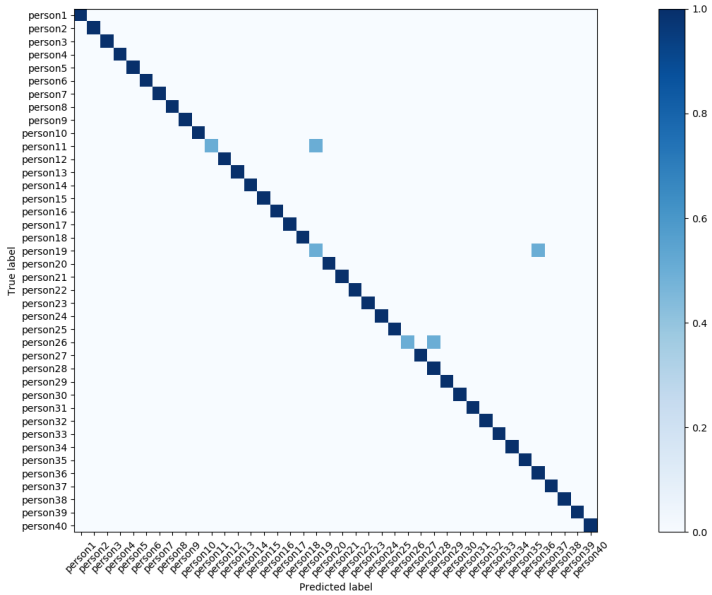

图 10　AT&T 分类结果的混淆矩阵

具体的混淆矩阵的值见附录部分。

4.2 实验总结

本文基于 Python 语言，编程实现了 Fisher 线性辨别分析算法，并将该算法运用到人脸数据集进行维数约简，继而通过分类实验分析降维效果。通过定量分析，我们发现：

1) 通过 5 折交叉验证得到了该分类器的平均分类准确率达到了 98%以上，且方差不超过 0.02；
2) 通过对测试数据的分类得到准确率均达到 96.1%以上，说明降维后的数据仍保留了原始重要信息；
3) 分类结果的 Kappa 系数分别为 0.9573、0.9615，远大于 0.8，证明了分类的准确，继而说明降维的有效性。

当然未来的工作还可以继续改善以下方面：

1) 在求解降维的最佳投影方向时需要计算协方差矩阵，然而当数据维数很高时，这将大大增加计算耗费，可以考虑先对数据进行预处理；
2) 可以考虑使用其他维数约简的算法去和 LDA 进行对比。

5. 参考文献

[1] Dimensionality reduction.
http://www.baike.com/wiki/%E7%BB%B4%E6%95%B0%E7%BA%A6%E7%AE%80.
[2] 单洪明,张军平.实值多变量维数约简:综述[J/OL].自动化学报:1-23[2018-01-10].
[3] LDA.
https://zh.wikipedia.org/wiki/%E7%B7%9A%E6%80%A7%E5%88%A4%E5%88%A5%E5%88%86%E6%9E%90.
[4] 吴松松. 维数约简研究及其在特征提取中的应用[D].南京理工大学,2012.
[5] K. Mardia, J. Kent, J. Bibby. Multivariate Analysis[M]. Academic press, 1980.
[6] F. Keinosuke. Introduction to Satistical Pattern Recognition[M]. 1990.
[7] R. Hogg, A. Craig. Introduction to Mathematical Statistics[M]. Prentice Hall, 2004.
[8] R. Duda, P. Hart, D. Stork. Pattern Classification and Scene Analysis 2nd Ed.[M]. 1995.
[9] R. Fisher. The Use of Multiple Measurement in Taxonomic Problems[J]. Annals of Human Genetics, 1936, 7(2):179—188.
[10] 张学工. 模式识别. 第三版. 北京：清华大学出版社,2010
[11] 线性判别分析. http://blog.csdn.net/daunxx/article/details/51881956.
[12] The Extended Yale Face Database B.
http://vision.ucsd.edu/~iskwak/ExtYaleDatabase/ExtYaleB.html.
[13] The Database of Faces.
http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[14] Lee K C, Ho J, Kriegman D. Nine Points of Light: Acquiring Subspaces for Face Recognition under Variable Lighting[C]// IEEE Computer Society Conference on Computer Vision & Pattern Recognition. IEEE Computer Society, 2001:519.
[15] Confusion matrix.
http://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#example-model-selection-plot-confusion-matrix-py .

6. 附录

6.1 Python 代码

```
# -*- coding: utf-8 -*-
'''
基于 LDA，对人脸数据集进行维数约减，并计算分类准确率
Created by Longrui Dong --2018.01.05
'''
from __future__ import print_function
import matplotlib.pyplot as plt
from time import time
import logging

import itertools
import numpy as np
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import cohen_kappa_score
from sklearn.metrics import confusion_matrix



print(__doc__)

# Display progress logs on stdout
logging.basicConfig(level=logging.INFO, format='%(asctime)s %(message)s')

PICTURE_PATH1 = "D:\\mycollege\\term7\\keshe\\CroppedYale"#yaleb 人脸数据  10*65
PICTURE_PATH2 = "D:\\mycollege\\term7\\keshe\\att_faces"#at&t 人脸数据     40*10
#载入 npy 文件，得到整个数据集的 ndarray
data_1=np.load(PICTURE_PATH1 + "\\"+"alldataset.npy")
label_1=np.load(PICTURE_PATH1 + "\\"+"alldatalabel.npy")
data_2=np.load(PICTURE_PATH2 + "\\"+"alldataset.npy")
label_2=np.load(PICTURE_PATH2 + "\\"+"alldatalabel.npy")

n_samples1,n_features1=data_1.shape
n_classes1 = len(np.unique(label_1))
```

```
n_samples2,n_features2=data_2.shape
n_classes2 = len(np.unique(label_2))

h1=192
w1=168
h2=112
w2=92

target_names1 = []
target_names2 = []
for i in range(1,11):
    names = "person" + str(i)
    target_names1.append(names)

for i in range(1,41):
    names = "person" + str(i)
    target_names2.append(names)

#输出数据集信息
print("Total 1th dataset size:")
print("n_samples: %d" % n_samples1)
print("n_features: %d" % n_features1)
print("n_classes: %d" % n_classes1)
print("Total 2nd dataset size:")
print("n_samples: %d" % n_samples2)
print("n_features: %d" % n_features2)
print("n_classes: %d" % n_classes2)
#将数据集划分为训练集和测试集
X_train1,X_test1,y_train1,y_test1=train_test_split(data_1,label_1,test_size=130,random_s
tate=0,stratify=label_1)
X_train2,X_test2,y_train2,y_test2=train_test_split(data_2,label_2,test_size=80,random_st
ate=0,stratify=label_2)

#对训练数据和测试数据降维
print("对训练数据和测试数据降维")
t0=time()
lda_1=LinearDiscriminantAnalysis(solver='svd')
```

```
lda_1=lda_1.fit(X_train1,y_train1)
X_train1_new=lda_1.transform(X_train1)
X_test1_new=lda_1.transform(X_test1)
print("transformed on data1 done in %0.3fs" % (time() - t0))

t0=time()
lda_2=LinearDiscriminantAnalysis(solver='svd')
lda_2=lda_2.fit(X_train2,y_train2)
X_train2_new=lda_2.transform(X_train2)
X_test2_new=lda_2.transform(X_test2)
print("transformed on data1 done in %0.3fs" % (time() - t0))

#输出降维后的数据集信息
print("Total 1th transformed dataset size:")
print("n_samples_train: %d" % X_train1_new.shape[0])
print("n_features_train: %d" % X_train1_new.shape[1])
print("n_samples_test: %d" % X_test1_new.shape[0])
print("n_features_test: %d" % X_test1_new.shape[1])
print("Total 2nd transformed dataset size:")
print("n_samples_train: %d" % X_train2_new.shape[0])
print("n_features_train: %d" % X_train2_new.shape[1])
print("n_samples_test: %d" % X_test2_new.shape[0])
print("n_features_test: %d" % X_test2_new.shape[1])
#使用 lda 进行分类验证
#使用 5 折交叉验证来展示分类效果
print("使用 5 折交叉验证来展示对训练数据的分类效果")
t0=time()
clf1=LinearDiscriminantAnalysis(solver='lsqr',shrinkage='auto')
clf2=LinearDiscriminantAnalysis(solver='lsqr',shrinkage='auto')
scores1=cross_val_score(clf1, X_train1_new, y_train1, cv=5)
scores2=cross_val_score(clf2, X_train2_new, y_train2, cv=5)
print("done in %0.3fs" % (time() - t0))

#print("5-fold cross validation scores:")
print("5-fold cross validation accuracy on data1: %0.2f (+/- %0.2f)" % (scores1.mean(),
scores1.std() * 2))
```

```
print("5-fold cross validation accuracy on data2: %0.2f (+/- %0.2f)" % (scores2.mean(),
scores2.std() * 2))



#使用训练好的分类器对测试数据进行预测
print("使用 lda 分类器先对训练数据进行训练，再对测试数据进行预测:")
t0=time()
y_test1_pred=clf1.fit(X_train1_new, y_train1).predict(X_test1_new)
y_test2_pred=clf2.fit(X_train2_new, y_train2).predict(X_test2_new)
print("done in %0.3fs" % (time() - t0))
#计算测试数据集上的准确率
comp1=y_test1_pred-y_test1#比较预测的结果和真实类标
comp2=y_test2_pred-y_test2
ac_rate1=np.sum(comp1==0)/X_test1_new.shape[0]#获得正确率
ac_rate2=np.sum(comp2==0)/X_test2_new.shape[0]

print("Accuracy on test data on data1: %0.4f" %ac_rate1)
print("Accuracy on test data on data2: %0.4f" %ac_rate2)
#计算 kappa score
kappa_1=cohen_kappa_score(y_test1, y_test1_pred)
kappa_2=cohen_kappa_score(y_test2, y_test2_pred)
print("Cohen's kappa score on test data on data1: %0.4f" %kappa_1)
print("Cohen's kappa score on test data on data2: %0.4f" %kappa_2)

def plot_confusion_matrix(cm, path, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

```
    print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    #是否填混淆矩阵的数字
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
    #
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    fig = plt.gcf()
    fig.set_size_inches(18.5, 10.5)
    fig.savefig(path+"\\"+"confusionmatrix.png", dpi=100)

#计算混淆矩阵
cnf_matrix_1 = confusion_matrix(y_test1, y_test1_pred)
cnf_matrix_2 = confusion_matrix(y_test2, y_test2_pred)
np.set_printoptions(precision=2)

plt.figure()
plot_confusion_matrix(cnf_matrix_1, PICTURE_PATH1, classes=target_names1,
normalize=True, title='Normalized confusion matrix on data1')
plt.figure()
plot_confusion_matrix(cnf_matrix_2, PICTURE_PATH2, classes=target_names2,
normalize=True, title='Normalized confusion matrix on data2')
```

```
plt.show()


def plot_gallery(images, titles, h, w, n_row, n_col):
    """Helper function to plot a gallery of portraits"""
    plt.figure(figsize=(1.8 * n_col, 2.4 * n_row))
    plt.subplots_adjust(bottom=0, left=.01, right=.99, top=.90, hspace=.35)
    for i in range(n_row * n_col):
        plt.subplot(n_row, n_col, i + 1)
        plt.imshow(images[i].reshape((h, w)), cmap=plt.cm.gray)
        plt.title(titles[i], size=12)
        plt.xticks(())
        plt.yticks(())



def title(y_pred, y_test, target_names, i):
        #展示测试集预测结果图的题目
    pred_name = target_names[y_pred[i]-1]
    true_name = target_names[y_test[i]-1]
    return 'predicted: %s\ntrue:      %s' % (pred_name, true_name)

prediction_titles1 = [title(y_test1_pred, y_test1, target_names1, i)
                      for i in range(y_test1_pred.shape[0])]

prediction_titles2 = [title(y_test2_pred, y_test2, target_names2, i)
                      for i in range(y_test2_pred.shape[0])]



print("画图展示分类效果：")
plot_gallery(X_test1, prediction_titles1, h1, w1, 10, 13)
plot_gallery(X_test2, prediction_titles2, h2, w2, 10, 8)
```

6.2 具体结果记录

Normalized confusion matrix on ExtYaleB

```
[[ 0.92  0.08  0.    0.    0.    0.    0.    0.    0.    0.  ]
 [ 0.    0.92  0.    0.    0.    0.    0.    0.08  0.    0.  ]
 [ 0.    0.    1.    0.    0.    0.    0.    0.    0.    0.  ]
```

```
[ 0.    0.    0.    1.    0.    0.    0.    0.    0.    0.  ]
[ 0.    0.    0.    0.    1.    0.    0.    0.    0.    0.  ]
[ 0.    0.    0.    0.    0.    1.    0.    0.    0.    0.  ]
[ 0.    0.    0.    0.08  0.    0.    0.92  0.    0.    0.  ]
[ 0.    0.08  0.    0.    0.    0.    0.    0.92  0.    0.  ]
[ 0.    0.08  0.    0.    0.    0.    0.    0.    0.92  0.  ]
[ 0.    0.    0.    0.    0.    0.    0.    0.    0.    1.  ]]
```

Normalized confusion matrix on AT&T

```
[[ 1.   0.   0. ...,   0.   0.   0.]
 [ 0.   1.   0. ...,   0.   0.   0.]
 [ 0.   0.   1. ...,   0.   0.   0.]
 ...,
 [ 0.   0.   0. ...,   1.   0.   0.]
 [ 0.   0.   0. ...,   0.   1.   0.]
 [ 0.   0.   0. ...,   0.   0.   1.]]
```