基于流形学习子空间的人脸识别算法

姜希成

（信息与电气工程学院，计算机科学与技术，2015级2班，20152203031）

**摘 要：** 近年来，人脸识别技术得到突飞猛进的发展，与此同时，流形学习方法也取得巨大进展，所以本文以人脸识别为应用背景，对几种常见的流形学习算法进行研究。主要研究了等距映射算法（IsoProjection）、邻域保护嵌入算法（NPE）以及正交的等距映射算法（OIsoProjection）和正交的邻域保护嵌入算法（ONPE）。在本文最后将对这些算法进行对比，研究每种算法的优缺点，并在ORL人脸库上进行试验，验证这些算法的有效性。

**关键字：**人脸识别，流形学习，数据降维，线性图嵌入，邻域保护嵌入，正交邻域保护嵌入，等距映射，正交等距映射，多维缩放算法，等度量映射，局部线性嵌入，正交化。

**Face recognition algorithm based on manifold learning subspace**

Jiang XiCheng

(School of Information and Electrical Engineering, Ludong University)

**Abstract:** In recent years, face recognition technology has developed by leaps and bounds. At the same time, the manifold learning method has made great progress. Therefore, this paper uses face recognition as the application background to study several common manifold learning algorithms. The isometric mapping algorithm (IsoProjection), the neighborhood protection embedding algorithm (NPE) and the orthogonal isometric mapping algorithm (OIsoProjection) and the orthogonal neighborhood protection embedding algorithm (ONPE) are mainly studied. At the end of the paper, we compare these algorithms, study the advantages and disadvantages of each algorithm, and test on the ORL face database to verify the effectiveness of these algorithms.

**Key words:** Face recognition, Manifold Learning, Data reduction dimension, LGE, NPE, ONPE, IsoProjection, OIsoProjection, MDS, ISOMAP, LLE, Orthogonal .

# 1引言

人脸识别技术和扩展已应用于生活的各个方面，人脸识别技术具有广泛的应用前景，可以用于考勤、门禁、关口同行、社区安防、民航、保险、军事安全、银行金融系统、追击嫌疑犯和反恐等等。近年来，人脸识别技术及相关算法的飞速发展也为该技术提供了强有力的支持。

人脸识别技术在生物特征识别技术中占据非常重要的位置，本文将介绍如何使用流形学习子空间方法来实现人脸识别技术。

## 1.1人脸识别技术简介

党的十九大报告提出要推进互联网、大数据、人工智能和实体经济的深度融合，人脸识别技术作为人工智能与实体经济相结合的重要技术支撑，近几年来得到飞速发展。

### 1.1.1人脸识别技术的发展简介

人脸识别是基于人脸特征信息的身份识别。人脸识别技术始于20世纪60年代，到现在已经有了近60年的发展史，在这60年中人脸识别技术得到突飞猛进的发展，相继出现了许多经典算法、经典思想和经典人脸库。

目前，人脸识别系统最高的正确率超过99.5%，而人眼在同等条件下识别的正确率仅为97.52%，人脸识别的准确率已经做到了比肉眼更精准，我相信在未来人脸识别技术一定能我们创造更加便利的生活条件。

### 1.1.2人脸识别技术的优势

人脸识别是生物识别的一种，生物识别因为具有易检测、唯一性和终身不变的特点，所以十分适合互联网时代用户对安全的需求，如今生物识别技术的识别速度更快，准确率更高，因此也更具有研究的价值。

目前主流的生物识别技术有人脸识别、指纹识别、虹膜识别、语音识别、静脉识别等。相对于其他的生物识别技术，人脸识别技术的成本更低、稳定性更好、准确率更高，因此也更具有发展优势，所以在多数应用场景中都会首选人脸识别技术。

### 1.1.3人脸识别技术的市场前景

市场前景是判断一项技术是否有研究价值的重要指标。

人脸识别市场前景非常好，发展势头也非常迅速，2017年全球人脸识别的市场规模超过40.5亿美元，预计2022年达到77.6亿美元，复合年增长率高达13.9%。

根据前瞻产业研究院的报告，目前，人脸识别在考勤/门禁领域的应用最为成熟，约占行业市场42%左右；安全防范作为人脸识别最早应用的领域之一，其市场份额占比在30%左右；金融作为人脸识别未来重要的应用领域之一，约占行业的 20%。从应用产品来看，人脸识别市场中嵌入式设备占比达53%，软件开发包(SDK)支持的联机应用占比为47%。

同时大规模普及的软硬件基础条件已具备，产品系列达20多种类型，下游应用领域（尤其是金融、安防、互联网领域）需求强劲，这一切都预示着人脸识别技术的爆发点已经到来。

### 1.1.4人脸识别系统的研究内容

图1-1显示了人脸识别的一般步骤：

特征提取与选择

训练

人脸图像库图像

人脸的检测与定位

图像的预处理理

特征提取与选择

获取人脸图像

人脸的检测与定位

图像的预处理理

识别

图 1-1

可见人脸识别系统一般包括人脸检测与定位、图像预处理、人脸图像的特征提取与选择等。横线上方的是人脸识别系统的训练部分，横线下方的人脸识别系统的识别部分。

人脸识别系统在进行特征提取之前需要进行人脸检测与定位和图像预处理，这些操作主要包括几何归一化和灰度归一化。几何归一化是根据人脸检测定位的结果将图像中的人脸变换到同一位置和大小，灰度归一化是将图像进行光照补偿等处理来克服图像中的光照变化对人脸识别的影响。人脸识别过程是将待识别图像的特征和库中的特征进行比对匹配来确定图像中人脸的身份。

## 1.2流形学习

是一类借鉴了拓扑流形概念的降维方法。"流形"是在局部与欧氏空间同胚的空间，换言之，它在局部具有欧氏空间的性质，能用欧氏距离来进行距离计算。这给降维方法带来了很大的启发：若低维流形嵌入到高维空间中，则数据样本在高维空间的分布虽然看上去非常复杂，但在局部上仍具有欧氏空间的性质，因此，可以容易地在局部建立阵维映射关系，然后再设法将局部映射关系推广到全局.当维数被降至二维或三维时，就可以对数据进行可视化展示因此流形学习也可被用于可视化。

流形学习的本质在于根据有限的离散样本学习和发现嵌入在高维空间中的低维光滑流形，从而揭示隐藏在高维数据中的内在低维结构，以实现非线性降维或者可视化。

### 1.2.1流形学习的研究背景

随着互联网时代的发展，在各个研究领域，每时每刻都在快速生成大量的数据，而在这些数据背后的规律却难以发现，人们虽然获取了海量的信息却发现自己正处于“数据丰富，知识匮乏”的尴尬境地，因此如何从海量信息中提取自己所需的知识是当今各个领域共同面临的巨大挑战。

在许多实际应用中，尤其是在人脸识别中，往往要对成千上万张图片进行处理，而每张图片又有极高的维度，这种高维的特质往往隐藏了数据间关系的本质，对于传统的数据分析方法往往会造成“维数灾难”，这种情况就需要通过降维来把高维空间的数据间的关系映射到低纬度空间，这样就可以更加方便快速的处理数据。因此降维就成了这一任务吸引了许多科研人员的注意，也成为了如人脸识别、机器学习和数据挖掘等领域的热门研究问题。而流形学习就是解决这类问题的方案之一。

自从流形学习方法被提出到现在，研究它的工作就一直在进行着，特别是近年来随着数据挖掘和人脸识别等技术的高速发展，“维度灾难”的问题就成了相关研究领域的重大障碍。本文将利用流行学习方法解决这类问题。

### 1.2.2流形学习的发展情况

1984年斯坦福大学统计系的Hastie在一份技术报告中首次提出主曲线和主曲面的概念。1995年Bregler和Omohundro首次提出“流形学习”的概念，主要应用于图像插值和语音识别中。2000年Seung和Lee在《科学》杂志上发表《认知的流行模式》一文，提出了视觉感知的流行结构假说。就在同年同期的《科学》上还刊登了另外两篇著名的文章，它们提出了两个经典算法LLE和Isomap。随着后期学者不断地深入研究，又有许多经典算法相继被提出，比如Belkin和Niyogi提出拉普拉斯特征映射算法（LE），使高维空间相近的点映射到低微空间时也相近；Donoho和Grimes提出海森特征映射（HLLE），是对LLE算法的扩展；He和Hiyogi提出局部保持投影算法（LPP），是对LE算法的线性扩展；Zhang和Zha提出局部切空间校准算法（LTSA），基于“局部拟合，全局整合”的思想；Lin和Zha提出黎曼流行（RML），利用局部黎曼正交坐标系将高维空间的数据映射到低微本质空间中去。这些经典算法被提出后，又有许多学者为了弥补这些算法的缺陷而相继提出了很多经典的改进算法。

此外，很多学者发现不同的流形学习算法之间存在着一定的联系，又提出了一些框架将多种流形学习算法纳入其中，比如经典的核主成分分析框架（KPCA）将MDS、LLE、LE、Isomap和谱聚类进行统一，GA框架在将LE、LLE、LPP和Isomap等流形学习方法纳入其中的同时又将PCA和LDA等传统线性降维方法统一进去。

本文将介绍LGE（线性图嵌入）框架，该框架由浙江大学计算机科学学院的蔡登教授等人开发，为基于图的子空间学习提供一般框架，该框架将LPP，NPE，IsoProjection，LSDA，MMP等流形学习算法进行统一，本文将在2.5节中详细介绍LGE算法的推导过程和算法实现流程。

### 1.2.3流形的定义

流形是局部具有欧几里得空间性质的空间，是欧几里得空间中的曲线、曲面等概念的推广。欧几里得空间就是最简单的流形的实例。地球表面这样的球面则是一个稍微复杂的例子。一般的流形可以通过把许多平直的片折弯并粘连而成。

流形在数学中用于描述几何形体，它们为研究形体的可微性提供了一个自然的平台。物理上，经典力学的相空间和构造广义相对论的时空模型的四维伪黎曼流形都是流形的实例。

拓扑流形的数学定义可以表述为：设M是豪斯多夫（Hausdorff）空间，若对任意一点，都有x在M中的一个邻域U同胚于m维欧几里得空间的一个开集，就称M是一个m维流形或 m维拓扑流形。

关于豪斯多夫（Hausdorff）空间，直观的理解：如果某空间中任两点可用开集合将彼此“豪斯多夫”开来，该空间就是“豪斯多夫”的。

## 1.3小结

本文主要用到NPE算法和IsoProjection算法并详细介绍算法实现原理，本文会在后面介绍正交的NPE和IsoProjection算法，并对比这些算法的优缺点。

# 2流形学习相关算法

NPE和IsoProjection算法的实现需要用到LGE算法，IsoProjection算法因为涉及最短路径选择所以用到迪杰斯特拉算法，而IsoProjection算法又是基于ISOMAP算法的改进，ISOMAP算法又涉及到MDS算法。通过LGE算法构造OLGE算法要用到规范正交基。

## 2.1规范正交基

定义：设是一个向量空间。

（1）若是向量空间V的一个基，且是两两正交的向量组，则称是向量空间V的正交基。

（2）若是向量空间V的一个基，两两正交，且都是单位向量，则称是向量空间V的一个规范正交基（或标准正交基）。

求法：设是向量空间V的一个基，要求V的一个规范正交基，也就是要找一组两两正交的单位向量，使与等价。这一过程称为把基规范正交化。

可以先进行施密特正交化在进行单位化。本节接下来将详细介绍施密特正交化和单位化的计算方法。

### 2.1.1施密特(Schmidt)正交化

令：

；

;

……

，

则易验证两两正交，且与等价。且满足：对任何，向量组与等价。

### 2.1.2单位化

令：

则是V的一个规范正交基。

施密特正交化过程中可将中的任一线性无关的向量组化为与之等价的正交向量组；在经过单位化，得到与等价的规范正交向量组。

## 2.2迪杰斯特拉(Dijkstra)算法

迪杰斯特拉(Dijkstra)提出了一个按路径长度递增的次序产生最短路径的算法。该算法是典型最短路径算法，用于计算一个节点到其他节点的最短路径。

它的主要特点是以起始点为中心向外层层扩展 (广度优先搜索思想)，直到扩展到终点为止。

### 2.2.1迪杰斯特拉(Dijkstra)算法原理

1. 通过 Dijkstra 计算图 G 中的最短路径时，需要指定起点 s（即从顶点 s 开始计算）。
2. 此外，引进两个集合 S 和 U。S 的作用是记录已求出最短路径的顶点 (以及相应的最短路径长度)，而 U 则是记录还未求出最短路径的顶点 (以及该顶点到起点 s 的距离)。
3. 初始时，S 中只有起点 s；U 中是除 s 之外的顶点，并且 U 中顶点的路径是“ 起点 s 到该顶点的路径”。然后，从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。 然后，再从 U 中找出路径最短的顶点，并将其加入到 S 中；接着，更新 U 中的顶点和顶点对应的路径。重复该操作，直到遍历完所有顶点。

### 2.2.2迪杰斯特拉(Dijkstra)算法步骤

1. 初始时，S 只包含起点 s；U 包含除 s 外的其他顶点，且 U 中顶点的距离为“起点 s 到该顶点的距离”。例如，U 中顶点 v 的距离为 的长度，然后 s 和 v 不相邻，则 v 的距离为∞。
2. 从 U 中选出“ 距离最短的顶点 k”，并将顶点 k 加入到 S 中；同时，从 U 中移除顶点 k。
3. 更新 U 中各个顶点到起点 s 的距离。之所以更新 U 中顶点的距离，是由于上一步中确定了 k 是求出最短路径的顶点，从而可以利用 k 来更新其它顶点的距离；例如，的距离可能大于 的距离。
4. 重复步骤 2 和3，直到遍历完所有顶点。

## 2.3 多维缩放(MDS)算法

MDS 与 PCA 一样，是一种有效的降维方式，其可获得样本间相似性的空间表达。MDS 的原理可以简述为，利用样本的成对相似性，构建一个低维空间，使每对样本在高维空间的距离与在构建的低维空间中的样本相似性尽可能保持一致。

### 2.3.1 MDS算法推导

MDS 算法的核心思想是：降维前后，各自样本间的距离是不变的。由此可得到如下关系：假设m个样本在原始空间的距离矩阵为，则原空间中的两样本之间的距离等于降维后这两样本之间的距离 ，即，B为降维后样本的内积矩阵：

Z就是样本集在降维后空间的坐标矩阵。

所以，根据降维前后各样本间欧式距离保持不变，得：

假设降维后的样本集Z被中心化，即，则矩阵B的每行之和均为零，每列之和均为零，即：

为表示方便将dist简写为d，则有：

其中tr表示矩阵的迹。

令：

代入得：

现在的问题是，已知B如何求Z。

对B做特征值分解：

这里V是特征向量矩阵、A是由特征值构成的对角阵。此时，把A中的特征值排序后，把其中每个非0特征值拿出来构成对角矩阵，其对应的特征向量也需按特征值的大小改变排列顺序，组成新的特征向量矩阵。最终，通过以下公式完成降维操作（降到d维）：

另外，在现实应用中为了有效降维，往往只需要降维后两样本间的距离应尽可能和原空间中两样本的距离相近就好了，不需要强行一致，因此上面特征值构成的对角阵和特征向量矩阵有了一些变化。

变化为：本来特征值取的是所有非0的特征值排序。现在变成排序好后，从大到小取特征值获得特征值对角矩阵与其对应的特征向量矩阵。比如要降维到n维，就从大到小取n个特征值。则通过以下公式完成降维操作（降到n维）：

### 2.3.2 MDS算法流程

输入：，低维空间数d。

过程：

根据，和计算，和。

根据计算矩阵B。

根据对矩阵B进行特征值分解。

取n个最大的特征值构成对角矩阵，并获得特征向量矩阵。

根据完成降维，获得低维空间中的矩阵Z。

输出：低维空间中的矩阵Z。

## 2.4 等度量映射(ISOMAP)算法

Isomap 通过 “改造一种原本适用于欧氏空间的算法”，达到了“将流形映射到一个欧氏空间” 的目的。

Isomap 所改造的这个方法是 MDS，它的目的就是使得降维之后的点两两之间的距离尽量不变。只是 MDS 是针对欧氏空间设计的，对于距离的计算也是使用欧氏距离来完成的。如果数据分布在一个流形上的话，欧氏距离就不适用了。

Isomap 把 MDS 中原始空间中距离的计算从欧氏距离转换为了流形上的测地距离。当然，如果流形的结构事先不知道的话，这个距离是没法算的，于是 Isomap 通过将数据点连接起来构成一个邻接 Graph 来离散地近似原来的流形，而测地距离也相应地通过 Graph 上的最短路径来近似了。



所以低维嵌入流形上的测地线距离（红色）不能用高维空间的直线距离计算，但能用近邻距离来近似。

TODO：参考论文，再加一个ISOMAP算法步骤。

### 2.4.1 ISOMAP算法步骤

ISOMAP算法分为三步：

1. 构造近邻连接图。利用流形与欧氏空间在局部上同胚的性质，使用欧氏距离找出每个点在低维流形上的近邻点，建立近邻连接图。
2. 计算测地距离。计算近邻连接图上任意两点的最短路径。
3. 数据嵌入。使用MDS算法。

### 2.4.2 ISOMAP算法流程

输入：样本集；近邻参数k；低维空间维数d。

过程：

确定的k近邻；

与k近邻之间的距离设 置为欧式距离，与其他点的距离设置为无穷大；

调用最短路径算法（迪杰斯特拉算法）计算任意两样本点之间的距离；

将作为MDS算法的输入；

MDS算法的输出；

输出：样本集D在低维空间的投影。

### 2.4.3 ISOMAP算法的优点

ISOMAP算法最主要的优点就是使用 “测地距离”，而不是使用原始的欧几里得距离，这样可以更好的控制数据信息的流失，能够在低维空间中更加全面的将高维空间的数据表现出来。

### 2.4.4 ISOMAP算法的缺点

虽然相对于MDS算法而言，Isomap算法可以更好的保持和表示数据内部的几何结构，但Isomap算法还存在着一些不足之处，其中最严重的一点是算法本身的拓扑不稳定性，即Isomap算法在图中可能会建立错误的连接，这将严重的影响算法的执行。其次，Isomap算法针对的是凸形的流形，对于非凸形的流形将不能处理。再者，对于流形中含有“孔洞”的情况，一般的Isomap算法也不能很好的处理。

## 2.5线性图嵌入算法(LGE)

为基于图的子空间学习提供一般框架。该算法将由NPE和IsoProjection调用。

### 2.5.1 LGE算法推导

数据集，n表示样本数量，m表示样本维度。LGE算法采用无向有权图描述数据集的流形结构，顶点集对应数据，，表示边的权重，W矩阵是对称的。LGE算法在保持图的邻接关系的前提下，寻找X的低维表示。令表示从图到实线的映射，则LGE的目标函数如下：

变换后可得：

L是图的拉普拉斯矩阵，D是对角矩阵，对应W矩阵第列（行）所有元素之和，即。

令（消除嵌入时的量化影响），则：

若图到实线的映射为线性，则，则：

则最优向量对应于以下最小特征值所对应的特征向量。

矩阵通常是奇异的，所以LGE算法要先把X集投影到PCA空间，使变成可逆矩阵，然后求解特征值问题。经过PCA处理后上式可通过奇异值分解进行求解：

可解出，设为PCA空间特征向量，则LGE算法的解为：

矩阵是非对称的，所以LGE算法的解往往是非正交的。本节会在接下来介绍正交的LGE算法(OLGE)。

许多常用的线性子空间算法如LPP等，都可以通过定义其权重矩阵W将该算法统一于LGE框架下，通过进行求解。

TODO：参考论文：直接线性图嵌入算法及其人脸识别中的应用

## 2.6局部线性嵌入算法（LLE）

LLE是流形学习方面经典的局部非线性方法，它有参数少、计算快、易求全局最优并在图像分类、图像识别、谱重建、数据可视化等方面都有着广泛的应用。

### 2.6.1 LLE算法原理

矩阵是数据矩阵，包括所有的训练样本 。维度通常是很大的，（稀疏）线性降维的目标是变换数据从原来的高维空间到低维空间。

，，并且是一个m维列向量。

LLE试图保留邻域内样本之间的线性关系。在假设相邻样本位于非线性流形的局部线性片上时，每个样本由其k个最近邻居的加权线性组合近似表示。最小化如下成本函数：

是的k个最近邻居索引集，是最佳局部最小二乘重建系数。得到W后，可以通过最小化一下函数来得到最终嵌入坐标。

。对应于特征函数的较小特征值的特征向量是数据集的最终嵌入。

其中是对应于特征值的特征向量。

### 2.6.2 LLE算法推导

### 2.6.3 LLE算法流程

输入：数据集X，邻居个数k，降至的维度d。

第一步：计算对应距离和寻找邻居。

先计算向量到的距离，得到距离矩阵distance，然后每行排序，取前k个，对应原下标就是近邻点的编号。

第二步：解决重建权重问题。

对于任意i，若j不属于，则，故W只需存K行N列。

当且仅当每一项求和项极小时，极小。因此需要计算每一项的极小值，也就是极小化。

第三步：从成本矩阵的特征嵌入计算。

第四步：嵌入计算。

输出：降维后的矩阵Y。

## 2.7等距映射算法（IsoProjection）

IsoProjection是ISOMAP的线性近似，

TODO：简介，

### 2.7.1 IsoProjection算法推导

设为输入数据集，其中，寻找投影矩阵W使得，其中。Y就是降维后的数据集。

定义一个距离矩阵D，表示到之间的测地距离，定义矩阵S令，令，I为单位矩阵，E为元素全为1的列向量，则可得内积矩阵：。则IsoProjection的目标函数如下：

使用拉格朗日算法将上式的解向量问题转化为求解下式的特征值和特征向量的问题。

所以要求是非奇异矩阵，因此IsoProjection的最佳投影向量就是上式最大特征值所对应的特征向量。

IsoProjection算法求出来的投影向量往往是非正交的，这种非正交的性质会在从高维空间向低维空间的投影过程中扰乱数据的流形结构，而且会使算法本身对降维后的子空间的维数十分敏感，很难估计样本的内蕴维数。

### 2.7.2 IsoProjection算法流程

IsoProjection算法分五步执行：

1. 先使用PCA算法对样本进行降维。因为是奇异矩阵，所以要进行去奇异处理。使用PCA算法的主要目的是将变为非奇异矩阵，然后在低维空间使用IsoProjection算法。
2. 使用K-近邻方法构造近邻图G。
3. 使用迪杰斯特拉算法计算最短路径矩阵D。
4. 计算最佳投影矩阵W。
5. 特征提取。令，则。

IsoProjection算法核心源码参见附录6.1.1。

## 2.8领域保护嵌入算法（NPE）

TODO：简介。参考论文：

### 2.8.1 NPE算法推导

令NPE算法的权重矩阵为S，w为一投影向量，使得满足。则NPE算法的目标函数如下：

对该目标函数做出如下推导：

上式中，令。则简化后的目标函数如下：

则NPE算法的最佳投影向量就是以下特征方程的最小特征值所对应的特征向量。

### 2.8.2 NPE算法流程

NPE算法的实现分为三步：

1. 构造近邻图。使用K-近邻法寻找与数据点欧氏距离最近的个近邻点。
2. 确定权值。用近邻对各个数据点进行重构。
3. 计算特征映射。在低维空间中，对各个进行数据点重构，即在保持重构权值不变的情况下，使重构误差最小，计算出降维矩阵。

NPE算法核心源码参见附录6.1.2。

# 3正交化算法

## 3.1 OIsoProjection

OIsoProjection算法是在IsoProjection算法的基础上，得到一组正交基向量，因此该算法在保留IsoProjection算法线性的特点的同时又能够保持高维数据的流形结构。

OIsoProjection算法希望通过一组正交基改造目标函数，正交基的求法也相类似，都是通过拉格朗日乘子法引入正交基约束条件来推导出一个特征方程，然后求出该特征方程的最大特征值对应的特征向量以求出正交基，再利用该正交基构造特征方程进行特征提取。

### 3.1.1 OIsoProjection算法推导

首先要计算出一组正交基，则目标函数如下：

则向量可以通过求解特征方程的最大特征值所对应的特征向量获得。

若已有d-1个正交基向量。则可以利用如下正交约束条件获得：。

利用拉格朗日乘子法得到如下方程：

令：，则：

方程两边同时左乘：

得：

将(9.1.1.2)式方程两边同时左乘，得到d-1个式子：

为方便求解做出如下定义：

；

；

；

；

所以就可以用矩阵形式表示这个方程了：

得：

将(9.1.1.2)两边同乘，得：

矩阵表示：

由和得：

得：

得特征方程：

可通过求解上式最大特征值对应的特征向量得到正交基。

所以，OIsoProjection算法的正交投影向量就是特征方程的最大特征值所对应的特征向量。

### 3.1.2 OIsoProjection算法流程

OIsoProjection算法的前三步与IsoProjection算法的步骤相同，此处不再赘述。直接从第四步开始介绍：

4．计算正交投影向量。

通过5.2.1中的算法计算出一组正交基，具体流程如下：

(2) 通过求解特征方程的最大特征值所对应的特征向量获得计算。

(2) 通过求解特征方程的最大特征值所对应的特征向量获得，。

5. 特征提取。

令通过OIsoProjection算法得到的最佳投影矩阵为d维的，通过PCA算法得到的投影矩阵为，则令，则通过线性嵌入：

可以得到对任意x的d维特征。

## 3.2 ONPE

ONPE算法在NPE算法的基础上添加了正交迭代处理。改进的思路与OIsoProjection算法类似，算法的推导过程也与OIsoProjection算法类似。

### 3.2.1 ONPE算法推导

定义邻域保护函数为：

最小化该函数将得到NPE算法，而ONPE算法既要找到一组正交向量又要满足上式最小，因此ONPE算法的目标函数如下：

可以通过求解的最小特征对应的特征向量获得。

若已有k-1个正交向量，则第k正交向量满足以下条件：

约束条件：。

利用拉格朗日乘子法得到如下方程：

令，则：

得：

式两边同时左乘，得到k-1个式子：

其中。

为方便求解做如下定义：

；

；

；

所以就可以用矩阵形式表示这个方程了：

得：

式两边同时左乘，得：

通过式和式得：

令：

得特征方程：

通过求解这个特征方程就能得到一组正交投影向量。

所以，ONPE算法的正交投影向量就是特征方程的最大特征值所对应的特征向量。

### 3.2.2 ONPE算法流程

ONPE算法的前两步与NPE算法一样，这里从第三步开始介绍：

3.计算正交投影向量。通过5.2.1中的算法计算出一组正交投影向量。

1. 通过求解的最小特征对应的特征向量获得。
2. 通过求解的最小特征对应的特征向量获得。

4. 特征提取。

令通过ONPE算法得到的最佳投影矩阵为d维的，通过PCA算法得到的投影矩阵为，则令，则通过线性嵌入：

可以得到对任意x的d维特征。

# 4算法实现

NPE算法、ONPE算法、IsoProjection算法和OIsoProjection算法在ORL人脸库上的识别率。

## 4.1常用的人脸数据库介绍

使用公开人脸数据库对程序进行测评是评判算法好坏的主要依据。常用的人脸数据库有：FERET人脸库，由美国国防部Counterdrug技术开发项目办公室和美国国防部高级研究计划局(DARPA)发起为研究开发一个可以协助安全、情报和执法人员执行任务的自动的人脸识别系统而开发；Yale人脸库和Yale B人脸库，由耶鲁大学计算视觉与控制中心创建；ORL人脸库，由剑桥大学AT&T实验室创建；AR人脸库，由西班牙巴塞罗那计算机视觉中心创建；XM2VTS人脸库，由欧洲ACTS项目的研究计划M2VTS资助建立的身份认证资料数据库，是一个商业收费数据库；CMU PIE库，是卡耐基梅隆大学创建。

本文将在ORL人脸库和Yale人脸库上进行试验。

### 4.1.1 ORL人脸库

ORL人脸库由剑桥大学AT&T实验室创建，该库由40个人，每人10张图像，共400张图像组成。图像均在不同光照、不同角度和不同表情下获得。每张图片的分辨率均为112\times92像素。该库的面部表情表情变化主要是笑/不笑和睁眼/闭眼，还有一些其他的面部细节。拍摄的是受试者的正面、垂直头像，允许倾斜或旋转。



图 4-1

图4-1是ORL人脸库的示例图片。

人脸库下载：<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>

### 4.1.2 Yale人脸库

Yale人脸数据库是耶鲁大学计算视觉与控制中心创建的。该库由15个人的人脸图像组成，共有165张图像，每个人有11张图像，分别基于不同的光照变化、表情变化和姿态变化，如眨眼、正常、瞌睡、惊奇、悲伤和高兴，还有佩戴眼镜的人脸图像。



图 4-2

图4-1是Yale人脸库的示例图片。

人脸库下载：<http://cvc.yale.edu/projects/yalefaces/yalefaces.html>

## 4.2实验结果

分别运行NPE、ONPE、IsoProjection和OIsoProjection算法，并对比这些算法，分析这些算法的优缺点。

### 4.2.1 NPE

以下为NPE算法的运行结果：

每类训练样本数为：9

最佳投影维数为：26

NPE的识别率为：75.00%

程序运行时间为：37.76s

程序运行时间为：37.76s

可见NPE算法的识别率为75%左右。

### 4.2.2 ONPE

以下为ONPE算法的运行结果：

### 4.2.3 IsoProjection

以下为IsoProjection算法的运行结果：

每类训练样本数为：9

最佳投影维数为：27

IsoP的识别率为：85.00%

程序运行时间为：32.90s

可见IsoProjection算法的识别率为85%左右。

### 4.2.4 OIsoProjection

以下为OIsoProjection算法的运行结果：

### 4.3各个算法的性能对比

# 5结束语

# 6附录

## 6.1 主要算法的核心代码

详见中国浙江大学计算机科学学院的蔡登教授的个人主页：

<http://www.cad.zju.edu.cn/home/dengcai/Data/DimensionReduction.html>

### 6.1.1 IsoProjection算法

if options.k <= 0 % Always supervised!

Label = unique(options.gnd);

nLabel = length(Label);

G = zeros(nSmp,nSmp);

for i=1:nLabel

classIdx = find(options.gnd==Label(i));

D = EuDist2(data(classIdx,:),[],1);

G(classIdx,classIdx) = D;

end

maxD = max(max(G));

% effectively infinite distance

INF = maxD\*INFratio;

D = INF\*ones(nSmp,nSmp);

for i=1:nLabel

classIdx = find(options.gnd==Label(i));

D(classIdx,classIdx) = G(classIdx,classIdx);

end

Else

switch lower(options.NeighborMode)

case {lower('KNN')}

D = EuDist2(data);

maxD = max(max(D));

% effectively infinite distance

INF = maxD\*INFratio;

[dump,iidx] = sort(D,2);

iidx = iidx(:,(2+options.k):end);

for i=1:nSmp

D(i,iidx(i,:)) = 0;

end

D = max(D,D');

D = sparse(D);

D = dijkstra(D, 1:nSmp);

D = reshape(D,nSmp\*nSmp,1);

infIdx = find(D==inf);

D = reshape(D,nSmp,nSmp);

case {lower('Supervised')}

Label = unique(options.gnd);

nLabel = length(Label);

G = zeros(nSmp,nSmp);

maxD = 0;

for idx=1:nLabel

classIdx = find(options.gnd==Label(idx));

nSmpClass = length(classIdx);

D = EuDist2(data(classIdx,:),[],1);

if maxD < max(max(D))

maxD = max(max(D));

end

if options.k >= nSmpClass

G(classIdx,classIdx) = D;

else

[dump,iidx] = sort(D,2);

iidx = iidx(:,(2+options.k):end);

for i=1:nSmpClass

D(i,iidx(i,:)) = 0;

end

D = max(D,D');

D = sparse(D);

D = dijkstra(D, 1:nSmpClass);

G(classIdx,classIdx) = D;

end

end

% effectively infinite distance

INF = maxD\*INFratio;

D = INF\*ones(nSmp,nSmp);

for i=1:nLabel

classIdx = find(options.gnd==Label(i));

D(classIdx,classIdx) = G(classIdx,classIdx);

end

end

end

S = D.^2;

sumS = sum(S);

H = sumS'\*ones(1,nSmp)/nSmp;

TauDg = -.5\*(S - H - H' + sum(sumS)/(nSmp^2));

TauDg = max(TauDg,TauDg');

[eigvector,eigvalue]=LGE(TauDg, [], options, data);

eigIdx = find(eigvalue < 1e-3);

eigvalue (eigIdx) = [];

eigvector(:,eigIdx) = [];

### 6.1.2 NPE算法

if options.k <= 0 % Always supervised!

W = zeros(nSmp,nSmp);

for ii=1:nSmp

idx = find(options.gnd==options.gnd(ii));

idx(find(idx==ii)) = [];

% shift ith pt to origin

z=data(idx,:)-repmat(data(ii,:),length(idx),1);

C = z\*z'; % local covariance

% regularlization

C = C + eye(size(C))\*tol\*trace(C);

tW = C\ones(length(idx),1); % solve Cw=1

tW = tW/sum(tW); % enforce sum(w)=1

W(idx,ii) = tW;

end

M = (eye(size(W)) - W);

M = M\*M';

M = max(M,M');

M = sparse(M);

Else

switch lower(options.NeighborMode)

case {lower('KNN')}

Distance = EuDist2(data,[],0);

[sorted,index] = sort(Distance,2);

neighborhood = index(:,2:(1+options.k));

case {lower('Supervised')}

Label = unique(options.gnd);

nLabel = length(Label);

neighborhood = zeros(nSmp,options.k);

for idx=1:nLabel

classIdx=find(options.gnd==Label(idx));

Distance=EuDist2(data(classIdx,:),[],0);

[sorted,index] = sort(Distance,2);

neighborhood(classIdx,:) = classIdx(index(:,2:(1+options.k)));

end

end

W = zeros(options.k,nSmp);

for ii=1:nSmp

% shift ith pt to origin

z = data(neighborhood(ii,:),:)-repmat(data(ii,:),options.k,1);

C = z\*z'; % local covariance

%regularlization

C = C + eye(size(C))\*tol\*trace(C);

W(:,ii) = C\ones(options.k,1);% solve Cw=1

% enforce sum(w)=1

W(:,ii) = W(:,ii)/sum(W(:,ii));

end

M = sparse(1:nSmp,1:nSmp,ones(1,nSmp),nSmp,nSmp,4\*options.k\*nSmp);

for ii=1:nSmp

w = W(:,ii);

jj = neighborhood(ii,:)';

M(ii,jj) = M(ii,jj) - w';

M(jj,ii) = M(jj,ii) - w;

M(jj,jj) = M(jj,jj) + w\*w';

end

M = max(M,M');

M = sparse(M);

end

M = -M;

for i=1:size(M,1)

M(i,i) = M(i,i) + 1;

end

[eigvector, eigvalue] = LGE(M, [], options, data);

eigIdx = find(eigvalue < 1e-10);

eigvalue (eigIdx) = [];

eigvector(:,eigIdx) = [];

### 6.1.3 LGE算法

MAX\_MATRIX\_SIZE = 1600;

EIGVECTOR\_RATIO = 0.1;

% SVD

if bPCA

[U, S, V] = mySVD(data);

[U, S, V]=CutonRatio(U,S,V,options);

eigvalue\_PCA = full(diag(S));

if bD

data = U\*S;

eigvector\_PCA = V;

DPrime = data'\*D\*data;

DPrime = max(DPrime,DPrime');

else

data = U;

eigvector\_PCA = V\*spdiags(eigvalue\_PCA.^-1,0,length(eigvalue\_PCA),length(eigvalue\_PCA));

end

else

if ~bChol

if bD

DPrime = data'\*D\*data;

else

DPrime = data'\*data;

end

switch lower(options.ReguType)

case {lower('Ridge')}

if options.ReguAlpha > 0

for i=1:size(DPrime,1)

DPrime(i,i) = DPrime(i,i) + options.ReguAlpha;

end

end

case {lower('Tensor')}

if options.ReguAlpha > 0

DPrime = DPrime + options.ReguAlpha\*options.regularizerR;

end

case {lower('Custom')}

if options.ReguAlpha > 0

DPrime = DPrime + options.ReguAlpha\*options.regularizerR;

end

end

DPrime = max(DPrime,DPrime');

end

end

WPrime = data'\*W\*data;

WPrime = max(WPrime,WPrime');

% Generalized Eigen

dimMatrix = size(WPrime,2);

if ReducedDim > dimMatrix

ReducedDim = dimMatrix;

end

if isfield(options,'bEigs')

bEigs = options.bEigs;

else

if (dimMatrix > MAX\_MATRIX\_SIZE) && (ReducedDim < dimMatrix\*EIGVECTOR\_RATIO)

bEigs = 1;

else

bEigs = 0;

end

end

if bEigs

%disp('use eigs to speed up!');

option = struct('disp',0);

if bPCA && ~bD

[eigvector, eigvalue] = eigs(WPrime,ReducedDim,'la',option);

else

if bChol

option.cholB = 1;

[eigvector, eigvalue] = eigs(WPrime,R,ReducedDim,'la',option);

else

[eigvector, eigvalue] = eigs(WPrime,DPrime,ReducedDim,'la',option);

end

end

eigvalue = diag(eigvalue);

else

if bPCA && ~bD

[eigvector, eigvalue] = eig(WPrime);

else

[eigvector, eigvalue] = eig(WPrime,DPrime);

end

eigvalue = diag(eigvalue);

[junk, index] = sort(-eigvalue);

eigvalue = eigvalue(index);

eigvector = eigvector(:,index);

if ReducedDim < size(eigvector,2)

eigvector = eigvector(:, 1:ReducedDim);

eigvalue = eigvalue(1:ReducedDim);

end

end

if bPCA

eigvector = eigvector\_PCA\*eigvector;

end

for i = 1:size(eigvector,2)

eigvector(:,i) = eigvector(:,i)./norm(eigvector(:,i));

end

%function CutonRatio

function [U, S, V]=CutonRatio(U,S,V,options)

if ~isfield(options, 'PCARatio')

options.PCARatio = 1;

end

eigvalue\_PCA = full(diag(S));

if options.PCARatio > 1

idx = options.PCARatio;

if idx < length(eigvalue\_PCA)

U = U(:,1:idx);

V = V(:,1:idx);

S = S(1:idx,1:idx);

end

elseif options.PCARatio < 1

sumEig = sum(eigvalue\_PCA);

sumEig = sumEig\*options.PCARatio;

sumNow = 0;

for idx = 1:length(eigvalue\_PCA)

sumNow = sumNow + eigvalue\_PCA(idx);

if sumNow >= sumEig

break;

end

end

U = U(:,1:idx);

V = V(:,1:idx);

S = S(1:idx,1:idx);

end