

CSE 446

Singular Value Decomposition

(SVD) → tool from lin alg to compute PCA

Natasha Jaques

why PCA?

- ① easier to work with smaller d data
a) computational efficiency
b) memory
} compro
- ② Understanding patterns/structure with d data
a) visualization!

W

Recap: PCA = Principal Component Analysis

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

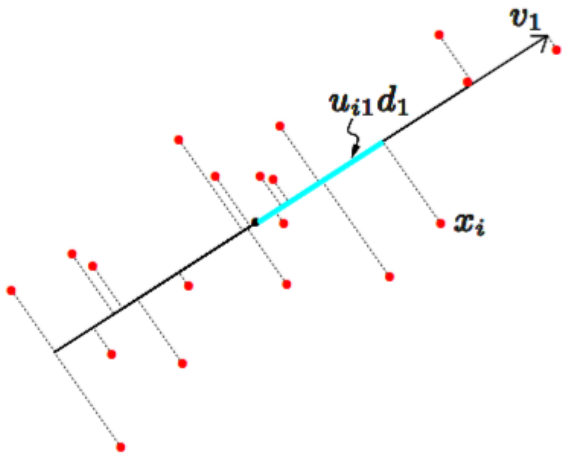
$$\min_{\mathbf{V}_q} \sum_{i=1}^N \|(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})\|_2^2$$

$$\min_{\mathbf{V}_q} \text{Tr}(\Sigma) - \text{Tr}(\mathbf{V}_q^T \Sigma \mathbf{V}_q)$$

Covariance
↓
Σ

Minimizing the reconstruction error for a compressed representation of the data

Maximum eigen-space of Σ



$$\text{Tr}(X) \triangleq \sum_{i,j} x_{ij}$$

$$\max_{\mathbf{V}_q} \text{Tr}(\mathbf{V}_q^T \Sigma \mathbf{V}_q)$$

$$\mathbf{V}_q = [v_1, \dots, v_q]$$

↑
eigenvectors

where $\mathbf{V}_q = [v_1, v_2, \dots, v_q]$ is orthonormal: $\mathbf{V}_q^T \mathbf{V}_q = I_q$

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

Warm-up: Eigenvalue decomposition

eigenvalues are only defined for square matrices

Suppose that a square matrix $A \in \mathbb{R}^{n \times n}$, has

a set of right eigen-vectors $\{v_i \in \mathbb{R}^n\}_{i=1}^n$ and corresponding eigenvalues $\{\lambda_i\}_{i=1}^n$

- $A v_1 = \lambda_1 v_1$

- $A [v_1, v_2] = [\lambda_1 v_1, \lambda_2 v_2] = [v_1, v_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$
↳ concat 2 vectors

- Let $V = [v_1, v_2, \dots, v_n]$, then $A V = V \Lambda$
↳ diagonal of eigenvalues

- Eigen value decomposition of A is $A = V \Lambda V^{-1}$
↳ matrix

- Now, suppose A is symmetric, then its eigenvectors are orthonormal, i.e.,

$$V^T V = V V^T = I, \text{ which is equivalent to saying that}$$

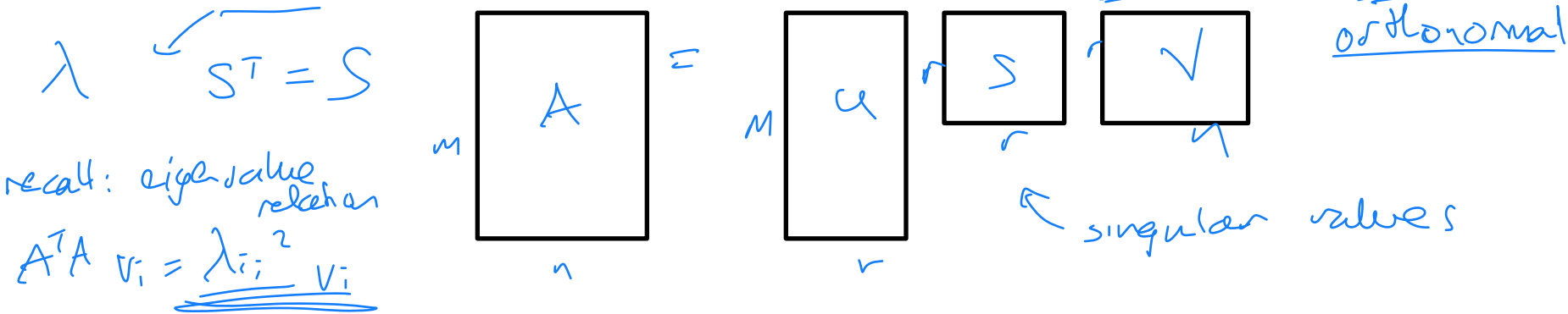
$$\underline{v_i^T v_j} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \quad \text{diagonal of } v^T v$$

→ no closed form soln, need to find numerically

Singular Value Decomposition (SVD): definition

→ equiv to eigenvalue decomp for non-square matrix

Theorem [SVD]: Let $A \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $A = USV^T$ where $S \in \mathbb{R}^{r \times r}$ is diagonal with non-negative entries, $U^T U = I$, and $V^T V = I$.



What is $A^T A v_i = (USV^T)^T (USV) v_i$

$AA^T =$

gives right $m \times m$ singular vectors

$$= V^T S U^T U S V v_i = V^T S^2 V v_i$$



What is $AA^T u_i =$
 gives left $n \times n$ singular vectors

$A^T A =$

eigen decomp of $A^T A$ by defn

- v_i 's are the r eigen vectors of $A^T A$ with corresponding eigen values S_{ii}^2 's
- u_i 's are the r eigen vectors of AA^T with corresponding eigen values S_{ii}^2 's

• Computing SVD takes $O(mnr)$ operations

Singular Value Decomposition (SVD): definition

Theorem [SVD]: Let $A \in \mathbb{R}^{m \times n}$ with rank $r \leq \min\{m, n\}$. Then $A = USV^T$ where $S \in \mathbb{R}^{r \times r}$ is diagonal with non-negative entries, $U^T U = I$, and $V^T V = I$.

$AV = \lambda V$



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

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

\mathbf{V} are the first r eigenvectors of $\mathbf{A}^T \mathbf{A}$ with eigenvalues $\text{diag}(\mathbf{S}^2)$

\mathbf{U} are the first r eigenvectors of $\mathbf{A} \mathbf{A}^T$ with eigenvalues $\text{diag}(\mathbf{S}^2)$

Singular Value Decomposition (SVD): property 1

- 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

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

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

- The optimal rank- r approximation is $L = U_{1:r} S_{1:r,1:r} V_{1:r}^T$ → SVD
- get whatever compression amount you want by reding off SVD matrix*
- ordered by largest singular values*

Singular Value Decomposition (SVD): property 2

- 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 subspace Q that maximizes

$$\text{maximize}_{\underline{Q \in \mathbb{R}^{n \times r}}} \text{Trace}(Q^T A^T A Q) = \sum_{j=1}^r Q_j^T A^T A Q_j$$

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

- The optimal rank- r subspace is $Q = V_{\underline{1:r}}$
↳ largest singular vectors

PCA is solved using SVD

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

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

Minimizing the reconstruction error
for a compressed representation of the data

Maximum eigen-space of Σ

where $\mathbf{V}_q = [v_1, v_2, \dots, v_q]$ is orthonormal: $\mathbf{V}_q^T \mathbf{V}_q = I_q$

$$\Sigma := \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T = \underline{\tilde{X}^T \tilde{X}}$$

$$\begin{aligned} \max_{V_q} \quad & V_q^T \Sigma V_q \\ \text{subject to} \quad & V_q^T V_q = I \end{aligned}$$

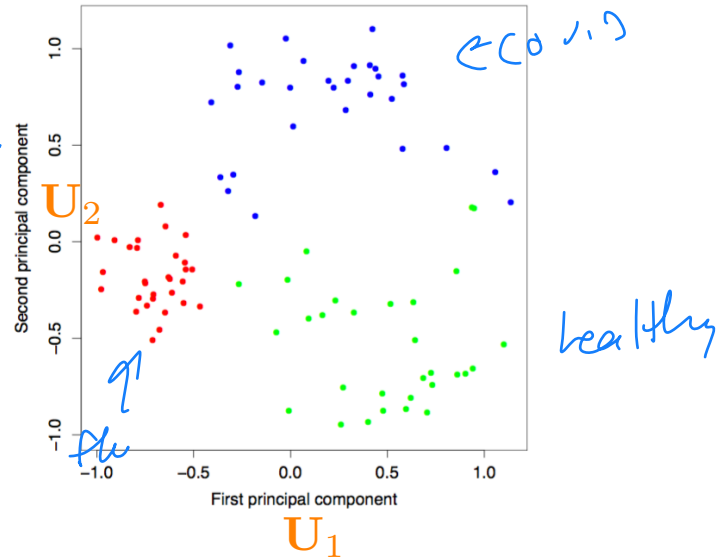
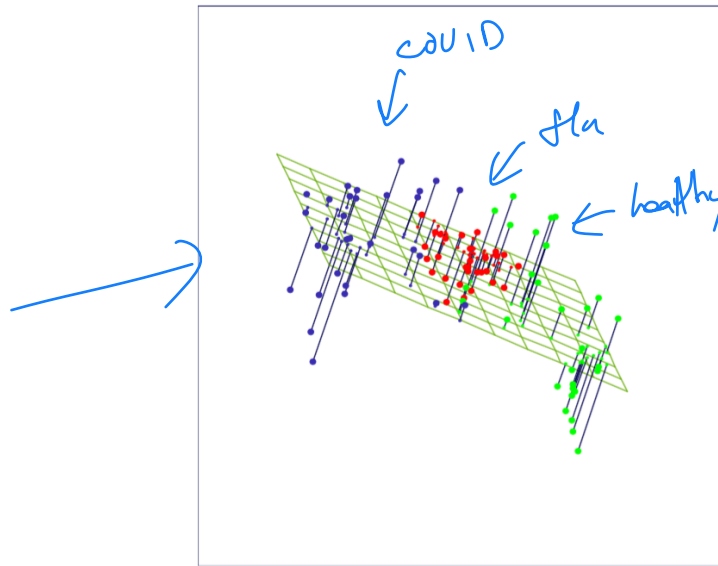
We can use SVD on the matrix \tilde{X} , and take the first r eigen pairs to find PCA.

↳ d e - mean

Example 1: Dimensionality reduction

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

$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$



high d
data each point
is a patient

Visualization
see clusters /
patterns

Example 2: Dimensionality reduction

\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 $x_i \in \mathbb{R}^{256}$

$$\begin{aligned} \hat{f}(\lambda) &= \bar{x} + \lambda_1 v_1 + \lambda_2 v_2 \\ &= \mathbf{3} + \lambda_1 \cdot \mathbf{3} + \lambda_2 \cdot \mathbf{3} \end{aligned}$$

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

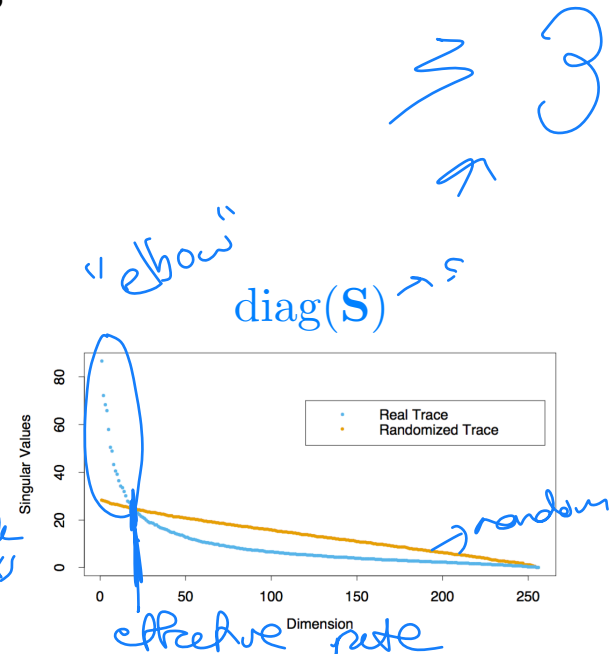
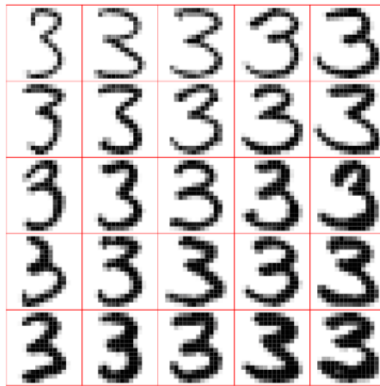
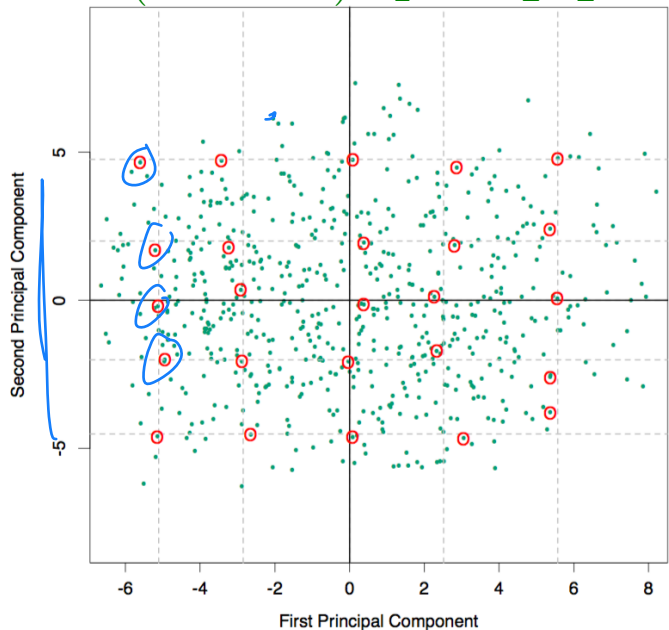


FIGURE 14.24. The 256 singular values for the digitized threes, compared to those for a randomized version of the data (each column of \mathbf{X} was scrambled).

$20 \rightarrow$ only need $q = 20$ to represent info in handwritten dig. 3

RECAP: PCA & SVD

$X \in \mathbb{R}^{n \times d}$
 unsupervised \rightarrow Want to compress to a $z \in \mathbb{R}^q$, $q \ll d$
 \rightarrow Learn some principle components $V_q = \begin{bmatrix} v_1 & v_2 & \dots \\ | & | & \\ | & | & \\ \dots & \dots & \dots \end{bmatrix}$
 $z_i = V_q^T (x_i - \bar{x})$
 $\hat{x}_i = \bar{x} + \underline{V_q z_i}$

1st lecture
 Goal's of PCA

\rightarrow Minimize reconstruction error

$$\underset{V_q}{\text{argmin}} \sum_{i=1}^n \| (x_i - \bar{x}) - \underbrace{V_q V_q^T}_{\text{projection matrix}} (x_i - \bar{x}) \|_2^2$$

 \rightarrow Maximizing variance in the projected data

$$\sum_{d \times d} = \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T = \underbrace{X}_{d \times n}^T \underbrace{X}_{n \times d}$$

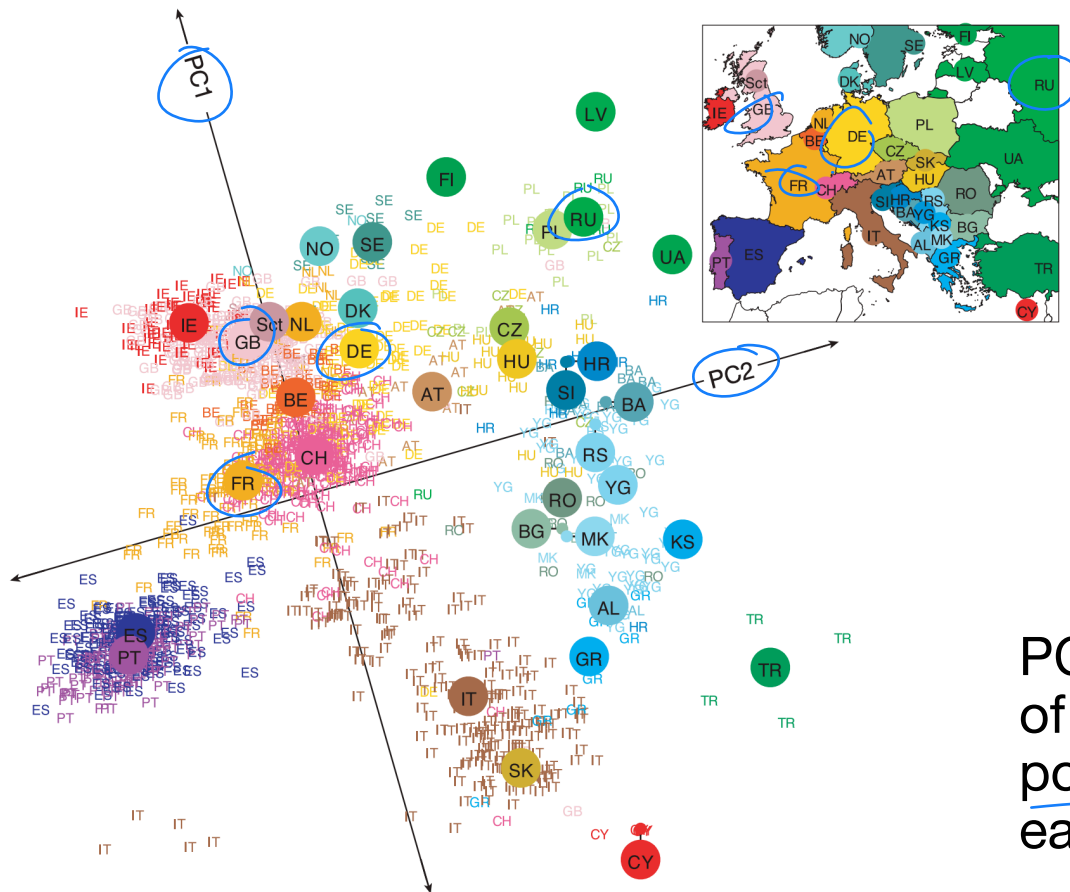
2nd lecture
 How?
 SVD

2nd lecture: How we do PCA \rightarrow SVD
do SVD on X

LETTERS

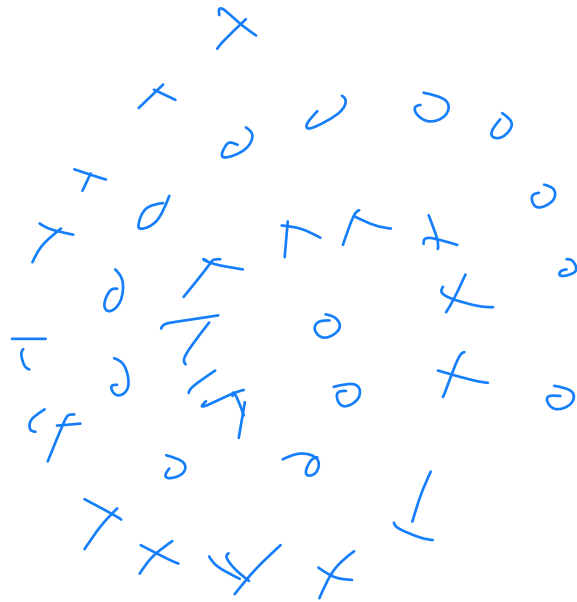
Genes mirror geography within Europe

John Novembre^{1,2}, Toby Johnson^{4,5,6}, Katarzyna Bryc⁷, Zoltán Kutalik^{4,6}, Adam R. Boyko⁷, Adam Auton⁷, Amit Indap⁷, Karen S. King⁸, Sven Bergmann^{4,6}, Matthew R. Nelson⁸, Matthew Stephens^{2,3} & Carlos D. Bustamante⁷



PCA on dataset consisting of 200,000 single nucleotide polymorphisms (SNPs) for each of ~1,400 individuals.

What is data we see?



→ PCA can't capture this non linear structure

Kernel PCA

- Assuming data is centered (i.e., zero-mean), PCA is defined by the eigenvectors corresponding to the largest eigenvalues of the covariance matrix

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

- To apply PCA to a non-linear feature mapping $\phi: \mathbb{R}^d \rightarrow \mathbb{R}^p$, we want to use SVD on:

$$\frac{1}{n} \phi(X)^T \phi(X) = \frac{1}{n} \sum_{i=1}^n \phi(x_i) \phi(x_i)^T$$

SVD would be prohibitively expensive

- However, this is a $p \times p$ matrix, and the dimension p can be very large.
- Hence, **Kernel PCA** attempts to make this efficient by considering only the subspaces spanned by the data, i.e., $v_j = \sum_{i=1}^n \phi(x_i) \alpha_{i,j}$, now parametrized by

$$\alpha^{(j)} = (\alpha_{1,j}, \dots, \alpha_{n,j})$$

Kernel PCA

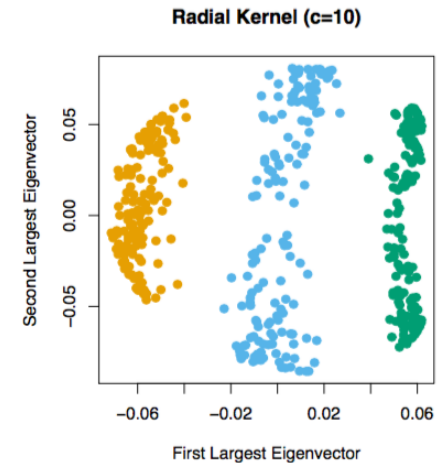
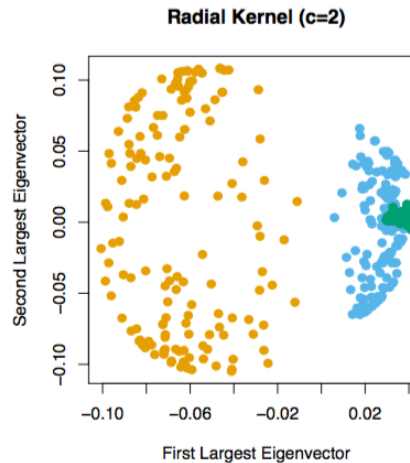
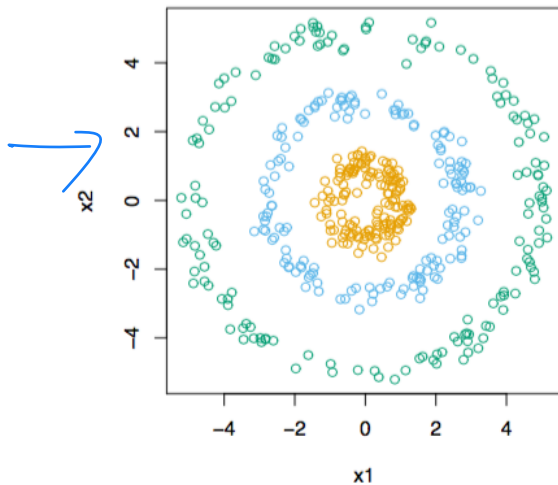
- Hence, **Kernel PCA** attempts to maximize

$$\max_{\alpha \in \mathbb{R}^{n \times r}} \sum_{j=1}^r \underbrace{(\alpha_j^T \phi(X))}_{v_j^T} \phi(X)^T \phi(X) \underbrace{(\phi(X)^T \alpha_j)}_{v_j}$$

- This is equivalent to using the eigenvectors

$$n \lambda_j \underline{K} \alpha_j = \underline{K}^2 \alpha_j$$

$K \in \mathbb{R}^{n \times n}$
 $k_{ij} = \phi(x_i)^T \phi(x_j)$
 $k(x_i, x_j) = e^{-c \|x_i - x_j\|^2}$



Matrix completion

2005 netflix challenge

NETFLIX






Given historical data on how users rated movies in past:

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

(Sparsity: 1.2%)

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

18k

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

low rank
sparse matrix

extract clustering / low rank representation of movies to determine which ones are similar
↳ genres

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

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

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

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

$$= \underline{U}a_i \approx \bigvee_{q=1}^r z_i$$

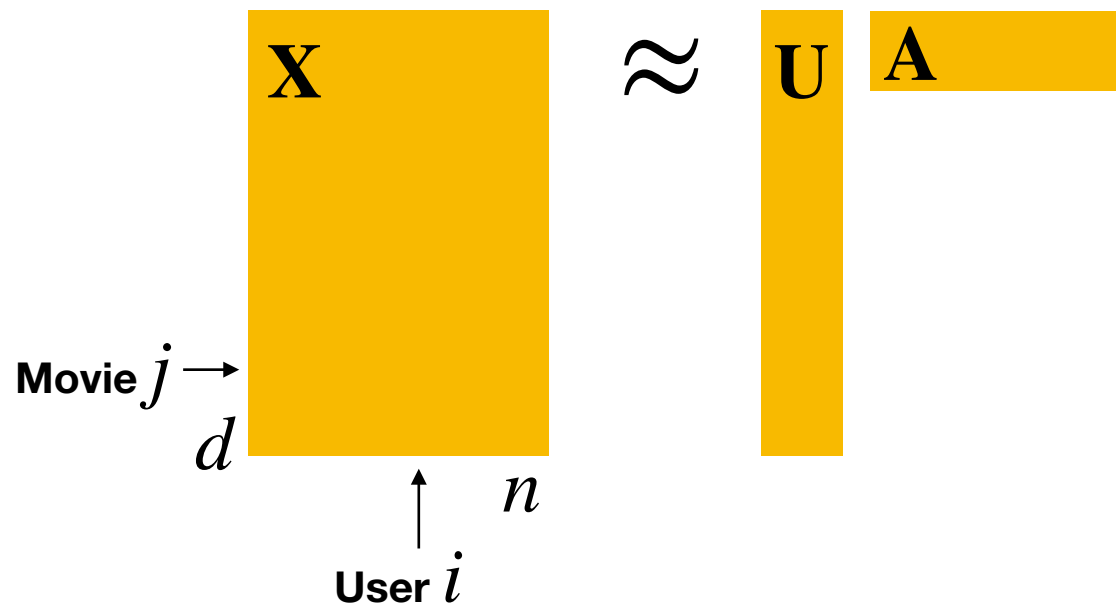
for some lower-dimensional $a_i \in \mathbb{R}^r$ for i -th user and some $r \ll d$

- 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

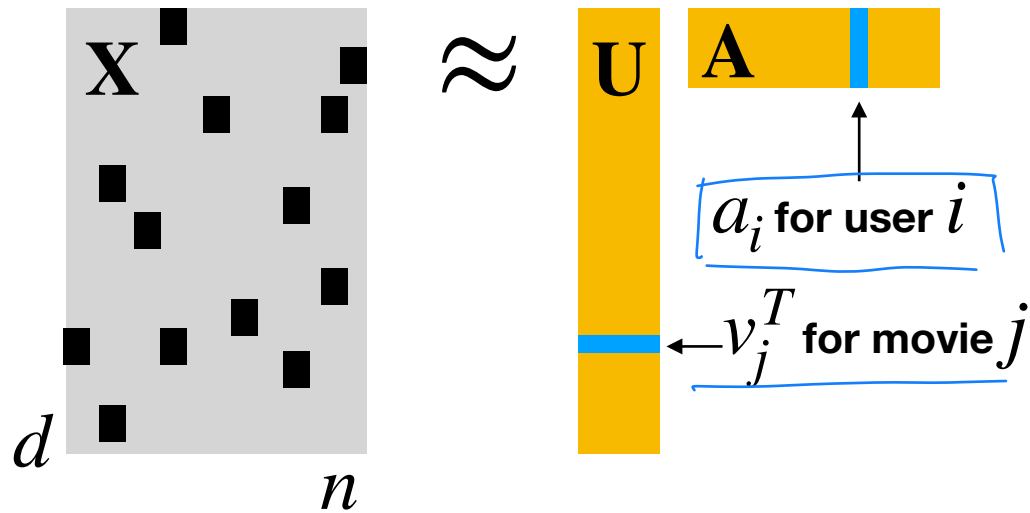
- let $\mathbf{X} = [x_1 \ x_2 \ \cdots \ x_n] \in \mathbb{R}^{d \times n}$ be the ratings matrix, and assume it is fully observed, i.e. we know all the entries
- then we want to find $\mathbf{U} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} = [a_1 \ a_2 \ \cdots \ a_n] \in \mathbb{R}^{r \times n}$ that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



- 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_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. 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

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

Handwritten annotations: "modify" with an arrow pointing to v_j , "use" with an arrow pointing to a_i . The term $(i,j) \in S_{\text{train}}$ is underlined.

- 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]

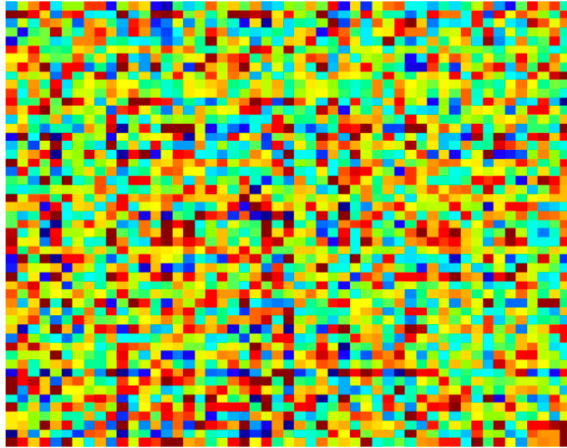
Assume the ground truths \mathbf{X} has rank r , then

(a variant of) gradient descent finds the optimal solution if we

observe more than $c r (d + n) \log(dn)$ entries at random positions

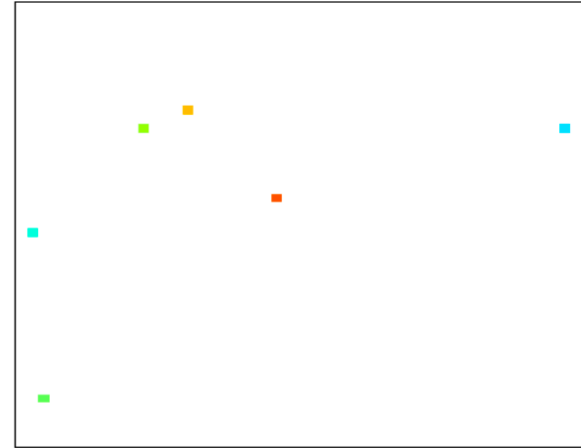
Example: 2000×2000 rank-8 random matrix

→ low-rank matrix \mathbf{X}

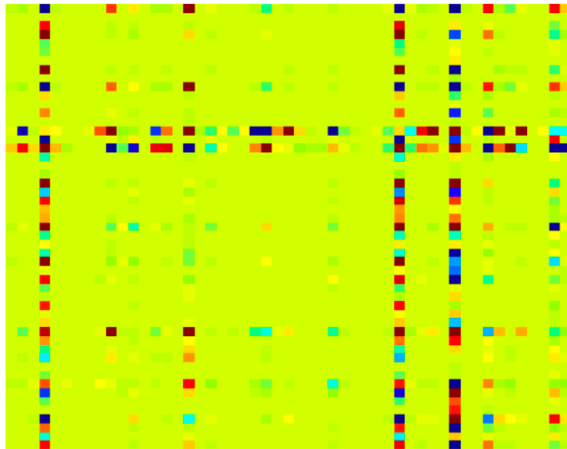


For illustration,
we zoom in to a
50x50 submatrix

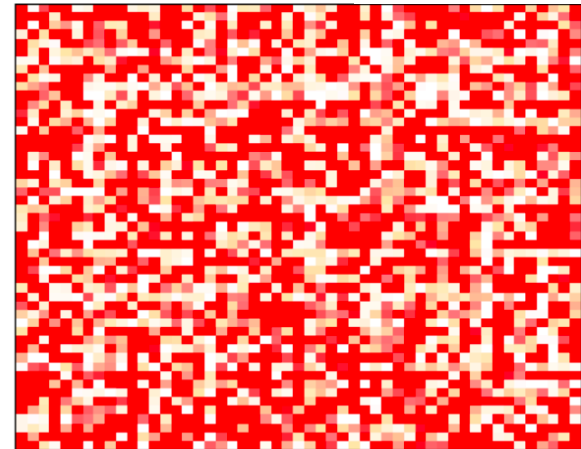
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



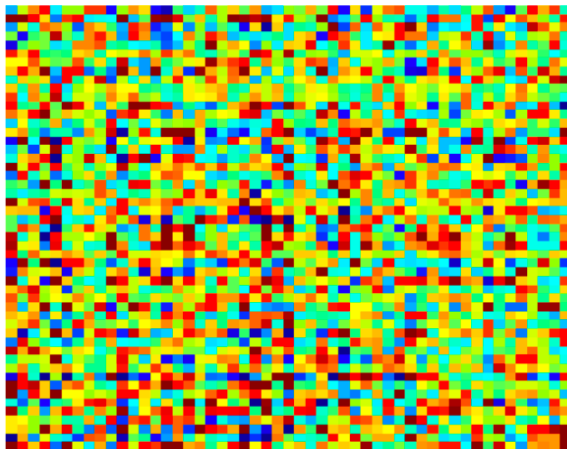
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



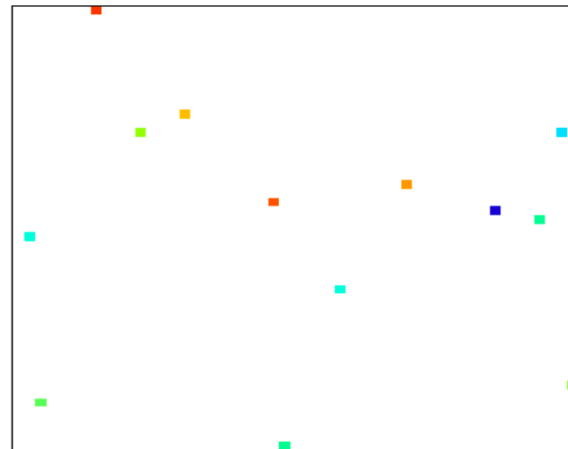
0.25% sampled

Example: 2000×2000 rank-8 random matrix

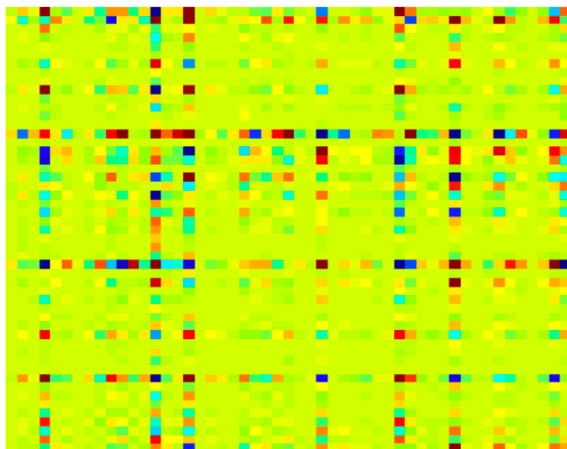
low-rank matrix \mathbf{X}



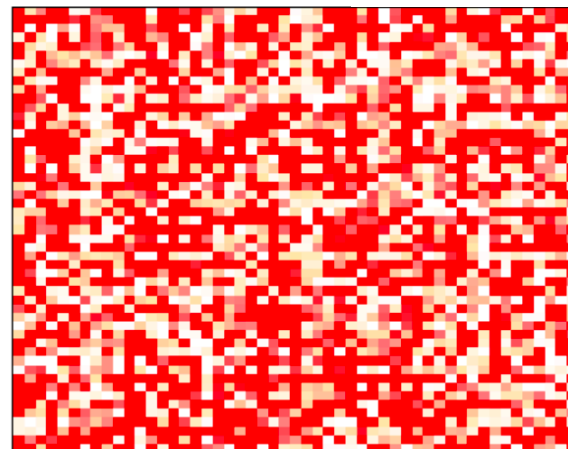
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



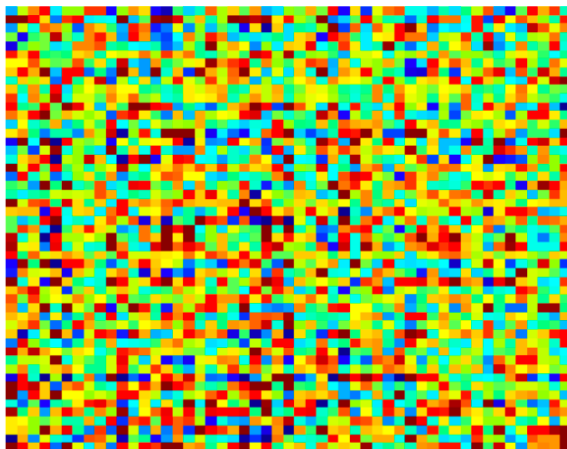
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



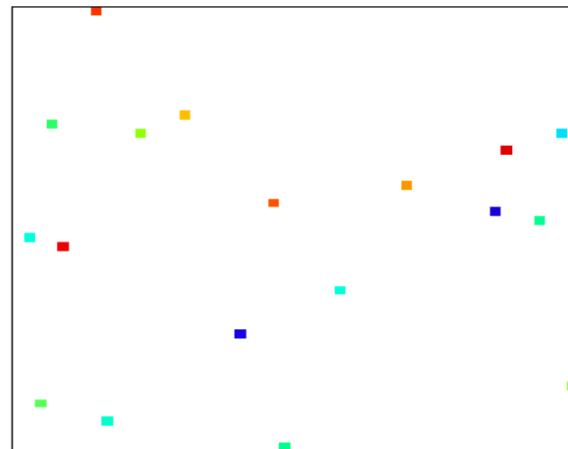
0.50% sampled

Example: 2000×2000 rank-8 random matrix

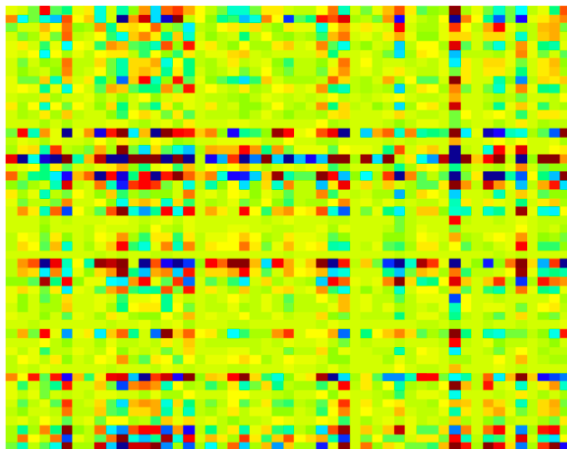
low-rank matrix \mathbf{X}



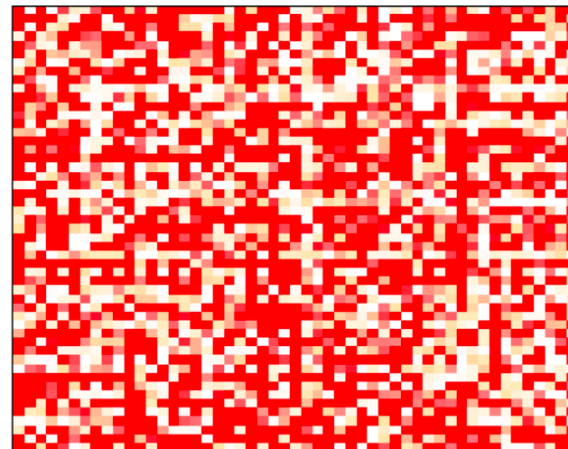
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



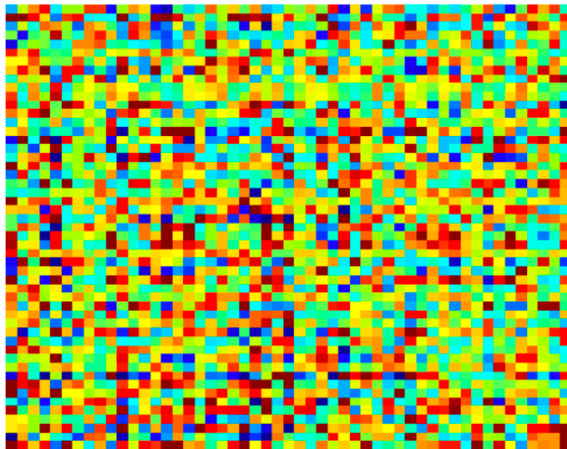
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



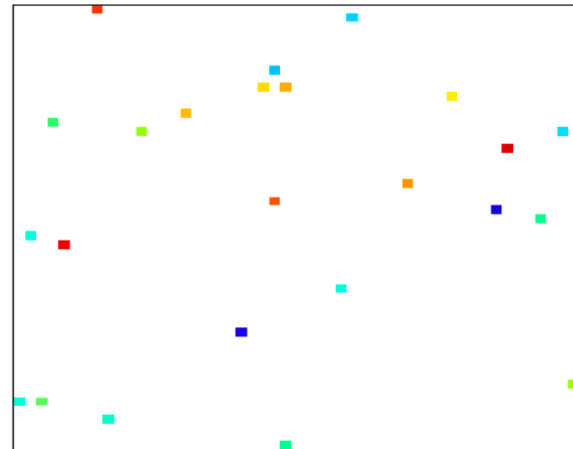
0.75% sampled

Example: 2000×2000 rank-8 random matrix

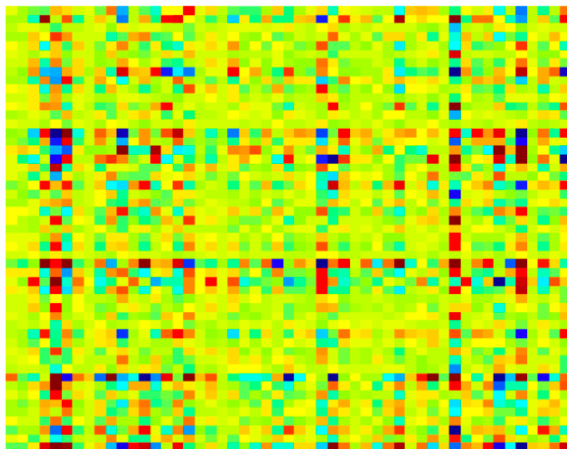
low-rank matrix \mathbf{X}



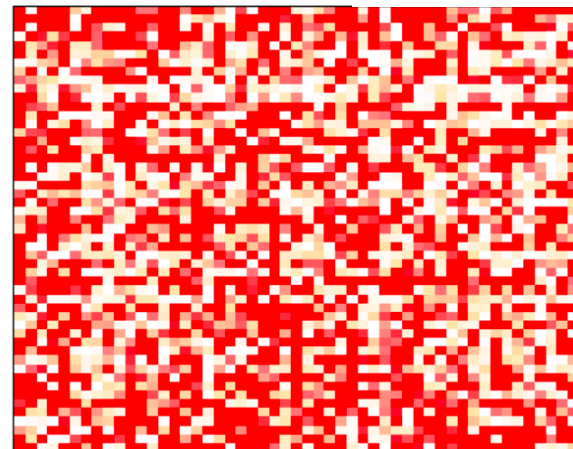
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



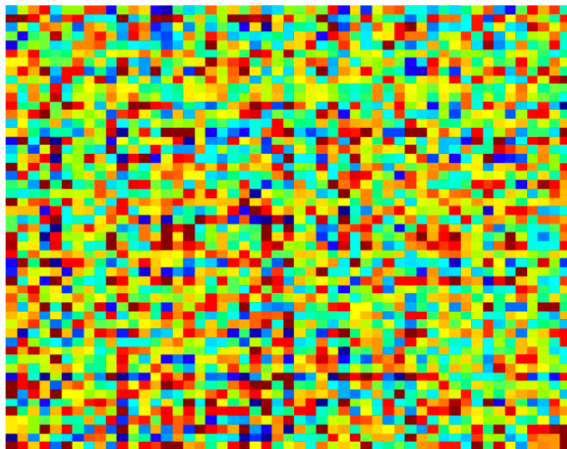
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



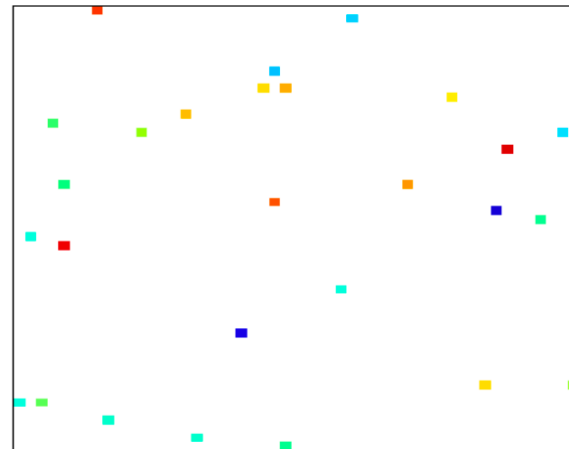
1.00% sampled

Example: 2000×2000 rank-8 random matrix

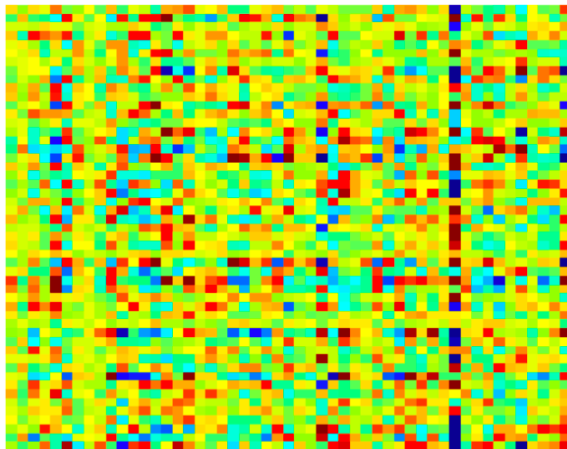
low-rank matrix \mathbf{X}



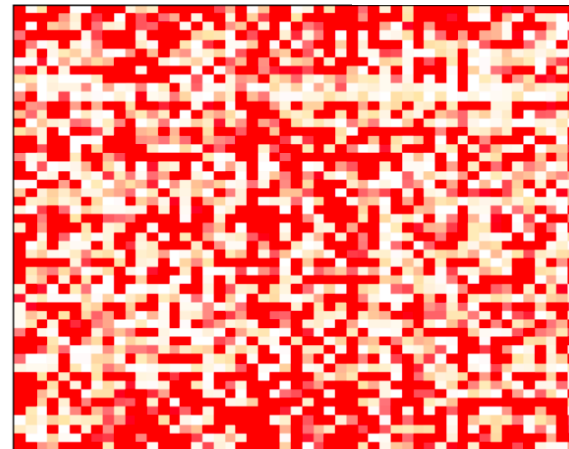
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



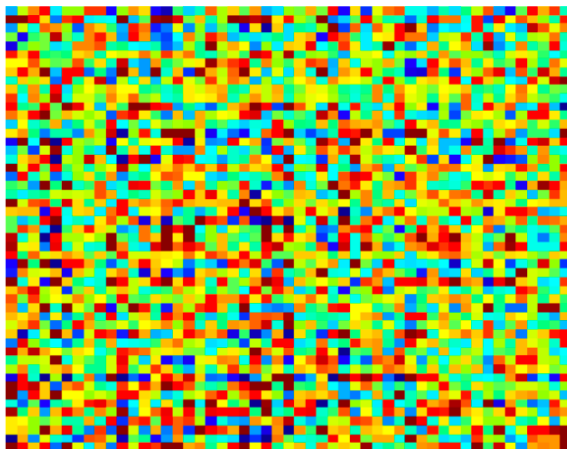
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



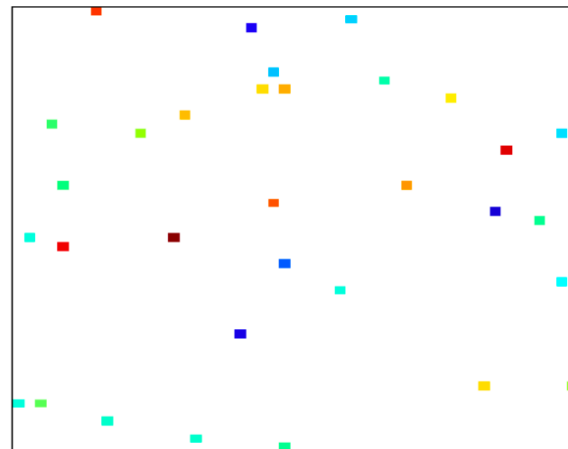
1.25% sampled

Example: 2000×2000 rank-8 random matrix

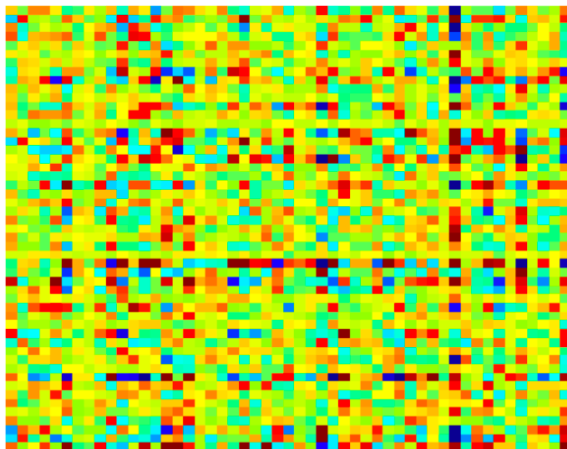
low-rank matrix \mathbf{X}



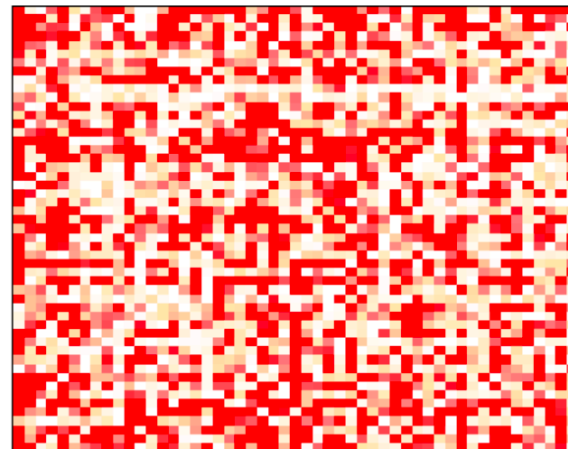
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



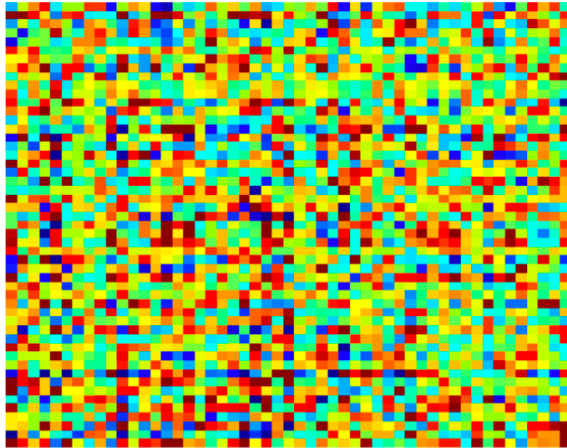
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



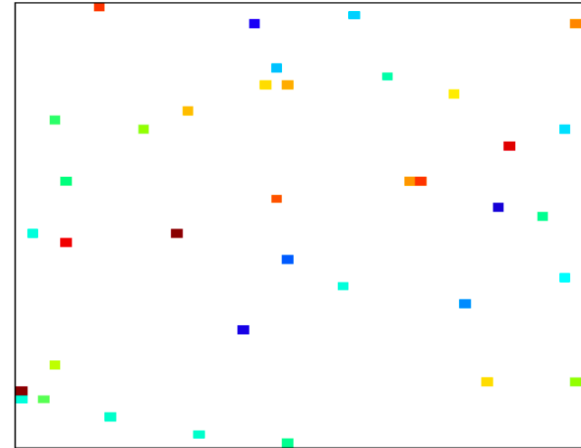
1.50% sampled

Example: 2000×2000 rank-8 random matrix

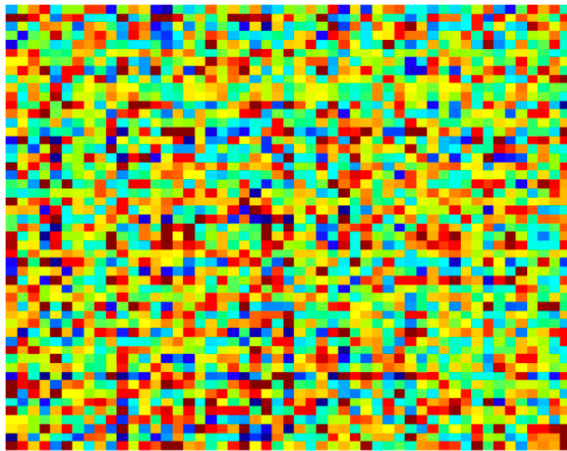
low-rank matrix \mathbf{X}



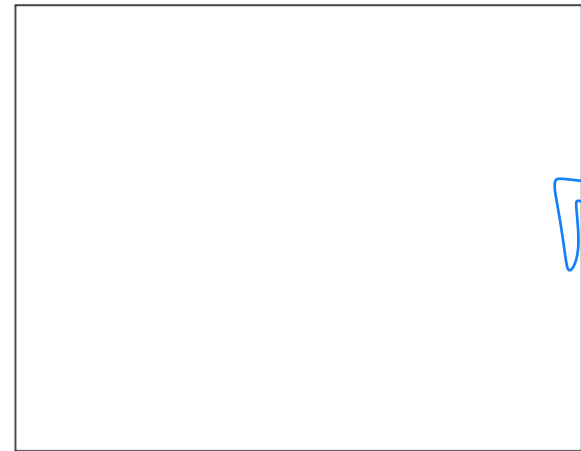
sampled matrix



Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T$



squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



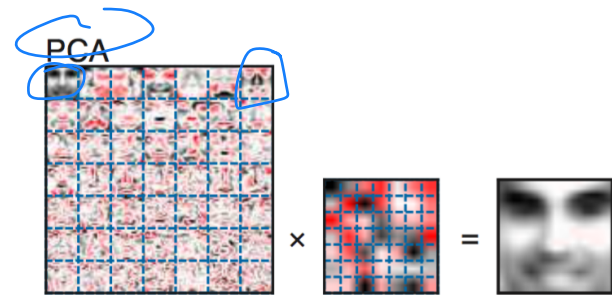
1.75% sampled

Other matrix factorizations

$$\mathbf{X} = \mathbf{U} \mathbf{S} \mathbf{V}^T$$

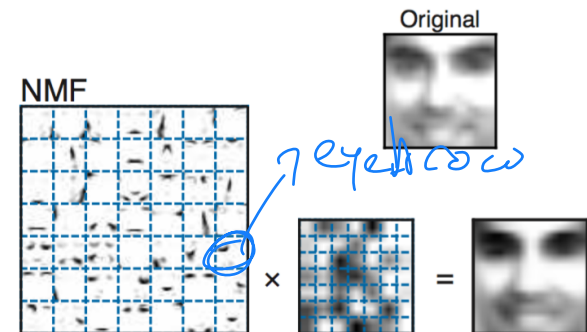
Singular value decomposition

Elements of \mathbf{U} , \mathbf{S} , \mathbf{V} in \mathbb{R}



Nonnegative matrix factorization (NMF)

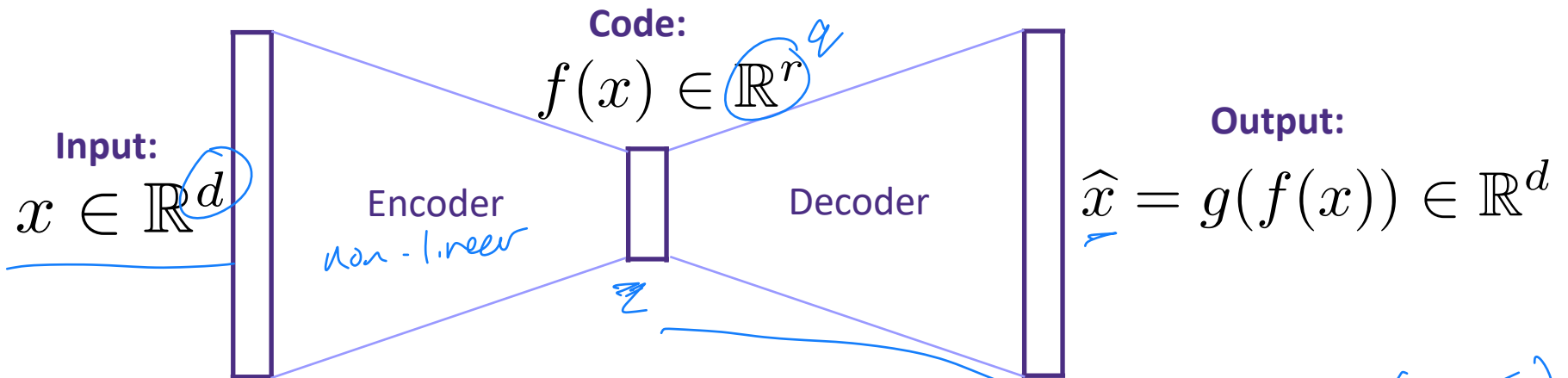
Elements of \mathbf{U} , \mathbf{S} , \mathbf{V} in \mathbb{R}_+



interpretable components

Autoencoders

Find a low dimensional representation for your data by predicting your data

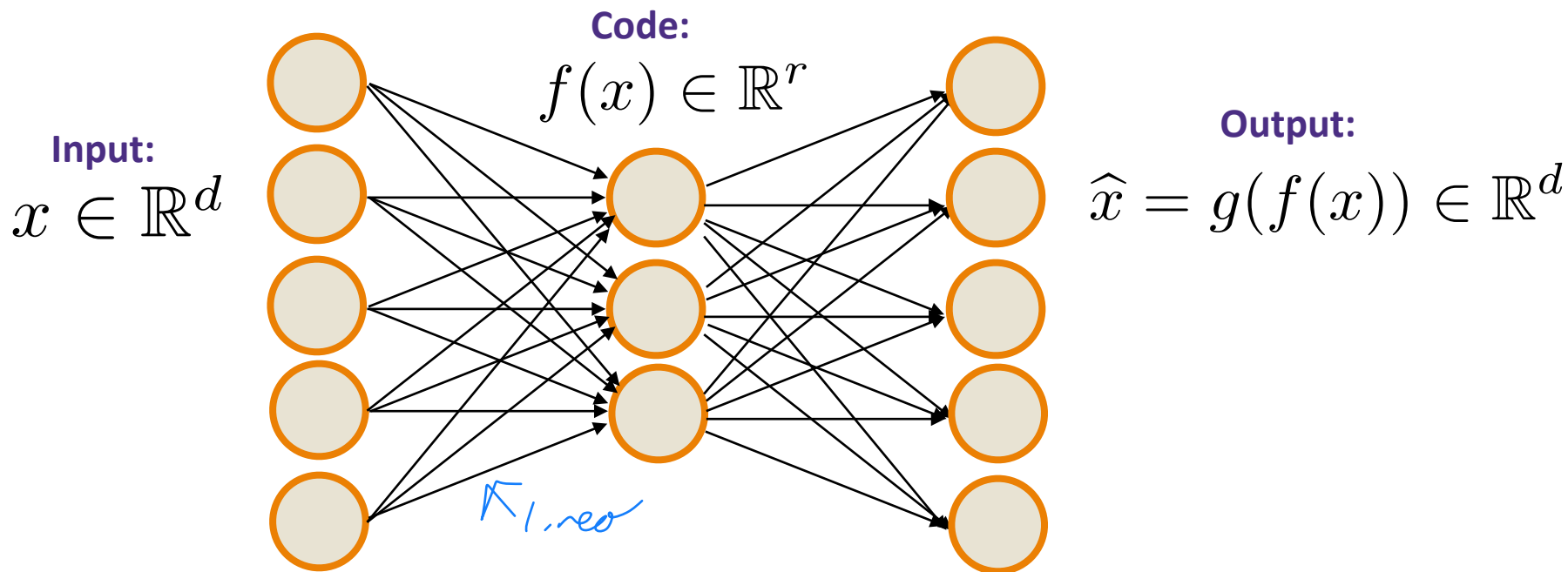


$$\text{minimize}_{f,g} \sum_{i=1}^n \|x_i - g(f(x_i))\|_2^2$$

$z \sim N(0, I)$
KL . regularizer, zahn
frisch

Variational
Autoencoder
(VAE)

Autoencoders



$$\underset{f, g}{\text{minimize}} \sum_{i=1}^n \|x_i - g(f(x_i))\|_2^2$$

What if $f(X) = Ax$ and $g(y) = By$?

\Rightarrow just PCA