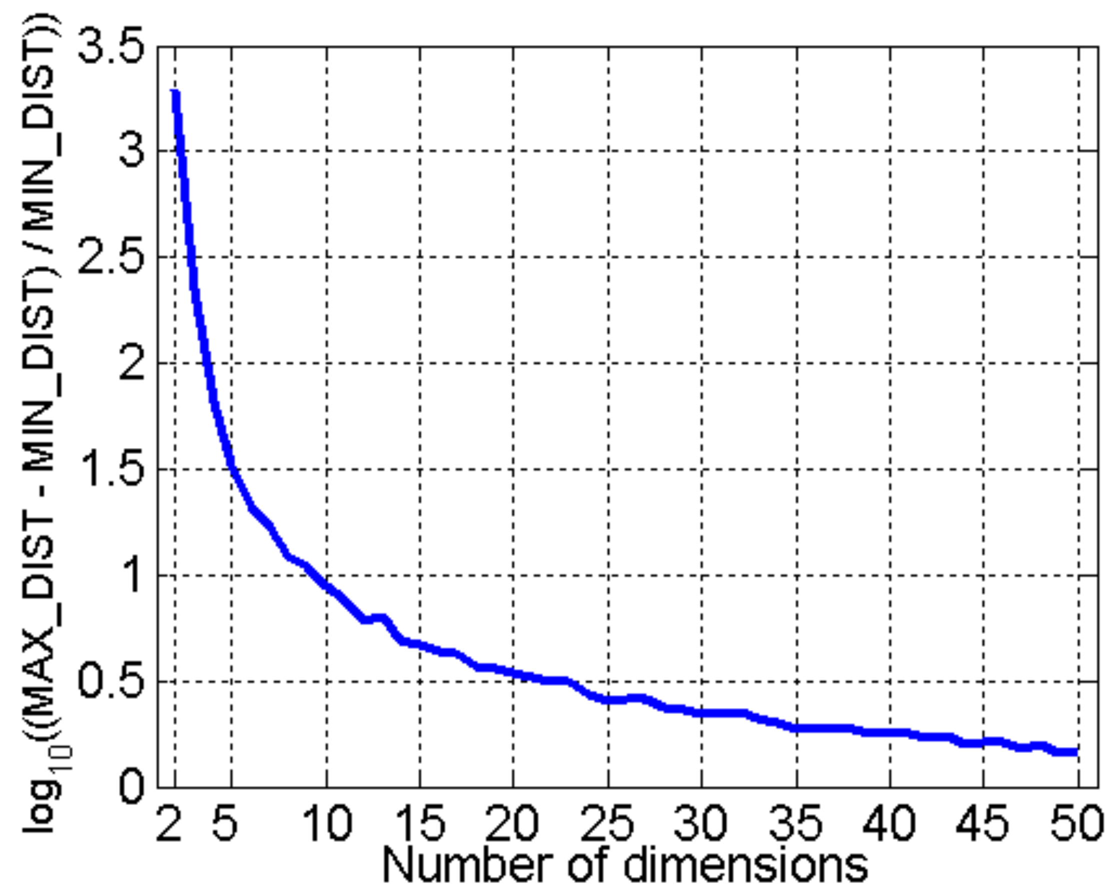
Dimensionality Reduction

Vinay Setty

Curse of Dimensionality

- When dimensionality increases, data becomes increasingly sparse in the space that it occupies

- Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful



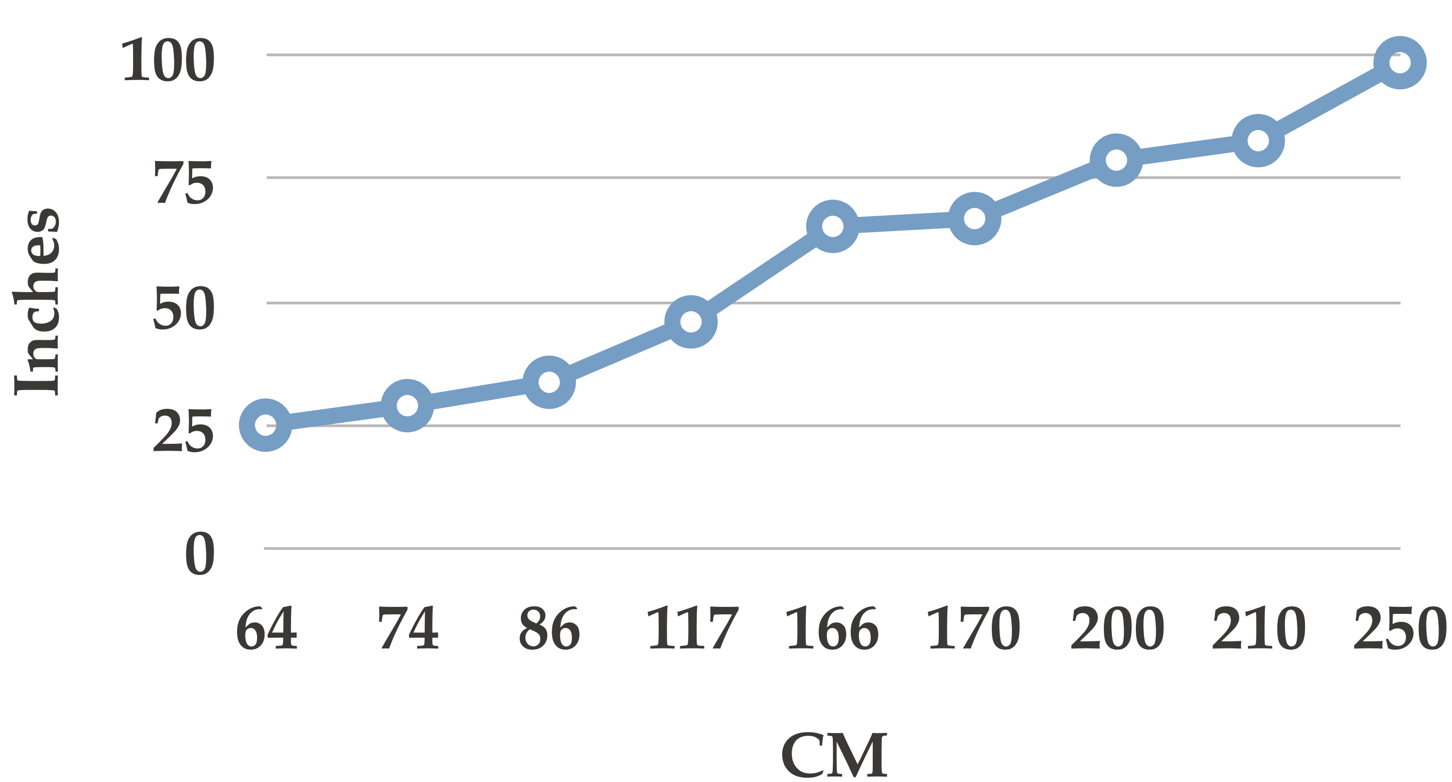
- **Randomly generate 500 points**
- **Compute difference between max and min distance between any pair of points**

Dimensionality Reduction

- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise
- Techniques
  - Principle Component Analysis
  - Singular Value Decomposition

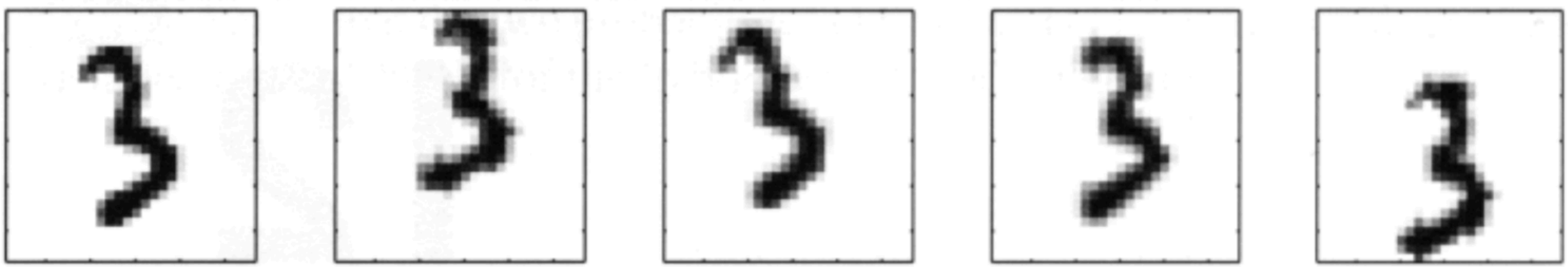
Motivation 1: Data compression

- Data Mining/Machine Learning algorithms can be trained faster
- Less storage/memory used
- What is dimensionality reduction?

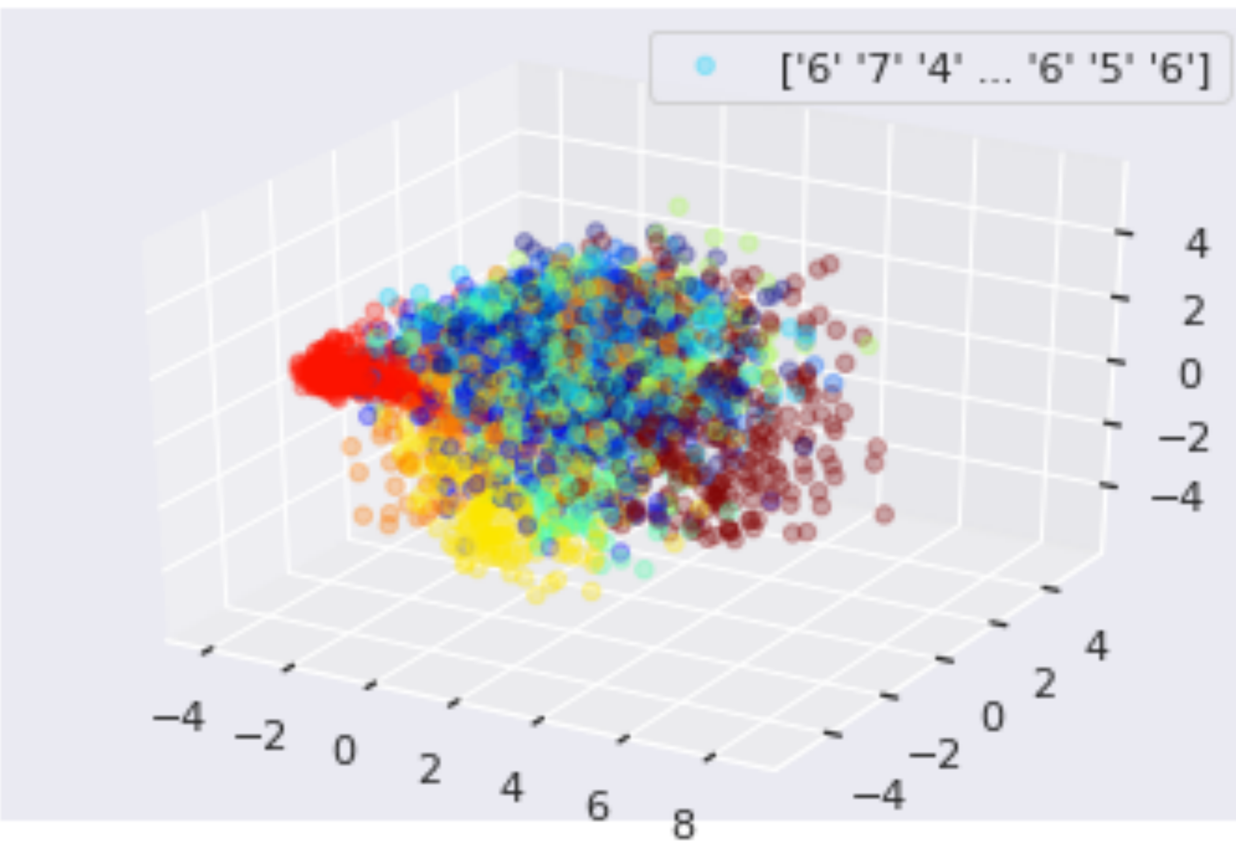


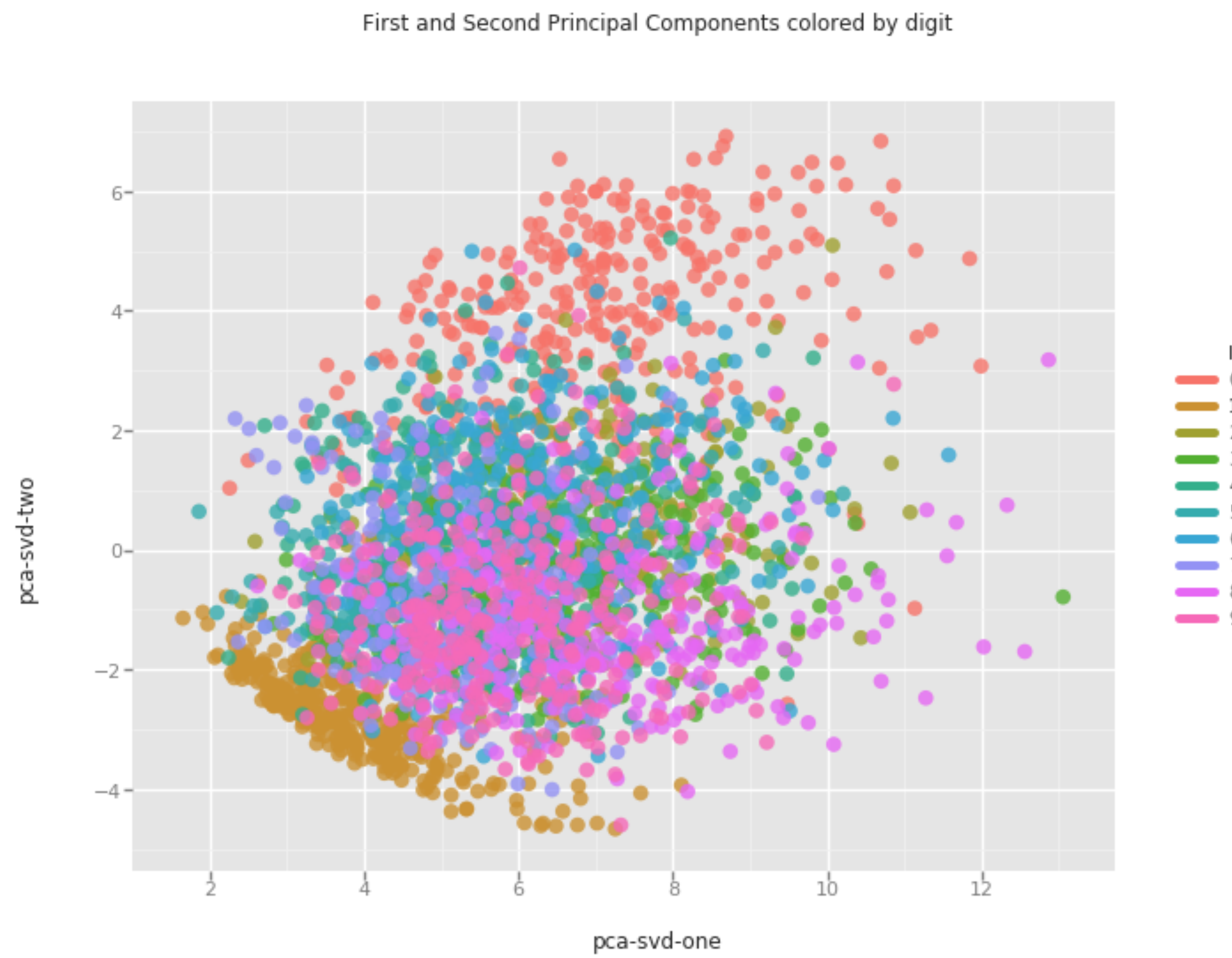
Data redundancy

Digit Example

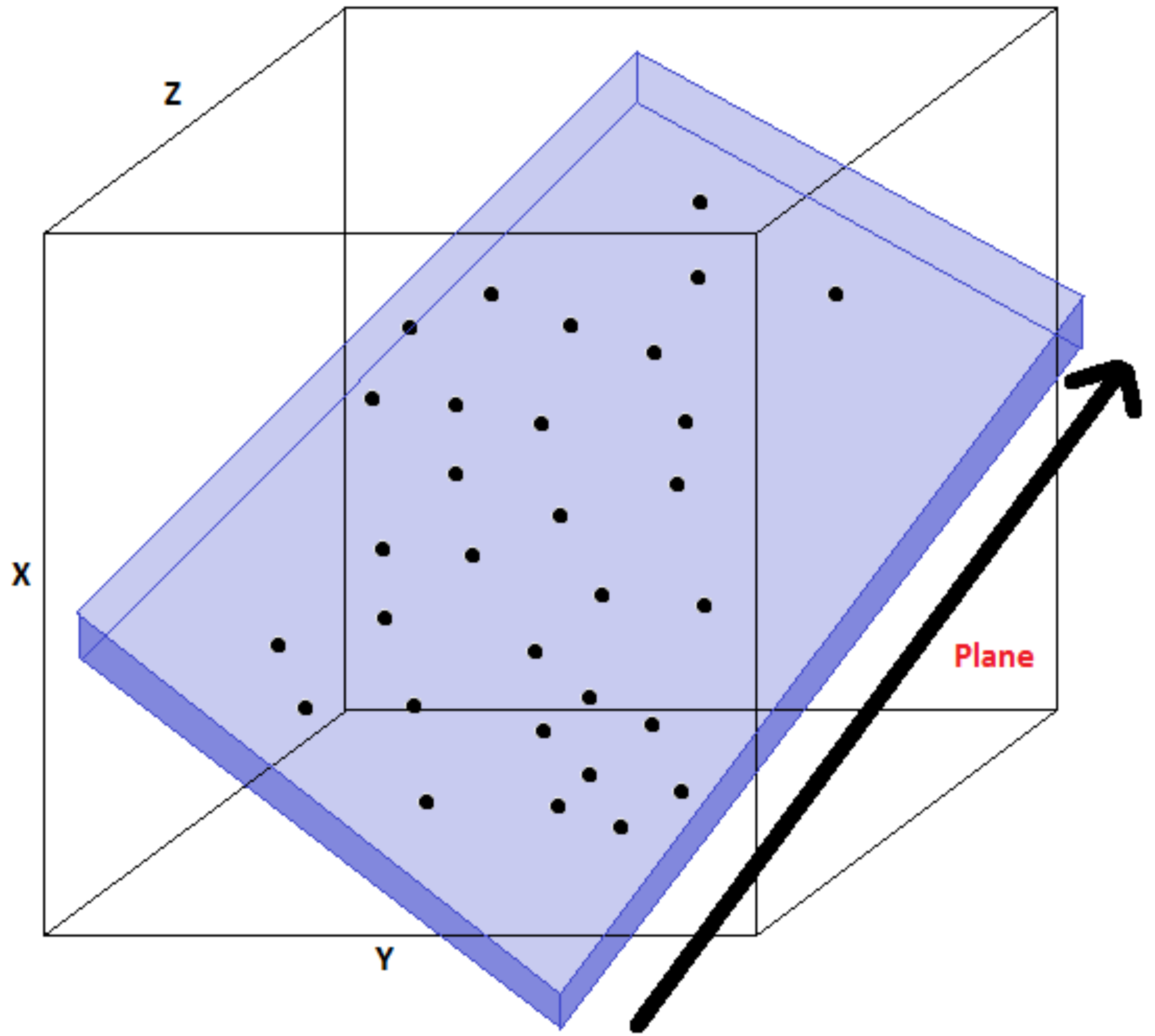


Another Example





Lower dimensional plane



Motivation 2: Visualization

- It's hard to visualize high dimensional data
  - Dimensionality reduction can improve how we display information in a tractable manner for human consumption
  - Why do we care?
    - Often helps to develop algorithms if we can understand our data better
    - Dimensionality reduction helps us do this, see data in a helpful
    - Good for explaining something to someone if you can "show" it in the data

Visualization Example

Collect a large data set about many facts of a country around the world

Country	GDP (trillions of US$)	Per capita GDP (thousands of intl. $)	Human Develop-ment Index	Life expectancy	Poverty Index (Gini as percentage)	Mean household income (thousands of US$)	...
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	...
...	...	...	...	...	...	...	

Visualization Example

- This gives us a 2-dimensional vector
- Reduce 50D to 2D and Plot as a 2D plot
- Typically you don't generally ascribe meaning to the new features
  - e.g. may find horizontal axis corresponds to overall country size/economic activity
- and y axis may be the per-person well being/economic activity
- So despite having 50 features, there may be two "dimensions" of information, with features associated with each of those dimensions

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
...	...	...

SVD (Singular Value Decomposition)

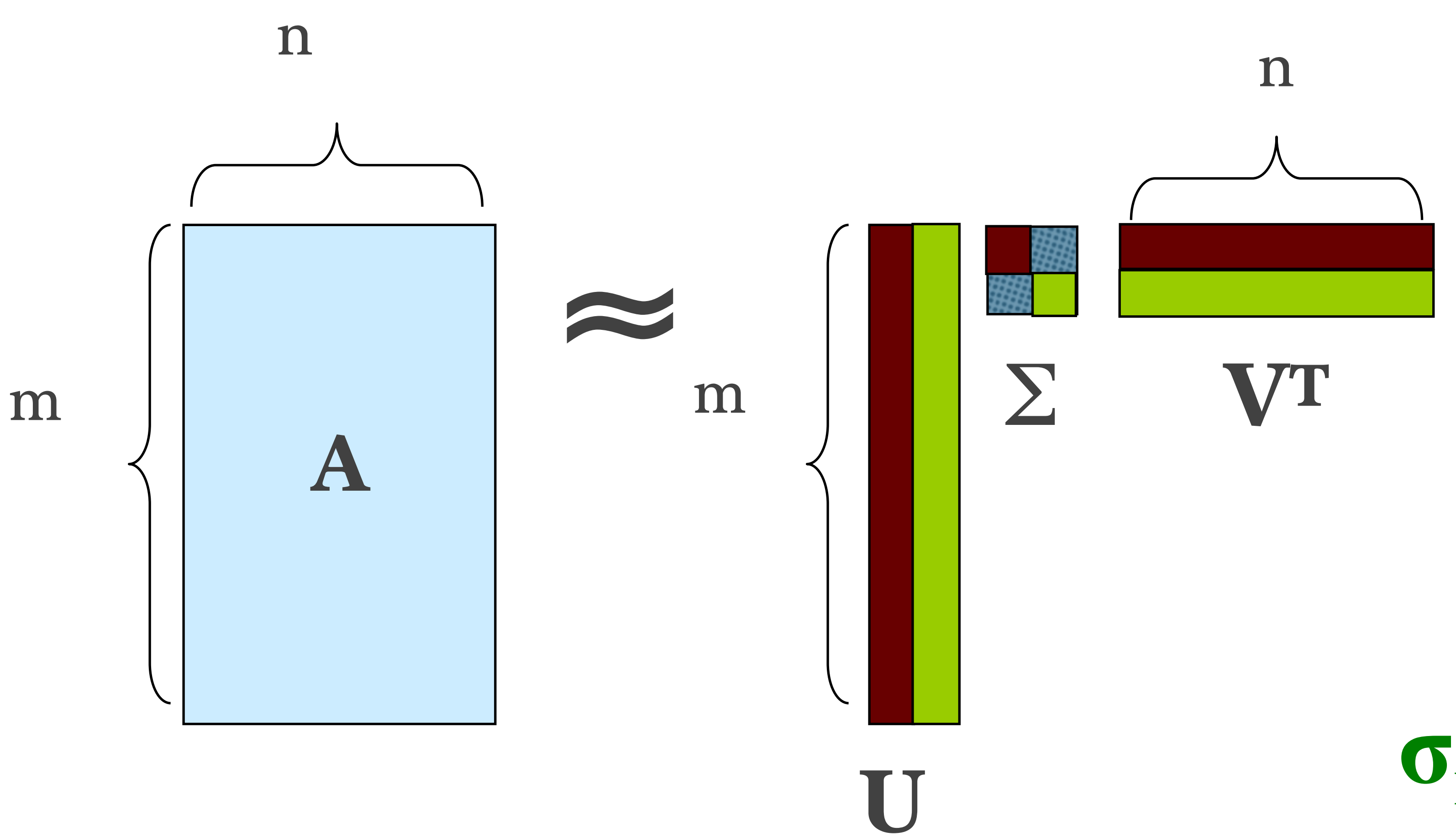
SVD - Definition

$$\boxed{A_{[m \times n]} = U_{[m \times r]} \Sigma_{[r \times r]} (V_{[n \times r]})^T}$$

- A: **Input data matrix**
  - m x n matrix (e.g., m documents, n terms)
- U: **Left singular vectors**
  - m x r matrix (m documents, r concepts)
- Σ: **Singular values**
  - r x r diagonal matrix (strength of each 'concept') (r : rank of the matrix **A**)
- V: **Right singular vectors**
  - n x r matrix (n terms, r concepts)

SVD

$$\mathbf{A} \approx \mathbf{U\Sigma V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^{\mathsf{T}}$$



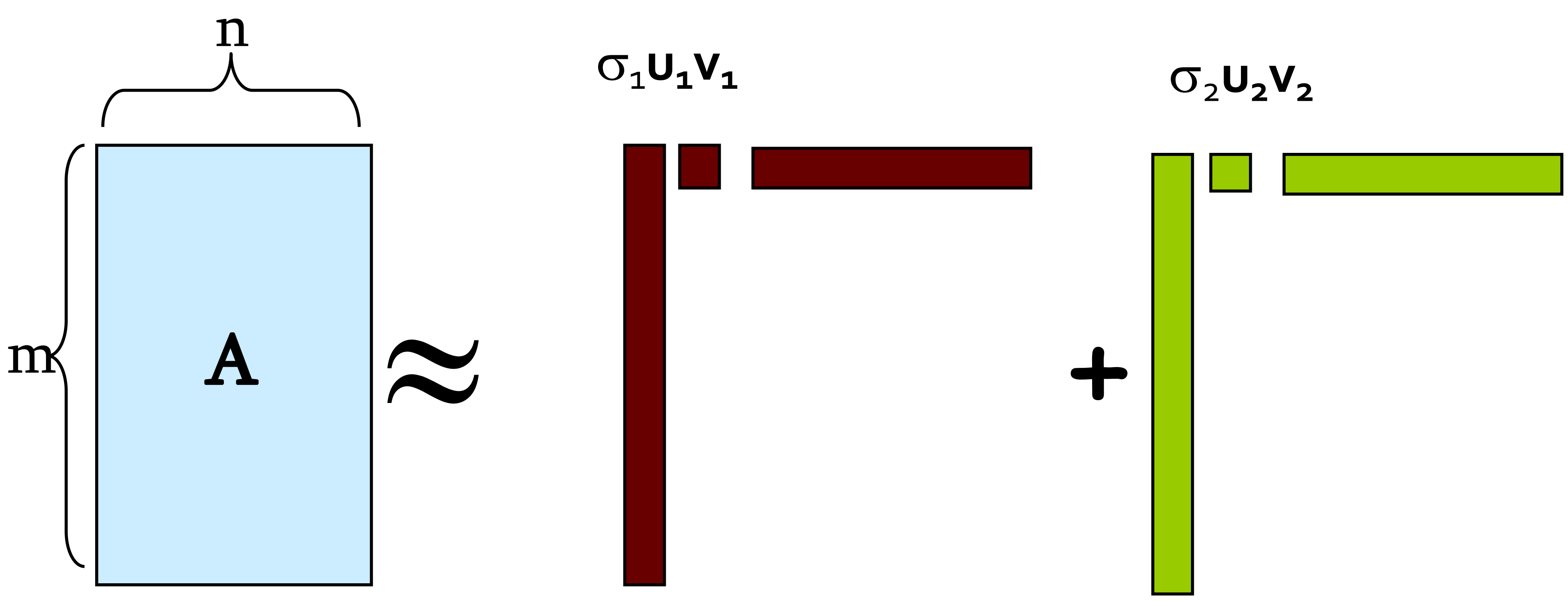
σ_i … scalar

u_i … vector

v_i … vector

SVD Intuition

$$\mathbf{A} \approx \mathbf{U\Sigma V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^{\mathsf{T}}$$



σ_i … scalar
u_i … vector
v_i … vector

SVD - Properties

It is **always** possible to decompose a real matrix $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{U}\,\Sigma\,\boldsymbol{V}^{\mathrm{T}}$, where

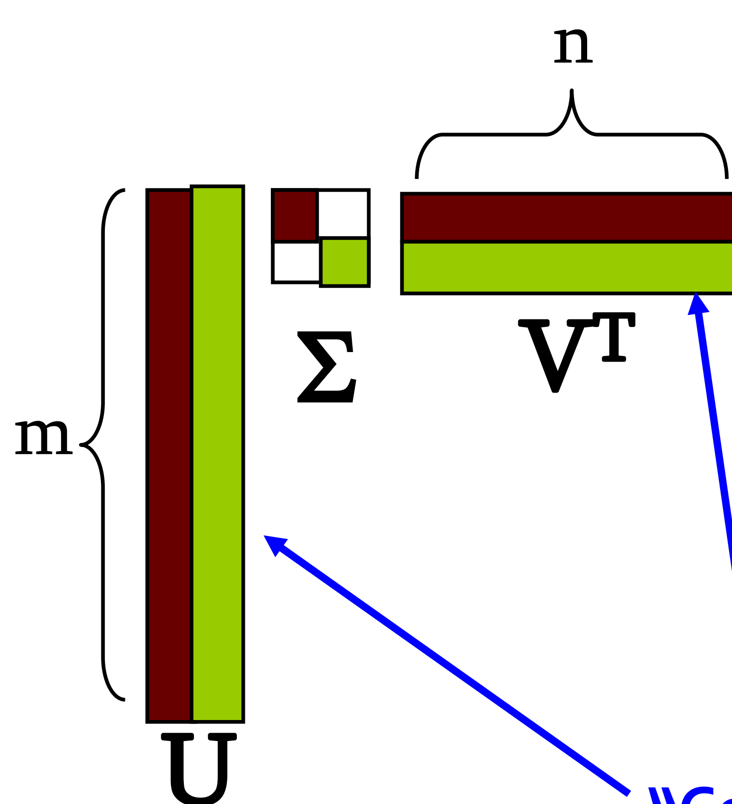
- $\boldsymbol{U}, \Sigma, \boldsymbol{V}$: unique
- $\boldsymbol{U}, \boldsymbol{V}$: column orthonormal
  - $\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}$; $\boldsymbol{V}^T \boldsymbol{V} = \boldsymbol{I}$ ($\boldsymbol{I}$: identity matrix)
  - (Columns are orthogonal unit vectors)
- Σ: diagonal
  - Entries (**singular values**) are positive, and sorted in decreasing order ($\boldsymbol{\sigma_1} \geq \boldsymbol{\sigma_2} \geq \ldots \geq 0$)

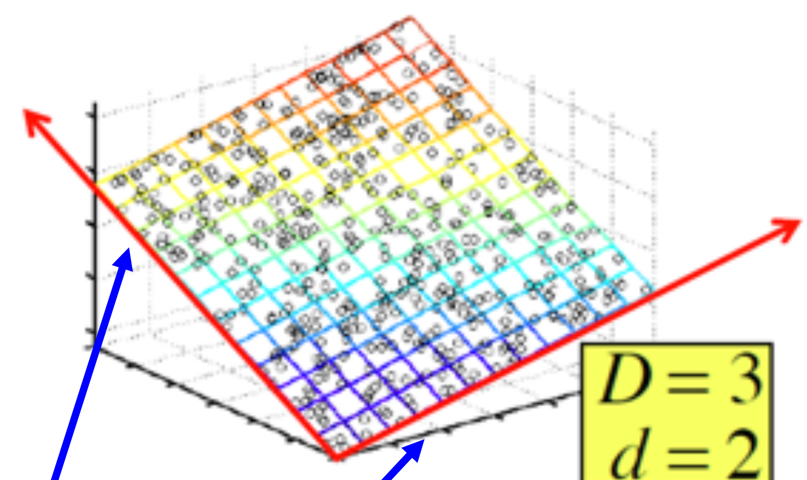
Nice proof of uniqueness: http://www.mpi-inf.mpg.de/~bast/ir-seminar-ws04/lecture2.pdf

SVD – Example: Users-to-Movies

- **A = U Σ V^T - example: Users to Movies**

$$
\begin{array}{cc}
 & \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{array}{c} \\ \text{SciFi} \\ \\ \\ \\ \text{Romnce} \\ \\ \end{array} & \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}
\end{array}
=
$$





"Concepts"
AKA Latent dimensions
AKA Latent factors

SVD – Example: Users-to-Movies

- **$A = U \Sigma V^T$ - example: Users to Movies**

$$
\begin{array}{c|c}
 & \begin{array}{ccccc} \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \end{array} \\
\begin{array}{c} \text{SciFi} \\ \\ \\ \\ \text{Romnce} \\ \\ \\ \end{array} &
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
\end{array}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \mathbf{1.3}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

Dimensionality Reduction Using SVD

More details

- **Q:** How exactly is dim. reduction done?

$$\begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\ \mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\ \mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\ 0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\ 0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\ 0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2} \end{bmatrix} = \begin{bmatrix} \mathbf{0.13} & 0.02 & -0.01 \\ \mathbf{0.41} & 0.07 & -0.03 \\ \mathbf{0.55} & 0.09 & -0.04 \\ \mathbf{0.68} & 0.11 & -0.05 \\ 0.15 & \mathbf{-0.59} & \mathbf{0.65} \\ 0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\ 0.07 & \mathbf{-0.29} & \mathbf{0.32} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \mathbf{1.3} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ 0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Dimensionality Reduction Using SVD

More details

- **Q: How exactly is dim. reduction done?**
- **A: Set smallest singular values to zero**

$$
\begin{bmatrix}
\mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 \\
\mathbf{3} & \mathbf{3} & \mathbf{3} & 0 & 0 \\
\mathbf{4} & \mathbf{4} & \mathbf{4} & 0 & 0 \\
\mathbf{5} & \mathbf{5} & \mathbf{5} & 0 & 0 \\
0 & \mathbf{2} & 0 & \mathbf{4} & \mathbf{4} \\
0 & 0 & 0 & \mathbf{5} & \mathbf{5} \\
0 & \mathbf{1} & 0 & \mathbf{2} & \mathbf{2}
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{0.13} & 0.02 & -0.01 \\
\mathbf{0.41} & 0.07 & -0.03 \\
\mathbf{0.55} & 0.09 & -0.04 \\
\mathbf{0.68} & 0.11 & -0.05 \\
0.15 & \mathbf{-0.59} & \mathbf{0.65} \\
0.07 & \mathbf{-0.73} & \mathbf{-0.67} \\
0.07 & \mathbf{-0.29} & \mathbf{0.32}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{12.4} & 0 & 0 \\
0 & \mathbf{9.5} & 0 \\
0 & 0 & \cancel{\mathbf{1.3}}
\end{bmatrix}
\times
\begin{bmatrix}
\mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\
0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\
0.40 & \mathbf{-0.80} & 0.40 & 0.09 & 0.09
\end{bmatrix}
$$

26

Dimensionality Reduction Using SVD

More details

- **Q:** How exactly is dim. reduction done?
- **A: Set smallest singular values to zero**

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix} \approx \begin{bmatrix} \mathbf{0.13} & 0.02 & \cancel{-0.01} \\ \mathbf{0.41} & 0.07 & \cancel{-0.03} \\ \mathbf{0.55} & 0.09 & \cancel{-0.04} \\ \mathbf{0.68} & 0.11 & \cancel{-0.05} \\ 0.15 & \mathbf{-0.59} & \cancel{\mathbf{0.65}} \\ 0.07 & \mathbf{-0.73} & \cancel{\mathbf{-0.67}} \\ 0.07 & \mathbf{-0.29} & \cancel{\mathbf{0.32}} \end{bmatrix} \times \begin{bmatrix} \mathbf{12.4} & 0 & 0 \\ 0 & \mathbf{9.5} & 0 \\ 0 & 0 & \cancel{\mathbf{1.3}} \end{bmatrix} \times \begin{bmatrix} \mathbf{0.56} & \mathbf{0.59} & \mathbf{0.56} & 0.09 & 0.09 \\ 0.12 & -0.02 & 0.12 & \mathbf{-0.69} & \mathbf{-0.69} \\ \cancel{0.40} & \cancel{\mathbf{-0.80}} & \cancel{0.40} & \cancel{0.09} & \cancel{0.09} \end{bmatrix}$$

28

Relation to Eigenvectors

- **SVD gives us:**
  - $A = U\,\Sigma\,V^T$
- **Eigen-decomposition:**
  - $A = X\,\Lambda\,X^T$
    - A is symmetric
    - U, V, X are orthonormal ($\mathbf{U}^T\mathbf{U}=\mathbf{I}$),
    - $\boldsymbol{\Lambda}, \boldsymbol{\Sigma}$ are diagonal

Shows how to compute SVD using eigenvalue decomposition!

$$X\,\Lambda^2\,X^T$$

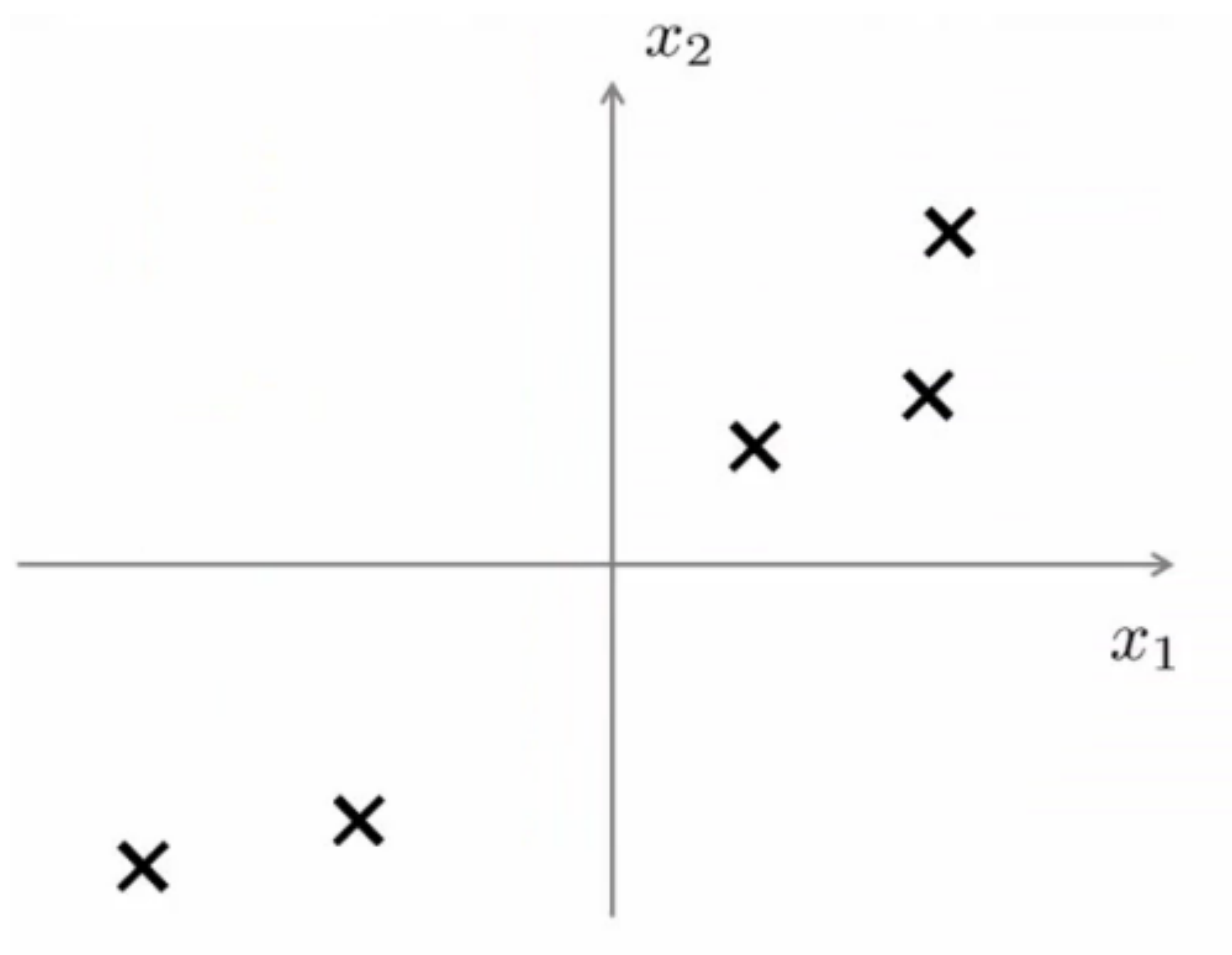
- **Now let's calculate:**
  - $\mathbf{A}\mathbf{A}^T = \mathbf{U}\Sigma\,\mathbf{V}^T(\mathbf{U}\Sigma\,\mathbf{V}^T)^T = \mathbf{U}\Sigma\,\mathbf{V}^T(\mathbf{V}\Sigma^T\mathbf{U}^T) = \mathbf{U}\Sigma\Sigma^T\,\mathbf{U}^T$
  - $\mathbf{A}^T\mathbf{A} = \mathbf{V}\,\Sigma^T\,\mathbf{U}^T\,(\mathbf{U}\Sigma\,\mathbf{V}^T) = \mathbf{V}\,\Sigma\Sigma^T\,\mathbf{V}^T$

$$X\,\Lambda^2\,X^T$$

Principal Component Analysis (PCA)

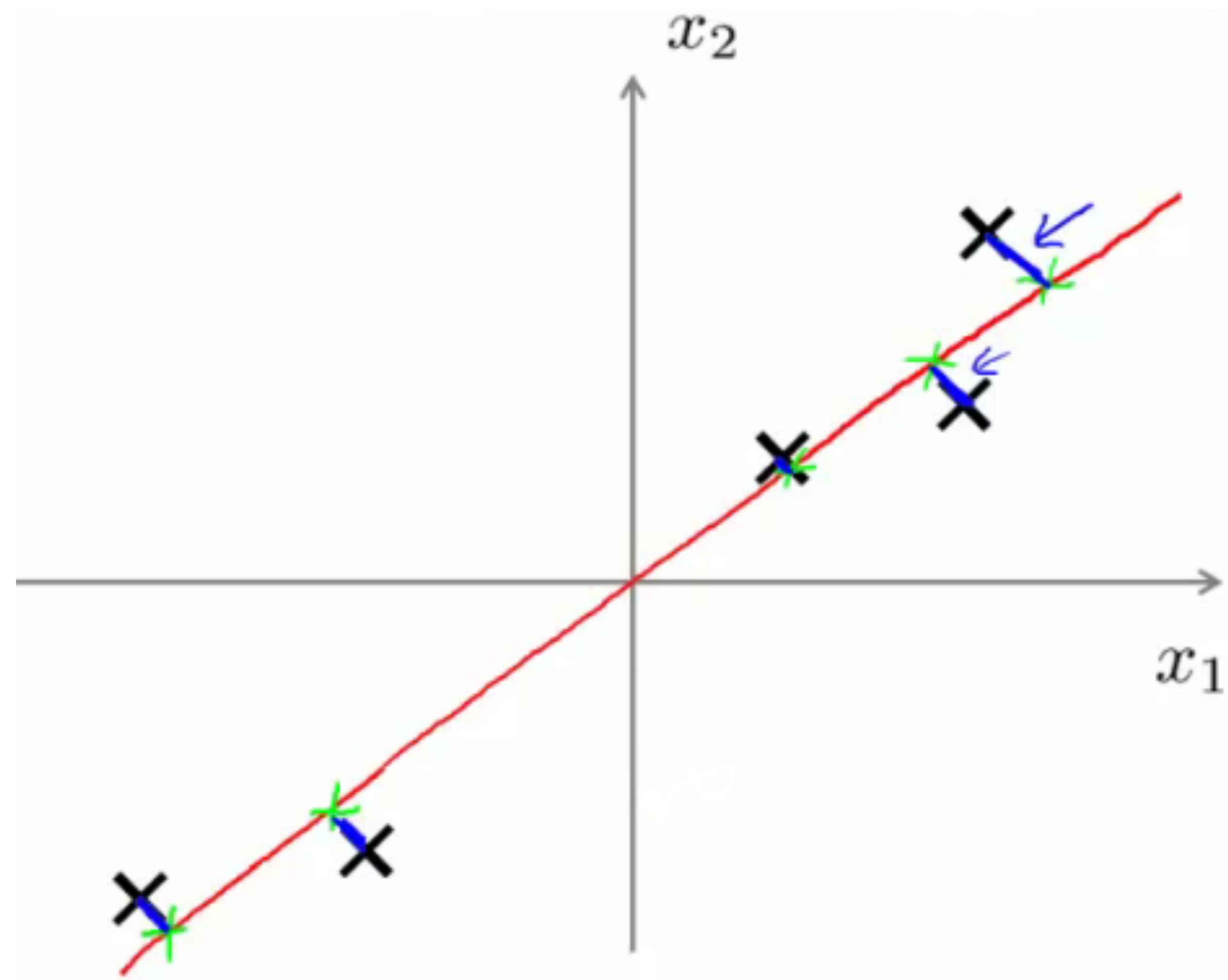
PCA - Problem Formulation

- PCA is the most commonly used dimensionality reduction
- Say we have a 2D data set which we wish to reduce to 1D



PCA Goal

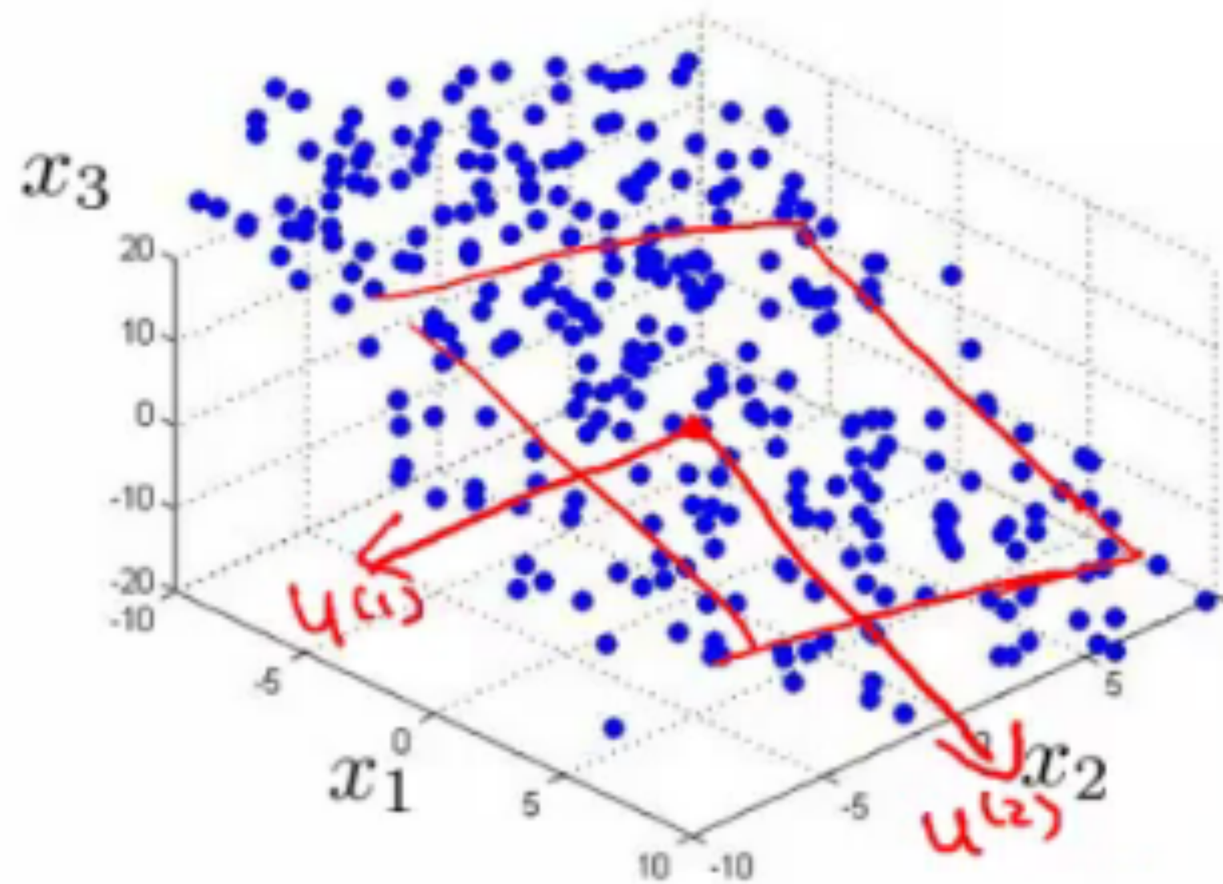
- The goal is to find a single line (or plane in higher dimensions) onto which to project the input data
- How to find this line/plane?
  - The distance between each point and the projected version should be small (blue lines below are short)
  - PCA tries to find a lower dimensional surface so the sum of squares onto that surface is minimized



Minimize projection error:
PCA tries to find the surface (a straight line in this case) which has the minimum projection error

General Case

- Given M X N Matrix (M rows with N-dimensions) the goal is to find M X K matrix which is a K-dimensional vector for each data point
- Find a set of vectors which we project the data onto the linear subspace spanned by that set of vectors
  - We can define a point in a plane with K dimensional vectors



PCA Algorithm

- Before applying PCS it is essential to do preprocessing
- Step 1: Preprocessing:
  - **Mean normalization:** Replace each x_j^i with $x_j - \mu_j$, subtract the mean from the value, so we re-scale the mean to be 0
  - **Feature scaling (depending on data):** If features have very different scales, normalize them by x_j^i is set to $(x_j - \mu_j) / s_j$ Where s_j is some measure of the range, so could be max - min or standard deviation

PCA Algorithm

- Step 2: Covariane matrix
  - Compute the covariance matrix

$$\Sigma = \frac{1}{n} \sum_{i=1}^{m} \left(x^{i}\right) \times \left(x^{i}\right)^{T}$$

  - This is an [M x M] matrix
  - Remember that x^i is a [M x 1] vector

PCA Algorithm

- Step 3:
  - Compute SVD
  - [U,S,V] = svd(sigma)
  - Could also use Eigen value decomposition
  - U matrix is also an [n x n] matrix
  - to reduce a system from n-dimensions to k-dimensions
  - Just take the first k-vectors from U (first k columns)

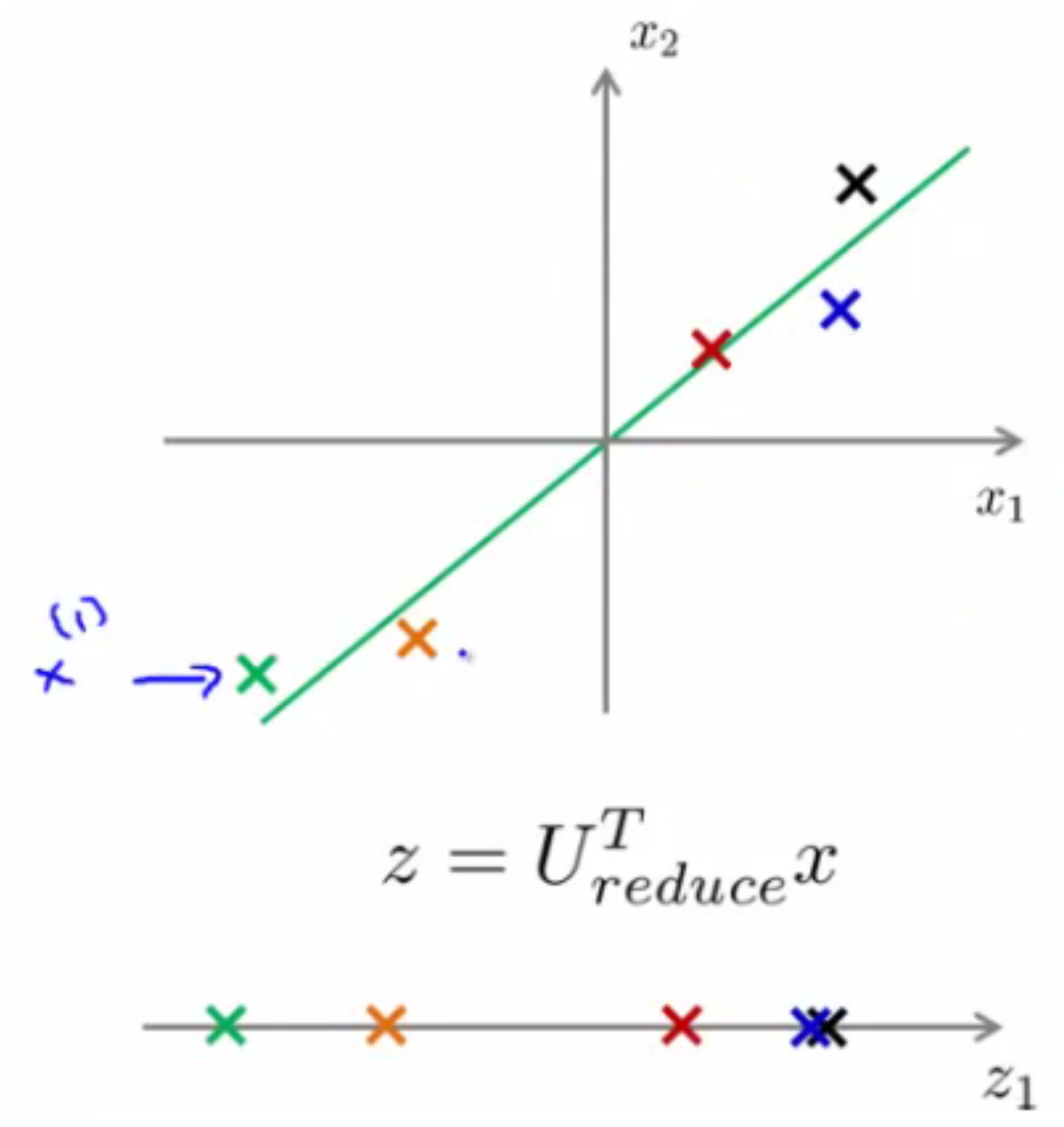
$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

PCA Transformatiom

- Next we need to find some way to change X (which is n dimensional) to z (which is k dimensional)
- (reduce the dimensionality)
- Take first k columns of the u matrix and stack in columns
- n x k matrix - call this $U_{reduced}$
- We calculate z as follows
- $z = (U_{reduced})^T * X$
- So [k x n] * [n x 1]
  - Generates a matrix which is
    - k * 1

How to measure quality of PCA?

- Reconstruction from Compressed Representation (decompress)



Reconstruction Error

- Given the reduced data in K dimensions (U_{reduce})
  - Xapprox = U_{reduce} . Z

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

How to choose K?

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

- Ratio between averaged squared projection error with total variation in data
    - Want ratio to be small - means we retain 99% of the variance
- If it's small (0) then this is because the numerator is small
    - The numerator is small when xi = x_{approx}i

Advice for applying PCA

- Given 10000 dimensional feature vector
  - E.g, 100x100 pixel images
  - This would slow down machine learning algorithms
- Reduce it to a lower dimensional vector
  - Apply PCA to x vectors
  - Take the reduced dimensionality data set and feed to a learning algorithm

Literature

- Appendix B in Tan, Steinbach and Kumar
- For additional reading about SVD Chapter 11 in Mining Massive Datasets
  - Some slides were also borrowed from mmds.org