PCA 学习 (二)

PCA 证明及特征脸提取

王泽民

691077364@qq.com

软件学院

中山大学

2014.10.30

提纲

PCA 数学证明

- 最大化方差
- 最小化误差

特征脸提取

- 代码
- 结果展示

目录

PCA 数学证明

最大化方差

最小化误差

特征脸提取

在 Ng 的机器学习课上说关于 PCA 有 9-10 种数学解释

今天主要从两个角度:

- 最大化方差理论
- 最小化误差

在 Ng 的机器学习课上说关于 PCA 有 9-10 种数学解释

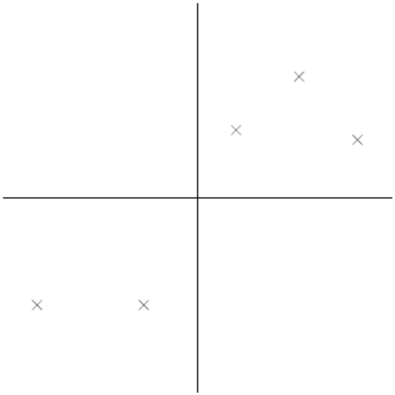
今天主要从两个角度：

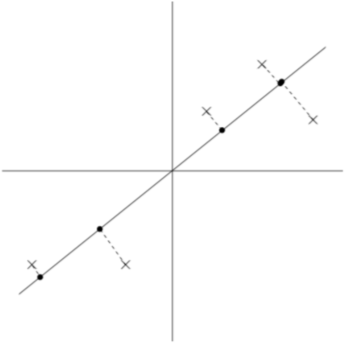
- 最大化方差理论
- 最小化误差

问题引出

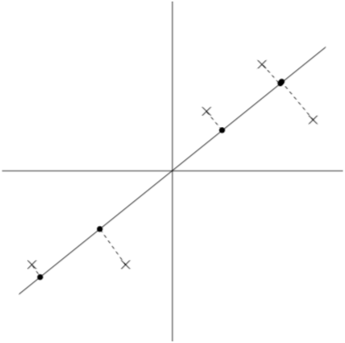
最好的 k 维特征是将 n 维样本点转换为 k 维后，每一维上的样本方差都很大

比如下图有 5 个样本点:(已经做过预处理，均值为 0，特征方差归一)

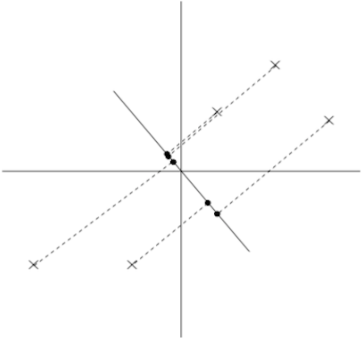




Figure： 此方向上投影方差较大

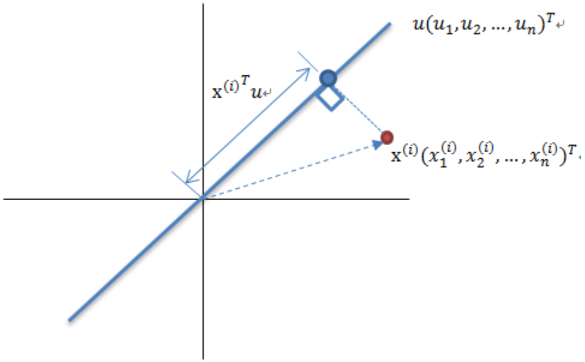


Figure： 此方向上投影方差较大



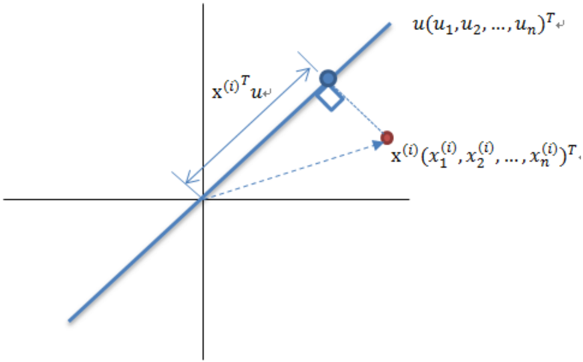
Figure： 此方向上投影方差较小

投影



红色点	样例
u	直线斜率，直线的方向向量 (单位向量)
蓝色点	样例在 u 上的投影

投影



这些样本点（样例）的每一维特征均值都为 0，因此投影到 u 上的样本点（只有一个到原点的距离值）的均值仍然是 0

$$
\begin{aligned}
Var(a) &= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)T}\mu)^2 \\
&= \frac{1}{m}\sum_{i=1}^{m}(x^{(i)T}u)^T(x^{(i)T}u) \\
&= \frac{1}{m}u^T x^{(i)^T} x^{(i)} u \\
&= u^T[\frac{1}{m}x^{(i)}x^{(i)T}]u
\end{aligned}
$$

记

$$
\Sigma = \frac{1}{m}x^{(i)}x^{(i)T}
$$

问题

$$\max \mu^T \Sigma \mu$$

$$st \ \mu^T \mu = 1$$

定义:

$$L(\mu, \lambda) = \mu^T \Sigma \mu - \lambda(\mu^T \mu - 1)$$

拉格朗日乘子法

是一种寻找变量受等值条件所限制的多元函数的极值的方法

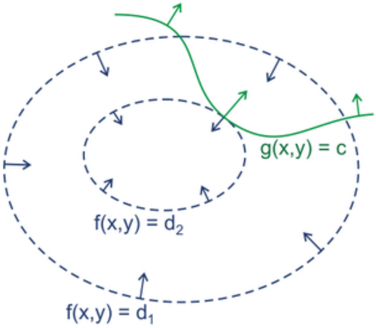
先看一个二维的例子:

假设有函数:f(x, y)，要求其极值.

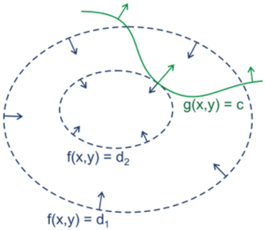
且满足条件:

$$g(x, y) = c$$

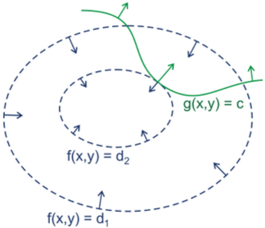
拉格朗日乘子法



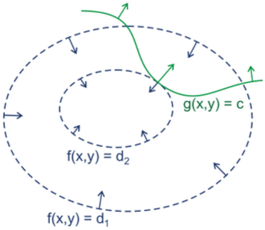
绿线	约束 g(x,y) = c 的点的轨迹
蓝线	f 的等高 (值) 线,$d_1 < d_2$
箭头	斜率，和等高线的法线平行



想像我们沿着 g = c 的可行集走；因为大部分情况下 f 的等高线和 g 的可行集线不会重合，但在有解的情况下，这两条线会相交。想像此时我们移动 g = c 上的点，因为 f 是连续的方程，我们因此能走到 $f(x, y) = d_n$ 更高或更低的等高线上，也就是说 d_n 可以变大或变小



只有当 $g = c$ 和 $f(x, y) = d_n$ 相切，也就是说，此时，我们正同时沿着 $g = c$ 和 $f(x, y) = d_n$ 走。这种情况下，会出现极值



用向量的形式来表达的话，我们说相切的性质在此意味着 f 和 g 的切线在某点上平行。此时引入一个未知标量 λ，并求解：

$$\nabla\left[f(x,y)+\lambda(g(x,y)-c)\right]=0$$

且 $\lambda \neq 0$

矩阵求导

行向量 y^T 对列向量 x 求导:

注意 1xM 向量对 Nx1 向量求导后是 NxM 矩阵

将 Y 的每一列对 x 求偏导，将各列构成一个矩阵

$$\begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix}$$

重要结论:

$$\frac{dx^T}{dx} = I$$

$$\frac{d(fg)}{dx} = (\frac{df^T}{dx})g + (\frac{dg}{dx})f^T$$

求解

$$L(\mu, \lambda) = \mu^T \Sigma \mu - \lambda(\mu^T \mu - 1)$$

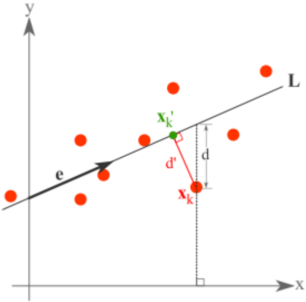
对 μ 求偏导:

$$\nabla L = 2\Sigma\mu - 2\lambda\mu = 0$$

μ 就是特征向量，选取前 k 个特征值:

$$y^{(i)} = \begin{bmatrix} \mu_1^T x^{(i)} \\ \mu_2^T x^{(i)} \\ \vdots \\ \mu_k^T x^{(i)} \end{bmatrix}$$

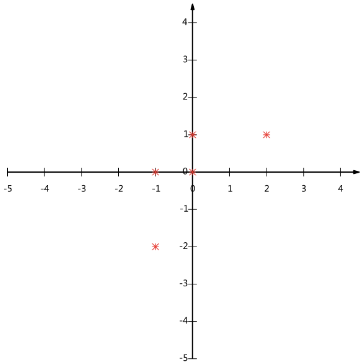
最小化误差



假设有这样的二维样本点（红色点），本质是求直线，那么度量直线求的好不好，不仅仅只有方差最大化的方法。再回想我们最开始学习的线性回归等，目的也是求一个线性函数使得直线能够最佳拟合样本点，那么我们能不能认为最佳的直线就是回归后的直线呢?

最小化误差

$$\begin{pmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{pmatrix}$$



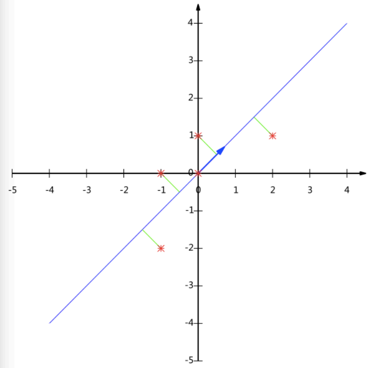
最小二乘法:

$$a = \frac{N\Sigma xy - \Sigma x \Sigma y}{N\Sigma x^2 - (\Sigma x)^2}$$

解得斜率为 $\frac{2}{3}$

PCA:

解得斜率为 1



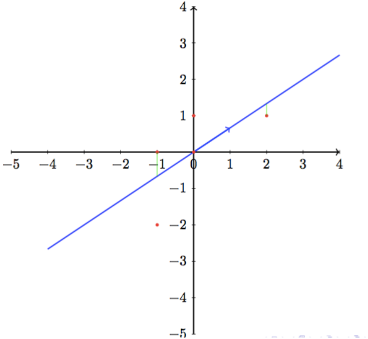


Figure : PCA 结果

Figure : 回归结果

投影并不改变数据的分布

令直线 L 穿越 m(样本中心点)，$\mathbf{w}$ 为其指向向量，且 $\|\mathbf{w}\| = 1$
直线 L 上的任一点可表示如下：

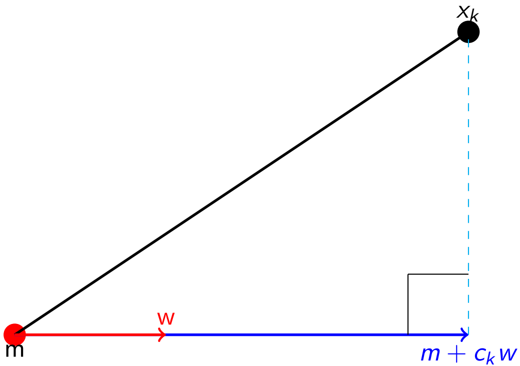
$$\mathbf{x} = \mathbf{m} + c\mathbf{w}$$

其中 c 为一纯量，$|c|$ 代表 $\mathbf{x}$ 至 $\mathbf{m}$ 的距离。一旦 $\mathbf{w}$ 给定，我们可以用 $\mathbf{m} + c_k\mathbf{w}$ 来近似 $\mathbf{x}_k$，
最佳的近似系数 $c_1, \ldots, c_n$ 必须最小化均方误差：

$$\begin{aligned}
E_1(\{c_k\}, \mathbf{w}) &= \frac{1}{n-1}\sum_{k=1}^{n} \|(\mathbf{m} + c_k\mathbf{w}) - \mathbf{x}_k\|^2 \\
&= \frac{1}{n-1}\sum_{k=1}^{n} \|c_k\mathbf{w} - (\mathbf{x}_k - \mathbf{m})\|^2 \\
&= \frac{1}{n-1}\left(\sum_{k=1}^{n} c_k^2\|\mathbf{w}\|^2 - 2\sum_{k=1}^{n} c_k\mathbf{w}^T(\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2\right).
\end{aligned}$$

从几何上来看：

$c_k\mathbf{w}$ 即是离差 $\mathbf{x}_k - \mathbf{m}$ 在直线 L 的正交投影



$$c_k = \mathbf{w}^T(\mathbf{x}_k - \mathbf{m})$$

利用这一关系进行化简:

$$\begin{aligned}
E_1(\mathbf{w}) &= \frac{1}{n-1}\left(\sum_{k=1}^{n} c_k^2 - 2\sum_{k=1}^{n} c_k^2 + \sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2\right) \\
&= -\frac{1}{n-1}\sum_{k=1}^{n}\left(\mathbf{w}^T(\mathbf{x}_k - \mathbf{m})\right)^2 + \frac{1}{n-1}\sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= -\frac{1}{n-1}\sum_{k=1}^{n}\mathbf{w}^T(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T\mathbf{w} + \frac{1}{n-1}\sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= -\mathbf{w}^T\left(\frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T\right)\mathbf{w} + \frac{1}{n-1}\sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2,
\end{aligned}$$

上面使用了 $\mathbf{w}^T(\mathbf{x}_k - \mathbf{m}) = (\mathbf{x}_k - \mathbf{m})^T\mathbf{w}$ 令

$$S = \frac{1}{n-1}\sum_{k=1}^{n}(\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T$$

$\sum_{k=1}^{n}\|\mathbf{x}_k - \mathbf{m}\|^2$ 是一常数，所以最小化 $E_1(\mathbf{w})$ 等价于最大化 $\mathbf{w}^T S\mathbf{w}$，这和之前转到同一个问题

再来推广一下:

$$\mathbf{x} = \mathbf{m} + z_1\mathbf{w}_1 + \cdots + z_r\mathbf{w}_r$$

$$E_r\left(\{\mathbf{w}_j\}\right) = \sum_{k=1}^{n}\left\|\left(\mathbf{m} + \sum_{j=1}^{r} z_{kj}\mathbf{w}_j\right) - \mathbf{x}_k\right\|^2$$

化简:

$$z_{kj} = w_j^T (x_k - m)$$

$$\begin{aligned}
E_r(\{\mathbf{w}_j\}) &= \sum_{k=1}^{n} \left\| \left(\mathbf{m} + \sum_{j=1}^{r} z_{kj}\mathbf{w}_j \right) - \mathbf{x}_k \right\|^2 \\
&= \sum_{k=1}^{n} \left\| \sum_{j=1}^{r} z_{kj}\mathbf{w}_j - (\mathbf{x}_k - \mathbf{m}) \right\|^2 \\
&= \sum_{k=1}^{n} \left\| \sum_{j=1}^{r} z_{kj}\mathbf{w}_j \right\|^2 - 2\sum_{k=1}^{n}\sum_{j=1}^{r} z_{kj}\mathbf{w}_j^T(\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= \sum_{k=1}^{n}\sum_{j=1}^{r} z_{kj}^2 - 2\sum_{k=1}^{n}\sum_{j=1}^{r} z_{kj}^2 + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= -\sum_{k=1}^{n}\sum_{j=1}^{r} \left(\mathbf{w}_j^T(\mathbf{x}_k - \mathbf{m})\right)^2 + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= -\sum_{j=1}^{r} \mathbf{w}_j^T \left(\sum_{k=1}^{n} (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T \right) \mathbf{w}_j + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2 \\
&= -\sum_{j=1}^{r} \mathbf{w}_j^T S \mathbf{w}_j + \sum_{k=1}^{n} \|\mathbf{x}_k - \mathbf{m}\|^2
\end{aligned}$$

$$\max_{\mathbf{w}_i^T\mathbf{w}_j=\delta_{ij}} \sum_{j=1}^{r} \mathbf{w}_j^T S \mathbf{w}_j$$

$\delta_{ij} = 1$ 若 $i = j$，否则 $\delta_{ij} = 0$

定义

$$L(\{\mathbf{w}_j\}, \{\mu_{ij}\}) = \sum_{j=1}^{r} \mathbf{w}_j^T S \mathbf{w}_j - \sum_{i=1}^{r}\sum_{j=1}^{r} \mu_{ij}(\mathbf{w}_i^T \mathbf{w}_j - \delta_{ij})$$

计算偏导数:

$$\frac{\partial L}{\partial \mathbf{w}_j} = 2S\mathbf{w}_j - 2\mu_{jj}\mathbf{w}_j - \sum_{i \neq j}(\mu_{ij} + \mu_{ji})\mathbf{w}_i = \mathbf{0}, \quad j = 1, \ldots, r$$

对于 $i \neq j$，设

$$\mu_{ij} + \mu_{ji} = 0S\mathbf{w}_j = \mu_{jj}\mathbf{w}_j, \quad j = 1, \ldots, r$$

当 $\mathbf{w}_1, \ldots, \mathbf{w}_r$ 是样本协方差矩阵 S 的最大 r 个特征值 $\lambda_1, \ldots, \lambda_r$ 的对应 (标准正交) 特征向量时，目标函数

$$\sum_{j=1}^{r} \mathbf{w}_j^T S\mathbf{w}_j = \sum_{j=1}^{r} \lambda_j \mathbf{w}_j^T \mathbf{w}_j = \sum_{j=1}^{r} \lambda_j$$

有最大值

目录

读取数据

Matlab:

```
numOfPic = 12;
picSize = 64;
trainSet = zeros( numOfPic, 64*64 );
for i = 1 : numOfPic
    str=sprintf('res/face%05d.bmp',i);
    img = imread( str );
    img = imresize(img,[64,64]);
    img = double(img);
    trainSet(i,:) = img(:);
end
```

读取数据

Matlab：

```
 numOfPic = 12;
picSize = 64;
trainSet = zeros( numOfPic, 64*64 );
for i = 1 : numOfPic
     str=sprintf('res/face%05d.bmp',i);
     img = imread( str );
     img = imresize(img,[64,64]);
     img = double(img);
     trainSet(i,:) = img(:);
end
```

- 一行是一个数据，12 个样本
- 每个样本 64*64=4096 维

读取数据

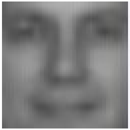
Matlab：

```
numOfPic = 12;
picSize = 64;
trainSet = zeros( numOfPic, 64*64 );
for i = 1 : numOfPic
    str=sprintf('res/face%05d.bmp',i);
    img = imread( str );
    img = imresize(img,[64,64]);
    img = double(img);
    trainSet(i,:) = img(:);
end
```

- 一行是一个数据，12 个样本
- 每个样本 64*64=4096 维

平均脸:

```
 meanValue = mean( trainSet );
trainSet_norm = bsxfun(@minus,trainSet,
    meanValue);
sigma = std(trainSet_norm); %每一列的标准差
trainSet_norm = bsxfun(@rdivide,
    trainSet_norm, sigma);
imwrite(uint8(reshape(meanValue, picSize,
    picSize)),'ans/mean.bmp');
```



协方差矩阵与特征向量

```
X = trainSet_norm;
[m, n] = size(X);

Cov = 1/m*X'*X;
[U ,S, V] =svd(Cov);

eigValue = diag(S);
AllSum = sum(eigValue);
cur_Sum = 0;
p =0;
while( cur_Sum/ AllSum < 0.9)
p = p + 1;
        cur_Sum = sum(eigValue(1:p));
end;
```

特征脸

```
for i = 1:p
ef = P(:,i)';
minVal = min(ef);
ef =ef - minVal;
max_val = max(abs(ef));
ef = ef/max_val;
Eigenface = reshape(ef,[64,64]);
str = sprintf('res/eigface%05d.bmp',i);
imwrite( Eigenface,str);
end
```

特征脸

测试与还原

```
testSet = zeros(4, 64*64);
for i = 3 :6
    str=sprintf('res/test%d.bmp',i);
    img =imread(str);
    img = double(img);
    testSet(i,:) = img(:);
end;
```

测试与还原


```
testNorm = bsxfun(@minus, testSet, meanValue
    );
Z = testNorm * P;
rc = Z * P';
for i = 3 : 6
    face = rc(i,:) + meanValue;
    rface = reshape(face, 64, 64);
    str = sprintf('res/test_re%d.bmp',i);
    imwrite(uint8(rface),str);

end;
```

Figure : 测试图片

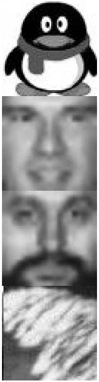


Figure : 测试图片

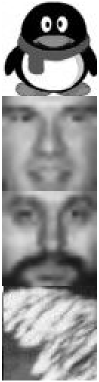
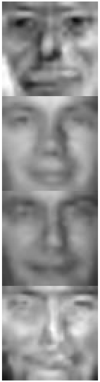
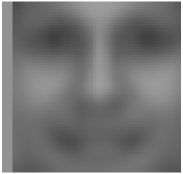


Figure : 还原结果



1200 个样本

平均脸



看上去比较模糊，只是人脸的轮廓

特征脸

从 19x19 降到 21

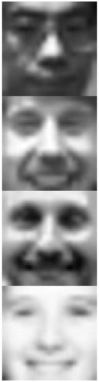
我吧结果从 19*19 放大到 64*64，看起来不太好

Ng 在课堂上展示的特征脸

1. 从左到右是否被照亮
2. 脸部明亮度
3. 脸部的阴影

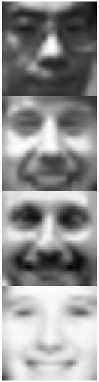
测试与还原

Figure : 测试数据



测试与还原

Figure： 测试数据



Figure： 还原结果