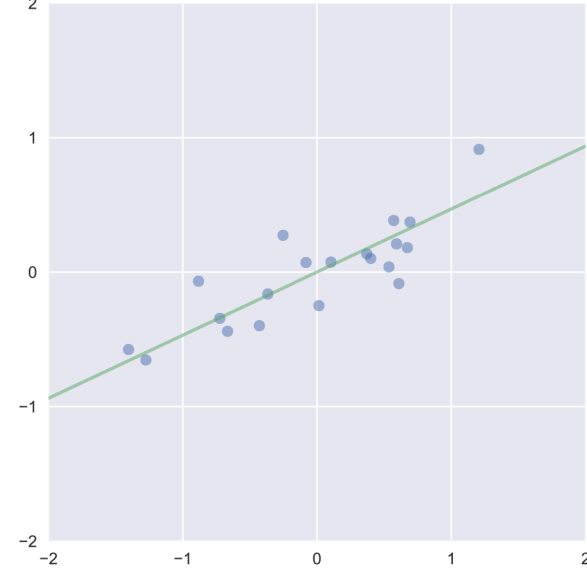


# Lecture 23:

# Principal Component Analysis



- Unsupervised learning
  - Dimensionality reduction
    - PCA
    - Auto-encoder
  - Clustering
    - $k$ -means
    - Spectral, t-SNE, UMAP
  - Generative models
  - Density estimation



# 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

$$\mathbf{C} \stackrel{\text{def}}{=} \frac{1}{n} \mathbf{X}^T \mathbf{X} \in \mathbb{R}^{d \times d}, \quad u \in \mathbb{R}^{d \times r} \quad \text{arg max} \quad \sum_{j=1}^r u_j^T \mathbf{C} u_j$$

- We can also use the Singular Value Decomposition (SVD) to find such eigen vector
- 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

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

# How do we compute singular vectors?

---

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

$$\begin{aligned}\tilde{v}_{t+1} &\leftarrow Cv_t \\ v_{t+1} &\leftarrow \frac{\tilde{v}_{t+1}}{|\tilde{v}_{t+1}|}\end{aligned}$$

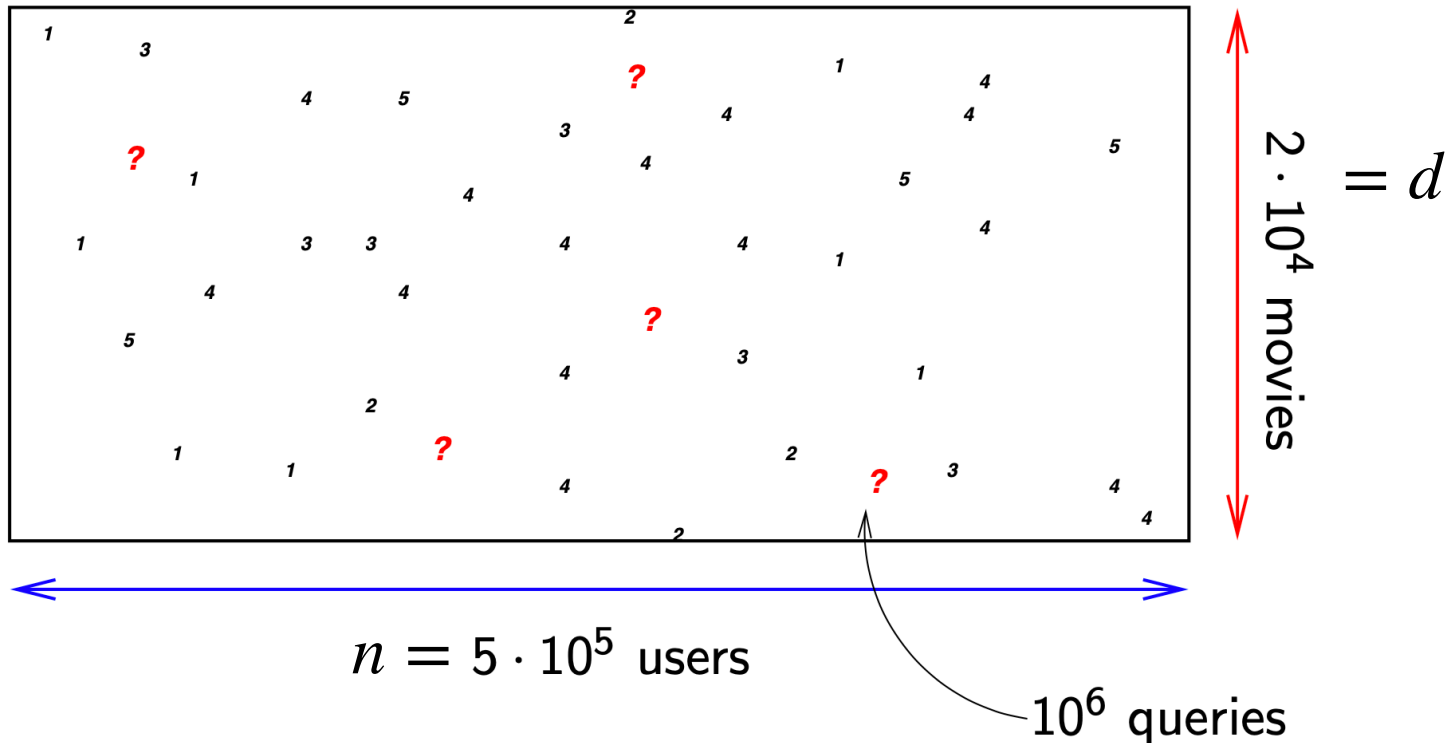
# Power iteration

---

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

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

# 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

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

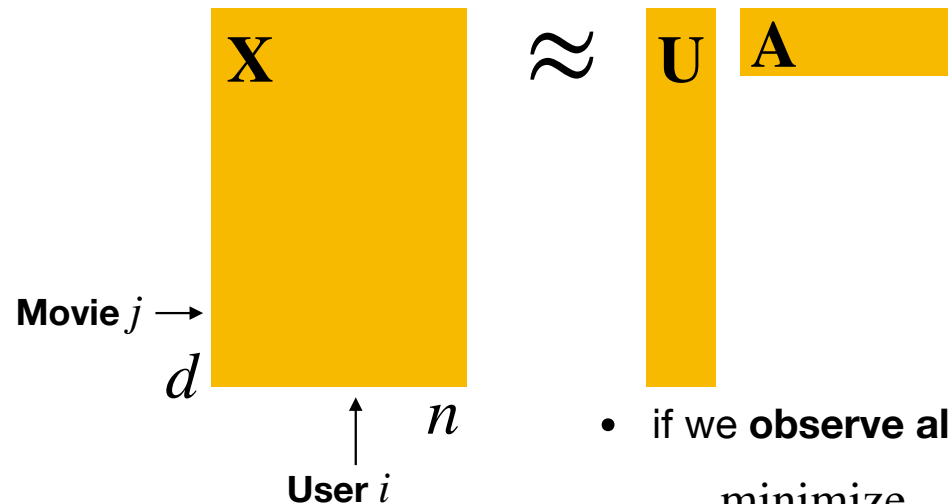
for some lower-dimensional  $a_i \in \mathbb{R}^r$  for  $i$ -th user and some  $r \ll d$

- 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

Handwritten diagram illustrating the vector  $u_1$ . A horizontal arrow labeled "Horror" points to the vector  $u_1$ . The vector  $u_1$  is represented as a column vector of size  $d$ , with the first element highlighted in a box and labeled "j-th movie". The entire vector is also labeled  $u_1[j]$  in a box.

# 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 solve

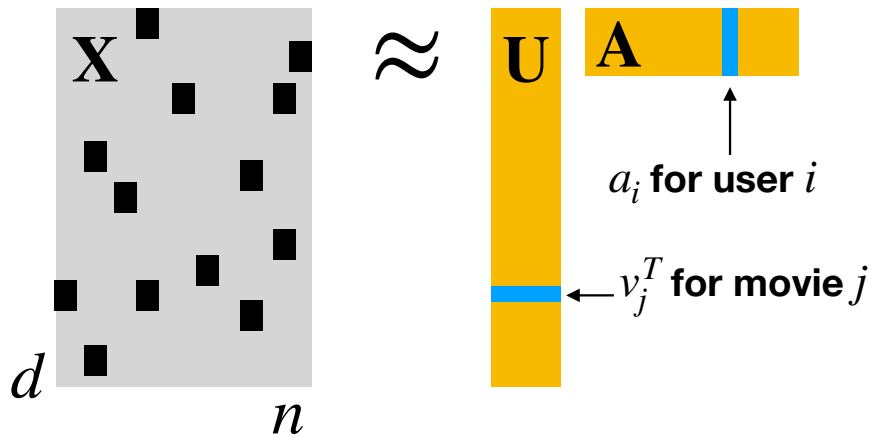
$$\text{minimize}_{\mathbf{U}, \mathbf{A}} \sum_{i=1}^n \|x_i - \mathbf{U}a_i\|_2^2$$

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  $i_\ell$  on movie  $j_\ell$



- 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

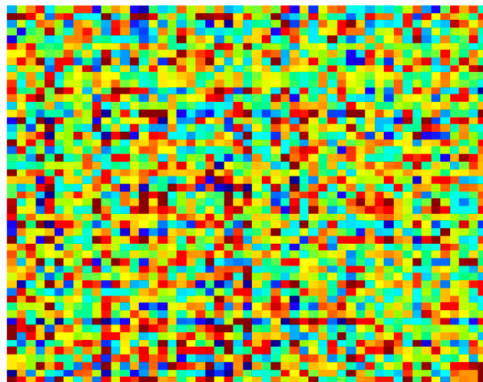
- a natural approach to fit  $v_j$ 's and  $a_i$ 's to given training data is to solve

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

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

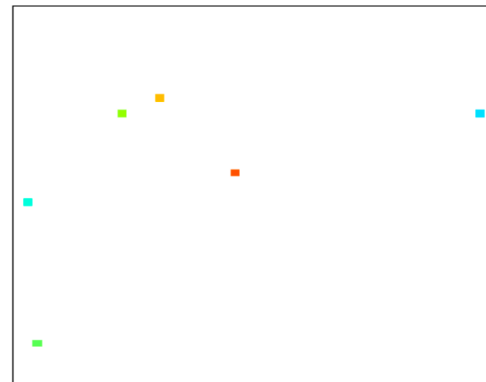
# Example: $2000 \times 2000$ rank-8 random matrix

low-rank matrix  $\mathbf{X}$

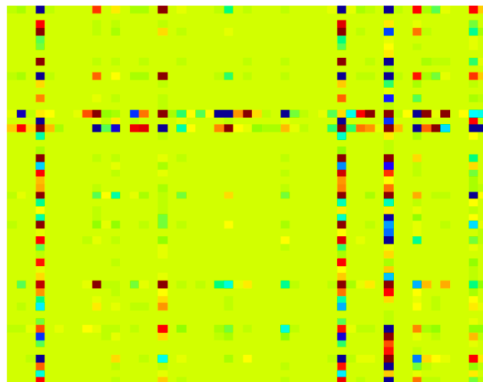


For illustration,  
we zoom in to a  
50x50 submatrix

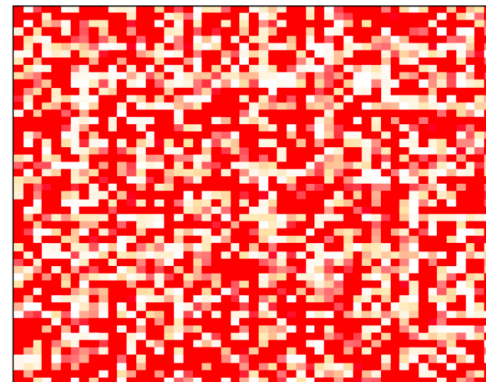
sampled matrix



Gradient descent output  $\mathbf{U}\mathbf{A}$



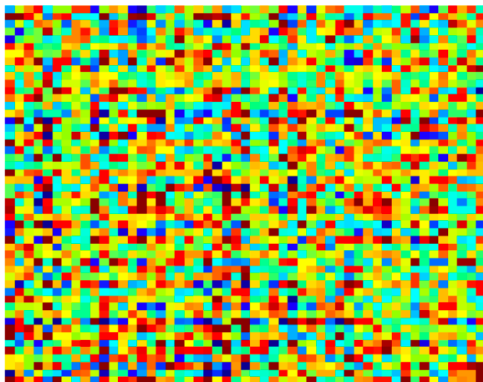
squared error  $(\mathbf{X}_{ji} - (\mathbf{U}\mathbf{A})_{ji})^2$



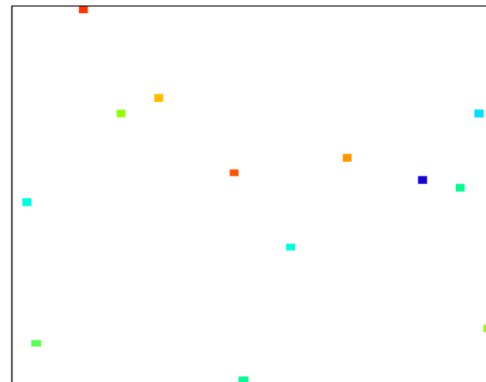
0.25% sampled

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

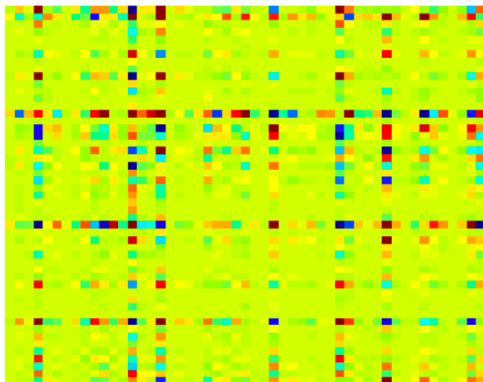
low-rank matrix  $\mathbf{X}$



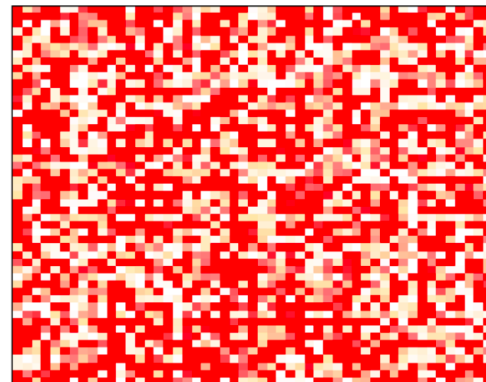
sampled matrix



Gradient descent output  $\mathbf{U}\mathbf{A}$



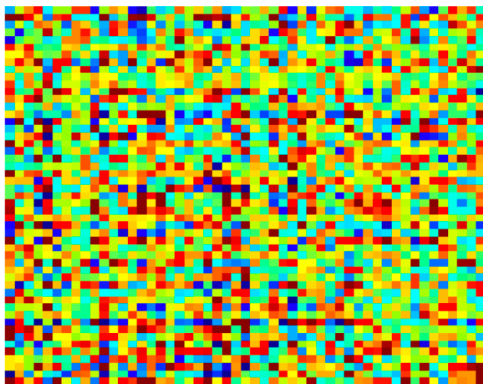
squared error  $(\mathbf{X}_{ji} - (\mathbf{U}\mathbf{A})_{ji})^2$



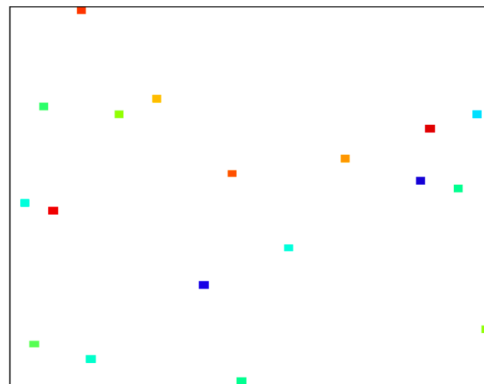
0.50% sampled

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

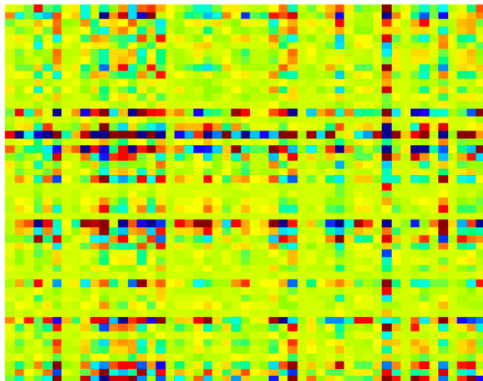
low-rank matrix  $\mathbf{X}$



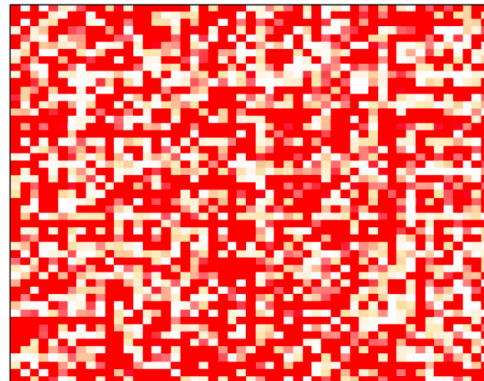
sampled matrix



Gradient descent output  $\mathbf{U}\mathbf{A}$



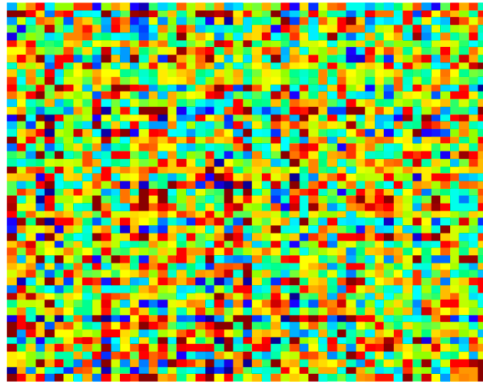
squared error  $(\mathbf{X}_{ji} - (\mathbf{U}\mathbf{A})_{ji})^2$



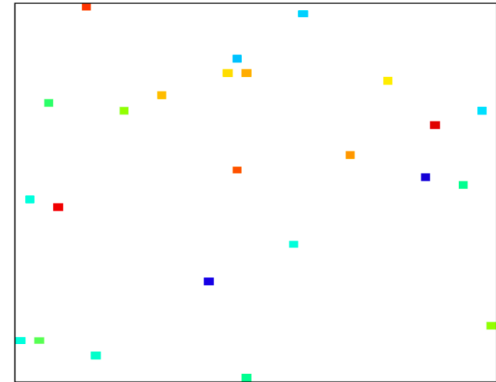
0.75% sampled

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

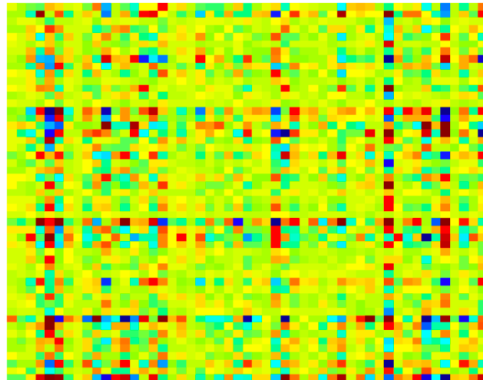
low-rank matrix  $\mathbf{X}$



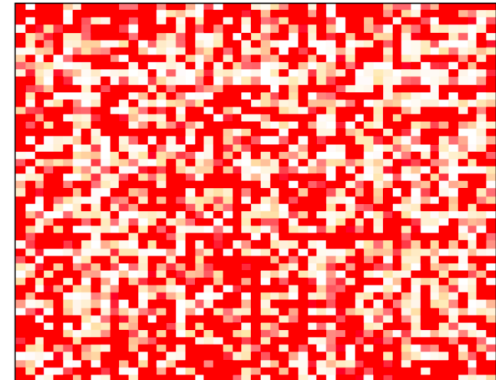
sampled matrix



Gradient descent output  $\mathbf{UA}$



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

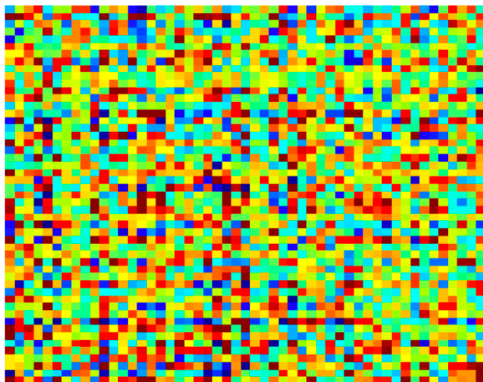


8 entries  
----- RC  
2000 entries  
S11  
0.4%

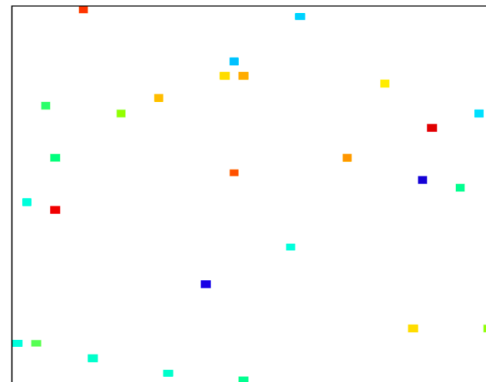
1.00% sampled

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

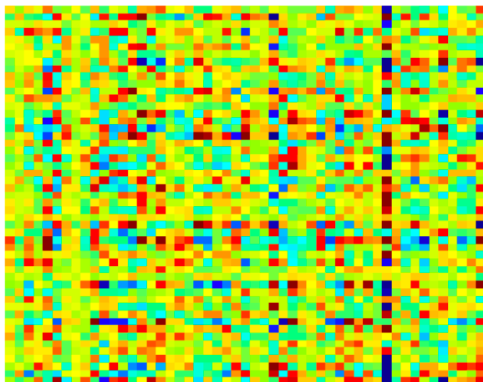
low-rank matrix  $\mathbf{X}$



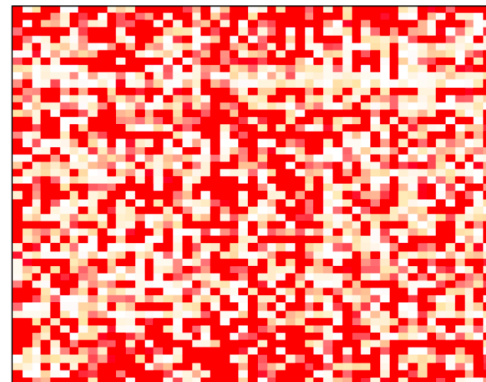
sampled matrix



Gradient descent output  $\mathbf{U}\mathbf{A}$



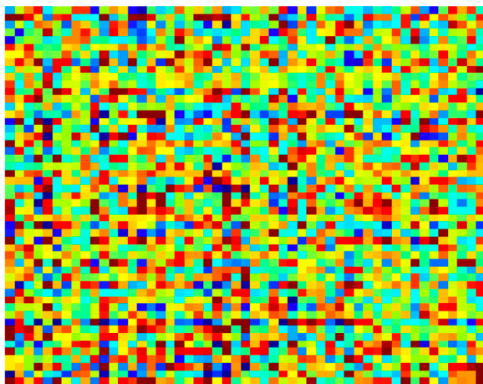
squared error  $(\mathbf{X}_{ji} - (\mathbf{U}\mathbf{A})_{ji})^2$



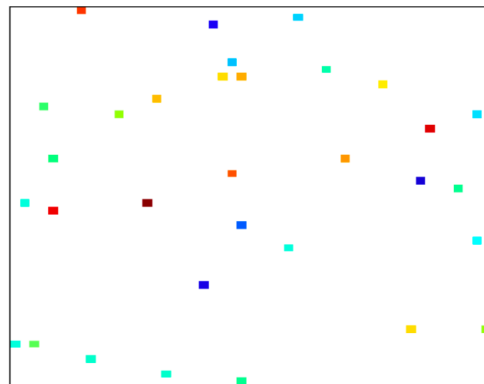
1.25% sampled

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

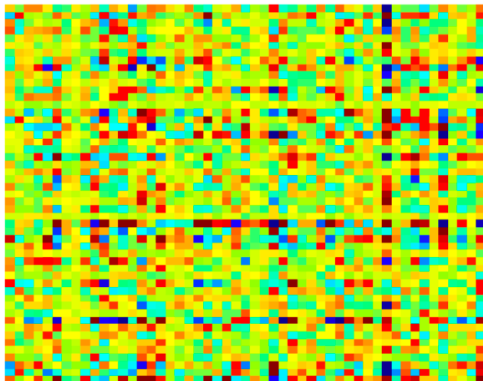
low-rank matrix  $\mathbf{X}$



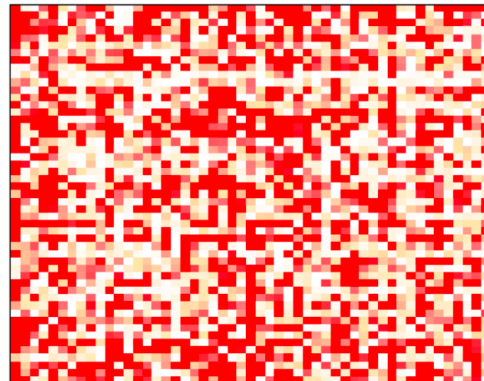
sampled matrix



Gradient descent output  $\mathbf{UA}$



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

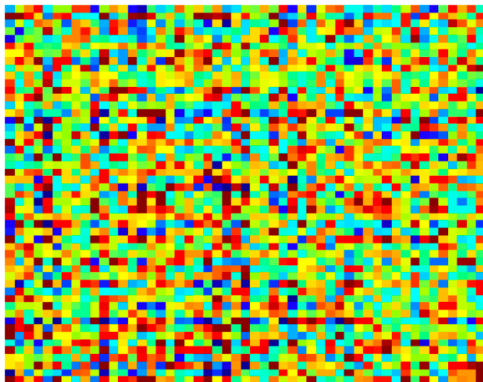


1.50% sampled

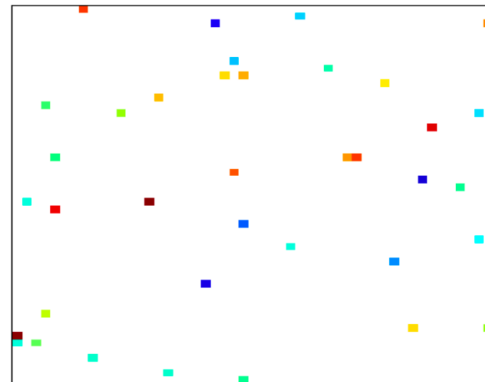


# Example: $2000 \times 2000$ rank-8 random matrix

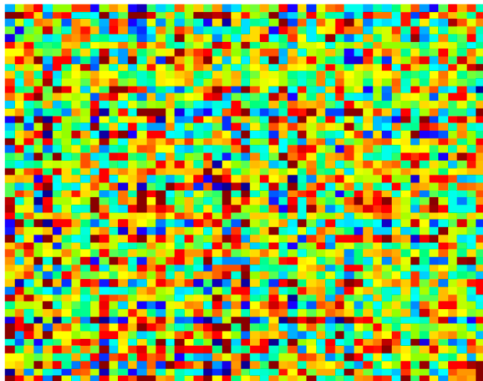
low-rank matrix  $\mathbf{X}$



sampled matrix



Gradient descent output  $\mathbf{U}\mathbf{A}$



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



1.75% sampled

# Matrix completion

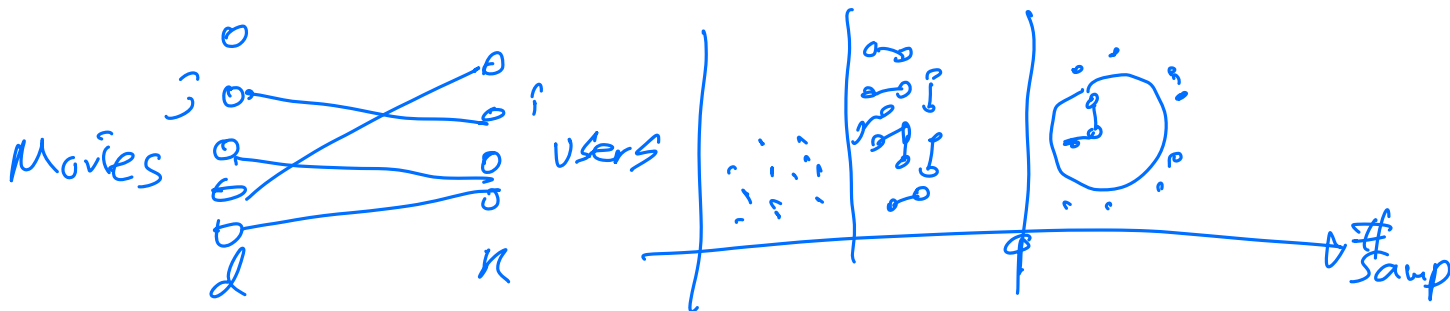
- $$\text{minimize}_{\mathbf{U}, \mathbf{A}} \sum_{(i,j) \in S_{\text{train}}} (\mathbf{X}_{ji} - v_j^T a_i)^2$$
- Gradient descent on  $\{v_j\}_{j=1}^d$  and  $\{a_i\}_{i=1}^n$  can be implemented via

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

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

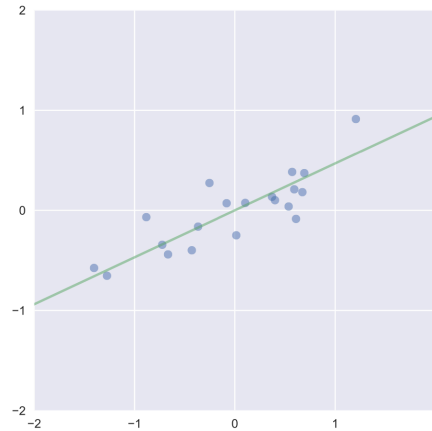


# Matrix completion

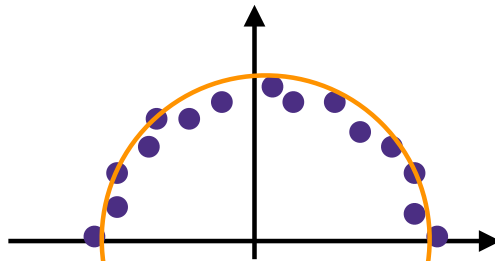
- $$\text{minimize}_{\mathbf{U}, \mathbf{A}} \sum_{(i,j) \in S_{\text{train}}} (\mathbf{X}_{ji} - v_j^T a_i)^2$$
- alternating minimization
  - repeat
    - fix  $v_j$ 's and find optimal  $a_i$ 's
      - for each  $i$ , set the gradient to zero:
$$2 \sum_{j \in S_i} ((v_j^{(t-1)})^T a_i - \mathbf{X}_{ji})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

# Autoencoders

- PCA is great in capturing variations in linear subspaces
  - It finds the best linear subspace for dimensionality reduction

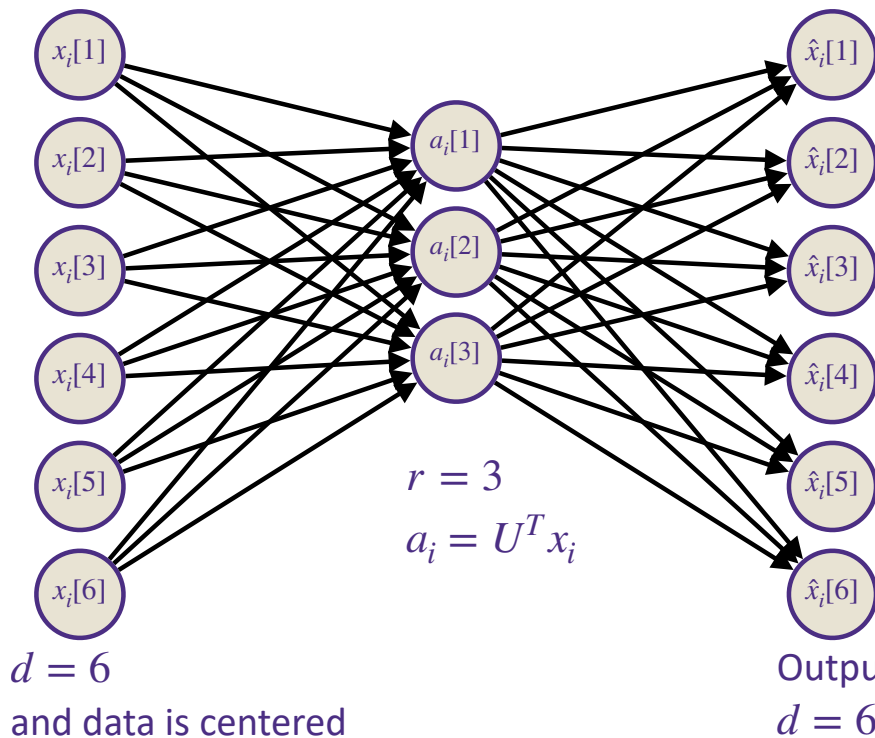


- PCA fails when variation is in non-linear manifolds
  - A non-linear encoding of data  $x_i$  for dimensionality reduction for these examples is to store the slope  $\alpha_i = x_i[1]/x_i[2]$



# Autoencoders

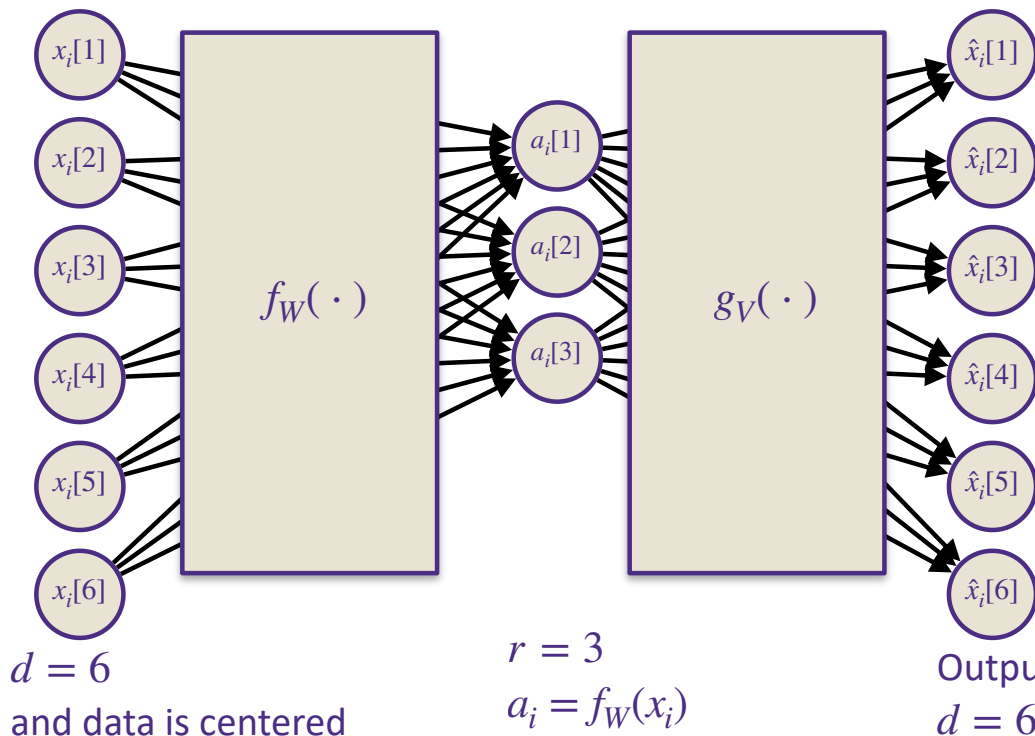
- Neural network perspective of PCA
  - Recall PCA reconstruction is  $\hat{x}_i = UU^T x_i$ , which can be encoded as a neural networks as follows
  - This is a special neural network for unsupervised learning (or label is the same as input), with first layer weight  $U^T \in \mathbb{R}^{r \times d}$  with no activation function and second layer weight  $U \in \mathbb{R}^{r \times d}$



- We train the weights of this neural network to minimize the squared loss
$$\arg \min_U \sum_{i=1}^n \|x_i - \hat{x}_i\|_2^2$$
- PCA is the optimal solution of this neural network training

# Autoencoders

- Autoencoders use neural networks to learn non-linear manifolds that minimize the reconstruction loss
  - $\hat{x}_i = g_V(f_W(x_i))$ , where the encoder  $f_W : \mathbb{R}^d \rightarrow \mathbb{R}^r$  and the decoder  $g_V : \mathbb{R}^r \rightarrow \mathbb{R}^d$  are neural networks
  - We are essentially trying to learn the identity function, but with smaller (non-linear) dimensionality

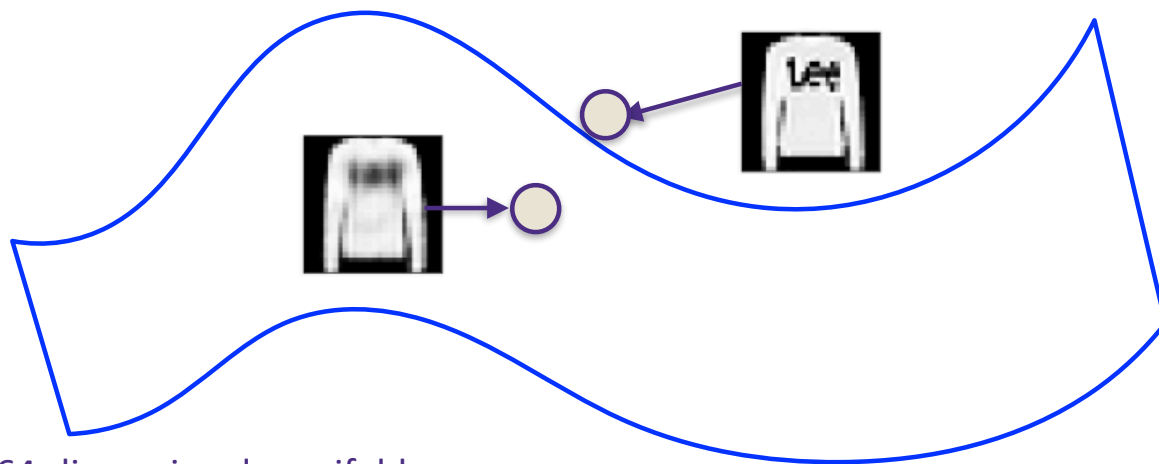
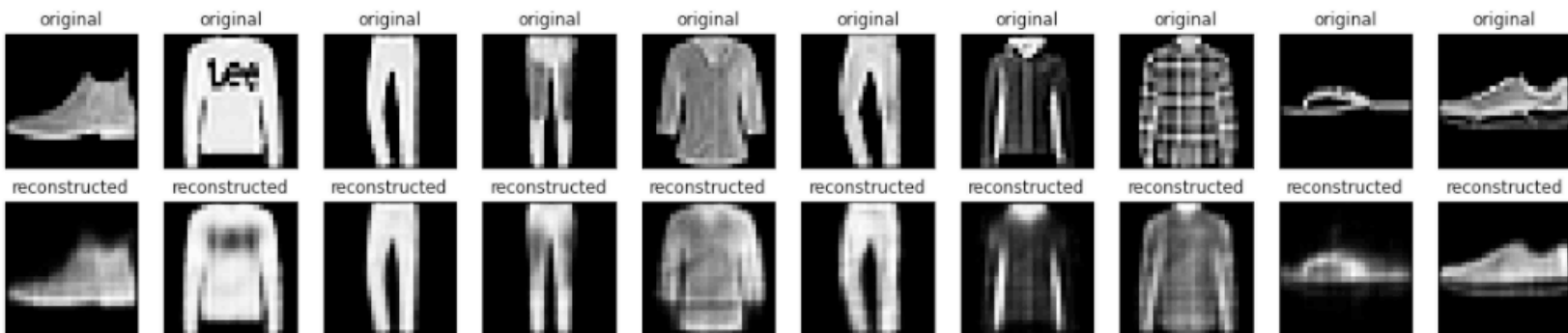


- We train the weights of this neural network to minimize the squared loss

$$\arg \min_U \sum_{i=1}^n \|x_i - \hat{x}_i\|_2^2$$

# Example

- Autoencoder trained on Fashion MNIST dataset with  $r=64$  and 2 fully connected layers for encoder and decoder

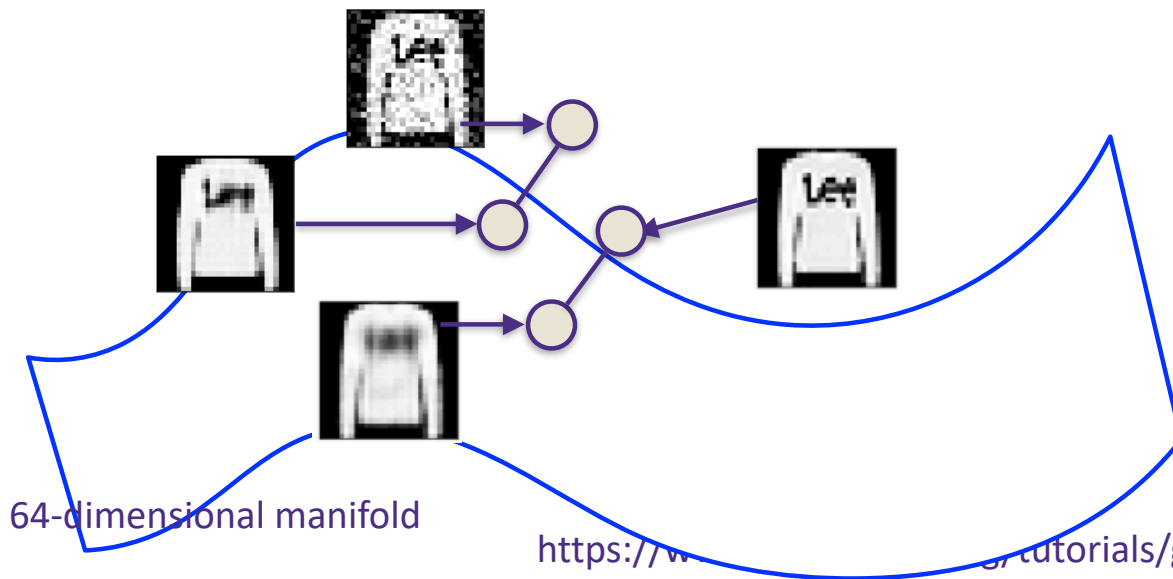
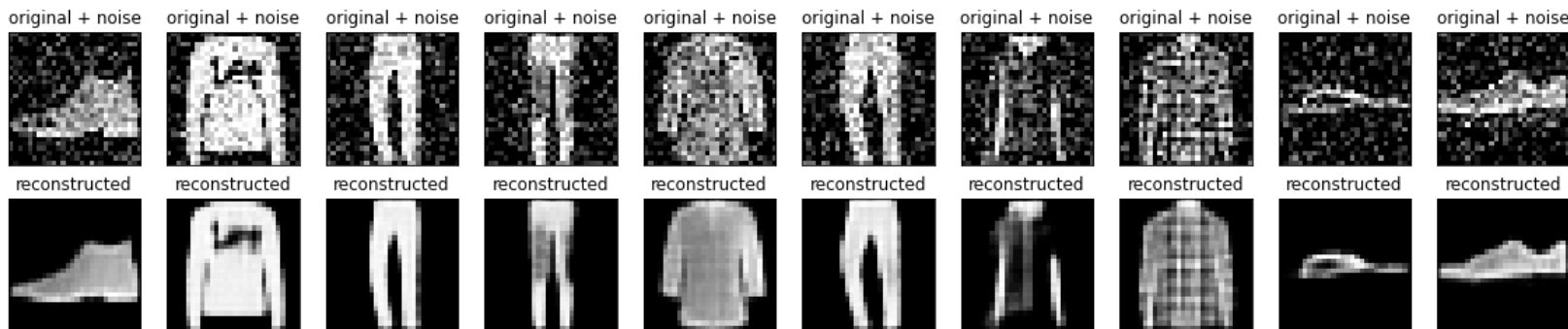


64-dimensional manifold

<https://www.tensorflow.org/tutorials/generative/autoencoder>

# Example

- An autoencoder trained on clean data can be used to denoise noisy data



$$g_v: \mathbb{R}^z \rightarrow \mathbb{R}^d$$



# Questions?

---

- Beyond obvious data compression and dimensionality reduction, such autoencoders have several important applications such as de-noising and anomaly detection

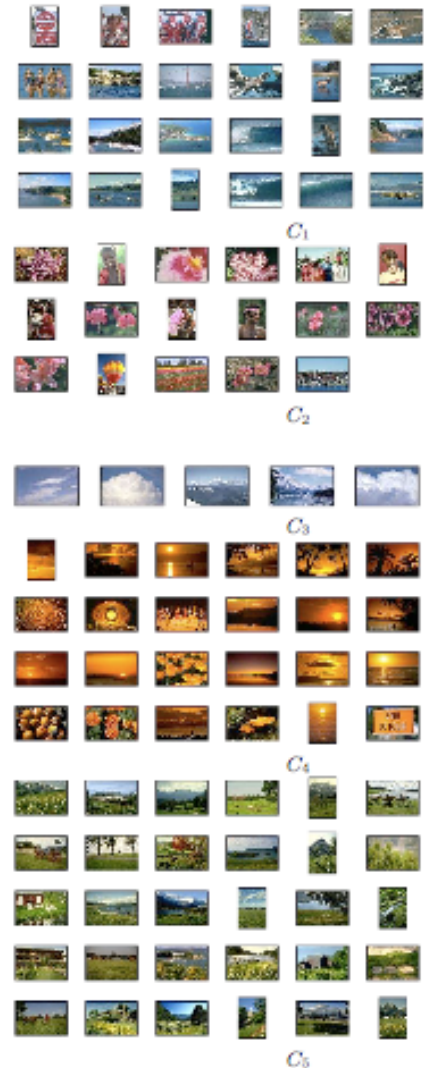
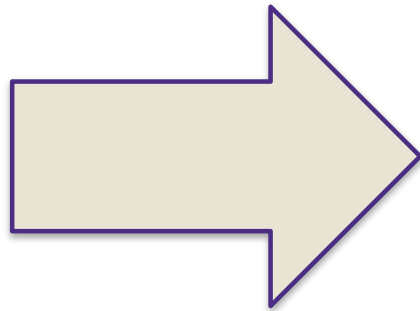
# Lecture 24:

## Clustering with $k$ -means

- Unsupervised learning
  - Dimensionality reduction
    - PCA
    - Auto-encoder
  - Clustering
    - $k$ -means
    - Spectral,t-SNE,UMAP
  - Generative models
  - Density estimation




# Clustering images



[Goldberger et al.]

# Clustering web search results



web news images wikipedia blogs jobs more »

race

Search

advanced preferences

clusters sources sites

remix

All Results (238)

Car (28)

Race cars (7)

Photos, Races Scheduled (5)

Game (4)

Track (3)

Nascar (2)

Equipment And Safety (2)

Other Topics (7)

Photos (22)

Game (14)

Definition (13)

Team (18)

Human (8)

Classification Of Human (2)

Statement, Evolved (2)

Other Topics (4)

Weekend (8)

Ethnicity And Race (7)

Race for the Cure (8)

Race Information (8)

more | all clusters

find in clusters:

Find

Cluster Human contains 8 documents.

Search Results

1. [Race \(classification of human beings\) - Wikipedia, the free ...](#)

The term **race** or racial group usually refers to the concept of dividing **humans** into populations or groups on the basis of various sets of characteristics. The most widely used **human** racial categories are based on visible traits (especially skin color, cranial or facial features and hair texture), and self-identification. Conceptions of **race**, as well as specific ways of grouping **races**, vary by culture and over time, and are often controversial for scientific as well as social and political reasons. History · Modern debates · Political and ...  
[en.wikipedia.org/wiki/Race\\_\(classification\\_of\\_human\\_beings\)](#) - [cache] - Live, Ask

2. [Race - Wikipedia, the free encyclopedia](#)

General. **Racing** competitions The **Race** (yachting **race**), or La course du millénaire, a no-rules round-the-world sailing event; **Race** (biology), classification of flora and fauna; **Race** (classification of human beings) **Race** and ethnicity in the United States Census, official definitions of "**race**" used by the US Census Bureau; **Race** and genetics, notion of racial classifications based on genetics. Historical definitions of **race**; **Race** (bearing), the inner and outer rings of a rolling-element bearing. **RACE** in molecular biology "Rapid ... General · Surnames · Television · Music · Literature · Video games  
[en.wikipedia.org/wiki/Race](#) - [cache] - Live, Ask

3. [Publications | Human Rights Watch](#)

The use of torture, unlawful rendition, secret prisons, unfair trials, ... Risks to Migrants, Refugees, and Asylum Seekers in Egypt and Israel ... In the run-up to the Beijing Olympics in August 2008, ...  
[www.hrw.org/backgrounder/usa/race](#) - [cache] - Ask

4. [Amazon.com: Race: The Reality Of Human Differences: Vincent Sarich ...](#)

Amazon.com: **Race: The Reality Of Human Differences: Vincent Sarich, Frank Miele: Books ...** From Publishers Weekly Sarich, a Berkeley emeritus anthropologist, and Miele, an editor ...  
[www.amazon.com/Race-Reality-Differences-Vincent-Sarich/dp/0813340861](#) - [cache] - Live

5. [AAPA Statement on Biological Aspects of Race](#)

AAPA Statement on Biological Aspects of **Race** ... Published in the American Journal of Physical Anthropology, vol. 101, pp 569-570, 1996 ... PREAMBLE As scientists who study **human** evolution and variation, ...  
[www.physanth.org/positions/race.html](#) - [cache] - Ask

6. [race: Definition from Answers.com](#)

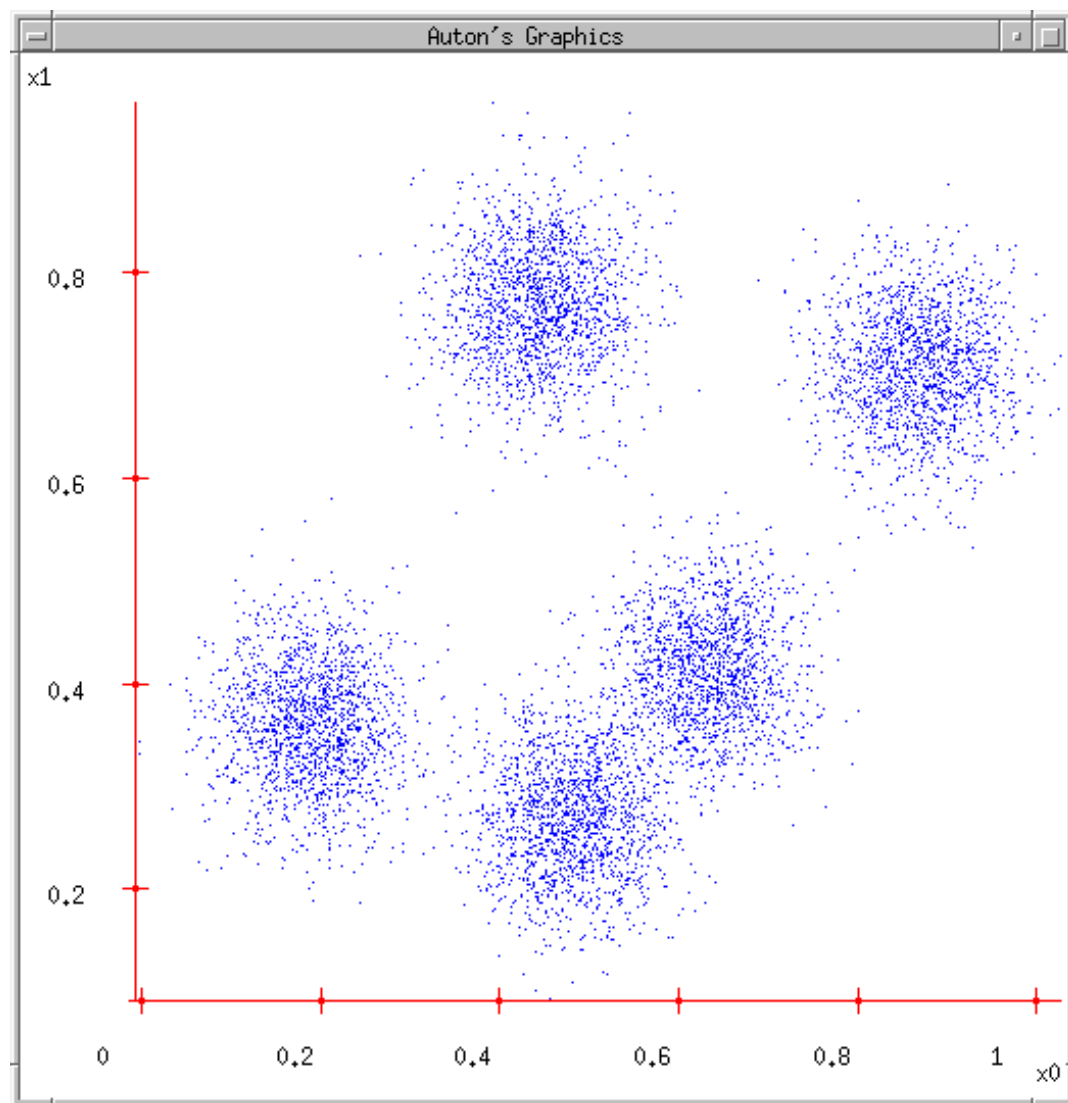
**race** n. A local geographic or global **human** population distinguished as a more or less distinct group by genetically transmitted physical  
[www.answers.com/topic/race-1](#) - [cache] - Live

7. [Dopefish.com](#)

Site for newbies as well as experienced Dopefish followers, chronicling the birth of the Dopefish, its numerous appearances in several computer games, and its eventual take-over of the **human race**. Maintained by Mr. Dopefish himself, Joe Siegler of Apogee Software.  
[www.dopefish.com](#) - [cache] - Open Directory

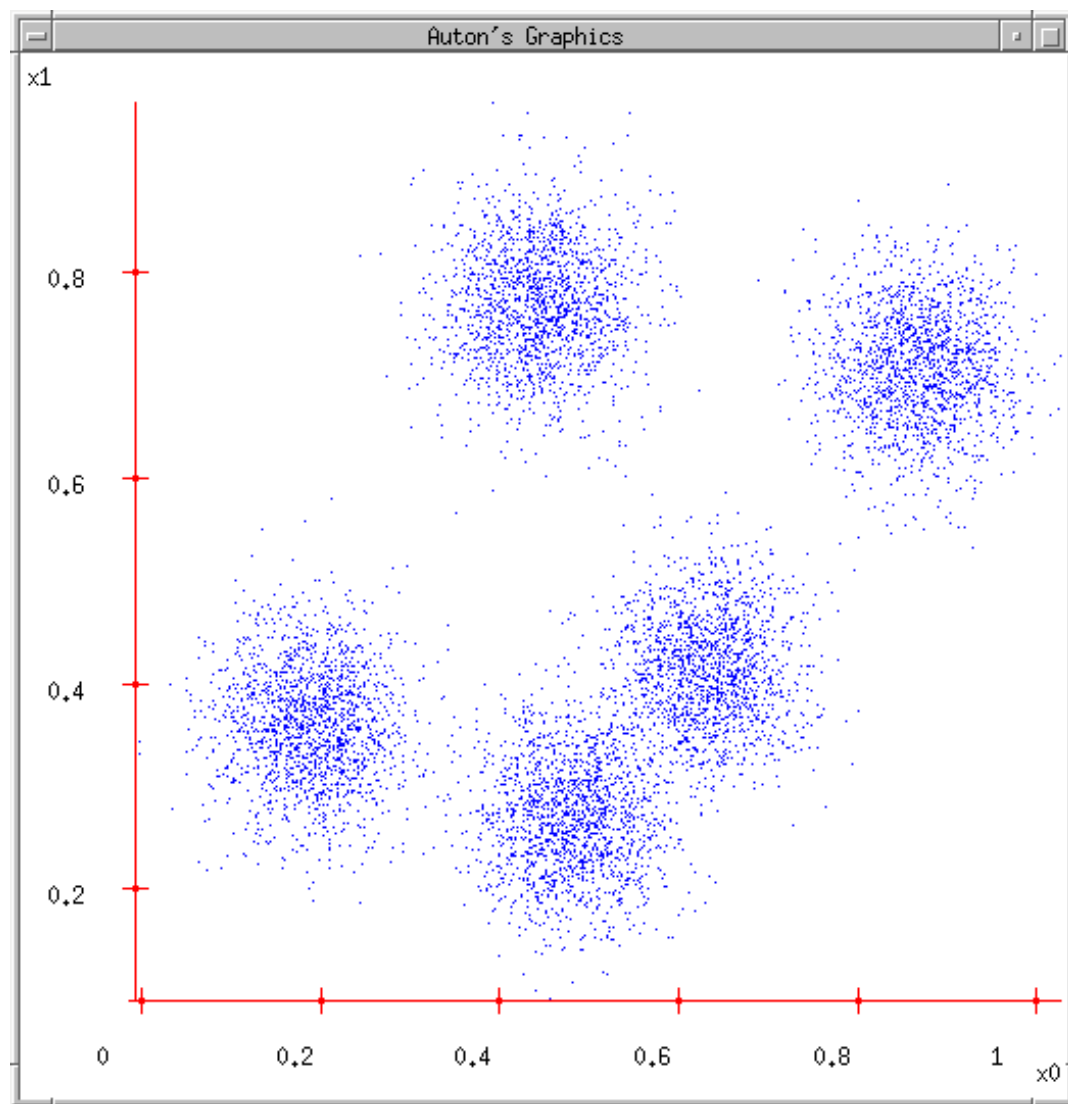
# Some Data

- K-mean algorithm assumes this kind of structured data



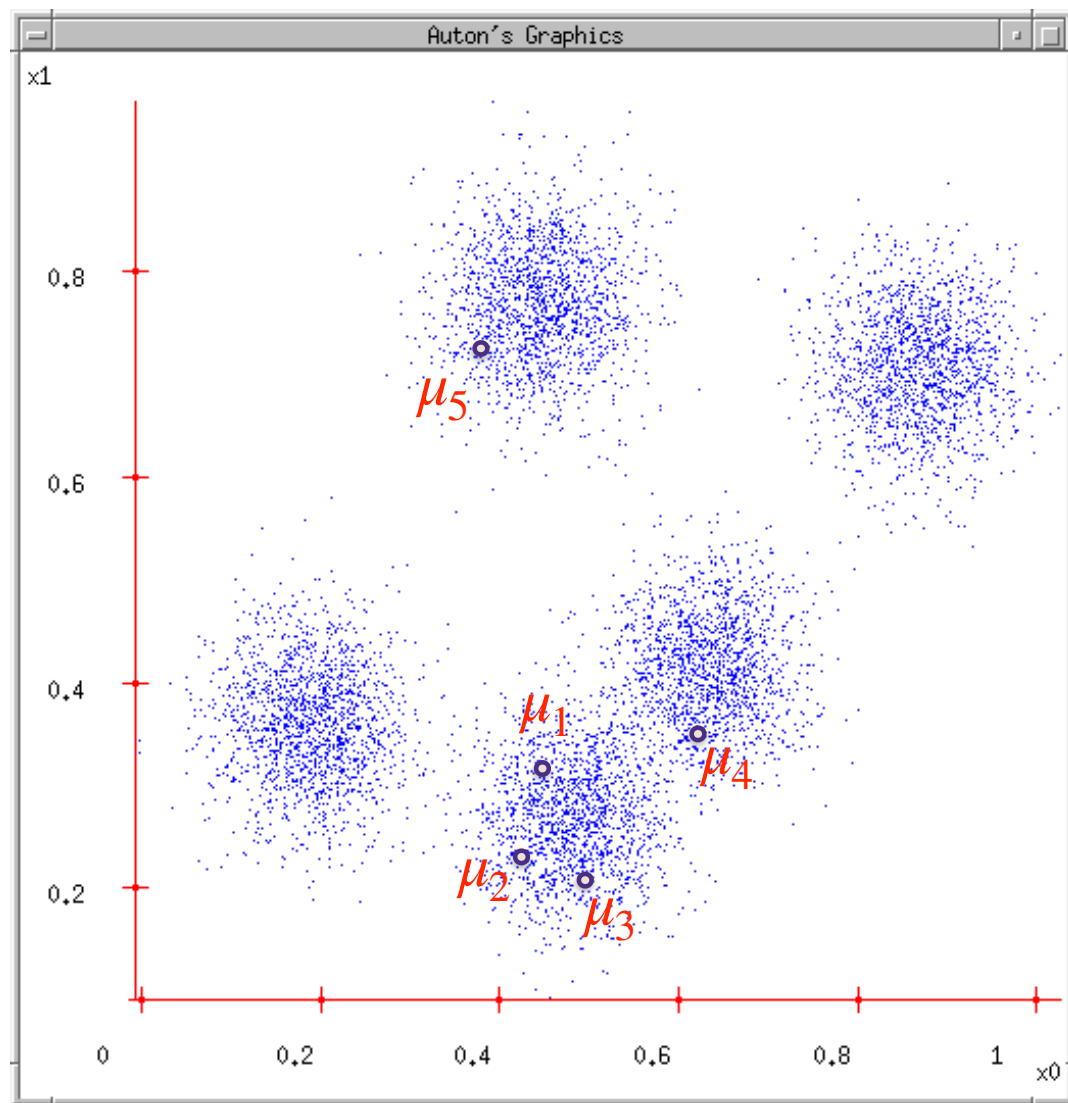
# K-means

1. Ask user how many clusters they'd like. (e.g.  $k=5$ )



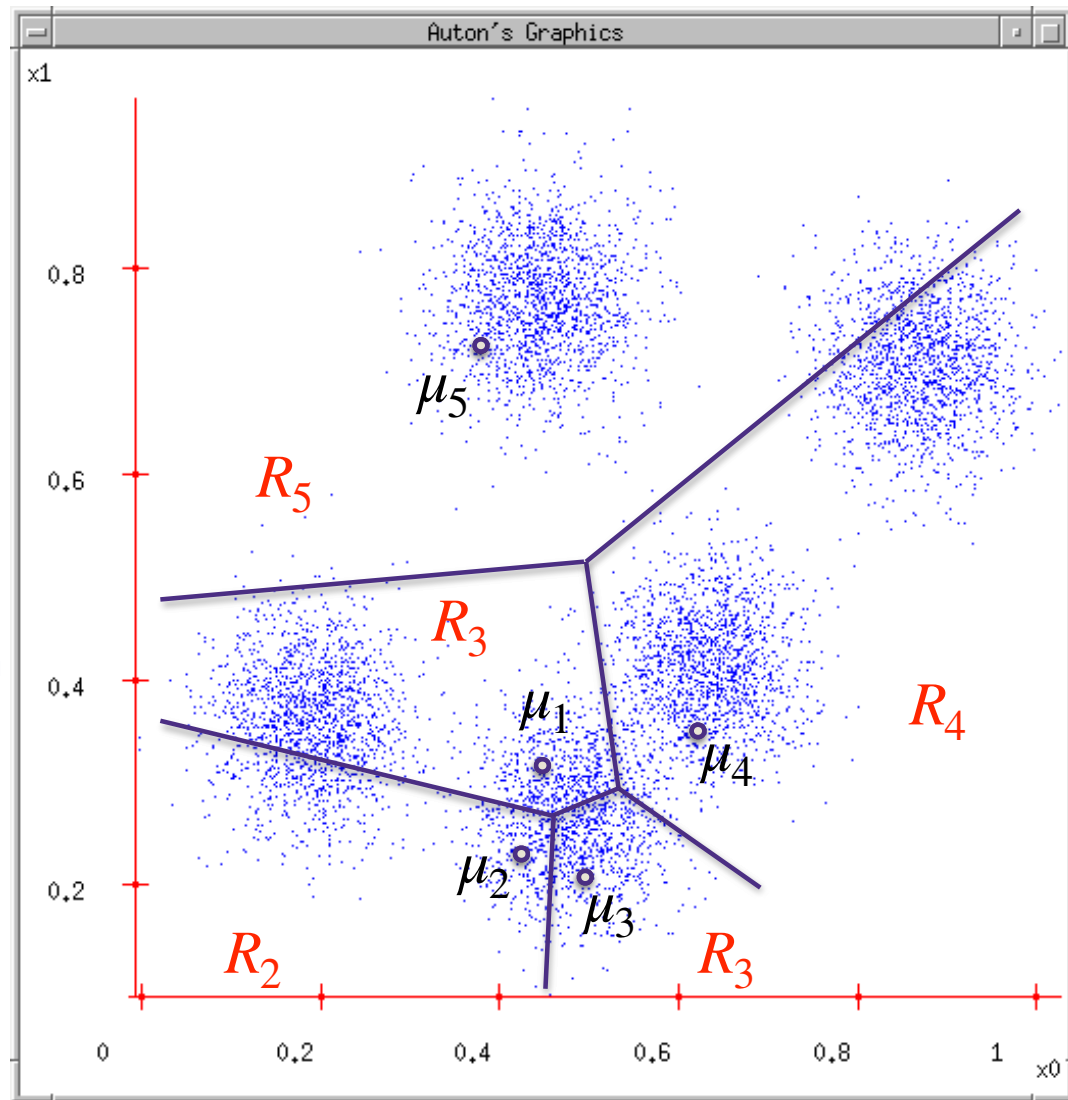
# K-means

1. Ask user how many clusters they'd like. (e.g.  $k=5$ )
2. Randomly guess  $k$  cluster Center locations  
 $\{\mu_1, \dots, \mu_5\}$



# K-means

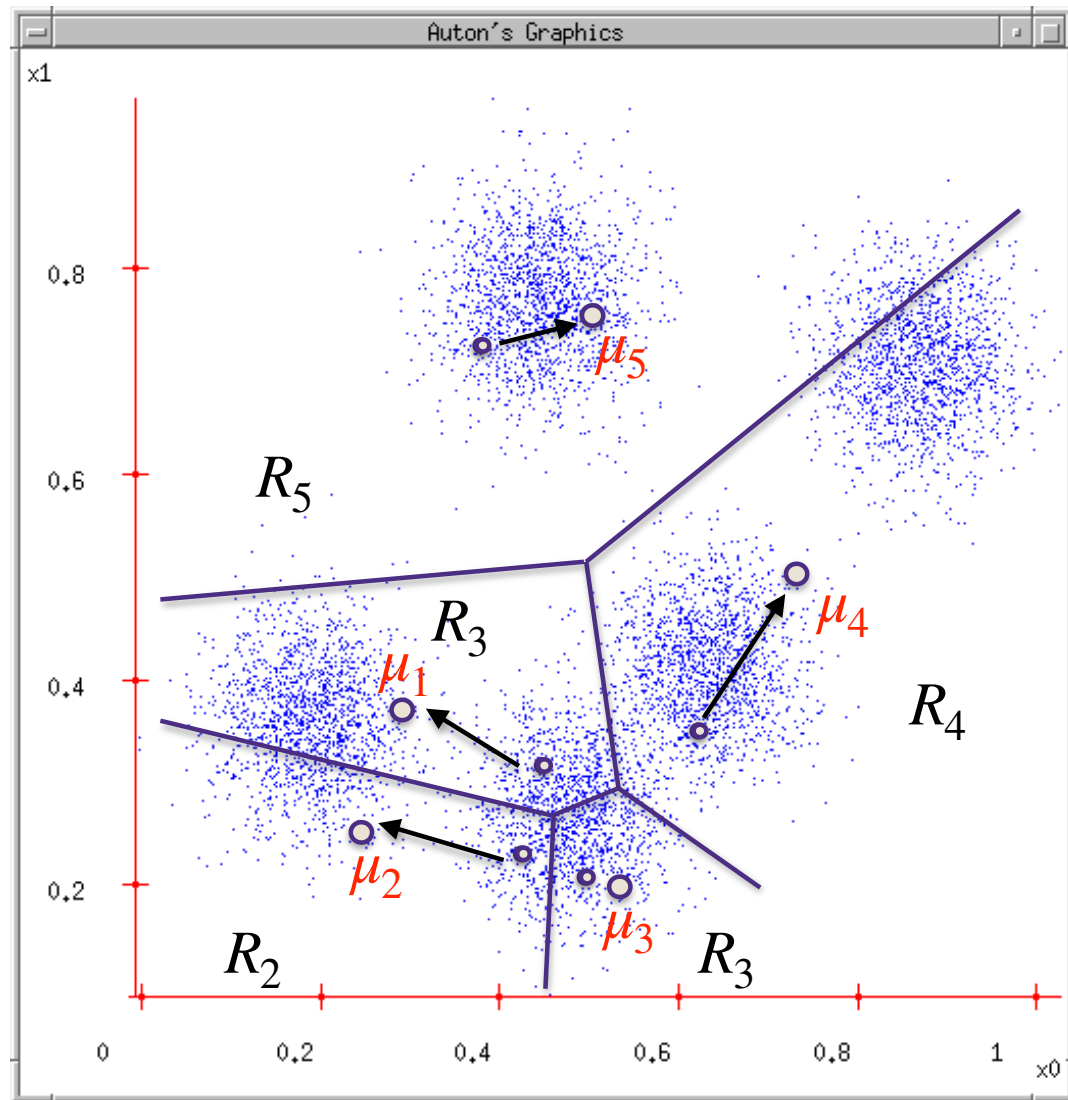
1. Ask user how many clusters they'd like. (e.g.  $k=5$ )
2. Randomly guess  $k$  cluster Center locations  
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.  
(Thus each Center "owns" a set of datapoints)





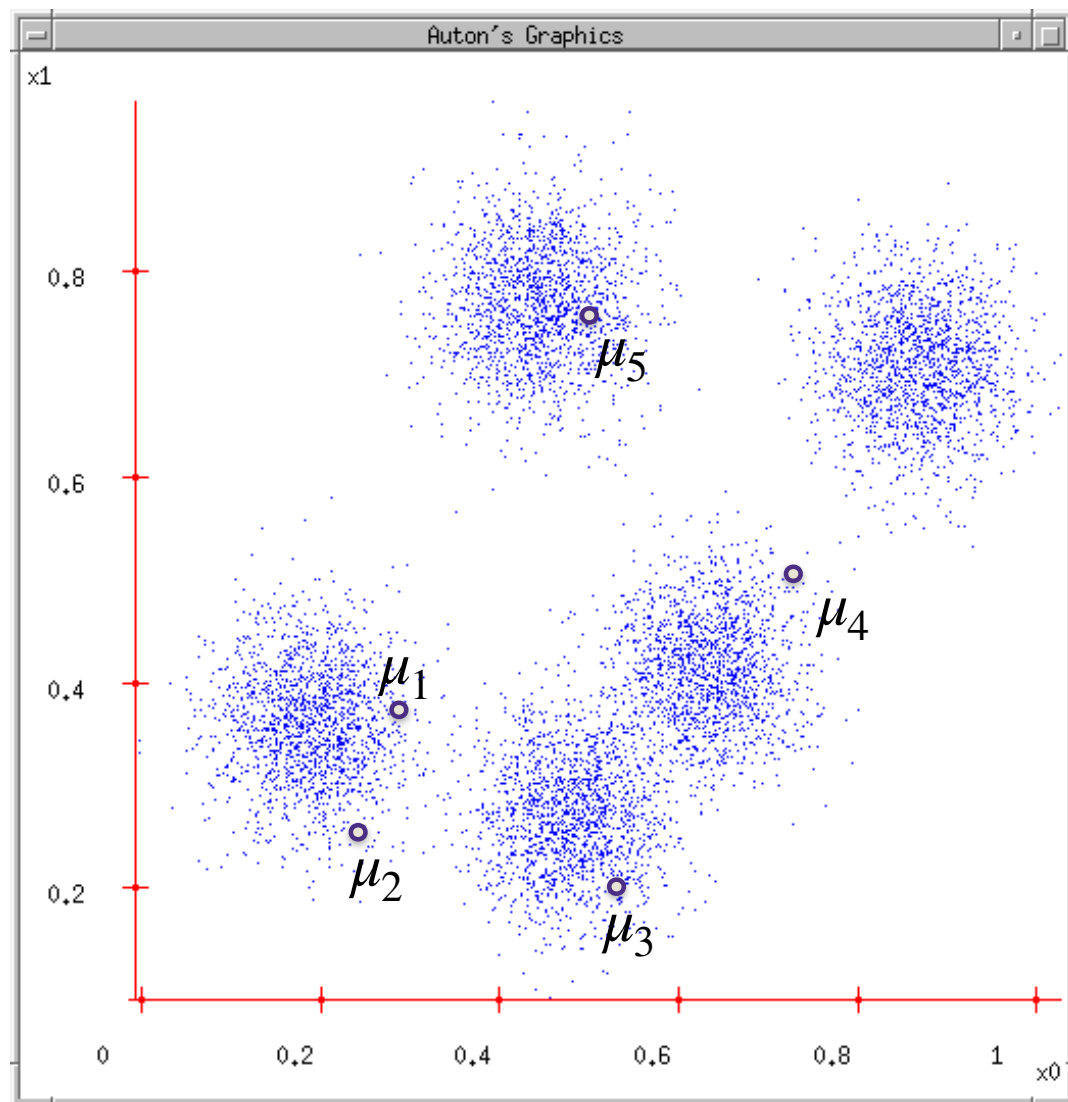
# K-means

1. Ask user how many clusters they'd like. (e.g.  $k=5$ )
2. Randomly guess  $k$  cluster Center locations  
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns



# K-means

1. Ask user how many clusters they'd like. (e.g.  $k=5$ )
2. Randomly guess  $k$  cluster Center locations  
 $\{\mu_1, \dots, \mu_5\}$
3. Each datapoint finds out which Center it's closest to.
4. Each Center finds the centroid of the points it owns...
5. ...and jumps there
6. ...Repeat until terminated!



# K-means

---

> Randomly initialize k centers

- $\mu(0) = \mu_1(0), \dots, \mu_k(0)$

> Classify: Assign each point  $j \in \{1, \dots, N\}$  to nearest center:

- Assignment:  $C^{(t)}(j) \leftarrow \arg \min_i \|\mu_i - x_j\|^2$

> Recenter:  $\mu_i$  becomes centroid of its point:

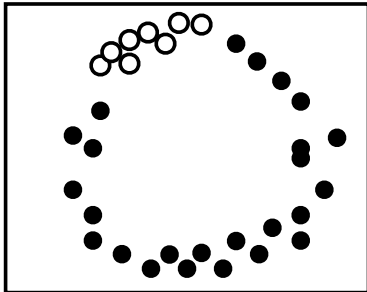
- $$\mu_i^{(t+1)} \leftarrow \arg \min_{\mu} \sum_{j: C(j)=i} \|\mu - x_j\|^2$$

- Equivalent to  $\mu_i \leftarrow$  average of all the points assigned to  $\mu_i$ !

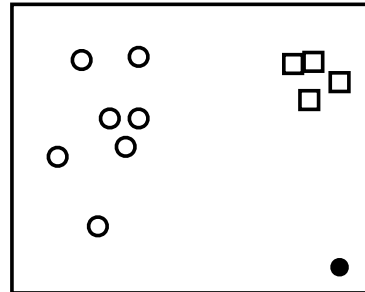
# Which one is a snapshot of a converged $k$ -means

When  $k$ -means is converged, there should be a set of centers and assignments that do not change when applying 1 step of  $k$ -means

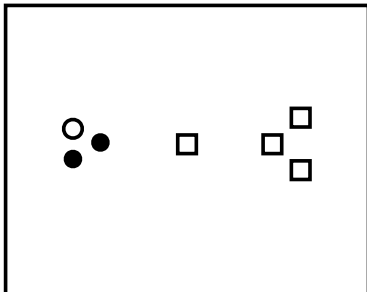
**Example (a)**



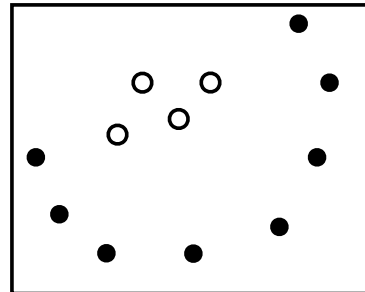
**Example (b)**



**Example (c)**



**Example (d)**



# Does $k$ -means converge??

---

>  $k$ -means is trying to minimize the following objective

> Optimize potential function:

$$\min_{\mu} \min_C F(\mu, C) = \min_{\mu} \min_C \sum_{i=1}^k \sum_{j:C(j)=i} \|\mu_i - x_j\|^2$$

> Via alternating minimization

> Fix  $\mu$ , optimize  $C$

> Fix  $C$  optimize  $\mu$

# Does $k$ -means converge??

---

- there is only a finite set of values that  $\{C(j)\}_{j=1}^n$  can take ( $k^n$  is large but finite)
- so there is only finite,  $k^n$  at most, values for cluster-centers also
- each time we update them, we will never increase the objective

function 
$$\sum_{i=1}^k \sum_{j:C(j)=i} \|x_j - \mu_i\|_2^2$$

- the objective is lower bounded by zero
- after at most  $k^n$  steps, the algorithm must converge (as the assignments  $\{C(j)\}_{j=1}^n$  cannot return to previous assignments in the course of  $k$ -means iterations)

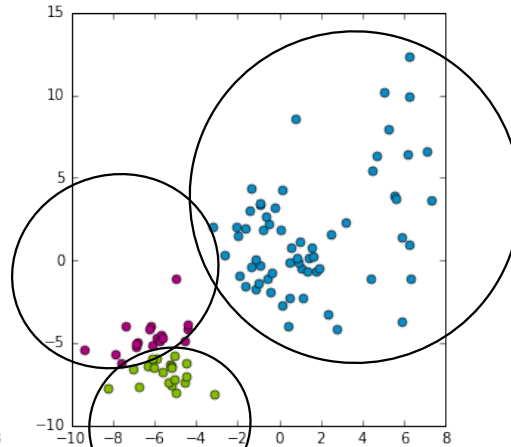
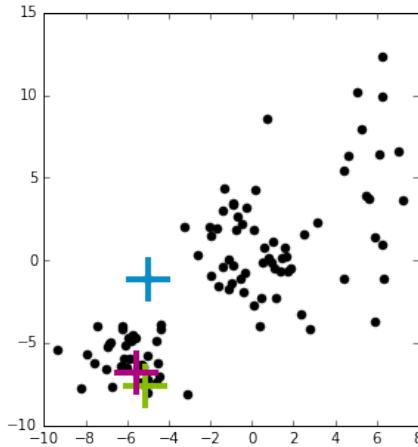
# downsides of $k$ -means

1. it requires the number of clusters  $K$  to be specified by us
2. the final solution depends on the initialization  
(does not find global minimum of the objective)

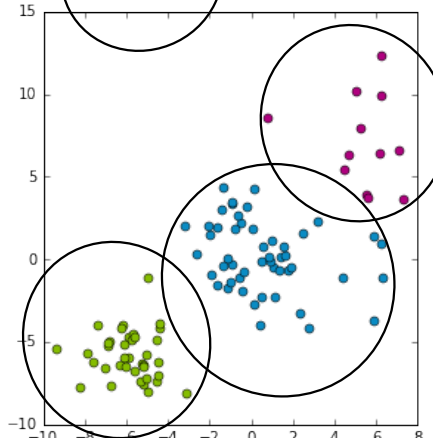
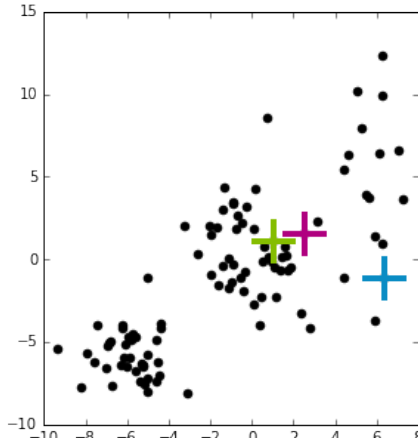
Initial position of centers

final converged assignment

Trial 1



Trial 2



# $k$ -means++: a smart initialization

## Smart initialization:


1. Choose **first** cluster center uniformly at random from data points
  2. Repeat  **$K-1$**  times
    3. For each data point  $\mathbf{x}_i$ , compute distance  $\mathbf{d}_i$  to nearest cluster center
    4. Choose new cluster center from amongst data points, with probability of  $\mathbf{x}_i$  being chosen proportional to  $(\mathbf{d}_i)^2$
- apply standard K-means after the initialization



# Questions?

---

# Lecture 25:

- 
- Unsupervised learning
    - Dimensionality reduction
      - PCA
      - Auto-encoder
    - Clustering
      - $k$ -means
      - **Spectral, t-SNE, UMAP**
    - Generative models
    - Density estimation



# Questions?

---

# Questions?

---