Lecture 15
Singular Value Decomposition

- reducing the dimension of the data and visualization

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

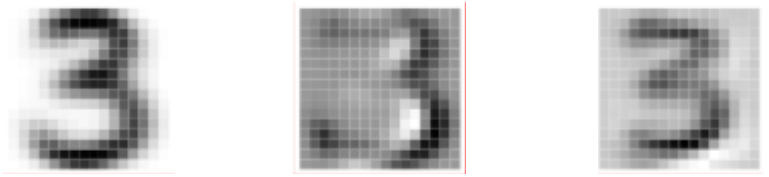
$$a_i = \mathbf{U}^T(x_i - \bar{x}) \in \mathbb{R}^r \text{ and}$$

each data point approximated by

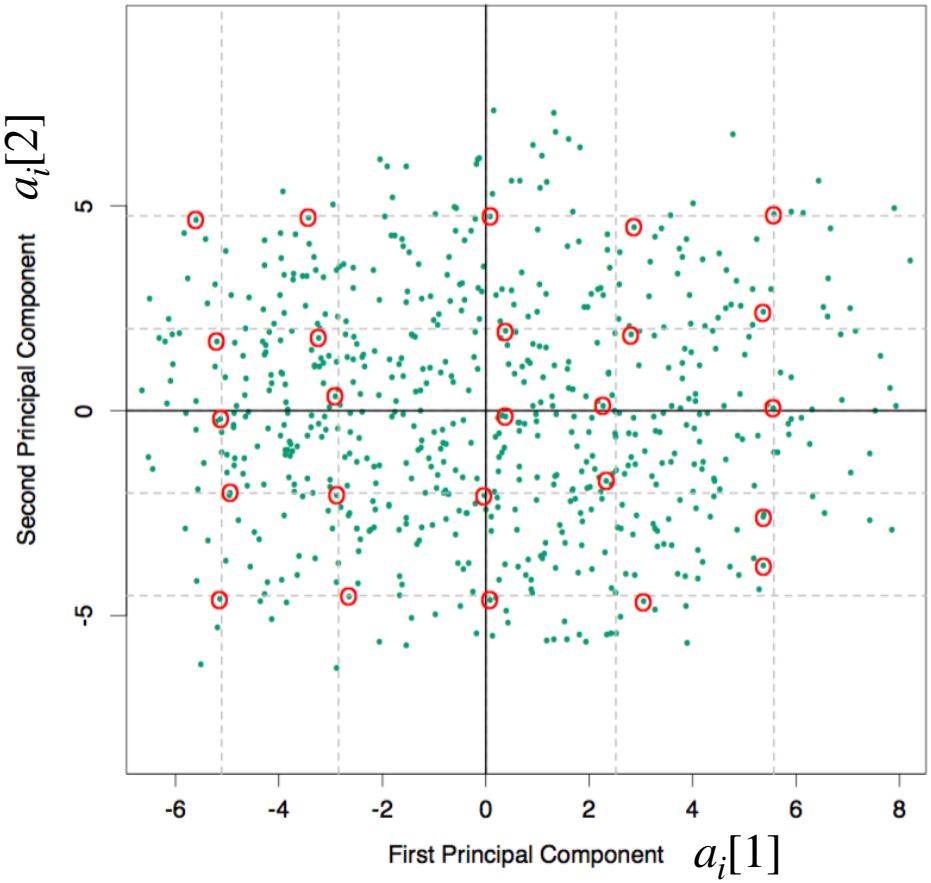
$$p_i = \bar{x} + U\, a_i$$

PCA use-cases that uncover hidden structures

- Consider a dataset of handwritten 3's
- Each one is 16x16 pixels such that $x_i \in \mathbb{R}^{256}$
- Using PCA on this dataset, we get



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

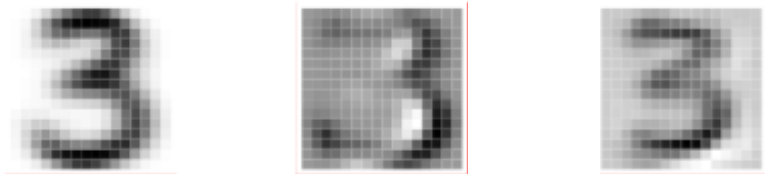


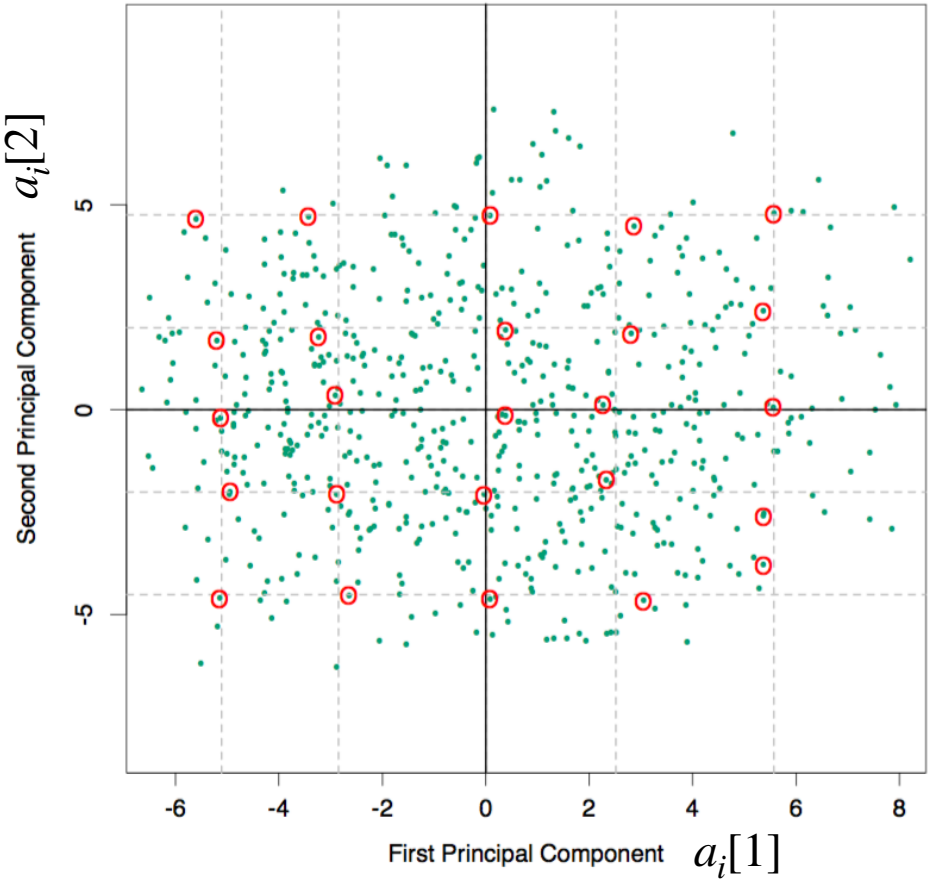
- What does u_1 represent?
- What does u_2 represent?

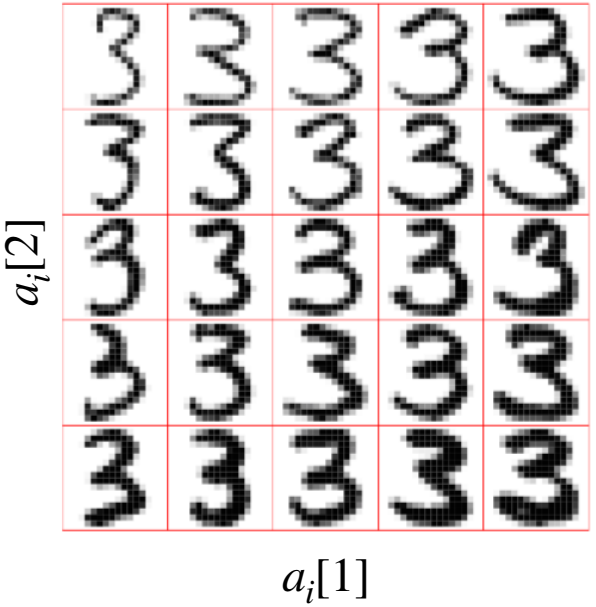
PCA use-cases that uncover hidden structures-cas

- Consider a dataset of handwritten 3's
- Each one is 16x16 pixels such that $x_i \in \mathbb{R}^{256}$
- Using PCA on this dataset, we get

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

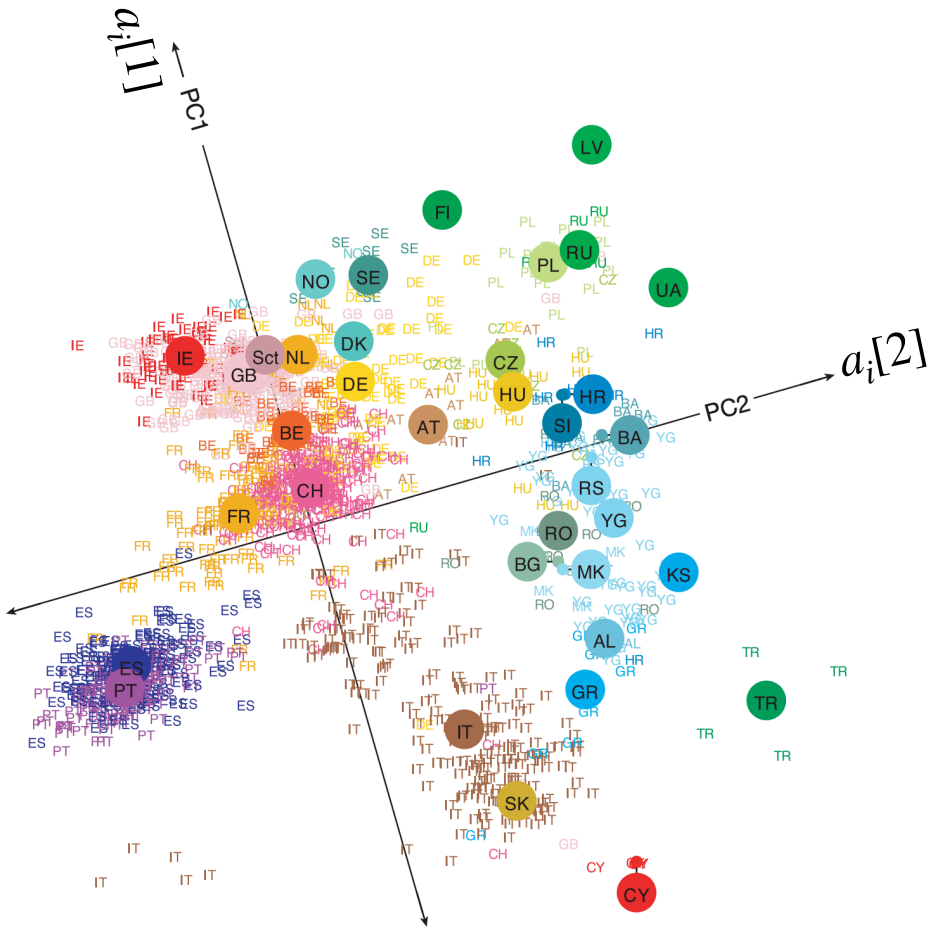






- What does u_1 represent?
- What does u_2 represent?
- Why did PCA find these?

PCA use-cases that uncover hidden structures



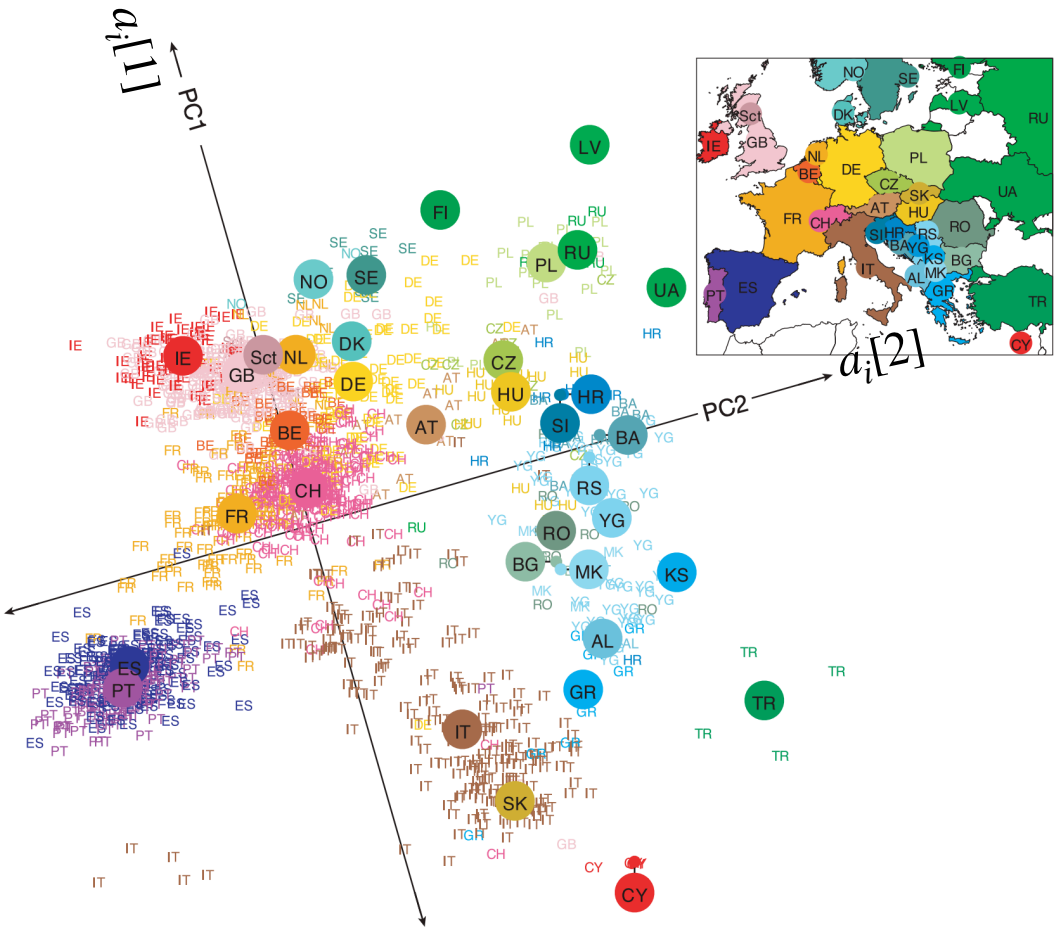
- Dataset consisting of 200,000 single nucleotide polymorphisms (SNPs)
- for $n \simeq 1400$ individuals
- PCA gives $x_i = \bar{x} + a_i[1] \cdot u_1 + a_i[2] \cdot u_2$
- each individual represented in this 2-d plot
- What does the color represent?

nature Vol 456|6 November 2008|doi:10.1038/nature07331

LETTERS

Genes mirror geography within Europe

John Novembre[1,2], Toby Johnson[4,5,6], Katarzyna Bryc[7], Zoltán Kutalik[4,6], Adam R. Boyko[7], Adam Auton[7], Amit Indap[7], Karen S. King[8], Sven Bergmann[4,6], Matthew R. Nelson[8], Matthew Stephens[2,3] & Carlos D. Bustamante[7]



- Dataset consisting of 200,000 single nucleotide polymorphisms (SNPs)
- for $n \simeq 1400$ individuals
- PCA gives $x_i = \bar{x} + a_i[1] \cdot u_1 + a_i[2] \cdot u_2$
- each individual represented in this 2-d plot
- What does the color represent?
- Why did PCA find these?

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:

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

- 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., $u_j = \sum_{i=1}^{n} \phi(x_i) b_{i,j} = \phi(X)^T b_j$,

  now parametrized by $b_j = (b_{1,j}, \ldots, b_{n,j})$

Kernel PCA

- Hence, plugging in $u_j = \phi(X)^T b_j$, ***Kernel PCA*** attempts to maximize

$$\max_{U=[u_1,\ldots,u_r]\in\mathbb{R}^{p\times r}} \sum_{j=1}^{r} u_j^T \phi(X)^T \phi(X) u_j = \max_{B=[b_1,\ldots,b_r]\in\mathbb{R}^{n\times r}} \sum_{j=1}^{r} \underbrace{(b_j^T \phi(X))}_{u_j^T} \phi(X)^T \phi(X) \underbrace{(\phi(X)^T b_j)}_{u_j}$$

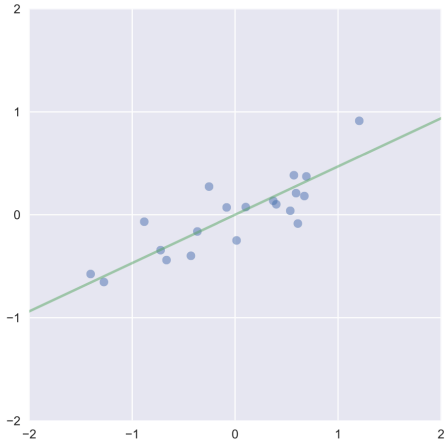
$$\text{s.t.} \quad U^T U = \mathbf{I}_{r\times r} \qquad\qquad \text{s.t.} \quad B^T \phi(X)\phi(X)^T B = \mathbf{I}_{r\times r}$$

- Define the kernel matrix $K = \phi(X)\phi(X)^T$, as usual, and suppose $r = 1$, then

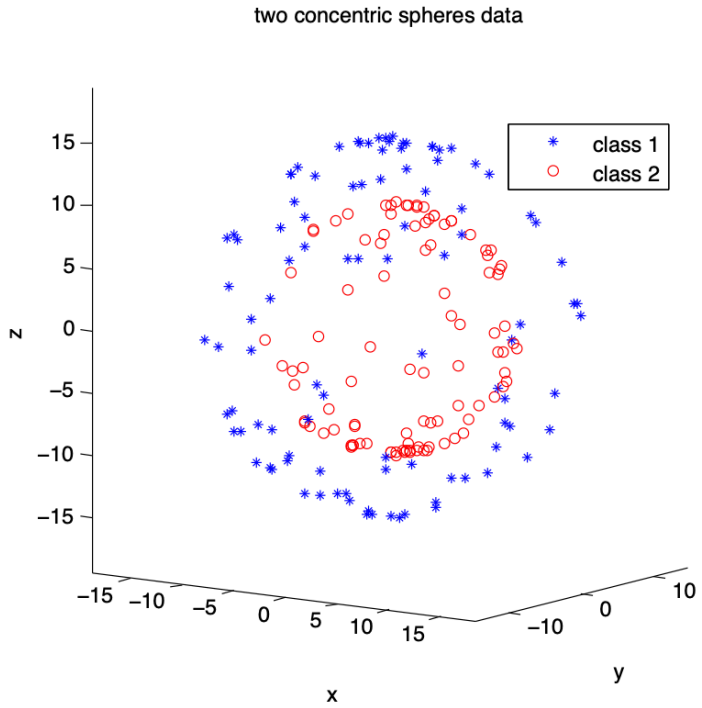
- The solution of this maximization is equivalent to the top eigenvectors of the kernel matrix K, known as **Kernel PCA**

Kernel PCA

- PCA can only capture linear relations



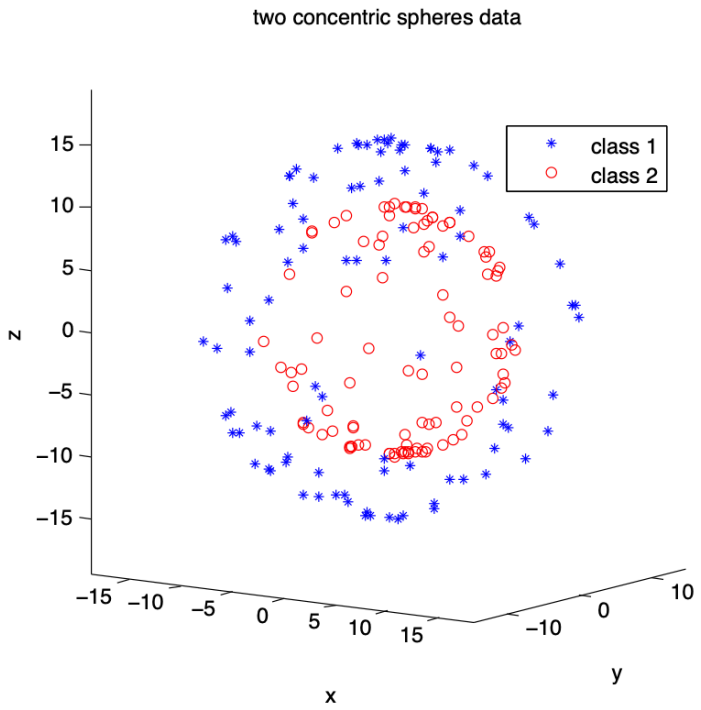
- In comparison, similar to the Kernel regression case, ***Kernel PCA*** can capture non-linear relations

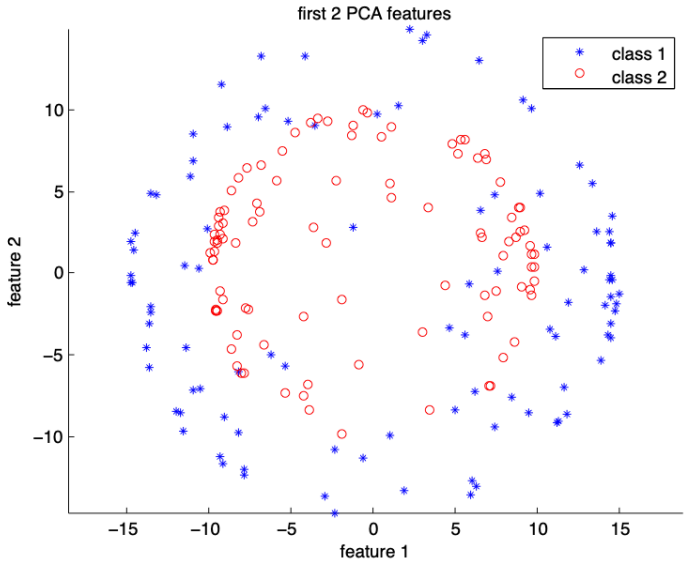


- What does 2-d PCA projection of the data points look like?

Kernel PCA

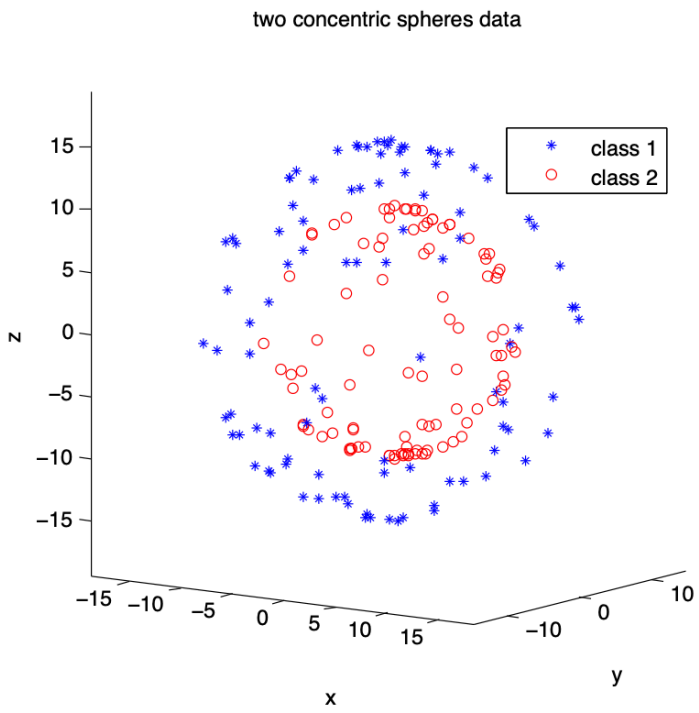
- In comparison, similar to the Kernel regression case, ***Kernel PCA*** can capture non-linear relations



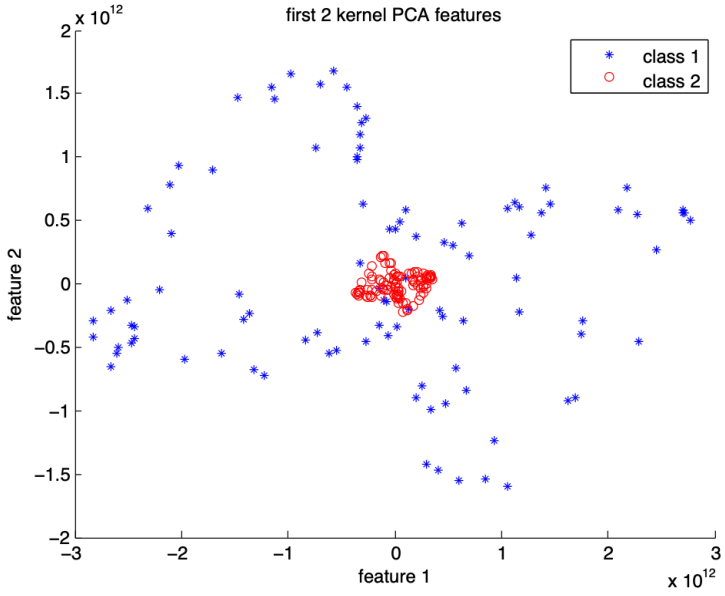


Kernel PCA

- In comparison, similar to the Kernel regression case, ***Kernel PCA*** can capture non-linear relations

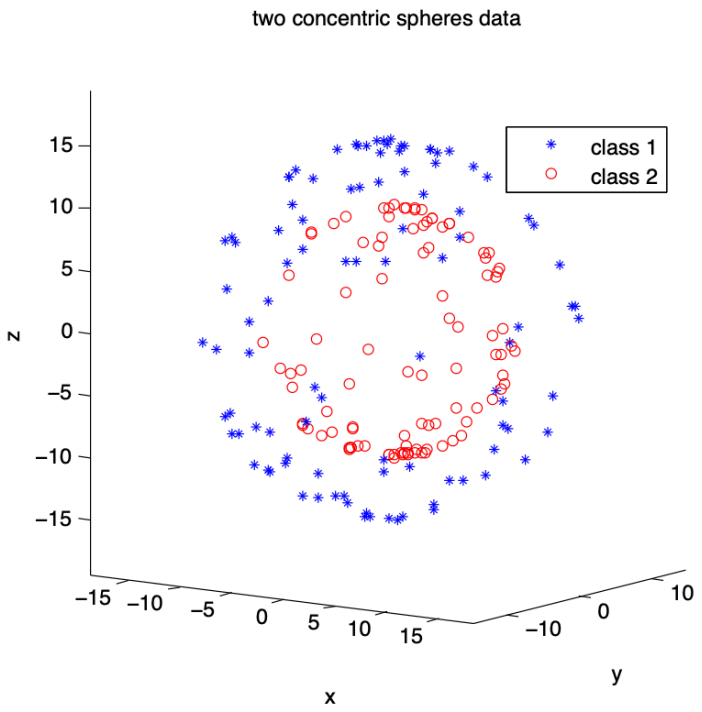


- Polynomial kernel with degree-5

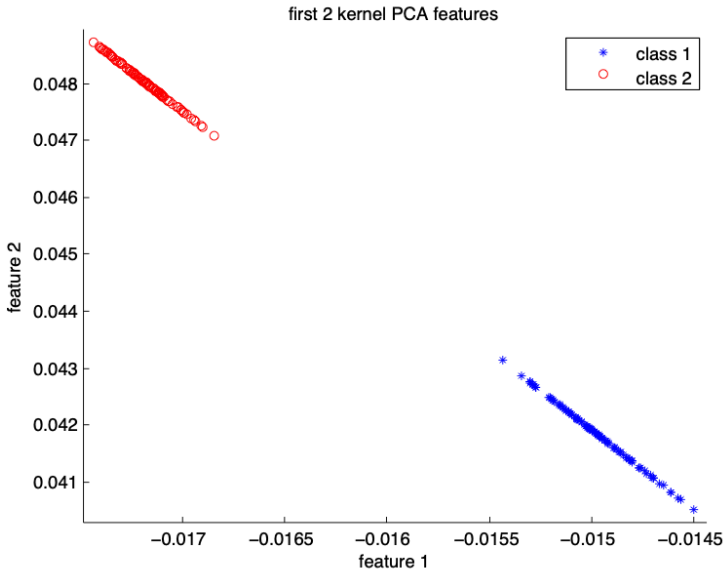


Kernel PCA

- In comparison, similar to the Kernel regression case, ***Kernel PCA*** can capture non-linear relations



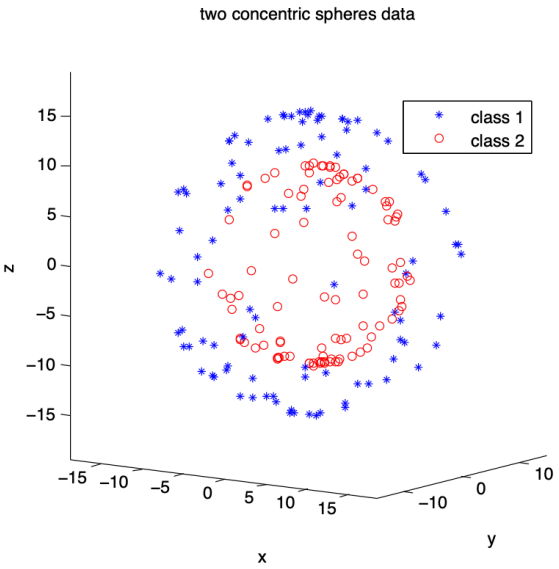
- RBF kernel with $\sigma = 20$

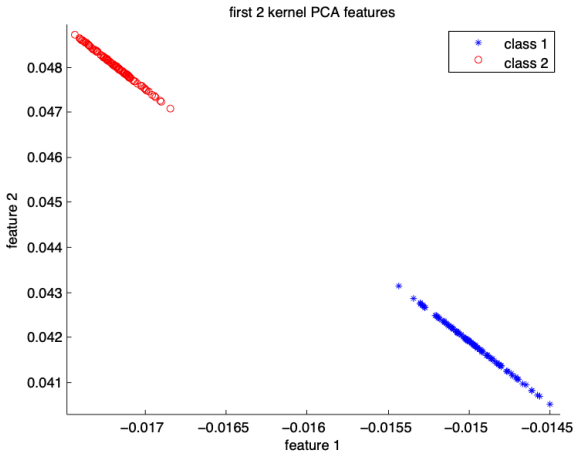


Kernel PCA

- In comparison, similar to the Kernel regression case, ***Kernel PCA*** can capture non-linear relations

- RBF kernel with $\sigma = 20$





- Kernel matrix $K \Longrightarrow$ Eigenvectors b_1, b_2

Example of denoising with PCA

Original data

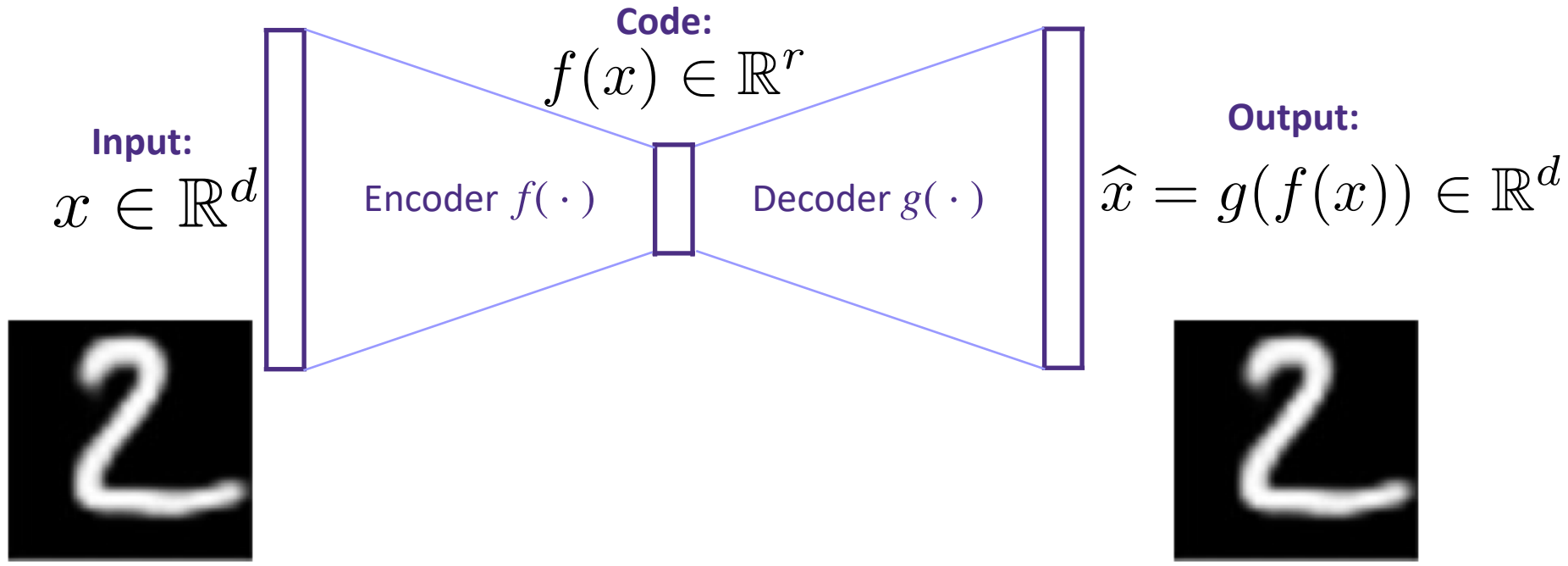
Data corrupted with Gaussian noise

Result after linear PCA

Result after kernel PCA, Gaussian kernel

Autoencoders

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

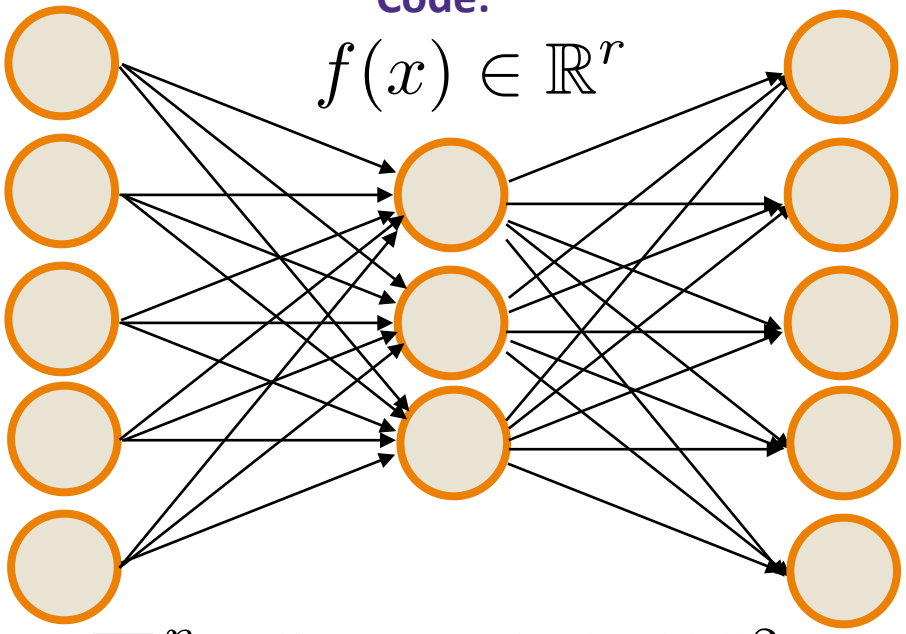


$$\underset{f,g}{\text{minimize}} \sum_{i=1}^{n} \|x_i - g(f(x_i))\|_2^2$$

Autoencoders

Input:
$x \in \mathbb{R}^d$

Code:
$f(x) \in \mathbb{R}^r$

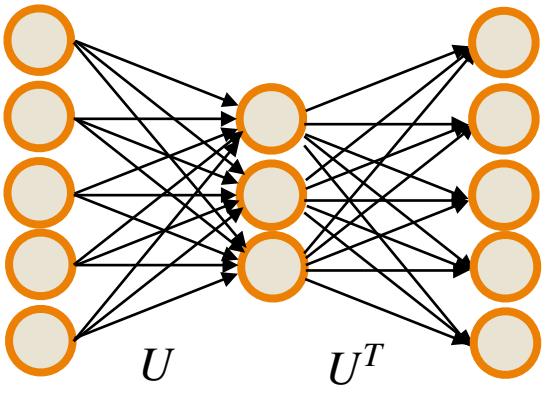


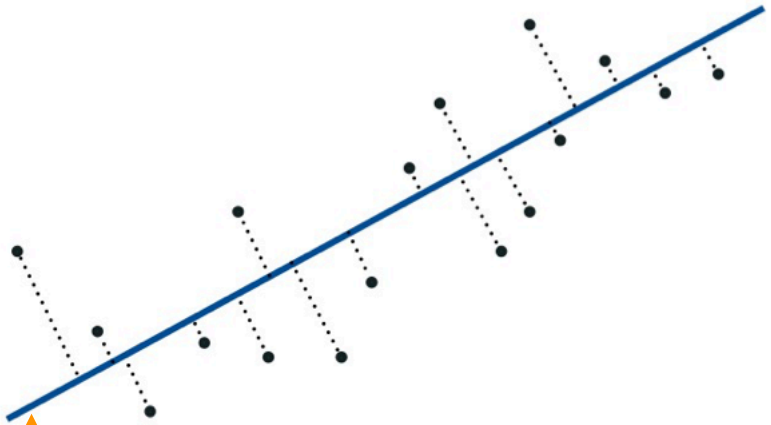
Output:
$\widehat{x} = g(f(x)) \in \mathbb{R}^d$

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

What if $f(x) = U^T x$ and $g(a) = Ua$?

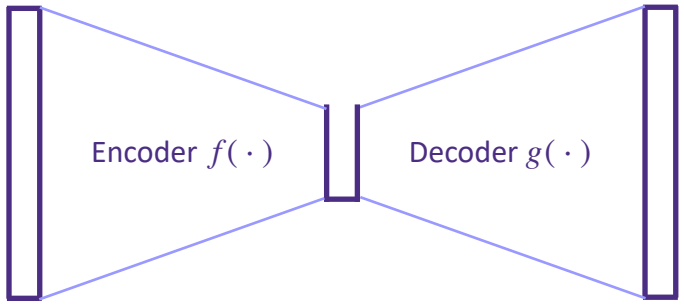
- PCA

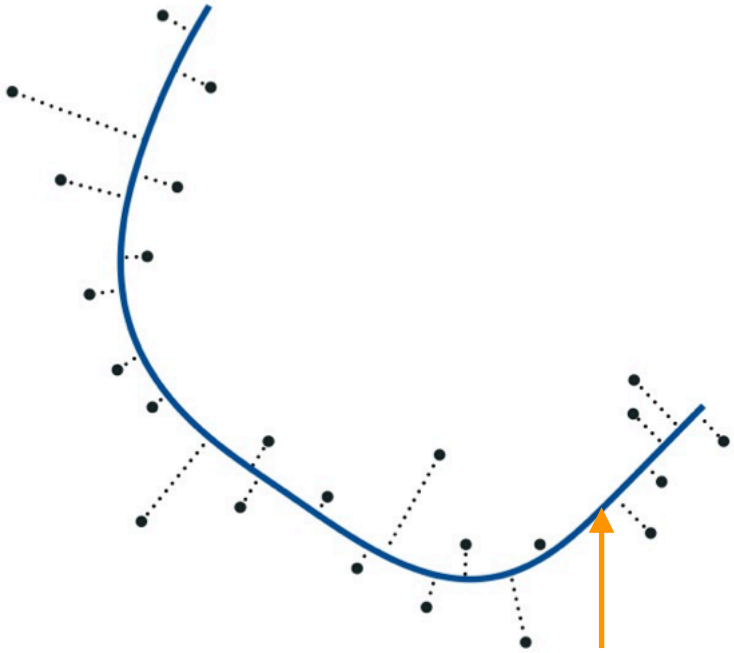




low-dimensional hyperplane
defined by the span of U

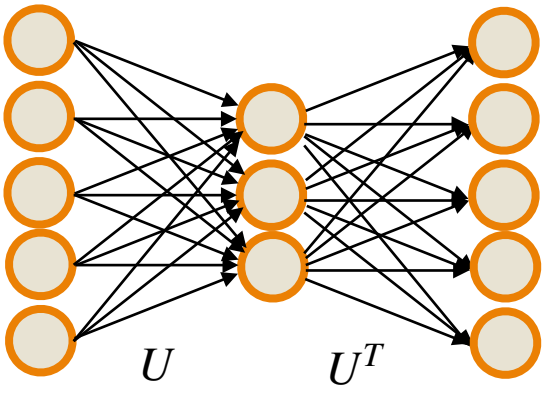
- Autoencoder

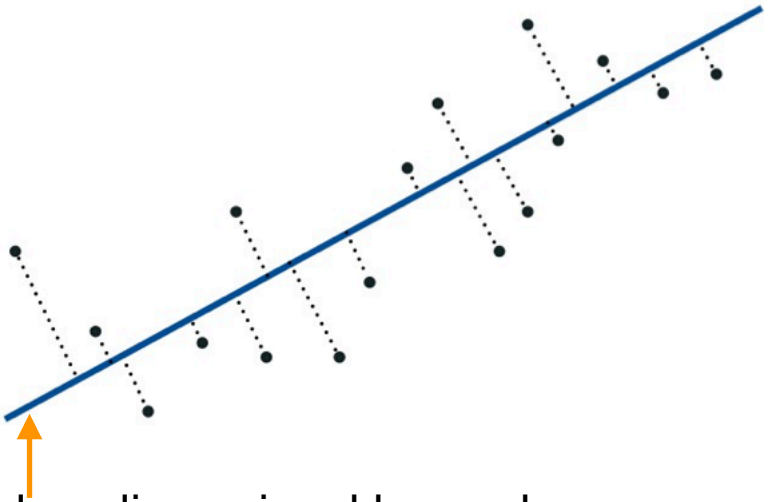




low-dimensional manifold
defined by the range of $f(\cdot)$

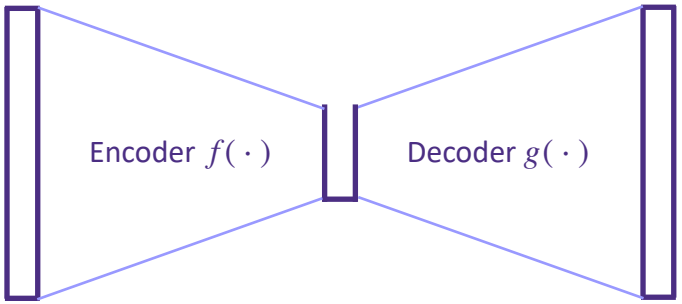
- PCA

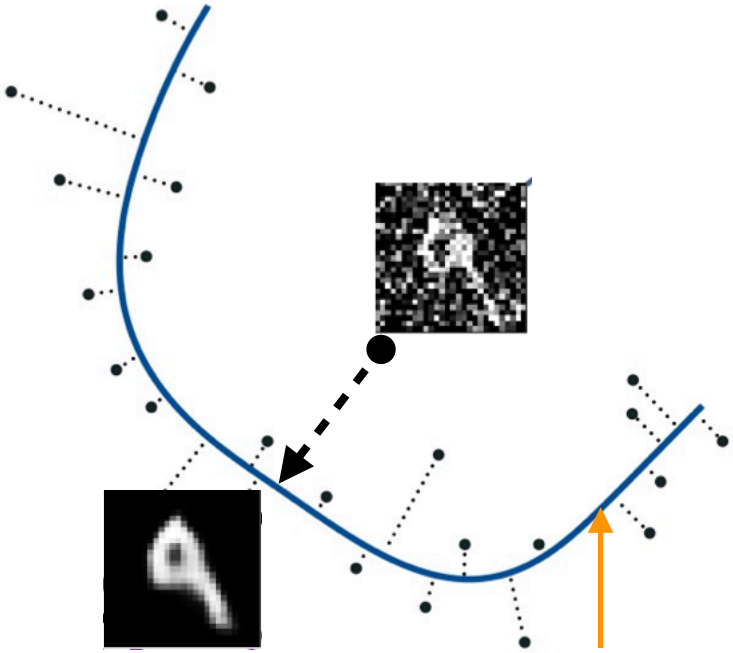




low-dimensional hyperplane
defined by the span of U

- Autoencoder





low-dimensional manifold
defined by the range of $f(\cdot)$

Warm-up: Eigenvalue decomposition

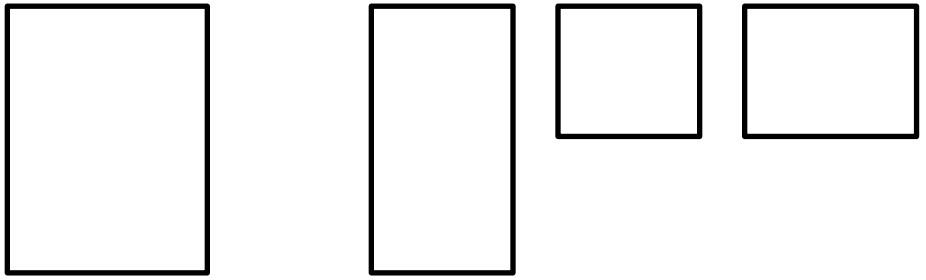
Suppose that a square matrix $A \in \mathbb{R}^{n \times n}$, has
a set of right eigen-vectors $\{v_i \in \mathbb{R}^n\}_{i=1}^n$ and corresponding eigenvalues $\{\lambda_i\}_{i=1}^n$

- $A\, v_1 \;=$
- $A\, [v_1, v_2] \;=$
- Let $V = [v_1, v_2, \ldots, v_n]$, then $A\, V \;=$
- Eigen value decomposition of A is $A \;=$

- Now, suppose A is symmetric, then its eigenvectors are orthonormal, i.e., $V^T V = V V^T = I$, which is equivalent to saying that $v_i^T v_j =$

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 $V^T V = I$.



What is $A^T A v_i =$

$AA^T =$

What is $AA^T u_i =$

$A^T A =$

- 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 $V^T V = I$.

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

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

$\mathbf{V}$ 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

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

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

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

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

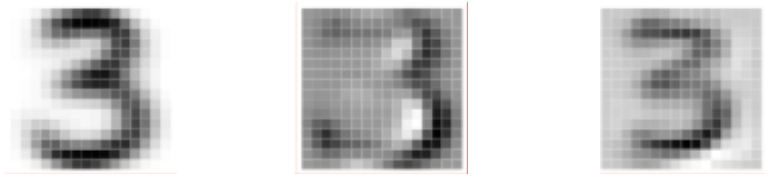
$$\text{maximize}_{Q \in \mathbb{R}^{n \times r}} \quad \text{Trace}(Q^T A^T A Q) = \sum_{j=1}^{r} Q_j^T A^T A Q_j$$

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

  - The optimal rank-r subspace is $Q = V_{1:r}$

PCA use-case

- Consider a dataset of handwritten 3's
- Each one is 16x16 pixels such that $x_i \in \mathbb{R}^{256}$
- Using PCA on this dataset, we get $x_i = \bar{x} + a_i[1] \cdot u_1 + a_i[2] \cdot u_2$



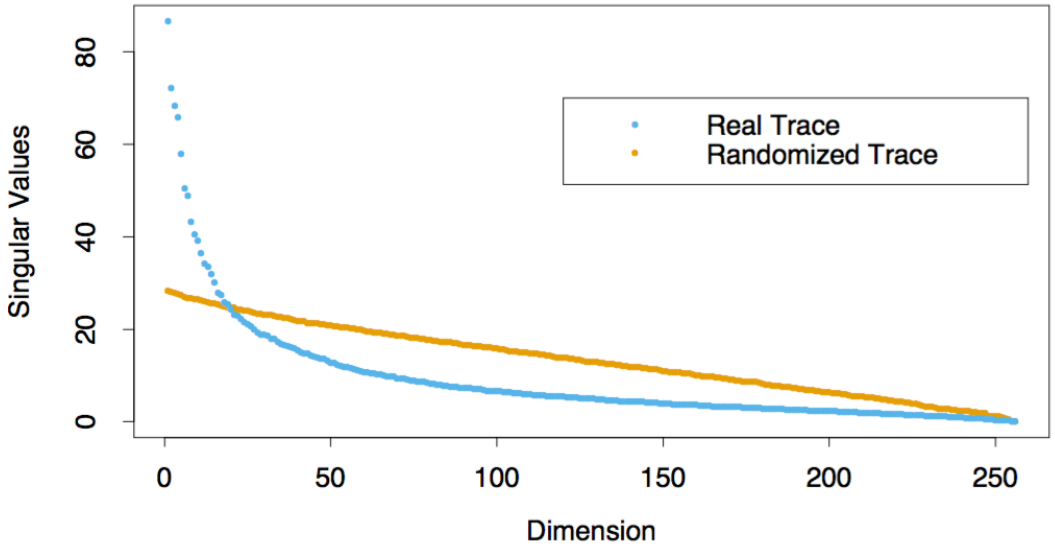


FIGURE 14.24. *The 256 singular values for the digitized threes, compared to those for a randomized version of the data (each column of* $\mathbf{X}$ *was scrambled).*

Matrix completion

Given historical data on how users rated movies in past:

NETFLIX

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)

Matrix completion problem

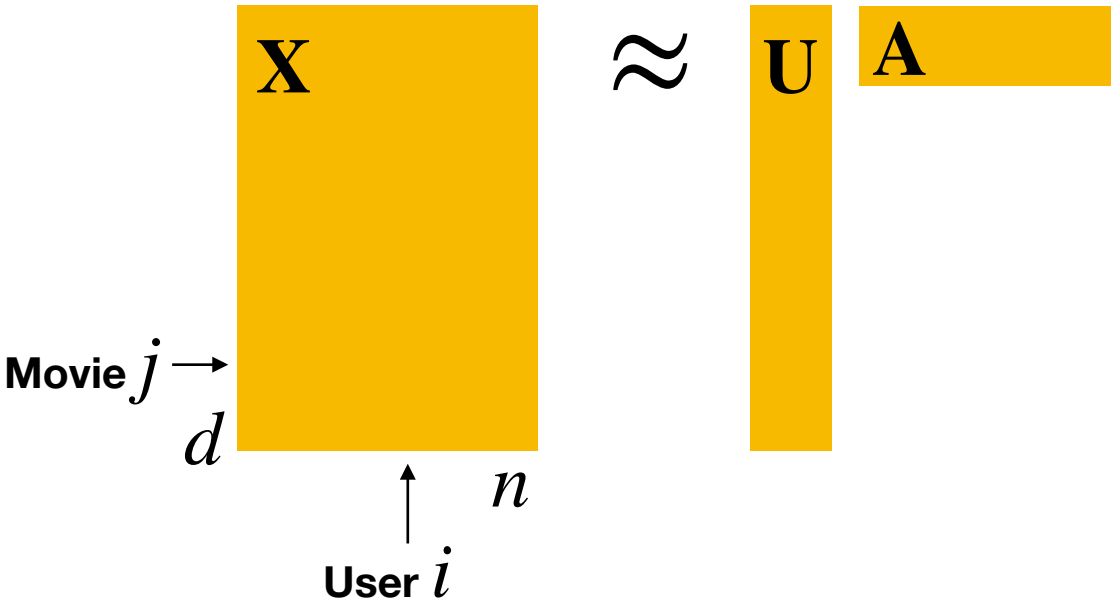
- however, the ratings are not arbitrary, but people with similar tastes rate similarly
- such structure can be modeled using low dimensional representation of the data as follows
- we will find a set of principal component vectors $\mathbf{U} = [u_1 \quad u_2 \quad \cdots \quad u_r] \in \mathbb{R}^{d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

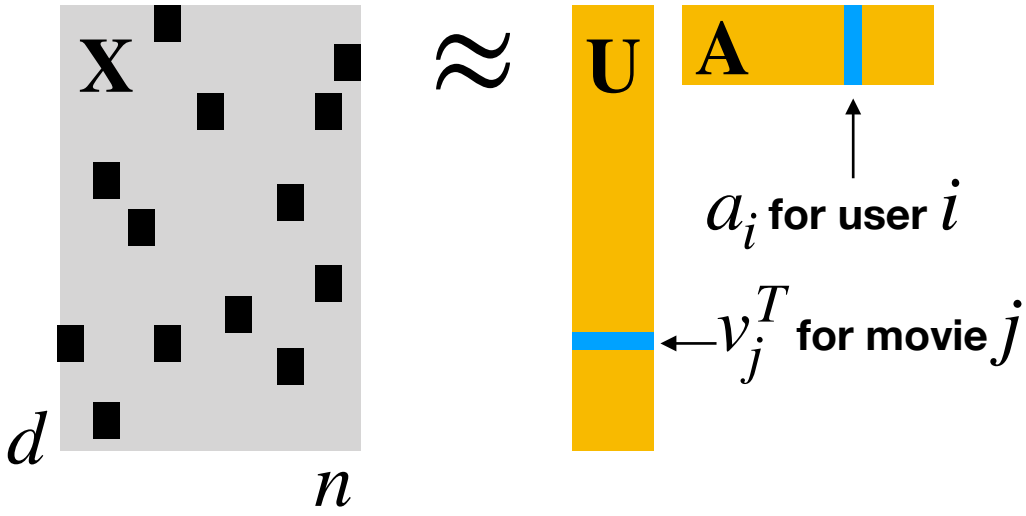
- let $\mathbf{X} = [x_1 \quad x_2 \quad \cdots \quad x_n] \in \mathbb{R}^{d \times n}$ be the ratings matrix, and assume it is fully observed, i.e. we know all the entries
- then we want to find $\mathbf{U} \in \mathbb{R}^{d \times r}$ and $\mathbf{A} = [a_1 \quad a_2 \quad \cdots \quad a_n] \in \mathbb{R}^{r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



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

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

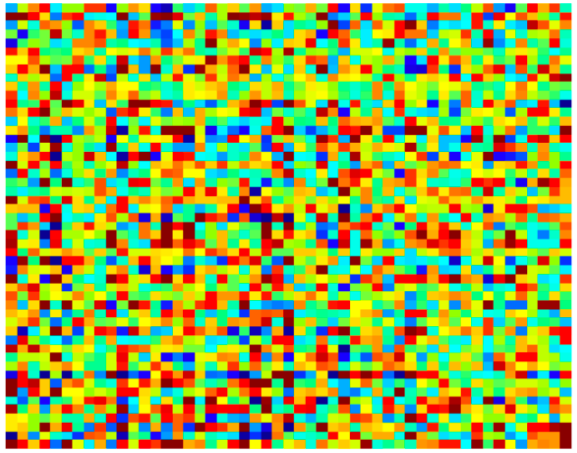
- this can be solved, for example via gradient descent or alternating minimization
- this can be quite accurate, with small number of samples

- 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*

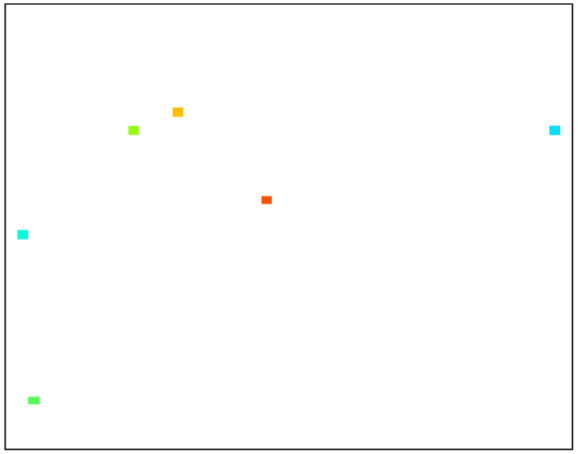
Example: 2000×2000 rank-8 random matrix

For illustration, we zoom in to a 50x50 submatrix

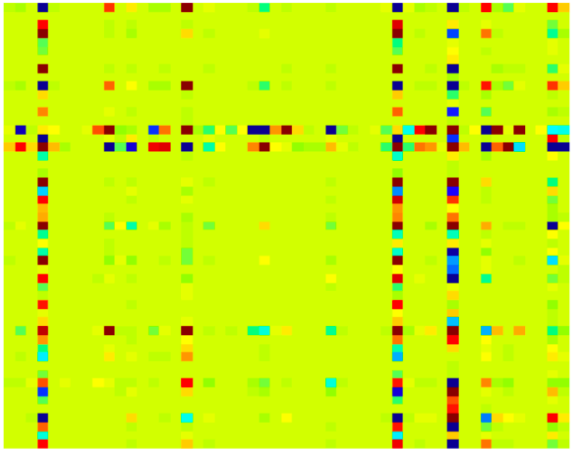
low-rank matrix $\mathbf{X}$



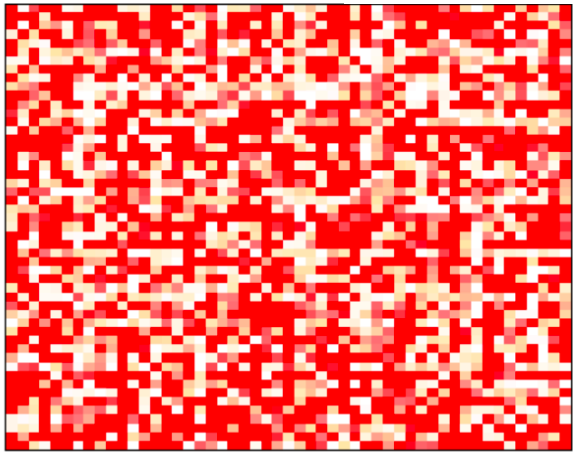
sampled matrix



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



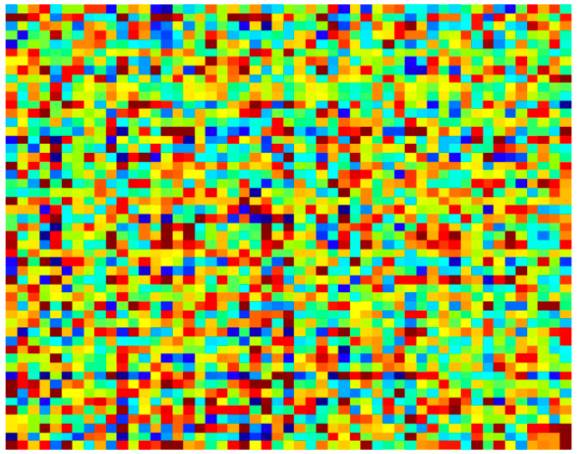
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top)_{ji})^2$



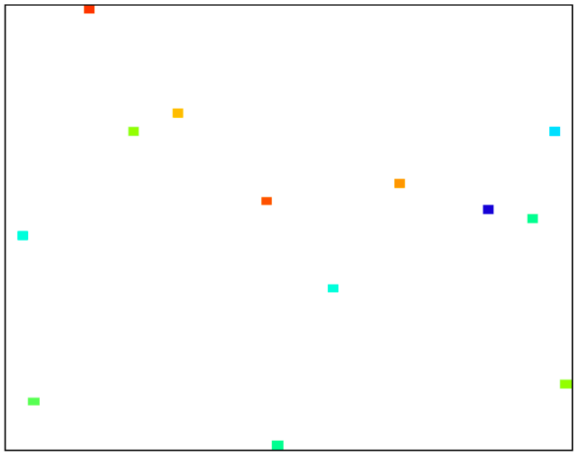
0.25% sampled

Example: 2000×2000 rank-8 random matrix

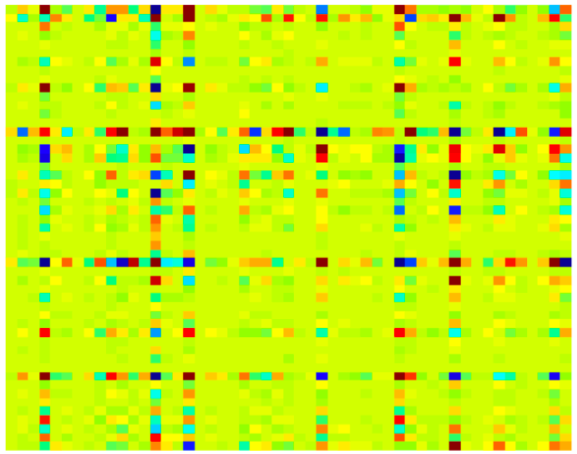
low-rank matrix $\mathbf{X}$



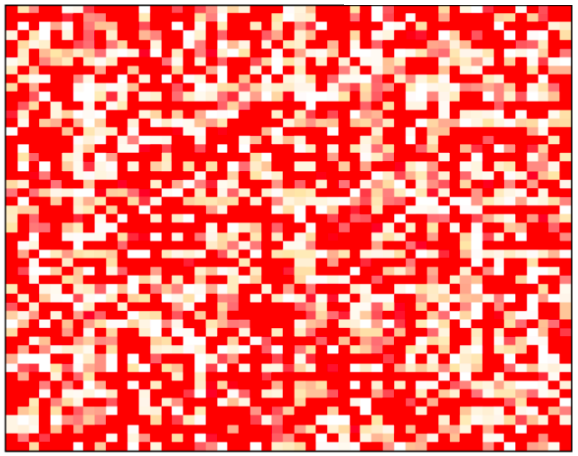
sampled matrix



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



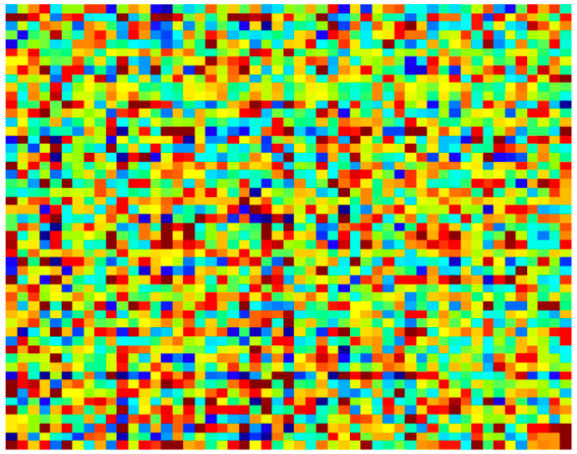
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top)_{ji})^2$



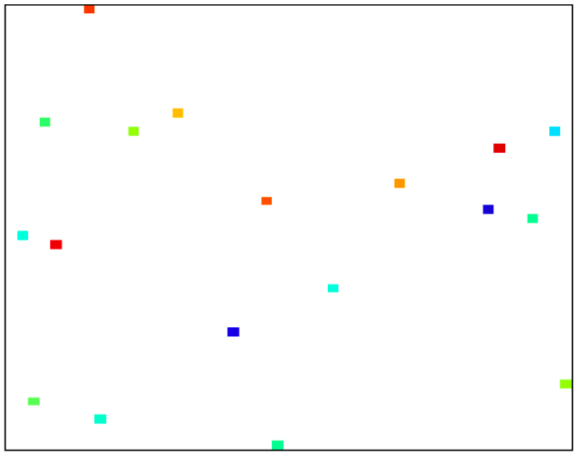
0.50% sampled

Example: 2000×2000 rank-8 random matrix

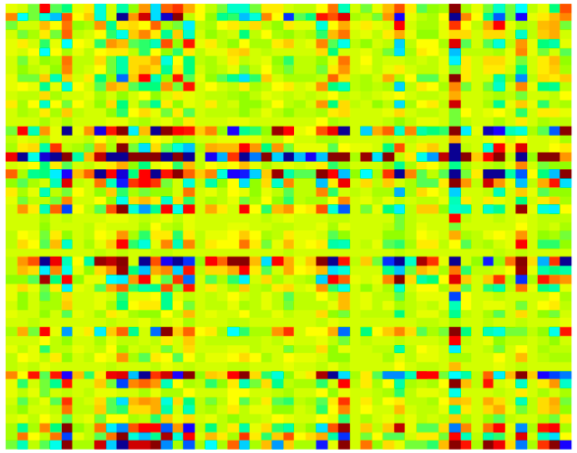
low-rank matrix $\mathbf{X}$



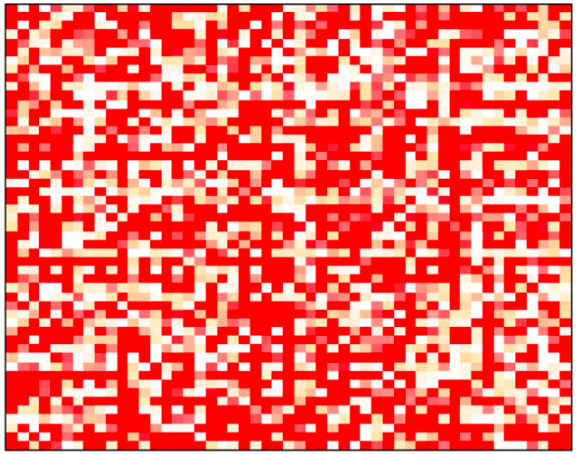
sampled matrix



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



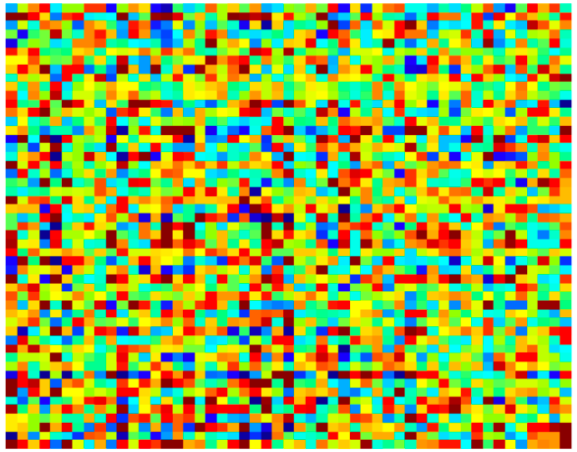
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top)_{ji})^2$



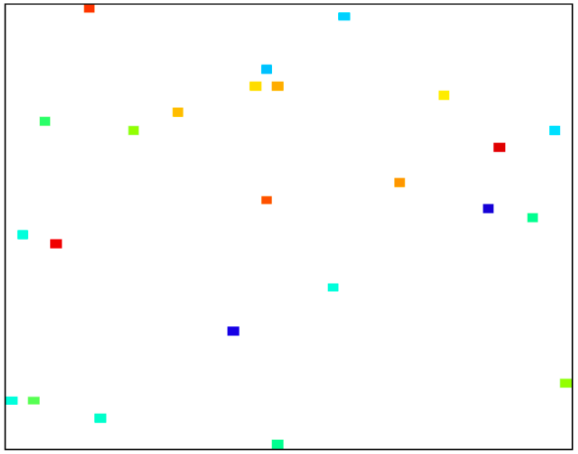
0.75% sampled

Example: 2000×2000 rank-8 random matrix

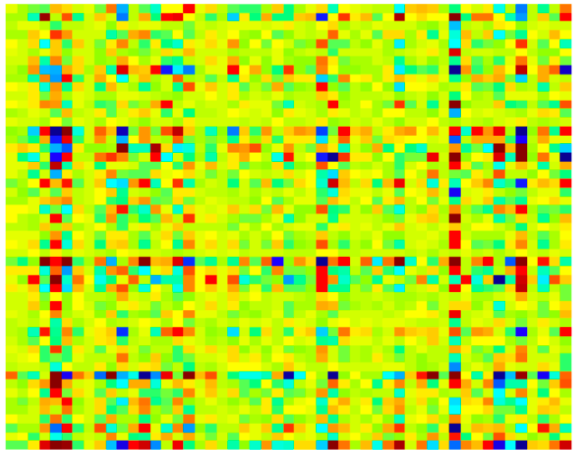
low-rank matrix $\mathbf{X}$



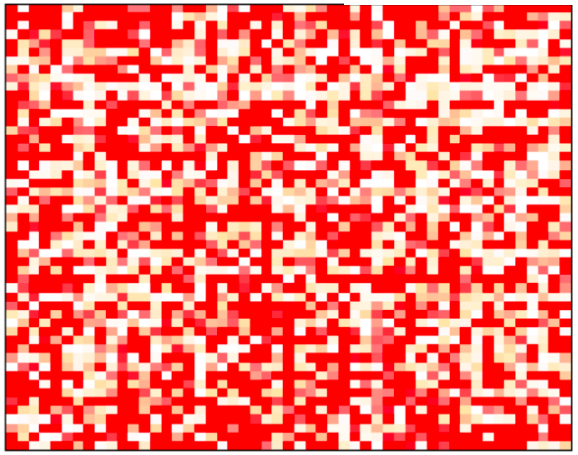
sampled matrix



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



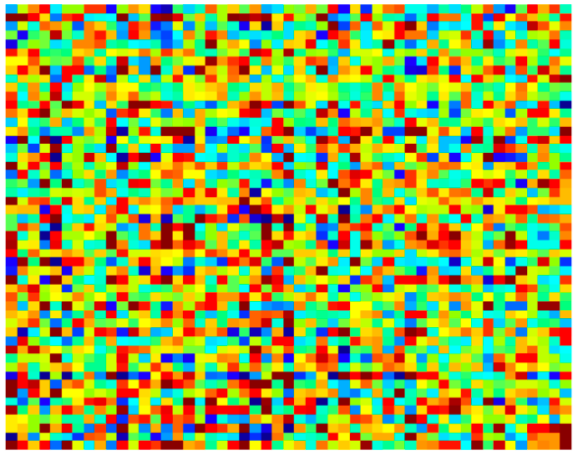
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top)_{ji})^2$



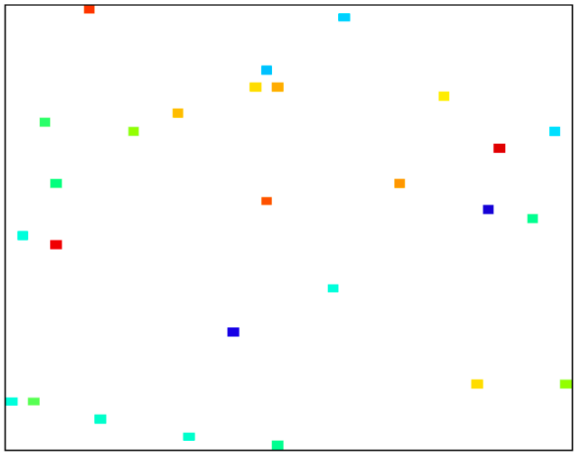
1.00% sampled

Example: 2000×2000 rank-8 random matrix

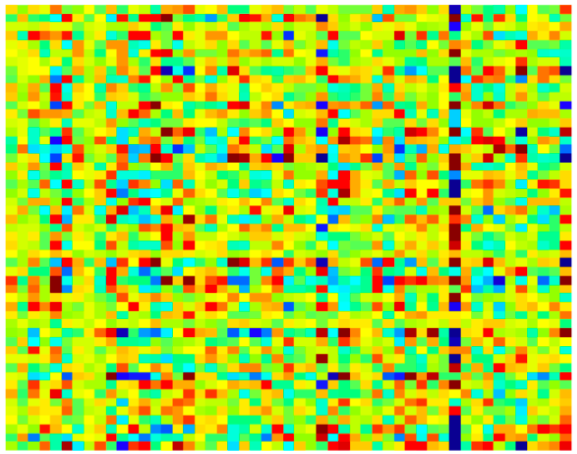
low-rank matrix $\mathbf{X}$



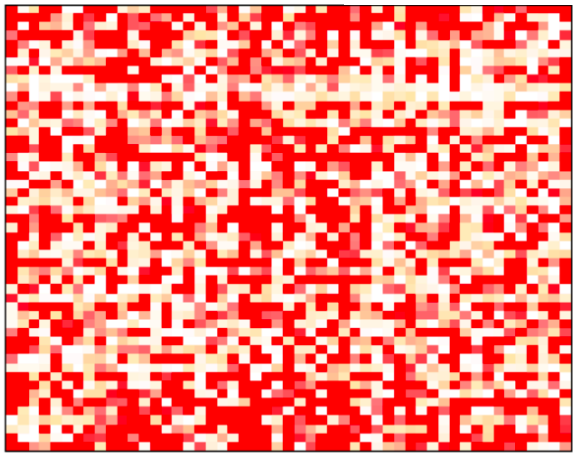
sampled matrix



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



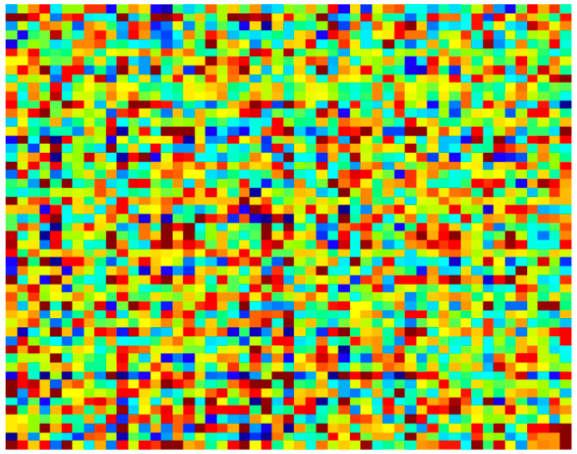
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^T)_{ji})^2$



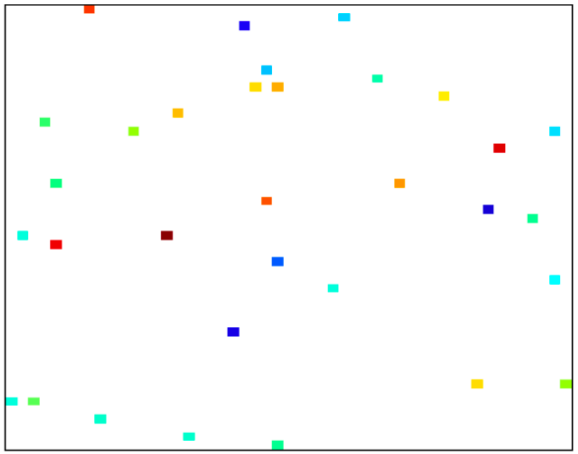
1.25% sampled

Example: 2000×2000 rank-8 random matrix

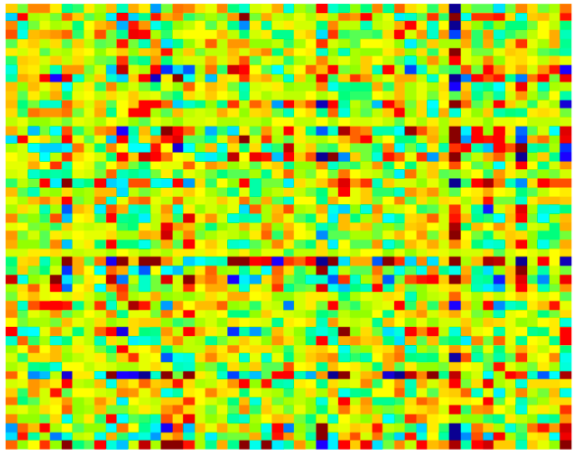
low-rank matrix $\mathbf{X}$



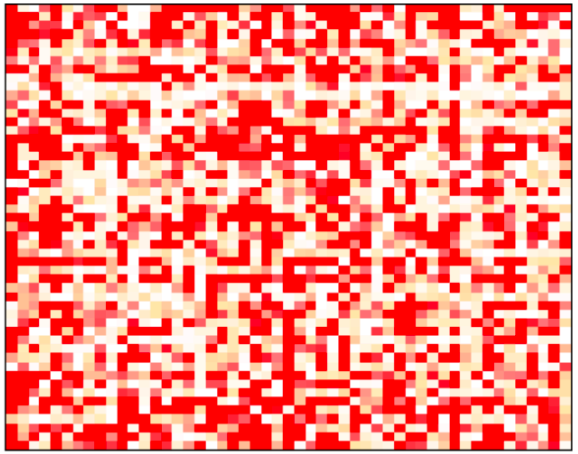
sampled matrix



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



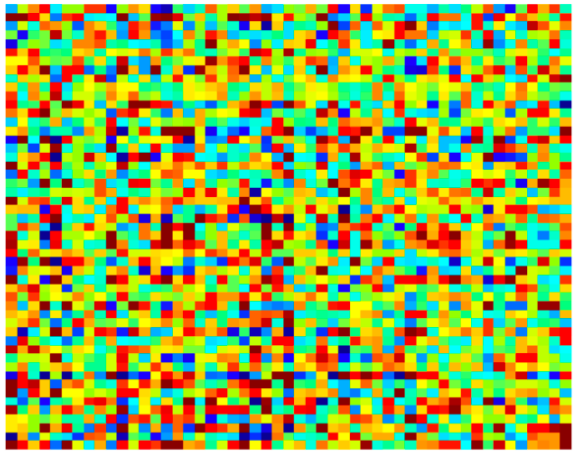
squared error $(\mathbf{X}_{ji} - (\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top)_{ji})^2$



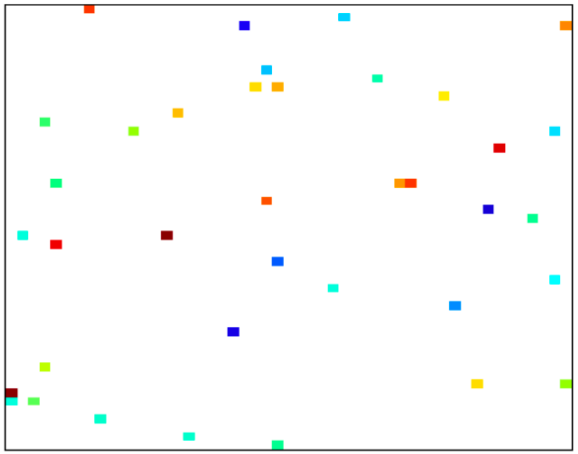
1.50% sampled

Example: 2000×2000 rank-8 random matrix

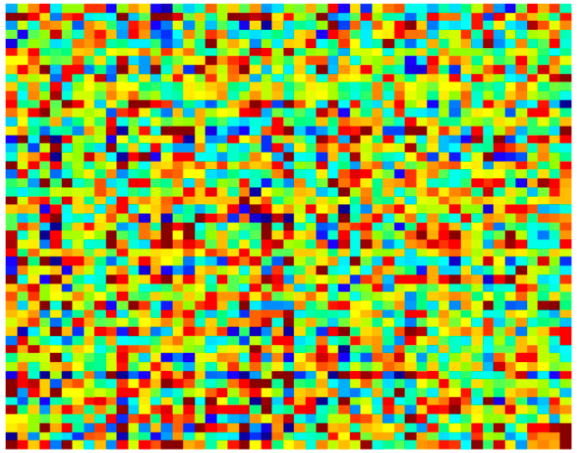
low-rank matrix $\mathbf{X}$



sampled matrix



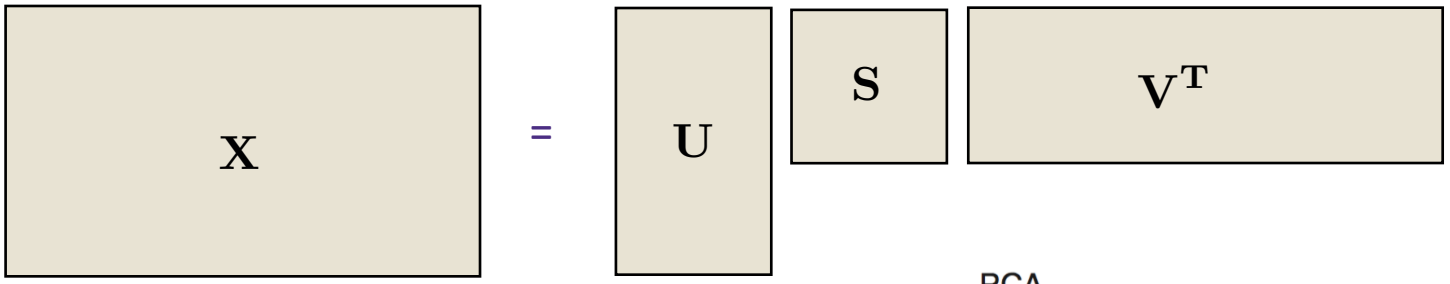
Gradient descent output $\tilde{\mathbf{U}}\tilde{\mathbf{V}}^\top$



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

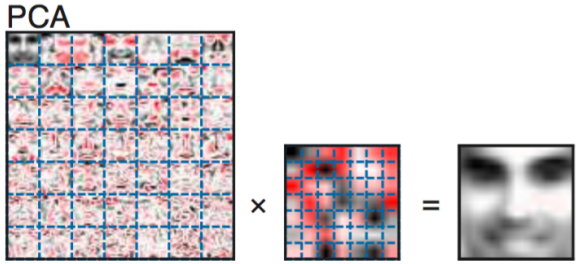
1.75% sampled

Other matrix factorizations



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 $\mathbb{R}_+$