Principal Component Analysis

Sewoong Oh

CSE446
University of Washington

Dimensionality reduction

- it takes $n \times d$ memory to store data $\{x_i\}_{i=1}^n$ with $x_i \in \mathbb{R}^d$
- but many real data have repeated patterns
- can we represent each image compactly, but still preserve most of information?

d pixels

n images.

$\boxed{d \times n}$ real numbers

Principal components

- patterns that capture the distinct features of the samples is called principal component (to be formally defined later)
- 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:

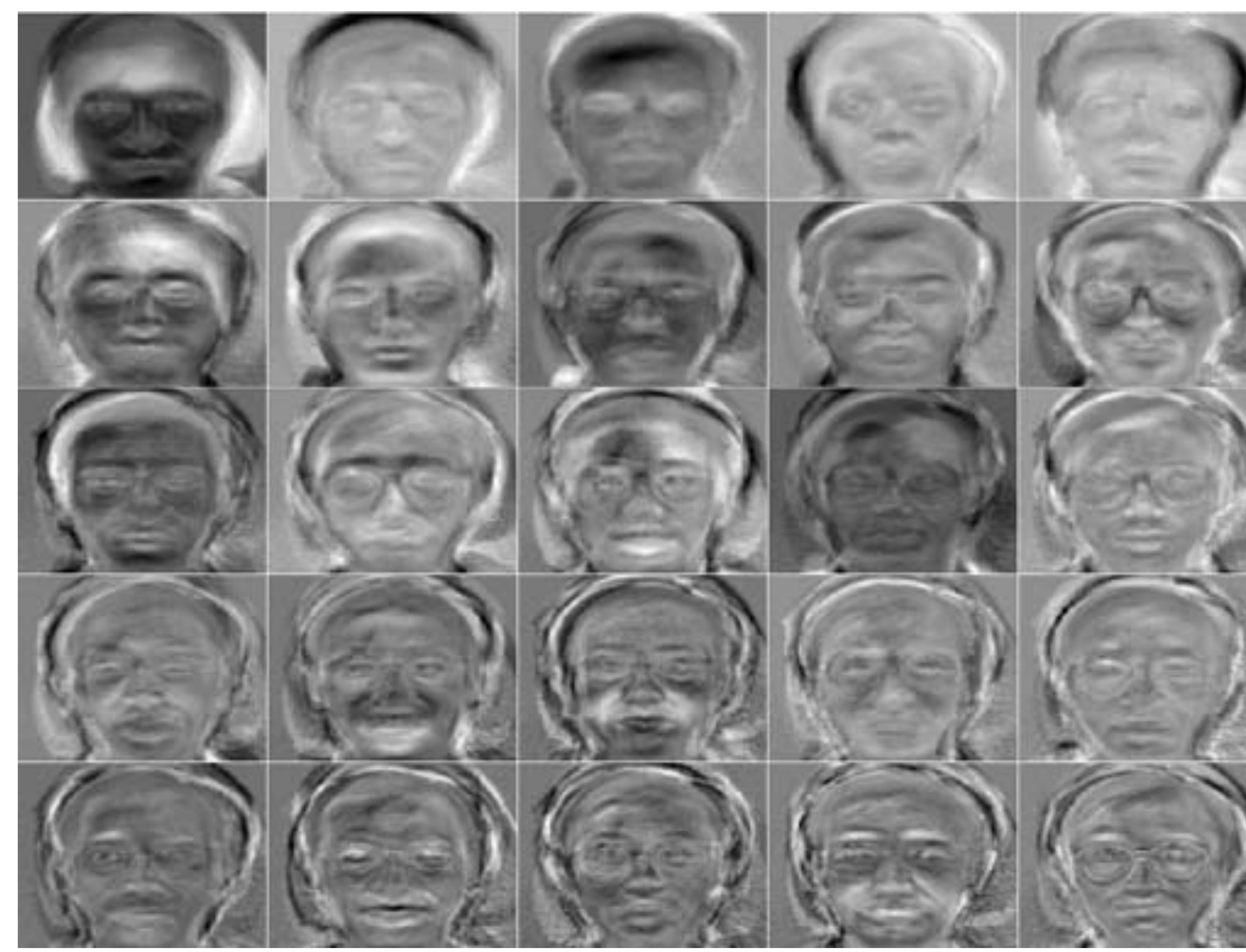
$$\approx \; a[1]u_1 + a[2]u_2 + \cdots + a[25]u_{25}$$

$d \times n \gg d \times r + n \times r$

$1{,}000 \gg 25 \ll 10{,}000$

$d \times r$

$r = 25$



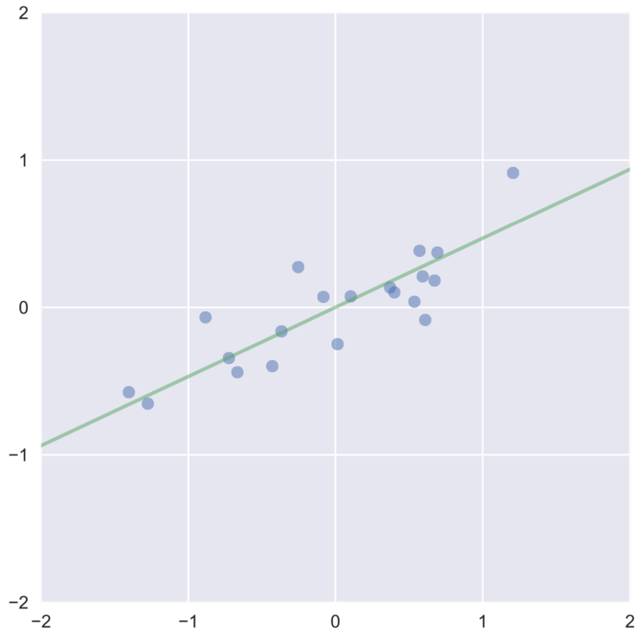
average face

real face

10 principal components give a pretty good reconstruction of the face

Principal Component Analysis (PCA)

Representing data compactly



PCA formulation 1: direction of greatest variance

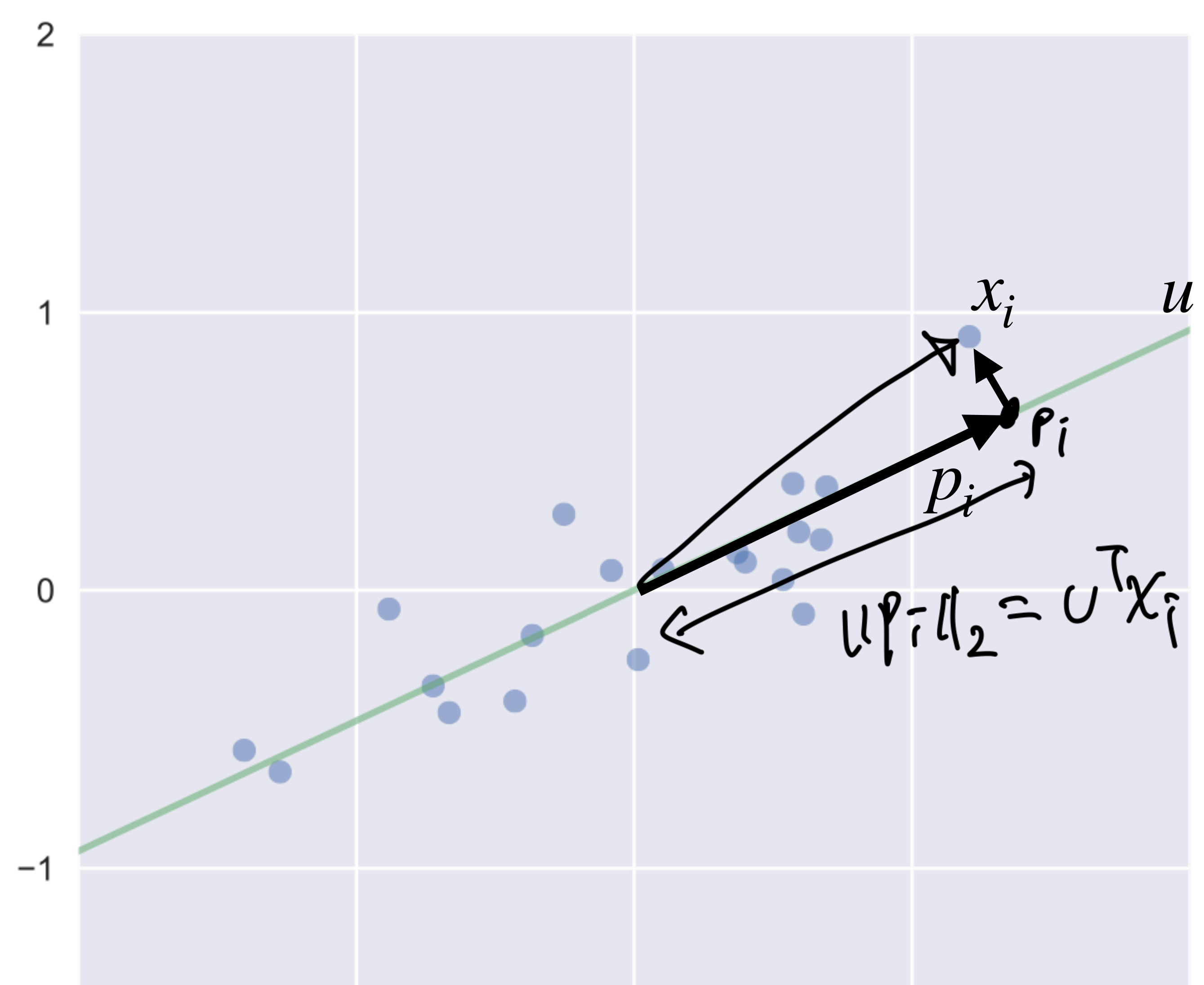
- given dataset $\{x_i\}_{i=1}^n$
- we will assume that the data is centered at the origin, such that $\frac{1}{n}\sum_{i=1}^n x_i = 0$
- otherwise, everything we do can be applied to the re-centered version of the data, i.e. $\{x_i - \bar{x}\}_{i=1}^n$, with $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$
- we want to find the **direction** $u \in \mathbb{R}^d$ **of greatest variance**,
and as we care about the direction, we will assume $\|u\|_2 = 1$
- we will justify why we care about greatest variance direction, later

PCA formulation 1: direction of greatest variance

- for a direction $u \in \mathbb{R}^d$ direction
  - $\boxed{p_i = \boxed{(u^T x_i)} u \in \mathbb{R}^d}$ is the projection of x_i onto u, i.e. the point on the direction of u that is closest to x_i
  - the length of the projection is $\|p_i\|_2 = u^T x_i$
  - mean of $\{p_i\}_{i=1}^n$ is zero, as $\sum_{i=1}^n p_i = \sum_{i=1}^n (u^T x_i) u = u^T \Big(\sum_{i=1}^n x_i \Big) u = 0$
  - similarly, mean of $\{\|p_i\|_2\}_{i=1}^n$ is also zero
  - so, variance is $\frac{1}{n} \sum_{i=1}^n \|p_i\|_2^2$
  - variance maximizing direction is

$$\arg\max_{u \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \left(\underbrace{u^T x_i}_{\|p_i\|_2} \right)^2$$

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



  - such variance maximizing directions are called the **principal components**
  - this is 1-dimensional PCA

The optimization problem in a matrix form

$$\arg\max_{u \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(u^T x_i \right)^2 = \sum_{i=1}^{n} u^T x_i x_i^T u = u^T \left(\sum_{i=1}^{n} x_i x_i^T \right) u$$

subject to $\|u\|_2^2 = 1$

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix}$$

- recall the data matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, and the optimization is

$$\arg \max_{u : \|u\|_2^2 = 1} u^T \mathbf{X}^T \mathbf{X} u$$

- assuming the data has zero mean, the **covariance matrix** of the data is defined as

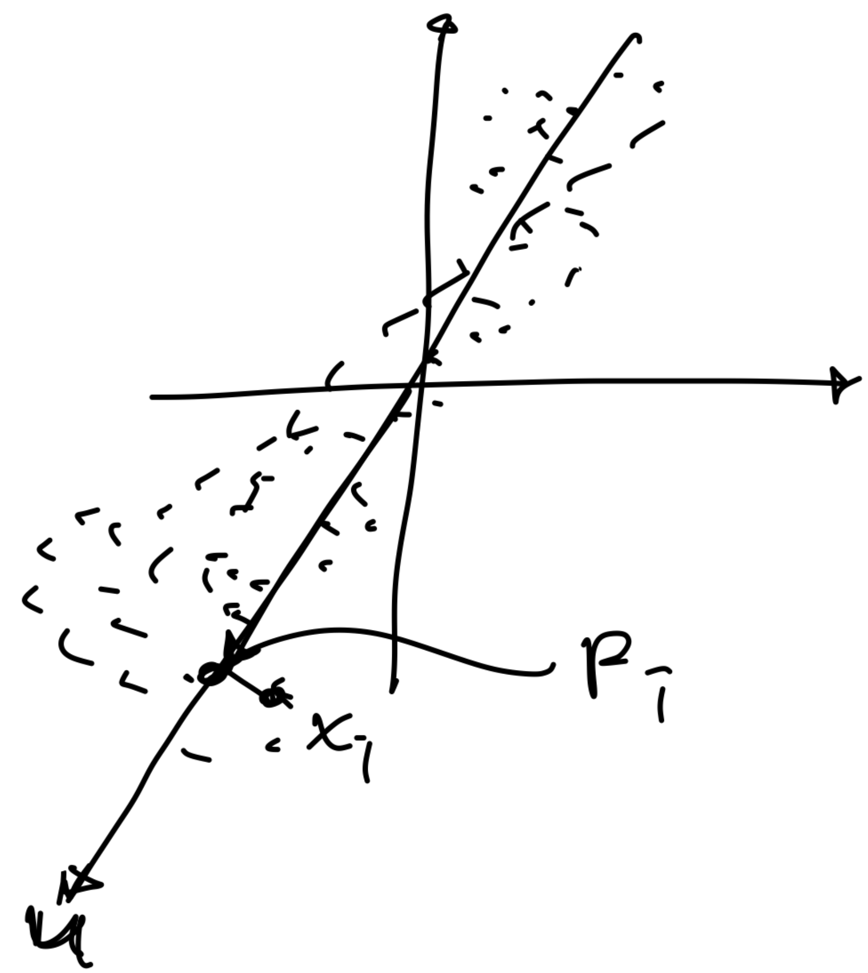
$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T = \frac{1}{n} \mathbf{X}^T \mathbf{X}$$

- which gives

$$\arg \max_{u : \|u\|_2^2 = 1} u^T \mathbf{C} u$$

Given data points $\{x_i\}_{i=1}^n$

Principal component $u \in \mathbb{R}^d$, $\|u\|_2^2 = 1$ is the direction of maximum variance.



$$p_i = \mathrm{Proj}_u(x_i)$$

$$= \underset{\mathbb{R}}{\underbrace{(u^T x_i)}} \cdot \underset{\mathbb{R}^d}{\underbrace{u}}$$

$$\|p_i\|_2 = u^T x_i$$

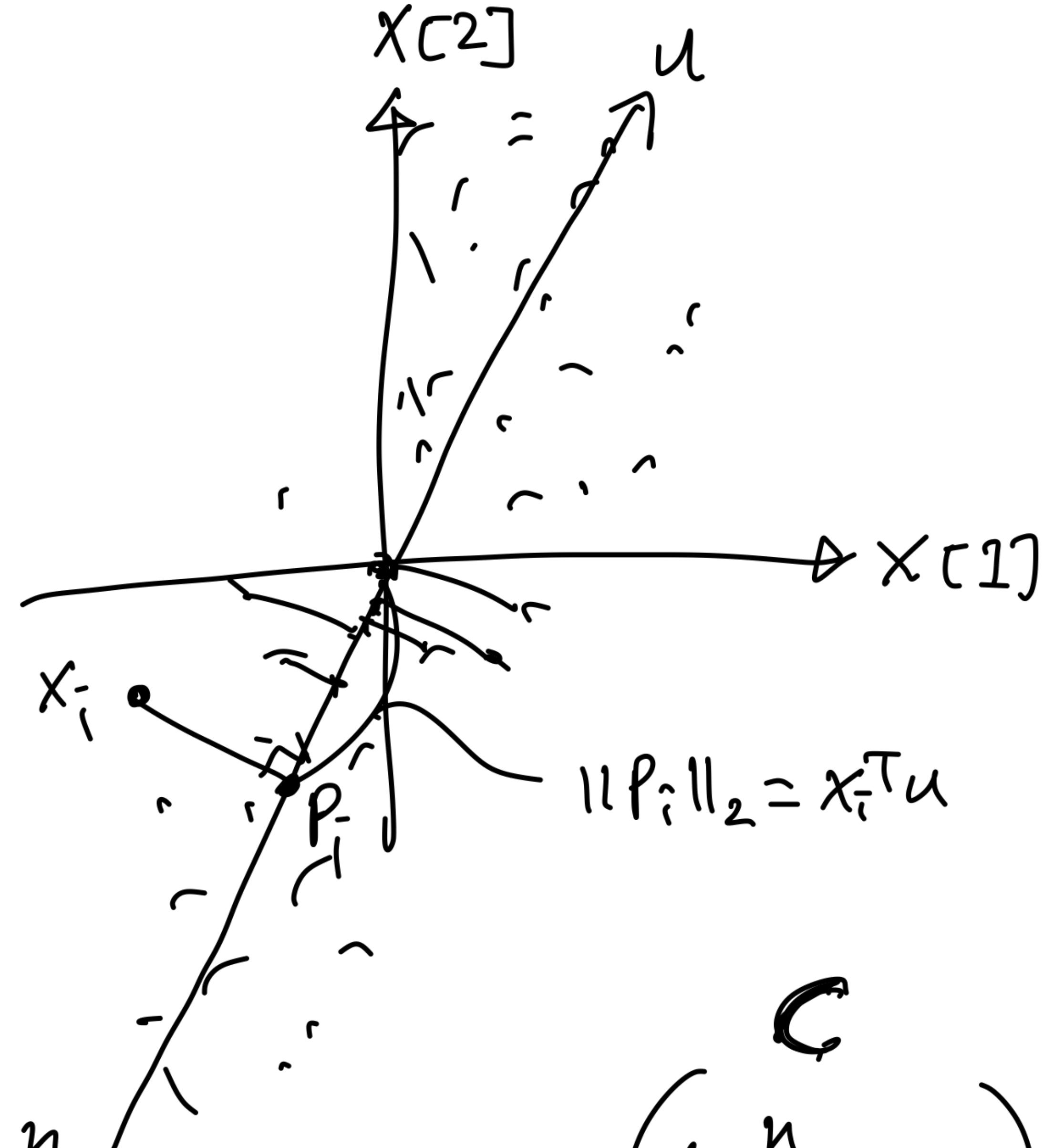
Variance $\frac{1}{n} \sum_{i=1}^n \|p_i\|_2^2 = \frac{1}{n} \sum_{i=1}^n (u^T x_i)^2$

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

Given dataset $\{x_i\}_{i=1}^{n}$

Principal Component $u \in \mathbb{R}^d$, $\|u\|_2^2 = 1$

maximum variance.



$$p_i = \mathrm{Proj}_u(x_i)$$

$$= \underbrace{(x_i^T u)}_{\mathbb{R}} \cdot \underbrace{u}_{\mathbb{R}^d}$$

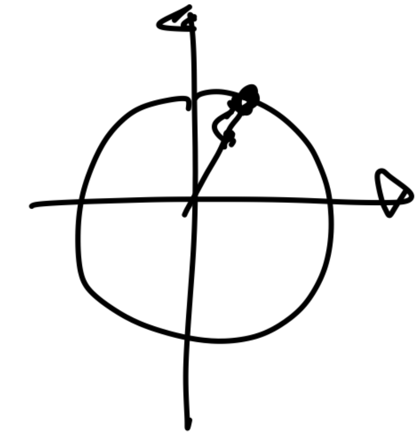
$$\underbrace{\|p_i\|_2}_{} = x_i^T u$$

$$\boxed{\begin{array}{ll} \underset{u}{\text{maximize}} & \frac{1}{n}\sum_{i=1}^{n} (u^T x_i)^2 \\ \text{s.t.} & \|u\|_2^2 = 1 \end{array}}$$

$$\frac{1}{n}\sum_{i=1}^{n} \underbrace{u^T x_i}_{\mathbb{R}} \underbrace{x_i^T u}_{\mathbb{R}} = u^T \overbrace{\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T\right)}_{\mathbb{R}^{d \times d}}}^{C} u$$

Covariance Matrix

$$\max_{u} \quad u^T C u \qquad \text{(a)}$$
$$\text{s.t.} \quad \|u\|_2^2 = 1.$$



$$\max_{u} \quad u^T C u \geq 0 \qquad \text{(b)}$$
$$\text{s.t.} \quad \boxed{\|u\|_2^2 \leq 1}$$

Claim: optimal u^* of (a) is the same as optimal u^* of (b)

$$u^T C u = \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2 \geq 0$$

$$\max_{u} \quad \underbrace{u^T C u - \lambda \|u\|_2^2}_{f_\lambda(u)} \qquad \text{(c)} \leftarrow \text{unconstrained}$$

Claim: $\exists \lambda \in \mathbb{R}^+$ s.t. u^* of (c) is equal to u^* of (b)

strategy: - Identify $u^*(\lambda)$ of (c).

└ Check which λ gives $\|u^*(\lambda)\|_2^2 = 1$

Goal: $u^*(\lambda)$ of $\max_u \; u^T C u - \lambda \|u\|_2^2$

$$\nabla_u F_\lambda(u) = 2 \cdot C \cdot u - 2\lambda \cdot u = 0$$

$$\boxed{\underset{\mathbb{R}^{d\times d}}{C} \cdot u = \underset{\mathbb{R}}{\lambda} \cdot u}$$

$\rightarrow$ Optimal $u^*(\lambda)$ has to be a eigenvector C.

let $(\lambda^{(1)}, u^{(1)})$ the eigen pair, s.t. $\lambda^{(1)} \geq \lambda^{(2)}, \dots, \lambda^{(d)}$

e.g. $\boxed{u^T C u \leq (u^{(1)})^T C u^{(1)} = \lambda^{(1)}}$

$\lambda > \lambda^{(1)}$, $\max_u F_\lambda(u) = 0$, with $\boxed{u^*(\lambda) = 0}$

$$F_\lambda(u) = u^T C u - \lambda \|u\|_2^2 \leq \underbrace{(\lambda^{(1)} - \lambda)}_{<0} \underbrace{\|u\|_2^2}_{\geq 0} \leq 0$$

$\lambda < \lambda^{(1)}$, $\max_u F_\lambda(u) = \infty$, with $\|u^*(\lambda)\|_2^2 = \infty$

$\lambda = \lambda^{(1)}$, $\max_u F_\lambda(u) = 0$, with $\|u^*(\lambda)\|_2^2 = 1 \leftarrow u^*$ s.t

Solving the optimization

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

subject to $\|u\|_2^2 = 1$

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

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

subject to $\|u\|_2^2 \leq 1$

- the reason is that, because $u^T \mathbf{C} u \geq 0$ for all $u \in \mathbb{R}^d$ (which we will prove in a bit), 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

Solving the optimization

- we are left to prove the following claim
- claim: $u^T \mathbf{C} u \geq 0$
  where $\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$
- proof:

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

$$= \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2 \geq 0$$

for any $u \in \mathbb{R}^d$

Solving the optimization

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

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

- 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 λ such that the optimal solution of (c) is the same as the optimal solution of (b)
- further, for this choice of λ, the optimal u has $\|u\|_2 = 1$

- our strategy is to analytically describe $u(\lambda)$ that is optimal solution of (c), and find λ such that $\|u(\lambda)\|_2^2 = 1$

Solving the optimization

- to find such λ and the corresponding u, we solve the unconstrained optimization, by setting the gradient to zero
$$\nabla F_\lambda(u) \;=\; 2\mathbf{C}u - 2\lambda u \quad = \; 0$$
- the candidate solution satisfies: $\mathbf{C}u = \lambda u$, i.e. an eigenvector of $\mathbf{C}$

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

- let $(\lambda^{(1)}, u^{(1)})$ denote the largest eigenvalue and corresponding eigenvector of $\mathbf{C}$, with norm one, i.e. $\|u^{(1)}\|_2^2 = 1$
- one property of the largest eigenvalue is that
  - $u^T\mathbf{C}u \leq \lambda^{(1)}\|u\|_2^2$ and the maximum is achieved with $u = u^{(1)}$
- we claim that for
  - $\lambda > \lambda^{(1)}$, the optimal solution is $u = 0$ with objective value zero
  - $\lambda < \lambda^{(1)}$, one optimal solution is $u = cu^{(1)}$ with $c = \infty$, with objective value infinity
  - $\lambda = \lambda^{(1)}$, one optimal solution is $u = u^{(1)}$, with objective value zero

The solution

$$\text{maximize}_u \;\; u^T \mathbf{C} u - \lambda \|u\|_2^2$$

- if $\lambda < \lambda^{(1)}$ then one can take $u = cu^{(1)}$, which gives
$$F_\lambda(u) = \lambda^{(1)} c^2 - \lambda c^2 = (\underbrace{\lambda^{(1)} - \lambda}_{>0}) c^2$$
and we can now take c as large as we want to make the objective unbounded (and hence optimal u has norm unbounded)
- if $\lambda > \lambda^{(1)}$ then one can show that the optimal $u = 0$, as for any u with norm c,
$$F_\lambda(u) \leq \lambda^{(1)} c^2 - \lambda c^2 = (\underbrace{\lambda^{(1)} - \lambda}_{<0}) c^2$$
and taking $c = 0$ maximizes the objective
- hence, only $\lambda = \lambda^{(1)}$ gives optimal u with unit norm, i.e. $\|u\|_2^2 = 1$
and the optimal solution is $u = u^{(1)}$
- finally, we found the optimal solution of $\quad \text{maximize}_u \; u^T \mathbf{C} u$
subject to $\;\|u\|_2^2 = 1$
which is the eigenvector $u^{(1)}$ corresponding to the top eigenvalue $\lambda^{(1)}$ of $\mathbf{C}$

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 $\mathbf{C}$

$$\arg\min_{u} \sum_{i=1}^{n} \| x_i - p_i \|_2^2$$

$$= \sum_{i=1}^{n} \underbrace{\| x_i - u(u^T x_i) \|_2^2}$$

$$= \sum_{i=1}^{n} \left\{ \underbrace{\| x_i \|^2} - 2 x_i^T u u^T x_i + x_i^T u \underbrace{\boxed{u^T u}}_{=1} u^T x_i \right\}$$

$$= \arg\min_{u} \sum_{i=1}^{n} - x_i^T u u^T x_i$$

$$= \boxed{\arg\max \sum_{i=1}^{n} u^T x_i x_i^T u}$$

$$p_i = u(u^T x_i)$$

$$\text{s.t. } \underbrace{\| u \|_2^2}_{= u^T u} = 1$$

Alternate view of PCA: minimizing reconstruction error

$X \in \mathbb{R}^{n\times d}$, $U = [u_1 \cdots u_r] \in \mathbb{R}^{d\times r}$

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

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

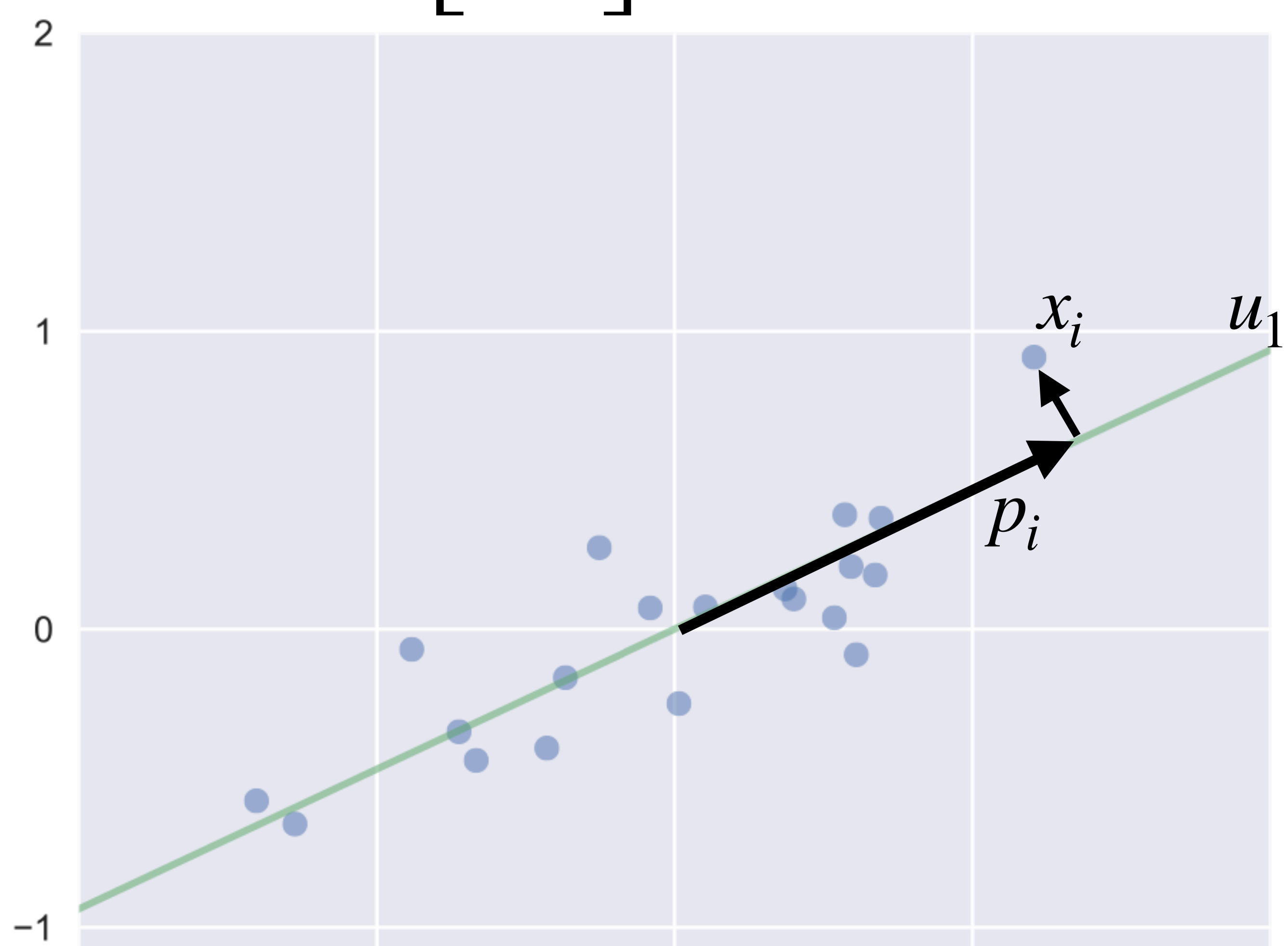
$$x_i = \begin{bmatrix} x_i[1] \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ x_i[d] \end{bmatrix} \longrightarrow a_i = \begin{bmatrix} a_i[1] \\ \vdots \\ a_i[r] \end{bmatrix}$$

- those directions that minimize the average reconstruction error for a dataset is called the **principal components**
- given a choice of $u_1, \ldots, u_r$, the best representation p_i of x_i is the projection of the point onto the subspace spanned by u_j's, i.e.

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

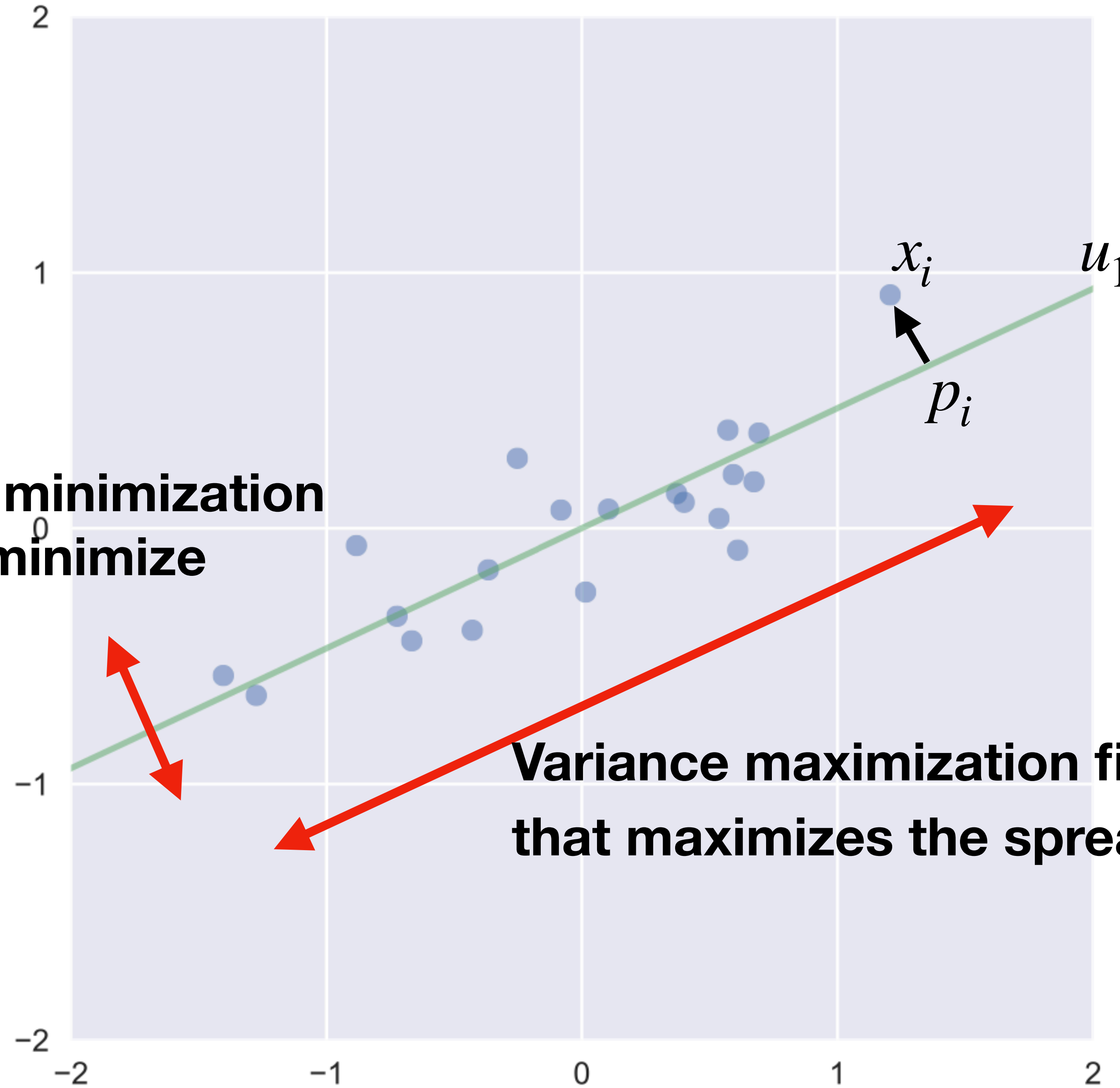
- the goal is to find $u_1, \ldots, u_r$ to minimize the reconstruction error

$$\frac{1}{n}\sum_{i=1}^{n} \|x_i - p_i\|^2$$



Variance maximization vs. reconstruction error minimization

- both give the same principal components as optimal solution



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

Alternate view of PCA: minimizing reconstruction error

$$\text{minimize} \quad \frac{1}{n}\sum_{i=1}^{n} \|x_i - p_i\|^2$$

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

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

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

- we will not formally prove it, but the optimal solution of this problem is the r principal components

Principal Component Analysis

- input: data points $\{x_i\}_{i=1}^n$, target dimension $r \ll d$
- output: r-dimensional subspace
- 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 $(u_1, \ldots, u_r)$ 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
  $$a_i = \mathbf{U}^T(x_i - \bar{x}) \in \mathbb{R}^r$$

reconstruction

- given principal component $\mathbf{U} \in \mathbb{R}^{d \times r}$ and $\bar{x} \in \mathbb{R}^d$, each data point is represented in a lower dimension as

$$a_i \;=\; \mathbf{U}^T(x_i - \bar{x})$$

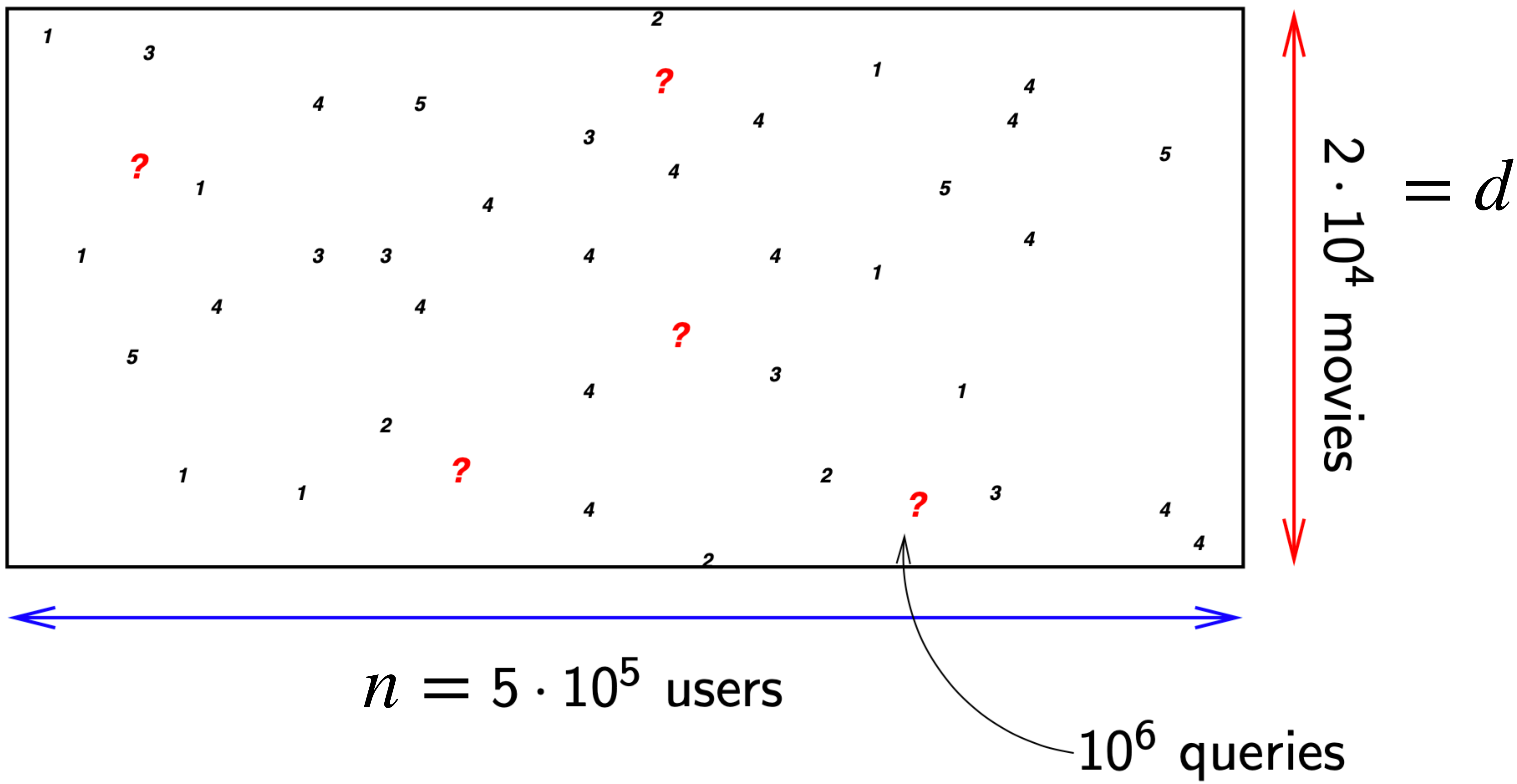
- then the reconstruction of the data point is

$$p_i \;=\; \bar{x} + \sum_{j=1}^{r} a_i[j] u_j \;=\; \bar{x} + \mathbf{U} a_i$$

- the reconstruction error is

$$\begin{aligned} \|x_i - p_i\|_2^2 \;&=\; \|(x_i - \bar{x}) - (p_i - \bar{x})\|_2^2 \\ &=\; \|(x_i - \bar{x}) - \mathbf{U} a_i\|_2^2 \end{aligned}$$

Matrix completion for recommendation systems



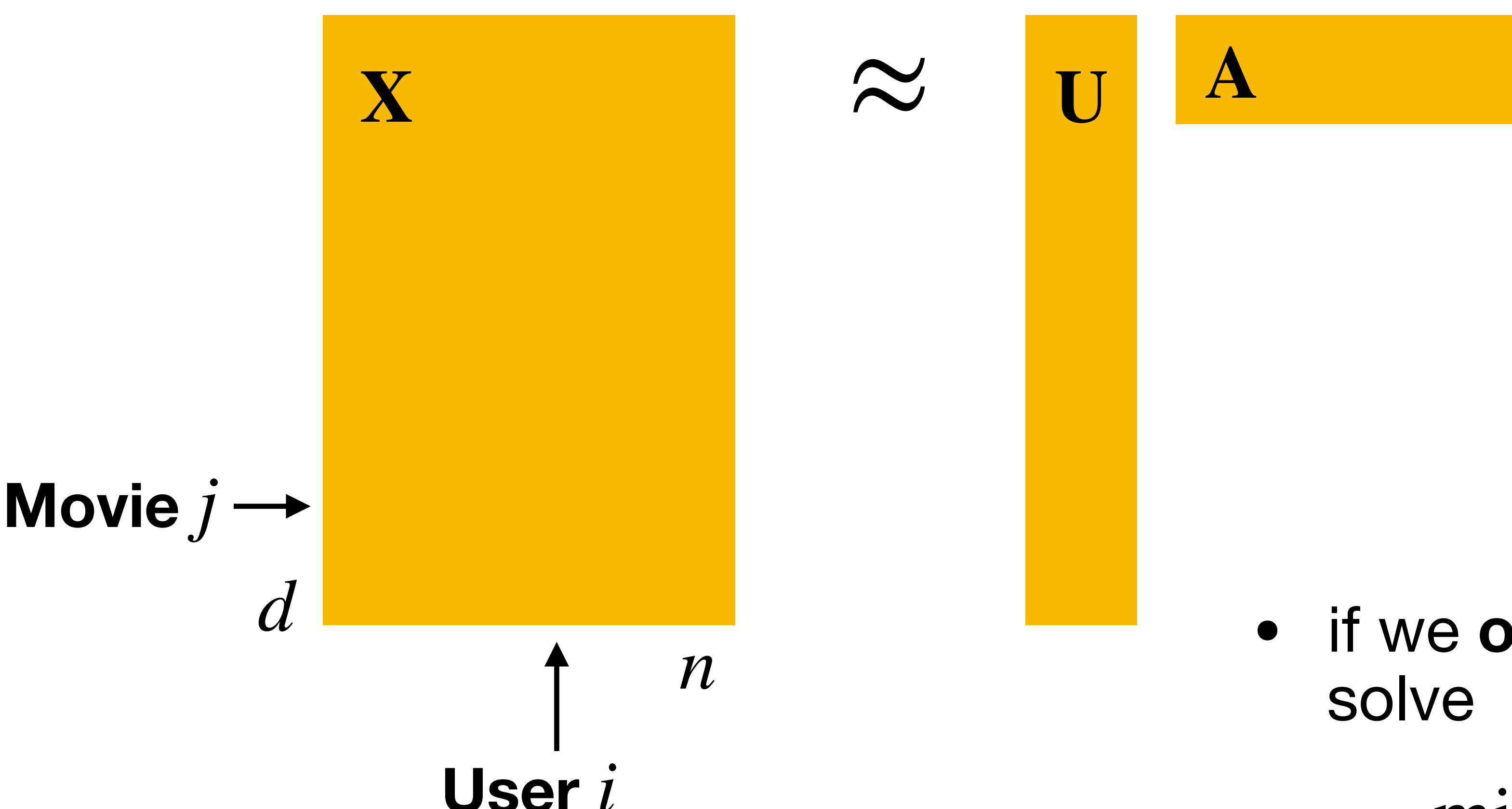
- users provide ratings on a few movies, and we want to predict the missing entries in this ratings matrix, so that we can make recommendations
- without any assumptions, the missing entries can be anything, and no prediction is possible

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 $\mathbf{X} = [x_1 \quad x_2 \quad \cdots \quad x_n] \in \mathbb{R}^{d \times n}$ 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 $\mathbf{X}$



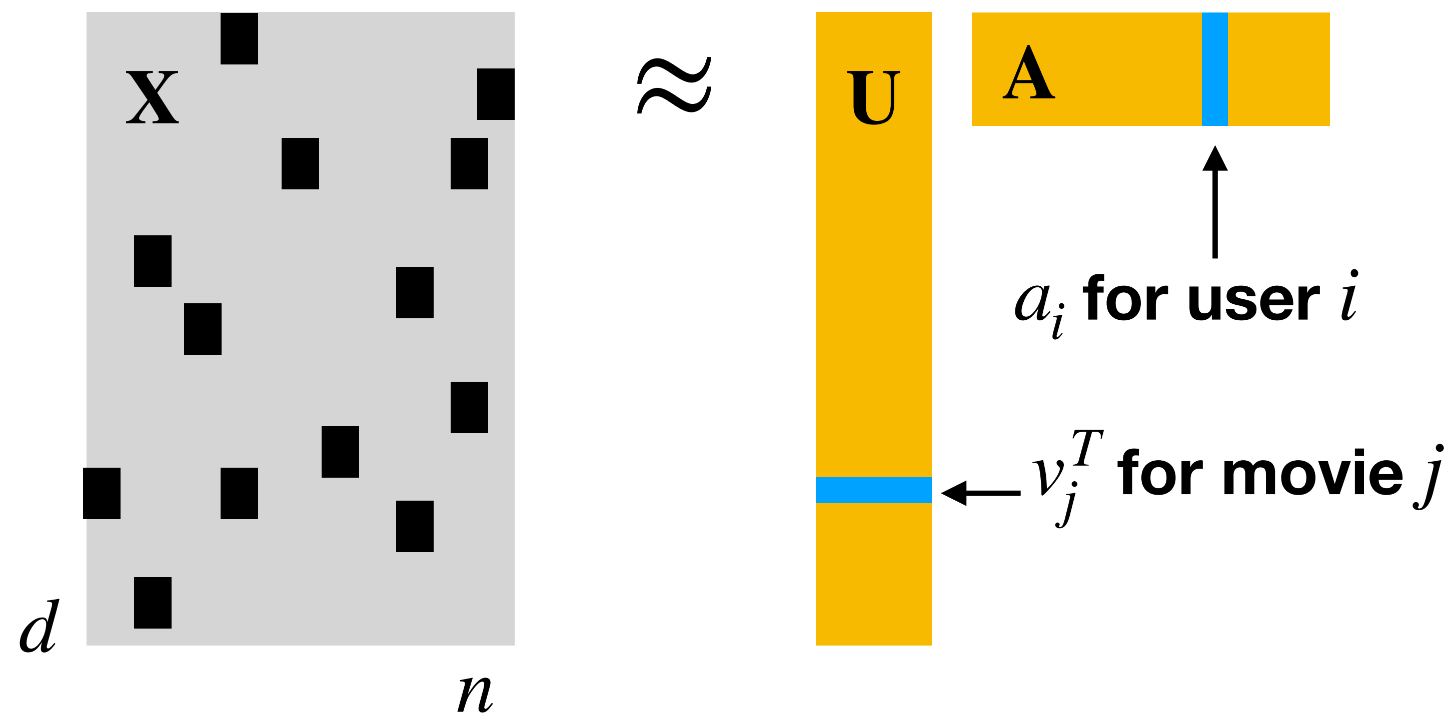
- if we **observe all entries** of $\mathbf{X}$, then we can solve

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

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

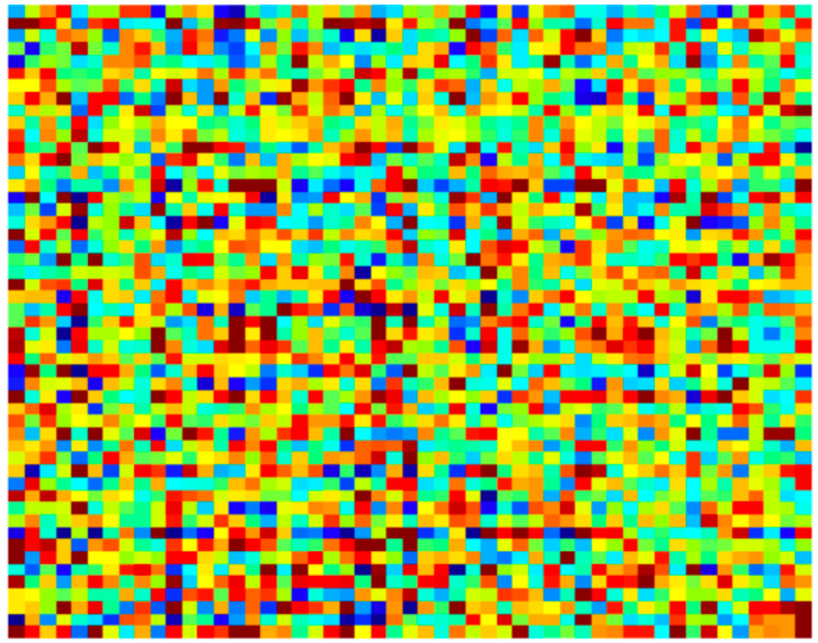
- a natural approach to fit v_j's and $a_i's$ to given training data is to solve

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

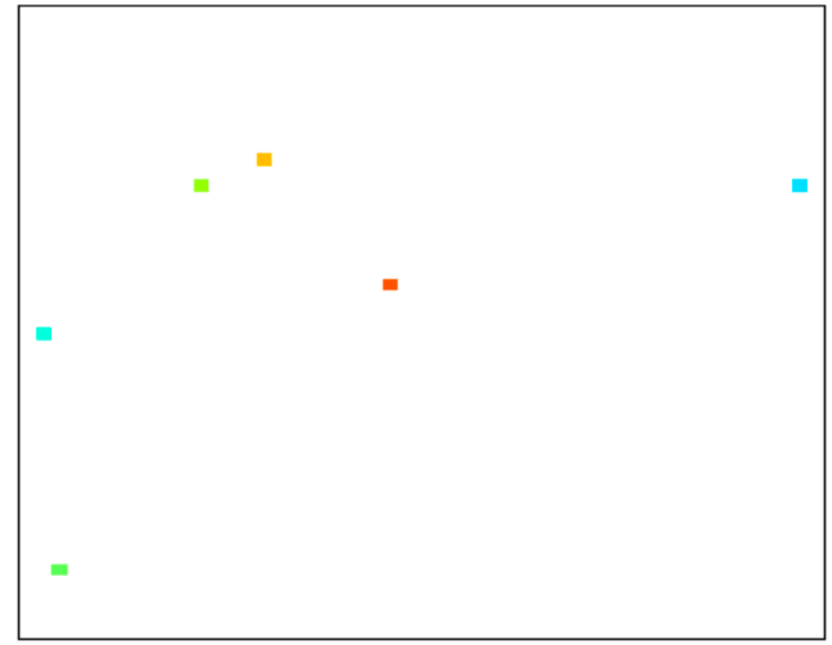
- this can be solved, for example via gradient descent or alternating minimization
- this can be quite accurate, with small number of samples

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

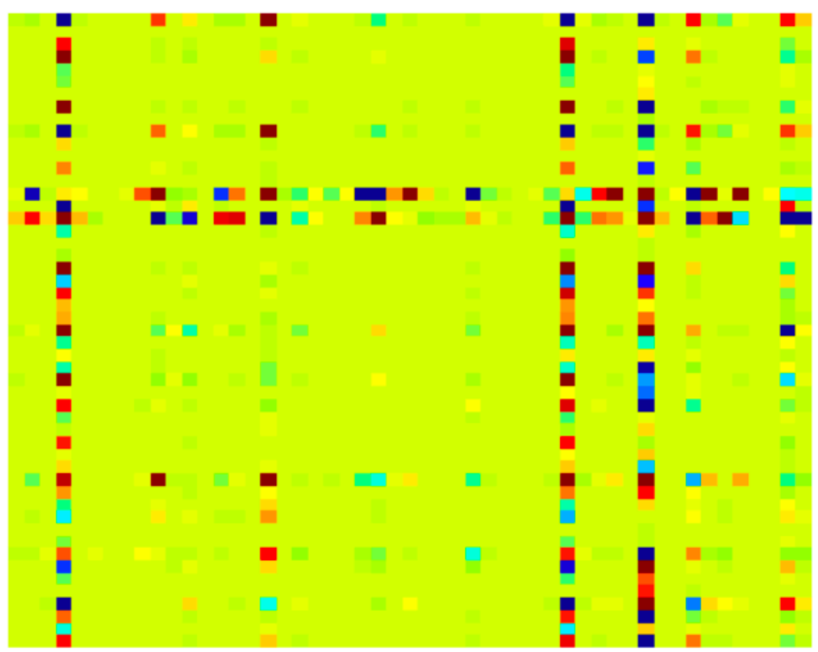
low-rank matrix $\mathbf{X}$



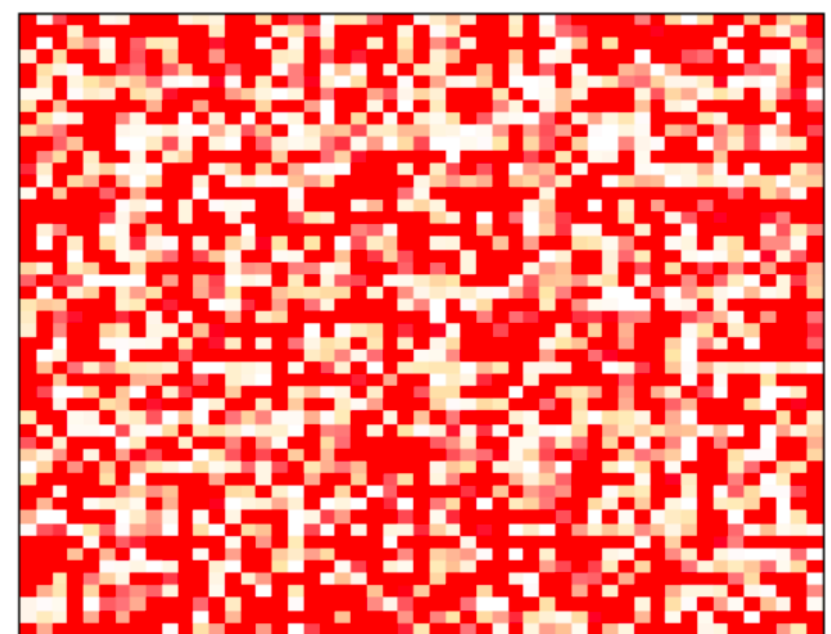
sampled matrix



Gradient descent output $\mathbf{UA}$



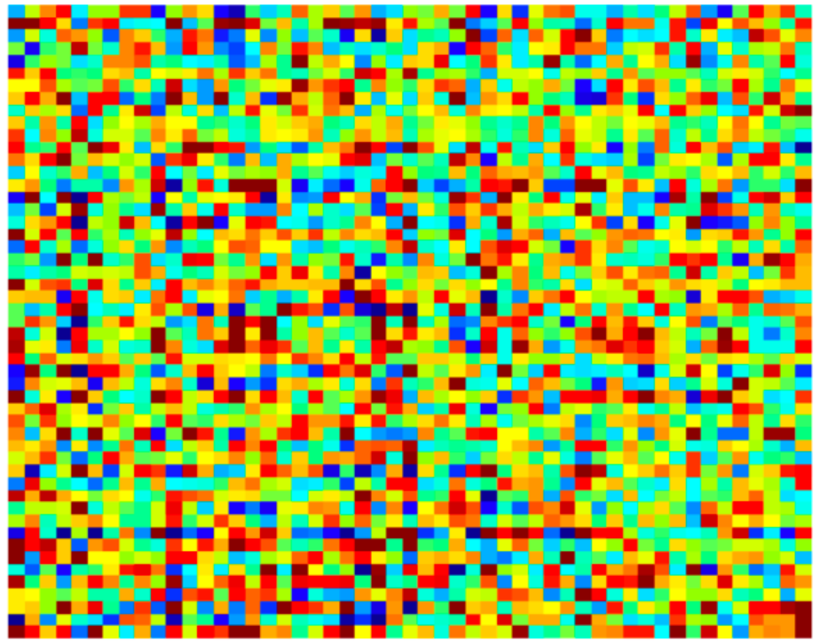
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



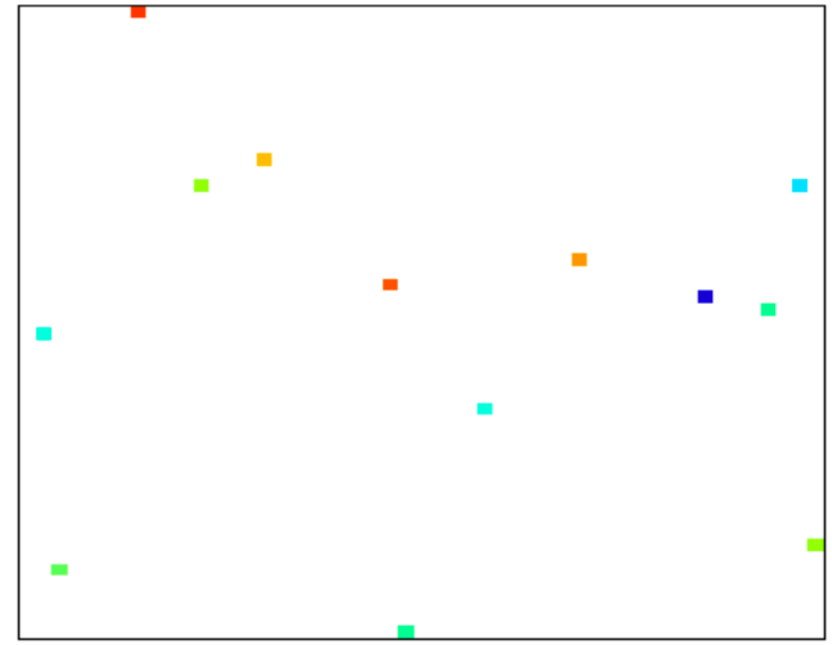
0.25% sampled

Example: 2000×2000 rank-8 random matrix

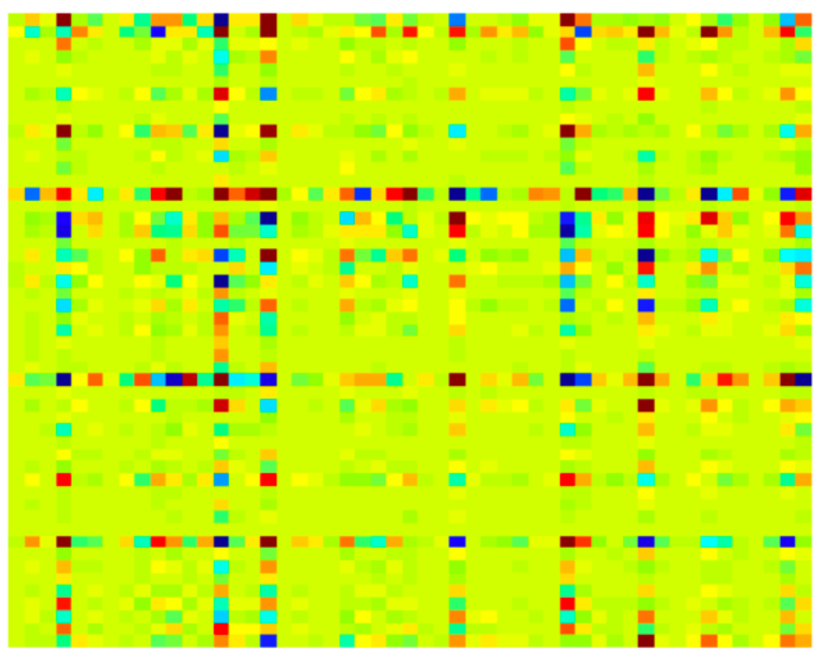
low-rank matrix $\mathbf{X}$



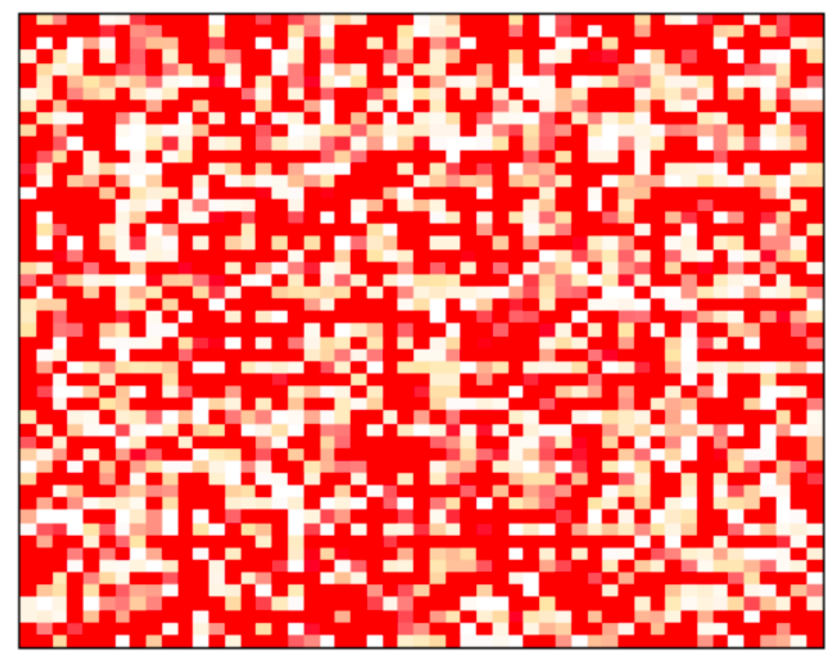
sampled matrix



Gradient descent output $\mathbf{UA}$



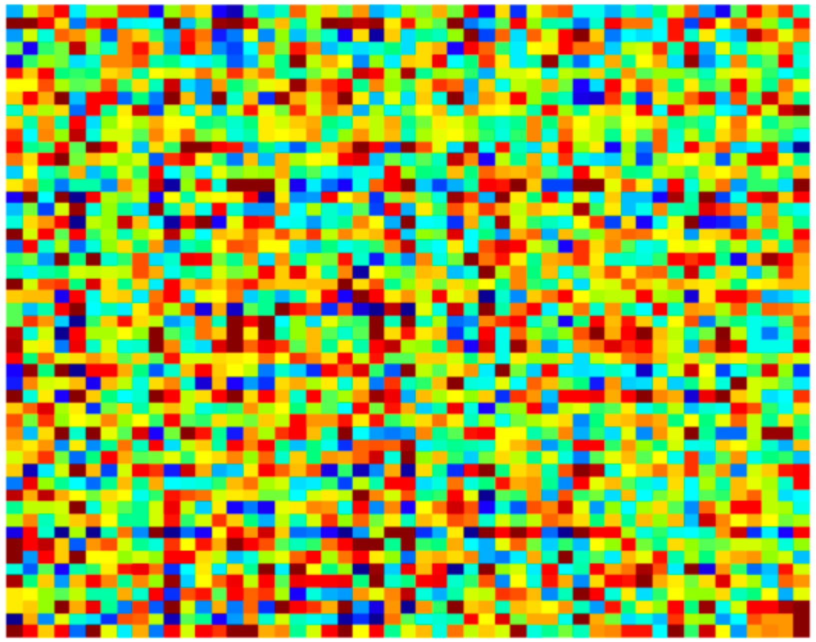
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



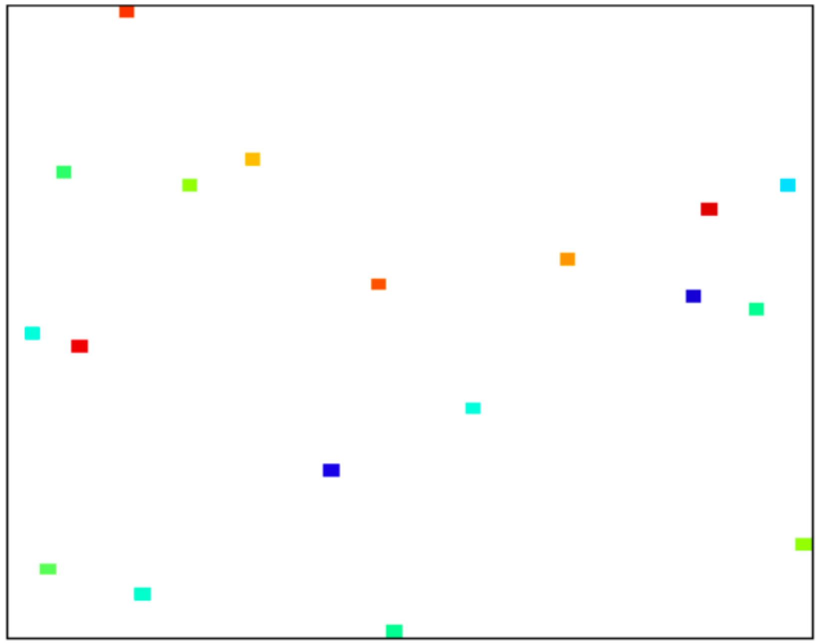
0.50% sampled

Example: 2000×2000 rank-8 random matrix

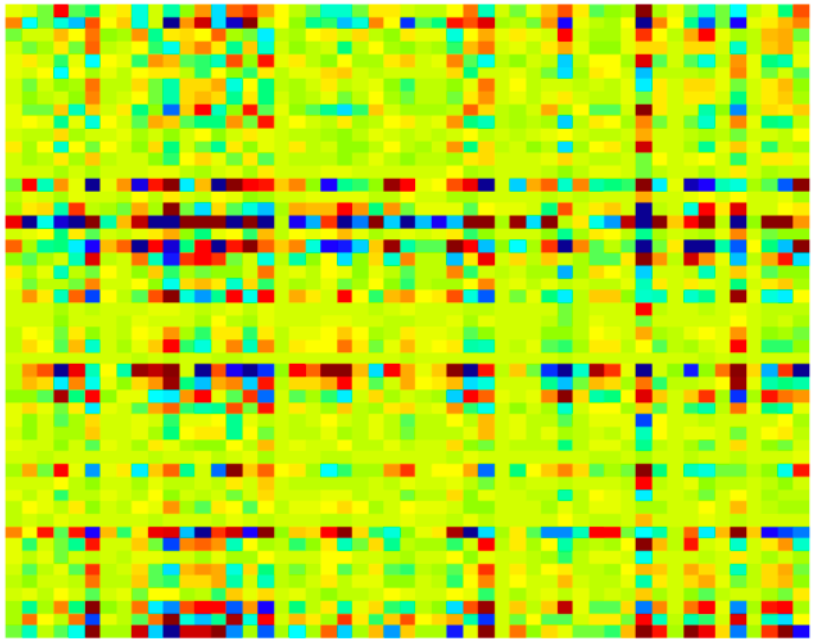
low-rank matrix $\mathbf{X}$



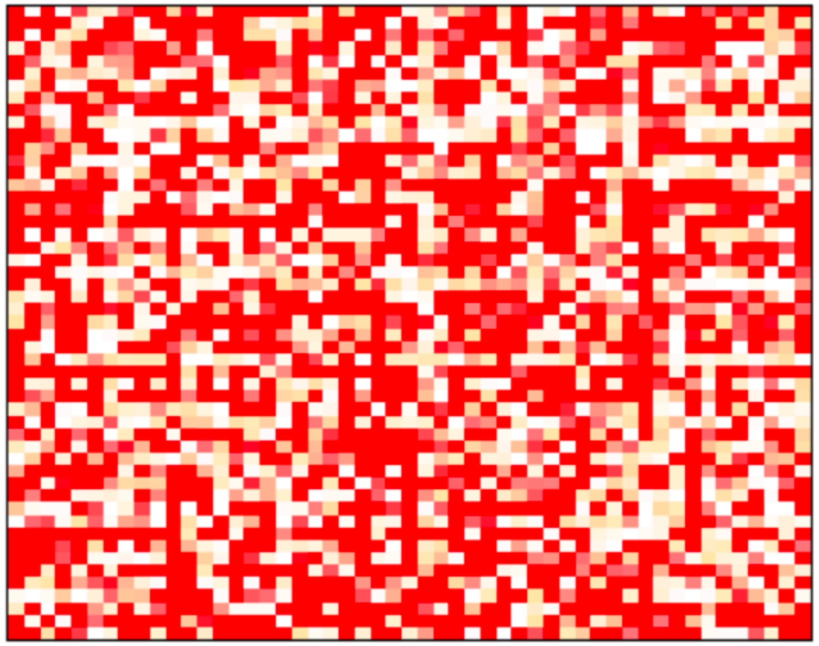
sampled matrix



Gradient descent output $\mathbf{UA}$



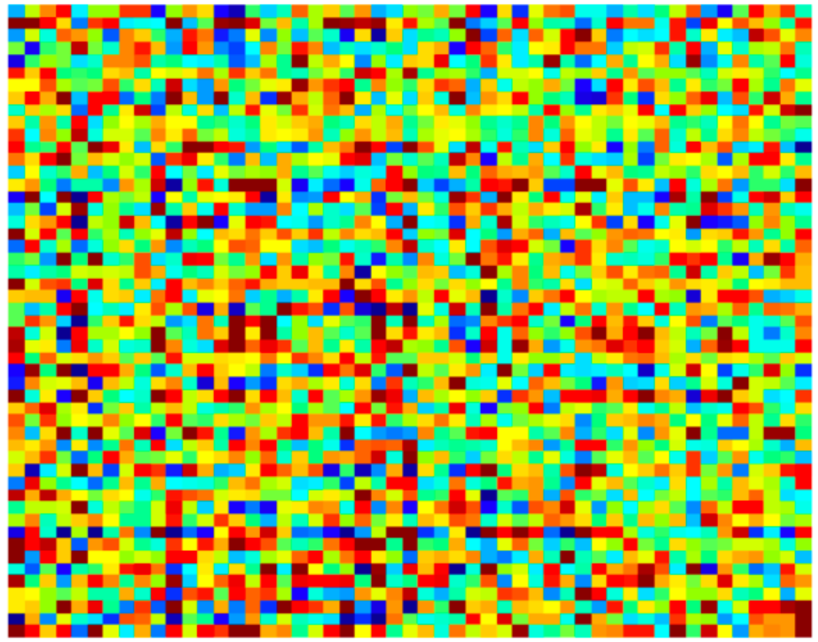
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



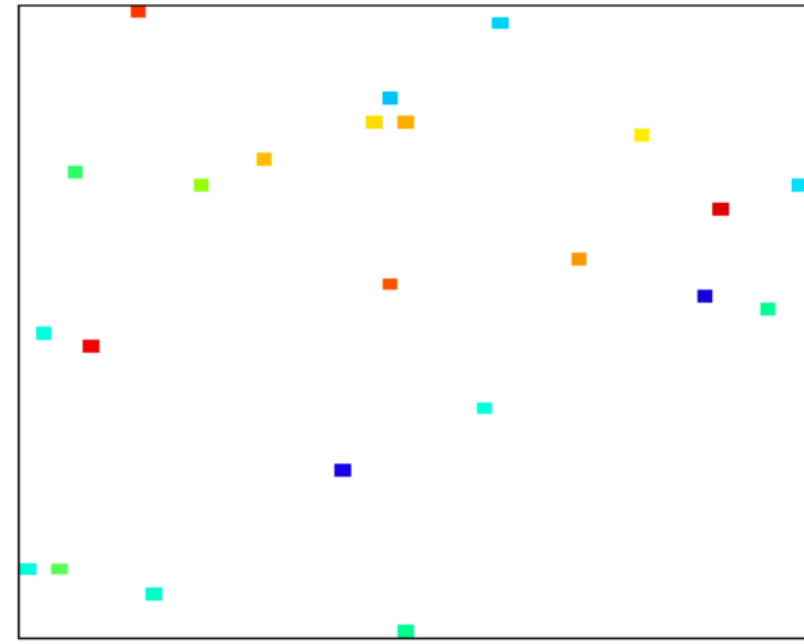
0.75% sampled

Example: 2000×2000 rank-8 random matrix

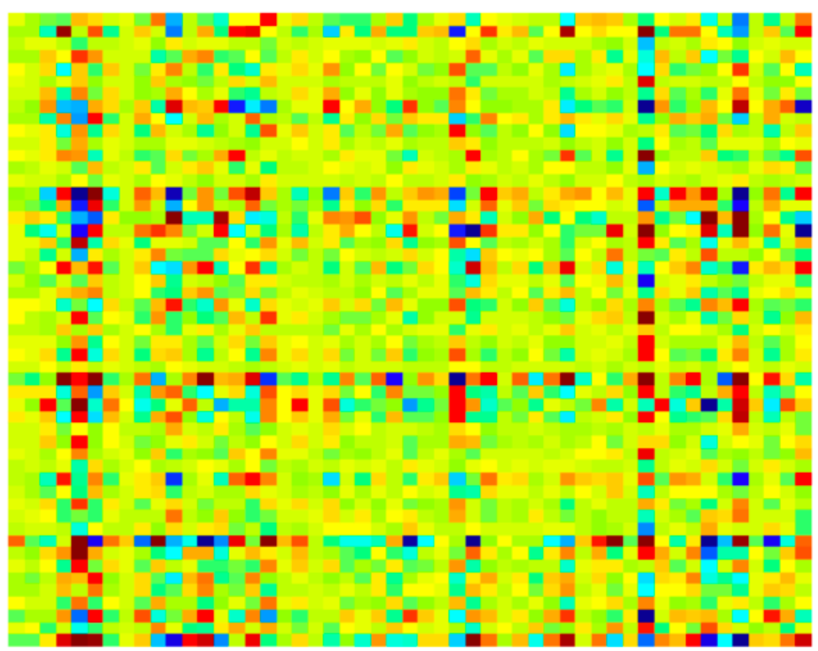
low-rank matrix $\mathbf{X}$



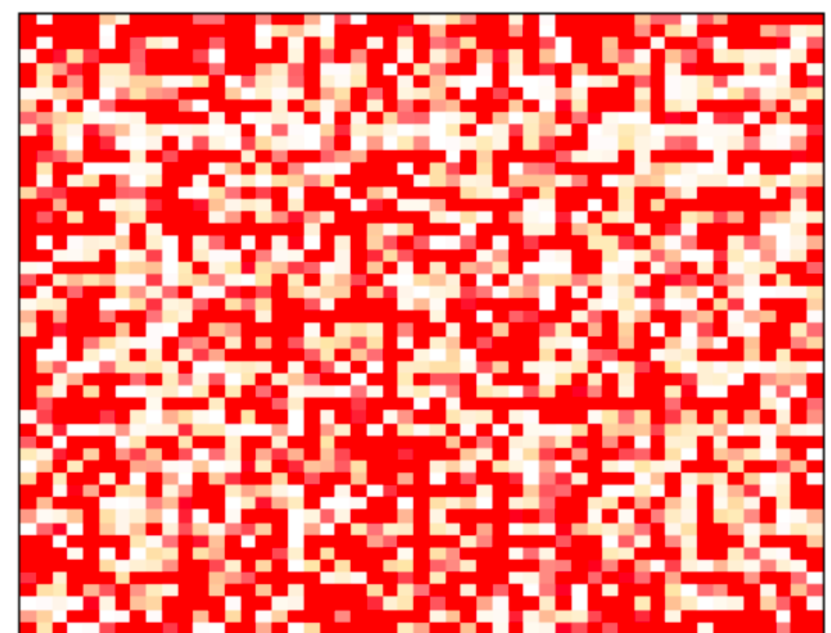
sampled matrix



Gradient descent output $\mathbf{UA}$



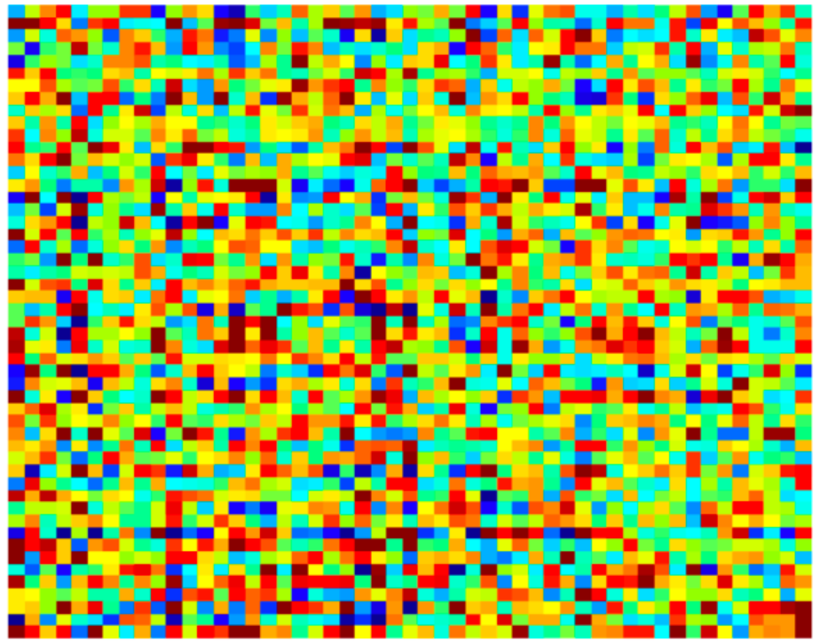
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



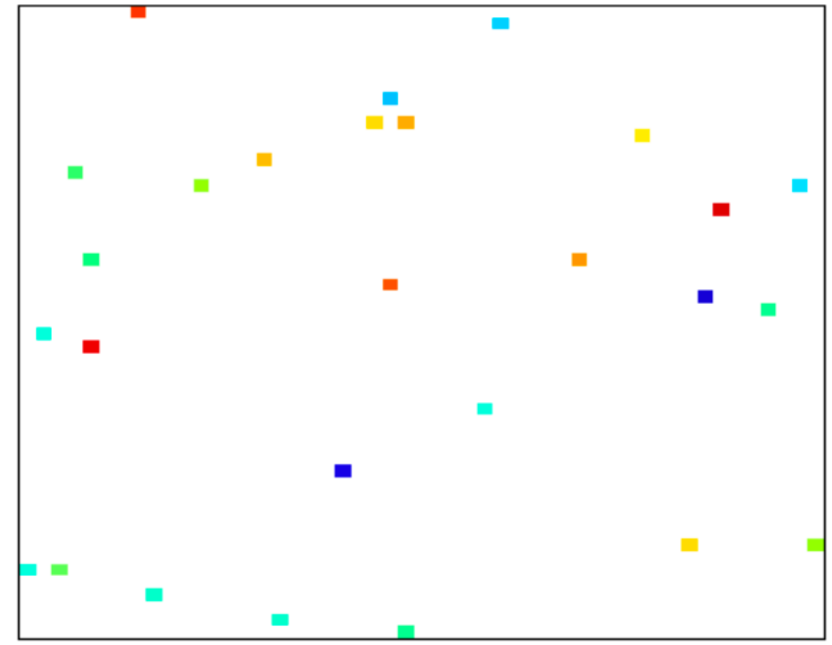
1.00% sampled

Example: 2000×2000 rank-8 random matrix

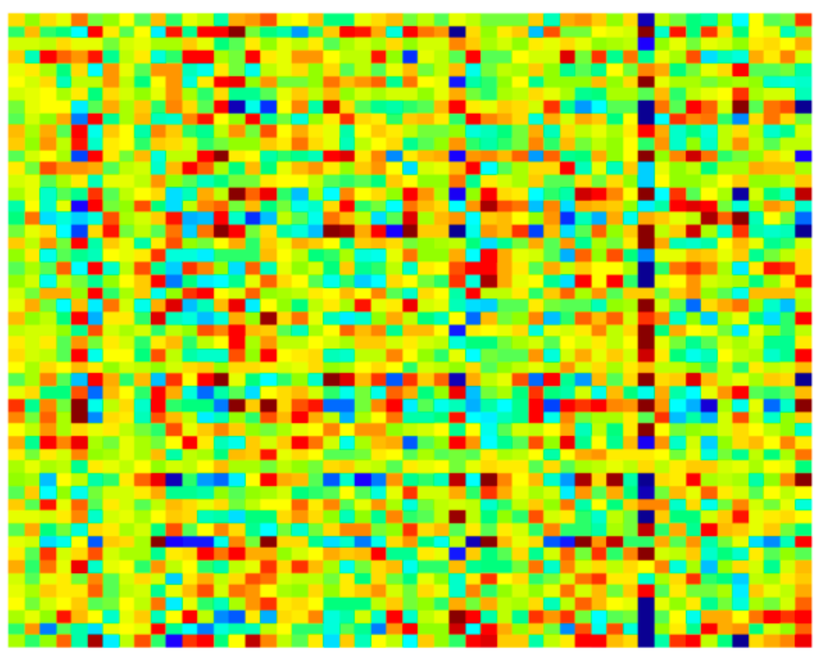
low-rank matrix $\mathbf{X}$



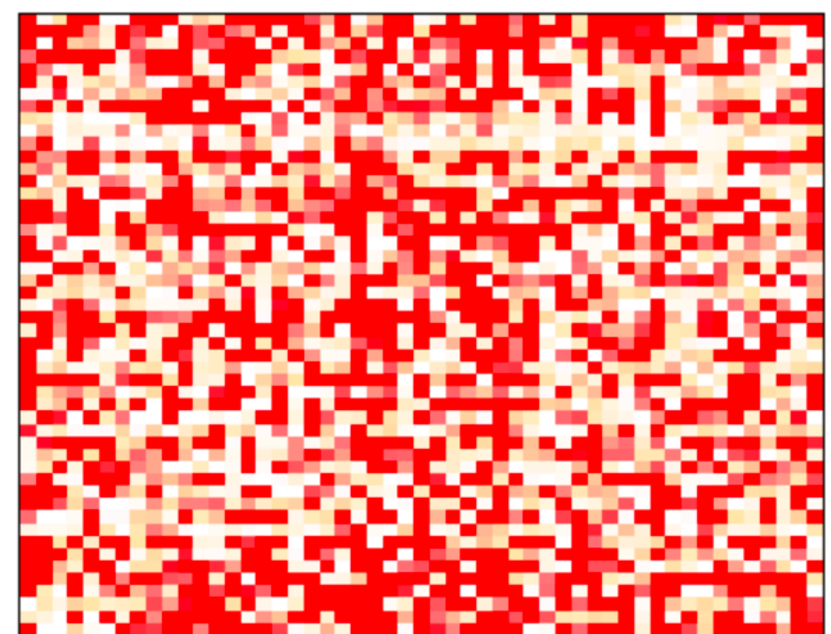
sampled matrix



Gradient descent output $\mathbf{UA}$



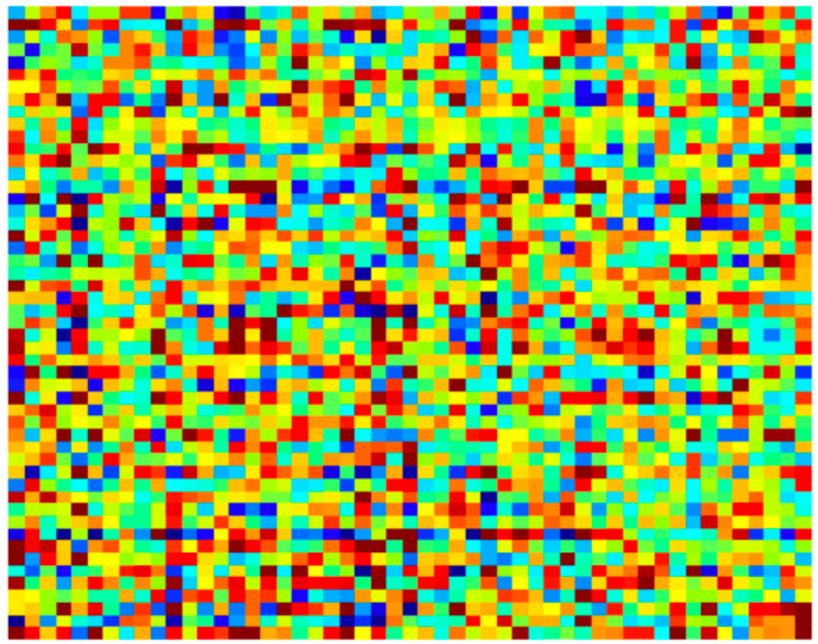
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



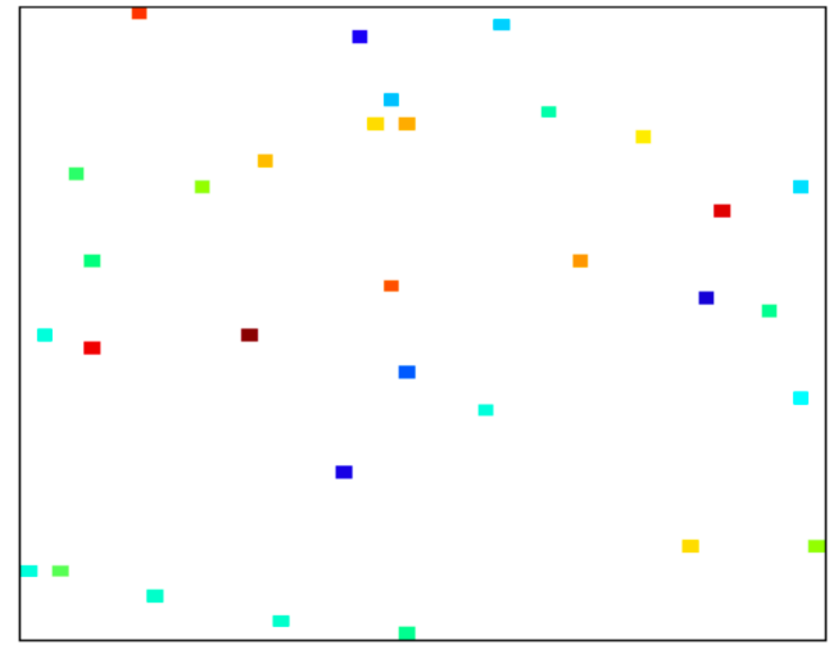
1.25% sampled

Example: 2000×2000 rank-8 random matrix

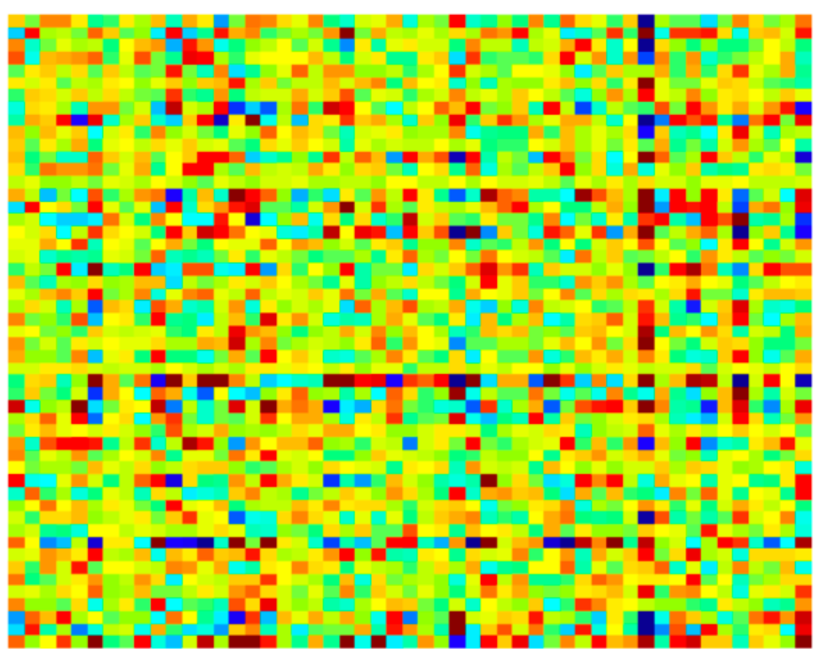
low-rank matrix $\mathbf{X}$



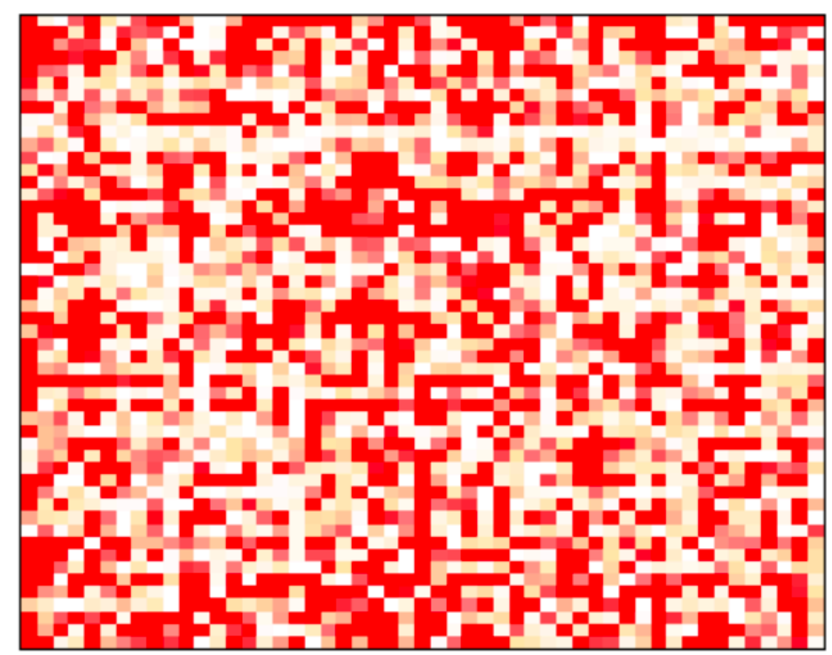
sampled matrix



Gradient descent output $\mathbf{UA}$



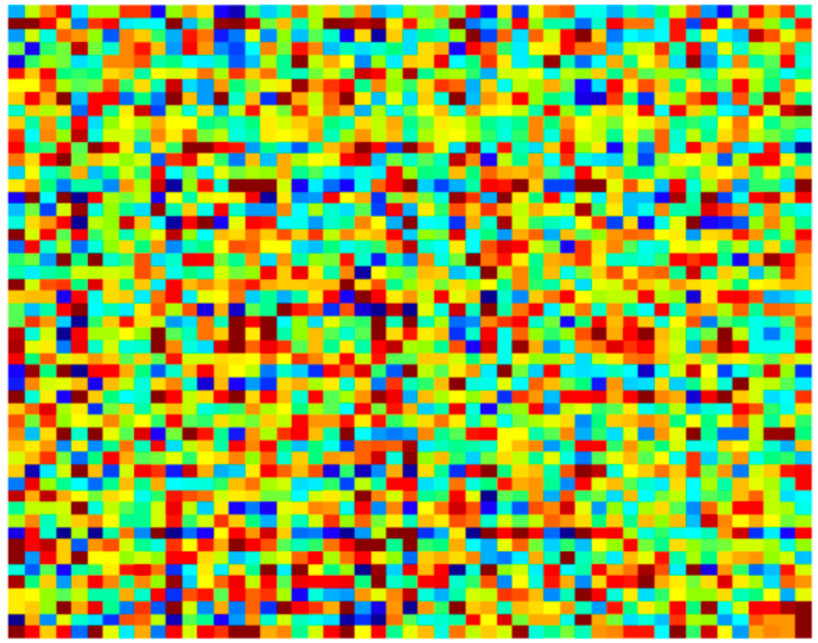
squared error $(\mathbf{X}_{ji} - (\mathbf{UA})_{ji})^2$



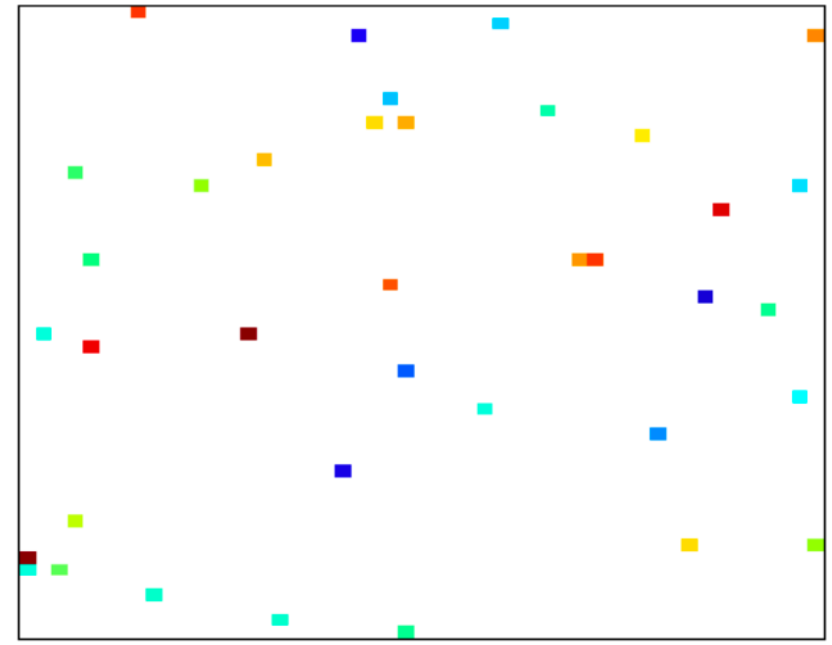
1.50% sampled

Example: 2000×2000 rank-8 random matrix

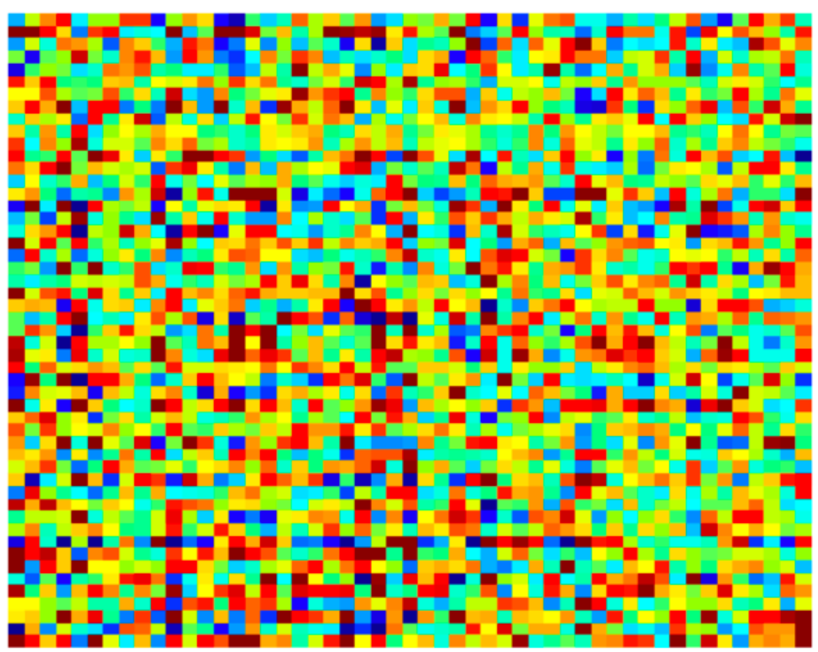
low-rank matrix $\mathbf{X}$



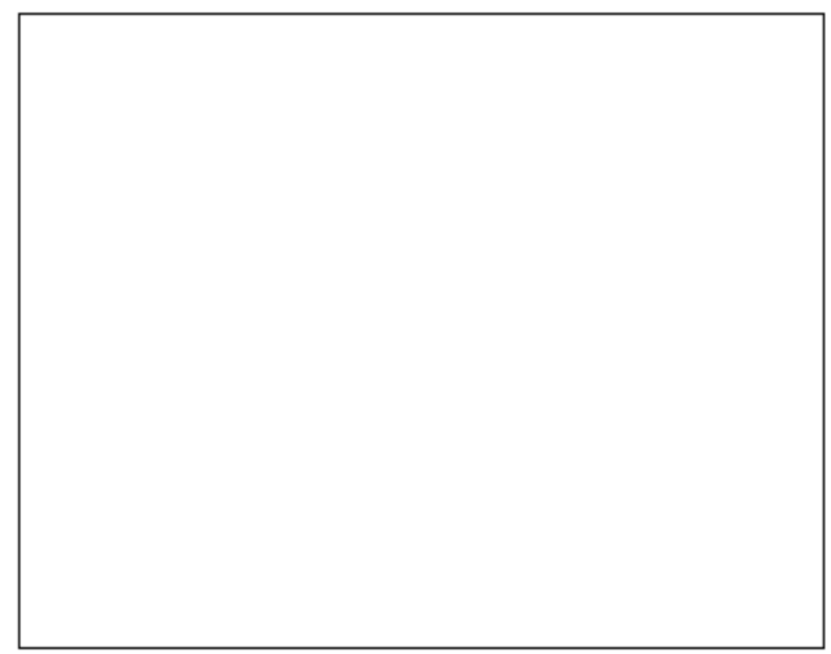
sampled matrix



Gradient descent output $\mathbf{UA}$



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



1.75% sampled

Matrix completion

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

- Gradient descent on $\{v_j\}_{j=1}^d$ and $\{a_i\}_{i=1}^n$ can be implemented via

$$v_j^{(t)} \leftarrow v_j^{(t-1)} - 2\eta \sum_{i\in S_j} ((v_j^{(t-1)})^T a_i^{(t-1)} - \mathbf{X}_{ji}) a_i^{(t-1)}$$

for all $j \in \{1,\ldots,d\}$, where S_j is the set of users who rated movie j and

$$a_i^{(t)} \leftarrow a_i^{(t-1)} - 2\eta \sum_{j\in S_i} ((v_j^{(t-1)})^T a_i^{(t-1)} - \mathbf{X}_{ji}) v_j^{(t-1)}$$

for all $i \in \{1,\ldots,n\}$, where S_i is the set of movies that were rated by user i

Matrix completion

- $$\text{minimize}_{\mathbf{U},\mathbf{A}} \sum_{(i,j)\in S_{\text{train}}} (\mathbf{X}_{ji} - v_j^T a_i)^2$$
- alternating minimization
  - repeat
    - fix v_j's and find optimal $a_i's$
      - for each i, set the gradient to zero:
        $2\sum_{j\in S_i}((v_j^{(t-1)})^T a_i - \mathbf{X}_{ji})v_j^{(t-1)} \;=\; 0$, which gives

$$a_i\Big(\sum_{j\in S_i} v_j v_j^T\Big) = \sum_{j\in S_i} \mathbf{X_{ij}} v_j$$

$$a_i \;=\; \Big(\sum_{j\in S_i} v_j v_j^T\Big)^{-1} \sum_{j\in S_i} \mathbf{X_{ij}} v_j$$

    - fix $a_i's$ and find optimal v_j's (similarly)