MG1833027 贺鑫 计算机系18级研究生

6.6 (PCA + FLD face recognition)

ORL dataset

1. 每张图片为92×112 分辨率的PGM灰度图像；
2. 规模：40个不同人，每人10张不同的图片；如改变光照、表情(睁眼、闭眼、微笑、扬眉)、脸部细节((不)带眼镜)；
3. 数据集文件夹层次结构：以$s_{1,2,...,40}$表示40个拍摄对象(目录)，s_i目录中包含$\{1.pgm, ..., 10.pgm\}$10张`pgm`图片；

Analyze the differences between recognition results of Eigenfaces and Fisherfaces

安装好opencv4之后，从OpenCV facerec_tutorial 下载并适当修改代码后，运行得到Eigenfaces的10个eigenvalue为：

```
EigenValue #0 = 2817599.36867
EigenValue #1 = 2069666.47426
EigenValue #2 = 1093913.87096
...
EigenValue #10 = 284752.77761
```

Fisherfaces的10个eigenvalue为：

```
EigenValue #0 = 19882.09108
EigenValue #1 = 11644.54238
...
EigenValue #10 = 36.57706
```

可以看出Eigenfaces前10个奇异值相差只有10倍，而Fisherfaces映射矩阵前10个奇异值相差相当大；即Eigenfaces的PCA降维保留了很多与人脸识别任务无关的特征(虚假特征)，保留了绝大部分图像信息，却没有降低人脸识别任务的难度；

注：无论是Eigenfaces还是Fisherfaces在背景色单一(如AT&T数据集orl_faces)，均能达到较高的识别准确度；

reconstruct face image

OpenCV tutorial 的source code中已经提供了reconstruct的代码($169-182$行)，最初10个奇异值重建，之后每次递增15个奇异值；可以看到用55个eigenfaces即可较清晰地重建原始图片；

> P.S. 建议更新讲义Fisher's linear discriminant习题`6(c)`关于facerec_tutorial的链接为Face Recognition with OpenCV[1]。理由如下:
>
> - OpenCV已更新到4.1.0，且OpenCV 3重构了OpenCV架构，重新封装了API接口，导致原链接中OpenCV 2.4 的代码无法直接运行在OpenCV 3/4中；
> - 随着Ubuntu系统版本的更新，在高版本系统中安装OpenCV 2.4的教程很旧且容易出错；
> - 新链接可自行选择OpenCV 3/4版本，Ubuntu 16.04可自行安装OpenCV3，且Ubuntu 18.04中安装OpenCV4的教程大多可无缝迁移到Ubuntu 16.04中；
>
> 下面给出我在ubuntu 18.04平台从源码安装OpenCV 4用于C++、Python(Anaconda)编程的安装笔记，目前已共享到百度网盘[2]：

7.2 (Additive kernels)

(a) Histogram intersection kernel

方法1：(证明有限正半定性质)

因为$\kappa_{HI}(x,y)$ 是valid kernel，所以$\kappa_{HI}(x,y)$对应的kernel matrix为**半正定矩阵**；从**能量相加**角度可知，$\kappa_{HI}(\mathbf{x},\mathbf{y})=\sum_{i=1}^{d}\min(\mathbf{x}_i,\mathbf{y}_i)$对应的kernel matrix也是**半正定矩阵**；且$\kappa_{HI}(x,y)$为对称函数，当给定任意有限样本时，κ_{HI}满足有限半正定的性质，所以$\kappa_{HI}(\mathbf{x},\mathbf{y})$是合法的kernel函数；

方法2：characterization of kernels定理

由于$\kappa_{HI}(x,y)=\min(x,y)$是合法的核函数，所以存在$\phi(x)$使得$\kappa_{HI}(x,y)=<\phi(x),\phi(y)>$.

令$\phi(\mathbf{x})=(\phi(\mathbf{x}_1),\dots,\phi(\mathbf{x}_d)))$，

$\kappa_{HI}(\mathbf{x},\mathbf{y})=\sum_{i=1}^{d}\min(x_i,y_i)=\sum_{i=1}^{d}<\phi(x_i,y_i)=<\phi(\mathbf{x}),\phi(\mathbf{y})>$.

由于$\kappa_{HI}(\mathbf{x},\mathbf{y})$可写成feature mapping 函数的内积，所以$\kappa_{HI}(\mathbf{x},\mathbf{y})$是合法的kernel function.

(b) Prove that X is PSD iff Y is PSD.

由题意易知，矩阵X和Y经过若干次行变换、列变换后等价，所以X和Y相似，即X和Y具有相同的特征值，所以X和Y同正定；

(c) prove kernel matrix X is PSD and κ_{HI} is valid kernel for non-negative vectors

易知X可分解为：LDL^T.其中L为下三角阵，对角线元素为1，非对角线元素为-1.D为对角阵，沿对角线方向元素依次为$x_1,x_2,\dots,x_n$.所以X的特征值为$x_1,x_2,\dots,x_n$，由于$x_1,x_2,\dots,x_n$非负，所以X为半正定矩阵；

(d) prove $\kappa_{HI}(x,y)\leq\kappa_{\chi^2}(x,y)$.

即证$\sum_{i=1}^{d}\min(x_i,y_i)\leq\sum_{i=1}^{d}\frac{2x_iy_i}{x_i+y_i}$.

不失一般性，对于任意的$t\in[m]$, 假设$x_t=\min(x_t,y_t)$.

如果我们能证明$x_t=\min(x_t,y_t)\leq\frac{2x_ty_t}{x_t+y_t}$，即可递推到累和形式；

$\because y_t\geq x_t$，$\therefore\frac{2x_ty_t}{x_t+y_t}=\frac{2}{\frac{1}{x_t}+\frac{1}{y_t}}\geq x_t$. 证毕！

(e) prove HE kernel is a valid kernel and $\kappa_{HE}\geq\kappa_{\chi^2}$.

$\kappa_{HE}(x,y)=\sum_{i=1}^{d}\sqrt{x_iy_i}=<x^{\frac{1}{2}},y^{\frac{1}{2}}>=<\phi(x),\phi(y)>$. 易知$\phi(x)=x^{\frac{1}{2}}$.

由于K_{HE}可表示成feature mapping $\phi(x)$的内积，所以κ_{HE}的合法的核函数；

证明$\kappa_{HE}\geq\kappa_{\chi^2}$的思路与`(d)`完全一致，只需证明$\sqrt{x_ty_t}\geq\frac{2x_ty_t}{x_t+y_t}$.

$\because(x_t+y_t)^2\geq 4x_ty_t,\therefore x_t+y_t\geq 2\sqrt{x_ty_t}$.得证！

(f) write out an explicit mapping for histogram intersection kernel

我们还可以求出$\kappa_{HI}(x,y)$对应的一个feature mapping function $\phi(x)$.

下面从像素值直方图计算图像相似性的应用出发，阐述如何构造特征映射函数$\phi(x)$。

假设χ表示一幅数值图像，图像像素点总数均为m，即$\chi\in\mathcal{R}^m$。

现将图像中的像素点按像素值划分为d个bins，易知每个bins最大为m。统计图像对应每个bin的像素个数，计第i个bins中包含的像素点的个数为x_i，统计完后，将每个bin编码成长度为m的01向量，其中前x_i个二进制位为1，剩余的$m-x_i$个二进制位为0，则图片χ表示成d个长度为m的二值向量；容易证明两幅图片x、y第i个bin所对应的向量的内积即为$\min(x_i,y_i)$.

构造的过程即为特征映射$\phi(x)$。应用到natural numbers vectors中，只需将$m=\max\{num:num\in x\cup y\}$.

参考：F. Odone, A. Barla, and A. Verri, “Building kernels from binary strings for image matching,” IEEE Transactions on Image Processing, vol. 14, no. 2, pp. 169–180, 2005.

8.1 Find the maximum likelihood estimate for λ

由题意可知：对数似然函数$ll(\lambda)=\sum_{i=1}^{n}\ln(\lambda\exp(-\lambda x_i))=\sum_{i=1}^{n}[\ln(\lambda)-\lambda x_i]$.

令$\frac{\partial l(\lambda)}{\partial\lambda}=\sum_{i=1}^{n}[\frac{1}{\lambda}-x_i]=0$, 得：$\lambda=\frac{n}{\sum_{i=1}^{n}x_i}$.

8.5

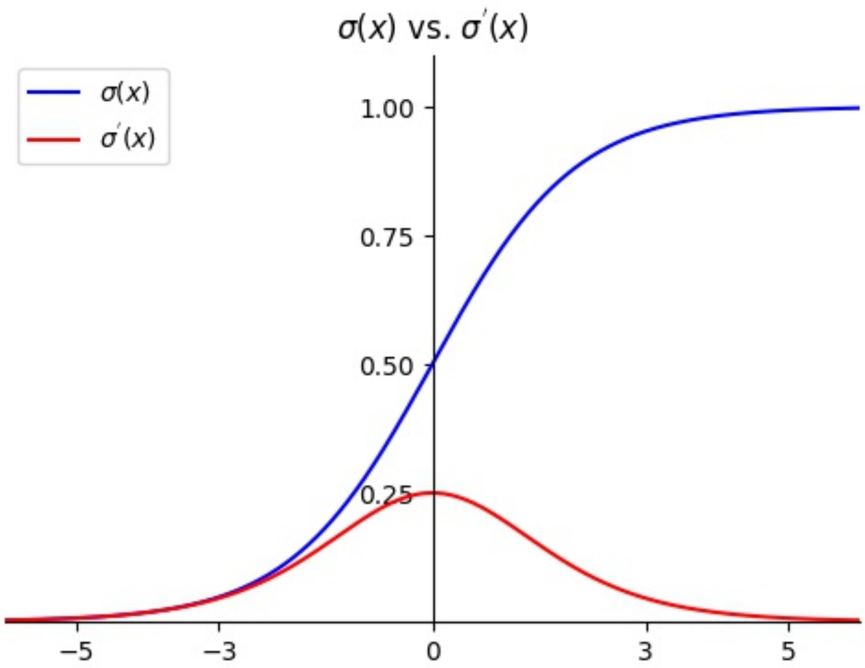
9.7 the Properity of Sigmod function

(a). Prove $1-\sigma(x)=\sigma(-x)$.

$1-\sigma(x)=\frac{e^{-x}}{1+e^{-x}}=\frac{1}{1+e^{x}}=\sigma(-x)$.

(b). Prove $\sigma^{'}(x)=\sigma(x)(1-\sigma(x))$ and plot the curves of $\sigma(x)$ and $\sigma^{'}(x)$.

$\sigma^{'}(x)=\frac{\exp(-x)}{(1+exp(-x))^2}=\sigma(x)(1-\sigma(x))$.



(c) show that sigmoid function is easily leads to the vanishing gradient difficulty.

由`(b)`中$\sigma^{'}(x)$的图像可知，$\sigma^{'}(x)\le\frac{1}{4}$，且$\sigma^{'}(3)=0.045$，$\sigma^{'}(5)=0.0066$.

由反向传播的链式法则(64) 可知，当$|y^{(i)}|>5$时，$\frac{\partial y^{(i)}}{\partial(\theta^{(i)})^T}\to 0$.所以$\frac{\partial l}{\partial(\theta^{(i)})^T}\to 0$。

由反向传播链式法则可知，当$l+1$层的误差项很小时，反向传播到第l层的梯度乘以一个小于0(更多时候接近于0)的因子导致梯度指数下降，所以深层神经网络使用sigmoid激活函数时，容易出现梯度消失而导致无法训练的问题。

10.1 Huffman tree for Fibonacci numbers

(a) first 6 Fibonacci numbers

```
1, 1, 2, 3, 5, 8
```

(b) Prove $f_n=\frac{\alpha^n-\beta^n}{\alpha-\beta}$ for all $n\in\mathbb{Z}^+$ where $\alpha=\frac{1+\sqrt{5}}{5}$, $\beta=\frac{1-\sqrt{5}}{2}$.

易知：$\alpha\beta=-1$,$\alpha+\beta=1$.

由**韦达定理**，构造一元二次方程：$x^2-x-1=0$.

易知α、β是上述一元二次方程组的两实根；得：$\alpha+1=\alpha^2$, $\beta+1=\beta^2$.

当$n \in \{1,2\}$时，等式成立；

令$n \geq 3$, $f_n = f_{n-1} + f_{n-2} = \frac{\alpha^{n-1}-\beta^{n-1}}{\alpha-\beta} + \frac{\alpha^{n-2}-\beta^{n-2}}{\alpha-\beta} = \frac{\alpha^{n-2}(1+\alpha)-\beta^{n-2}(1+\beta)}{\alpha-\beta} = \frac{\alpha^{n-2}\alpha^2-\beta^{n-2}\beta^2}{\alpha-\beta} = \frac{\alpha^n-\beta^n}{\alpha-\beta}$.

(c) Prove that $\sum_{i=1}^n F_i = F_{n+2} - 1$.

当$n = 1$时，$F_1 = F_3 - 1 = 2 - 1 = 1$成立；

假设当$n = k(k \geq 1)$时，$\sum_{i=1}^k F_i = F_{k+2} - 1$成立，

则当$n = k+1$时，$\sum_{i=1}^{k+1} F_i = \sum_{i=1}^k F_i + F_{k+1} = F_{k+2} + F_{k+1} - 1 = F_{k+3} - 1$成立；

由数学归纳法可知，对于$\forall n \in \mathbb{Z}^+, \sum_{i=1}^n F_i = F_{n+2} - 1$成立；

(d) Prove that $\sum_{i=1}^n iF_i = nF_{n+2} - F_{n+3} + 2$.

证明方法与(c)类似：

当$n = 1$时，$1F_1 = 1F_3 - F_4 + 2 = 1 * 2 - 3 + 2 = 1$成立。

假设$n = k(k \geq 1)$时，$\sum_{i=1}^k iF_i = kF_{k+2} - F_{k+3} + 2$成立，

当$n = k+1$时，$\sum_{i=1}^{k+1} iF_i = \sum_{i=1}^k iF_i + (k+1)F_{k+1} = kF_{k+2} - F_{k+3} + 2 + (k+1)F_{k+1}$

由于$kF_{k+2} - F_{k+3} = (k+1)F_{k+2} - F_{k+2} - F_{k+3} = (k+1)F_{k+2} - F_{k+4}$，

所以$\sum_{i=1}^{k+1} iF_i = (k+1)F_{k+2} - F_{k+4} + 2 + (k+1)F_{k+1} = (k+1)F_{k+3} - F_{k+4} + 2$.得证！

(e) draw huffman tree for $\frac{F_i}{F_7-1}$ distribution and the general case

由于Huffman树构建过程非常简单，这里直接叙述构造过程，不做图。

由$F_n = F_{n-1} + F_{n-2}$可知，易知当$n > 2$时，$F_{n+1} < \sum_{i=1}^n F_i$，且$F_{n+2} > \sum_{i=1}^n F_i$，所以Huffman 树每层只有2个节点。

构建Huffman树时最先选取F_1、F_2作为叶子节点，为保持Huffman树构建过程一致，将F_3作为左叶子节点($p(F_3) = p(F_1) + p(F_2)$)，易知Huffman树为向右下生长的树，每个非叶子节点向左有一个左叶子节点。

得：$F_5, F_4, \ldots, F_2, F_1$的Huffman 编码依次为: $0, 10, 110, 1110, 1111$.

对于$\frac{F_i}{F_{n+2}-1}(1 \leq i \leq n)$时，$F_n$编码为0，$F_i$的编码为$1 + F_{i+1}$ $(2 \leq i < n)$.F_1将F_2编码最末尾的0换成1；

(f) Prove that $B_n = \frac{F_{n+4}-(n+4)}{F_{n+2}-1}$.

由(e)易知，$B_n = \sum_{i=1}^n (n-i+1)\frac{F_i}{F_{n+2}-1} - 1 \cdot \frac{F_1}{F_{n+2}-1}$.

为计算简便，省略分母$F_{n+2} - 1$.

即只需证明：$F_{n+4} - (n+4) = \sum_{i=1}^n (n-i+1)F_i - F_1 = (n+1)\sum_{i=1}^n F_i - \sum_{i=1}^n iF_i - F_1$.

由(c)和(d)可知：

$(n+1)\sum_{i=1}^n F_i - \sum_{i=1}^n iF_i - F_1 = (n+1)(F_{n+2} - 1) - nF_{n+2} - F_{n+3} - 2 - 1 = F_{n+2} + F_{n+3} - n - 4 = F_{n+4} - (n+4)$

证毕！

(g) What is $\lim_{n\to\infty} B_n$

由于F_n是n的指数函数，$\lim_{n\to\infty} B_n = \lim_{n\to\infty} \frac{F_{n+4}}{F_{n+2}} = \lim_{n\to\infty} \frac{\sum_{i=1}^{n+2} F_i}{F_{n+2}} = \lim_{n\to\infty} \frac{F_{n+2}+F_{n+1}+\sum_{i=1}^n F_i}{F_{n+2}}$

易知：$\lim_{n\to\infty} \frac{F_{n+1}}{F_{n+2}} = \frac{1}{\alpha}$. $\lim_{n\to\infty} \frac{F_{n+2}+\sum_{i=1}^n F_i}{F_n+2} = 2$.

所以：$\lim_{n\to\infty} B_n = 2 + \frac{1}{\alpha}$.

> 讲义10_IT P3 Huffman tree构建过程第三步：

> - Repeat the above two steps until the priority queue is ~~**empty**~~. 更正为
>
> > - Repeat the above two steps until there is only one node(root) left in the priority queue.

10.2 some questions.

(a) what are the properties that distance metric function must satisfy?

- $d(x, y) \geq 0$, non-negative, and $d(x, y) = 0$ implies $x = y$；
- $d(x, y) = d(y, x)$, symmetric；
- $d(x, k) + d(k, y) \geq d(x, y)$, triangle inequality；

(b) is KL divergence a valid distance metric

KL divergence不是合法的distance metric. 因为KL 散度不具有对称性和三角不等式，但符合非负性；由于计算太*费劲*了，下面直接使用`python`计算出KL divergence.

(c) Write code to verify your calculation.

```python
import math

probability = [[0.5, 0.5], [0.25, 0.75], [0.125, 0.875]]
events = ['A', 'B', 'C']

def KL_divergence(prob1, prob2):
    divergence = 0
    for p_x, q_x in zip(prob1, prob2):
        divergence += p_x * math.log(p_x / q_x, 2)
    return divergence

for event1, prob1 in zip(events, probability):
    for event2, prob2 in zip(events, probability):
        divergence = KL_divergence(prob1, prob2)
        print("KL({} || {}) = {:.3f}".format(event1, event2, divergence))
```

输出结果如下：

```
KL(A || A) = 0.000
KL(A || B) = 0.208
KL(A || C) = 0.596
KL(B || A) = 0.189
KL(B || B) = 0.000
KL(B || C) = 0.083
KL(C || A) = 0.456
KL(C || B) = 0.070
KL(C || C) = 0.000
```

- KL非负，KL等于0时当且仅当两个分布完全相等；
- 不满足对称性，即$KL(A||B) \neq KL(B||A)$；
- 不满足三角不等式：$KL(A||B) + KL(B||C) = 0.291 \leq KL(A||C) = 0.596$；

1. https://docs.opencv.org/4.1.0/da/d60/tutorial_face_main.html.↩
2. https://pan.baidu.com/s/1L34lfFdD5NBTeHIR4YQauA 密码: **yzq0**.↩