From Singular Value Decomposition (SVD) to Principle Component Analysis (PCA)

Jingbo Xia

Huazhong Agricultural University

xiajingbo.math@gmail.com

2023-12-19

开宗明义 VI

- 对矩阵的操作必要性在于数据本身的矩阵存储形式。
- 矩阵分解的目的之一是把放置在矩阵中的数据进行低秩逼近——可用以进行数据压缩；可用以消除数据稀疏性。
- SVD 是一个知名的矩阵分解算法，该算法的自然应用是衍生到推荐系统，也可以是 NLP 领域的潜在语义分析。
- PCA 同样是对存储在数据中的矩阵进行操作。我们介绍该算法的两个应用——第一是数据的降维，第二是数据的重构。
- 这个教学材料中包括一个图片处理的示例，它能帮助同学们较为生动地理解 PCA 算法中的“维”。
- 最后，为什么要把 SVD 和 PCA 放在一起讲授呢？难道仅仅因为它们都同时是在对矩阵类型的数据进行算法处理吗？

Table of contents I

Outline

SVD 是一个知名的矩阵算法，广泛地（并不限定于）应用于以下场景

- Image compression（图像压缩），
- Matrix completion（矩阵补全），
- Recommendation system（推荐系统），
- Latent semantic index (有时候被称作 Latent semantic analysis，潜在语义分析),
- ...

定理（Singular Value Decomposition）

For an $m \times n$ *matrix* A *of rank* r*, there exists a factorization (分解) as follows:*

$$A = U \Sigma V^T,$$

where U *and* V *are orthogonal matrices*（正交矩阵），

$\Sigma = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix}$,

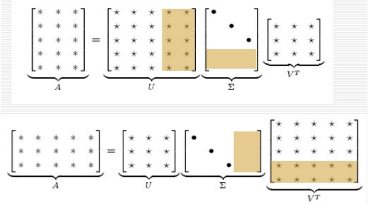
$\Delta = diag(\sigma_1, \cdots, \sigma_r)$,

$\sigma_i = \sqrt{\lambda_i}$ *(which is called as singular value*（奇异值）*of* A*),* $i = 1, 2, \cdots, r$,

$r = Rank(A)$,

λ_i *is the eigenvalue*（特征值）*of* AA^T.

For better understanding, a 5×3 matrix A is shown here:



Here, we rewrite the above formula to

$$A = (A_1, A_2, A_3) = (U_1, \cdots, U_5) \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \\ V_3^T \end{pmatrix},$$

where A_i is the i-th column of A, U_k is the k-th column of V, and V_j is the j-th column of V.

奇异值分解/Singular Value Decomposition IV

SVD——定理及其证明

Straightforward computation leads to

$$A = \sigma_1 U_1 V_1^T + \sigma_2 U_2 V_2^T + \sigma_3 U_3 V_3^T.$$

Thus $\sigma_1 U_1 V_1^T + \sigma_2 U_2 V_2^T$ is named as a "low-rank approximation of " A.
Generally, $\sigma_{i=1}^{R} U_i V_i^T$ is a rank-R approximation of an arbitrary A.

Why low rank approximation（低秩逼近）

Why does it need low rank approximation? The answer is to remove sparsity（稀疏性）in a proper way.

有关数据的稀疏性，请联想一下推荐系统的应用场景。

Lemma 1.
Suppose $A \in \mathbb{R}^{m\times n}$, the eigenvalues of A^TA and AA^T are all nonnegative（非负的）.

Lemma 1.
Suppose $A \in \mathbb{R}^{m\times n}$, the eigenvalues of A^TA and AA^T are all nonnegative（非负的）.

Proof: Suppose λ is the eigenvalue of A^TA, and x is the corresponding eigenvector, then we have

$$A^TAx = \lambda x.$$

Since A^TA is symmetrical（对称的）, so λ is a real number, and we have

$$0 \le (Ax, Ax) = (Ax)^T(Ax) = \lambda x^Tx.$$

Because $x^Tx > 0$, we have $\lambda \ge 0$.
Similarly, we know that eigenvalues of AA^T is nonnegative. □

Lemma 2.
Suppose $A \in \mathbb{R}^{m\times n}$, we have $r(A) = r(A^TA) = r(AA^T)$.

Lemma 2.
Suppose $A \in \mathbb{R}^{m\times n}$, we have $r(A) = r(A^TA) = r(AA^T)$.

Proof of this lemma is straightforward. Please consider the solution of linear equation system（齐次线性方程组）$Ax = 0$, and $A^TAx = 0$, and proof that the solutions are the same.

Lemma 3:
Suppose $A \in \mathbb{R}^{m \times n}$, then AA^T and A^TA have the same eigenvalues（奇异值）.

Lemma 3:
Suppose $A \in \mathbb{R}^{m\times n}$, then AA^T and A^TA have the same eigenvalues（奇异值）.

Proof. Assume λ is an eigenvalue of A^TA, then we have $A^TAx = \lambda x$.
By multiplying A, the equation turns to be $AA^TAx = \lambda Ax$.
It suffices to know that λ is an eigenvalue of AA^T.
Vice versa, eigenvalue of AA^T is also eigenvalue of A^TA.
The above two statements suffice to show the lemma.

Lemma 4:

If two matrices are orthogonally equivalent（**正交等价**）to each other, they have the same singular values.

注：此处请留意一下正交等价的定义。

Lemma 4:
If two matrices are orthogonally equivalent（正交等价）to each other, they have the same singular values.

Proof. Suppose $A, B \in \mathbb{R}^{m\times n}$ are orthogonally equivalent, there exists orthogonal matrix $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, s.t, $A = UBV$.
From $A^TA = (UBV)^T(UBV) = V^TB^TBV = V^{-1}B^TBV$, we know that A^TA and B^TB have the same eigenvalues, so do the singular values. □

现在，是时候来证明 SVD 定理了。

[Singular Value Decomposition]

For an $m \times n$ matrix A of rank r, there exists a factorization（分解）as follows:

$$A = U\Sigma V^T,$$

where U and V are orthogonal matrices（正交），
$\Sigma = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix}$,
$\Delta = diag(\sigma_1, \cdots, \sigma_r)$,
$\sigma_i = \sqrt{\lambda_i}$ (which is called as singular value（奇异值）of A), $i = 1, 2, \cdots, r$,
$r = Rank(A)$,
λ_i is the eigenvalue（特征值）of AA^T.

[Main proof]

Proof. Since A^TA is symmetrical matrix, there exists orthogonal matrix V, ($r(V) = r$), s.t.,

$$V^T(A^TA)V = \begin{pmatrix} \Delta^2 & 0 \\ 0 & 0 \end{pmatrix},$$

here $\Delta = diag(\sigma_1, \cdots, \sigma_r)$, $\sigma_i^2 = \lambda_i$.

注：请回忆如何使用施密特正交化，将一个方阵正交对角化。
另外一个需要留意的结论：实对称矩阵永可对角化。

Splitting V, we have

$$V = (V_1, V_2),$$

where V_1 consists of the first r columns of V.

注：此处 V的矩阵分块方式是一个技巧。

$$\begin{pmatrix} \Delta^2 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix} (A^T A)(V_1, V_2)$$

$$= \begin{pmatrix} V_1^T A^T A V_1 & * \\ * & V_2^T A^T A V_2 \end{pmatrix}.$$

By comparing the left and right sides, we have

$$V_2^T A^T A V_2 = (A V_2)^T (A V_2) = 0.$$

It leads to $A V_2 = 0$. In addition, we also have

$$V_1^T A^T A V_1 = (A V_1)^T (A V_1) = \Delta^2.$$

For simplicity（为了记号的简洁）, we denote $U_1 = AV_1\Delta^{-1}$. Now, it is straightforward to verify that

$$U_1^T U_1 = \Delta^{-1} V_1^T A^T A V_1 \Delta^{-1} = E_r.$$

The first r columns of U (它们实际上构成了 U_1) are orthogonal unit vectors（正交单位向量）. Hence, there exists an extension of U_1 by $U_2 \in \mathbb{R}^{m\times m-r}$ and make $U = (U_1, U_2)$ an orthogonal matrix.

注：我们永可这样做——首先扩充 r个线性无关的向量到一个包含 m个向量的无关组，其次用施密特正交变换和单位化后，这 m个正交单位向量则组合成一个正交矩阵。

By doing the above, we already obtained V, Σ and U.

Let's check'em out.

$$U^TAV = \begin{pmatrix} U_1^T \\ U_2^T \end{pmatrix} A(V_1, V_2) = \begin{pmatrix} U_1^TAV_1 & U_1^TAV_2 \\ U_2^TAV_1 & U_2^TAV_2 \end{pmatrix}$$

$$= \begin{pmatrix} U_1^TU_1\Delta & U_1^T(AV_2) \\ U_2^TU_1\Delta & U_2^T(AV_2) \end{pmatrix} = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix},$$

as both $U_2^TU_1$ and AV_2 equal to 0.

So the theorem follows（定理得证）.

1. Exercise.
   SVD of A. Here $A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$.
   Answer sheet:

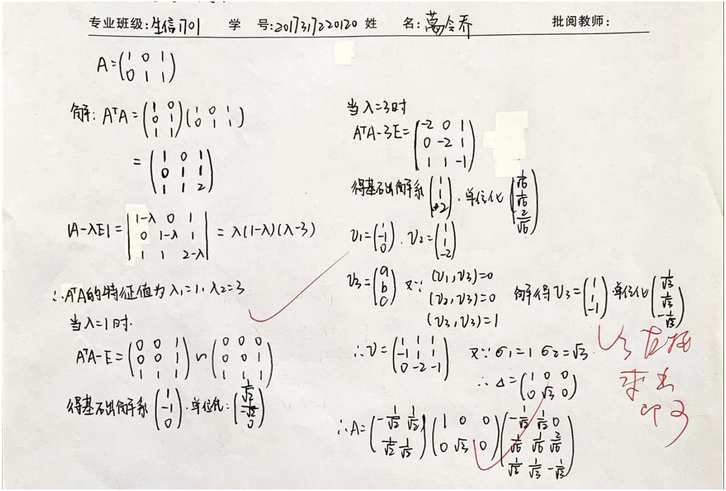
$$A = \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \sqrt{3} & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} \\ -\frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix}.$$

2. 请思考：SVD 能否用于数据逼近、矩阵补全等不同的应用场景。
3. 除了 SVD，还有哪些矩阵分解的算法？
4. 试着读读 NMF 的论文？(参考课程网页)

Singular Vector Decomposition II

练习

Answer Sheet



专业班级:生信1701　学　号:2017317220120　姓　名:葛会乔　批阅教师:

$$A=\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix}$$

解: $A^TA=\begin{pmatrix}1&0\\0&1\\1&1\end{pmatrix}\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix}=\begin{pmatrix}1&0&1\\0&1&1\\1&1&2\end{pmatrix}$

$$|A-\lambda E|=\begin{vmatrix}1-\lambda&0&1\\0&1-\lambda&1\\1&1&2-\lambda\end{vmatrix}=\lambda(1-\lambda)(\lambda-3)$$

∴ A^TA的特征值为 $\lambda_1=1$, $\lambda_2=3$

当 $\lambda=1$ 时

$$A^TA-E=\begin{pmatrix}0&0&1\\0&0&1\\1&1&1\end{pmatrix}\sim\begin{pmatrix}0&0&0\\0&0&1\\1&1&1\end{pmatrix}$$

得基础解系 $\begin{pmatrix}1\\-1\\0\end{pmatrix}$, 单位化: $\begin{pmatrix}\frac{1}{\sqrt2}\\-\frac{1}{\sqrt2}\\0\end{pmatrix}$

当 $\lambda=3$ 时

$$A^TA-3E=\begin{pmatrix}-2&0&1\\0&-2&1\\1&1&-1\end{pmatrix}$$

得基础解系 $\begin{pmatrix}1\\1\\2\end{pmatrix}$, 单位化 $\begin{pmatrix}\frac{1}{\sqrt6}\\\frac{1}{\sqrt6}\\\frac{2}{\sqrt6}\end{pmatrix}$

$v_1=\begin{pmatrix}1\\-1\\0\end{pmatrix}$, $v_2=\begin{pmatrix}1\\1\\-2\end{pmatrix}$

$v_3=\begin{pmatrix}a\\b\\c\end{pmatrix}$ 又∵ $(v_1,v_3)=0$, $(v_2,v_3)=0$, $(v_3,v_3)=1$ 解得 $v_3=\begin{pmatrix}1\\1\\-1\end{pmatrix}$ 单位化 $\begin{pmatrix}\frac{1}{\sqrt3}\\\frac{1}{\sqrt3}\\-\frac{1}{\sqrt3}\end{pmatrix}$

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

又∵ $\sigma_1=1$, $\sigma_2=\sqrt3$

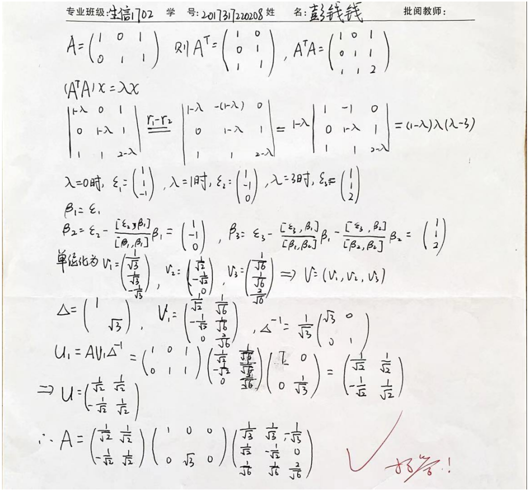
$$\therefore \Delta=\begin{pmatrix}1&0&0\\0&\sqrt3&0\end{pmatrix}$$

$$\therefore A=\begin{pmatrix}-\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\\\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\end{pmatrix}\begin{pmatrix}1&0&0\\0&\sqrt3&0\end{pmatrix}\begin{pmatrix}-\frac{1}{\sqrt2}&\frac{1}{\sqrt2}&0\\\frac{1}{\sqrt6}&\frac{1}{\sqrt6}&\frac{2}{\sqrt6}\\\frac{1}{\sqrt3}&\frac{1}{\sqrt3}&-\frac{1}{\sqrt3}\end{pmatrix}$$

Singular Vector Decomposition III

练习

Answer Sheet



专业班级:生信1702 学 号:2017317220208 姓 名:彭钱钱 批阅教师:

$A=\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix}$ 则 $A^T=\begin{pmatrix}1&0\\0&1\\1&1\end{pmatrix}$, $A^TA=\begin{pmatrix}1&0&1\\0&1&1\\1&1&2\end{pmatrix}$

$(A^TA)x=\lambda x$

$$\begin{vmatrix}1-\lambda&0&1\\0&1-\lambda&1\\1&1&2-\lambda\end{vmatrix} \xlongequal{r_1-r_2} \begin{vmatrix}1-\lambda&-(1-\lambda)&0\\0&1-\lambda&1\\1&1&2-\lambda\end{vmatrix} = 1-\lambda\begin{vmatrix}1&-1&0\\0&1-\lambda&1\\1&1&2-\lambda\end{vmatrix} = (1-\lambda)\lambda(\lambda-3)$$

$\lambda=0$时, $\xi_1=\begin{pmatrix}1\\1\\-1\end{pmatrix}$, $\lambda=1$时, $\xi_2=\begin{pmatrix}1\\-1\\0\end{pmatrix}$, $\lambda=3$时, $\xi_3=\begin{pmatrix}1\\1\\2\end{pmatrix}$

$\beta_1=\xi_1$

$\beta_2=\xi_2-\frac{[\xi_2,\beta_1]}{[\beta_1,\beta_1]}\beta_1=\begin{pmatrix}1\\-1\\0\end{pmatrix}$, $\beta_3=\xi_3-\frac{[\xi_3,\beta_1]}{[\beta_1,\beta_1]}\beta_1-\frac{[\xi_3,\beta_2]}{[\beta_2,\beta_2]}\beta_2=\begin{pmatrix}1\\1\\2\end{pmatrix}$

单位化为 $v_1=\begin{pmatrix}\frac{1}{\sqrt3}\\\frac{1}{\sqrt3}\\-\frac{1}{\sqrt3}\end{pmatrix}$, $v_2=\begin{pmatrix}\frac{1}{\sqrt2}\\-\frac{1}{\sqrt2}\\0\end{pmatrix}$, $v_3=\begin{pmatrix}\frac{1}{\sqrt6}\\\frac{1}{\sqrt6}\\\frac{2}{\sqrt6}\end{pmatrix}$ $\Rightarrow V=(v_1,v_2,v_3)$

$\Delta=\begin{pmatrix}1&\\&\sqrt3\end{pmatrix}$, $V_1=\begin{pmatrix}\frac{1}{\sqrt2}&\frac{1}{\sqrt6}\\-\frac{1}{\sqrt2}&\frac{1}{\sqrt6}\\0&\frac{2}{\sqrt6}\end{pmatrix}$, $\Delta^{-1}=\frac{1}{\sqrt3}\begin{pmatrix}\sqrt3&0\\0&1\end{pmatrix}$

$U_1=AV_1\Delta^{-1}=\begin{pmatrix}1&0&1\\0&1&1\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt2}&\frac{1}{\sqrt6}\\-\frac{1}{\sqrt2}&\frac{1}{\sqrt6}\\0&\frac{2}{\sqrt6}\end{pmatrix}\begin{pmatrix}1&0\\0&\frac{1}{\sqrt3}\end{pmatrix}=\begin{pmatrix}\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\\-\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\end{pmatrix}$

$\Rightarrow U=\begin{pmatrix}\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\\-\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\end{pmatrix}$

$\therefore A=\begin{pmatrix}\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\\-\frac{1}{\sqrt2}&\frac{1}{\sqrt2}\end{pmatrix}\begin{pmatrix}1&0&0\\0&\sqrt3&0\end{pmatrix}\begin{pmatrix}\frac{1}{\sqrt3}&\frac{1}{\sqrt3}&-\frac{1}{\sqrt3}\\\frac{1}{\sqrt2}&-\frac{1}{\sqrt2}&0\\\frac{1}{\sqrt6}&\frac{1}{\sqrt6}&\frac{2}{\sqrt6}\end{pmatrix}$

✓ 好答!

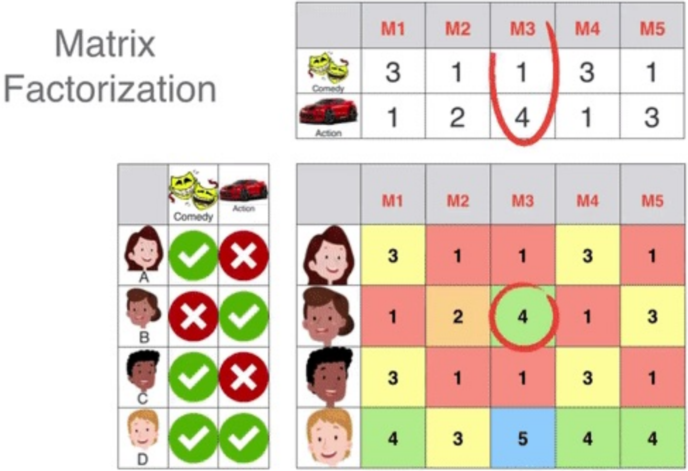
Outline

Application of SVD II

Matrix completion

Reference[1].

[1] https://technotipsondatascience.wordpress.com/2018/10/22/recommendation-system/?fbclid=IwAR2GRkXqvWMtgE_nG-hdJpmYFTOx6JWaYDFUaN3zQKpVa5S63ZOpElPL7xk

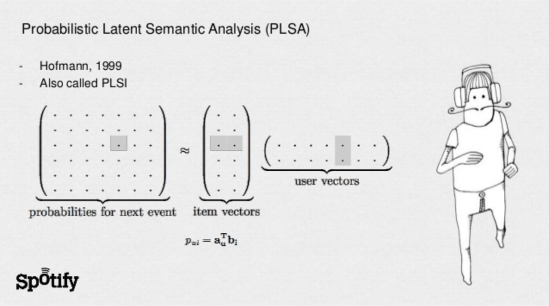


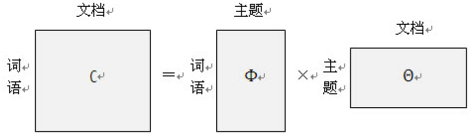
[2]

[2] How does Netflix recommend movies? Matrix Factorization. https://aitube.io/video/matrix-factorization/

Application of SVD

Latent semantic analysis





例 (Matlab Codes)

```
library(jpeg)
img <- readJPEG(system.file("img", ?xiajingbo.jpg", package="jpeg"))
if(exists("rasterImage")){
      plot(1:2, type='n')
      rasterImage(img,1,1,2,2)
}
```

例 (Matlab Codes)

```
dim(img)   #200*200*3
img_grey <- img[ , ,1]
dim(img_grey) #200*200
if(exists("rasterImage")){
      plot(1:2, type='n')
      rasterImage(img_grey,1,1,2,2)
}
```

Application of SVD

Image compression

Singular values:

[1] 1.069733e+02 2.919619e+01 2.205404e+01 1.689986e+01 1.326622e+01
[6] 8.760134e+00 8.373214e+00 6.950724e+00 5.907265e+00 4.979078e+00
[11] 4.717919e+00 4.288516e+00 4.127117e+00 3.861279e+00 3.502851e+00
[16] 3.053432e+00 2.676388e+00 2.546025e+00 2.414010e+00 2.184452e+00
...
[61] 3.274435e-01 3.149630e-01 2.963879e-01 2.894864e-01 2.858502e-01
[66] 2.645407e-01 2.500984e-01 2.384135e-01 2.315172e-01 2.262126e-01
...
[191] 2.782755e-03 2.684218e-03 2.081970e-03 1.854204e-03 1.662375e-03
[196] 1.387572e-03 1.324852e-03 8.272929e-04 5.080000e-04 2.307262e-04

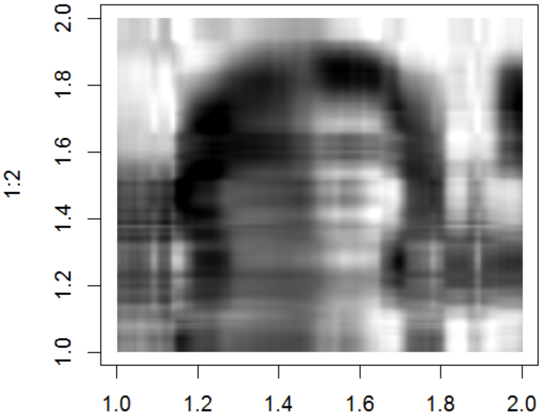
Note: In this case, there are 200 nonzero singular values for the pixel matrix. Let's watch how we reconstruct the matrix with low rank approximation. In the next page, R denotes the amount of singular values we use for the matrix approximation.

例 (Matlab Codes)

```
n=4
D <- diag(s$d[1:n])
img_compressed <- s$u[,1:n] %*% D %*% t(s$v[,1:n])

# To see how close the two figures are
img_grey-img_compressed
```

A 200×200 pixel figure. $R = 4$



SVD in image compression might be a way to explain the reason for low-rank approximation, and in the meantime, Eckart-Young-Mirsky Theorem evaluates the approximation via estimating the Frobenius norm of $||A - \tilde{A}||_F^2$

定理 (Eckart-Young-Mirsky Theorem)

For an $m \times n$ *matrix* A *of rank* r*, assume the SVD is* $A = U\Sigma V^T$*, where* $\Sigma = \begin{pmatrix} \Delta & 0 \\ 0 & 0 \end{pmatrix}$, $\Delta = diag(\sigma_1, \cdots, \sigma_r)$*. For an approximation of* A *as* $\tilde{A} = U\tilde{\Delta}V^T$*, where* $\tilde{\Delta} = diag(\sigma_1, \cdots, \sigma_l)$*, we have*

$$\min_{rank(\tilde{A} \leq l)} ||A - \tilde{A}||_F = \sigma_l^2 + \cdots + \sigma_r^2.$$

Outline

Principal component analysis (PCA) aims to use "an orthogonal transformation（正交变换）to convert a set of observations（观测值）of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called **principal component（主成分）**"[3].

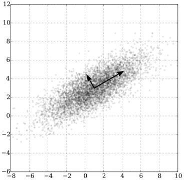


图 1: PCA of a multivariate Gaussian distribution（多维高斯分布）centered at $(1, 3)$ with a standard deviation（标准差）of 3 in roughly the $(0.866, 0.5)$ direction and of 1 in the orthogonal direction. The vectors shown are the eigenvectors of the covariance matrix（协方差矩阵）scaled by the square root（平方根）of the corresponding eigenvalue, and shifted so their tails are at the mean.

[3] https://en.wikipedia.org/wiki/Principal_component_analysis

投影/Projection!

As shown in figure 2, different selection of projection line leads to difference in the variance. It's no hard to observe that the left projection brings greater variance than the right projection does.

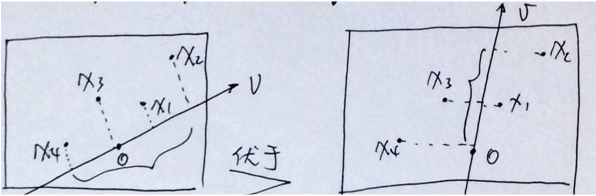


图 2: An example of four samples/points which are projected to different vector/direction.

Then, how to calculate the projection coordinates（投影坐标）?

Let's discuss it in a $\mathbb{R}^n$ space. If α project to β, then the projection[4] is

$$\frac{(\alpha,\beta)}{(\beta,\beta)}\beta = \frac{(\alpha,\beta)}{|\beta|}\frac{\beta}{|\beta|},$$

while the projection coordinate[5] in β is

$$\frac{(\alpha,\beta)}{|\beta|}.$$

Without the loss of the generality（不失一般性）, we assume $|\beta| = 1$, and then the projection coordinate is (α,β).

[4]Note: This is a vector, and this vector is with the same/ opposite direction of β

[5]Note: This is a scalar value.

Now it is time to discuss the variance. Assume there are l samples $X_i \in \mathbb{R}^d$, ($i = 1, 2, \cdots, l$), and V is a vector to which X_i projects. For simplicity, we assume that the mean of samples $\bar{X} = 0$. Denote σ^2 as variance of all of the projection coordinates, we know

$$\begin{aligned}\sigma^2 &= \frac{1}{l}\sum_{i=1}^{l}(V^T X_i - 0)^2 = \frac{1}{l}\sum_{i=1}^{l}(V^T X_i)(V^T X_i)^T \\ &= \frac{1}{l}\sum_{i=1}^{l} V^T X_i X_i^T V = V^T(\frac{1}{l}\sum_{i=1}^{l} X_i X_i^T)V \\ &:= V^T C V\end{aligned} \tag{1}$$

We denote C as the covariance matrix.

As a general form, i.e., if we remove the assumption that the mean of X_i equals to zero, the above deduction leads to:

$$\begin{aligned} C &= \frac{1}{l}\sum_{i=1}^{l}(X_i - \bar{X})(X_i - \bar{X})^T. \\ &= E[(X-\mu)(X-\mu)^T] \qquad (\mu = E[X]) \end{aligned} \tag{2}$$

Here, C is actually the covariance matrix.

Solve PCA model by applying some optimization theory and matrix calculus tricks.

The optimization problem with PCA is

$$V = \underset{V \in \mathbb{R}^d, |V|=1}{\arg\max} \; V^T C V \tag{3}$$

The Lagrangean function is

$$f(v, \lambda) = V^T C V - \lambda (V^T V - 1).$$

——————**计算环节**: $\frac{\partial f}{\partial V} = 0$.——————

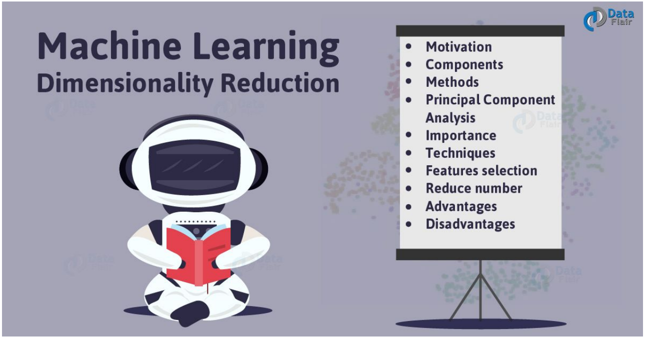
By solving $\frac{\partial f}{\partial V} = 0$, we have

$$CV = \lambda V.$$

This means that V, as a projection vector, is the eigenvector of the covariance matrix.

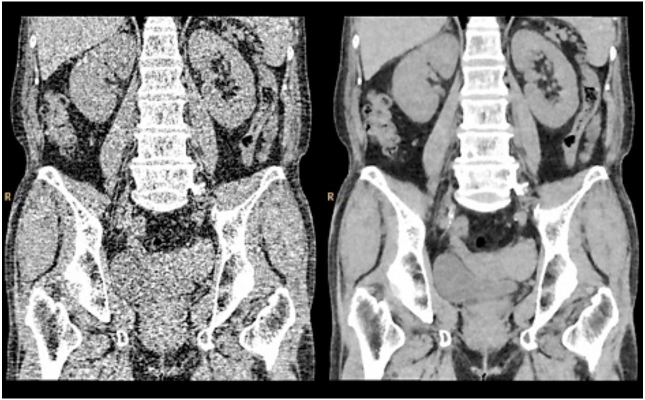
主成分分析/Principal Component Analysis
特征值排序带来的应用

Sorting of the eigenvalues (or singular values) leads to two main application scenarios: feature reduction（特征约简）, or dimentionality reduction（维度约简）[6],



[6] https://data-flair.training/blogs/dimensionality-reduction-tutorial/.

and data reconstruction（数据重构）[7].



[7] https://www.usa.philips.com/healthcare/product/HCNCTD449/iterative-model-reconstruction-reconstruction-technology

By sorting the singular values, or eigenvalues, with a descending order, one can select different V_j, and project $X_i \in \mathbb{R}^d$ to different V_j and obtain corresponding projection coordinate $(X_i, V_j) = X_i^T V_j$.

Feature reduction（特征约简）

Let's fix the first J V_j with greatest eigenvalues, and replace X_i with a J-tuple vector, $\hat{X}_i = (X_i^T V_1, X_i^T V_2, \cdots, X_i^T V_J)^T$. [a]

[a] We now convert the d-tuple vector X_i into a J-tuple one, i.e., $\hat{X}_i$, thus reducing the dimentionality of the data.

Data reconstruction（数据重构）

If we represented the sample X_i with an approximation, $\tilde{X}_i = (X_i^T V_1) V_1 + (X_i^T V_2) V_2 + \cdots + (X_i^T V_J) V_J$. [a]

[a] Here, $\tilde{X}_i$ is a low-rank approximate of X_i, which reconstructs the original data.

Outline

SVD and principle component analysis have close relevance.
Assume the matrix $A \in \mathbb{R}^{m\times n}$ contains m "n-tuple" sample vectors A_i ($i = 1, 2, .., \cdots, m$), i.e.,

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_m \end{pmatrix}, \tag{4}$$

When performing low-rank approximation, the purpose of PCA is to find several (let's say r) "n-tuple" principle components $V_1^T, \cdots, V_r^T$.

V_j $(j = 1, 2, \cdots, r)$ are the eigenvectors of co-variance matrix A^TA, s.t.,

$$A_i \approx b_{i1}\, V_1^T + b_{i2}\, V_2 + \cdots + b_{ir}\, V_r^T.$$

To put all A_i into A, we have

$$A = BV^T.$$

Interestingly, SVD of A provides solution of PCA:

$$A = U\Sigma V^T.$$

By collecting $U\Sigma$ and V, SVD provides B and V.

A tricky way to visualize the principle component, V.

Generally speaking, it is not that direct to figure out how A_i and V_j look like. A nice visualization example is recommended here, http://www.infoq.com/cn/articles/matrix-decomposition-of-recommend-system.

In this example, A_i is a 4096-tuple vector which presents a picture, and V_js are so-called "creepy faces" which represent different characteristics of faces. Here, these weighted sum of "creepy faces" reconstruct A_I.

Each A_i is a 4096-tuple vector, which represents a face figure.

Each A_i is approximated by a weighted sum of the below "creepy faces":

PCA 的若干有趣阐释 III

A tricky way to visualize the principle component, V.

V_j^T, "creepy faces":

例 (Python Codes)

```
from sklearn.datasets import fetch_olivetti_faces
from sklearn.decomposition import PCA

import matplotlib.pyplot as plt
%matplotlib inline
faces = fetch_olivetti_faces()
print(faces.DESCR)
```

There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement).

例 (Python Codes)

```
# Here are the first ten guys of the dataset
fig = plt.figure(figsize=(10, 10))
for i in range(5):
    ax = plt.subplot2grid((1, 5), (0, i))
    ax.imshow(faces.data[i * 10].reshape(64, 64), cmap=plt.cm.gray)
    ax.axis('off')

# Let's compute the PCA
pca = PCA()
pca.fit(faces.data)
```

例 (Python Codes)

```
# Now, the creepy guys are in the components_ attribute.
# Here are the first four ones:

fig = plt.figure(figsize=(10, 10))
for i in range(4):
    ax = plt.subplot2grid((2, 2), (0, i))

    ax.imshow(pca.components_[i].reshape(64, 64), cmap=plt.cm.gray)
    ax.axis('off')
```

例 (Python Codes)

```
# Reconstruction process

from skimage.io import imsave
face = faces.data[0]  # we will reconstruct the first face

# During the reconstruction process we are actually computing,
# at the kth frame, a rank k approximation of the face.
# To get a rank k approximation of a face,
# we need to first transform it into the 'latent space', and then
# transform it back to the original space
# Step 1: transform the face into the latent space.
# It's now a vector with 400 components. The kth component gives
# the importance of the kth  creepy guy

trans = pca.transform(face.reshape(1, -1))
# Reshape for scikit learn
```

PCA 的若干有趣阐释

A tricky way to visualize the principle component, V.

例 (Python Codes)

```
# Step 2: reconstruction. To build the kth frame, we use all the cre
# up until the kth one.
# Warning: this will save 400 png images.

for k in range(400):
    rank_k_approx = trans[:, :k].dot(pca.components_[:k])
            + pca.mean_
    imsave('{:>03}'.format(str(k))
            + '.jpg', rank_k_approx.reshape(64, 64))
```

Acknowledgements

Thank you!