Principal Component Analysis

UNIVERSITY *of* WASHINGTON

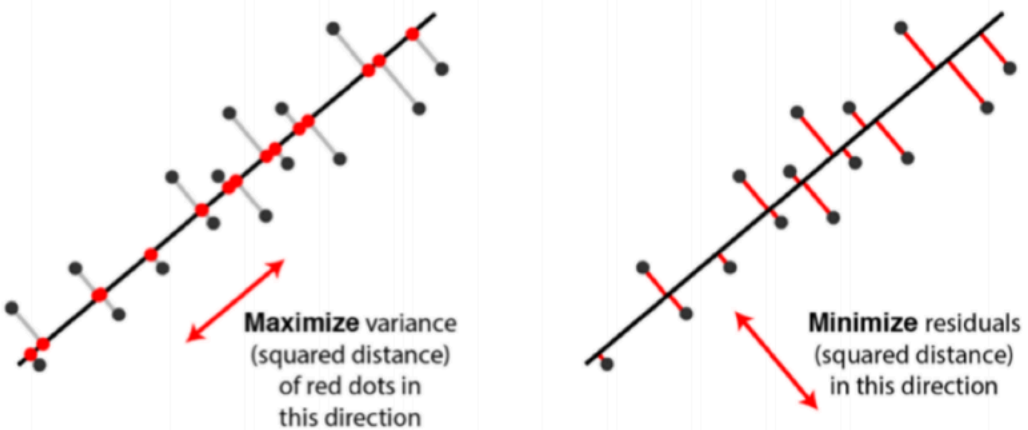
PCA in one dimension, 2 equivalent views

Goal: find a k < d-dimensional representation of X

For k = 1:

Choose $\vec{v} \in \mathbb{R}^d, ||v|| = 1$
to minimize

$$\frac{1}{n} \sum_{i=1}^{n} dist(x_i, \text{line defined by } \vec{v})$$



Two equivalent views of principal component analysis.

PCA: a high-fidelity linear projection

$$\sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||_2^2$$

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$ → orthonormal

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||_2^2 = \min_{\mathbf{V}_q} Tr(\Sigma) - Tr(\mathbf{V}_q^T \Sigma \mathbf{V}_q)$$

Eigenvalue decomposition

$\mathbf{V}_q$ are the first q eigenvectors of Σ

Minimize reconstruction error and capture the most variance in your data.

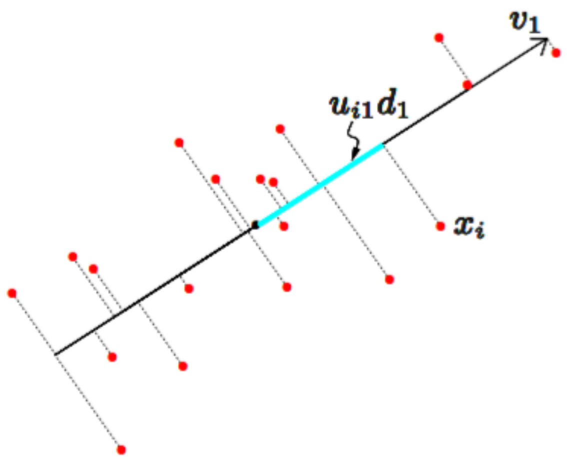
PCA: a high-fidelity linear projection

Given $x_i \in \mathbb{R}^d$ and some $q < d$ consider

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2.$$

where $\mathbf{V}_q = [v_1, v_2, \ldots, v_q]$ is orthonormal:

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$



$\mathbf{V}_q$ are the first q eigenvectors of Σ

$\mathbf{V}_q$ are the first q *principal components*

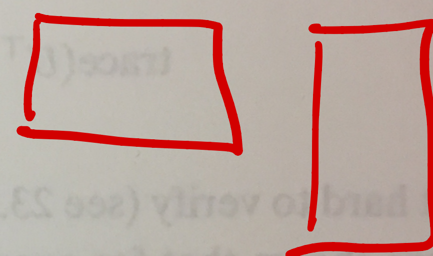
$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

Principal Component Analysis (PCA) projects $(\mathbf{X} - \mathbf{1}\bar{x}^T)$ down onto $\mathbf{V}_q$

$$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_q = \mathbf{U}_q \text{diag}(d_1, \ldots, d_q) \qquad \mathbf{U}_q^T \mathbf{U}_q = I_q$$

PCA: finding the principal components

PCA



input
A matrix of m examples $X \in \mathbb{R}^{m,d}$
number of components n

if $(m > d)$
$A = X^\top X$ → $d \times d$ →
Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be the eigenvectors of A with largest eigenvalues

else
$B = X X^\top$ → $m \times m$
Let $\mathbf{v}_1, \ldots, \mathbf{v}_n$ be the eigenvectors of B with largest eigenvalues
for $i = 1, \ldots, n$ set $\mathbf{u}_i = \frac{1}{\|X^\top \mathbf{v}_i\|} X^\top \mathbf{v}_i$

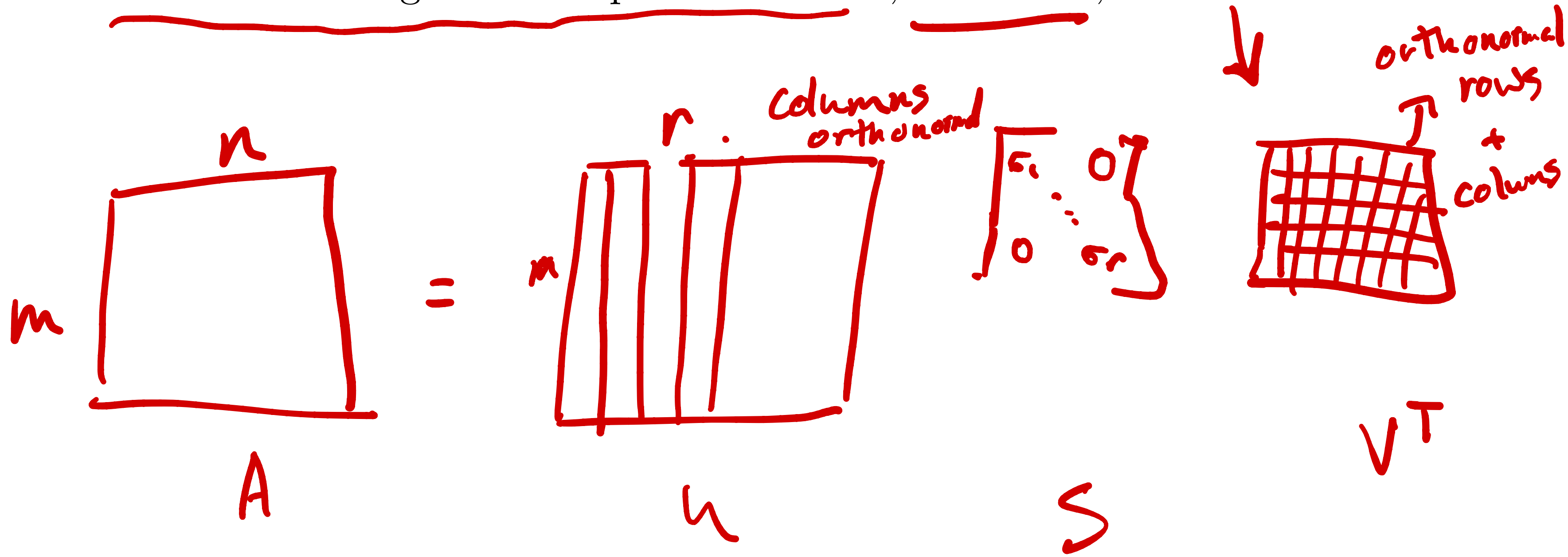
output: $\mathbf{u}_1, \ldots, \mathbf{u}_n$

How do we compute the principal components?

1. Power iteration
2. **Solving for a singular value decomposition (SVD)**

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.



Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$A^TA = (VS^TU^T)USV^T, \quad U^TU = I, \quad S^TS := D$$

$$= VDV^T$$

$$A^TAV_i = VDV^TV = V_iD_{i,i}$$

$$\mathbf{A}^T\mathbf{A}v_i = v_iD_{i,i} = v_is_{i,i}^2$$

$$\mathbf{A}\mathbf{A}^Tu_i = u_is_{i,i}^2$$

$$AA^T = (USV^T)(VS^TU^T) = UDU^T$$

$$AA^TU = UDU^TU = UD$$

Singular Value Decomposition (SVD)

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$$\mathbf{A}^T\mathbf{A}v_i = \mathbf{S}_{i,i}^2 v_i$$

$$\mathbf{A}\mathbf{A}^T u_i = \mathbf{S}_{i,i}^2 u_i$$

$\mathbf{V}$ are the first r eigenvectors of $\mathbf{A}^T\mathbf{A}$ with eigenvalues $\text{diag}(\mathbf{S})$
$\mathbf{U}$ are the first r eigenvectors of $\mathbf{A}\mathbf{A}^T$ with eigenvalues $\text{diag}(\mathbf{S})$

Computational complexity of SVD

Theorem (SVD): Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ where $\mathbf{S} \in \mathbb{R}^{r \times r}$ is diagonal with positive entries, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

at most r singular values

irrelevant | n - m | last columns of U

Computing the remaining economy-sized SVD takes time O(n m r)

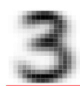
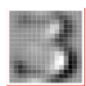
This is ⊥ of k!

PCA on MNIST

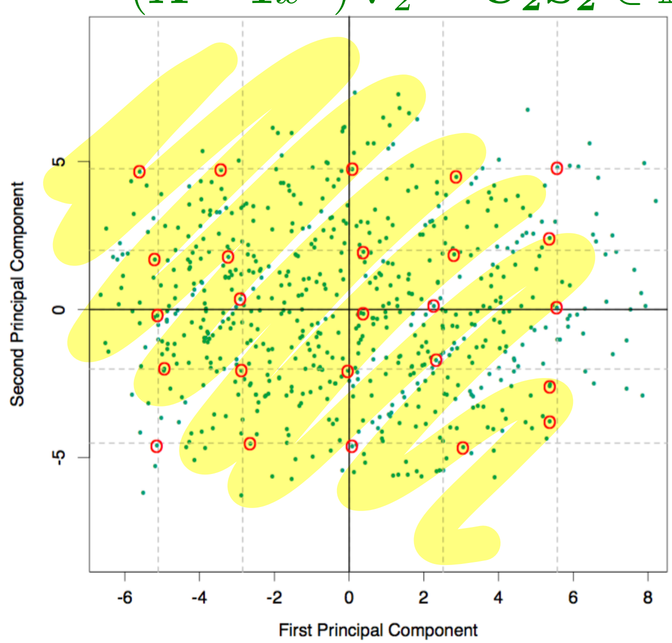
$\mathbf{V}_q$ are the first q eigenvectors of Σ and *SVD* $\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T$

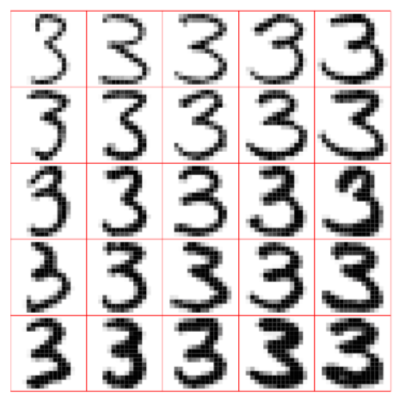
Handwritten 3's, 16x16 pixel image so that $x_i \in \mathbb{R}^{256}$

$$\hat{f}(\lambda) = \bar{x} + \lambda_1 v_1 + \lambda_2 v_2$$

$=$  $+ \lambda_1 \cdot$  $+ \lambda_2 \cdot$  .

$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_2 = \mathbf{U_2 S_2} \in \mathbb{R}^{n \times 2}$





diag($\mathbf{S}$)

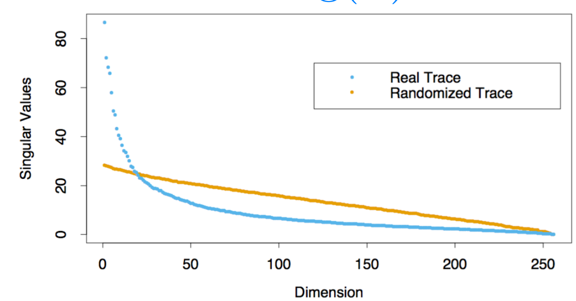


FIGURE 14.24. *The 256 singular values for the digitized threes, compared to those for a randomized version of the data (each column of* **X** *was scrambled).*

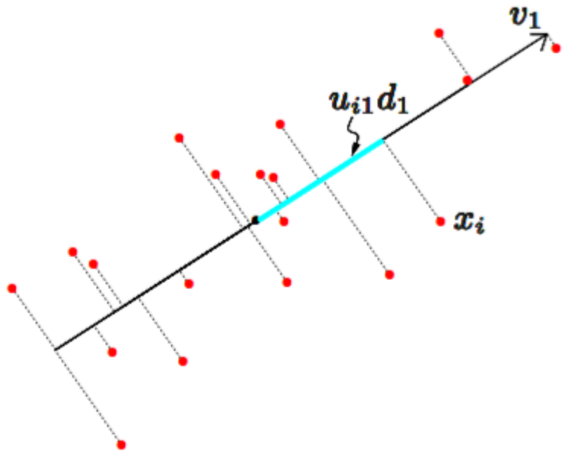
Linear projections

Given $x_i \in \mathbb{R}^d$ and some $q < d$ consider

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2.$$

where $\mathbf{V}_q = [v_1, v_2, \ldots, v_q]$ is orthonormal:

$$\mathbf{V}_q^T \mathbf{V}_q = I_q$$



$\mathbf{V}_q$ are the first q eigenvectors of Σ

$\mathbf{V}_q$ are the first q *principal components*

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

Principal Component Analysis (PCA) projects $(\mathbf{X} - \mathbf{1}\bar{x}^T)$ down onto $\mathbf{V}_q$

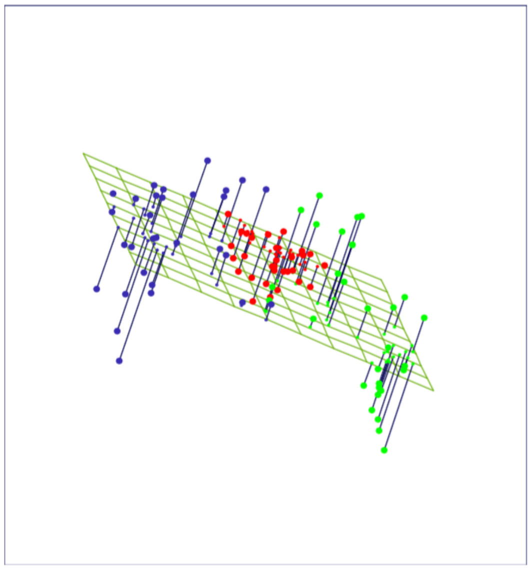
$$(\mathbf{X} - \mathbf{1}\bar{x}^T)\mathbf{V}_q = \mathbf{U}_q \text{diag}(d_1, \ldots, d_q) \qquad \mathbf{U}_q^T \mathbf{U}_q = I_q$$

Singular Value Decomposition defined as

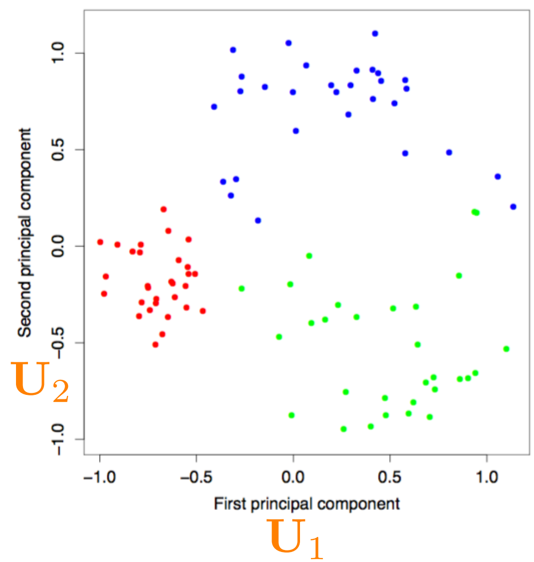
$$\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{USV}^T$$

SVD and PCA

$\mathbf{V}_q$ are the first q eigenvectors of Σ and *SVD* $\mathbf{X} - \mathbf{1}\bar{x}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T$



$\mathbf{X} - \mathbf{1}\bar{x}^T$



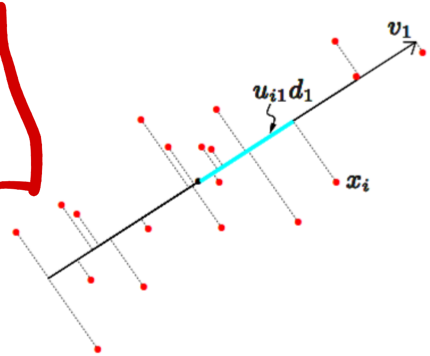
How do we compute the principal components?

1. **Power iteration**
2. Solving for a singular value decomposition (SVD)

Power method - one vector at a time

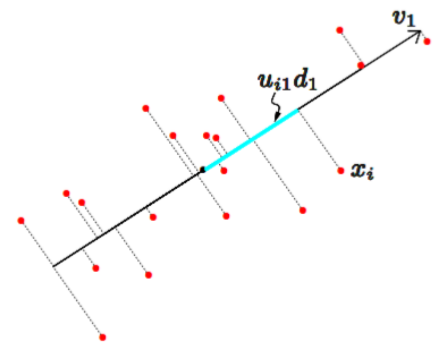
$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

$$v_* = \arg\max_v v^T \Sigma v$$



Power method - one vector found iteratively

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \qquad v_* = \arg\max_v \; v^T \Sigma v$$



$z_0 \sim \mathcal{N}(0, I)$

Iterate: $z_{t+1} = \dfrac{\Sigma z_t}{\|\Sigma z_t\|_2}$

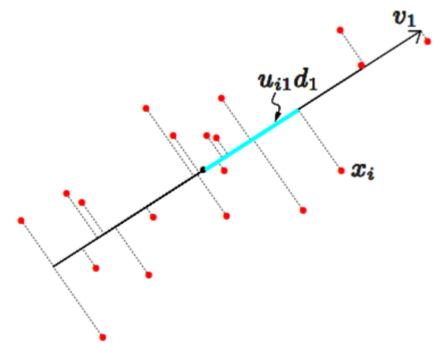
$$z_1 = \frac{\Sigma z_0}{\|\Sigma z_0\|}$$

$$z_2 = \frac{\Sigma z_1}{\|\Sigma z_1\|}$$

Power method - one vector found iteratively

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T$$

$$v_* = \arg\max_{v} v^T \Sigma v$$



$z_0 \sim \mathcal{N}(0, I)$

Iterate: $z_{t+1} = \frac{\Sigma z_t}{\|\Sigma z_t\|_2}$

To analyze write: $\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T$ $\quad z_t =: \mathbf{V}\alpha_t$

$X^T X$, S^2

$\alpha_t = V^T z_t$

$$z_{t+1} = V\alpha_{t+1} = \frac{\Sigma z_t}{\|\Sigma z_t\|}$$

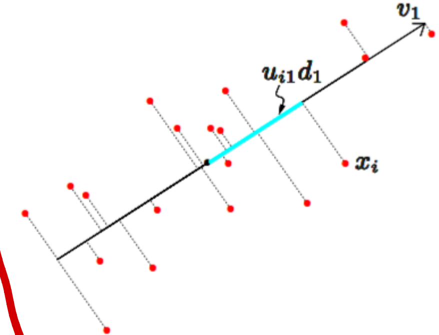
$$\alpha_{t+1} = V^T z_{t+1} = \frac{V^T \Sigma z_t}{\|\Sigma z_t\|} = \frac{D\alpha_t}{\|D\alpha_t\|} = \frac{D^t \alpha_0}{\|D^t \alpha_0\|}$$

$V^T \Sigma = V^T V D V^T = D V^T$ ($V^T V = I$)

$V^T \Sigma z_t = D V^T V \alpha_t = D\alpha_t$ ($V^T V = I$)

Power method - analysis

$$\Sigma := \sum_{i=1}^{N} (x_i - \bar{x})(x_i - \bar{x})^T \qquad v_* = \arg\max_v \; v^T \Sigma v$$



$z_0 \sim \mathcal{N}(0, I)$ Iterate: $z_{t+1} = \dfrac{\Sigma z_t}{\|\Sigma z_t\|_2}$

To analyze write: $\Sigma = \mathbf{V}\mathbf{D}\mathbf{V}^T \qquad z_t =: \mathbf{V}\alpha_t$

$$\alpha_{t+1} = \mathbf{V}^T z_{t+1} = \frac{\mathbf{V}^T \Sigma z_t}{\|\Sigma z_t\|} = \frac{\mathbf{D}\alpha_t}{\|\mathbf{D}\alpha_t\|} = \frac{\mathbf{D}^2 \alpha_{t-1}}{\|\mathbf{D}^2 \alpha_{t-1}\|} = \frac{\mathbf{D}^t \alpha_0}{\|\mathbf{D}^t \alpha_0\|}$$

$$\mathbf{D}^t = (\mathbf{D}_{1,1})^t (\mathbf{D}/\mathbf{D}_{1,1})^t \rightarrow (\mathbf{D}_{1,1})^t \mathbf{e}_1 \mathbf{e}_1^T \text{ since } \mathbf{D}_{i,i}/\mathbf{D}_{1,1} < 1$$

An application: Matrix completion

Given historical data on how users rated movies in past:

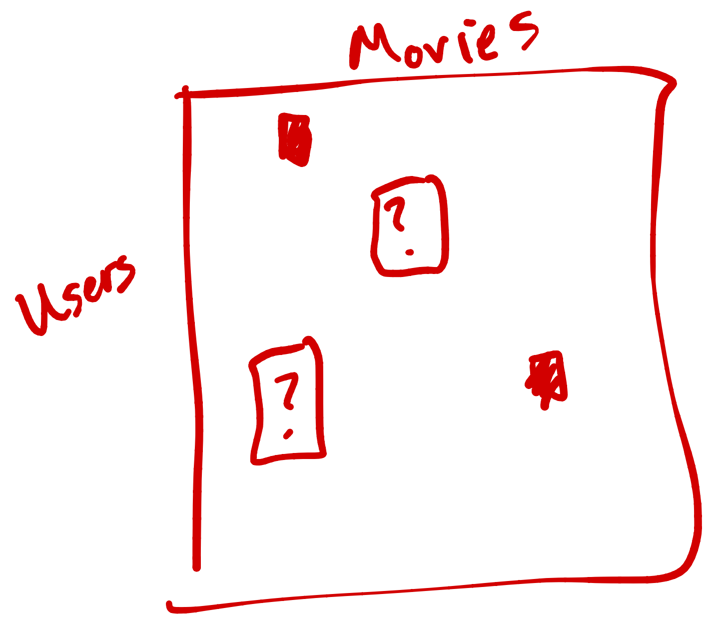
17,700 movies, 480,189 users, 99,072,112 ratings

NETFLIX

(Sparsity: 1.2%)

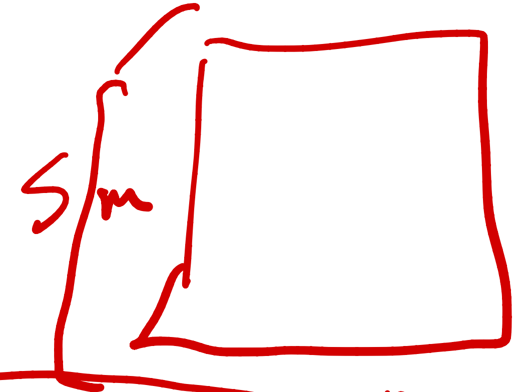
Predict how the same users will rate movies in the future (for $1 million prize)

						...
Alice	1	?	?	4	?	
Bob	?	2	5	?	?	
Carol	?	?	4	5	?	
Dave	5	?	?	?	4	
⋮						



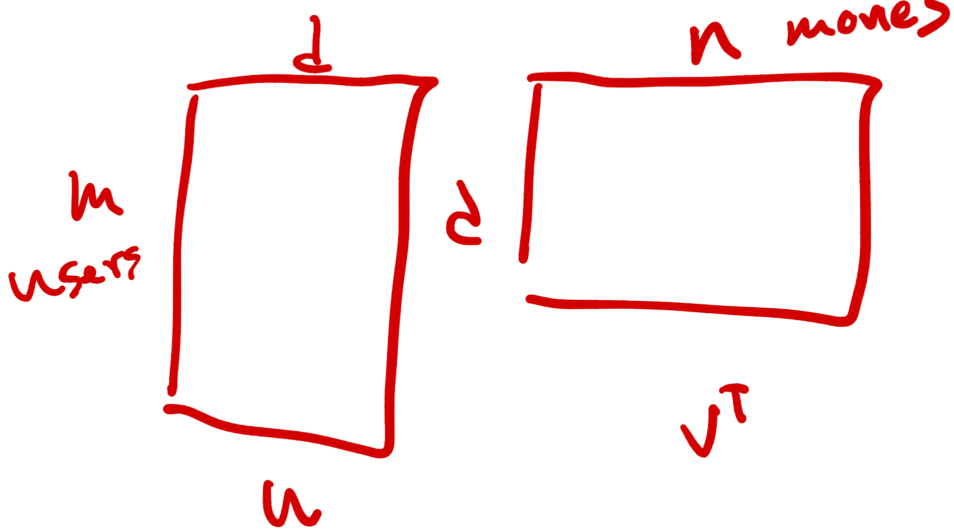
Matrix completion

n movies, m users, |*S*| ratings

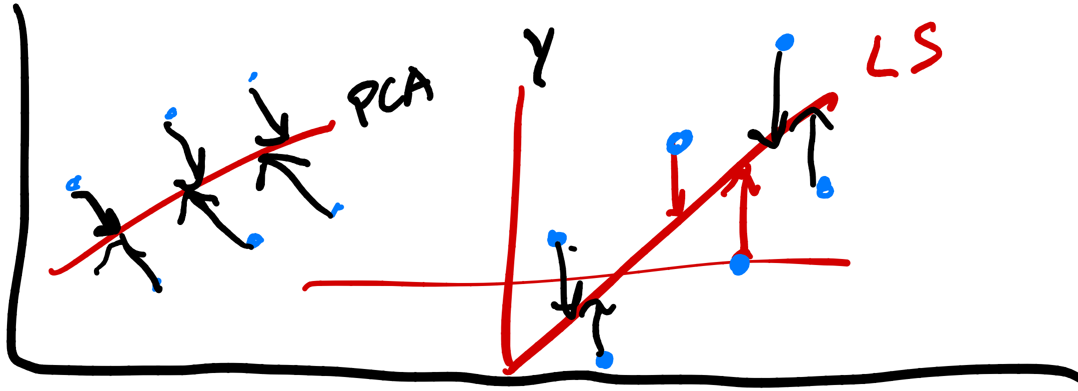


$$\underset{U\in\mathbb{R}^{m\times d}, V\in\mathbb{R}^{n\times d}}{\arg\min} \sum_{(i,j,s)\in\mathcal{S}} ||(UV^T)_{i,j} - s_{i,j}||_2^2$$

→ reconstruction error



Matrix completion



n movies, m users, $|S|$ ratings

$$\arg\min_{U \in \mathbb{R}^{m \times d}, V \in \mathbb{R}^{n \times d}} \sum_{(i,j,s) \in \mathcal{S}} ||(UV^T)_{i,j} - s_{i,j}||_2^2$$

How do we solve it? With full information? PCA

$S \in \mathbb{R}^{n \times m}$

(ie we have all ratings)

if not:

Need some assumptions

$\text{rank}(S) = k$

$\#$ entries in $S \geq f(k)$

PCA and SVD take-aways

PCA finds a d-dimensional representation with:

- Highest variance in any d-dimensional space
- Lowest reconstruction error
- spanned by the top d eigenvectors of covariance matrix

How to find the top d eigenvectors?

- SVD: $(X - I\,\boldsymbol{\mu}) := A = U\,S\,V^T$
    - ***V are the eigenvectors of A^TA***
    - U are the eigenvectors of AA^T
- Power method

This is one way to represent data in lower dimensions: there are others with other properties

- E.g., that approximately maintain pairwise distances

Expectation Maximization: an algorithmic template

UNIVERSITY of WASHINGTON

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

4. Each Center finds the centroid of the points it owns…

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

5. …and jumps there
6. …Repeat until terminated!

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Fixing centers, assign points to "most probable" cluster

Fixing assignment, compute "most likely" center

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Fixing centers, assign points to "most probable" cluster

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

Fixing assignment, compute "most likely" center

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Expectation: Fixing centers, assign points to "most probable" cluster

Maximization: Fixing assignment, compute "most likely" center

Recall Lloyd's algorithm

$$F(\mu, C) = \sum_{j=1}^{m} ||\mu_{C(j)} - x_j||^2$$

1. Ask user how many clusters they'd like. *(e.g. k=5)*
2. Randomly guess k cluster Center locations
3. Each datapoint finds out which Center it is closest to.
4. Each Center finds the centroid of the points it owns…
5. …and jumps there
6. …Repeat until terminated!

Expectation: Fixing centers, assign points to "most probable" cluster

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

Maximization: Fixing assignment, compute "most likely" center

What likelihood function?

There are k truncated Gaussians each generating data

Expectation: Fixing centers,
assign points to
"most probable" cluster

Maximization: Fixing assignment,
compute "most likely" center

What likelihood function?

There are k truncated Gaussians each generating data

Expectation: Fixing centers, assign points to "most probable" cluster

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

Maximization: Fixing assignment, compute "most likely" center

The Expectation Maximization template

Expectation:

Fix parameters,
estimate unobserved data

Maximization:

Fix unobserved data,
find MLE for parameters

Expectation: Fixing centers, assign points to "most probable" cluster

Maximization: Fixing assignment, compute "most likely" center

The Expectation Maximization template

Expectation:

Fix parameters,
estimate unobserved data

Maximization:

Fix unobserved data,
find MLE for parameters

Expectation: Fixing centers, assign points to "most probable" cluster

$$C^{(t)}(j) \leftarrow \arg\min_i ||\mu_i - x_j||^2$$

$$\mu_i^{(t+1)} \leftarrow \arg\min_\mu \sum_{j:C(j)=i} ||\mu - x_j||^2$$

Maximization: Fixing assignment, compute "most likely" center

Why use this template?

Expectation:

Fix parameters,
estimate unobserved
data

Maximization:

Fix unobserved data,
find MLE for parameters

Usually, the joint optimization problem is hard to solve (e.g., finding the global optimum to k-means)