表情特征信息的降维和提取

杨超群

摘要

在多元统计分析中, 主成分分析 (Principal components analysis, PCA) 是一种分析、简化数据集的技术.PCA 经常用于减少数据集的维数, 同时保持数据集中的对方差贡献最大的特征 (主成分).

本文先介绍了面部表情特征提取和 PCA 的背景, 并给出 PCA 算法的数学推导, 再对数据使用逻辑回归算法, 比较分析了使用 PCA 前后逻辑回归的各项指标, 得出了较好的结果.

1 背景与现状分析

1.1 面部表情特征提取

心理学家拉塞尔发现，在人们的日常生活中，通过语言传播的信息只有 7%，通过声音传播的信息是 38%，大约 55%的信息是通过面部表情传播的。在这一理论的基础上，Paul Ekman 向前迈进了一步。他将人类的面部表情分为七类：快乐，悲伤，恐惧，愤怒，厌恶，惊讶和自然。这些表情被用来表达人类的心理状态。

面部表情是非语言交际中情感的重要表达。随着计算机的发展和应用，人脸表情识别在人机交互和医疗保健中具有重要的应用，已成为人工智能和计算机视觉领域的热门研究课题。但对于计算机而言，面部表情识别是一项复杂的任务。计算机识别面部表情过程分为三个步骤，图像预处理，特征提取和面部表情分类。如何有效地提取面部表情图像的特征是面部表情识别的关键步骤。早期的面部表情识别方法是通过设计特征提取算法来手动提取面部表情特征。这些方法包括基于特征点定位进行人脸建模的主动外观模型算法，还有基于局部特征的提取算法等。

表情识别技术是指利用计算机从面部表情中提取面部特征，根据人类的认知和思维进行分类和理解，然后从面部信息中分析和理解人的情感。它是机器视觉和人机互动的热点。

1.2 主成分分析（PCA）

主成分分析（Principal Component Analysis，简称 PCA）是一种减少数据维度和获取特征数据的方法。PCA 通过 KL 变换消除了原始数据之间的相关性，减少了信息冗余量，达到了减小特征空间维度的目的。

PCA 算法的优点是不需要根据经验设置模型或参数，结果仅仅与数据本身有关。PCA 是一种简化数据集的技术。它是一个线性变换。这个变换把数据变换到一个新的坐标系统中，使

得任何数据投影的第一大方差在第一个坐标 (称为第一主成分) 上，第二大方差在第二个坐标 (第二主成分) 上，依次类推。主成分分析经常用减少数据集的维数，同时保持数据集的对方差贡献最大的特征。这是通过保留低阶主成分，忽略高阶主成分做到的。这样低阶成分往往能够保留住数据的最重要方面。但是，这也不是一定的，要视具体应用而定。

概括起来说，主成分分析主要有以下几个方面的作用。

- 主成分分析能降低所研究的数据空间的维数。即研究 m 维的 Y 空间来代替 p 维的 X 空间 (m p)，而低维的 Y 空间代替高维的 X 空间所损失的信息很少。即：使只有一个主成分 Y_1(即 m 1) 时，这个 Y_1 仍是使用全部 X 变量 (p 个) 得到的。例如要计算 Y_1 的均值也得使用全部 Y_1 的均值。在所选的前 m 个主成分中，如果某个 X_i 的系数全部近似于零的话，就可以把这个 X_i 删除，这也是一种删除多余变量的方法。
- 有时可通过因子负荷的相关结论，弄清 X 分量间的某些关系。
- 多维数据的一种图形表示方法。我们知道当维数大于 3 时便不能画出几何图形，多元统计研究的问题大都多于 3 个变量。要把研究的问题用图形表示出来是不可能的。然而，经过主成分分析后，我们可以选取前两个主成分或其中某两个主成分，根据主成分的得分，画出 n 个样品在二维平面上的分布况，由图形可直观地看出各样品在主分量中的地位，进而还可以对样本进行分类处理，可以由图形发现远离大多数样本点的离群点。
- 由主成分分析法构造回归模型。即把各主成分作为新自变量代替原来自变量 x 做回归分析。
- 用主成分分析筛选回归变量。回归变量的选择有着重要的实际意义，为了使模型本身易于做结构分析、控制和预报，好从原始变量所构成的子集合中选择最佳变量，构成最佳变量集合。用主成分分析筛选变量，可以用较少的计算量来选择变量，获得选择最佳变量子集合的效果。

2 PCA 算法的数学推导 [?]

假设在 $\mathbb{R}^n$ 空间中有 m 个点 $\{\boldsymbol{x}^{(1)}, \cdots, \boldsymbol{x}^{(m)}\}$, 我们希望在精度损失尽可能少的情况下, 使用更少的内存去储存这些点. 如果对于每一个点 $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$, 有一个对应的编码向量 $\boldsymbol{c}^{(i)} \in \mathbb{R}^l$. 如果 $l < n$, 那么就使用了更少的内存来储存原来的数据. 通过找到一个编码函数, 根据输入返回编码, $f(\boldsymbol{x}) = \boldsymbol{c}$; 再找到一个解码函数, 给定编码重构输入, $\boldsymbol{x} \approx g(f(\boldsymbol{x})) = g(\boldsymbol{c})$, 使 x 与 $g(\boldsymbol{c})$ 之间的距离最小.

为了简化解码器, 可以使用矩阵乘法将编码映射回 $\mathbb{R}^n$, 即 $g(\boldsymbol{c}) = \boldsymbol{D}\boldsymbol{c}$, 其中 $\boldsymbol{D} \in \mathbb{R}^{n \times l}$ 是定义解码的矩阵. 为了使问题有唯一解, 需要限制 $\boldsymbol{D}$ 中所有列向量都有单位范数. 为了使编码问题简单一些, PCA 限制 $\boldsymbol{D}$ 的列向量彼此正交.

在 PCA 算法中, 使用 L^2 范数作为距离的度量,

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \|\boldsymbol{x} - g(\boldsymbol{c})\|_2 \tag{1}$$

因为平方 L^2 范数和 L^2 范数在相同的 $\boldsymbol{c}$ 上取得最小值, 故上式可变为

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \|\boldsymbol{x} - g(\boldsymbol{c})\|_2^2 \tag{2}$$

即

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} (\boldsymbol{x} - g(\boldsymbol{c}))^{'} (\boldsymbol{x} - g(\boldsymbol{c})) \tag{3}$$

将等式右边化简得

$$\boldsymbol{x}^{'}\boldsymbol{x} - 2\boldsymbol{x}^{'} g(\boldsymbol{c}) + g(\boldsymbol{c})^{'} g(\boldsymbol{c}) \tag{4}$$

因为 $\boldsymbol{x}^{'}\boldsymbol{x}$ 与 $\boldsymbol{c}$ 无关, $g(\boldsymbol{c}) = \boldsymbol{D}\boldsymbol{c}$ 且 $\boldsymbol{D}$ 的列向量彼此正交, 故

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} \boldsymbol{c}^{'}\boldsymbol{D}^{'}\boldsymbol{D}\boldsymbol{c} - 2\boldsymbol{x}^{'}\boldsymbol{D}\boldsymbol{c} \tag{5}$$

$$= \arg\min_{\boldsymbol{c}} \boldsymbol{c}^{'}\boldsymbol{c} - 2\boldsymbol{x}^{'}\boldsymbol{D}\boldsymbol{c} \tag{6}$$

为解得 $\boldsymbol{c}$, 对 $\boldsymbol{c}$ 求导并使导数为 0 得

$$\nabla_{\boldsymbol{c}}\big(\boldsymbol{c}^{'}\boldsymbol{c} - 2\boldsymbol{x}^{'}\boldsymbol{D}\boldsymbol{c}\big) = 0 \tag{7}$$

$$2\boldsymbol{c} - 2\boldsymbol{D}^{'}\boldsymbol{x} = 0 \tag{8}$$

$$\boldsymbol{c} = \boldsymbol{D}^{'}\boldsymbol{x} \tag{9}$$

所以编码函数为

$$f(\boldsymbol{x}) = \boldsymbol{D}^{'}\boldsymbol{x} \tag{10}$$

PCA 重构操作为

$$r(\boldsymbol{x}) = g(f(\boldsymbol{x})) = \boldsymbol{D}\boldsymbol{D}^{'}\boldsymbol{x} \tag{11}$$

再来确定编码矩阵 $\boldsymbol{D}$, 使所有维数和所有点上的误差矩阵的 Fribenius 范数最小

$$\boldsymbol{D}^* = \arg\min_{\boldsymbol{D}} \sqrt{\sum_{i,j} \big(\boldsymbol{x}_j^{(i)} - r(\boldsymbol{x}^{(i)})_j\big)^2}, \boldsymbol{D}^{'}\boldsymbol{D} = \boldsymbol{I}_l \tag{12}$$

先考虑 $l = 1$ 的情况, 此时 $\boldsymbol{D}$ 为向量, 记为 $\boldsymbol{d}$, 则上式化简为

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i \|\big(\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\boldsymbol{d}\boldsymbol{d}^{'}\big)\|_2^2, \|\boldsymbol{d}\|_2 = 1 \tag{13}$$

记 $\boldsymbol{X}$ 为各个点堆叠成的矩阵, 其中 $\boldsymbol{X}_{i,:} = \boldsymbol{x}^{(i)^{'}}$, 则上式可化为

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\|_F^2, \|\boldsymbol{d}\|_2 = 1 \tag{14}$$

因为

$$\arg\min_{\boldsymbol{d}} \|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\|_F^2 \tag{15}$$

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\Big(\big(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big)^{'}\big(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big)\Big) \tag{16}$$

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X} - \boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'} - \boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X} + \boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) \tag{17}$$

(因为 $\boldsymbol{X}^{'}\boldsymbol{X}$ 与 $\boldsymbol{d}$ 无关)

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) - \mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) - \mathrm{Tr}\big(\boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\big) \tag{18}$$

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) - 2\mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) \tag{19}$$

(循环改变迹运算中相乘矩阵的顺序不影响结果)

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\boldsymbol{d}\boldsymbol{d}^{'}\big) - 2\mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) \tag{20}$$

(因为 $\boldsymbol{d}^{'}\boldsymbol{d} = 1$)

$$= \arg\min_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big) - 2\mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big), \boldsymbol{d}^{'}\boldsymbol{d} = 1 \tag{21}$$

$$= \arg\min_{\boldsymbol{d}} -\mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big), \boldsymbol{d}^{'}\boldsymbol{d} = 1 \tag{22}$$

$$= \arg\max_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^{'}\big), \boldsymbol{d}^{'}\boldsymbol{d} = 1 \tag{23}$$

$$= \arg\max_{\boldsymbol{d}} \mathrm{Tr}\big(\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}\big), \boldsymbol{d}^{'}\boldsymbol{d} = 1 \tag{24}$$

(因为 $\boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d} \in \mathbb{R}$)

$$= \arg\max_{\boldsymbol{d}} \boldsymbol{d}^{'}\boldsymbol{X}^{'}\boldsymbol{X}\boldsymbol{d}, \boldsymbol{d}^{'}\boldsymbol{d} = 1 \tag{25}$$

所以最优的 $\boldsymbol{d}$ 是 $\boldsymbol{X}^{'}\boldsymbol{X}$ 最大特征值对应的特征向量.

对于 $l = 1$, 以上推导得到了第一个主成分. 更一般的, 对于 $1 < l < n$, 可由归纳法得出矩阵 $\boldsymbol{D}$ 由 $\boldsymbol{X}^{'}\boldsymbol{X}$ 的前 l 个最大的特征值对应的特征向量组成.

3 在 MATLAB 上通过 PCA 优化表情识别

3.1 简要过程

本项目[1] 的人脸表情识别算法采用的是前馈神经网络, 通过反向传播算法优化权重矩阵, 神经网络尺寸为 $n \times 25 \times 5$, 有一个中间层, 其中 n 由输入数据维度决定, 输出为一 5 维的单位向量, 根据其单位 1 出现位置, 分别表示惊讶、微笑、愤怒、失望和中性. PCA 的实现利用了 MATLAB 的内置函数 svd.

[1] 由于所用代码较多, 在此只列出部分.

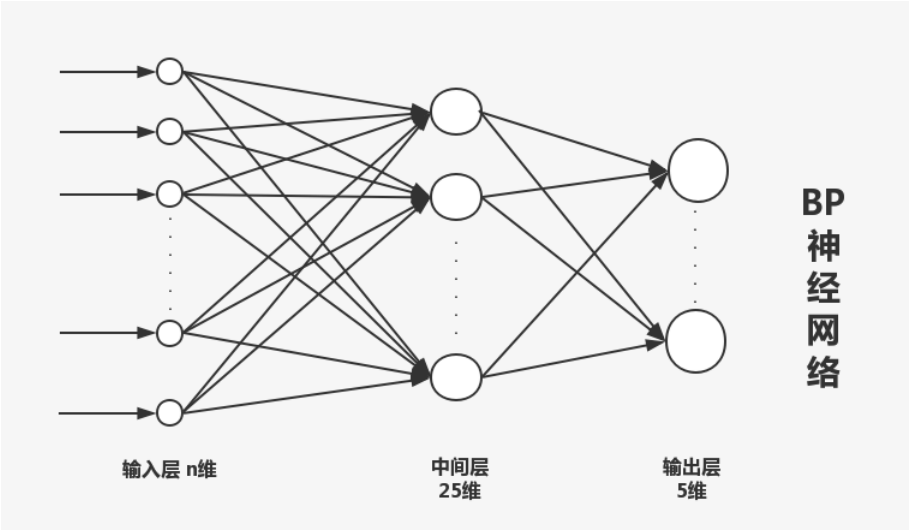


图 1: 神经网络结构

本项目的数据[2] 为一 1965×560 的矩阵, 代表 1965 个实例, 每个实例是一 560 维向量, 另有一 1965×1 的向量, 其中各个元均为一 1 到 5 的整数, 分别代表其对应实例的具体表情——惊讶、微笑、愤怒、失望和中性. 先对数据进行特征标准化, 使数据具有 0 均值和单位方差, 以便于进行 PCA 处理和加速神经网络的权重矩阵的训练,

```
function [X_norm, mu, sigma] = featureNormalize(X)

%FEATURENORMALIZE Normalizes the features in X
%   FEATURENORMALIZE(X) returns a normalized version of X where
%   the mean value of each feature is 0 and the standard deviation
%   is 1.
mu = mean(X);
X_norm = bsxfun(@minus, X, mu);

sigma = std(X_norm);
X_norm = bsxfun(@rdivide, X_norm, sigma);
end
```

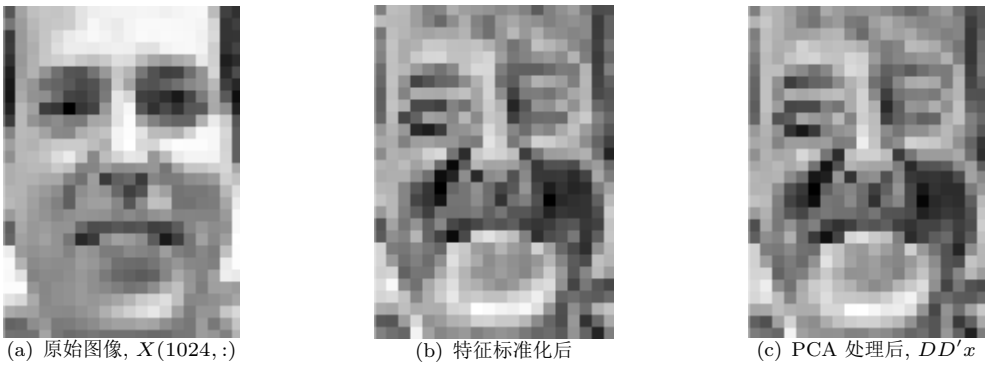
若不经过 PCA 的降维处理, 神经网络的尺寸为 560×25×5; 在讯息损失不超过 1% 的情况下, 经 PCA 处理后神经网络的尺寸降为 226×25×5, 输入维度变为原来的 40.36%, 大大减少了神经网络的复杂度.

```
function [U, S] = pca(X)
%   PCA Run principal component analysis on the dataset X
%   [U, S] = pca(X) computes eigenvectors of the covariance matrix of X
```

[2]数据集来自https://cs.nyu.edu/~roweis/data.html, Faces, Frey Face, From Brendan Frey. Almost 2000 images of Brendan's face, taken from sequential frames of a small video. Size: 20 × 28.

```
4  %   Returns the eigenvectors U, the eigenvalues (on diagonal) in S
5
6       [m, n] = size(X);
7
8       U = zeros(n);
9       S = zeros(n);
10      sigma = X'*X/m;
11      [U, S, ~] = svd(sigma);
12  end
```

对比同一数据在三个阶段的图片，可以看出即使数据维度降低过半, PCA 处理后图片仍与处理前几乎无差别.



(a) 原始图像, $X(1024,:)$ (b) 特征标准化后 (c) PCA 处理后, $DD'x$

图 2

将处理后所得数据随机分成比例为 6:2:2 的训练集、交叉验证集、测试集, 使用训练集训练神经网络的权重矩阵. 使用交叉熵损失函数, 为防止出现过拟合, 引入 $L2$ 正则, 加入惩罚参数 λ, 选取范围为 0.01 到 1.8, 步长为 0.01. 对每个 λ, 将训练好的神经网络的权重矩阵用于预测交叉验证集的表情, 选择正确率最高的一组权重矩阵, 再用这组权重矩阵预测测试集所对应的表情, 得到最终正确率.

在使用 PCA 的算法中, 需先对训练集进行 PCA 处理, 将所得的特征向量作用于训练集、交叉验证集、测试集, 然后进行训练、预测.

3.2 结果比较

下图是未使用 PCA 的运行摘要[3], 列出了主要函数的运行情况, 运行总耗时为 2320.821 秒, 在 $\lambda = 0.45$ 时在交叉验证集上的正确率最大, 在测试集上的正确率为 97.201018%,

[3]由于权重矩阵需先进行随机初始化, 故每次运行的结果会有微小的差异, 但使用 PCA 的结果都显著优于不使用 PCA 的结果. 这里是任意选择的一组进行比较.

函数名称	调用次数	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
face	1	2320.821 s	1.347 s	
fmincg	180	2318.732 s	297.268 s	
...ze,num_labels,X_train,y_train,lambda)	271430	2021.464 s	10.291 s	
nnCostFunction	271507	2011.189 s	1837.198 s	
sigmoid	543376	174.039 s	174.039 s	

图 3: 无 PCA 处理的运行摘要

下图是使用 PCA 后的运行摘要, 运行总耗时为 1438.173 秒, $\lambda = 0.41$ 时在交叉验证集上的正确率最大, 在测试集上的正确率为 99.745547% .

函数名称	调用次数	总时间	自用时间*	总时间图 (深色条带 = 自用时间)
face_pca	1	1438.173 s	1.029 s	
fmincg	180	1436.392 s	258.853 s	
...ze,num_labels,X_train,y_train,lambda)	268884	1177.540 s	9.492 s	
nnCostFunction	268961	1168.063 s	1006.537 s	
sigmoid	538284	161.574 s	161.574 s	

图 4: PCA 处理后的运行摘要

比较两篇运行摘要可以明显看出, 使用 PCA 后运行时间大幅减少, 后者为前者的 61.97%, 而且精确度在 97.201018% 的基础上提高到了 99.745547%. 这显示出使用 PCA 后前馈神经网络对人脸表情识别效率的巨大提升, 达到了本项目的实验目的.