Machine Learning

Data Compression

Dr. Mehran Safayani

safayani@iut.ac.ir

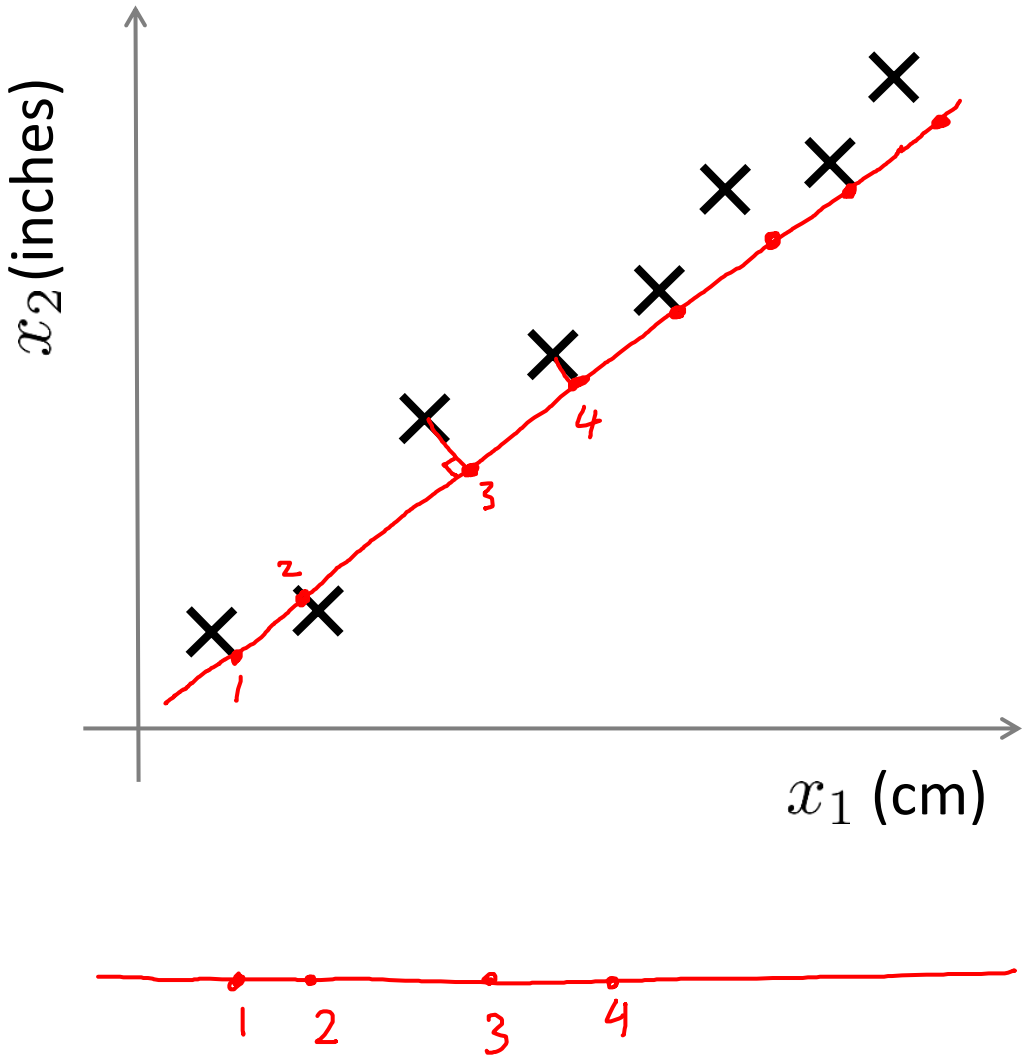
safayani.iut.ac.ir

https://www.aparat.com/mehran.safayani

https://github.com/safayani/machine_learning_course

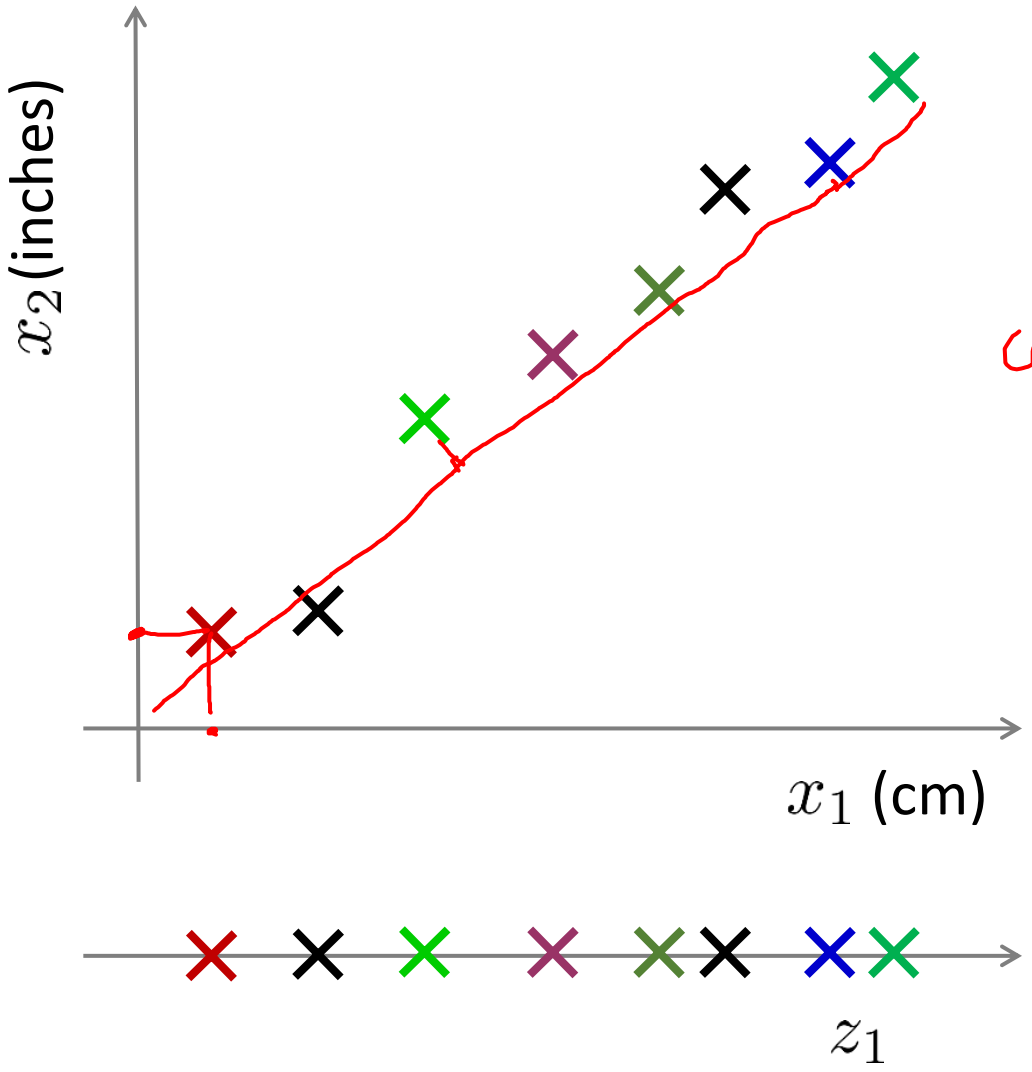
Department of Electrical and computer engineering, Isfahan university of technology, Isfahan, Iran

Data Compression



Reduce data from 2D to 1D

Data Compression



Reduce data from 2D to 1D

دوبعدی

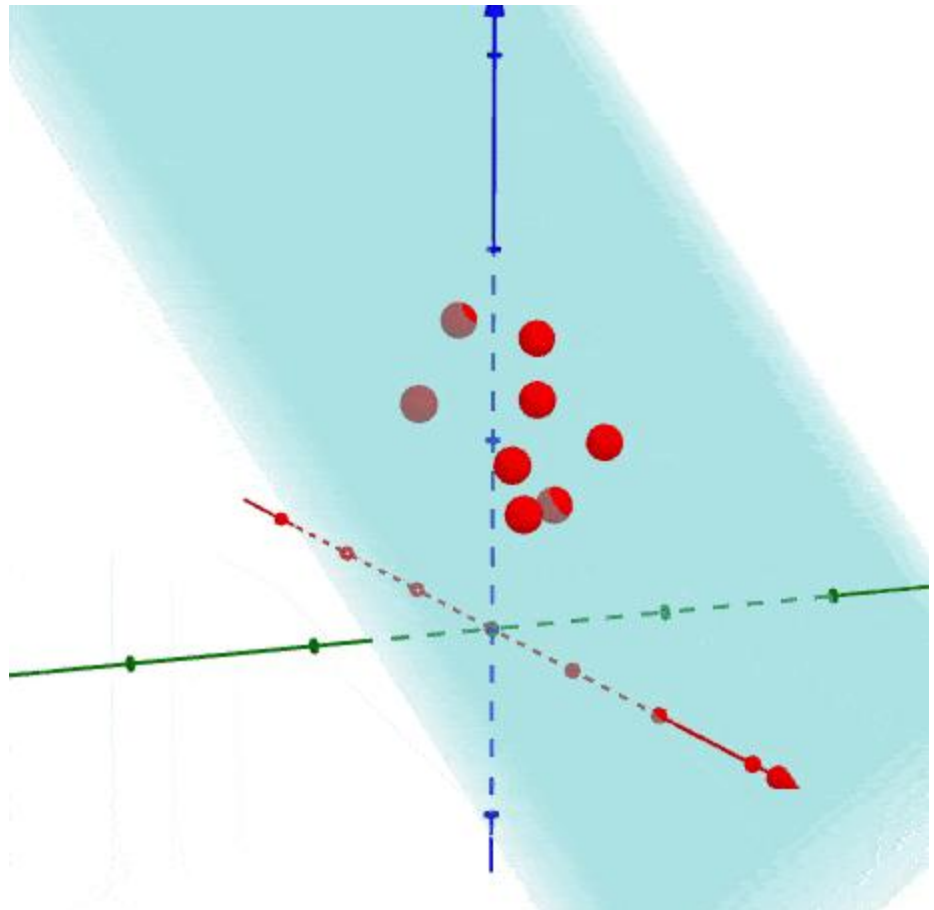
$$x^{(1)} \rightarrow z^{(1)}$$

یک بعدی

$$x^{(2)} \rightarrow z^{(2)}$$

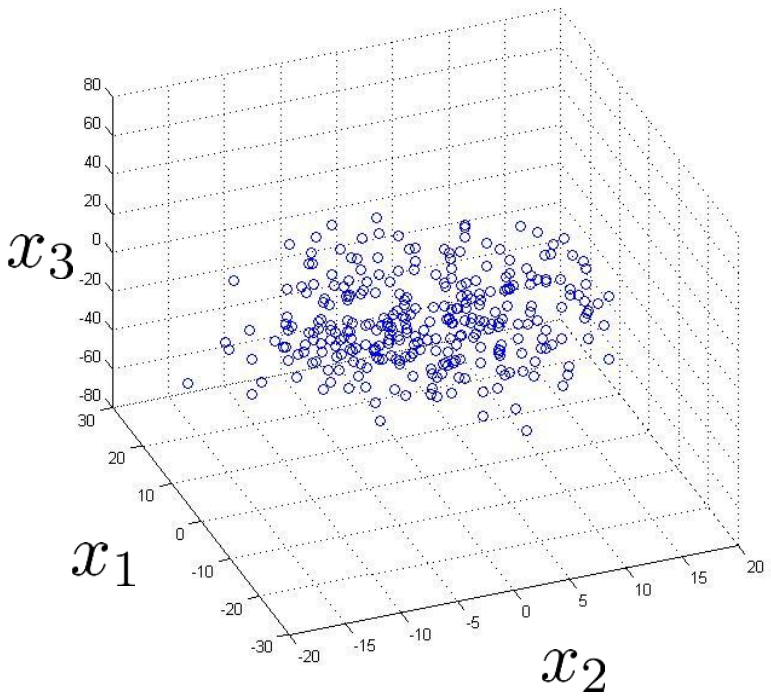
$$\vdots$$

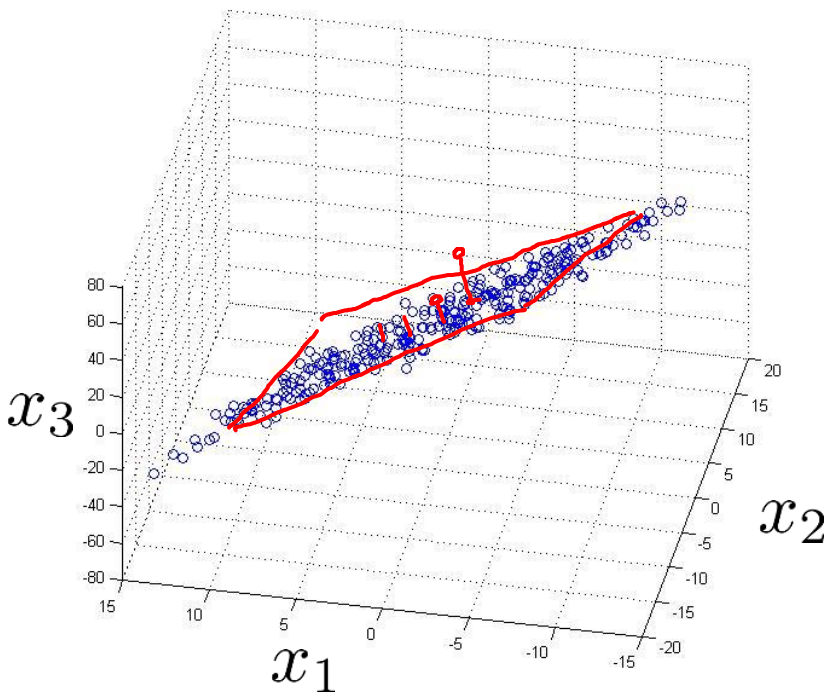
$$x^{(m)} \rightarrow z^{(m)}$$

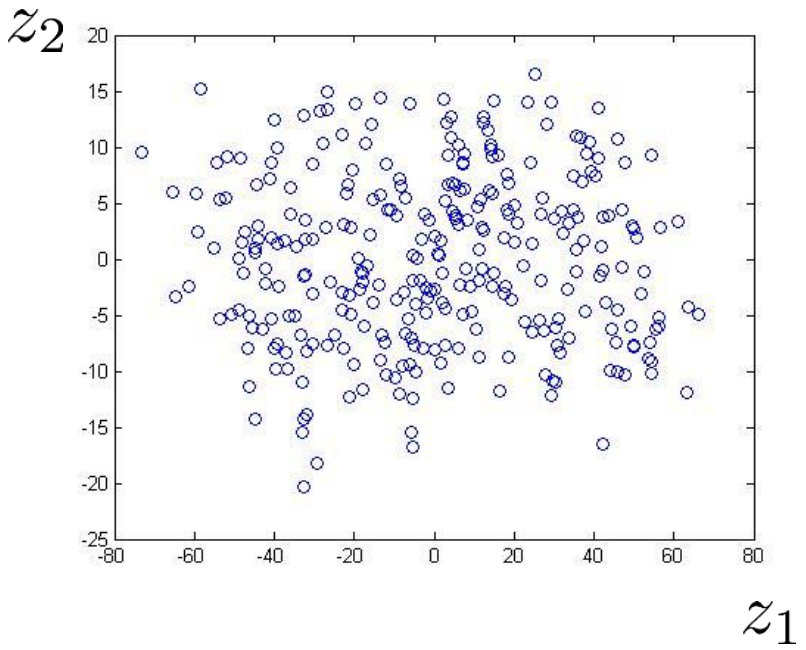


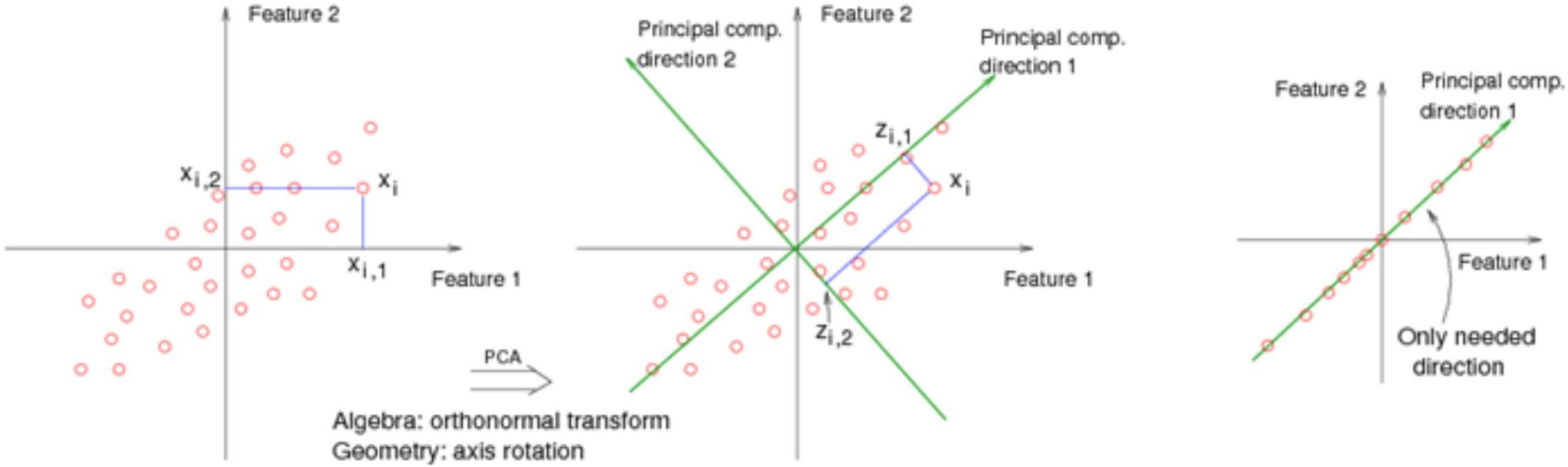
Data Compression

Reduce data from 3D to 2D









Algebra: orthonormal transform
Geometry: axis rotation

Data Visualization

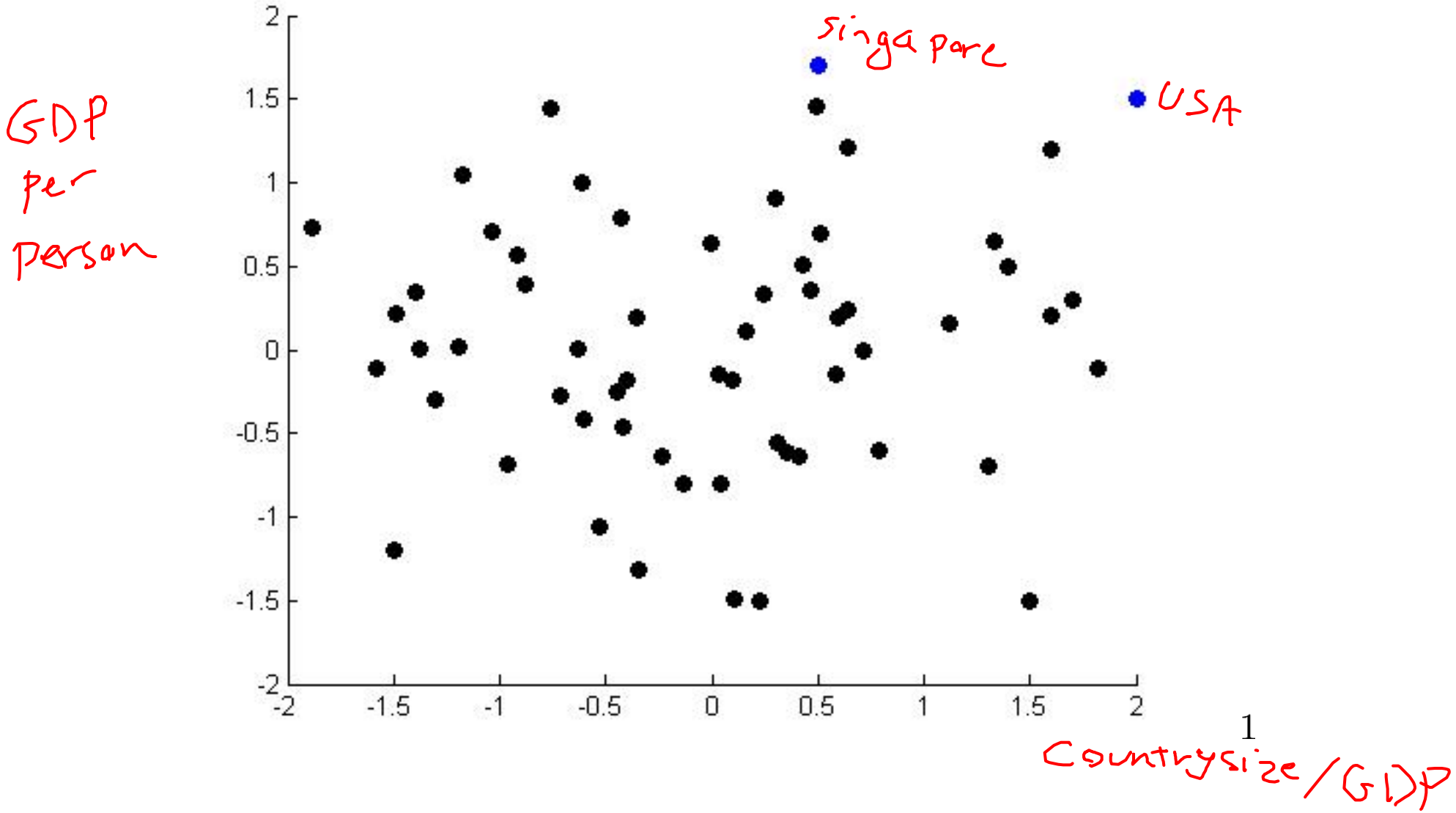
Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectanc y	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	…
Canada	1.577	39.17	0.908	80.7	32.6	67.293	…
China	5.878	7.54	0.687	73	46.9	10.22	…
India	1.632	3.41	0.547	64.7	36.8	0.735	…
Russia	1.48	19.84	0.755	65.5	39.9	0.72	…
Singapore	0.223	56.69	0.866	80	42.5	67.1	…
USA	14.527	46.86	0.91	78.3	40.8	84.3	…
…	…	…	…	…	…	…	

[resources from en.wikipedia.org]

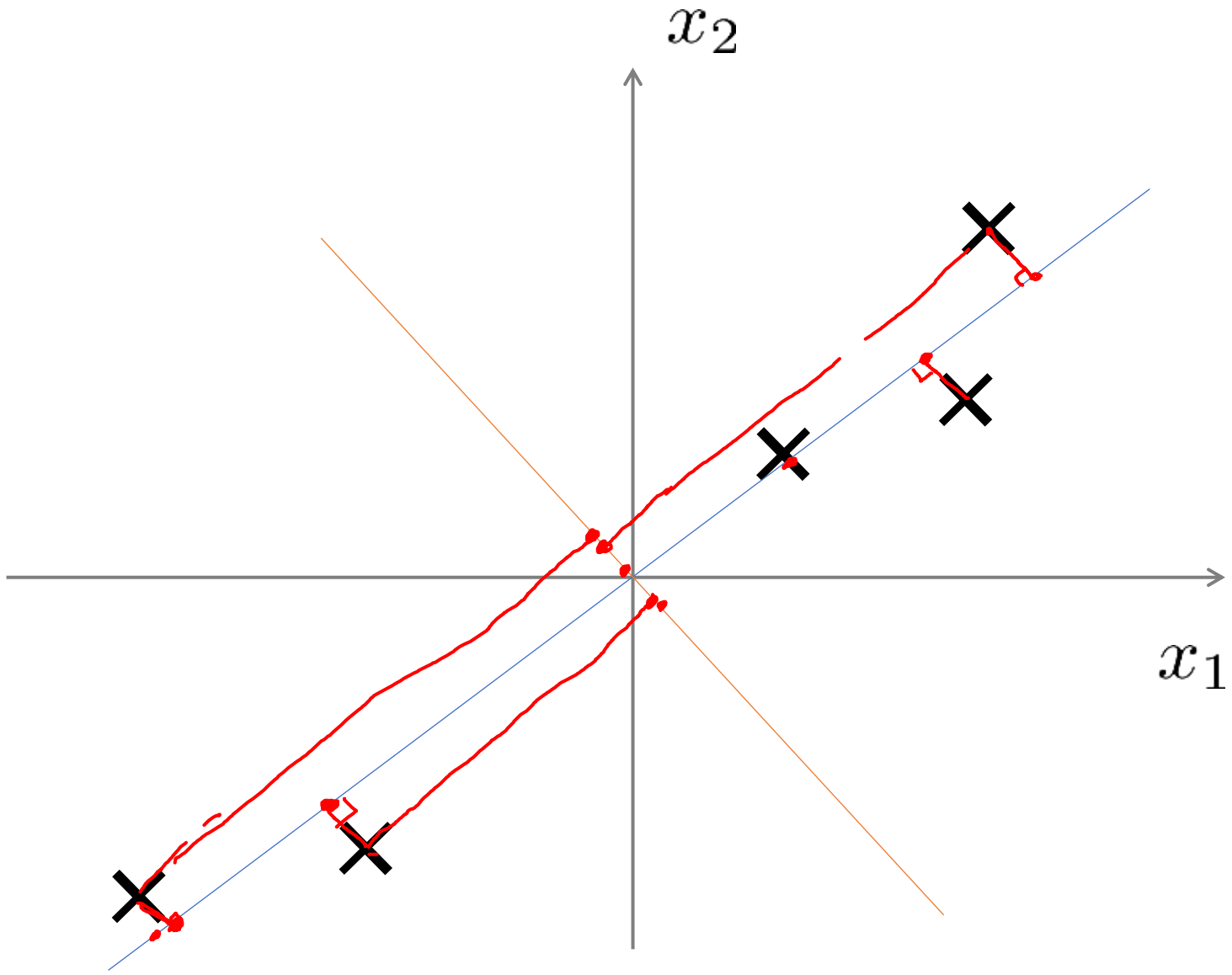
Data Visualization

Country	z_1	z_2
Canada	1.6	1.2
China	1.7	0.3
India	1.6	0.2
Russia	1.4	0.5
Singapore	0.5	1.7
USA	2	1.5
…	…	…

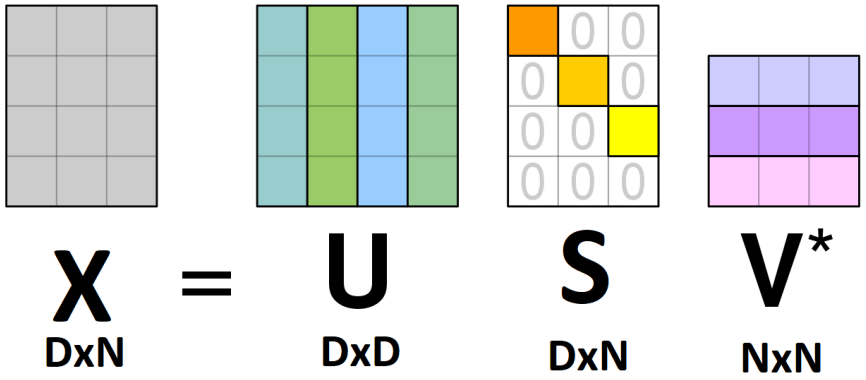
Data Visualization



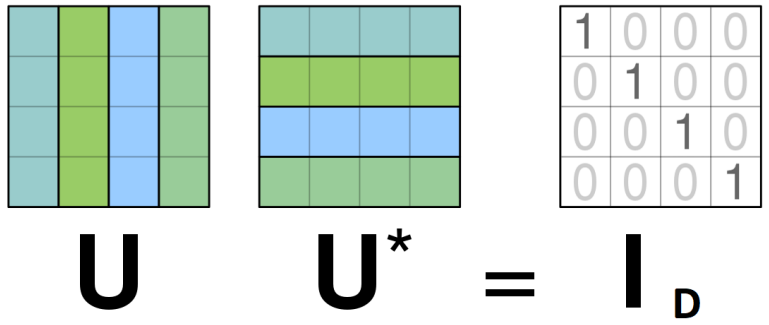
Principal Component Analysis (PCA) problem formulation



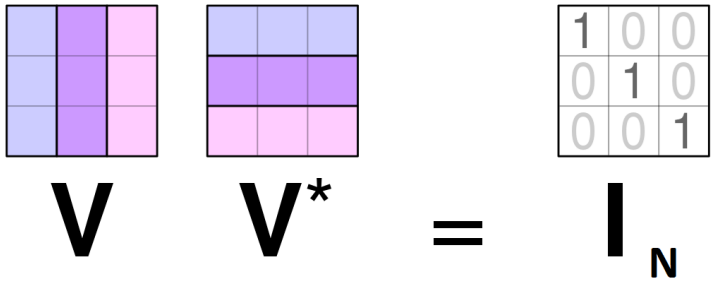
Singular Value Decomposition (SVD)



$$X = USV^T$$



$$X = \sum_{i=1}^{r} s_i u_i v_i^T$$



Unitary matrices:

$$\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}_{D\times D},$$
$$\mathbf{V}\mathbf{V}^\top = \mathbf{V}^\top\mathbf{V} = \mathbf{I}_{N\times N}.$$

$$s_1 \geq s_2 \geq s_3 \ldots \geq s_D \geq 0$$

The matrix $\mathbf{S}$ is a diagonal matrix of size $D \times N$ with non-negative entries along the diagonal. These diagonal entries are called the singular values. The columns of $\mathbf{U}$ and $\mathbf{V}$ are called the left and right singular vectors, respectively.

Proposition Let A be a $K \times L$ matrix and B an $L \times M$ matrix. Then,

$$\operatorname{rank}(AB) \leq \min(\operatorname{rank}(A), \operatorname{rank}(B))$$

Proposition Let A be a $K \times L$ matrix and B a square $L \times L$ matrix. If B is full-rank, then

$$\operatorname{rank}(AB) = \operatorname{rank}(A)$$

Proposition Let A be a $K \times L$ matrix and B a square $K \times K$ matrix. If B is full-rank, then

$$\operatorname{rank}(BA) = \operatorname{rank}(A)$$

SVD and Dimensionality Reduction

We want to "compress" the data matrix $\mathbf{X}$ from dimension D to lets say dimension K, $1 \leq K \leq D$. More precisely, we are looking for a linear transform given by the $K \times D$ matrix $\mathbf{C}$ (the compression) and a second linear transform given by the $D \times K$ matrix $\mathbf{R}$ (the reconstruction) so that

$$\|\mathbf{X} - \mathbf{RCX}\|_F^2$$

is minimized over all choices of $\mathbf{C}$ and $\mathbf{R}$.
In words, we want to compress the $D \times N$ data matrix $\mathbf{X}$ into the $K \times N$ matrix $\mathbf{CX}$ in such a way that the data is represented "as faithful as possible".

Lemma. *For any $D \times N$ matrix $\mathbf{X}$ and any $D \times N$ rank-K matrix $\hat{\mathbf{X}}$*

$$\|\mathbf{X} - \hat{\mathbf{X}}\|_F^2 \geq \|\mathbf{X} - \mathbf{U}_K \mathbf{U}_K^\top \mathbf{X}\|_F^2 = \sum_{i \geq K+1} s_i^2,$$

where $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$ is the SVD of $\mathbf{X}$, the s_i are the singular values of $\mathbf{X}$, and $\mathbf{U}_K$ is the $D \times K$ matrix consisting of the first K columns of $\mathbf{U}$.

- r=rank(X)
- If $k \leq r$: $U_k U_k^T X$ is rank k, the best rank-k approximation in Frobenius norm.
- If $k > r$: $U_k U_k^T X = X$,rank r.

More precisely, the most important information about the data is contained in the projection onto the first left singular vector, the second most important information is contained in the projection onto the second left singular vector etc. So the components are ordered in terms of importance, with the most important one being the first. In other words, our analysis/processing of the data uses the principal/most important components. This is why the above scheme is called the principal component analysis (PCA).

The expression $\mathbf{U}_K\mathbf{U}_K^\top\mathbf{X}$ has a very simple interpretation. Let $\mathbf{S}^{(K)}$ be the $D \times N$ diagonal matrix that is equal to $\mathbf{S}$ for the first K diagonal entries but is 0 thereafter.
We claim that

$$\mathbf{U}_K\mathbf{U}_K^\top\mathbf{X} = \mathbf{U}_K\mathbf{U}_K^\top\mathbf{U}\mathbf{S}\mathbf{V}^\top = \mathbf{U}\mathbf{S}^{(K)}\mathbf{V}^\top. \qquad (2)$$

With this interpretation, the lemma states that the best rank-K approximation to a matrix $\mathbf{X}$ is obtained by computing the SVD and by setting all the singular values s_j, $j \geq K+1$ to zero.

$X \epsilon \mathcal{R}^{D \times N}$

$U_k \epsilon \mathcal{R}^{D \times K}$

$\mathbf{X_p} = \mathbf{U}_K^\top \mathbf{X}$ $\quad X_P \epsilon \mathcal{R}^{K \times N}$

$\tilde{\mathbf{X}} = \mathbf{U}_K \mathbf{X_p}$ $\quad \tilde{X} \epsilon \mathcal{R}^{D \times N}$

Figure 2: *Compression via PCA. The original image is* 50×50. *The large image on the right is reconstructed from the top* $K = 10$ *principal components.*

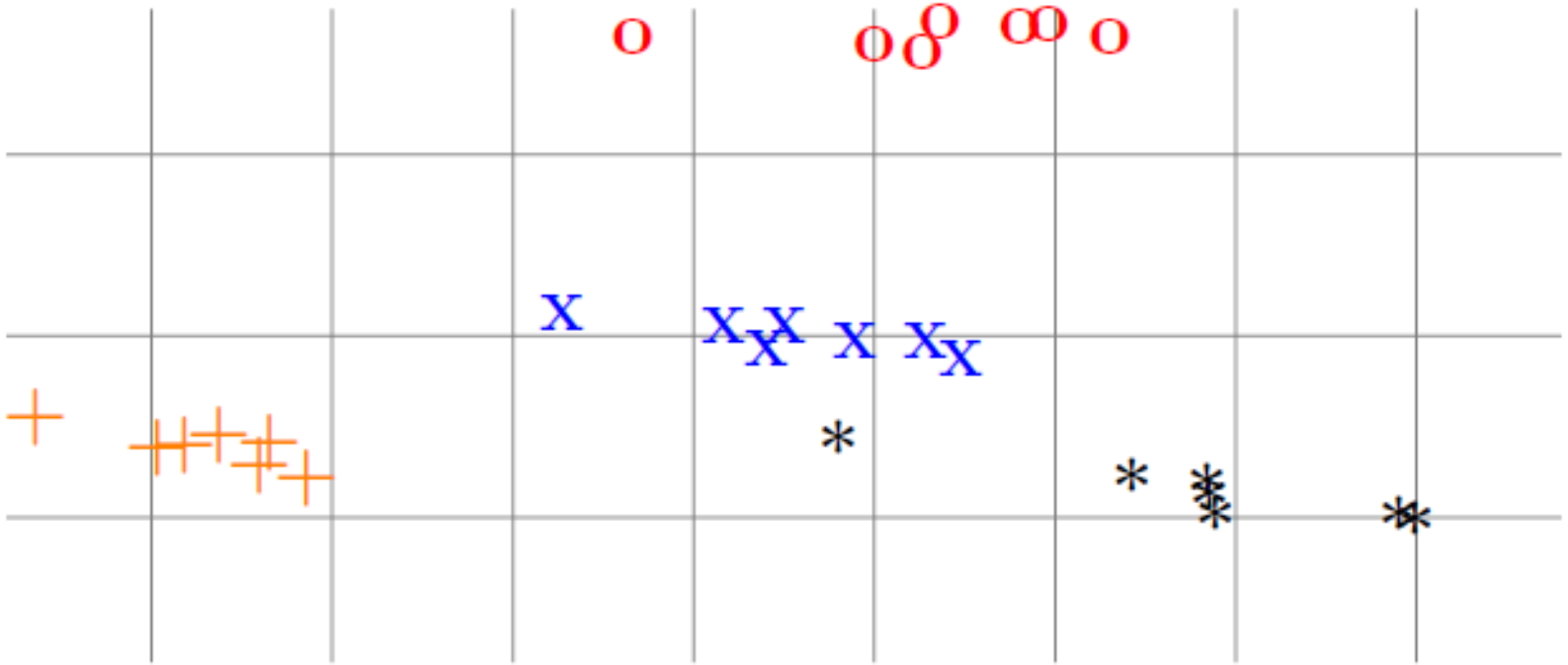


Figure 3: *Compression via PCA. The images after dimensionality reduction to* $\mathbb{R}^2$ *(*$K = 2$*). The different marks indicate different individuals.*

PCA and Decorrelation

$$\bar{\mathbf{x}} := \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n \quad , \quad \mathbf{K} := \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^\top$$

Assume that we have pre-processed the data matrix $\mathbf{X}$ by subtracting the mean from each row. Using the SVD, the empirical covariance matrix can be written as

$$\begin{aligned} N\mathbf{K} &= \sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^\top \\ &= \mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{S}\mathbf{V}^\top\mathbf{V}\mathbf{S}^\top\mathbf{U}^\top = \mathbf{U}\mathbf{S}\mathbf{S}^\top\mathbf{U}^\top = \mathbf{U}\mathbf{S}_D^2\mathbf{U}^\top, \end{aligned}$$

where $\mathbf{S}_D$ is the $D \times D$ diagonal matrix consisting of the D first columns of $\mathbf{S}$.

Now consider instead the transformed data $\tilde{\mathbf{X}} = \mathbf{U}^\top \mathbf{X}$. It has a sample co-variance matrix of

$$N\tilde{\mathbf{K}} = \tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top = \mathbf{U}^\top \mathbf{X}\mathbf{X}^\top \mathbf{U} = \mathbf{U}^\top \mathbf{U}\mathbf{S}_D^2 \mathbf{U}^\top \mathbf{U} = \mathbf{S}_D^2.$$

This means, we have linearly transformed the data in such a way that the empirical co-variance matrix is diagonal, i.e., the various components are uncorrelated. This gives us some intuition why it is perhaps useful to first linearly transform the data via the "rotation" $\mathbf{U}^\top \mathbf{X}$.

More is true. Note that by definition of the SVD, the first singular value, s_1, is the largest of all singular values. And the empirical variance of the first feature component is equal to s_1^2 according to our calculation. This means that of all the components in our feature vector $\tilde{\mathbf{X}}$, the first component has the largest variance.

Assume that we are doing classification. It is then intuitive that it is easier to classify features that have a large variance than those that have a small variance. To see this, consider the extreme case where the variance is 0 in a particular component, i.e., the data is constant in this component. This component is then not useful for classification.
From this point of view, it is then intuitive why it is good to keep the first K rows of $\tilde{\mathbf{X}}$ when we perform a dimensionality reduction. These are the components that have the highest variance and they are uncorrelated.

How to Compute U and S Efficiently

We start again with the SVD

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^\top.$$

Consider the $D \times D$ matrix $\mathbf{X}\mathbf{X}^\top$. We have

$$\mathbf{X}\mathbf{X}^\top = \mathbf{U}\mathbf{S}\mathbf{S}^\top\mathbf{U}^\top = \mathbf{U}\mathbf{S}_D^2\mathbf{U}^\top.$$

Let $\mathbf{u}_j$, $j = 1, \ldots, D$, denote the columns of $\mathbf{U}$. Then

$$\mathbf{X}\mathbf{X}^\top \mathbf{u}_j = \mathbf{U}\mathbf{S}_D^2 \mathbf{U}^\top \mathbf{u}_j = s_j^2 \mathbf{u}_j.$$

So we see that the j-th column of $\mathbf{U}$ is an eigenvector of $\mathbf{X}\mathbf{X}^\top$ with eigenvalue s_j^2. Therefore, solving the eigenvector/value problem for the matrix $\mathbf{X}\mathbf{X}^\top$ gives us a way to compute $\mathbf{U}$ and $\mathbf{S}$.

The columns of **U** (left-singular vectors) are eigenvectors of XX^T.

The columns of **V** (right-singular vectors) are eigenvectors of X^TX

The non-zero elements of S are the square roots of the non-zero eigenvalues of X^TX or XX^T

Example:

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

$$AA^T = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}, \qquad A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix}$$

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9$

eigenvectors

eigenvalues: $\lambda_1 = 25, \lambda_2 = 9, \lambda_3 = 0$

eigenvectors

$$u_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \quad u_2 = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

$$v_1 = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix} \quad v_2 = \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix} \quad v_3 = \begin{pmatrix} 2/3 \\ -2/3 \\ -1/3 \end{pmatrix}$$

$$A = USV^T = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

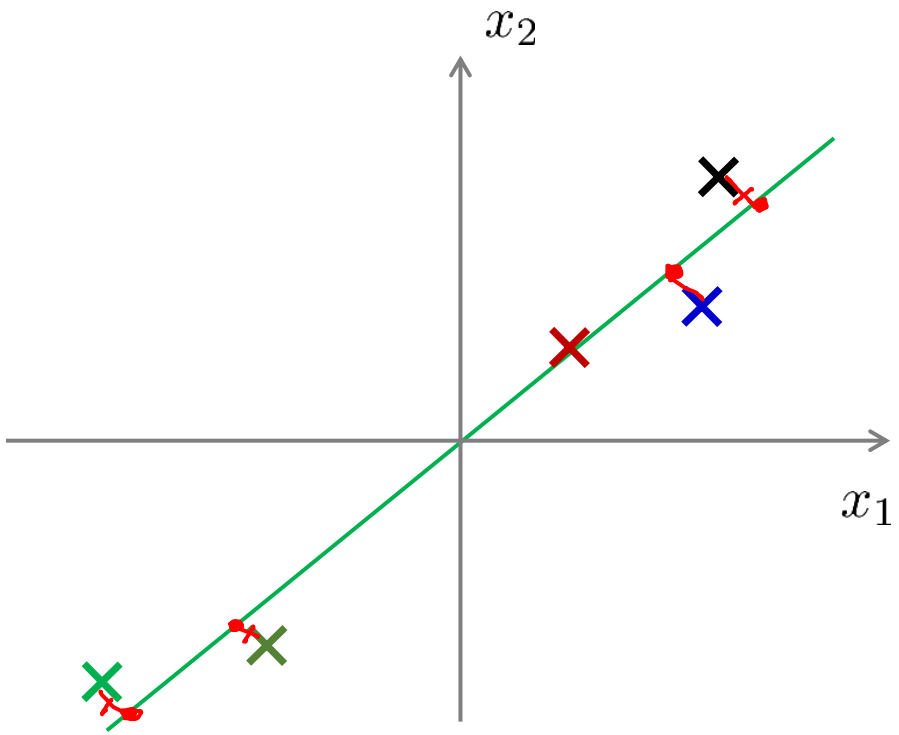
$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

The matrix A above can be decomposed as

$$= 5 \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \end{pmatrix} + 3 \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \end{pmatrix}$$

$$= \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

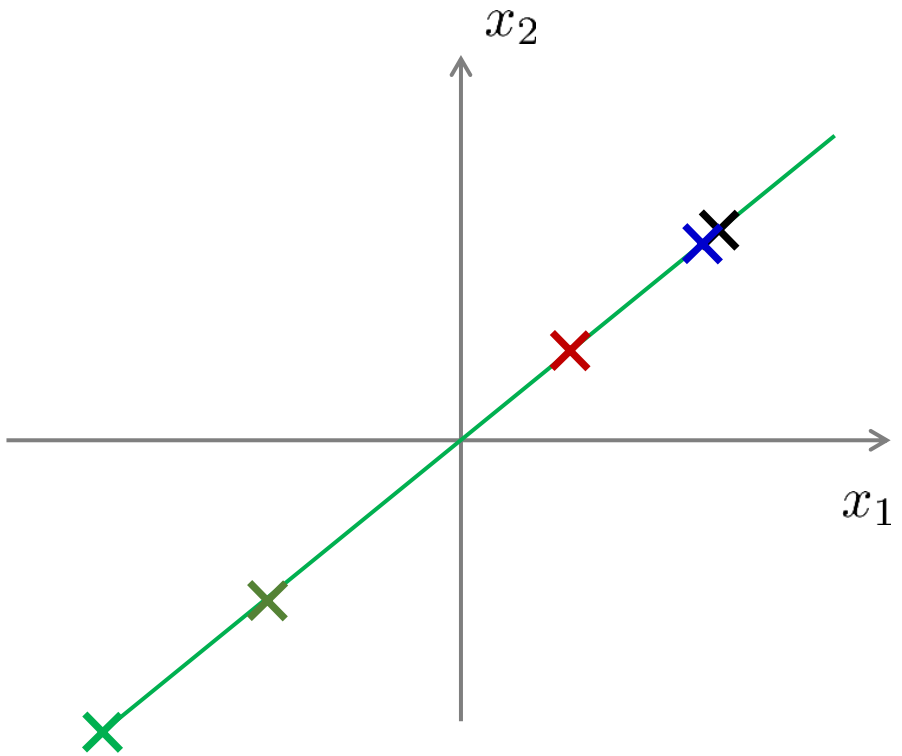
Reconstruction from compressed representation



$$z = U_{reduce}^T x$$

$k \times 1$ $\quad$ $k \times n$ $\quad$ $n \times 1$

$k = 1$ $\quad$ $n = 2$



$\hat{x} = U z$

$n \times 1$ $\quad$ $n \times k$ $\quad$ $k \times 1$

Choosing k (number of principal components)

Average squared projection error:

Total variation in the data:

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01 \qquad (1\%)$$

"99% of variance is retained"

Choosing k (number of principal components)

Algorithm:

Try PCA with $k = 1$

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Check if

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01?$$

```
[U,S,V] = svd(Sigma)
```

Choosing k (number of principal components)

```
[U,S,V] = svd(Sigma)
```

Pick smallest value of k for which

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$$

(99% of variance retained)

Supervised learning speedup

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

Extract inputs:

Unlabeled dataset: $x^{(1)}, x^{(2)}, \ldots, x^{(m)} \in \mathbb{R}^{10000}$

$\downarrow PCA$

$z^{(1)}, z^{(2)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$

New training set:

$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})$

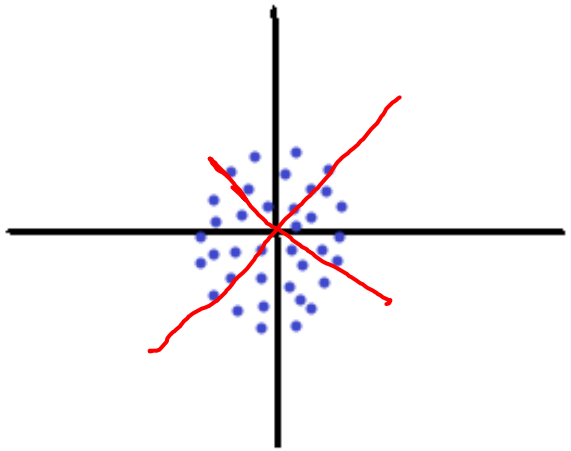
Note: Mapping $x^{(i)} \rightarrow z^{(i)}$ should be defined by running PCA only on the training set. This mapping can be applied as well to the examples $x_{cv}^{(i)}$ and $x_{test}^{(i)}$ in the cross validation and test sets.

Application of PCA

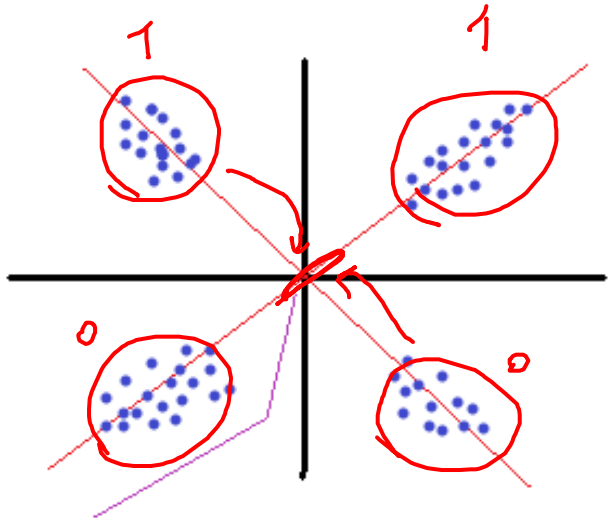
- Compression
  - Reduce memory/disk needed to store data
  - Speed up learning algorithm

- Visualization

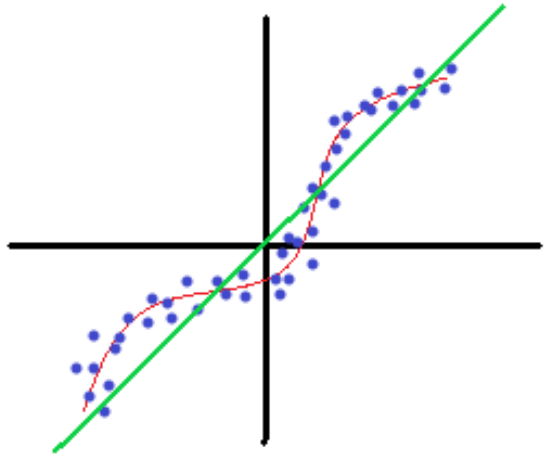
PCA does not work well



$\lambda_1 \sim \lambda_2$
Information lost is vey high



Points projected here from two clusters will not be distinguishable



The plot will loose the information it depicts through its wave when points will get projected

Bad use of PCA: To prevent overfitting

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features to $k < n$.

Thus, fewer features, less likely to overfit.

This might work OK, but isn't a good way to address overfitting. Use regularization instead.

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$
- Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$
- Train logistic regression on $\{(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x_{test}^{(i)}$ to $z_{test}^{(i)}$. Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots, (z_{test}^{(m)}, y_{test}^{(m)})\}$

How about doing the whole thing without using PCA?

Before implementing PCA, first try running whatever you want to do with the original/raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.