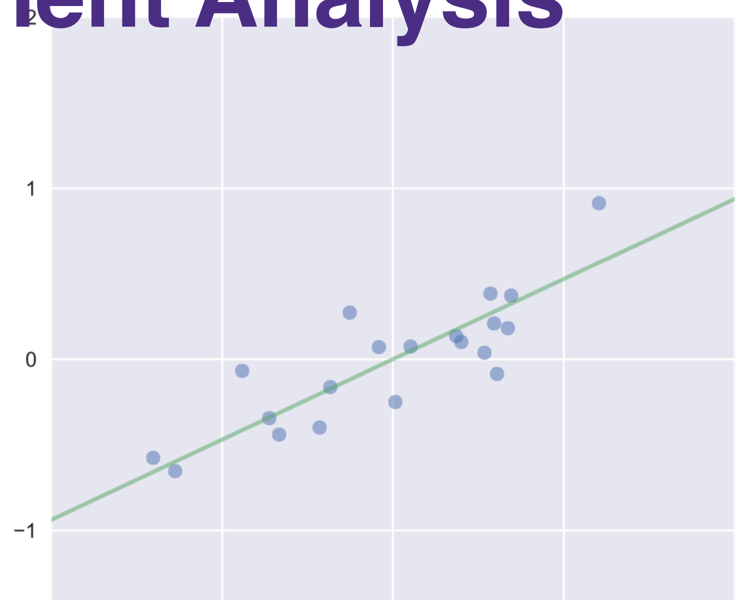
# Principal Component Analysis



# Principal components is the subspace that minimizes the reconstruction error

$$\underset{u_1,\ldots,u_r}{\text{minimize}} \; \frac{1}{n}\sum_{i=1}^{n} \|x_i - p_i\|_2^2$$

$$p_i = \sum_{j=1}^{r} (u_j^T x_i) u_j = \mathbf{U}\mathbf{U}^T x_i$$

where $\mathbf{U} = [u_1 \quad u_2 \quad \cdots \quad u_r] \in \mathbb{R}^{d\times r}$

$$\underset{U}{\text{minimize}} \; \frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathbf{U}\mathbf{U}^T x_i\|_2^2$$

$$\text{subject to } \; \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$

Q. How do we solve this optimization?

# Minimizing reconstruction error to find principal components

$$\underset{U}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n}\|x_i - \mathbf{U}\mathbf{U}^T x_i\|_2^2$$

$$\text{subject to} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$

$$\frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - 2x_i^T U U^T x_i + x_i^T U \underbrace{U^T U}_{I} U^T x_i$$

$$= \frac{1}{n}\sum_{i=1}^{n} x_i^T x_i - x_i^T U U^T x_i$$

$$= \underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i^T x_i}_{\text{Const.}} - \frac{1}{n}\sum_{i=1}^{n} \underbrace{x_i^T U U^T x_i}_{\sum_{j=1}^{r} (x_i^T u_j)^2}$$

$$U = [u_1 \ u_2 \ \cdots \ u_r]$$

$$\underset{u_1 \cdots u_r}{\text{minimize}} \ -\sum_{j=1}^{r} \frac{1}{n}\sum_{i=1}^{n} (x_i^T u_j)^2$$

$$\text{s.t.} \quad U^T U = I.$$

# Minimizing reconstruction error to find principal components

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i - UU^T x_i\|_2^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} \left\{ \|x_i\|_2^2 - 2x_i^T UU^T x_i + x_i^T U \underbrace{U^T U}_{=\mathbf{I}} U^T x_i \right\}$$

$$= \underbrace{\frac{1}{n}\sum_{i=1}^{n} \|x_i\|_2^2}_{\text{does not depend on } U} - \frac{1}{n}\sum_{i=1}^{n} x_i^T UU^T x_i$$

$$= C - \sum_{j=1}^{r} \underbrace{\frac{1}{n}\sum_{i=1}^{n} (u_j^T x_i)^2}_{\text{Variance in direction } u_j}$$

$$\underset{U}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} \|x_i - \mathbf{U}\mathbf{U}^T x_i\|_2^2$$

$$\text{subject to} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$

Variance along $u_j$

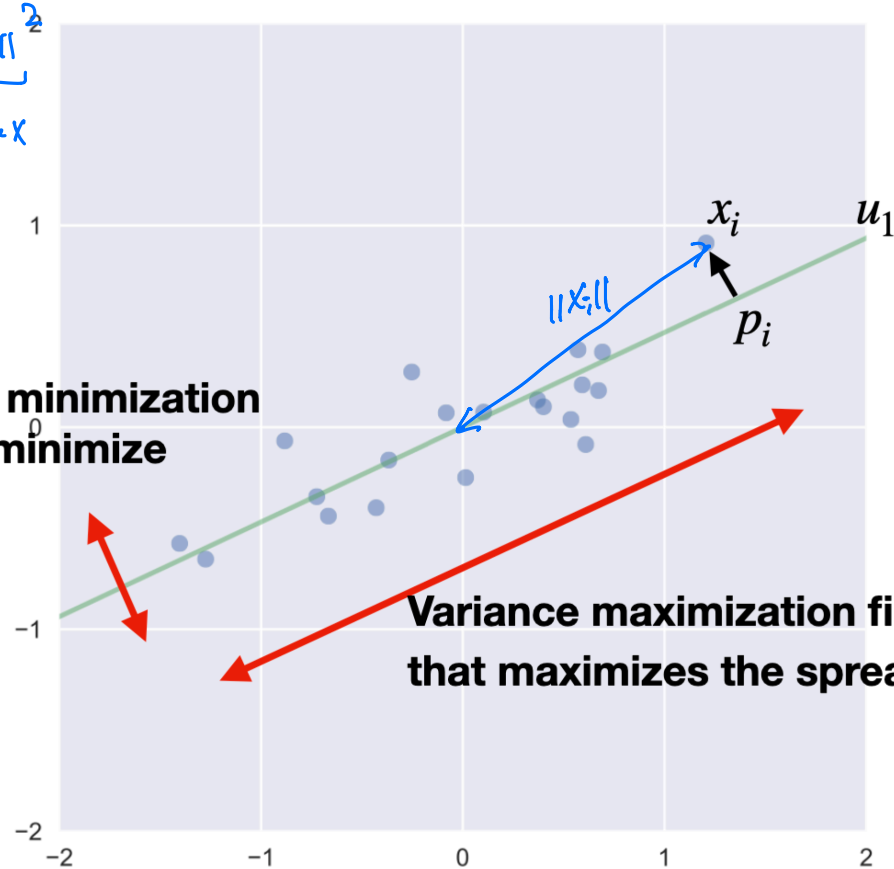
$$\underset{U}{\text{maximize}} \quad \sum_{j=1}^{r} \boxed{\frac{1}{n}\sum_{i=1}^{n} (u_j^T x_i)^2}$$

$$\text{subject to} \quad \mathbf{U}^T\mathbf{U} = \mathbf{I}_{r\times r}$$

# Variance maximization vs. reconstruction error minimization

- both give the same principal components as optimal solution

$$\|x_i\|_2^2 = \underbrace{\|x_i - p_i\|^2}_{\min} + \underbrace{\|p_i\|^2}_{\max}$$




**Reconstruction error minimization finds directions that minimize the distances to $p_i$'s**

**Variance maximization finds directions that maximizes the spread of $p_i$'s**

# Maximizing variance to find principal components

$$\underset{U}{\text{maximize}} \quad \sum_{j=1}^{r} \frac{1}{n} \sum_{i=1}^{n} (u_j^T x_i)^2$$

$$\text{subject to} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}_{r \times r}$$

We will solve it for $r = 1$ case,
and the general case follows similarly

$$\underset{u:\|u\|_2=1}{\text{maximize}} \quad \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2 \quad = \frac{1}{n} \sum_{i=1}^{n} u^T x_i x_i^T u$$

$$C = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$$

$$\underset{u:\|u\|_2=1}{\text{maximize}} \quad u^T C u$$

# Maximizing variance to find principal components

$C = \frac{1}{n}\sum_{i=1}^{n} \boxed{x_i x_i^T}$
PSD

$$\text{maximize}_u \; u^T \mathbf{C} u \qquad (a)$$

$$\textbf{subject to} \quad \|u\|_2^2 = 1$$

- we first claim that this optimization problem has the same optimal solution as the following **inequality constrained** problem

$$\text{maximize}_u \; u^T \mathbf{C} u \qquad (b)$$

$$\textbf{subject to} \quad \|u\|_2^2 \leq 1$$

- the reason is that, because $u^T \mathbf{C} u \geq 0$ for all $u \in \mathbb{R}^d$, the optimal solution of $(b)$ has to have $\|u\|_2^2 = 1$

$\hat{U} = \sqrt{\frac{10}{9}}\, U^* \longrightarrow \frac{10}{9}$ larger

- if it did not have $\|u\|_2^2 = 1$, say $\|\hat{u}\|_2^2 = 0.9$, then we can just multiply this $u$ by a constant factor of $\boxed{\sqrt{10/9}}$ and increase the objective by a factor of $10/9$ while still satisfying the constraints

$$\text{maximize}_u \; u^T \mathbf{C} u \qquad (b)$$

$$\textbf{subject to} \quad \|u\|_2^2 \leq 1$$

- we are maximizing the variance, while **keeping $u$ small**
- this can be reformulated as an unconstrained problem, with Lagrangian encoding, to move the constraint into the objective

$$\text{maximize}_u \quad \underbrace{u^T \mathbf{C} u - \lambda \|u\|_2^2}_{F_\lambda(u)} \qquad (c)$$

- this encourages small $u$ as we want, and we can make this connection precise: there exists a (unknown) choice of $\lambda$ such that the optimal solution of $(c)$ is the same as the optimal solution of $(b)$
- further, for this choice of $\lambda$, the optimal $u$ has $\|u\|_2 = 1$

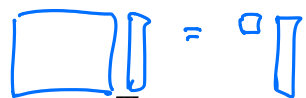
# Solving the unconstrained optimization

$$\text{maximize}_u \quad \underbrace{u^T \mathbf{C} u - \lambda \|u\|_2^2}_{F_\lambda(u)}$$

- to find such $\lambda$ and the corresponding $u$, we solve the unconstrained optimization, by setting the gradient to zero

$$\nabla_u F_\lambda(u) \;=\; 2\mathbf{C}u - 2\lambda u \quad = \; 0$$

- the candidate solution satisfies: $\mathbf{C}u^* = \lambda u^*$, i.e. an eigenvector of $\mathbf{C}$



$$\text{maximize}_u \; u^T \mathbf{C} u$$

$$\textbf{subject to} \quad \|u\|_2^2 = 1$$

- let $(\lambda^{(1)}, u^{(1)})$ denote the largest eigenvalue and corresponding eigenvector of $\mathbf{C}$, with norm one, i.e. $\|u^{(1)}\|_2^2 = 1$
- The maximum is achieved when $u = u^{(1)}$

# The principal component analysis

- so far we considered finding ONE principal component $u \in \mathbb{R}^d$
- it is the eigenvector corresponding to the maximum eigenvalue of the covariance matrix

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X} \in \mathbb{R}^{d \times d}$$

- We can use Singular Value Decomposition (SVD) to find such eigen vector
- note that is the data is not centered at the origin, we should re-center the data before applying SVD

- in general we define and use multiple principal components
- if we need $r$ principal components, we take $r$ eigenvectors corresponding to the largest $r$ eigenvalues of $\mathbf{C}$

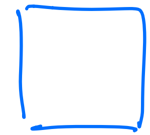
# Algorithm: Principal Component Analysis

- **input**: data points $\{x_i\}_{i=1}^n$, target dimension $r \ll d$
- **output**: $r$-dimensional subspace $U$

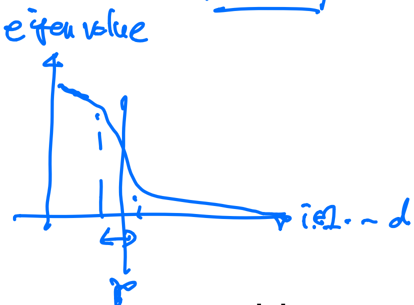
$X_i \sim$ face

- **algorithm**:

$U_1 \cdots U_{25}$



  - compute mean $\bar{x} = \dfrac{1}{n}\displaystyle\sum_{i=1}^{n} x_i$




  - compute covariance matrix

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T$$

  - let $(u_1, \ldots, u_r)$ be the set of (normalized) eigenvectors with corresponding to the largest $r$ eigenvalues of $\mathbf{C}$
  - return $\mathbf{U} = [u_1 \quad u_2 \quad \cdots \quad u_r]$

- further the data points can be represented compactly via

$$a_i = \mathbf{U}^T(x_i - \bar{x}) \in \mathbb{R}^r$$

# Singular Value Decomposition (SVD)

$V = \begin{bmatrix} v_1 & v_2 & \cdots & v_r \end{bmatrix}$

**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, $\mathbf{U}^T\mathbf{U} = I$, $\mathbf{V}^T\mathbf{V} = I$.

$A = \begin{bmatrix} \; \end{bmatrix} \begin{bmatrix} \ddots \end{bmatrix} \begin{bmatrix} \; \end{bmatrix}$, $\begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_r \end{bmatrix}$

$V S \underbrace{U^T U}_{=I} S \boxed{V^T \cdot v_i}^{e_i} = V \cdot S^2 \cdot e_i = \sigma_i^2 \cdot v_i$

What is $A^T A v_i = \sigma_i^2 v_i$

$AA^T = U S^2 U^T$

What is $AA^T u_i = \sigma_i^2 \cdot u_i$

$A^T A = V S^2 V^T$

- $v_i$'s are the $r$ eigen vectors of $A^T A$ with corresponding eigen values $S_{jj}^2$'s
- $u_i$'s are the $r$ eigen vectors of $AA^T$ with corresponding eigen values $S_{jj}^2$'s
- Computing SVD takes $O(mnr)$ operations

# Singular Value Decomposition (SVD)

m=n

- Consider a full rank matrix $A \in \mathbb{R}^{m\times n}$ whose SVD is $A = USV^T$, and we want to find the best rank-$r$ approximation of $A$ that minimizes the error

$$\text{minimize}_{L\in\mathbb{R}^{m\times n}} \sum_{i=1}^{m}\sum_{j=1}^{n}(A_{i,j} - L_{i,j})^2$$

$$\text{subject to } \operatorname{rank}(L) = r$$

- The optimal rank-$r$ approximation is $U_{1:r}S_{1:r,1:r}V_{1:r}^T$

# How do we compute singular vectors?

- In practice: Lanczos method
- We will learn: power iteration
- Let $C = USU^T \in \mathbb{R}^{d \times d}$ be SVD of the matrix we want to compute the top one singular vector
  - $U = [u_1, u_2, \ldots, u_d]$ are the singular vectors (ordered in the decreasing order of the corresponding singular values)
  - We also assume $\lambda_1 > \lambda_2$ in order to ensure uniqueness of $u_1$

$V_0 \sim$ anything random

$$\tilde{v}_{t+1} \leftarrow Cv_t$$

$$v_{t+1} \leftarrow \frac{\tilde{v}_{t+1}}{\| \tilde{v}_{t+1} \|_2}$$

Claim: $V_t \xrightarrow[t \to \infty]{} U_1$

# Power iteration

1st singular value

$$u_1^T C = \lambda_1 \cdot u_1^T$$

$$\tilde{v}_{t+1} \leftarrow C v_t$$

$$v_{t+1} \leftarrow \frac{\tilde{v}_{t+1}}{\|\tilde{v}_{t+1}\|_2}$$

$$\|u_1 - v_{\underset{t+1}{1}}\|^2 = 1 - 2u_1^T v_1 + 1$$

$$= 2(1 - u_1^T v_1) = 2\left(1 - u_1^T \frac{C v_0}{\|C v_0\|_2}\right)$$

$$= 2\left(1 - \frac{\lambda_1}{\|C v_0\|_2} \cdot u_1^T v_0\right)$$

$$= 2\left(1 - u_1^T v_0 + u_1^T v_0 - \frac{\lambda_1}{\|C v_0\|_2} \cdot u_1^T v_0\right)$$

$$= \underbrace{\|u_1 - v_0\|^2}_{\text{original dist.}} - 2\underbrace{\left(\frac{\lambda_1}{\|C v_0\|_2} - 1\right)}_{>0} \cdot u_1^T v_0 \leq \|u_1 - v_0\|^2$$

$\boxed{\text{equal only if } v_t = u_1}$

$$\|C \cdot v_0\|_2 < \lambda_1$$

# Power iteration for general rank-$r$

$$\tilde{v}_{t+1} \leftarrow C v_t$$

$$v_{t+1} \leftarrow \frac{\tilde{v}_{t+1}}{\| \tilde{v}_{t+1} \|_2}$$

$$C - \lambda_1 \cdot u_1 u_1^T \longrightarrow C$$

- First approach:
  - Repeat r times
    - Run rank-1 power iteration
    - Subtract $C - (v_t^T C v_t) v_t v_T^T \longrightarrow C$

$u_1, u_2, u_3, \cdots, u_r$

- Second approach:

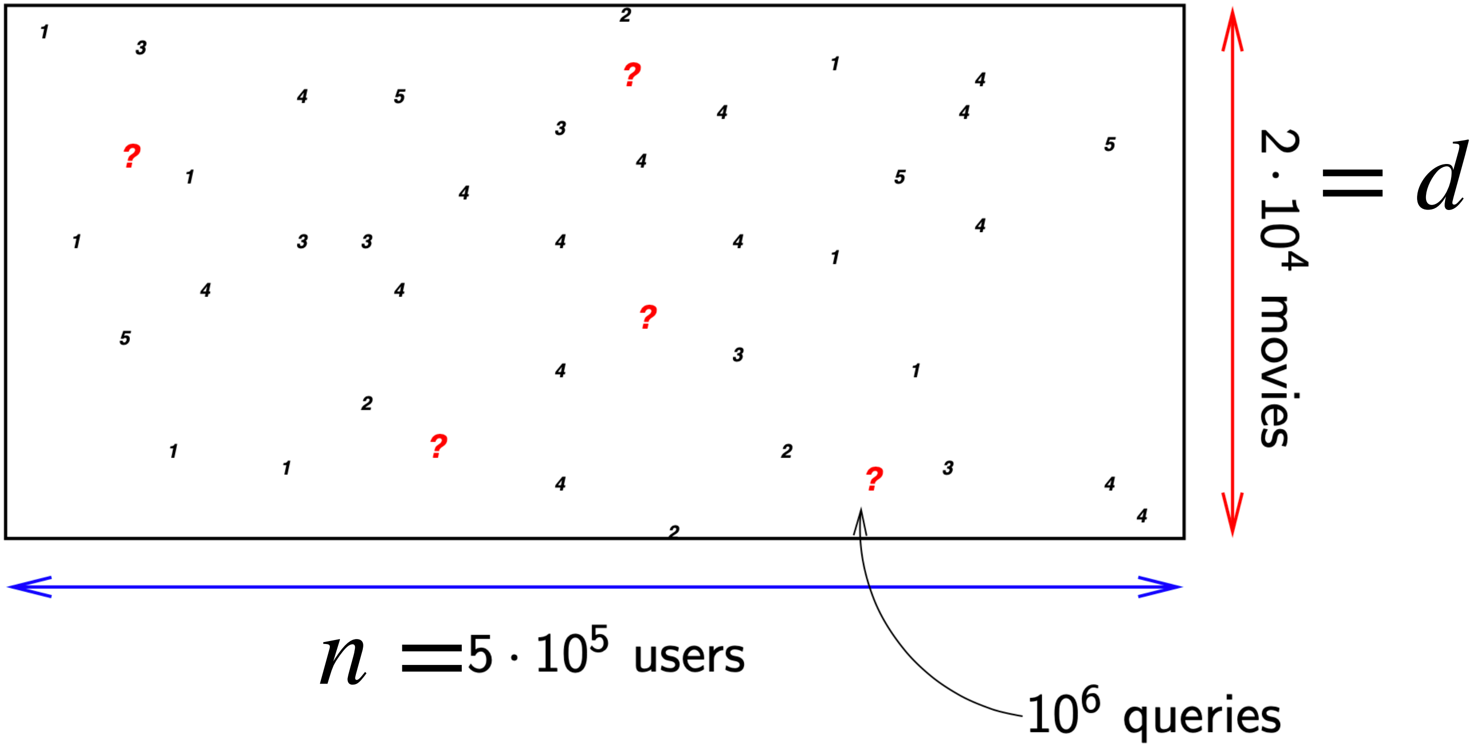
$$V_0 = \begin{array}{cc} & r \\ d & \begin{bmatrix} \quad \end{bmatrix} \end{array}, \quad s.t \quad V_0^T V_0 = I$$

$$\tilde{V}_{t+1} \leftarrow C \cdot V_t$$

$$V_{t+1} \leftarrow \underbrace{Q.R.}_{\text{QR decomposition.}} \left( \underbrace{\tilde{V}_{t+1}} \right)$$

# Matrix completion for recommendation systems

Netflix challenge dataset




- users provide ratings on a few movies, and we want to predict the missing entries in this ratings matrix, so that we can make recommendations
- without any assumptions, the missing entries can be anything, and no prediction is possible

# Matrix completion problem

- however, the ratings are not arbitrary, but people with similar tastes rate similarly
- such structure can be modeled using low dimensional representation of the data as follows
- we will find a set of principal component vectors
  $\mathbf{U} = [u_1 \quad u_2 \quad \cdots \quad u_r] \in \mathbb{R}^{d \times r}$ = d × r } # of movies
- such that that ratings $x_i \in \mathbb{R}^d$ of user $i$, can be represented as

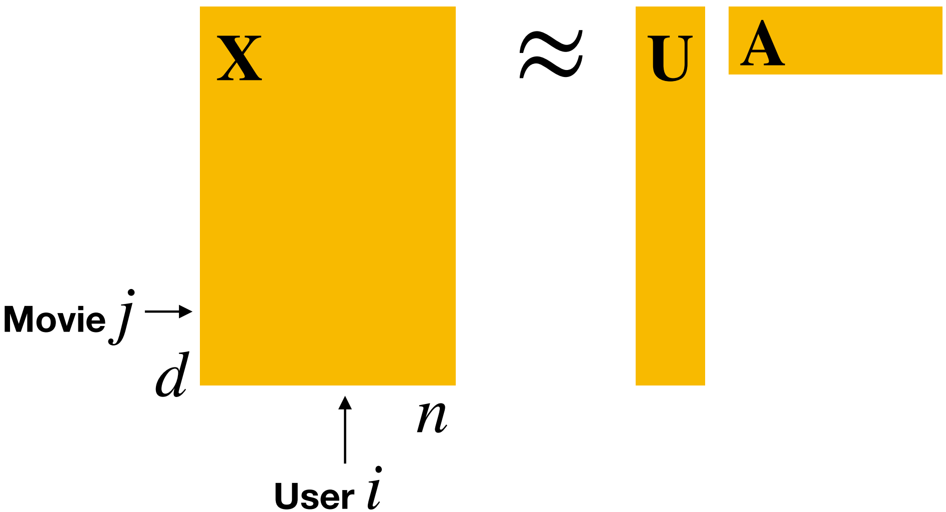
  user i's ratings d × 1

$$
\begin{aligned}
x_i &= a_i[1]u_1 + \cdots a_i[r]u_r \\
&= \mathbf{U}a_i
\end{aligned}
$$

  for some lower-dimensional $a_i \in \mathbb{R}^r$ for $i$-th user and some $r \ll d$
- for example, $u_1 \in \mathbb{R}^d$ means how horror movie fans like each of the $d$ movies,
- and $a_i[1]$ means how much user $i$ is fan of horror movies

# Matrix completion

- let $\mathbf{X} = [x_1 \quad x_2 \quad \cdots \quad x_n] \in \mathbb{R}^{d\times n}$ be the ratings matrix, and assume it is fully observed, i.e. we know all the entries
- then we want to find $\mathbf{U} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} = [a_1 \quad a_2 \quad \cdots \quad a_n] \ \in \mathbb{R}^{r\times n}$ that approximates $\mathbf{X}$

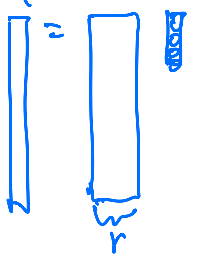


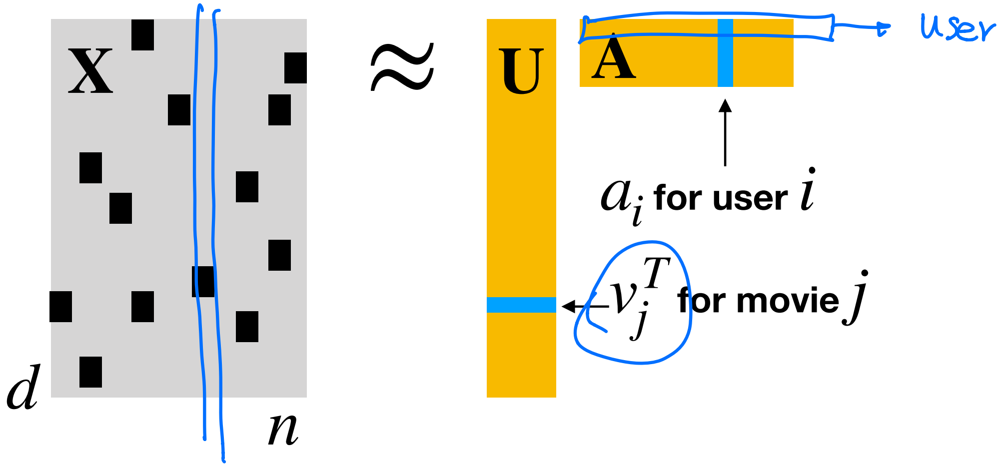

- if we **observe all entries** of $\mathbf{X}$, then we can find the best rank-$r$ approximation with SVD

# Matrix completion

- in practice, we only observe $\mathbf{X}$ partially
- let $S_{\text{train}} = \{(i_\ell, j_\ell)\}_{\ell=1}^{N}$ denote $N$ observed ratings for user $i_\ell$ on movie $j_\ell$

$X_i = U \cdot a_i$




- let $v_j^T$ denote the $j$-th row of $\mathbf{U}$ and $a_i$ denote $i$-th column of $\mathbf{A}$
- then user $i$'s rating on movie $j$, i.e. $\mathbf{X}_{ji}$ is approximated by $v_j^T a_i$, which is the inner product of $v_j$ (a column vector) and a column vector $a_i$
- we can also write it as $\langle v_j, a_i \rangle = v_j^T a_i$

# Matrix completion

- a natural approach to fit $v_j$'s and $a_i's$ to given training data is to solve

$$\text{minimize}_{\mathbf{U},\mathbf{A}} \sum_{(i,j)\in S_{\text{train}}} (\mathbf{X}_{ji} - v_j^T a_i)^2$$

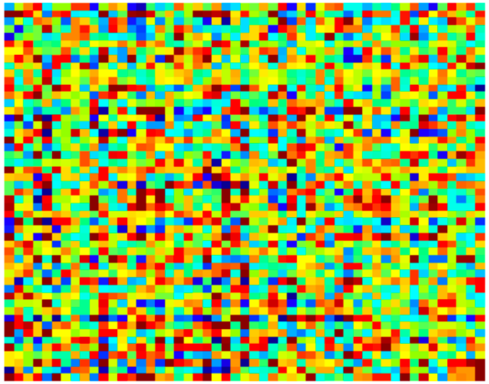
$d \times r$ $\quad$ $n \times r$ $\quad$ $\boxed{r(d+n)}$

- this can be solved, for example via gradient descent or alternating minimization
- this can be quite accurate, with small number of samples

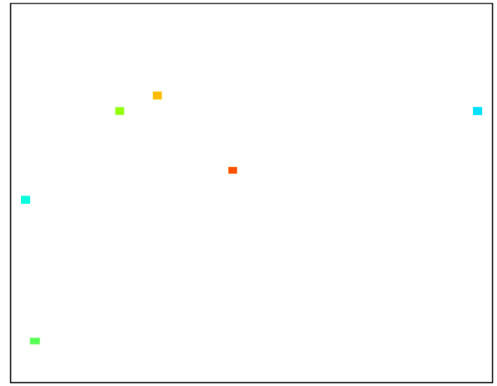
- Theorem [Keshavan,Montanari,Oh 2009]
  *Assume the ground truths* $\mathbf{X}$ *has rank* $r$, *then*
  *(a variant of) gradient descent finds the optimal solution if we observe more than* $c\, r\, (d+n)\, \log(dn)$ *entries at random positions*

# Example: $2000 \times 2000$ rank-8 random matrix

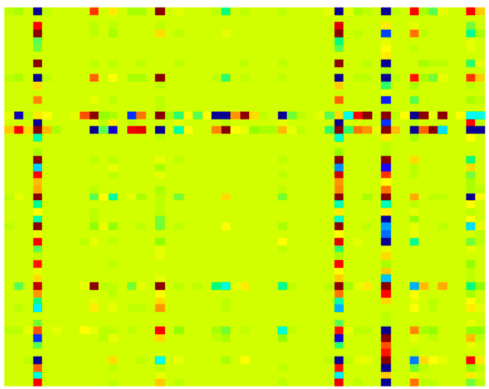
low-rank matrix $\mathbf{X}$



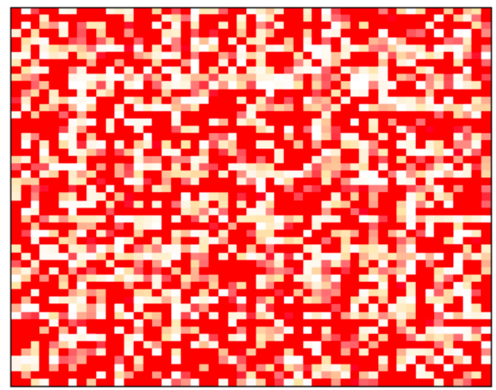
sampled matrix



Gradient descent output $\mathbf{UA}$



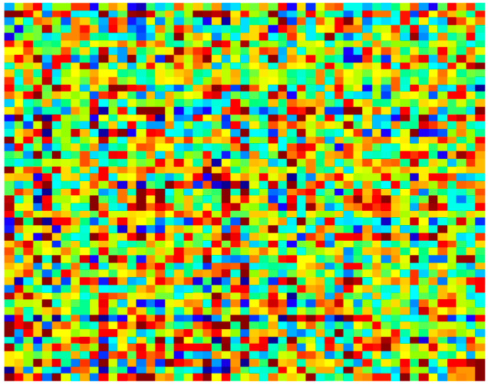
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



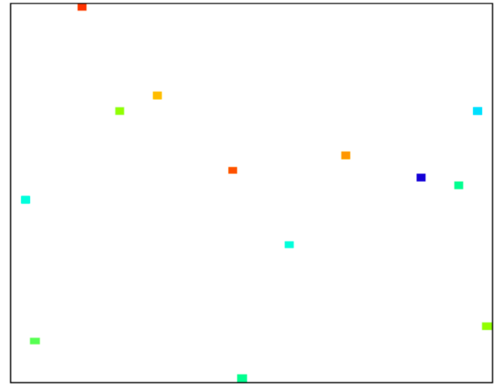
0.25% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

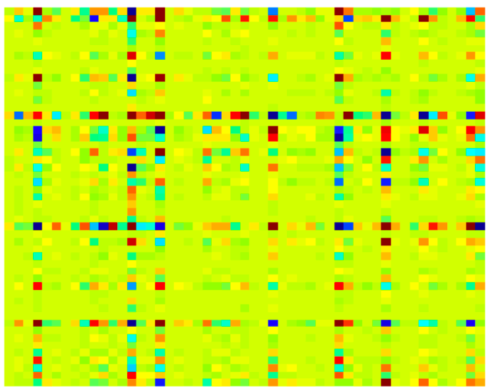
low-rank matrix $\mathbf{X}$



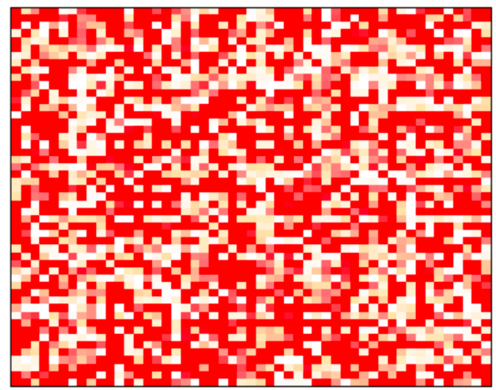
sampled matrix



Gradient descent output $\mathbf{UA}$



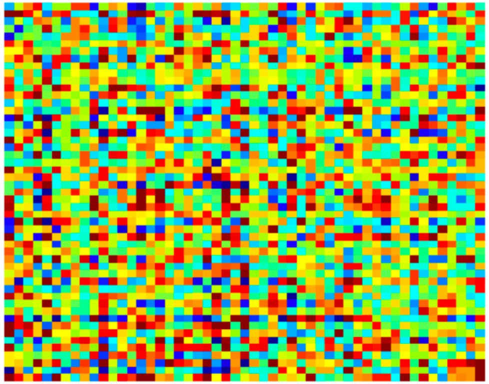
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



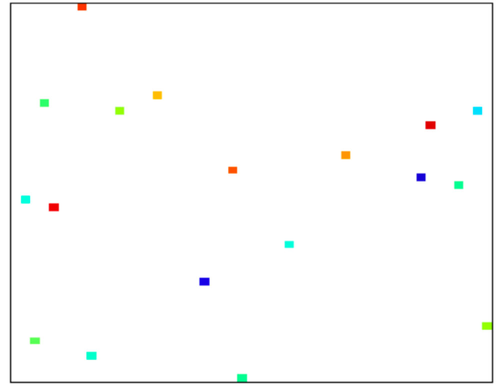
0.50% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

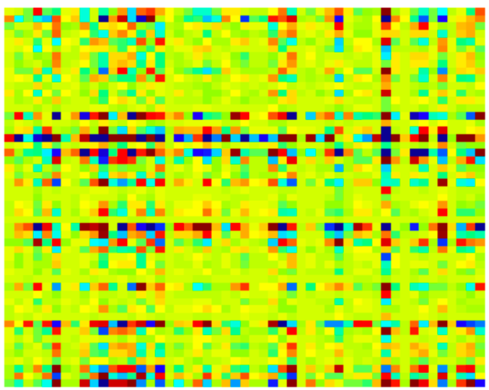
low-rank matrix $\mathbf{X}$



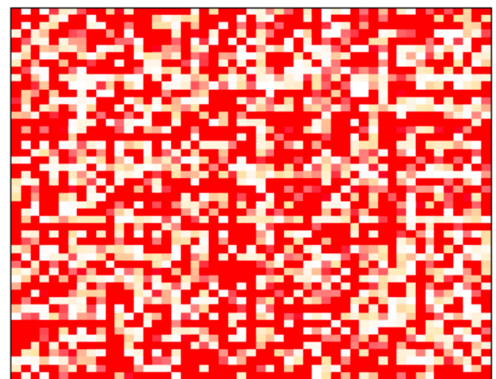
sampled matrix



Gradient descent output $\mathbf{UA}$



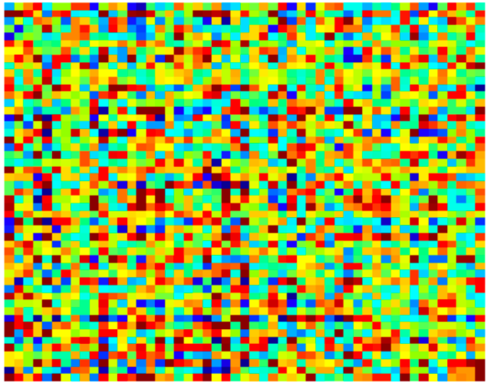
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



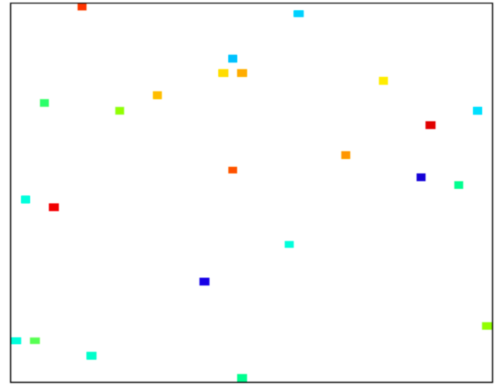
0.75% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

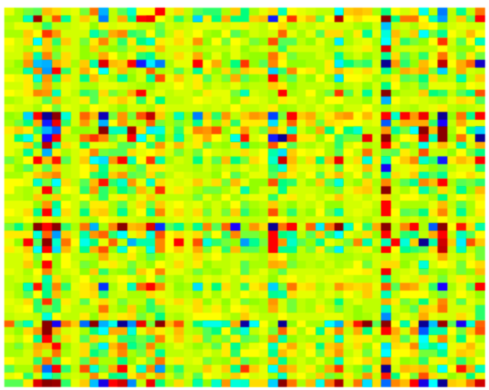
low-rank matrix $\mathbf{X}$



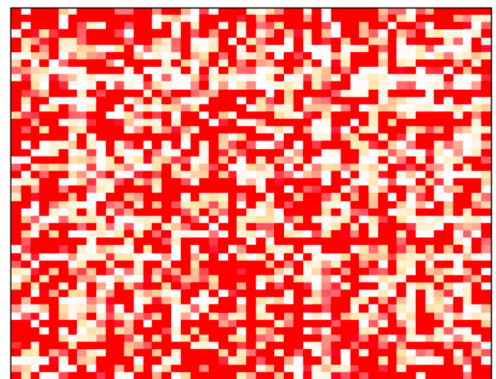
sampled matrix



Gradient descent output $\mathbf{UA}$



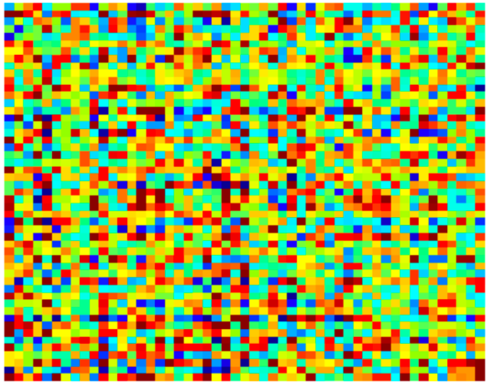
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



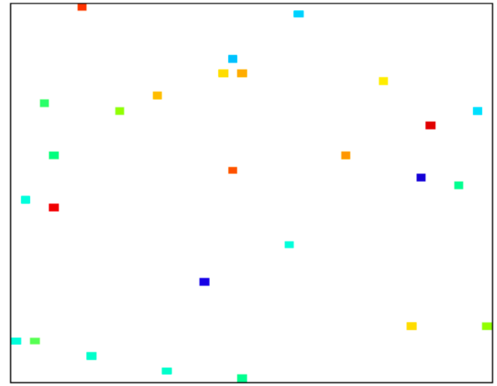
1.00% sampled

# Example: $2000 \times 2000$ rank-8 random matrix

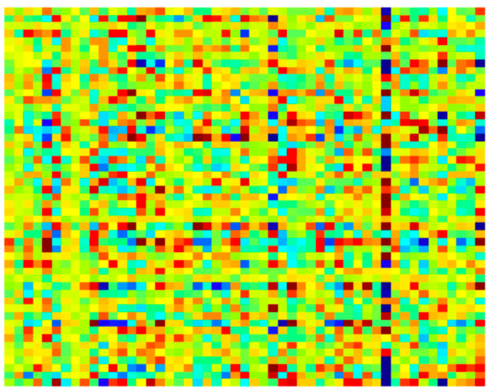
low-rank matrix $\mathbf{X}$



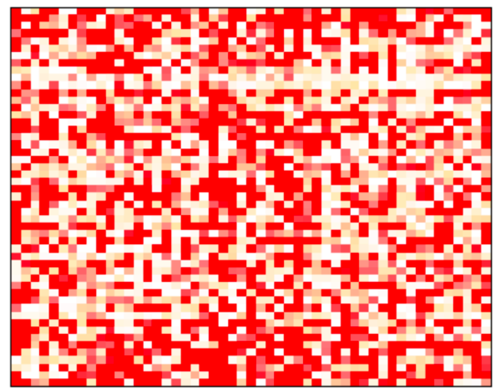
sampled matrix



Gradient descent output $\mathbf{UA}$

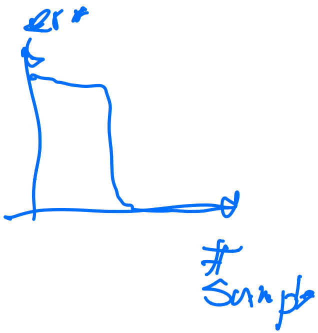


squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$

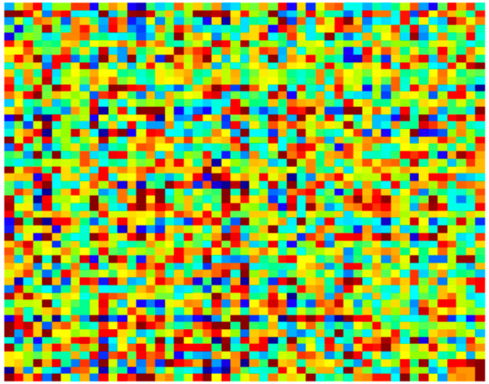


1.25% sampled

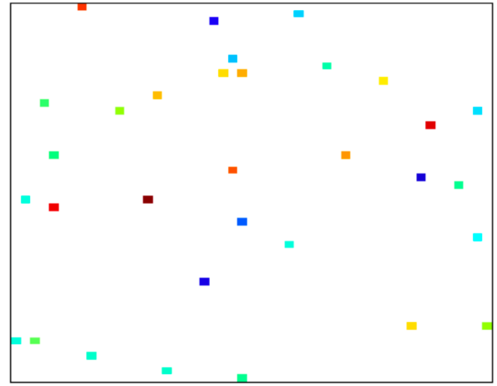
# Example: 2000 $\times$ 2000 rank-8 random matrix



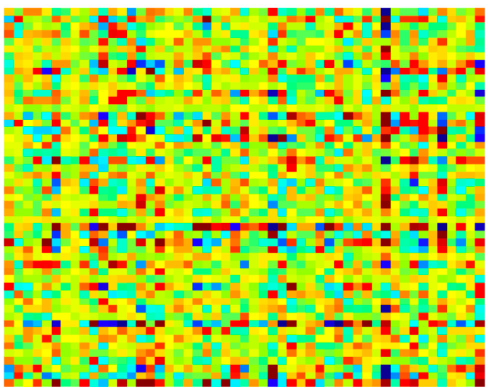

low-rank matrix $\mathbf{X}$



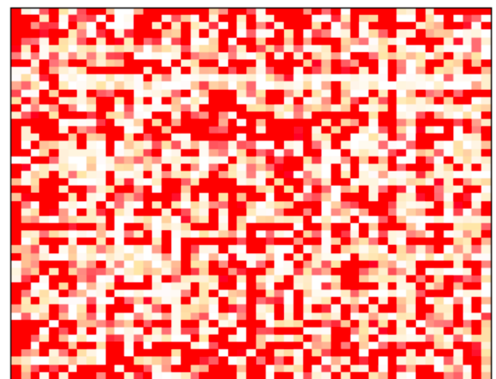
sampled matrix



Gradient descent output $\mathbf{UA}$

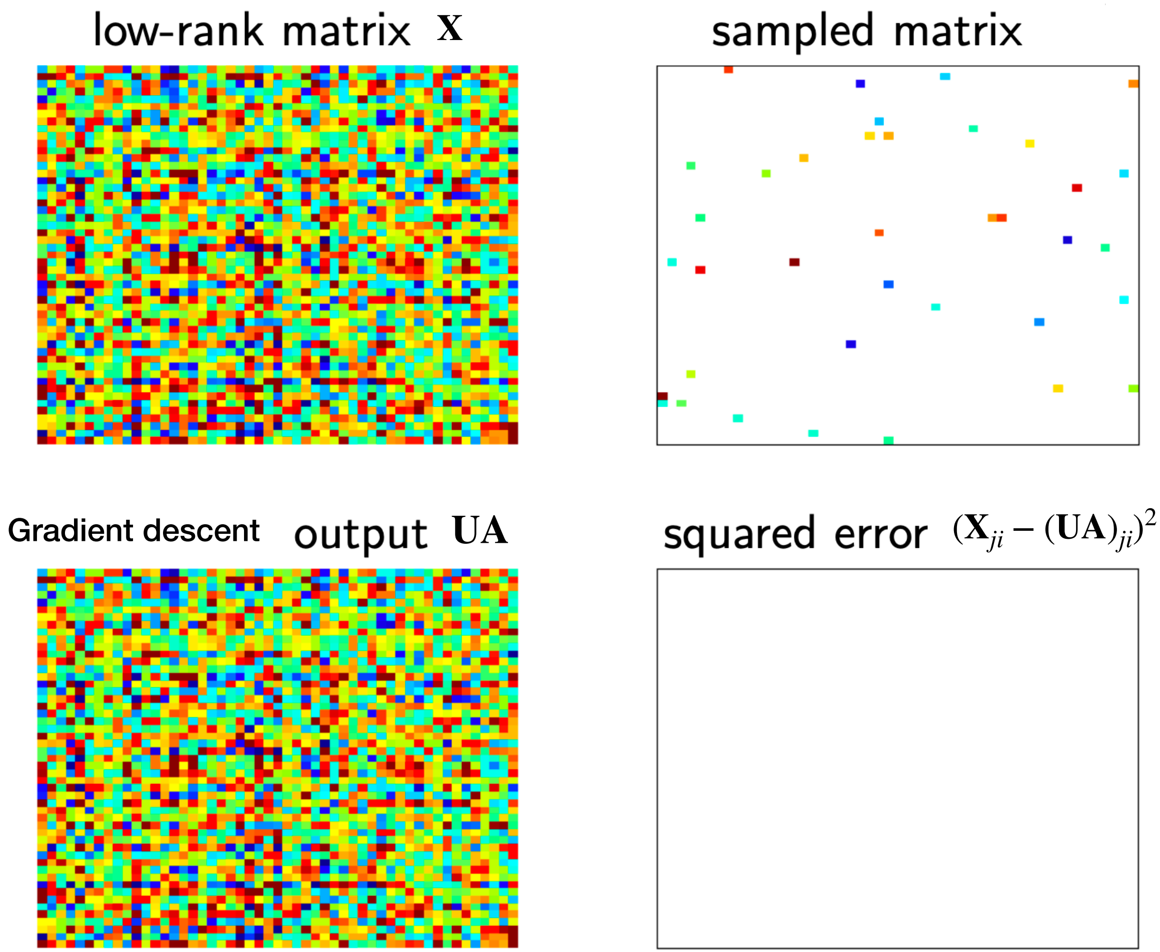


squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



1.50% sampled

# Example: 2000 $\times$ 2000 rank-8 random matrix




1.75% sampled