

Lecture 14: Principal Component Analysis

- reduce the dimension of the data to visualize and understand

- No OH for Sewoong this Friday → make up OH later.
- Homework 3, due Wednesday, November 19th midnight



CSE 446/546 2025 Autumn

- Supervised learning: $\{(x_i, y_i)\}_{i=1}^n$ $\xrightarrow{\text{Task}}$ Goal: predict y given x .
 - Linear models (lectures 1-9)
 - Linear regression
 - Ridge regression
 - LASSO regression $\} y_i \in \mathbb{R}$
 - Logistic regression $\} y_i \in \mathcal{Y}$
 - Non-linear models (lectures 10-13)
 - Kernel methods
 - Neural Networks
 - Non-parametric methods

Today \rightarrow

• Unsupervised learning (lectures 14-16):

- PCA/SVD
- Clustering

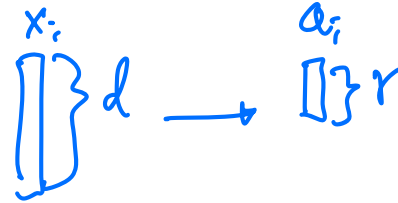
$\{x_i\}_{i=1}^n = D$

Goal
if
Task:

Self-supervised

learning. \rightarrow Pengwei's last lecture.

Motivation: dimensionality reduction



- Definition:
 - **Dimensionality reduction:** Given $\{x_i\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$, represent each data point in a lower dimension $r < d$ in order to visualize data, understand the pattern, de-noise, and compress data.
- Key idea: many real data have patterns that repeat over samples
- Can we exploit this redundancy? Can we find some patterns and use them?
- Toy example: Food Ratings Data with $n = 4$ samples in $d = 4$ dimensions

A table showing food ratings data. The columns are labeled Alice= x_1 , Bob= x_2 , Carolyn= x_3 , and Dave= x_4 . The rows are labeled kale, taco bell, sushi, and pop tarts. Handwritten blue annotations include $n=4$ above the columns, $d=4$ to the left of the rows, and a box around the first column labeled x_1 with $x_i[1]$ next to the first cell.

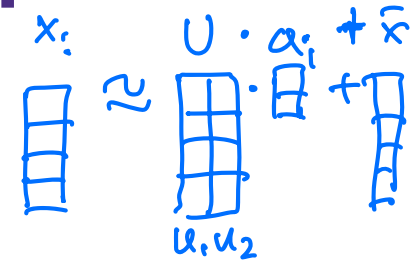
	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
kale	10	7	2	3
taco bell	1	2	9	6
sushi	2	9	7	10
pop tarts	7	6	3	2

- This is **unsupervised**, in the sense that there is no label, just input data

Motivation: dimensionality reduction

- Toy example: Food Ratings Data x_1

	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
kale	10	7	2	3
taco bell	1	2	9	6
sushi	2	9	7	10
pop tarts	7	6	3	2



- Goal of **dimensionality reduction** is, for example, visualization, compression, and/or finding pattern. One of the most important tool for dimensionality reduction is **Principal Component Analysis (PCA)**
- If we represent each data point approximately as a combination of (the mean and) two **principal vectors** $u_1, u_2 \in \mathbb{R}^d$

$$x_i \approx \bar{x} + a_i[1] \cdot u_1 + a_i[2] \cdot u_2$$

with $\bar{x} = \begin{bmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{bmatrix}$, $u_1 = \begin{bmatrix} 3 \\ -3 \\ -3 \\ 3 \end{bmatrix}$, and $u_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$

Diagram illustrating the decomposition: A vertical vector x_i is shown on the left. To its right, a 2x2 grid labeled U is multiplied by a vertical vector a_i . This result is added to another vertical vector \bar{x} . The final result is a vertical vector x_i . The number 27 is written between the first two vectors.

Diagram illustrating the decomposition: A vertical vector x_i is shown on the left. To its right, a 2x2 grid labeled U is multiplied by a vertical vector a_i . This result is added to another vertical vector \bar{x} . The final result is a vertical vector x_i . The number 27 is written between the first two vectors.

Motivation: dimensionality reduction

	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
kale	10	7	2	3
taco bell	1	2	9	6
sushi	2	9	7	10
pop tarts	7	6	3	2

$$x_i \approx \bar{x} + a_i[1] \cdot u_1 + a_i[2] \cdot u_2$$

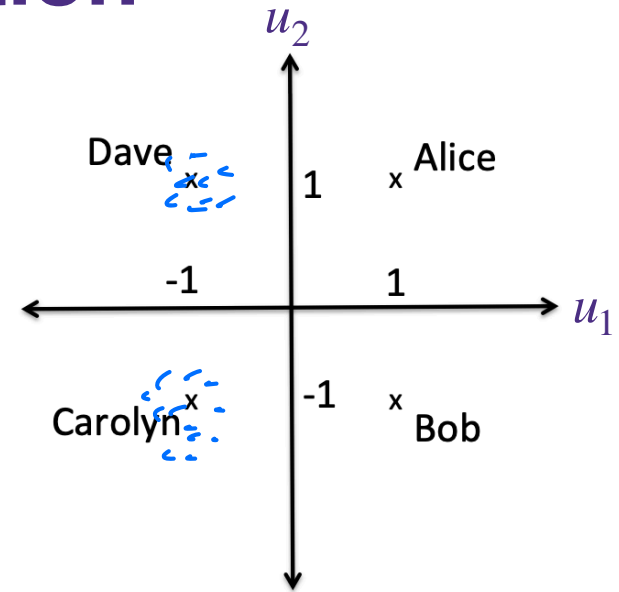
with $\bar{x} = \begin{bmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{bmatrix}$, $u_1 = \begin{bmatrix} 3 \\ -3 \\ -3 \\ 3 \end{bmatrix}$, and $u_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$. We can transform the data into

	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
$a_i[1]$	1	1	-1	-1
$a_i[2]$	1	-1	-1	1

What good is this new representation of the data?

Motivation: dimensionality reduction

	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
kale= $x_i[1]$	10 x_i	7	2	3
taco bell= $x_i[2]$	1	2	9	6
sushi= $x_i[3]$	2	9	7	10
pop tart= $x_i[4]$	7	6	3	2
$a_i[1]$	1 a_i	1	-1	-1
$a_i[2]$	1	-1	-1	1

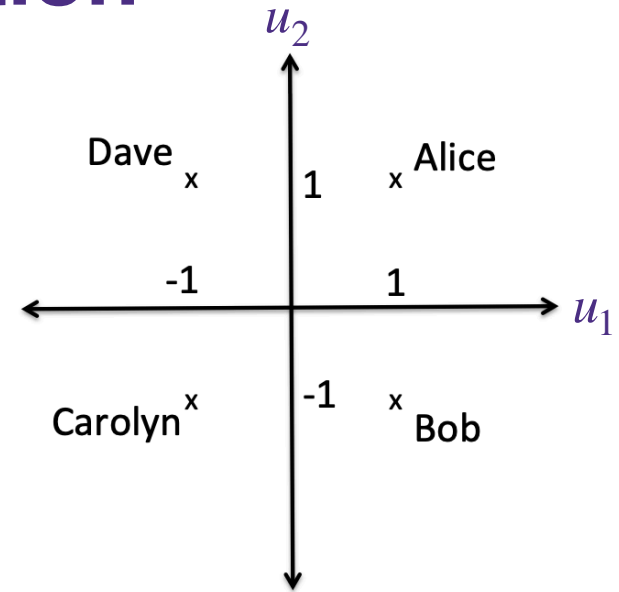


- **Visualization of data** can help understand the data and also group together data points with similar patterns (also known as clustering)

$$\bar{x} = \begin{bmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{bmatrix}, \quad u_1 = \begin{bmatrix} 3 \\ -3 \\ -3 \\ 3 \end{bmatrix}, \quad \text{and} \quad u_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Motivation: dimensionality reduction

	Alice= x_1	Bob= x_2	Carolyn= x_3	Dave= x_4
kale= $x_i[1]$	10	7	2	3
taco bell= $x_i[2]$	1	2	9	6
sushi= $x_i[3]$	2	9	7	10
pop tart= $x_i[4]$	7	6	3	2
$a_i[1]$	1	1	-1	-1
$a_i[2]$	1	-1	-1	1



- **Visualization of data** can help understand the data and also group together data points with similar patterns (also known as clustering)
- **Interpretation of data** to understand the underlying pattern
 - What does u_1 mean?
 - What does u_2 mean?

$u_1, u_2 \leftarrow PCA(\{x_1, \dots, x_n\})$

$$\bar{x} = \begin{bmatrix} 5.5 \\ 4.5 \\ 5 \\ 5.5 \end{bmatrix}, u_1 = \begin{bmatrix} 3 \\ -3 \\ -3 \\ 3 \end{bmatrix}, \text{ and } u_2 = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

- if normalized to be unit norm, u_1 and u_2 are called **principal components**

Goal of Principal Component Analysis (PCA)

- **PCA** approximately expresses each data point $x_i \in \mathbb{R}^d$ as a linear combination of k d -dimensional vectors $u_1, \dots, u_r \in \mathbb{R}^d$, such that

$$x_i \approx \bar{x} + a_i[1] \cdot u_1 + \dots + a_i[r] \cdot u_r$$

$$U = [u_1, \dots, u_r] \leftarrow \text{PCA}(\{x_i\}_{i=1}^n)$$

- **PCA** is a formal definition of the “best” such vectors u_1, \dots, u_r .

- **Pre-processing:**

- To avoid keeping track of the mean vector $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ and make the following linear algebra simple, we will assume that the data is pre-processed to remove the mean such that $\bar{x} = 0$.

Real example: eigen-face

$d=32 \times 32$ pixels per image

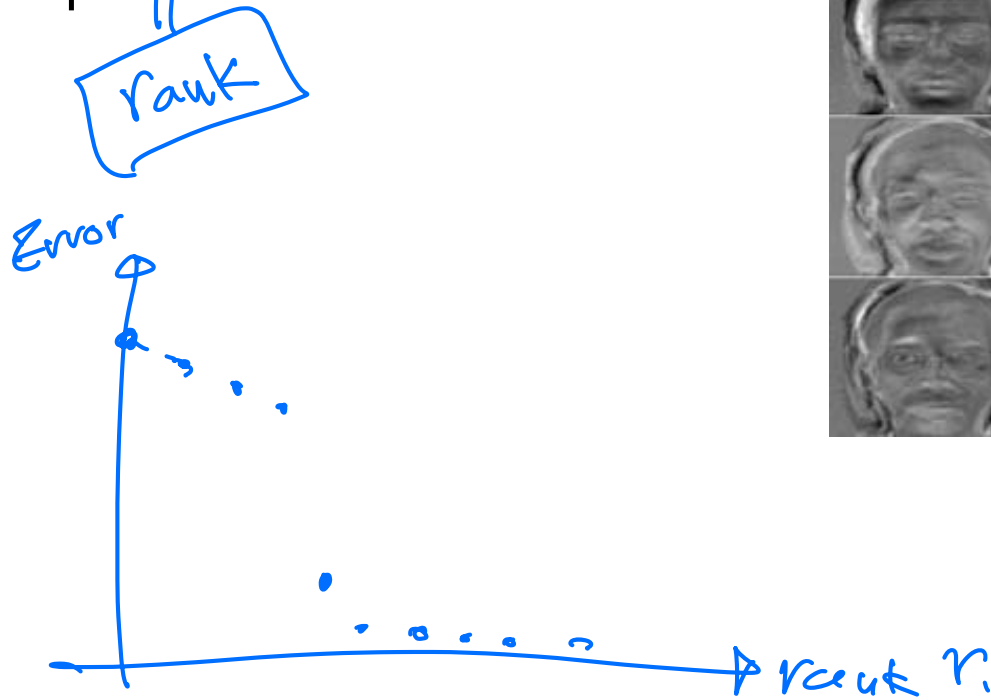
n images of faces

$x_1 =$

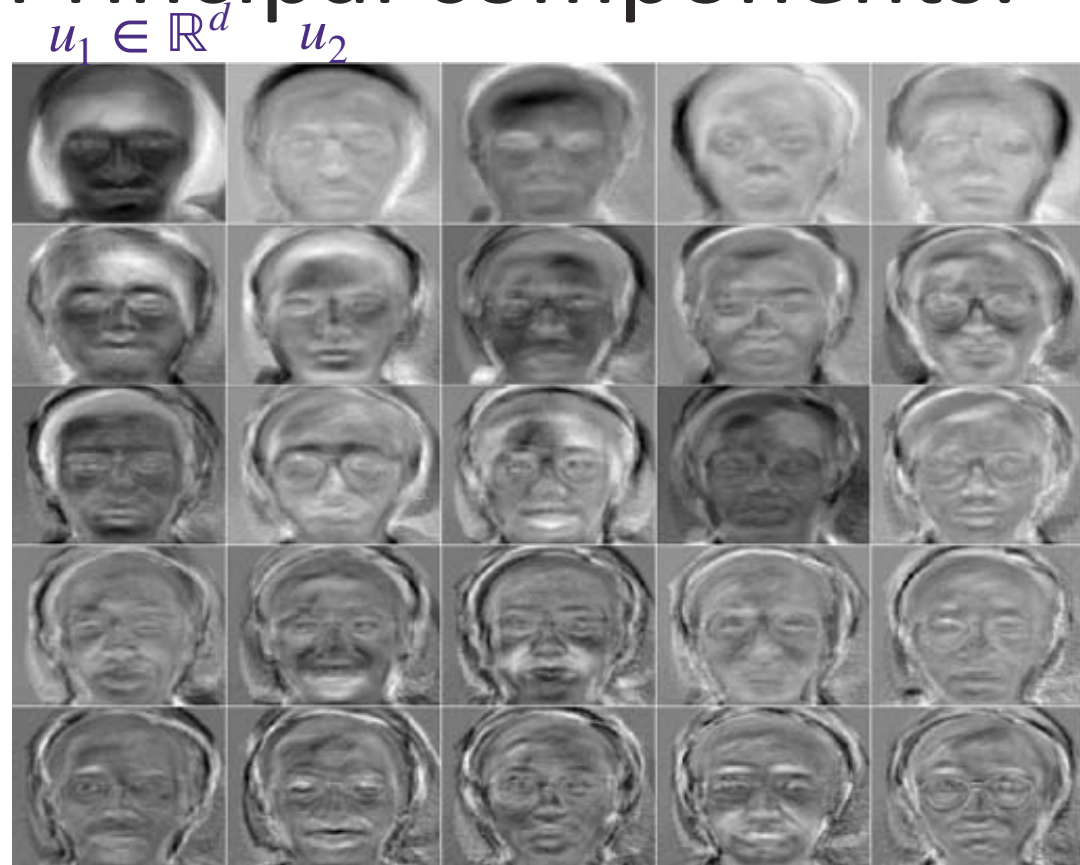


Principal component analysis finds a compact linear representation

- **patterns** that capture the distinct features of the samples is called **principal component** (to be formally defined later)
- we use $r = 25$ principal components



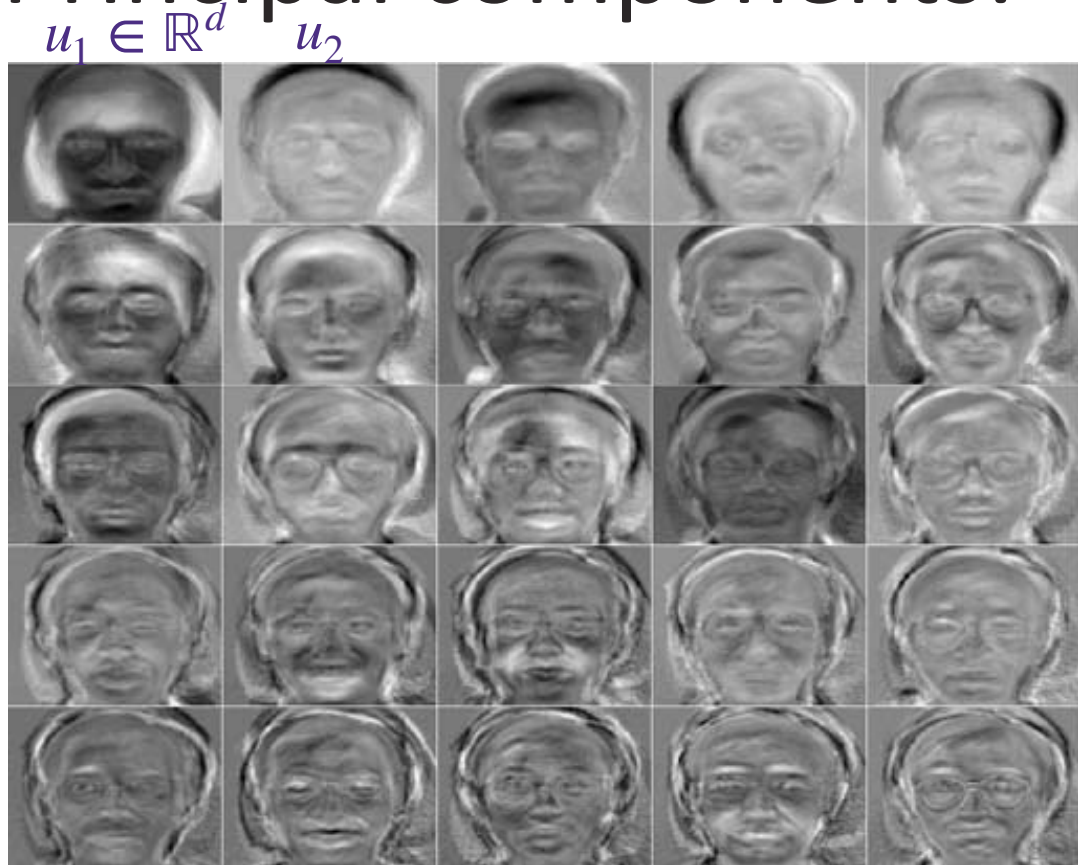
Principal components:



Principal component analysis finds a compact linear representation

- **patterns** that capture the distinct features of the samples is called **principal component** (to be formally defined later)
- we use $r = 25$ principal components
- we can represent each sample as a **weighted linear combination** of the principal components, and just store the weights (as opposed to all pixel values)

Principal components:



$$\approx a[1]u_1 + a[2]u_2 + \dots + a[25]u_{25}$$

- Each image is now represented by $r = 25$ numbers $a = (a[1], \dots, a[25])$
- To store n images, it requires memory of only $d \times r + r \times n \ll d \times n$
 $1,000 \times 25 + 25 \times n \ll 1,000 \times n$

10 principal components give a pretty good reconstruction of a face

average face

$$\bar{x} + a[1]u_1$$

$$\bar{x} + a[1]u_1 + a[2]u_2$$

↓ \bar{x}

↓ $r=1$

↓ $r=2$

$r=3$

$r=4$



$r=7$

$r=8$

$r=9$

$r=10$

↑
Ground truths real face

Pre and Post processing

- Notice how we started with the average face $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$
- PCA is applied to pre-processed data $\{x_i - \bar{x}\}_{i=1}^n$
- For simplicity, we will assume that x_i 's are centered such that $\frac{1}{n} \sum_{i=1}^n x_i = 0$
- otherwise, without loss of generality, everything we do can be applied to the re-centered version of the data, i.e. $\{x_i - \bar{x}\}_{i=1}^n$, with $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$

How do we define the principal components?

- **Dimensionality reduction** (for some $r \ll d$):
we would like to have a set of orthogonal directions $u_1, \dots, u_r \in \mathbb{R}^d$, with $\|u_j\|_2 = 1$ for all j to uniquely define principal components when we can, such that each data can be represented as linear combination of those direction vectors, i.e.

$$x_i \approx p_i = a_i[1]u_1 + \dots + a_i[r]u_r$$



d=32x32



$$x_i = \begin{bmatrix} x_i[1] \\ \vdots \\ x_i[d] \end{bmatrix} \xrightarrow{\text{Dimensionality Reduction}} a_i = \begin{bmatrix} a_i[1] \\ \vdots \\ a_i[r] \end{bmatrix}$$

- **Key question:** Which choice of the principal components, $\{u_1, \dots, u_r\}$, are better?
- But first, how do we find a_i for a data point x_i given $\{u_1, \dots, u_r\}$?

How do we find the a_i 's?

Given $x_i, \{u_1, \dots, u_r\}$

- **Dimensionality reduction** (for some $r \ll d$):

we would like to have a set of orthogonal directions $u_1, \dots, u_r \in \mathbb{R}^d$, with $\|u_j\|_2 = 1$ for all j , such that each data can be represented as linear combination of those direction vectors, i.e.

$$x_i \approx p_i = a_i[1]u_1 + \dots + a_i[r]u_r$$

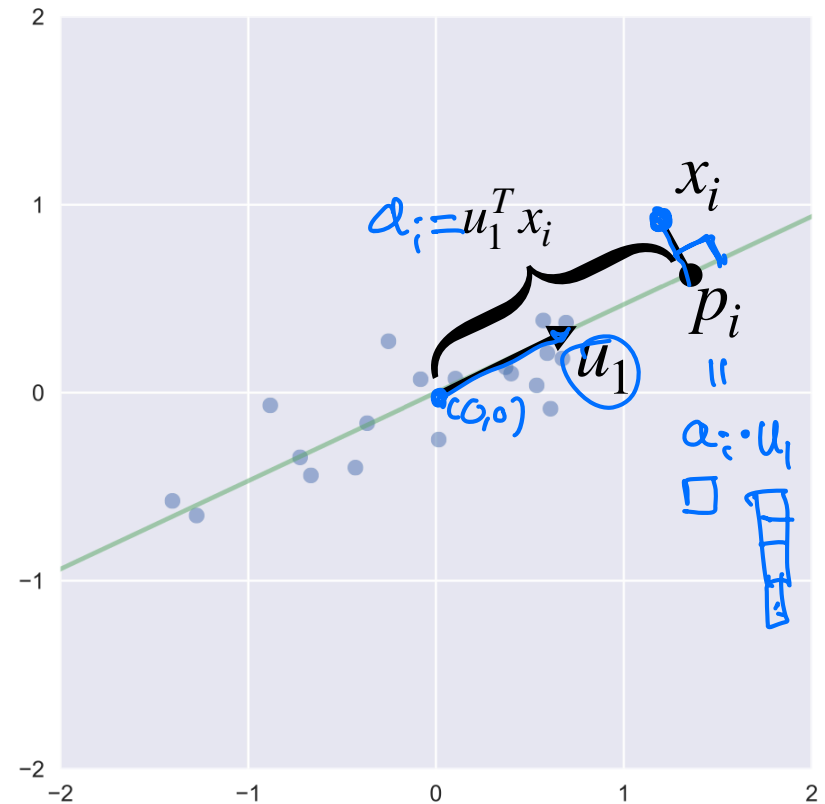
- **given a set of u_1, \dots, u_r with**

$$\|u_j\| = 1 \text{ for all } j \in [r] \text{ and } u_{j_1}^T u_{j_2} = 0,$$

the best representation p_i of x_i is the projection of the point onto the subspace spanned by u_j 's, i.e.

$$a_i[j] = u_j^T x_i$$

$$x_i \approx p_i = \sum_{j=1}^r \underbrace{(u_j^T x_i)}_{a_i[j]} u_j$$



How do we find the a_i 's?

$$a_i = U^T x_i$$

- **Dimensionality reduction** (for some $r \ll d$):

we would like to have a set of orthogonal directions $u_1, \dots, u_r \in \mathbb{R}^d$, with $\|u_j\|_2 = 1$ for all j , such that each data can be represented as linear combination of those direction vectors, i.e.

$$x_i \approx p_i = a_i[1]u_1 + \dots + a_i[r]u_r$$

- **given a set of u_1, \dots, u_r with $\|u_j\| = 1$ for all $j \in [r]$ and $u_{j_1}^T u_{j_2} = 0$,**

the best representation p_i of x_i is the projection of the point onto the subspace spanned by u_j 's, i.e.

$$a_i[j] = u_j^T x_i$$

$$p_i = \sum_{j=1}^r \underbrace{(u_j^T x_i)}_{a_i[j]} u_j$$

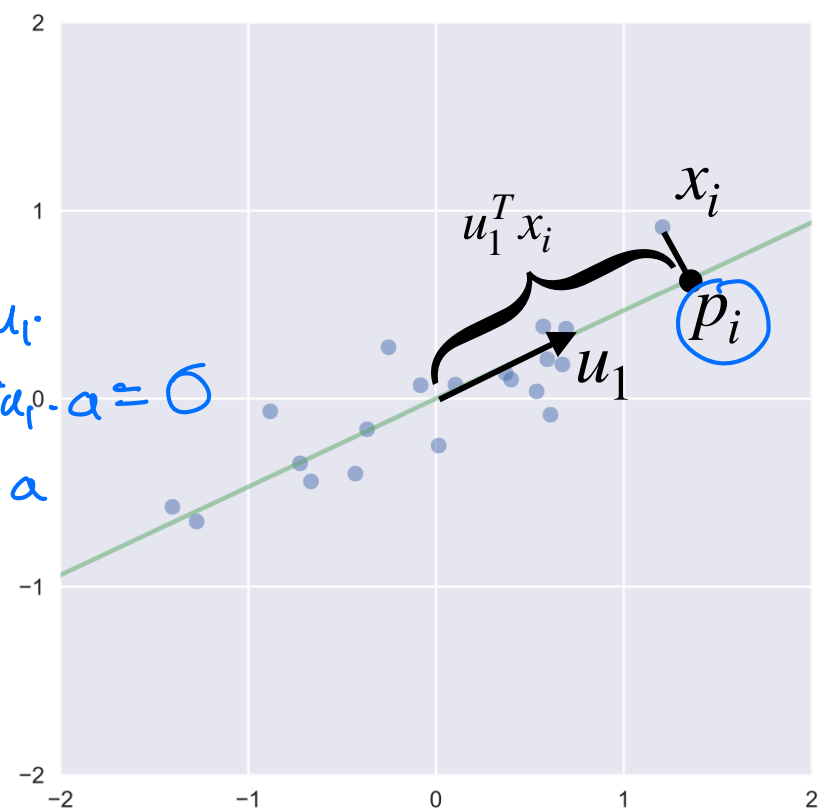
$$\begin{aligned} \nabla f(a) &= -2(x_i - a u_1)^T u_1 \\ &= -2x_i^T u_1 + 2u_1^T a \cdot 0 \cdot a = 0 \end{aligned}$$

$$x_i^T u_1 = u_1^T u_1 \cdot a = 1 \cdot a$$

reconstruction error

- proof for a single u_1 : $\min_a \|x_i - a u_1\|^2 = f(a)$

- Next question: how do we find the u_j 's?



Principal components, $\{u_1, \dots, u_r\}$, on a subspace that minimizes the reconstruction error

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

recon. error?

subject to $\|u_j\|_2 = 1$ for all j and $u_j^T u_\ell = 0$ for all $j \neq \ell$

min $\frac{1}{n} \sum_{i=1}^n \|x_i - \sum_{j=1}^r a_j [u_j]^T \cdot u_j\|_2^2$

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

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

$$P_i = \sum_{j=1}^r (u_j^T x_i) u_j = \left\{ \sum_{j=1}^r (u_j u_j^T) \right\} x_i = \mathbf{U} \mathbf{U}^T \cdot x_i$$

$\begin{matrix} d & r & d \\ \uparrow & \uparrow & \uparrow \\ \mathbf{U} & & \end{matrix}$

 $u_1 \ u_2 \ \dots \ u_r$

Principal components, $\{u_1, \dots, u_r\}$, on a subspace that minimizes the reconstruction error

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

Solution
 \Leftarrow PCA

subject to $\|u_j\|_2 = 1$ for all j and $u_j^T u_\ell = 0$ for all $j \neq \ell$

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

PCA =

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

\leftarrow objective

subject to $U^T U = I_{r \times r}$ \leftarrow condition

- Small rank r gives efficiency and large r gives less reconstruction error
- Q. How do we solve this optimization?

Minimizing reconstruction error to find principal components

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

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

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

$U = [u_1 \dots u_r]$

$$= \sum_j (u_j^T x_i)^2$$

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

$\langle x_i - U U^T x_i, x_i - U U^T x_i \rangle$

$$\text{subject to } \underline{U^T U = I_{r \times r}}$$

$$\begin{bmatrix} u_1^T \\ \vdots \\ u_r^T \end{bmatrix}^T \begin{bmatrix} u_1 & \dots & u_r \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}$$

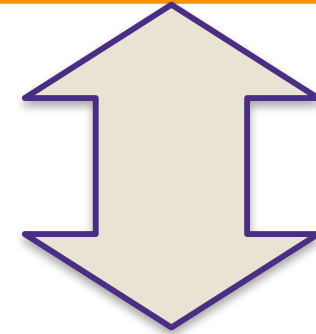
Minimizing reconstruction error to find principal components

$$\begin{aligned} & \frac{1}{n} \sum_{i=1}^n \|x_i - UU^T x_i\|_2^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left\{ \|x_i\|_2^2 - 2x_i^T UU^T x_i + x_i^T U \underbrace{U^T U}_{=I} U^T x_i \right\} \\ &= \underbrace{\frac{1}{n} \sum_{i=1}^n \|x_i\|_2^2}_{\text{does not depend on } U} - \frac{1}{n} \sum_{i=1}^n x_i^T UU^T x_i \\ &= C - \sum_{j=1}^r \underbrace{\frac{1}{n} \sum_{i=1}^n (u_j^T x_i)^2}_{\text{Variance in direction } u_j} \end{aligned}$$

Recall we assumed x_i 's are centered, i.e., zero-mean

Minimize Reconstruction Error

$$\begin{aligned} & \underset{U}{\text{minimize}} \quad \frac{1}{n} \sum_{i=1}^n \|x_i - UU^T x_i\|_2^2 \\ & \text{subject to} \quad U^T U = \mathbf{I}_{r \times r} \end{aligned}$$



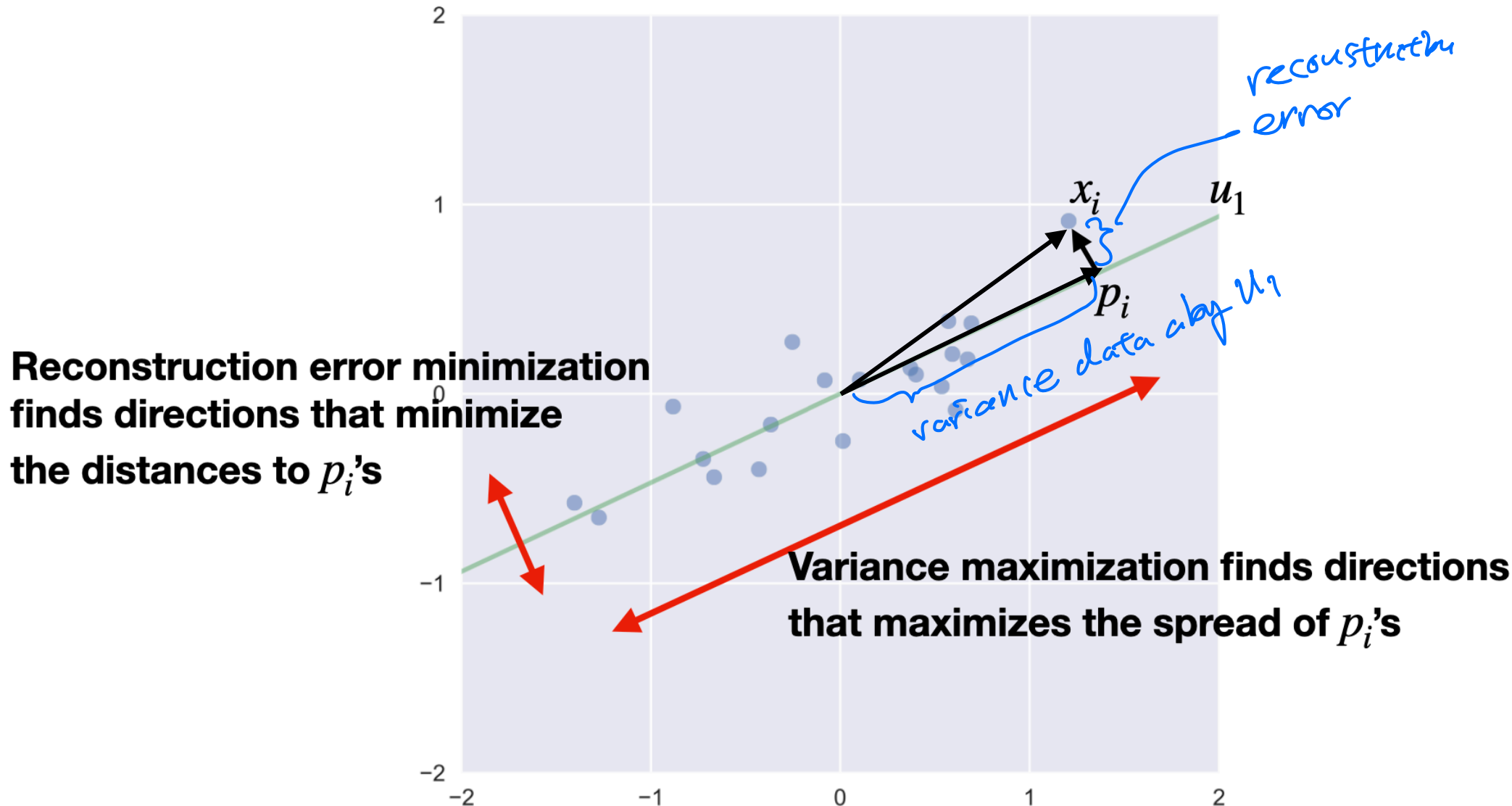
Maximizing Variance captured in principal directions

$$\begin{aligned} & \underset{U}{\text{maximize}} \quad \sum_{j=1}^r \left\{ \frac{1}{n} \sum_{i=1}^n (u_j^T x_i)^2 \right\} \\ & \text{subject to} \quad U^T U = \mathbf{I}_{r \times r} \end{aligned}$$

variance of data in u_j direction

Variance maximization vs. reconstruction error minimization

- both give the same principal components as optimal solution, because by **Pythagorean Theorem**: $\text{Error}^2 + \text{Variance of data} = \|x_i\|_2^2$



Maximizing variance to find principal components

$$\begin{aligned} & \underset{U}{\text{maximize}} \quad \sum_{j=1}^r \frac{1}{n} \sum_{i=1}^n (u_j^T x_i)^2 \\ & \text{subject to} \quad \mathbf{U}^T \mathbf{U} = \mathbf{I}_{r \times r} \end{aligned}$$

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

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

$$\begin{aligned} & \Rightarrow \frac{1}{n} \sum_{i=1}^n u^T x_i x_i^T u \\ & = u^T \left(\frac{1}{n} \sum_{i=1}^n x_i x_i^T \right) u \end{aligned}$$

Correlation data

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

How do you find u ?

Maximizing variance to find principal components

Claim: Solution of (a)
is the top eigenvector
of the matrix C .

$$\text{maximize}_u u^T C u \quad (a)$$

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

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

$$\text{maximize}_u u^T C u \quad (b)$$

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

- Why?

Maximizing variance to find principal components

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

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


Convex Maximization
= non-convex
(a)

← non-convex


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

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

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

(b) 
← Convex.

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

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

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

- 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

$$\text{maximize}_{u \in \mathbb{R}^d} \underbrace{u^T \mathbf{C} u - \lambda \|u\|_2^2}_{F_\lambda(u)} \quad (c)$$

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

Solving the unconstrained optimization

(λ, u) eigenpair maximize $_{u \in \mathbb{R}^d} \underbrace{u^T \mathbf{C} u - \lambda \|u\|_2^2}_{F_\lambda(u)} \quad (\text{C})$

iff $\lambda \cdot \mathbf{C} = \mathbf{C} \cdot u \iff \lambda \mathbf{C} - \mathbf{C}u = 0$

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

$$\nabla F_\lambda(u) = 2\mathbf{C}u - 2\lambda u = 0 \quad (\text{***})$$

- the candidate solution satisfies: $\mathbf{C}u = \lambda u$,
i.e. an **eigenvector** of \mathbf{C}
- All eigenvalue-eigenvector pairs $\{(\lambda^{(i)}, u^{(i)})\}_{i=1}^d$ of \mathbf{C} satisfy the above equation (and we will only consider normalized $\|u^{(i)}\| = 1$)
- We know that $\lambda^{(i)} \geq 0$, because \mathbf{C} is positive semidefinite
- We assume these are ordered such that $\lambda^{(1)} \geq \lambda^{(2)} \geq \dots \geq \lambda^{(d)}$
- We only need to consider the eigenvectors in maximizing the original objective: $u^T \mathbf{C} u$, which is maximized by the first eigenvector $u^{(1)}$

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 also use the Singular Value Decomposition (SVD) to find such an eigenvector
- note that if 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

- **algorithm:**

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

- compute covariance matrix

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

- let (u_1, \dots, u_r) be the set of (normalized) eigenvectors with corresponding to the largest r eigenvalues of \mathbf{C}

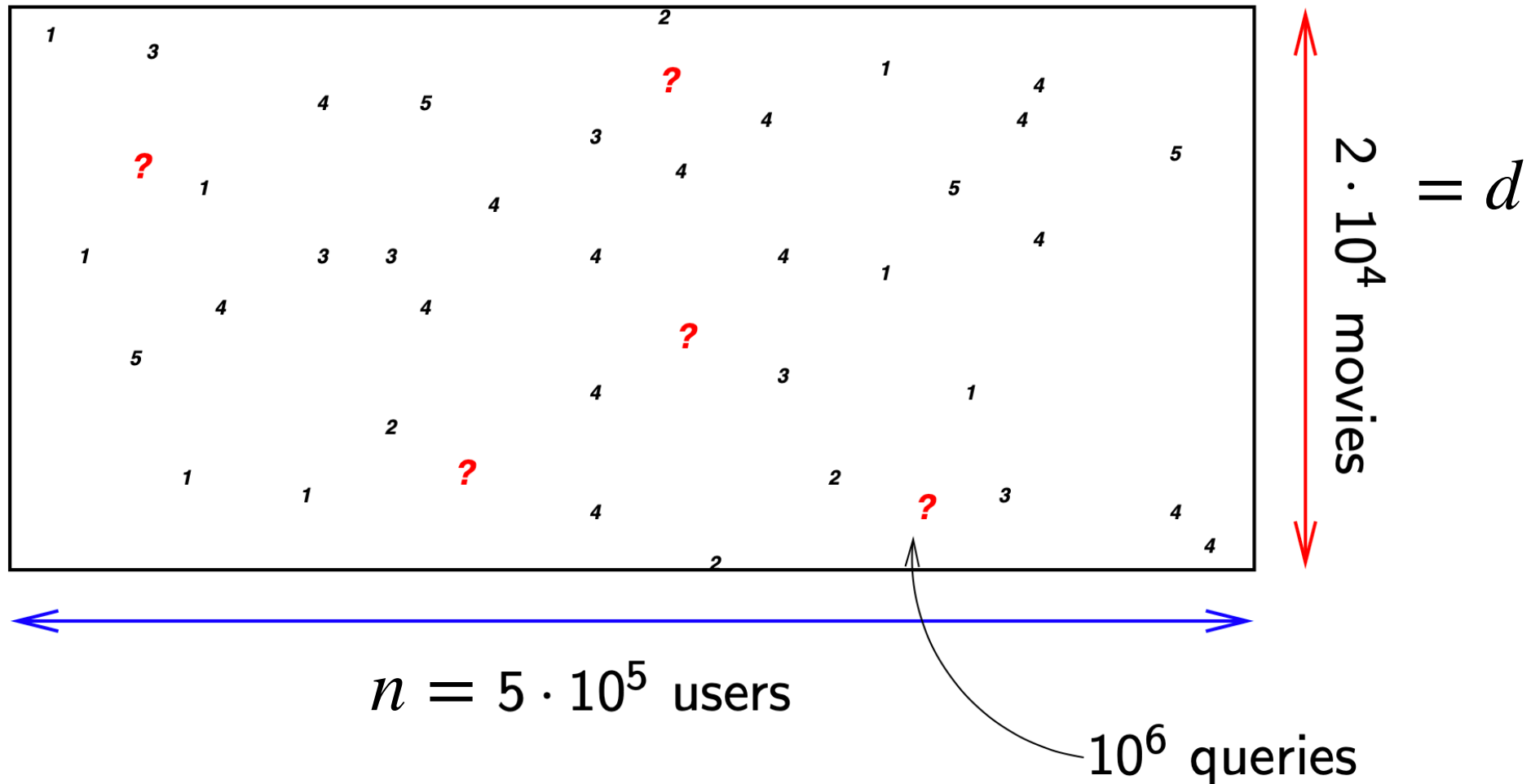
- return $\mathbf{U} = [u_1 \quad u_2 \quad \cdots \quad u_r]$

- further the data points can be represented compactly via

$a_i = \mathbf{U}^T(x_i - \bar{x}) \in \mathbb{R}^r$ and
each data point approximated by

$$p_i = \bar{x} + \mathbf{U} a_i$$

Matrix completion for recommendation systems



- 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

- 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

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

- such that that ratings $x_i \in \mathbb{R}^d$ of user i , can be represented as

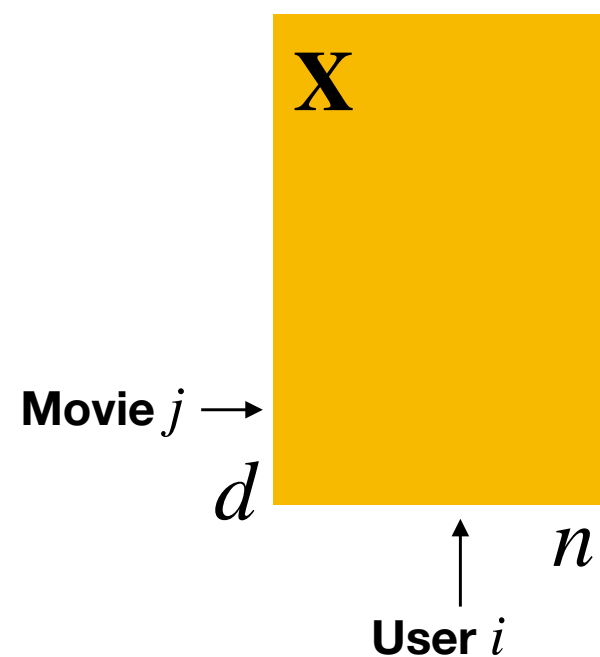
$$\begin{aligned} x_i &= a_i[1]u_1 + \cdots a_i[r]u_r \\ &= \mathbf{U}a_i \end{aligned}$$

for some lower-dimensional $a_i \in \mathbb{R}^r$ for i -th user and some $r \ll d$

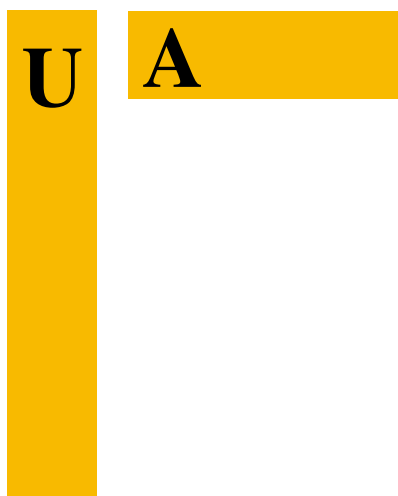
- for example, $u_1 \in \mathbb{R}^d$ means how horror movie fans like each of the d movies,
- and $a_i[1]$ means how much user i is fan of horror movies

Matrix completion

- let $\mathbf{X} = [x_1 \ x_2 \ \cdots \ x_n] \in \mathbb{R}^{d \times n}$ 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 $\mathbf{A} = [a_1 \ a_2 \ \cdots \ a_n] \in \mathbb{R}^{r \times n}$ that approximates \mathbf{X}



\approx



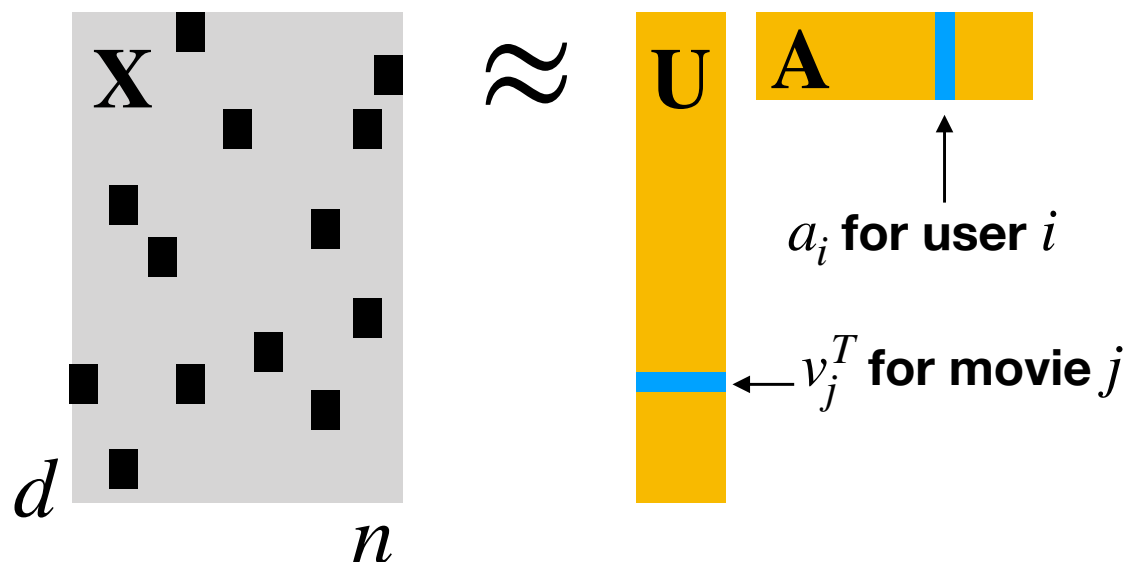
- if we **observe all entries** of \mathbf{X} , then we can solve

$$\text{minimize}_{\mathbf{U}, \mathbf{A}} \sum_{i=1}^n \|x_i - \mathbf{U}a_i\|_2^2$$

which can be solved using PCA (i.e. 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_ℓ on movie j_ℓ



- 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

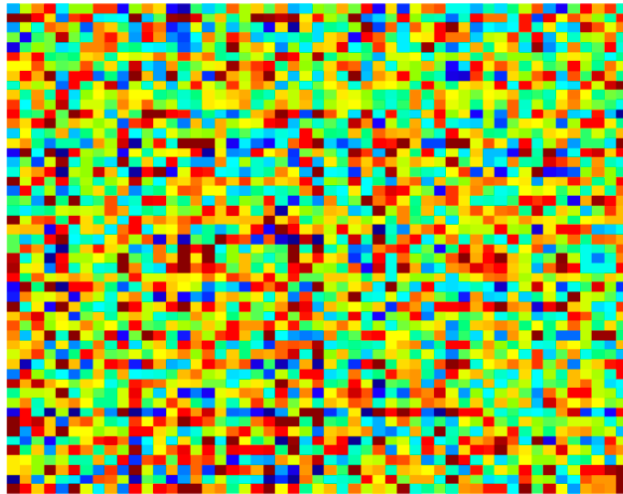
- a natural approach to fit v_j 's and a_i 's to given training data is to solve

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

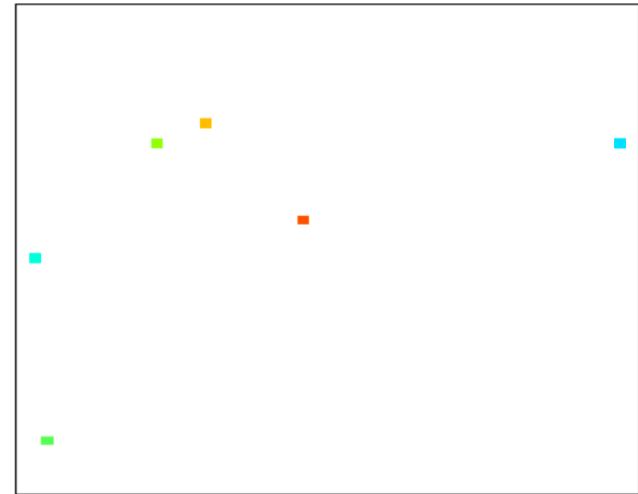
- this can be solved, for example via gradient descent or alternating minimization
- this can be quite accurate, with small number of samples

Example: 2000×2000 rank-8 random matrix

low-rank matrix \mathbf{X}

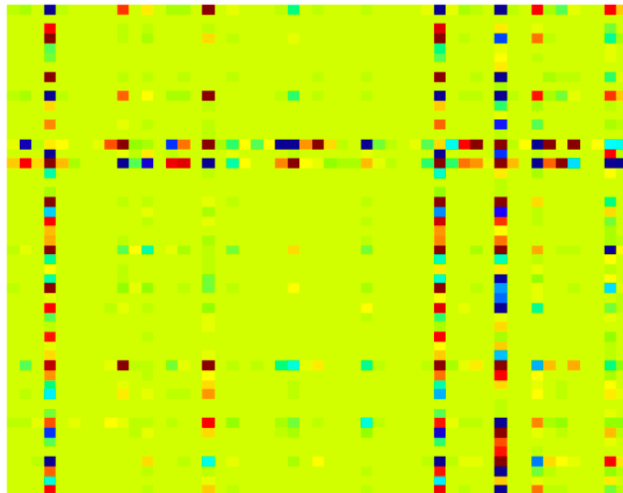


sampled matrix

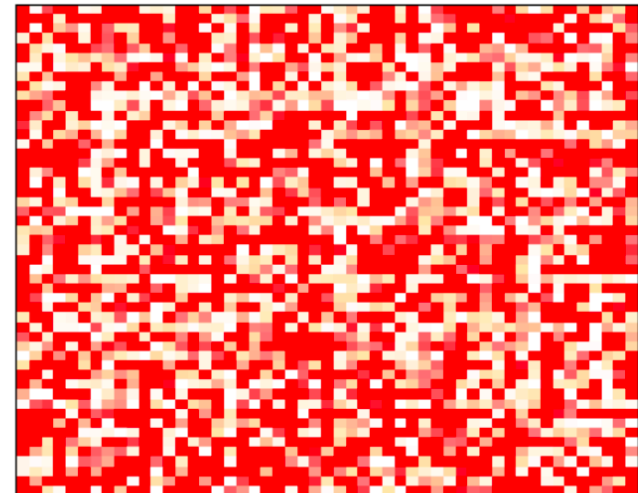


For illustration,
we zoom in to a
50x50 submatrix

Gradient descent output \mathbf{UA}



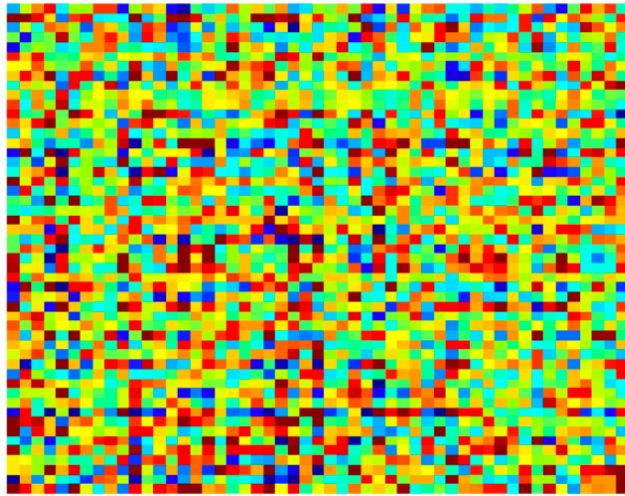
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



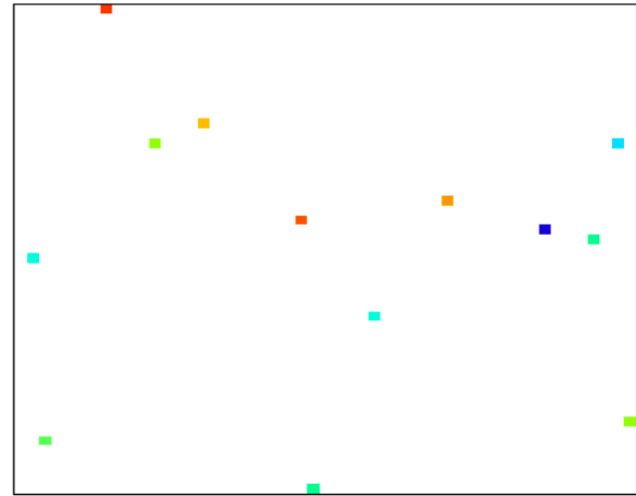
0.25% sampled

Example: 2000×2000 rank-8 random matrix

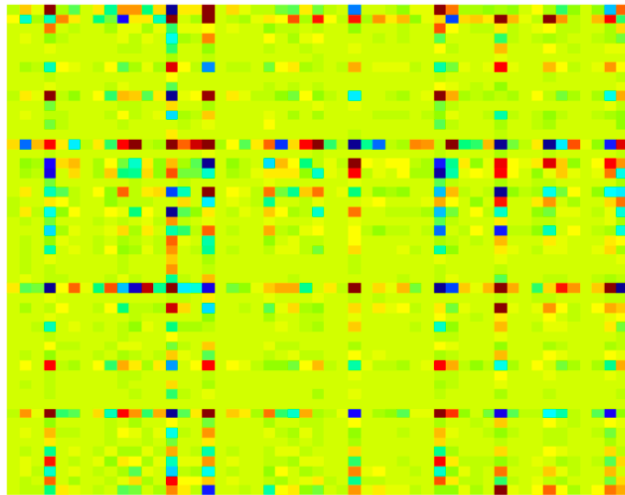
low-rank matrix \mathbf{X}



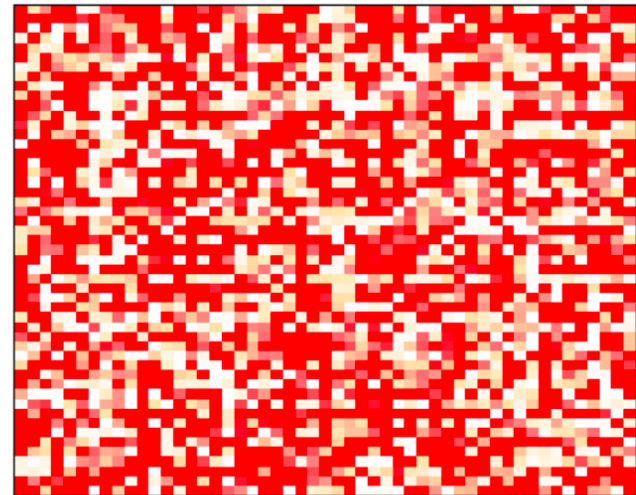
sampled matrix



Gradient descent output \mathbf{UA}



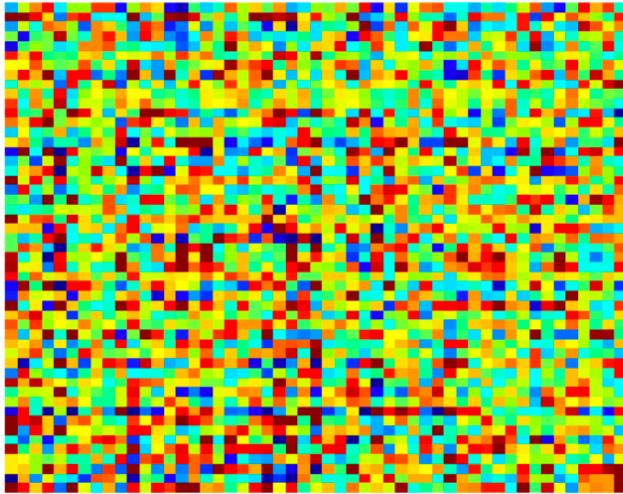
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



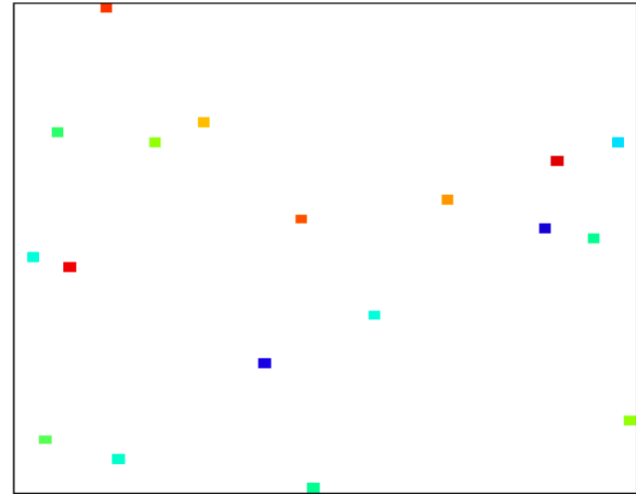
0.50% sampled

Example: 2000×2000 rank-8 random matrix

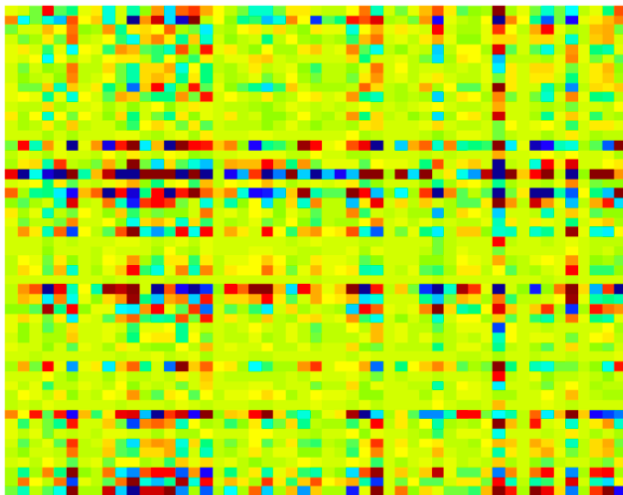
low-rank matrix \mathbf{X}



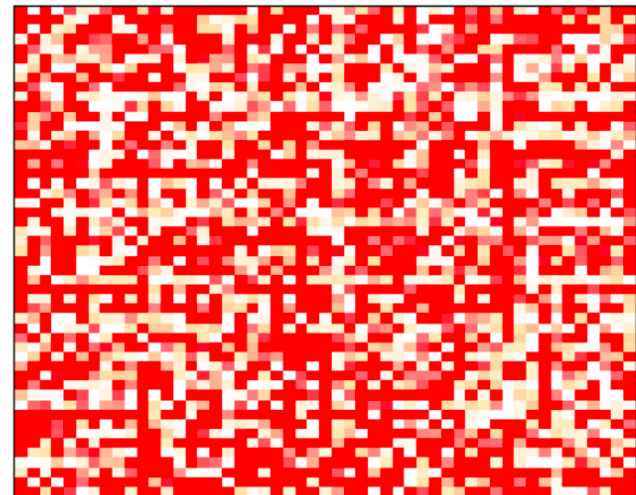
sampled matrix



Gradient descent output \mathbf{UA}



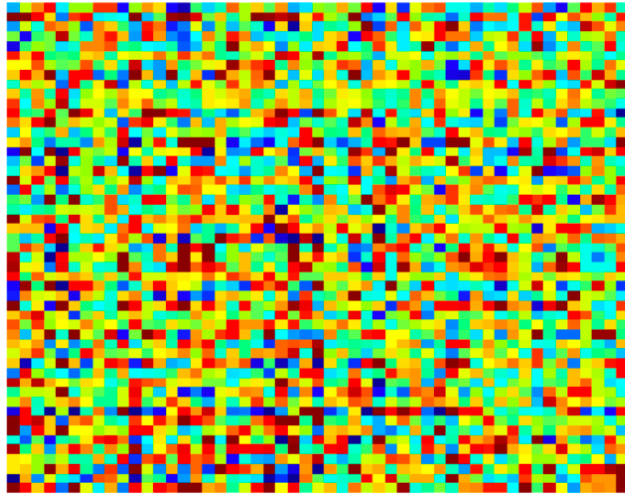
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



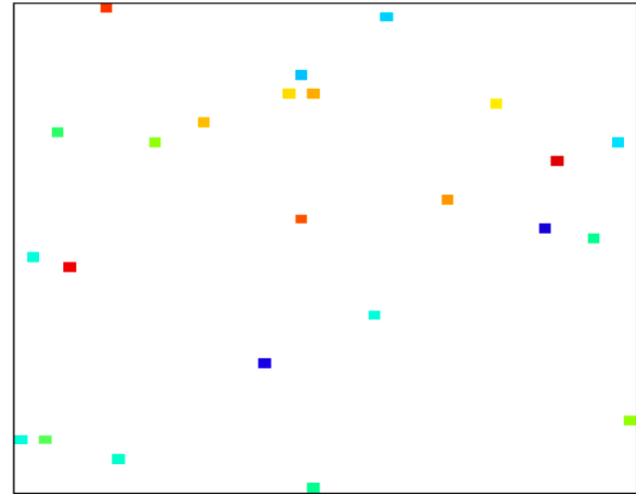
0.75% sampled

Example: 2000×2000 rank-8 random matrix

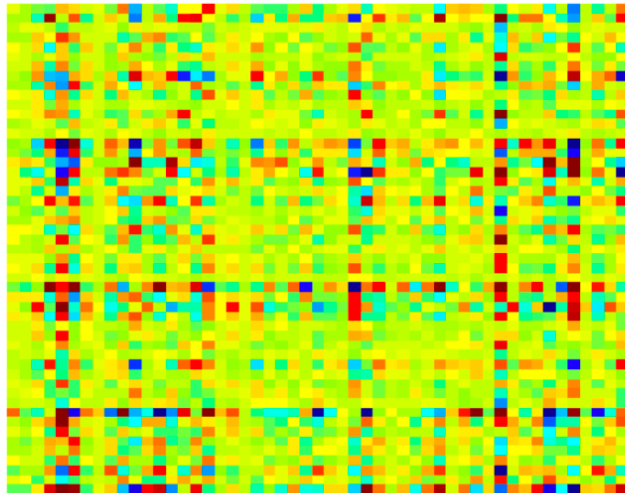
low-rank matrix \mathbf{X}



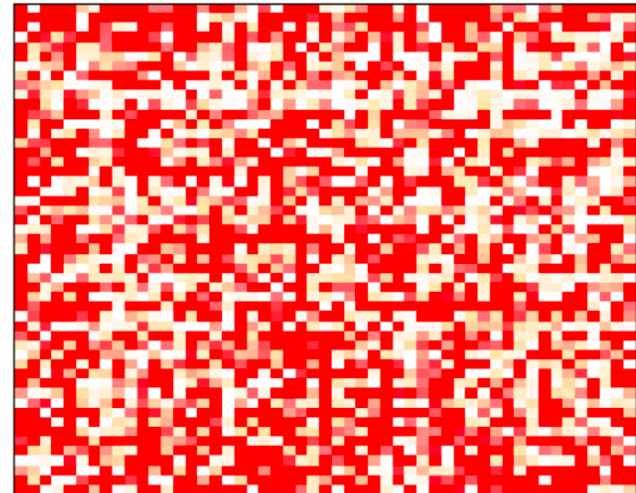
sampled matrix



Gradient descent output \mathbf{UA}



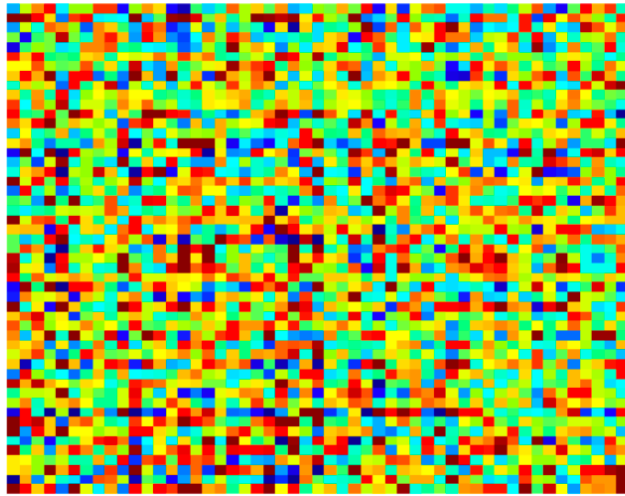
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



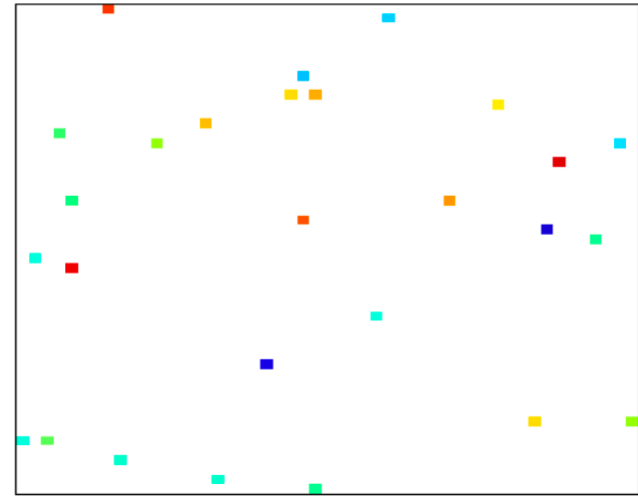
1.00% sampled

Example: 2000×2000 rank-8 random matrix

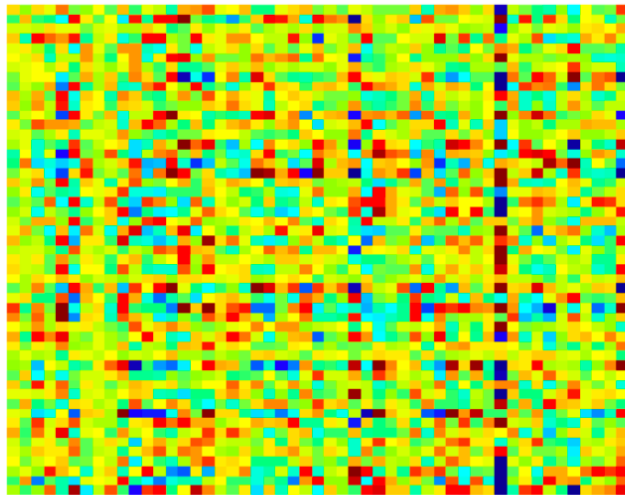
low-rank matrix \mathbf{X}



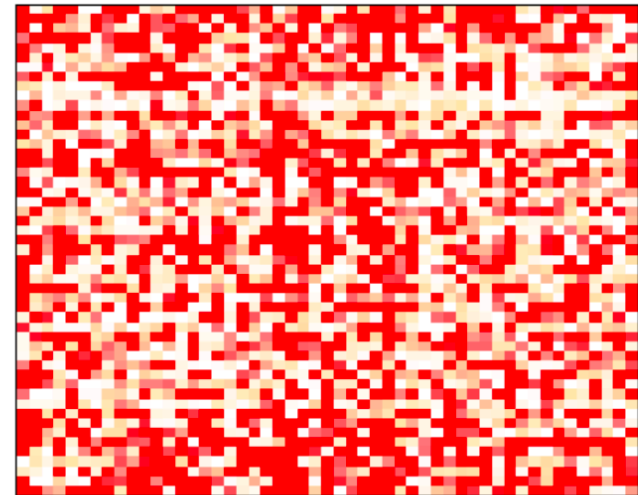
sampled matrix



Gradient descent output \mathbf{UA}



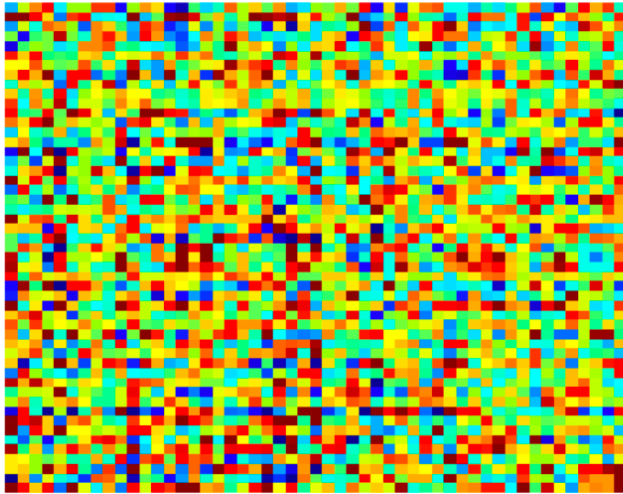
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



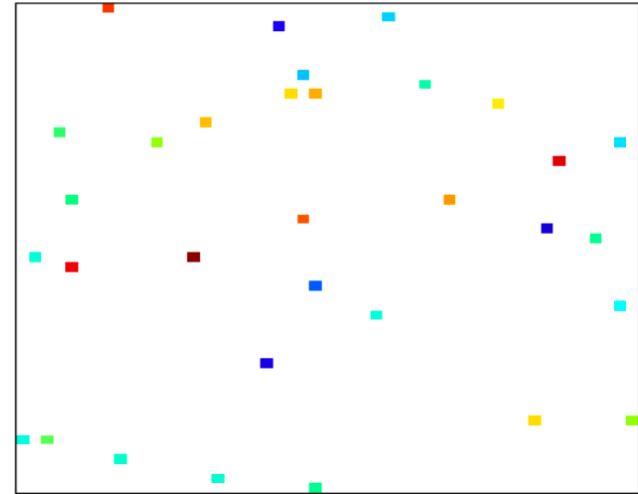
1.25% sampled

Example: 2000×2000 rank-8 random matrix

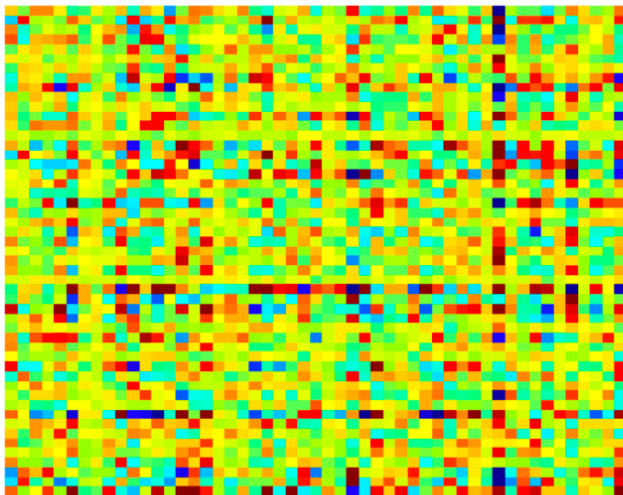
low-rank matrix \mathbf{X}



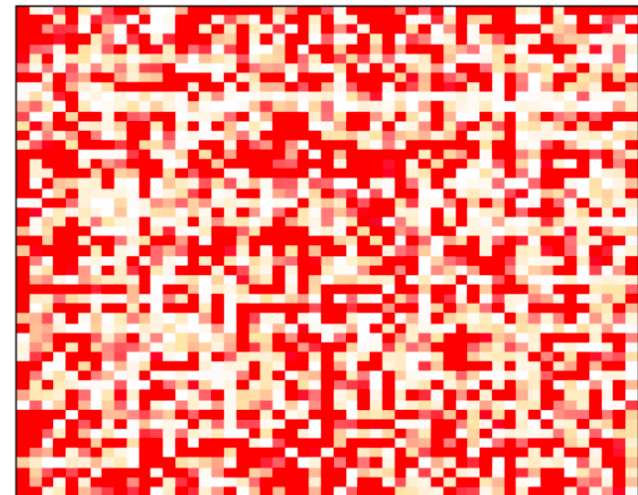
sampled matrix



Gradient descent output \mathbf{UA}



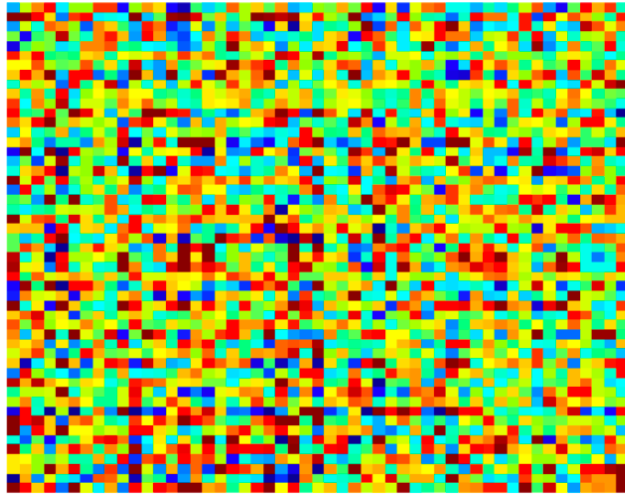
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



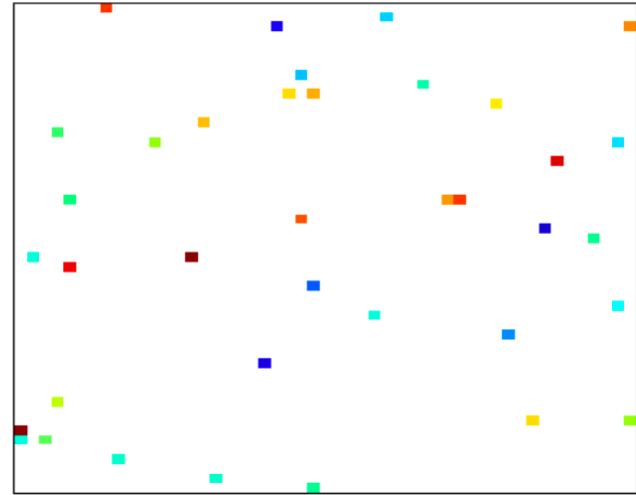
1.50% sampled

Example: 2000×2000 rank-8 random matrix

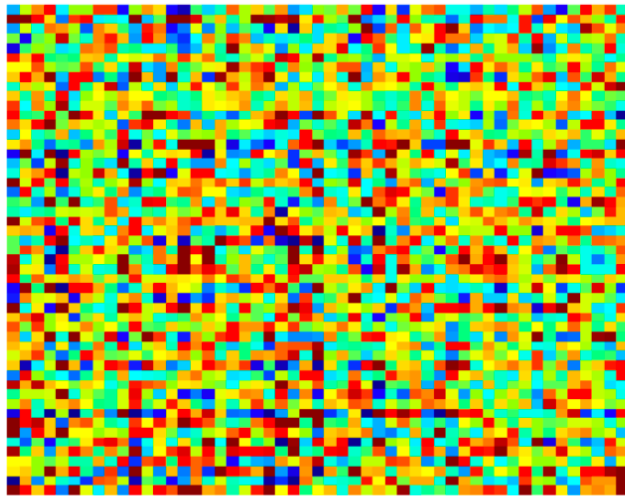
low-rank matrix \mathbf{X}



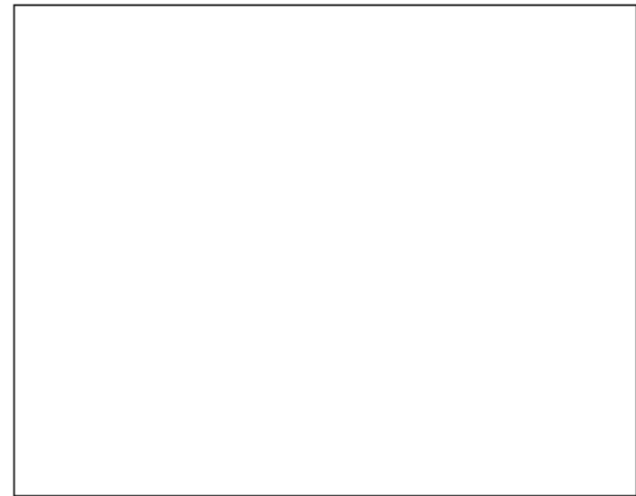
sampled matrix



Gradient descent output \mathbf{UA}



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



1.75% sampled

Questions?
