# Matrix Decomposition for Dimensionality Reduction

BY GUOKAN SHANG

Research Engineer @ Labs, Linagora &
PhD Candidate @ DaSciM, LIX

*Email:* guokan.shang@hotmail.com
*Web:* http://www.lix.polytechnique.fr/Labo/Guokan.Shang/

*July 27, 2017*

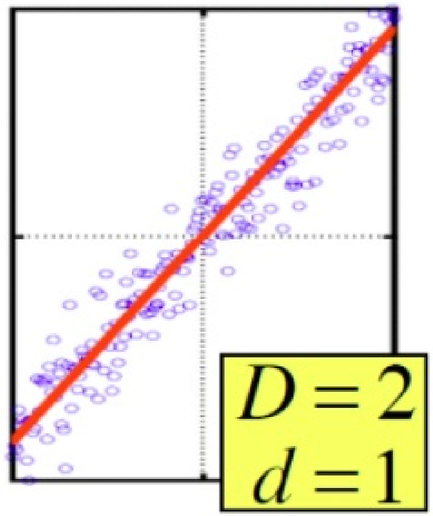
> Many algorithms that work fine in low dimensions become **intractable** when the input is high-dimensional.
>
> Bellman, 1961

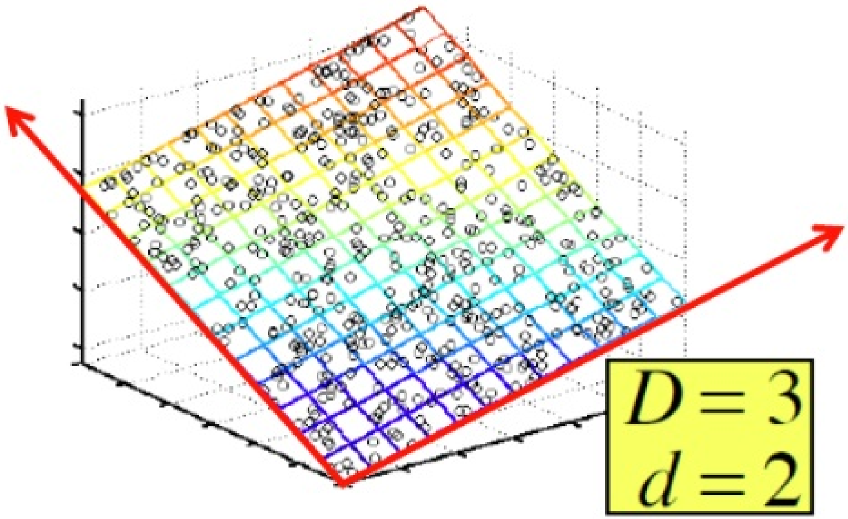
Real data usually have thousands or millions of dimensions,

- E.g. Documents, where the dimensionality is the vocabulary of words.

Huge number of dimensions causes problems,

- Data becomes very sparse, some algorithms become meaningless (e.g. density based clustering).
- The complexity of algorithms depends on the dimensionality and they become infeasible.




- Assumption: Data **reside** in a low $d$-dimensional **subspace**, axes of this subspace are **effective** representation of the data, then data can be represented by these axes without losing much of the meaning of the original data.
- Objectives: Discover hidden correlations/topics, Remove redundant and noisy features, Interpretation and visualization, Easier storage and processing of the data.

***Dimensionality reduction*** is the process of reducing the number of random variables (features) under consideration, via obtaining a set of principal variables (features). Approches can be divided into:

***Feature selection***

Try to find a subset of the original features. There are three strategies: ***filter*** (e.g. information gain) and ***wrapper*** (e.g. search guided by accuracy) approaches, and ***embedded*** (features are selected to add or be removed while building the model based on the prediction errors).

***Feature extraction***

Transforms the data in the high-dimensional space to a space of **fewer** dimensions. The data transformation may be **linear**, as in principal component analysis (**PCA**), but many **nonlinear** dimensionality reduction techniques also exist, as in t-distributed stochastic neighbor embedding (t-SNE).

For a square $n \times n$ matrix $A$, if there is a pair of (**nonzero unit** vector $\boldsymbol{x}$, scalar $\lambda$),

$$A\boldsymbol{x} = \lambda\boldsymbol{x} \qquad \|\boldsymbol{x}\| = 1$$
$$(A - \lambda I)\boldsymbol{x} = \mathbf{0}$$

such a $\lambda$ is called an ***eigenvalue***, $\boldsymbol{x}$ is called an ***unit eigenvector*** **corresponding** to $\lambda$.

Suppose a square $n \times n$ matrix $A$ has $k$ ***eigenpairs*** $(\boldsymbol{x}_1, \lambda_1), (\boldsymbol{x}_2, \lambda_2), \ldots, (\boldsymbol{x}_k, \lambda_k)$, and eigenvectors $\boldsymbol{x}_{1,\ldots,}\boldsymbol{x}_k$ are linearly independent, then ***eigen-decomposition*** of $A$,

$$A = X \Lambda X^{-1}$$

where $X$ is the ***eigenvector matrix*** $(\boldsymbol{x}_1 \ \boldsymbol{x}_2 \ , \ldots, \ \boldsymbol{x}_k)$.

where $\Lambda$ is the ***eigenvalue matrix*** $\begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & ,\ldots, & \\ & & & \lambda_k \end{pmatrix}$.

**Example 1.** $A = \begin{pmatrix} 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1\\ 1&1&1&1&1&1 \end{pmatrix}$, $A = 1 \times \begin{pmatrix} 1\\1\\1\\1\\1\\1 \end{pmatrix} \begin{pmatrix} 1&1&1&1&1&1 \end{pmatrix} = 1 \times \boldsymbol{p}_1 \boldsymbol{q}_1^T$

**Example 2.** $A = \begin{pmatrix} 1&1&1&0\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix}$, $A = 1 \times \begin{pmatrix} 1\\1\\1\\1\\1\\1 \end{pmatrix} \begin{pmatrix} 1&1&1&1 \end{pmatrix} + (-1) \times \begin{pmatrix} 1\\0\\0\\0\\0\\0 \end{pmatrix} \begin{pmatrix} 0&0&0&1 \end{pmatrix} =$
$1 \times \boldsymbol{p}_1 \boldsymbol{q}_1^T + (-1) \times \boldsymbol{p}_2 \boldsymbol{q}_2^T$

***low-rank approximation*** $\hat{A} \approx 1 \times \boldsymbol{p}_1 \boldsymbol{q}_1^T = 1 \times \begin{pmatrix} 1\\1\\1\\1\\1\\1 \end{pmatrix} \begin{pmatrix} 1&1&1&1 \end{pmatrix} = \begin{pmatrix} 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix}$,

Frobenius norm: $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2}$ Reconstruction error: $\|A - \hat{A}\|_F = 1$

Any matrix can be decomposed into simple pieces $e, \boldsymbol{p}, \boldsymbol{q}^T$

$$A = e_1 \boldsymbol{p_1} \boldsymbol{q_1^T} + e_2 \boldsymbol{p_2} \boldsymbol{q_2^T} = \begin{pmatrix} \boldsymbol{p_1} & \boldsymbol{p_2} \end{pmatrix} \begin{pmatrix} e_1 & 0 \\ 0 & e_2 \end{pmatrix} \begin{pmatrix} \boldsymbol{q_1^T} \\ \boldsymbol{q_2^T} \end{pmatrix} = PEQ^T$$

SVD choices **special** $PEQ^T$ called by $U\Sigma V^T$, which gives an **exact** decomposition, and will produce a ***smaller*** second piece $e_2 \boldsymbol{p_2} \boldsymbol{q_2^T}$, meaning after droping it, reconstruction error will be smaller.

$$\begin{aligned}
A &= U\Sigma V^T \\
A^T A &= (U\Sigma V^T)^T U\Sigma V^T = (V^{T^T}\Sigma^T U^T)U\Sigma V^T && \Sigma \text{ is diagonal} \\
A^T A &= (V\Sigma U^T)U\Sigma V^T && \text{Suppose } U \text{ is orthonormal} \\
A^T A &= V\Sigma^2 V^T && \text{Suppose } V \text{ is orthonormal} \\
A^T A V &= V\Sigma^2 \\
V \text{ is the orthonormal matrix of eigenvectors of } & A^T A \\
\Sigma^2 \text{ is the diagonal matrix of eigenvalue of } & A^T A \\
A^T A \text{ is symmetric matrix}
\end{aligned}$$

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

$$AA^T = U\Sigma V^T (U\Sigma V^T)^T = U\Sigma V^T (V^{T^T} \Sigma^T U^T) \qquad \Sigma \text{ is diagonal}$$

$$AA^T = U\Sigma V^T (V\Sigma U^T) \quad \text{Suppose } V \text{ is orthonormal}$$

$$AA^T = U\Sigma^2 U^T \quad \text{Suppose } U \text{ is orthonormal}$$

$$AA^T U = U\Sigma^2$$

$U$ is the orthonormal matrix of eigenvectors of $AA^T$

$\Sigma^2$ is the diagonal matrix of eigenvalue of $AA^T$

$AA^T$ is symmetric matrix

$$\boxed{A = U\Sigma V^T = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 \end{pmatrix} \begin{pmatrix} \sigma_{11} & 0 \\ 0 & \sigma_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \end{pmatrix} = \sigma_1 \boldsymbol{u}_1 \boldsymbol{v}_1^T + \sigma_2 \boldsymbol{u}_2 \boldsymbol{v}_2^T}$$

$U$ is left-singularvector orthonormal matrix, eigenvector matrix of $AA^T U = U\Sigma^2$

$V$ is right-singularvector orthonormal matrix, eigenvector matrix of $A^T A V = V\Sigma^2$

$\Sigma$ is singularvalue diagnal matrix, where $\sigma_{ii} > \sigma_{jj}$, when $i > j$

$$A = U\Sigma V^T \quad AV = U\Sigma$$

The singularvalue theorem for any (rectangular) matrix is the eigenvalue theorem for square matrix.

**Example 3.**

```
>>> import numpy as np
>>> A = np.ones((6,4)); A[0][3] = 0; print A
[[ 1.  1.  1.  0.]
 [ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]
 [ 1.  1.  1.  1.]]
>>> U, s, V = np.linalg.svd(A, full_matrices=False)
```

# SVD Example





```
>>> U
[[ -3.27881790e-01   9.44718758e-01  -2.23946720e-16   3.90867419e-17]
 [ -4.22491073e-01  -1.46633194e-01   8.94427191e-01   5.14994127e-17]
 [ -4.22491073e-01  -1.46633194e-01  -2.23606798e-01   8.6602ca5404e-01]
 [ -4.22491073e-01  -1.46633194e-01  -2.23606798e-01  -2.88675135e-01]
 [ -4.22491073e-01  -1.46633194e-01  -2.23606798e-01  -2.88675135e-01]
 [ -4.22491073e-01  -1.46633194e-01  -2.23606798e-01  -2.88675135e-01]]
>>> np.diag(s)
[[  4.72527289e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   8.19631677e-01   0.00000000e+00   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   4.02445156e-17   0.00000000e+00]
 [  0.00000000e+00   0.00000000e+00   0.00000000e+00   4.20691660e-33]]
>>> V
[[ -5.16443644e-01  -5.16443644e-01  -5.16443644e-01  -4.47054680e-01]
 [  2.58107140e-01   2.58107140e-01   2.58107140e-01  -8.94506631e-01]
 [ -6.43572054e-01   7.56942618e-01  -1.13370564e-01   6.10622664e-16]
 [  5.02475549e-01   3.06111973e-01  -8.08587523e-01  -7.69597298e-16]]
>>> np.allclose(A, np.dot(U, np.dot(np.diag(s), V)))
True
```

```
>>> np.diag([s[0],0,0,0])
```

```
[[ 4.72527289  0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 0.          0.          0.          0.        ]]
```

```
>>> A_prime = np.dot(U, np.dot(np.diag([s[0],0,0,0]), V)); print A_prime
```

```
[[ 0.80014211  0.80014211  0.80014211  0.69263564]
 [ 1.03102066  1.03102066  1.03102066  0.89249353]
 [ 1.03102066  1.03102066  1.03102066  0.89249353]
 [ 1.03102066  1.03102066  1.03102066  0.89249353]
 [ 1.03102066  1.03102066  1.03102066  0.89249353]
 [ 1.03102066  1.03102066  1.03102066  0.89249353]]
```

```
>>> np.linalg.norm(A - A_prime)
```

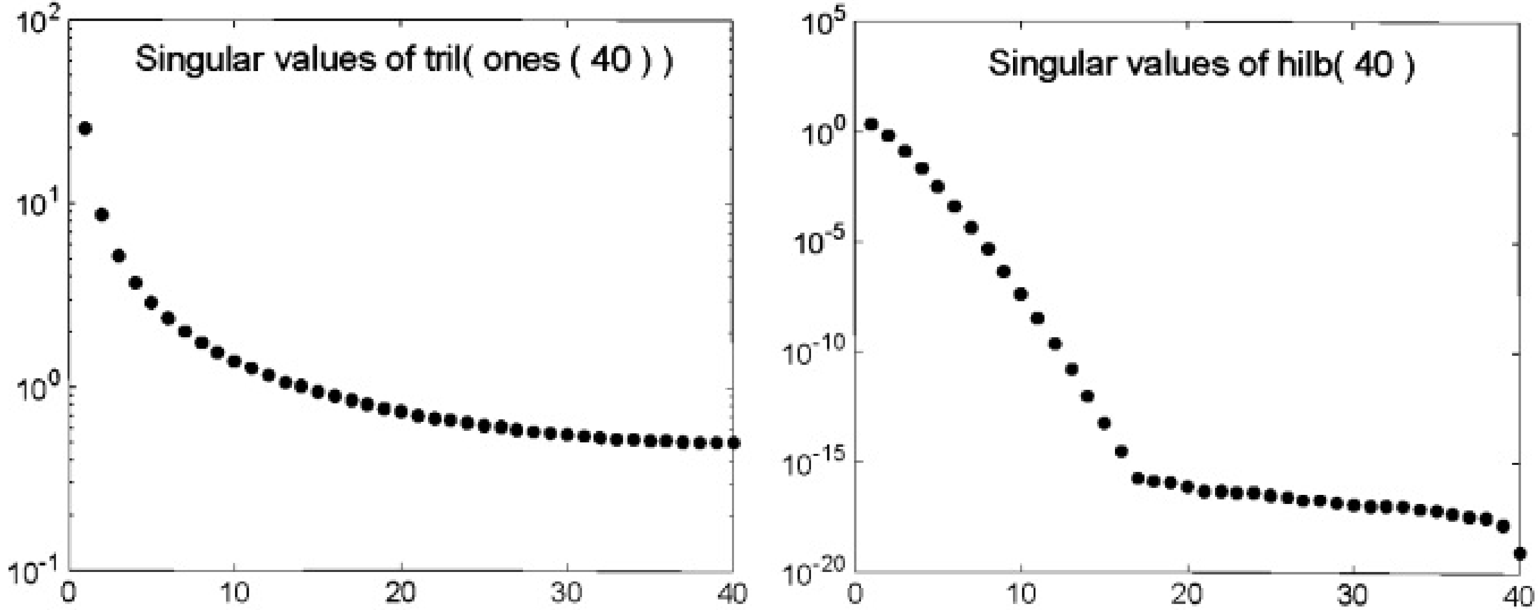
```
0.819631677125
```

$$A - A' = U(\Sigma - \Sigma')V^T \quad \|A\|_F = \sqrt{\sum_{ij} a_{ij}^2} = \text{trace}(A^T A) = \sqrt{\sum \sigma_i^2}$$

$$\text{Reconstruction Error } \|A - A'\|_F^2 = \text{trace}((\Sigma - \Sigma')(\Sigma - \Sigma')^T)$$

Two matrix: lower triangular matrix of 1 (left), and Hilbert matrix (right): $H(i,j) = (i+j-1)^{-1}$, plot the $n$ singular values.

Singular values of triangular drop off not deep, so the SVD gives only moderate compression of this triangular, but great compression for Hilbert.




According the rule-of-thumb, keep 80-90% of *energy* $\Sigma_i \sigma_i^2$ (related to PCA)

# Lena Image Compression with SVD



Original

60 singular values

100 singular values

120 singular values

Given a rank-2 matrix representing ratings of movies by users. In this example, there are two "**concepts**" underlying the movies: science-fiction and romance. All the boys rate only science-fiction, and all the girls rate only romance.

| | Matrix | Alien | Star Wars | Casablanca | Titanic |
|---|---|---|---|---|---|
| Joe | 1 | 1 | 1 | 0 | 0 |
| Jim | 3 | 3 | 3 | 0 | 0 |
| John | 4 | 4 | 4 | 0 | 0 |
| Jack | 5 | 5 | 5 | 0 | 0 |
| Jill | 0 | 0 | 0 | 4 | 4 |
| Jenny | 0 | 0 | 0 | 5 | 5 |
| Jane | 0 | 0 | 0 | 2 | 2 |

$$\underset{M}{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}} = \underset{U}{\begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}} \underset{V^{\mathrm{T}}}{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}$$

The key to understanding what SVD offers is in viewing the $r$ columns of $U$, $\Sigma$, and $V$ as representing concepts that are hidden in the original matrix M.

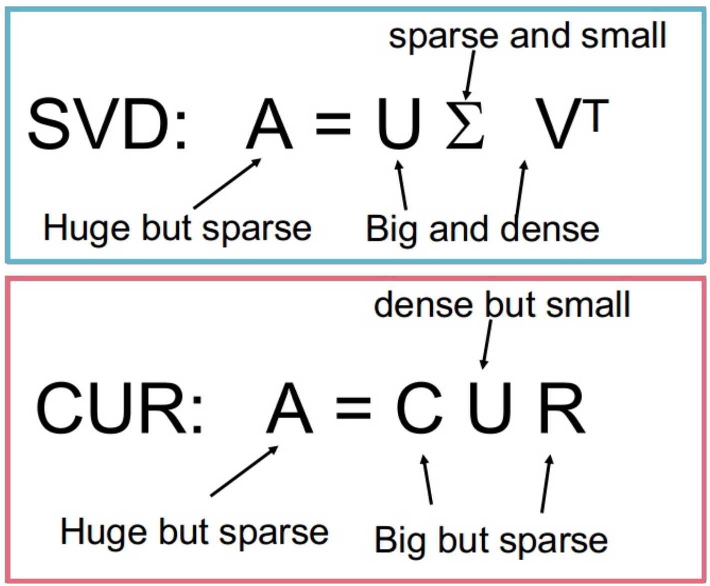
$U_{m \times r}$: user-to-concept matrix.

$V_{n \times r}$: movie-to-concept matrix.

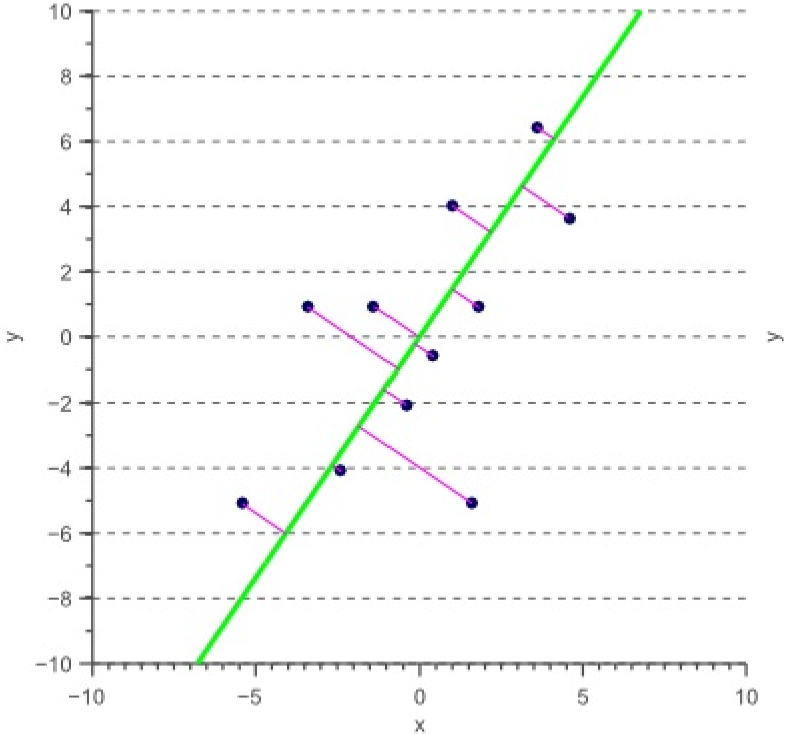
$\Sigma_{r \times r}$: its diagonal elements represent of each concept.

Interpretability problem: “concepts” are not always **semantically interpretable**, a singular vector specifies a linear combination of all input columns or rows.

Lack of sparsity: $U$ and $V$ are dense, inapplicable to large-scale case, this leads us to ***CUR***-decomposition.




SVD is limited to linear projections: lower-dimensional linear projection which preserves Euclidean distances. **Isomap**: a nonlinear dimensionality reduction method.




***Principal Component Analysis*** (PCA), is a technique for taking a dataset consisting of a set of tuples representing points in a high-dimensional space and finding the directions along which the tuples **line up best**.

When you apply this transformation to the original data, the principal component axis is the one along which the points are most "**spread out**".

More precisely, this axis is the one along which the **variance** of the data is maximized.

$$\mathrm{var}(X) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})}{n-1}$$
$$\mathrm{cov}(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

if $\mathrm{cov}(X, Y) > 0$, positive correlation, indicates that both dimensions increase together.

if $\mathrm{cov}(X,Y) < 0$, negative correlation, indicates that as one dimension increases, the other decreases.

if the $\mathrm{cov}(X,Y) = 0$, it indicates that the two dimensions are independent of each other

**Covariance Matrix** for a 3 dimensional data set,

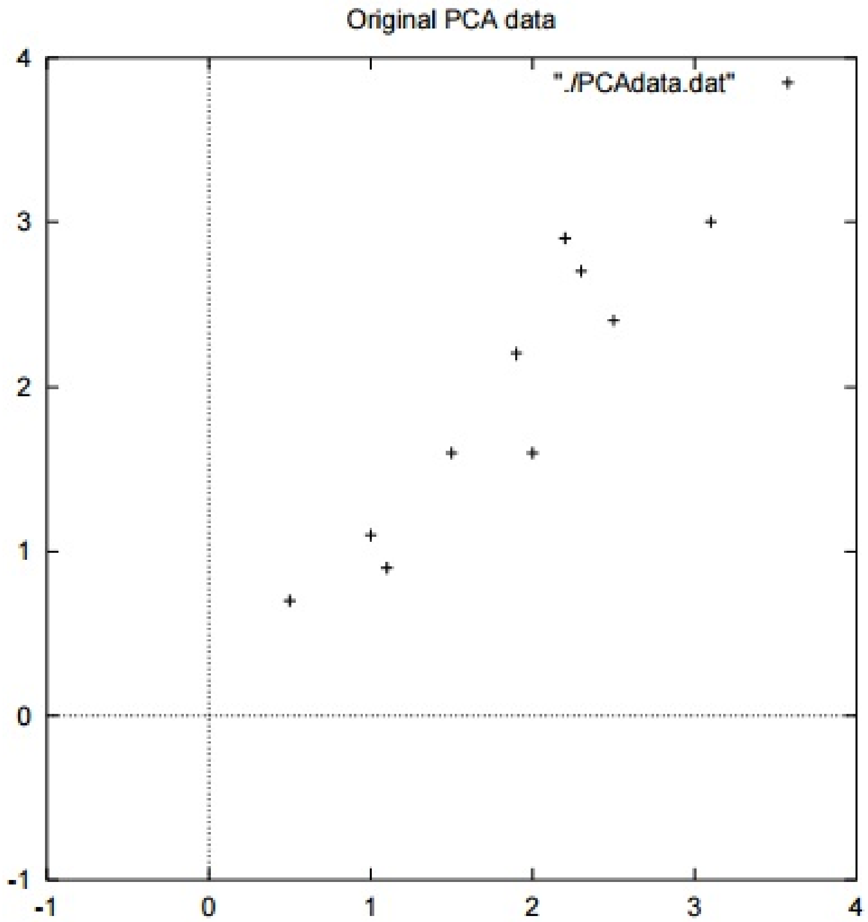
$$C = \begin{pmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,Z) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{pmatrix}$$

Data =

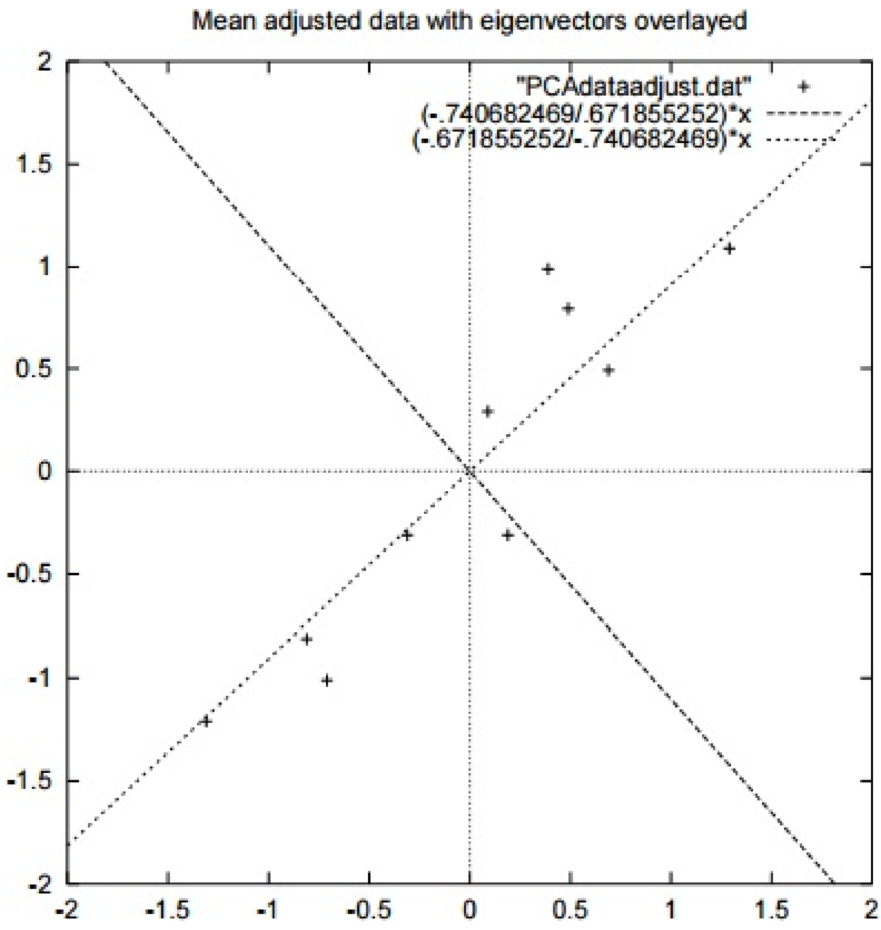
| $x$ | $y$ |
|---|---|
| 2.5 | 2.4 |
| 0.5 | 0.7 |
| 2.2 | 2.9 |
| 1.9 | 2.2 |
| 3.1 | 3.0 |
| 2.3 | 2.7 |
| 2 | 1.6 |
| 1 | 1.1 |
| 1.5 | 1.6 |
| 1.1 | 0.9 |

DataAdjust =

| $x$ | $y$ |
|---|---|
| .69 | .49 |
| -1.31 | -1.21 |
| .39 | .99 |
| .09 | .29 |
| 1.29 | 1.09 |
| .49 | .79 |
| .19 | -.31 |
| -.81 | -.81 |
| -.31 | -.31 |
| -.71 | -1.01 |




$$C = \begin{pmatrix} .61655556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$




***The eigenvector of covariance matrix with the highest eigenvalue is the principle component of the data set.*** (which is the direction which has the maximum variance of this data)

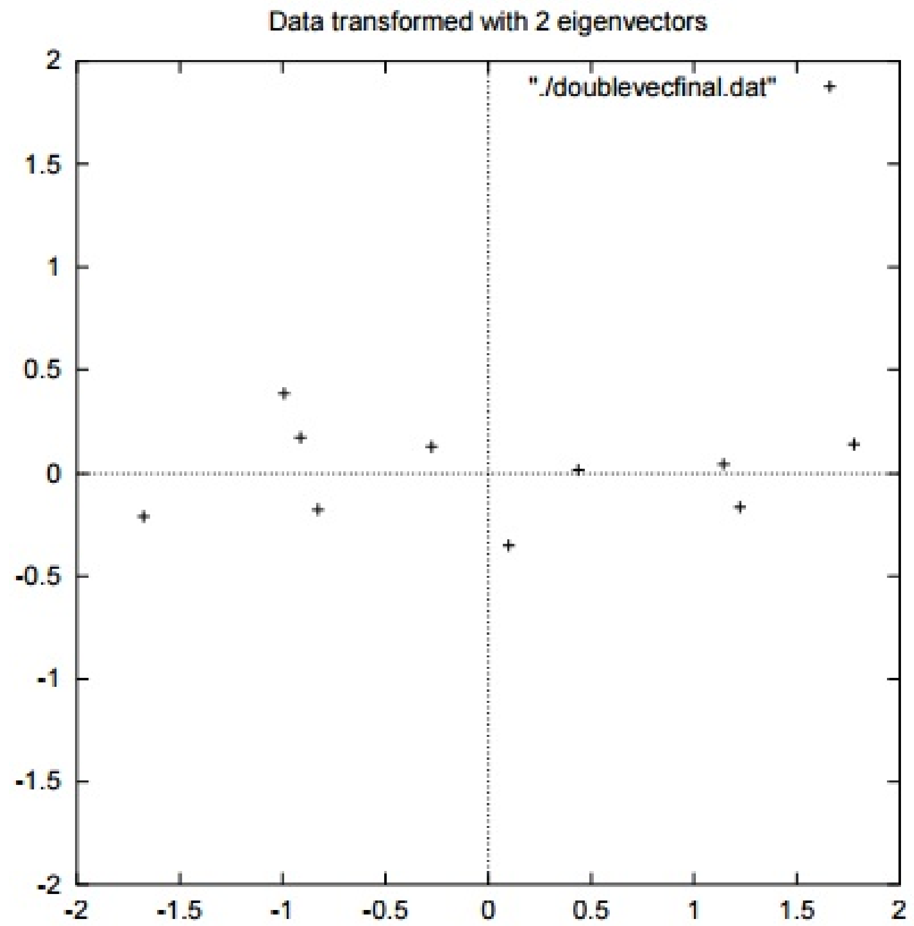
$$\text{eigenvalues} = \begin{pmatrix} 1.28402771 \\ .0490833989 \end{pmatrix}$$

$$\text{unit eigenvectors} = \begin{pmatrix} -.677873399 & -.735178656 \\ -.735178656 & .677873399 \end{pmatrix}$$

# PCA Example

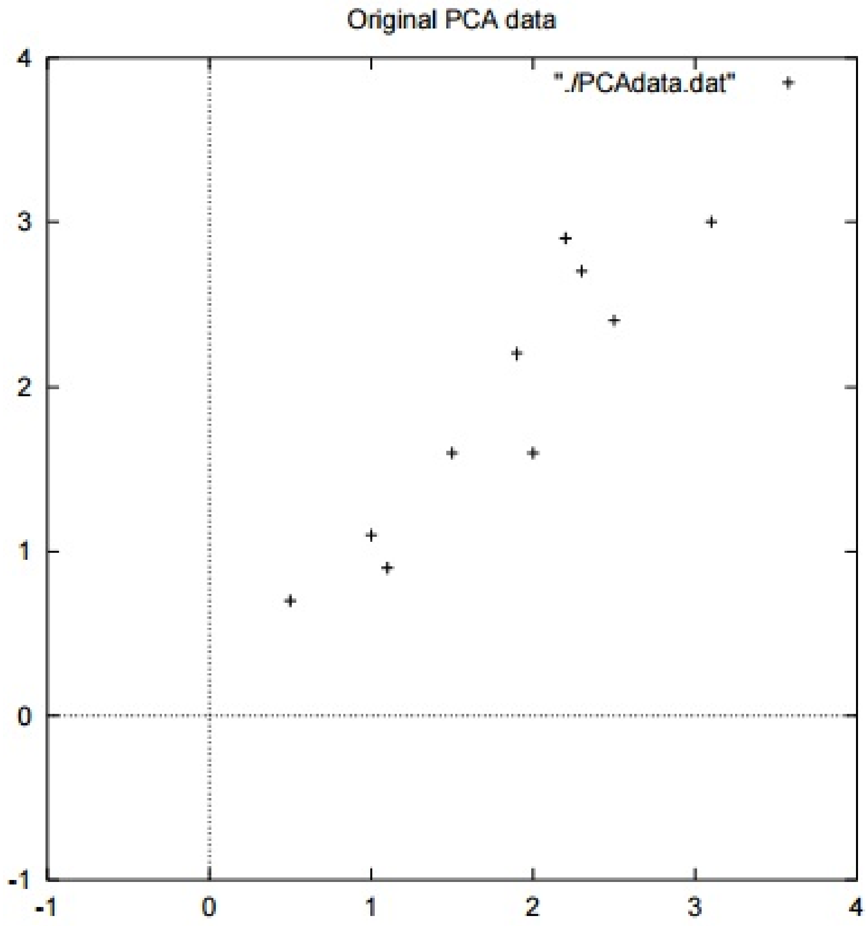
Transformed Data=

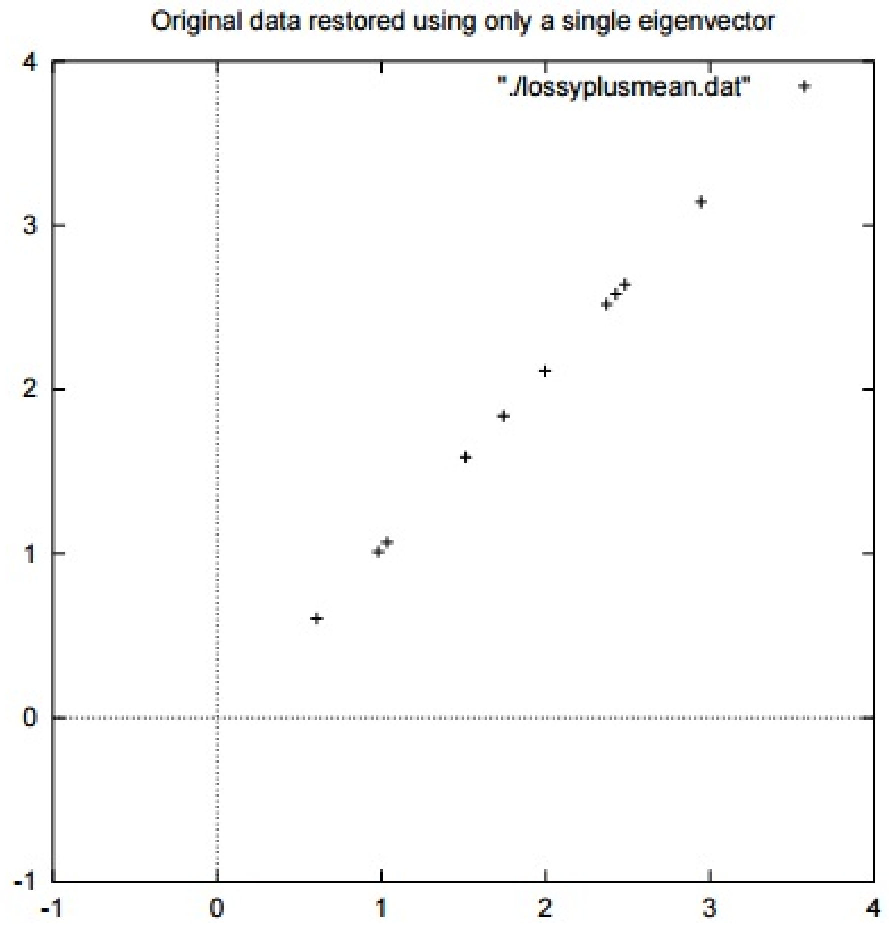
| $x$ | $y$ |
|---|---|
| -.827970186 | -.175115307 |
| 1.77758033 | .142857227 |
| -.992197494 | .384374989 |
| -.274210416 | .130417207 |
| -1.67580142 | -.209498461 |
| -.912949103 | .175282444 |
| .0991094375 | -.349824698 |
| 1.14457216 | .0464172582 |
| .438046137 | .0177646297 |
| 1.22382056 | -.162675287 |



# PCA Example




Original PCA data
"./PCAdata.dat"
4
3
2
1
0
-1
-1
0
1
2
3
4


Original data restored using only a single eigenvector
"./lossyplusmean.dat"
4
3
2
1
0
-1
-1
0
1
2
3
4

# Maximize variance of the projection





$$C_{n\times n} = \begin{pmatrix} \mathrm{cov}(X,X) & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(Y,Z) & \mathrm{cov}(Y,Y) & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(Z,X) & \mathrm{cov}(Z,Y) & \mathrm{cov}(Z,Z) \end{pmatrix} = \frac{\bar{D}^T_{m\times n}\bar{D}_{m\times n}}{m-1}$$

Let $\boldsymbol{w}$ be a unit vector specifying an axis in the coloum feature space, we want $\boldsymbol{w}$ to be the first principal axis.

First principal axis maximizes the variance of the projection $\bar{D}_{m\times n}\boldsymbol{w}_{\boldsymbol{n\times 1}}$, (variance of the first principal component). This variance is given by the

$$\mathrm{var}(\bar{D}\boldsymbol{w}) = \frac{\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T\bar{D}\boldsymbol{w}}{m-1} = \boldsymbol{w}^{\boldsymbol{T}}C\boldsymbol{w}$$

Constrained Optimization for quadratic form $\boldsymbol{w}^{\boldsymbol{T}}C\boldsymbol{w}$ when $\|\boldsymbol{w}\| = 1$.

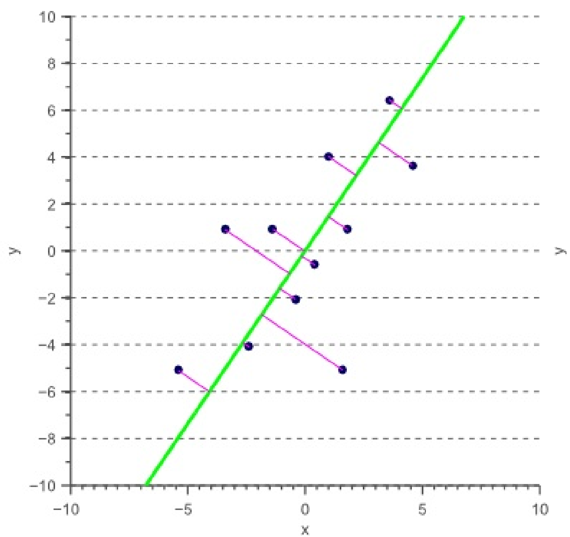
**Theorem 4.** *Let $A$ be a symmetric matrix, and define $m$ and $M$ as*

$$m = \min\{\boldsymbol{x}^T A \boldsymbol{x};\ \|\boldsymbol{x}\| = 1\},\ M = \max\{\boldsymbol{x}^T A \boldsymbol{x};\ \|\boldsymbol{x}\| = 1\}$$

*Then $M$ is the greatest eigenvalue $\lambda_1$ of $A$ and $m$ is the least eigenvalue of $A$ .*

*The value of $\boldsymbol{x}^T A \boldsymbol{x}$ is $M$ when $\boldsymbol{x}$ is a unit eigenvector $\boldsymbol{u}_1$ corresponding to $M$.*

*The value of $\boldsymbol{x}^T A \boldsymbol{x}$ is $m$ when $\boldsymbol{x}$ is a unit eigenvector corresponding to $m$.*



First principal axis minimizes the reconstruction error between $\bar{D}$ and its reconstruction $\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}$, i.e. the sum of squared distances between the original points and their projections onto w. The square of the reconstruction error is given by,

$$\begin{aligned}
\|\bar{D}-\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\|^2 =& \operatorname{trace}((\bar{D}-\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}})(\bar{D}-\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}})^T) \\
=& \operatorname{trace}((\bar{D}-\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}})(\bar{D}^T-\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T)) \\
=& \operatorname{trace}(\bar{D}\bar{D}^T)-2\operatorname{trace}(\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T)+\operatorname{trace}(\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T) \\
=& \text{const}-\operatorname{trace}(\bar{D}\boldsymbol{w}\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T) \\
=& \text{const}-\operatorname{trace}(\boldsymbol{w}^{\boldsymbol{T}}\bar{D}^T\bar{D}\boldsymbol{w}) \\
=& \text{const}-\text{const}\cdot\boldsymbol{w}^{\boldsymbol{T}}C\boldsymbol{w}
\end{aligned}$$

Notice the minus sign before the main term. Because of that, minimizing the reconstruction error amounts to maximizing $\boldsymbol{w}^{\boldsymbol{T}}C\boldsymbol{w}$, which is the variance.

So minimizing reconstruction error is equivalent to maximizing the variance; both formulations yield the same $\boldsymbol{w}$.

Origianl Data: $D_{m\times n}$, $m$ is the number of samples, $n$ is the number of features

Column centered Data: $\bar{D}_{m\times n}$, PCA transformed Data: $\tilde{D}_{m\times r}$, $\tilde{D}_{m\times k}$

1. $D_{m\times n} \rightarrow$ Covariance matrix $C_{n\times n} \rightarrow$ EigenDecomposition $C_{n\times n} = V_{n\times r}\Lambda_{r\times r}V_{n\times r}^T \rightarrow \tilde{D}_{m\times r} = \bar{D}_{m\times n}V_{n\times r} \rightarrow \tilde{D}_{m\times k}$
2. $D_{m\times n} \rightarrow \bar{D}_{m\times n} \rightarrow C_{n\times n} = \frac{\bar{D}_{m\times n}^T\bar{D}_{m\times n}}{m-1} \rightarrow$ EigenDecomposition $C_{n\times n} = V_{n\times r}\Lambda_{r\times r}V_{n\times r}^T \rightarrow \tilde{D}_{m\times r} = \bar{D}_{m\times n}V_{n\times r} \rightarrow \tilde{D}_{m\times k}$
3. $D_{m\times n} \rightarrow \bar{D}_{m\times n} \rightarrow$ SingularValueDecomposition $\bar{D}_{m\times n} = U_{m\times r}\Sigma_{r\times r}V_{n\times r}^T \rightarrow \tilde{D}_{m\times r} = \bar{D}_{m\times n}V_{n\times r} = U_{m\times r}\Sigma_{r\times r} \rightarrow \tilde{D}_{m\times k}$

$$\text{where} \min\{m,n\} \geqslant r \geqslant k$$

$$C_{n\times n} = \frac{\bar{D}^T_{m\times n}\bar{D}_{m\times n}}{m-1} = V_{n\times r}\Lambda_{r\times r}V^T_{n\times r}$$

$$C_{n\times n} = \frac{V'_{n\times r}\Sigma_{r\times r}U'^T_{m\times r}U'_{m\times r}\Sigma_{r\times r}V'^T_{n\times r}}{m-1}$$

$$= \frac{V'_{n\times r}\Sigma^2_{r\times r}V'^T_{n\times r}}{m-1}$$

$$= V'_{n\times r}\frac{\Sigma^2_{r\times r}}{m-1}V'^T_{n\times r}$$

$$\Rightarrow V'_{n\times r} = V_{n\times r} \text{ and } \Lambda_{r\times r} = \frac{\Sigma^2_{r\times r}}{m-1}$$

Right singularvectors $V$ of $\bar{D}$ are eigenvectors of covariance matrix $C$.

Singularvalues $\Sigma$ of $\bar{D}$ are related to the eigenvalues of covariance matrix $C$,

Eigenvalues $\lambda_i$ show variances of the respective PCs.

Principal components are given by,

$$\bar{D}^T_{m\times n}V_{n\times r} = U_{m\times r}\Sigma_{r\times r}V^T_{n\times r}V_{n\times r} = U_{m\times r}\Sigma_{r\times r}$$

$$\underset{M}{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}} = \underset{U}{\begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}} \underset{V^{\mathrm{T}}}{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}$$

$V$ is eigenvectors of covariance matrix $\bar{D}^T\bar{D}$, is a set of mutually orthonormal basis, maximize variance of the projection, used for reducing dimensionality alongside column.

$U$ is eigenvectors of covariance matrix of $\bar{D}\bar{D}^T$, is a set of mutually orthonormal basis, , maximize variance of the projection, used for reducing dimensionality alongside row.

$\Lambda_{r\times r} = \frac{\Sigma^2_{r\times r}}{m-1}$ show variances of the respective PCs (“concepts”).

The method of least squares is a way of "solving" an overdetermined inconsistent system of linear equations

$$w_0x_0 + w_1x_1^{(1)} + w_2x_2^{(1)} = y_1$$
$$w_0x_0 + w_1x_1^{(2)} + w_2x_2^{(2)} = y_2$$
$$w_0x_0 + w_1x_1^{(3)} + w_2x_2^{(3)} = y_3$$

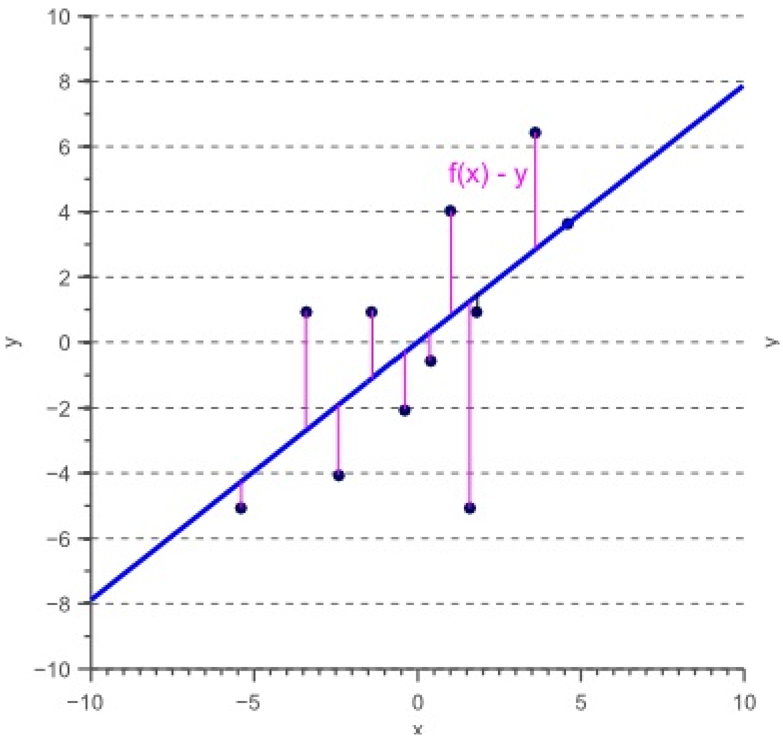
$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} x_0 & x_1^{(1)} & x_2^{(1)} \\ x_0 & x_1^{(2)} & x_2^{(2)} \\ x_0 & x_1^{(3)} & x_2^{(3)} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = w_0 \begin{pmatrix} x_0 \\ x_0 \\ x_0 \end{pmatrix} + w_1 \begin{pmatrix} x_1^{(1)} \\ x_1^{(2)} \\ x_1^{(3)} \end{pmatrix} + w_2 \begin{pmatrix} x_2^{(1)} \\ x_2^{(2)} \\ x_2^{(3)} \end{pmatrix}$$

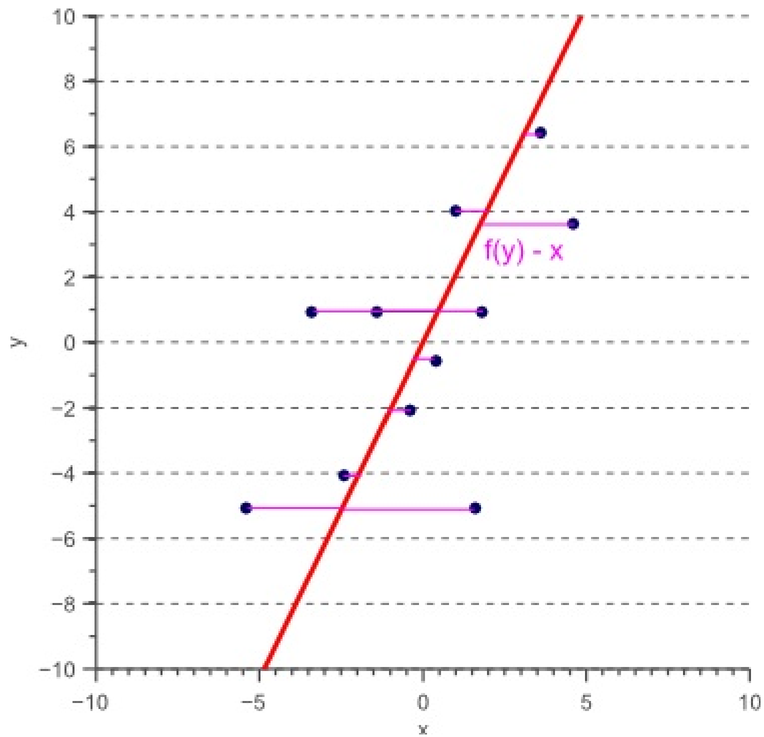
$$\boldsymbol{Xw} = \boldsymbol{y}$$

i.e., a system in which $\boldsymbol{X}$ is a rectangular $m \times n$ matrix with more equations than unknowns (when $m > n$). no solution

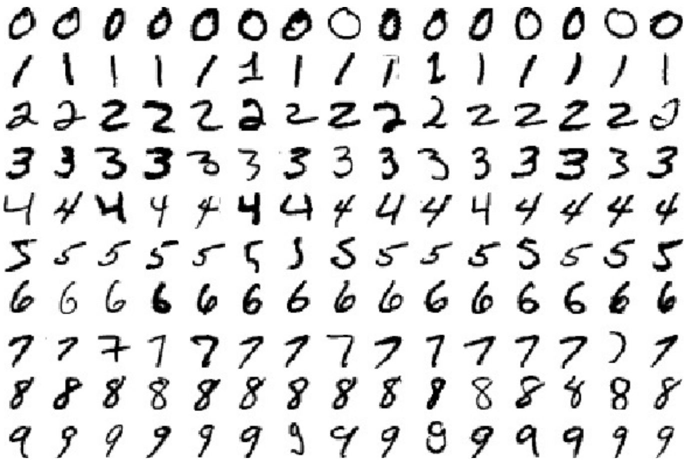
The idea is to determine $\hat{w}$, "least-squares solution" so that it minimizes the sum of the squares of the errors, namely $\|X\hat{w} - y\|^2$

$$X^T X\hat{w} = X^T y$$
$$\hat{w} = (X^T X)^{-1} X^T y$$



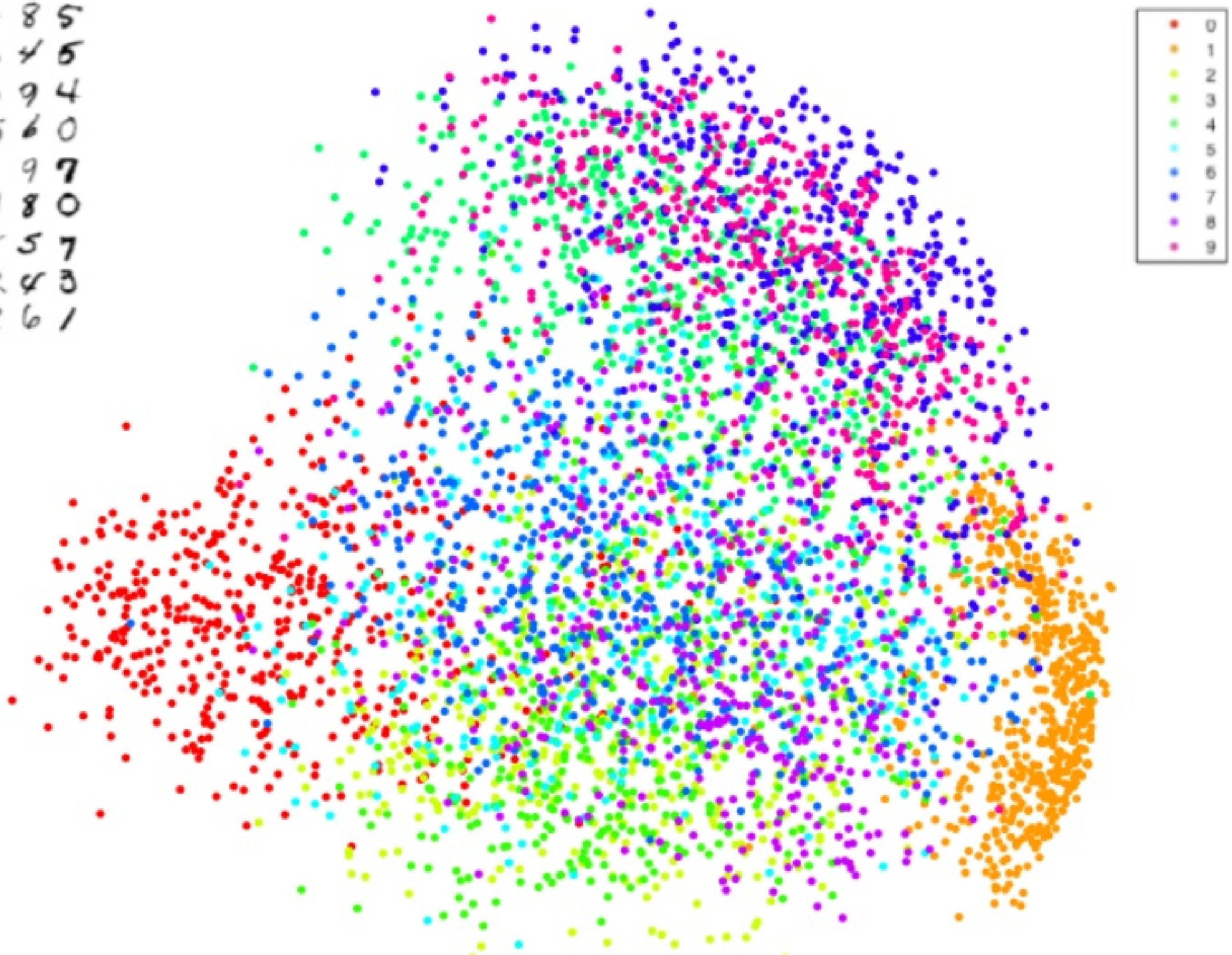



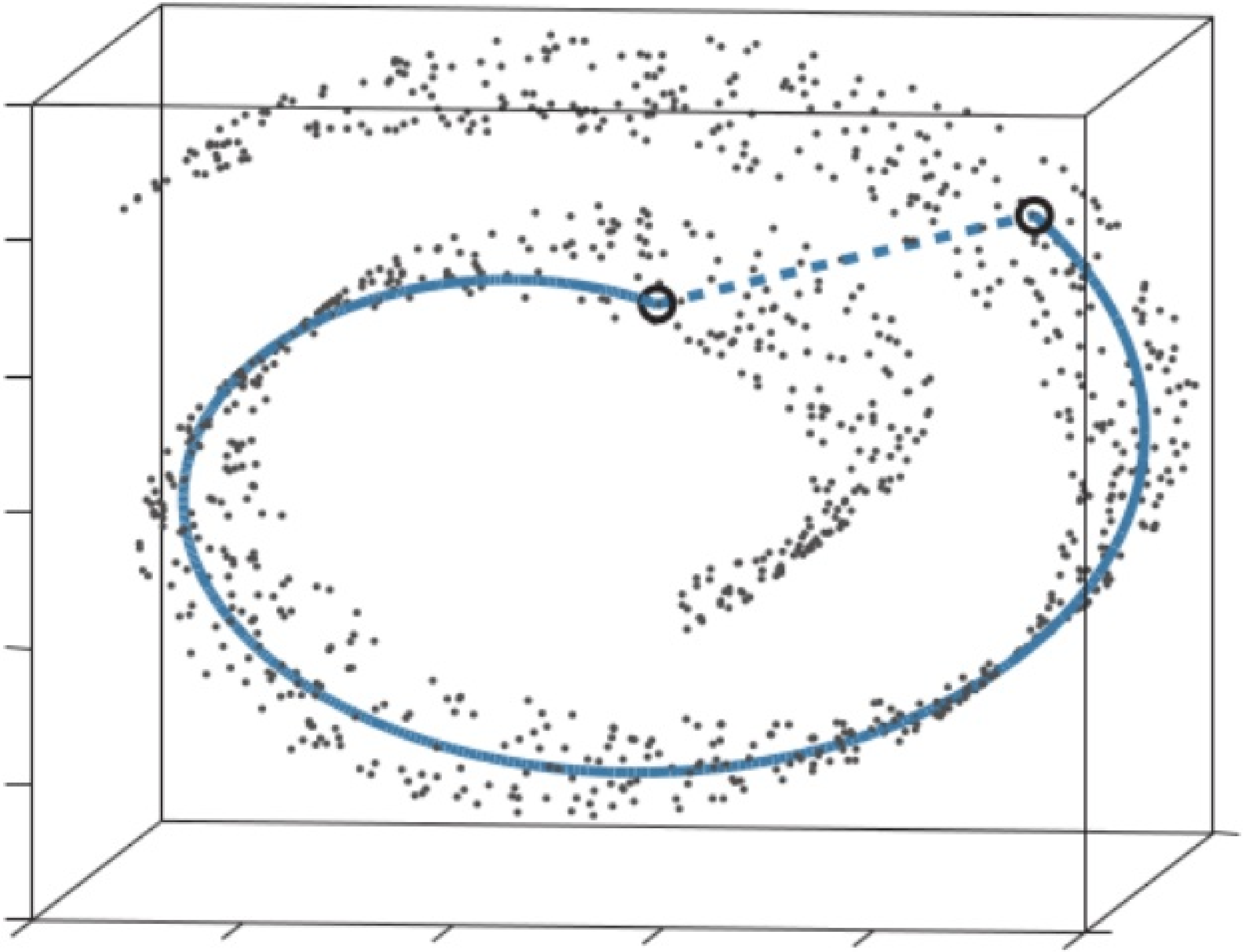
MNIST is a simple computer vision dataset. It consists of $28 \times 28$ pixel images of handwritten digits, such as:

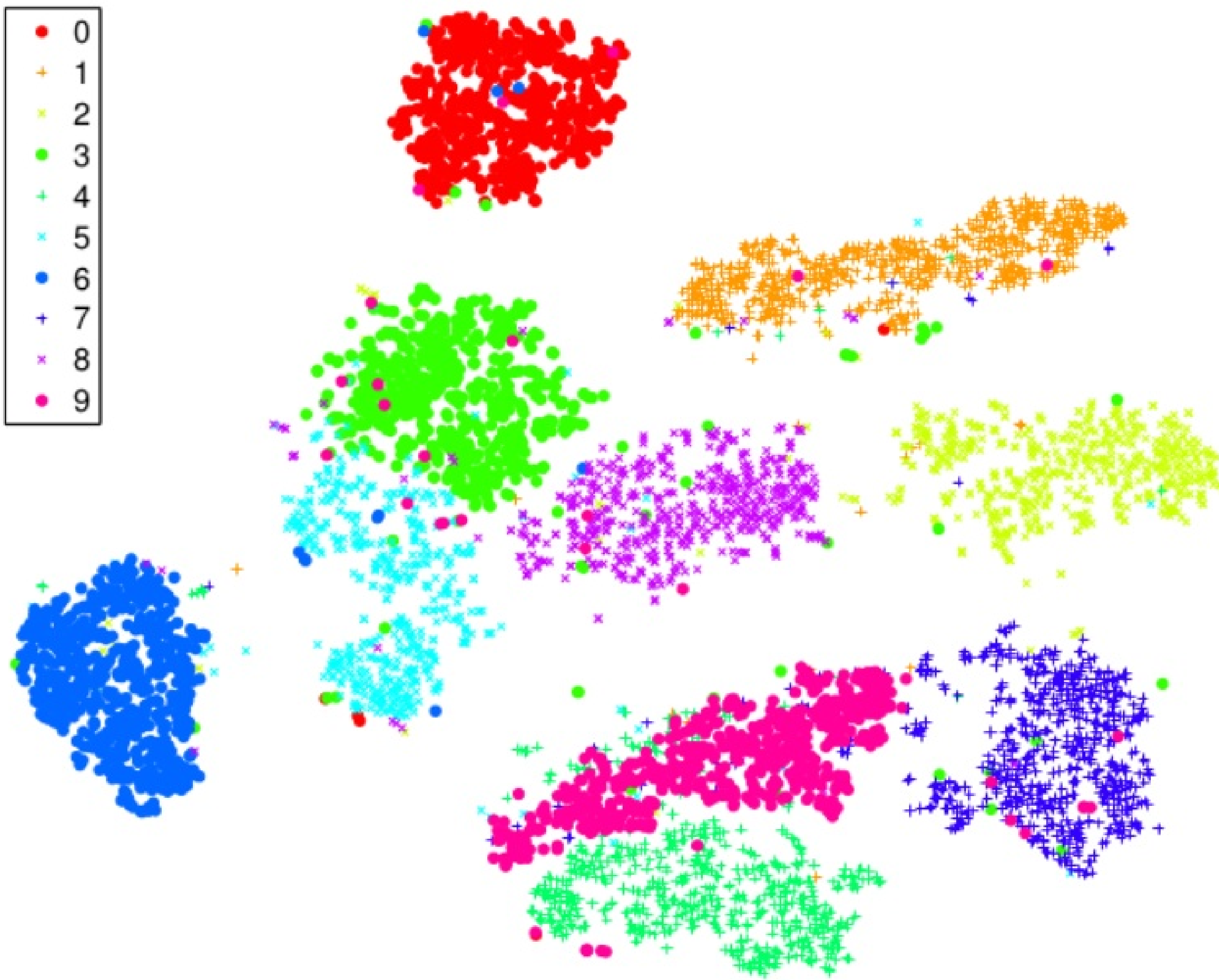


We flatten each array into a $28 \times 28$=748 dimensional vector. Each component of the vector is a value between zero and one describing the intensity of the pixel.

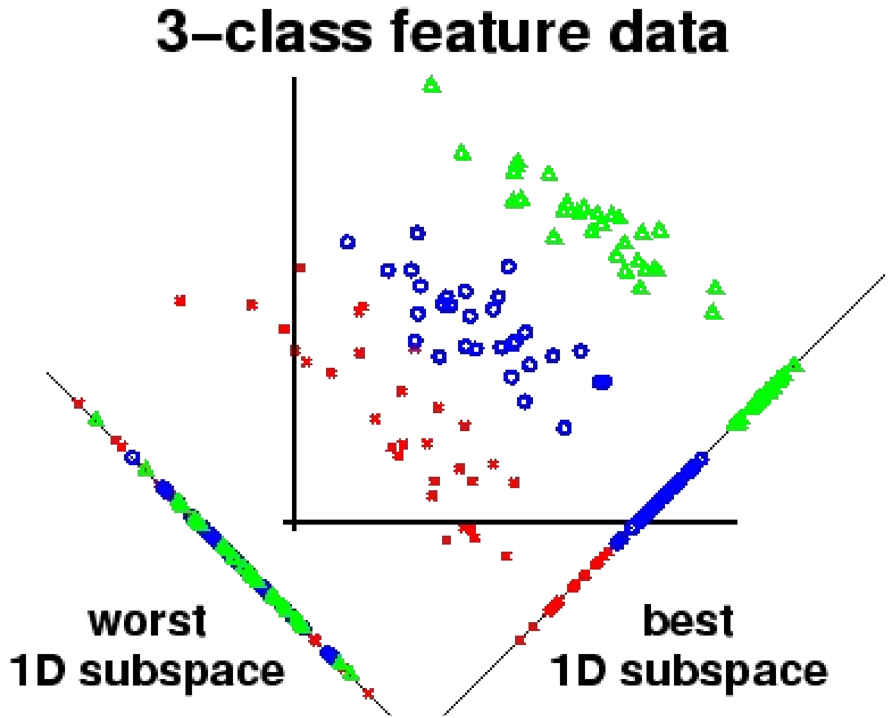
Images like MNIST digits are very rare in 748 dimensions. While the MNIST data points are embedded in 784-dimensional space, they live in a very small subspace. With some slightly harder arguments, we can see that they occupy a lower dimensional subspace.


0
1
2
3
4
5
6
7
8
9




0
1
2
3
4
5
6
7
8
9

Supervised Dimensionality Reduction,




PCA: Component axes that maximize the variance.

LDA: Maximizing the component axes for class-seperation.

Strang, G., Strang, G., Strang, G., & Strang, G. (1993). *Introduction to linear algebra* (Vol. 3). Wellesley, MA: Wellesley-Cambridge Press.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of massive datasets*. Cambridge university press. http://www.mmds.org

David C. Lay, Steven R. Lay, Judi J. McDonald (2015). *Linear Algebra and Its Applications.* Pearson.

https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/

http://www.cerebralmastication.com/2010/09/principal-component-analysis-pca-vs-ordinary-least-squares-ols-a-visual-explination/

# References



https://stats.stackexchange.com/questions/189822/how-does-centering-make-a-difference-in-pca-for-svd-and-eigen-decomposition

https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca

https://stats.stackexchange.com/questions/130721/what-norm-of-the-reconstruction-error-is-minimized-by-the-low-rank-approximation

https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues/140579

https://stats.stackexchange.com/questions/32174/pca-objective-function-what-is-the-connection-between-maximizing-variance-and-m/136072

http://setosa.io/ev/

http://projector.tensorflow.org/