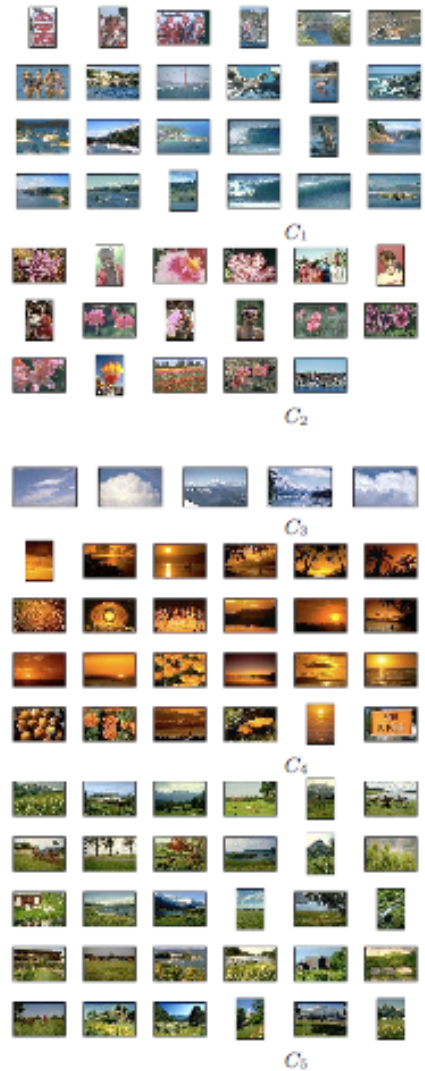
Demo: https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Lecture 24: k-means clustering and spectral clustering

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

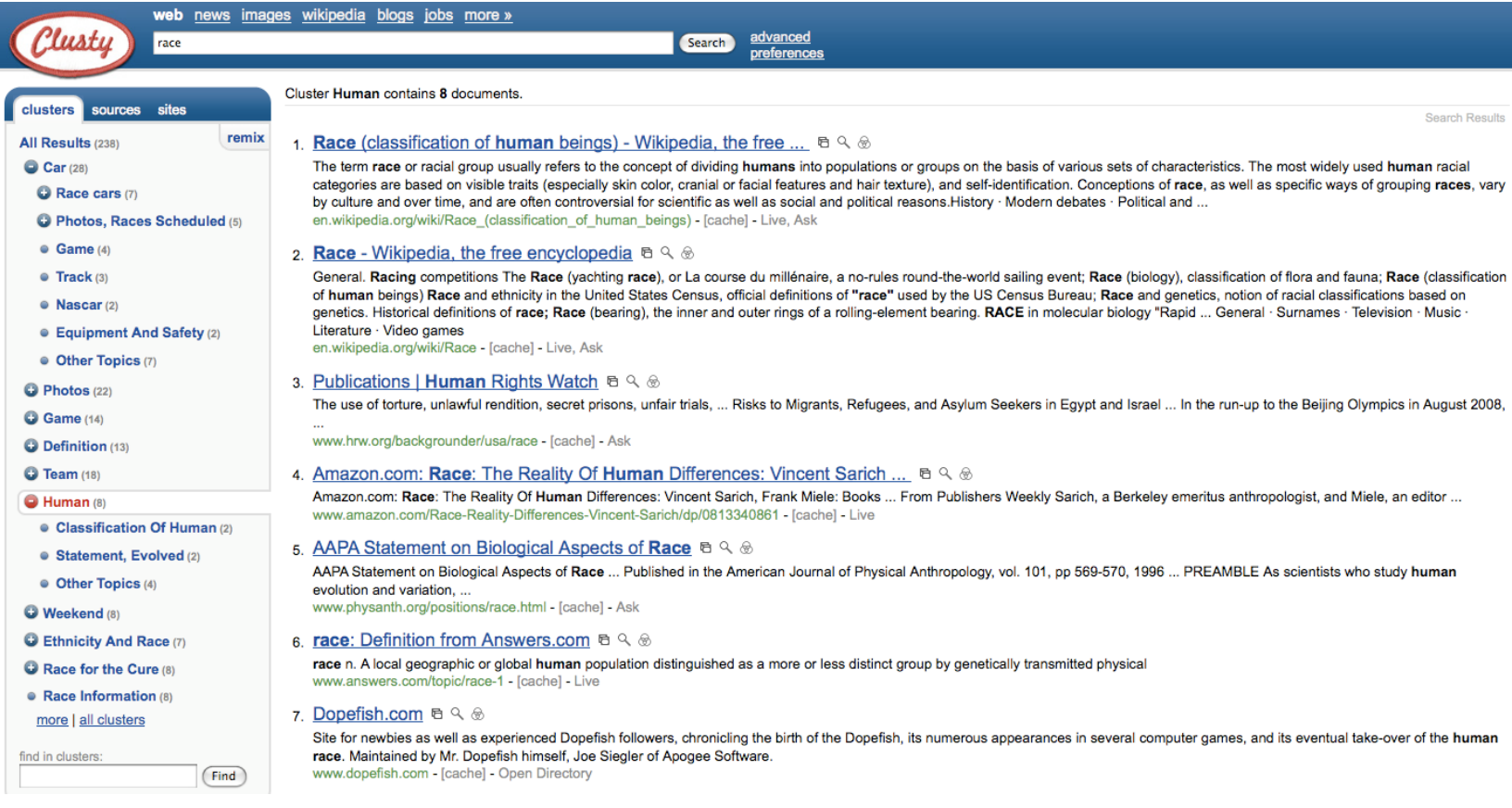
UNIVERSITY *of* WASHINGTON

Clustering images



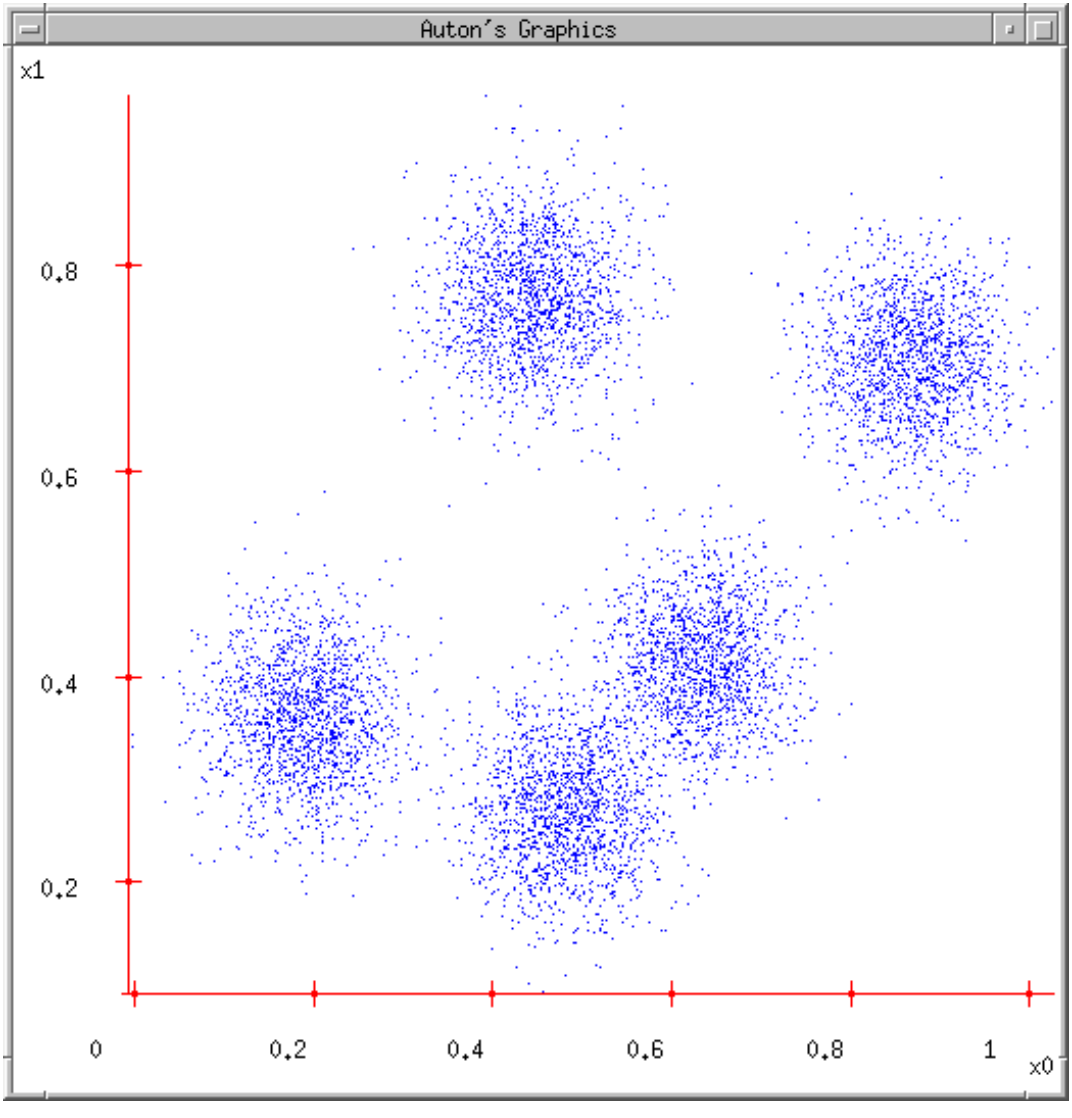
[Goldberger et al.]

Clustering web search results



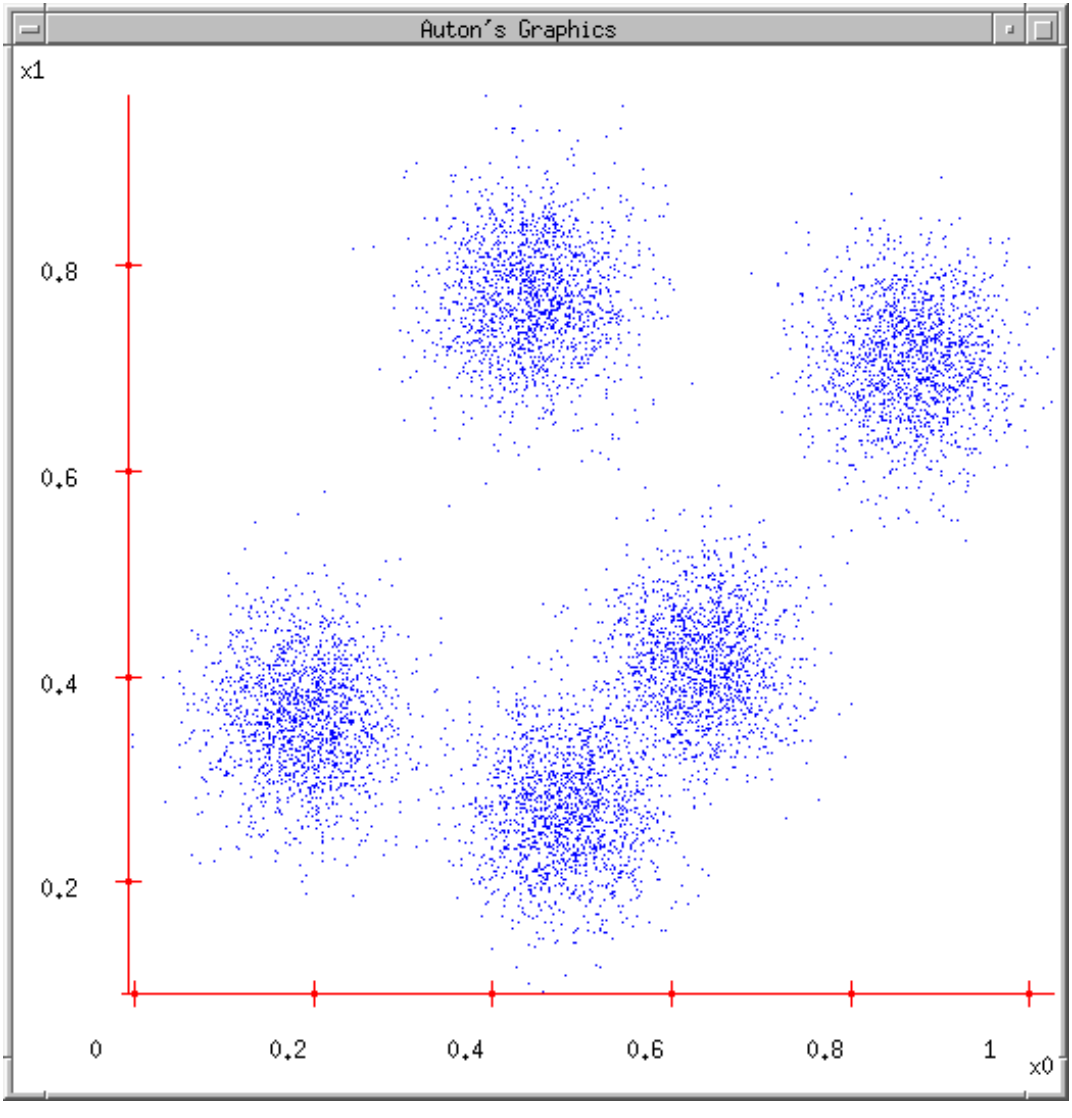
Some Data

- K-mean algorithm assumes this kind of structured data



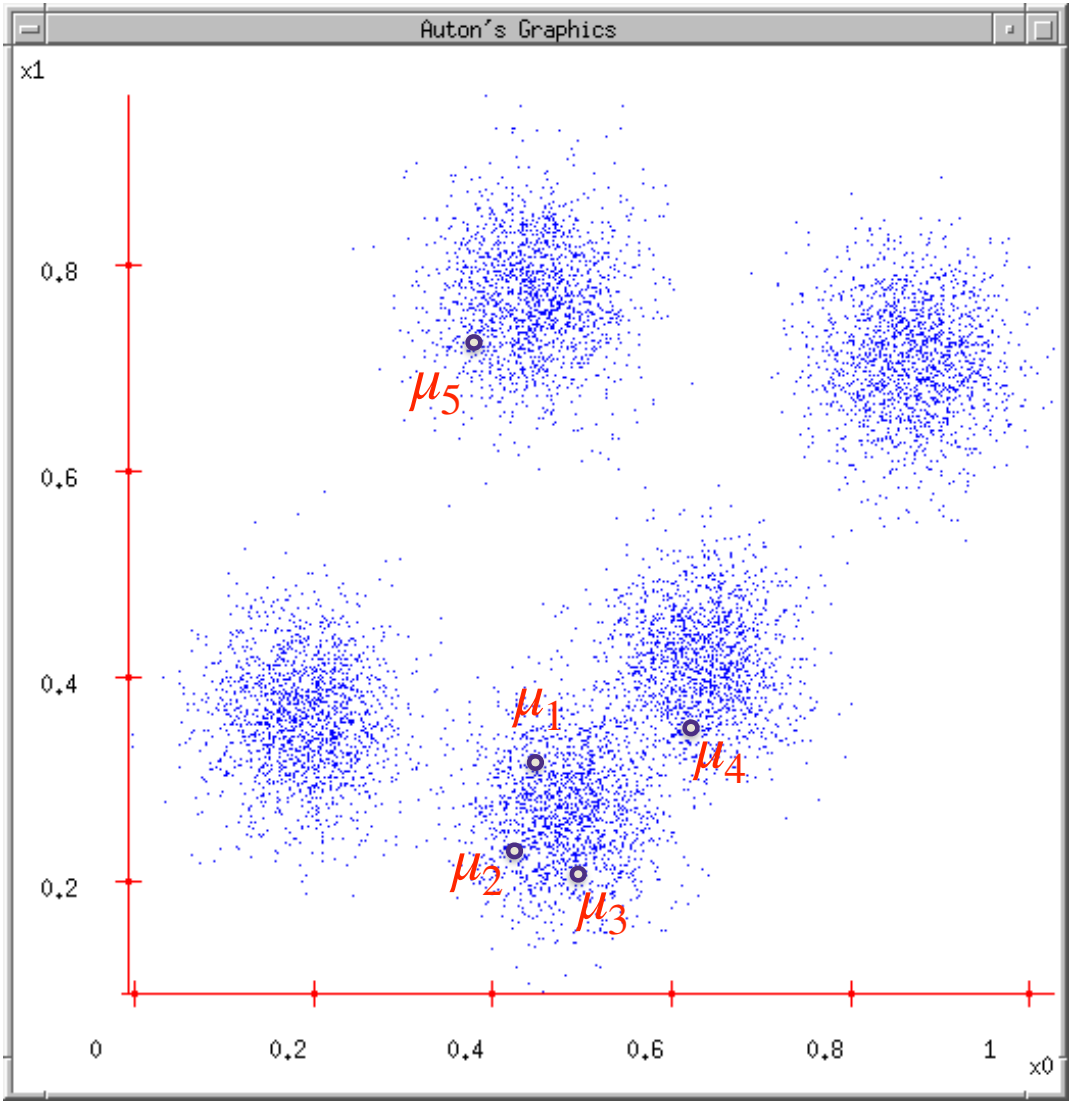
K-means

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



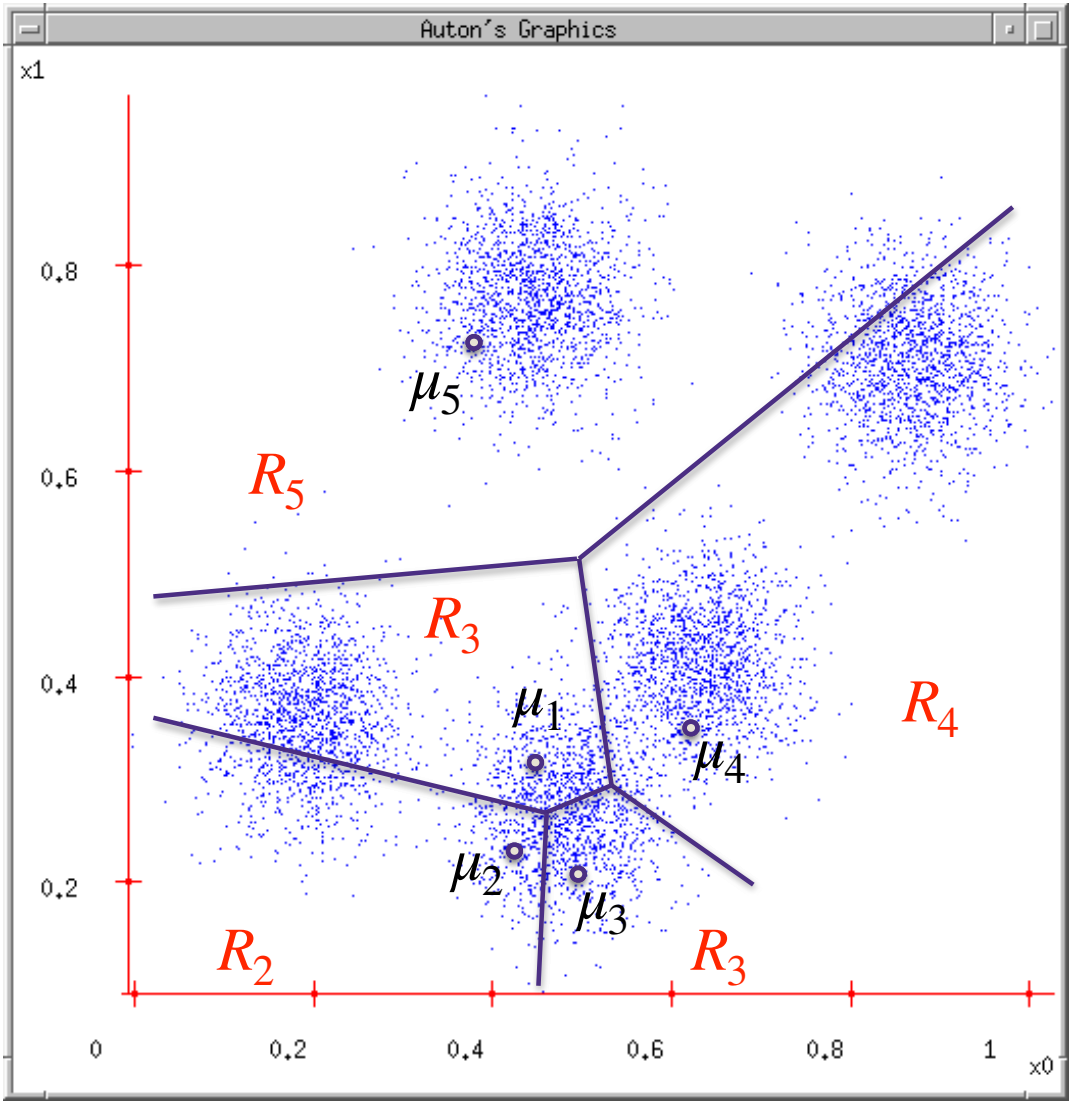
K-means

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



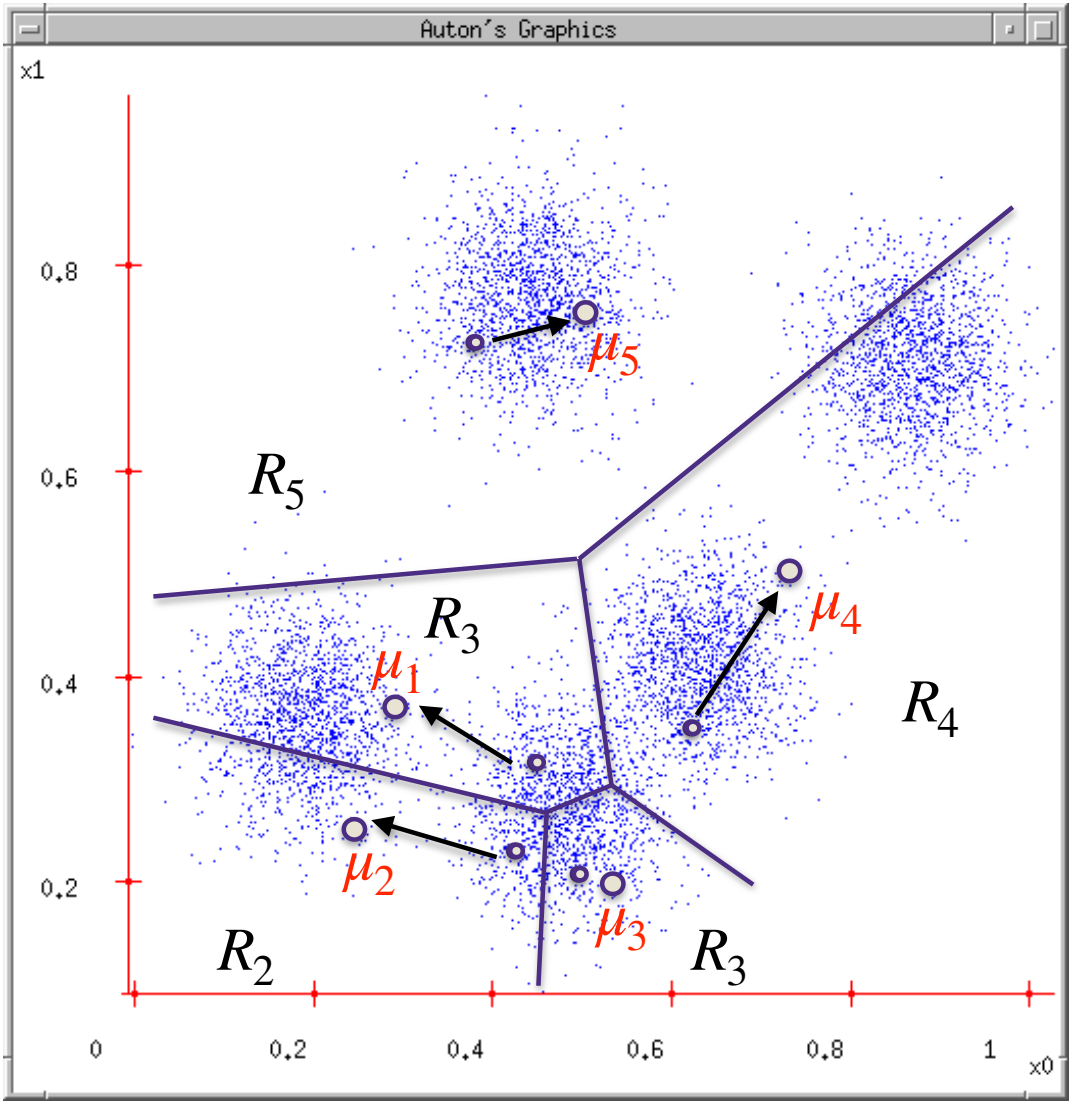
K-means

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



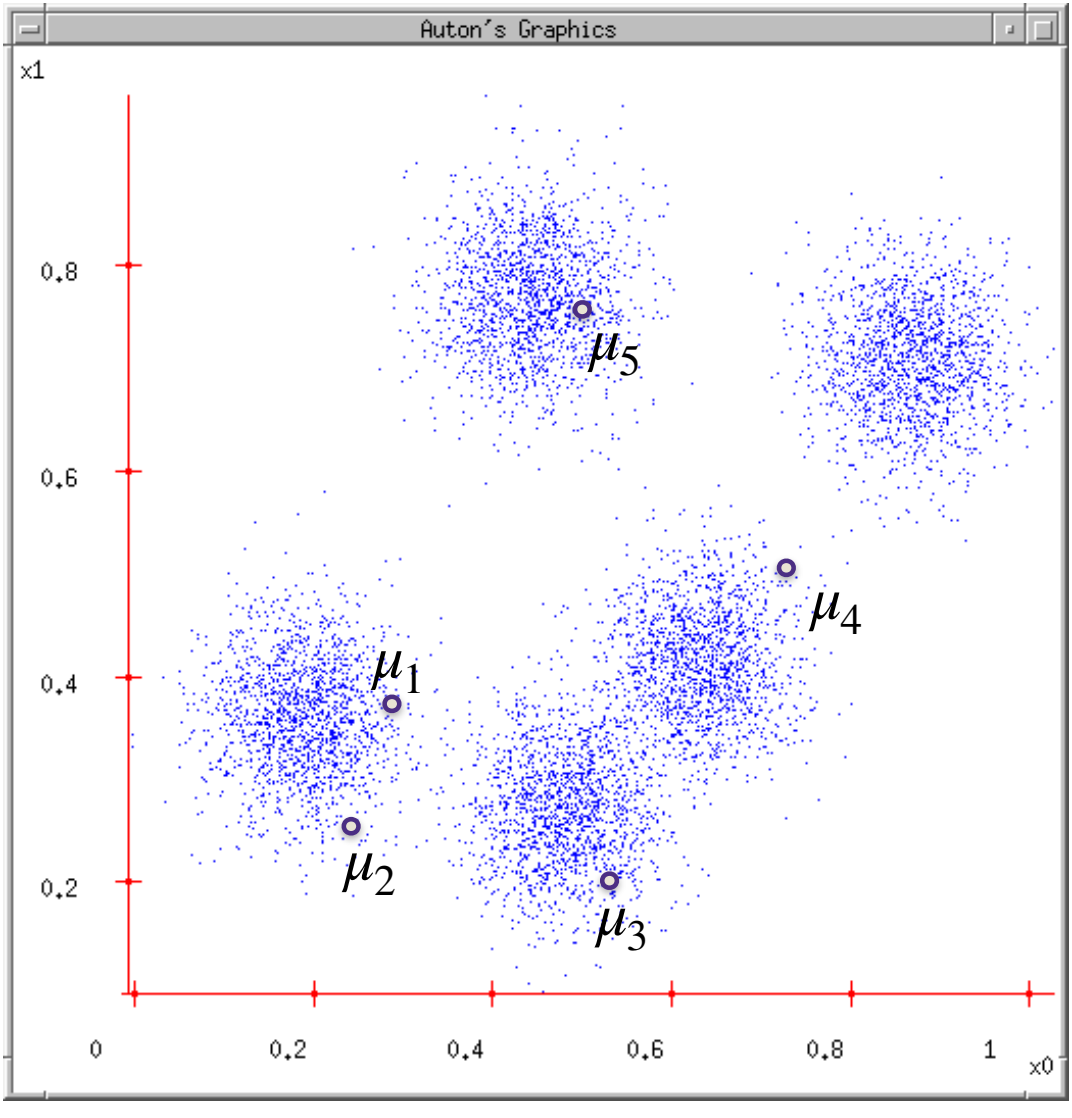
K-means

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

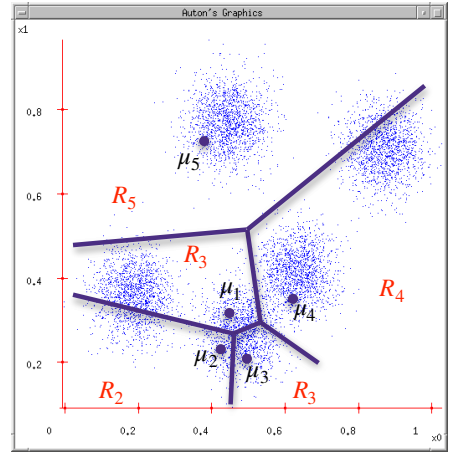
1. Choose k, how many clusters to find
2. Randomly initialize k centers
   - $\mu^{(0)} = [\mu_1^{(0)}, \ldots, \mu_k^{(0)}] \in \mathbb{R}^{d \times k}$
   - Usually randomly chosen from the data points, to make sure they are in the right domain
3. Assign each point j∈{1,…,n} to its nearest center:

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

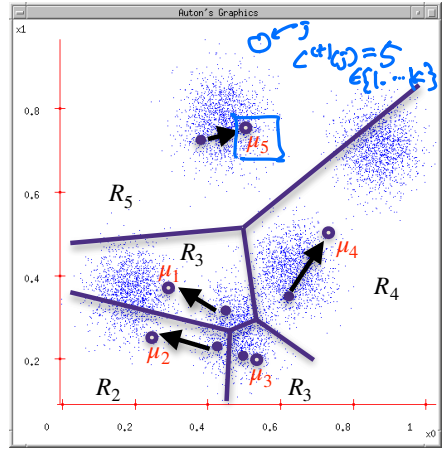
4. Recenter: μ_i becomes centroid of its point:
   - $$\mu_i^{(t+1)} \leftarrow \arg\min_{\boxed{\mu}} \sum_{j:C(j)=i} ||\mu - x_j||^2$$

     $\hookrightarrow \mu_i^* = \sum_{j: C(j)=i} x_j \frac{1}{|C(j)|}$
   - Equivalent to $\mu_i^{(t+1)} \leftarrow$ average of all the points assigned to $\mu_i^{(t)}$
5. Repeat 3-4.

Assignment:

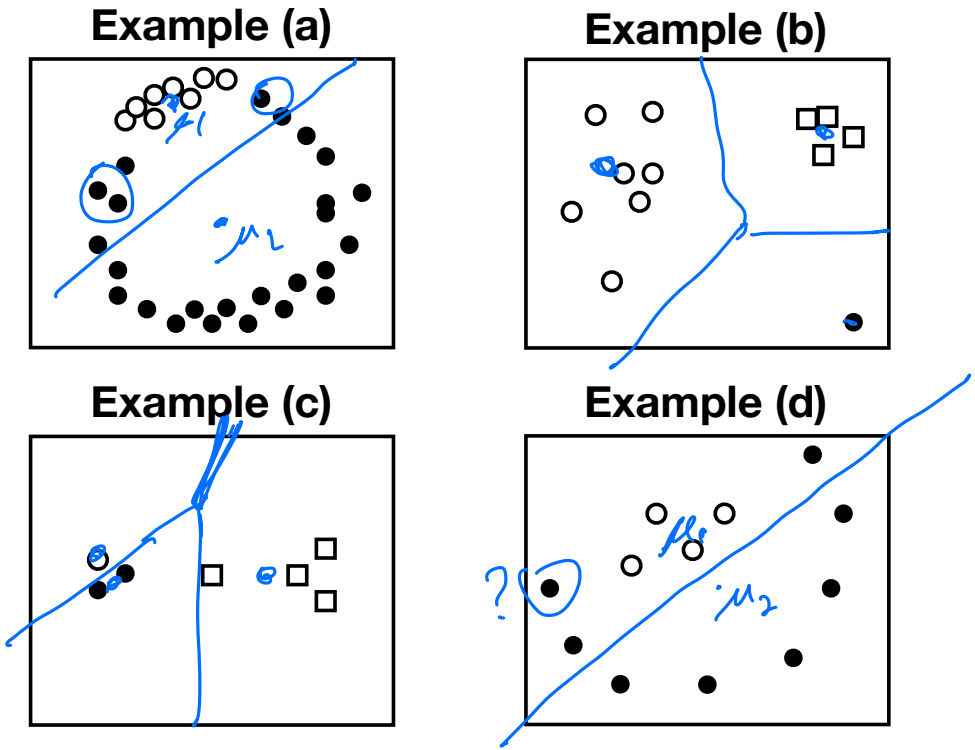


Recenter:



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



Does k-means converge??

$C \in \{1, \cdots, k\}^n$

- k-means is trying to minimize the following objective

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

via alternating minimization
(equivalent to coordinate descent)

  - Fix μ, optimize C
  - Fix C optimize μ

- Does this converge? Does this terminate in finite time?

Does k-means converge??

- there is only a finite set of values that $\{C(j)\}_{j=1}^{n} \in \{1,\ldots,k\}^n$ 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)

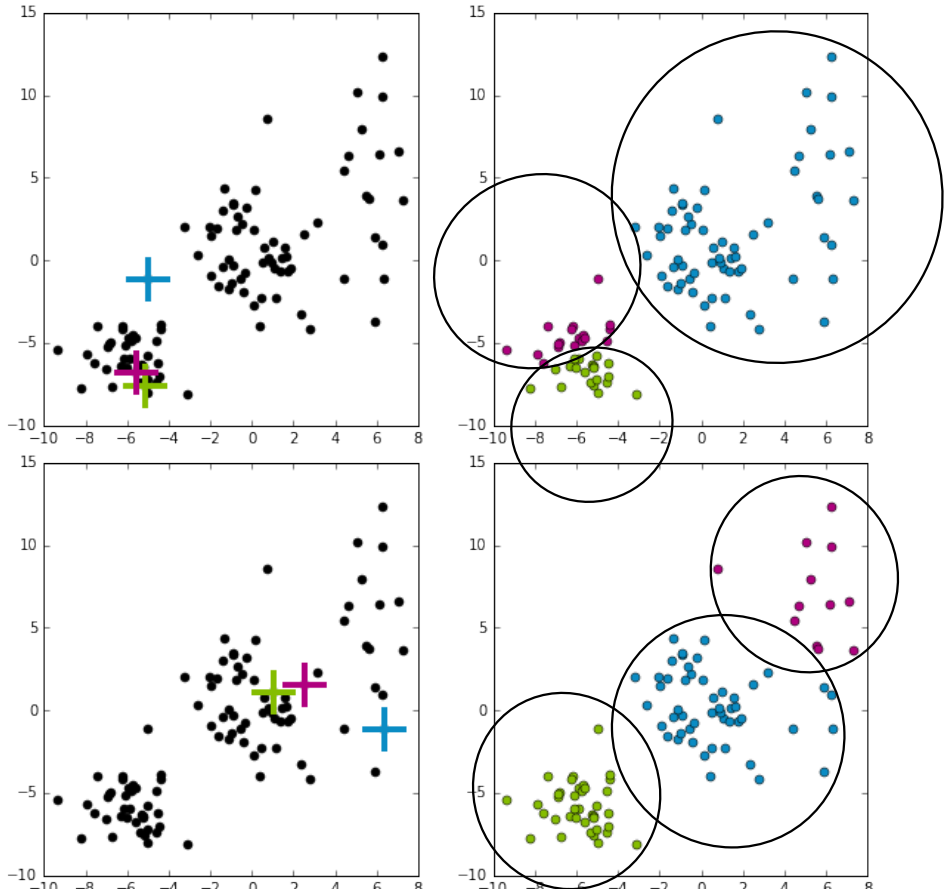
downsides of k-means

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 μ_1 uniformly at random from data points
2. For k=2,… K

   3. For each data point x_i, compute distance d_i to nearest cluster center
   4. Choose new cluster center from amongst data points, with probability

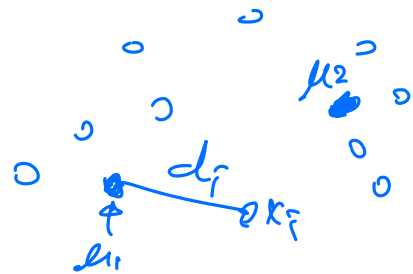
of x_i being chosen proportional to $(d_i)^2$

precisely,

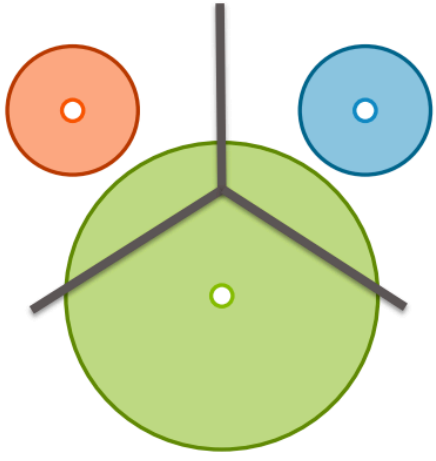
$$d_i \leftarrow \min_{j \in \{1,\ldots,k-1\}} \|\mu_j - x_i\|, \text{ for all } i \text{ that is not chosen already}$$

$$\text{Prob}(x_i \text{ chosen as the next center}) = \frac{(d_i)^2}{\sum_\ell (d_\ell)^2}$$

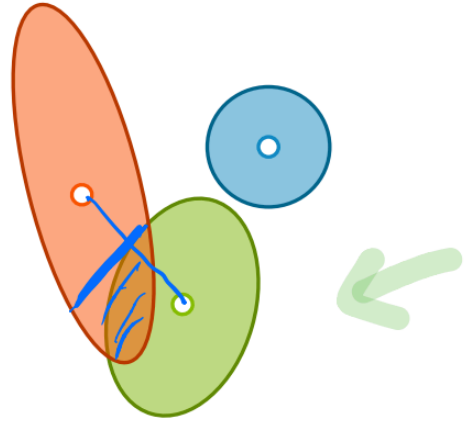
- apply standard K-means after this initialization



- K-means algorithm fails, when



disparate cluster sizes



different shaped/oriented clusters

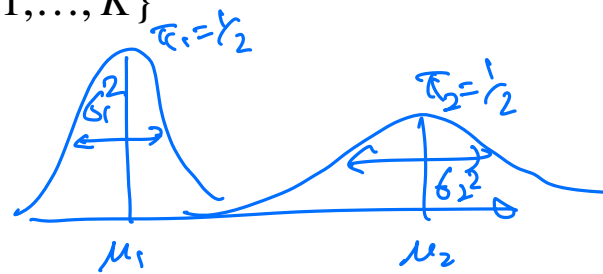
- What can we do?

Gaussian Mixture Model

- input: data $\{x_i\}_{i=1}^n$ in $\mathbb{R}^d$
- parameters of a **Gaussian Mixture Model**
  - mixing weights:
    - $\pi_j = \mathbf{P}(\text{cluster membership} = j)$ for $j \in \{1, \ldots, K\}$

      $[0,1]$
  - means:
    - $\mu_j \in \mathbb{R}^d$ for $j \in \{1, \ldots, K\}$
  - covariance matrices:
    - $\mathbf{C}_j \in \mathbb{R}^{d \times d}$ for $j \in \{1, \ldots, K\}$

$X_i \sim P\begin{pmatrix} \pi_1 \pi_2 \cdots \\ \mu_1 \cdots \mu_K \\ \Sigma_1 \cdots \Sigma_K \end{pmatrix}$



- we suppose that the given data has been generated from a GMM, and try to find the best GMM parameters (this naturally will define clustering of the training data)

- under the GMM, the i-th sample is drawn as follows
  - first sample a cluster $z_i \in \{1, \ldots, K\}$, from $\pi = [\pi_1, \ldots, \pi_K]$
  - conditioned on this cluster, x_i is sampled from

$$x_i \sim N(\mu_{z_i}, \mathbf{C}_{z_i})$$

demo: https://lukapopijac.github.io/gaussian-mixture-model/

Maximum likelihood estimation (MLE)

- we can find the best GMM, by MLE
- for simplicity, suppose $d = 1$ and $K = 2$
- Model parameters are $\pi_1, \pi_2, \mu_1, \mu_2, \mathbf{C}_1, \mathbf{C}_2 \in \mathbb{R}$
- the probability of observing a sample x_i can be written as

$$\mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \mathbf{C}_1, \mathbf{C}_2) \;=\; \pi_1 \underbrace{\frac{1}{\sqrt{2\pi\mathbf{C}_1}} e^{-\frac{(x_i-\mu_1)^2}{2\mathbf{C}_1}}}_{\triangleq\, N(x_i;\mu_1,\mathbf{C}_1)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\mathbf{C}_2}} e^{-\frac{(x_i-\mu_2)^2}{2\mathbf{C}_2}}}_{\triangleq\, N(x_i;\mu_2,\mathbf{C}_2)}$$

- MLE tries to find

$$\arg \max_{\pi_1,\pi_2,\mu_1,\mu_2,\mathbf{C}_1,\mathbf{C}_2} \sum_{i=1}^{n} \log \mathbf{P}(x_i; \pi_1, \pi_2, \mu_1, \mu_2, \mathbf{C}_1, \mathbf{C}_2)$$

- however, unlike least squared or logistic regression, this is not a concave function of the parameters (thus hard to find the optimal solution)
- in general, MLE of a mixture model is not convex/concave optimization

Recall lecture 2: fitting a single Gaussian model

- given $\{x_i\}_{i=1}^n \in \mathbb{R}$,fit the best Gaussian model with mean $\mu \in \mathbb{R}$ and variance $\mathbf{C} \in \mathbb{R}$
- using MLE we want to solve

$$\text{maximize}_{\mu,\mathbf{C}} \quad \mathscr{L}(\mu, \mathbf{C}) \;=\; \sum_{i=1}^{n} \underbrace{\left(-\frac{(x_i - \mu)^2}{2\mathbf{C}} - \log\left(\sqrt{2\pi\mathbf{C}}\right) \right)}_{\log N(x_i|\mu,\mathbf{C})}$$

- we compute gradient and set it to zero:
  - $$\nabla_\mu \mathscr{L}(\mu, \mathbf{C}) \;=\; \frac{1}{\mathbf{C}} \sum_{i=1}^{n} (\mu - x_i)$$

    which is zero for $\boxed{\mu = \frac{1}{n} \sum_{i=1}^{n} x_i}$

    (which makes sense as it is the empirical mean)
  - $$\nabla_{\mathbf{C}} \mathscr{L}(\mu, \mathbf{C}) \;=\; \frac{\sum_{i=1}^{n} (x_i - \mu)^2}{2\mathbf{C}^2} - \frac{n}{2\mathbf{C}}$$

    which is zero for $\boxed{\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)^2}$

    (which makes sense as it is the empirical variance)

MLE for GMM

- we want to fit a model by solving

$$\text{maximize}_{\pi_1,\pi_2,\mu_1,\mu_2,\mathbf{C}_1,\mathbf{C}_2} \sum_{i=1}^{n} \log\Bigg(\pi_1 \underbrace{\frac{1}{\sqrt{2\pi\mathbf{C}_1}} e^{-\frac{(x_i-\mu_1)^2}{2\mathbf{C}_1}}}_{\triangleq N(x_i;\mu_1,\mathbf{C}_1)} + \pi_2 \underbrace{\frac{1}{\sqrt{2\pi\mathbf{C}_2}} e^{-\frac{(x_i-\mu_2)^2}{2\mathbf{C}_2}}}_{\triangleq N(x_i;\mu_2,\mathbf{C}_2)}\Bigg)$$

- define $r_i = \mathbf{P}(z_i = 1 \mid x_i) = \dfrac{\mathbf{P}(z_i = 1, x_i)}{\mathbf{P}(z_i = 1, x_i) + \mathbf{P}(z_i = 2, x_i)}$

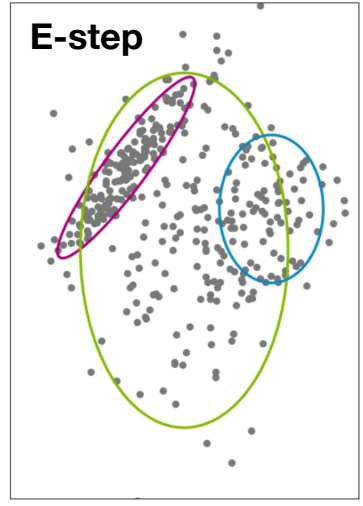
$$= \frac{\pi_1 N(x_i; \mu_1, \mathbf{C}_1)}{\pi_1 N(x_i; \mu_1, \mathbf{C}_1) + \pi_2 N(x_i; \mu_2, \mathbf{C}_2)}$$

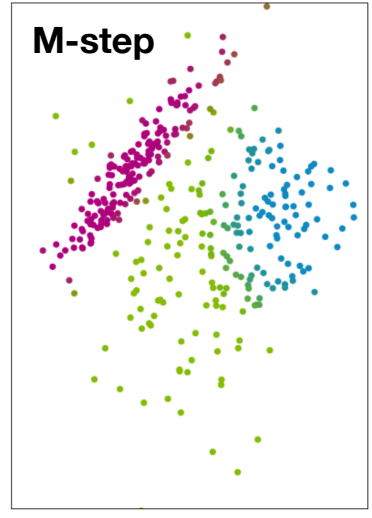
- setting the gradient to zero, we get
  - $\pi_1 = \dfrac{N_1}{n}$ where $N_1 = \sum_{i=1}^{n} r_i$, and $\pi_2 = \dfrac{N_2}{n}$ where $N_2 = \sum_{i=1}^{n} (1 - r_i)$
  - $\mu_1 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i x_i$ and $\mu_2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i) x_i$
  - $\mathbf{C}_1 = \dfrac{1}{N_1} \sum_{i=1}^{n} r_i (x_i - \mu_1)^2$ and $\mathbf{C}_2 = \dfrac{1}{N_2} \sum_{i=1}^{n} (1 - r_i)(x_i - \mu_2)^2$
- both LHS and RHS depend on the parameters, and no closed form solution exists
- **note that if we know r_i's it is trivial to compute parameters, and vice versa**

Expectation Maximization (EM) algorithm

- EM is a popular method to solve MLE for mixture models
- input: training data $\{x_i\}_{i=1}^n$
- output: $\pi_1, \pi_2, \mu_1, \mu_2, \mathbf{C}_1, \mathbf{C}_2 \in \mathbb{R}$

- initialization: randomly initialize the parameters
- repeat
  - **E-step** (Expectation): parameters → soft membership
    - $$r_i = \frac{\pi_1 N(x_i; \mu_1, \mathbf{C}_1)}{\pi_1 N(x_i; \mu_1, \mathbf{C}_1) + \pi_2 N(x_i; \mu_2, \mathbf{C}_2)}$$
  - **M-step** (Maximization): soft membership → parameters
    - $\pi_1 = \frac{N_1}{n}$ where $N_1 = \sum_{i=1}^n r_i$, and $\pi_2 = \frac{N_2}{n}$ where $N_2 = \sum_{i=1}^n (1 - r_i)$
    - $\mu_1 = \frac{1}{N_1} \sum_{i=1}^n r_i x_i$ and $\mu_2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i) x_i$
    - $\mathbf{C}_1 = \frac{1}{N_1} \sum_{i=1}^n r_i (x_i - \mu_1)^2$ and $\mathbf{C}_2 = \frac{1}{N_2} \sum_{i=1}^n (1 - r_i)(x_i - \mu_2)^2$





For general number of clusters K and dimension d

- we can derive EM for general case, in an analogous way
- Initialize parameters: $\pi_1, \ldots, \pi_K, \mu_1, \ldots, \mu_K, \mathbf{C}_1, \ldots, \mathbf{C}_K$

- **E-step**:
  - For k=1,…,K

$$r_{i,k} = \frac{\pi_k \, N(x_i \,|\, \mu_k, \mathbf{C}_k)}{\sum_{j=1}^{K} \pi_j \, N(x_i \,|\, \mu_j, \mathbf{C}_j)}$$

- **M-step**:
  - For k=1,..,K

$$\pi_k = \frac{N_k}{n} \quad \text{where} \quad N_k = \frac{\sum_{i=1}^{n} r_{i,k}}{n}$$

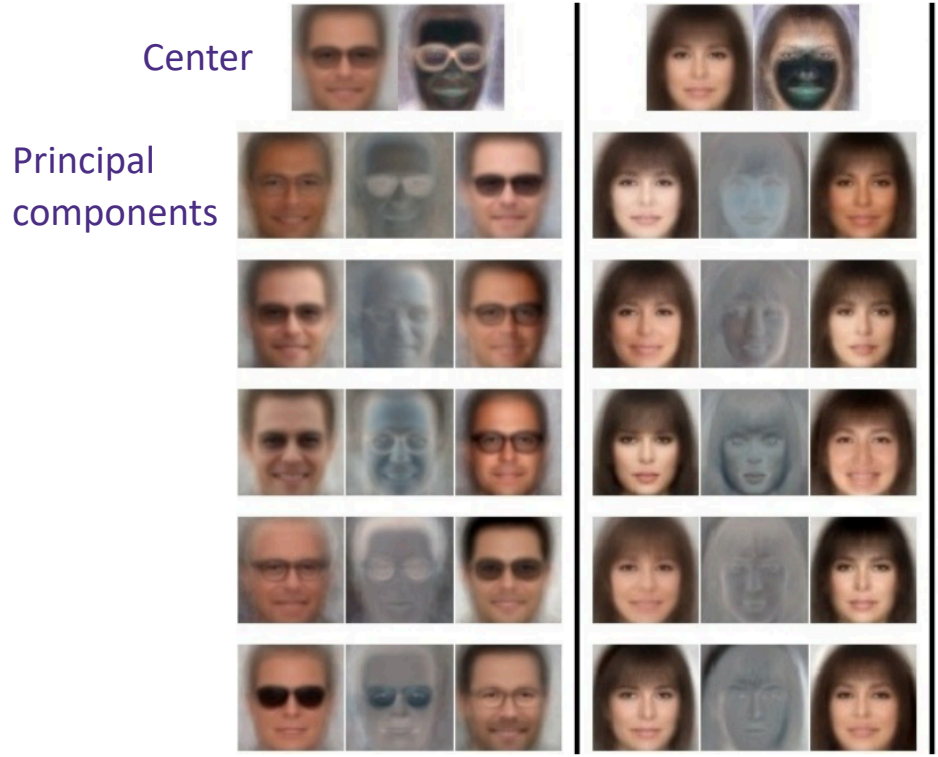
$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} x_i \quad \text{and} \quad \mathbf{C}_k = \frac{1}{N_k} \sum_{i=1}^{n} r_{i,k} (x_i - \mu_k)(x_i - \mu_k)^T$$

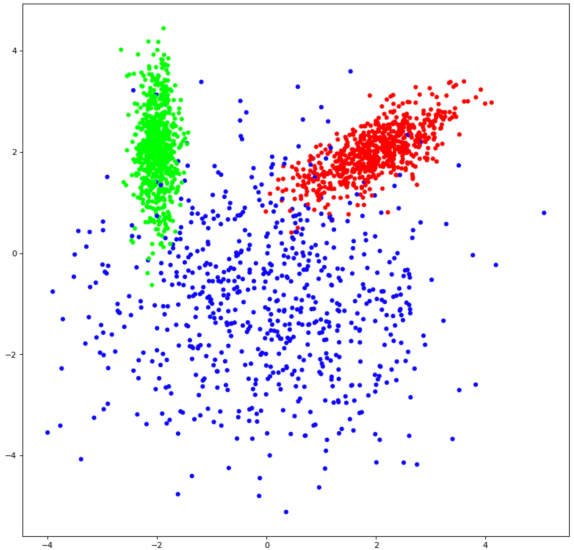
- **once GMM is learned, clustering is straight forward: cluster according to the $r_{i,k}$'s**

GMM for real data

- these are generated samples, from GMM trained on CelebA dataset
- image: 64*64*3=288 dimension
- covariance: restricted to rank-10 matrices
- mixture: K=1,000

Images from “on GANs and GMMs”, 2018, Richardson &Weiss

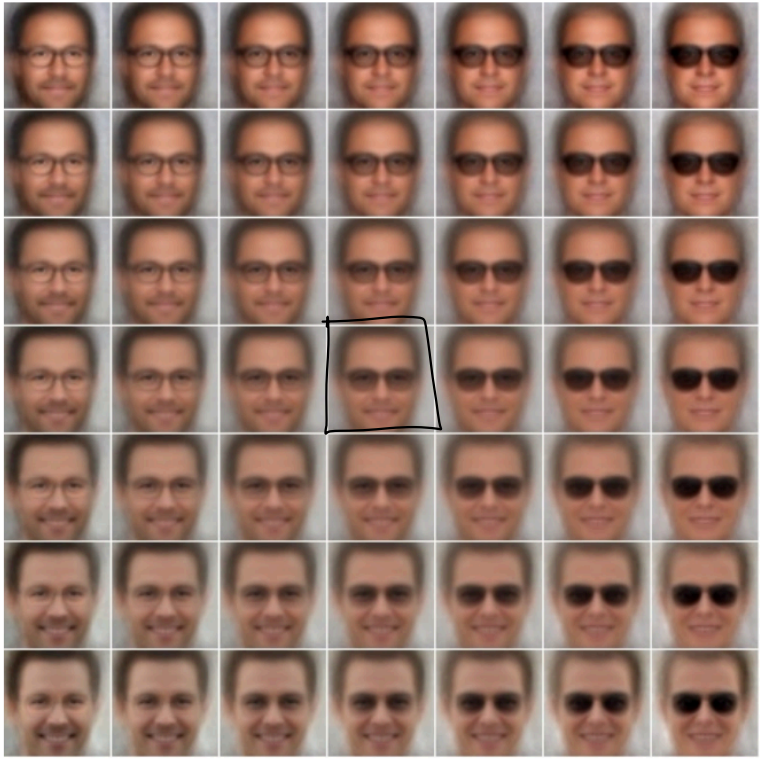




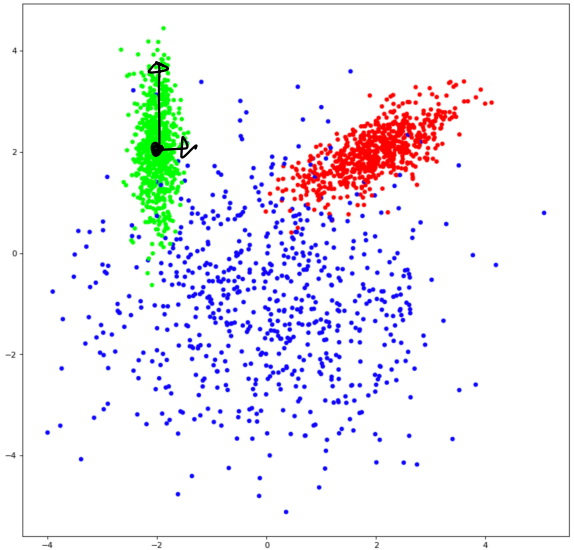
- **top**: center of a cluster μ_k and the diagonal entries of the covariance matrix $\mathbf{C}_k$
- note that we have trained 10-dimensional covariance matrix $\mathbf{C}_k = AA^T$, with $A \in \mathbb{R}^{288\times 10}$, and let $A^{(j)}$ be the j-th column
- **bottom**: each row corresponds to different j, and we show $\mu_k + A^{(j)},\ \ 0.5 + A^{(j)},\ \ \mu_k - A^{(j)}$

Images from "on GANs and GMMs", 2018, Richardson &Weiss

Principal Component direction 2



Principal Component direction 1



- middel: μ_k
- Each row: middel + $c \times A^{(1)}$
- Each column: middle + $c \times A^{(2)}$

Images from "on GANs and GMMs", 2018, Richardson &Weiss

Mixture model for documents

- Input: n documents $\{x_i\}_{i=1}^n$
- Each document is a sequence of words of length T
  $x_i = (w_1, w_2, \ldots, w_T)$
- Bag-of-words model:
  - parameters:
    - mixing weights: $\pi_k = \mathbf{P}(\text{topic} = k)$ for $k \in \{1, \ldots, K\}$
    - word probability: $b_{wk} = \mathbf{P}(\text{word} = w \,|\, \text{topic} = k)$
  - the generative model
    - first sample topic from $\pi = (\pi_1, \ldots, \pi_K)$
    - next sample T words i.i.d. from $b_k = (b_{w_1 k}, \ldots, b_{w_{200,000} k})$
  - to make the problem tractable, this completely ignores the order of the words in the document (but still very successful in document clustering)

$$\mathbf{P}(\text{topic } z_i = k, x_i = (w_1, \ldots, w_T)) = \pi_k b_{w_1 k} \cdots b_{w_T k}$$

Topic modeling

- to fit a topic model, we solve the following

$$\text{maximize}_{b \in \mathbb{R}^{K \times T}, \pi \in \mathbb{R}^K} \sum_{i=1}^{n} \log \mathbf{P}(x_i \,|\, b, \pi)$$

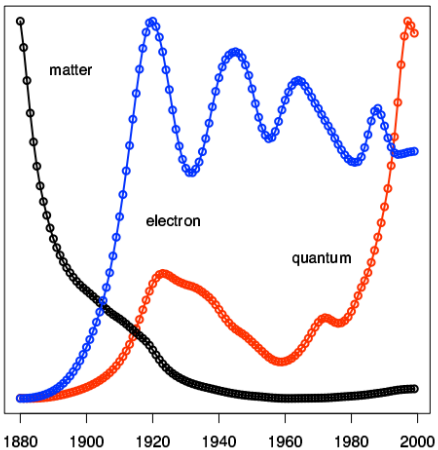
- we can apply EM algorithm
- initialize b, π
- **E-step**: parameters → soft assignments
  - $r_{ik} \;=\; \mathbf{P}(\text{topic } z_i = k \,|\, x_i) = \dfrac{\pi_k b_{w_1 k} \cdots b_{w_T k}}{\sum_{k'=1}^{K} \pi_{k'} b_{w_1 k'} \cdots b_{w_T k'}}$

- **M-step**: soft assignments → parameters
  - $\pi_k \;=\; \dfrac{N_k}{n}$ where $N_k \;=\; \sum_{i=1}^{n} r_{ik}$
  - $b_{wk} \;=\; \dfrac{1}{N_k} \sum_{i=1}^{n} r_{ik} \dfrac{\text{Count}(w \text{ in } x_i)}{T}$

Dynamic topic modeling (over time)

1881	1890	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000
force	motion	magnet	force	atom	ray	energy	energy	radiat	electron	electron	electron	state
energy	force	electric	magnet	theory	measure	measure	radiat	energy	energy	energy	atom	energy
motion	magnet	measure	theory	electron	energy	electron	ray	electron	atom	particle	energy	electron
differ	energy	force	electric	energy	theory	light	electron	measure	measure	field	structur	magnet
light	measure	theory	atom	measure	light	atom	measure	ray	radiat	radiat	field	field
measure	differ	system	system	ray	wave	particle	atom	atom	field	model	model	atom
magnet	direct	motion	measure	electr	radiat	ray	particle	field	ray	atom	state	system
direct	line	line	line	line	atom	radiat	two	two	model	two	two	two
matter	result	point	energy	force	electric	point	light	particle	particle	ray	magnet	quantum
result	light	differ	body	value	value	theory	absorpt	observe	magnet	measure	ray	physic

"Atomic Physics"



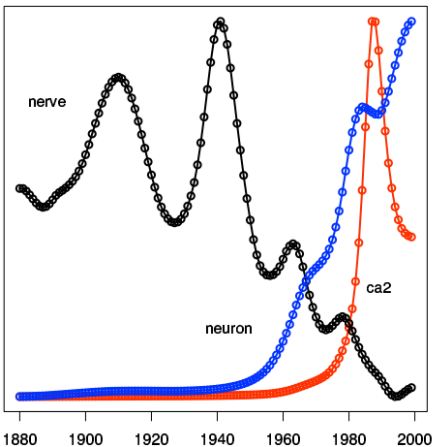
1881 On Matter as a form of Energy
1892 Non-Euclidean Geometry
1900 On Kathode Rays and Some Related Phenomena
1917 ``Keep Your Eye on the Ball''
1920 The Arrangement of Atoms in Some Common Metals
1933 Studies in Nuclear Physics
1943 Aristotle, Newton, Einstein. II
1950 Instrumentation for Radioactivity
1965 Lasers
1975 Particle Physics: Evidence for Magnetic Monopole Obtained
1985 Fermilab Tests its Antiproton Factory
1999 Quantum Computing with Electrons Floating on Liquid Helium

From "Dynamic Topic Models" Blei & Lafferty 2006

Dynamic topic modeling (over time)

1881	1890	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000
brain	movement	brain	movement	movement	stimulate	record	respons	response	respons	cell	cell	neuron
movement	eye	eye	brain	sound	muscle	nerve	record	stimulate	cell	neuron	channel	active
action	right	movement	sound	muscle	sound	stimulate	stimulate	record	potential	response	neuron	brain
right	hand	right	nerve	active	movement	response	nerve	condition	stimul	active	ca2	cell
eye	brain	left	active	nerve	response	muscle	muscle	active	neuron	brain	active	fig
hand	left	hand	muscle	stimulate	nerve	electrode	active	potential	active	stimul	brain	response
left	action	nerve	left	fiber	frequency	active	frequency	stimulus	nerve	muscle	receptor	channel
muscle	muscle	vision	eye	reaction	fiber	brain	electrode	nerve	eye	system	muscle	receptor
nerve	sound	sound	right	brain	active	fiber	potential	subject	record	nerve	respons	synapse
sound	experiment	muscle	nervous	response	brain	potential	study	eye	abstract	receptor	current	signal

"Neuroscience"



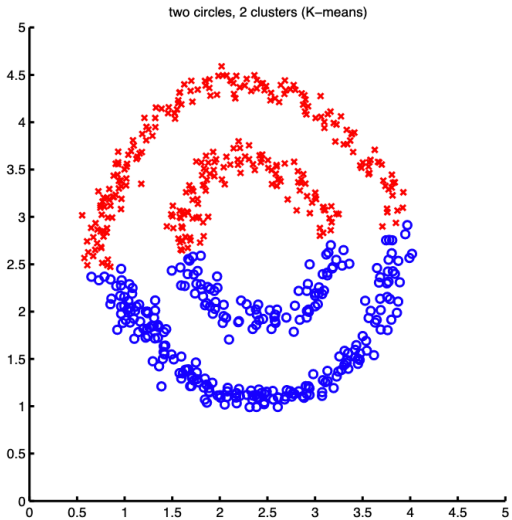
1887 Mental Science
1900 Hemianopsia in Migraine
1912 A Defence of the ``New Phrenology"
1921 The Synchronal Flashing of Fireflies
1932 Myoesthesis and Imageless Thought
1943 Acetylcholine and the Physiology of the Nervous System
1952 Brain Waves and Unit Discharge in Cerebral Cortex
1963 Errorless Discrimination Learning in the Pigeon
1974 Temporal Summation of Light by a Vertebrate Visual Receptor
1983 Hysteresis in the Force-Calcium Relation in Muscle
1993 GABA-Activated Chloride Channels in Secretory Nerve Endings

From "Dynamic Topic Models" Blei & Lafferty 2006

k-means and GMMs are inherently linear

- It tries to find linear boundaries between centers
- It fails completely on non-linearly clustered datasets such as

K-means

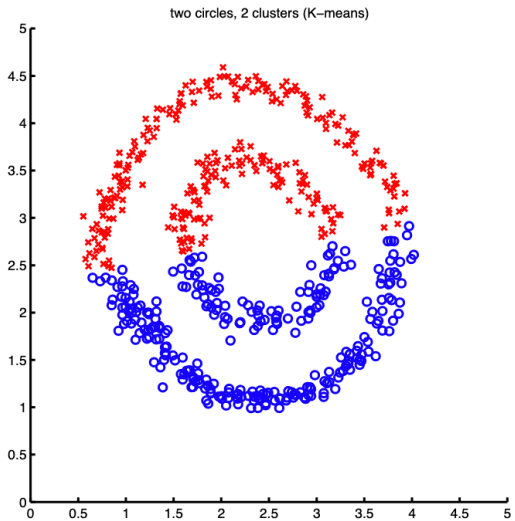


[Shi,Malik,'00],[Ng,Jordan,Weiss,'01]

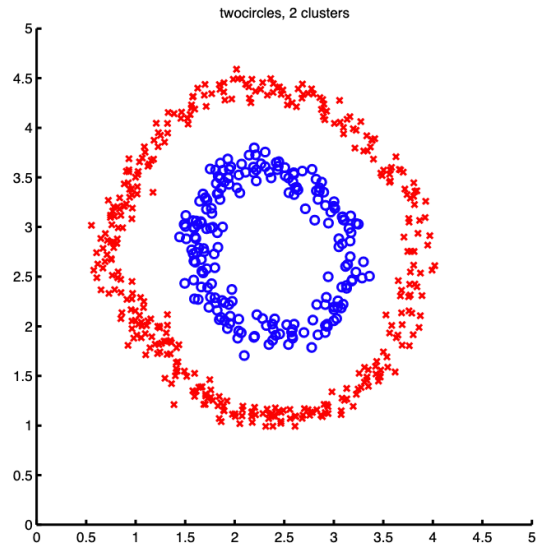
Spectral clustering

- Main idea:
  - Transform the dataset into a graph
  - Use eigenvalues (also called spectrum) and vectors of a graph to cluster

K-means



Spectral clustering



[Shi,Malik,'00],[Ng,Jordan,Weiss,'01]

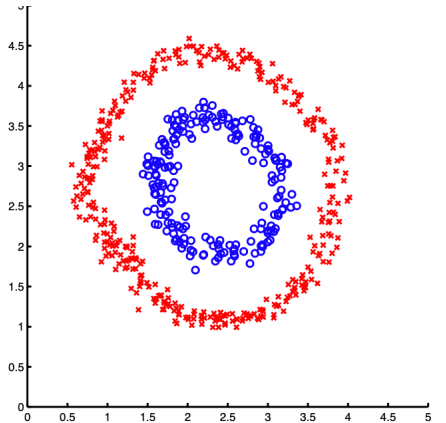
Step 1. From dataset to a graph

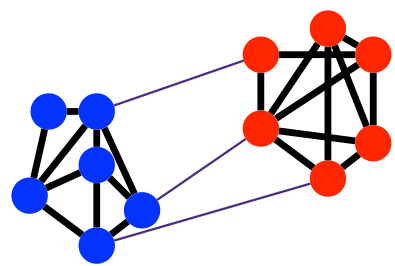
- Given $\mathscr{D} = \{x_i \in \mathbb{R}^d\}_{i=1}^n$, create a graph with n nodes and weighted edges $\{w_{ij}\}$, where each node represents each sample and each edge measures the similarity between the two nodes
  - Example 1: Gaussian kernel

$$w_{ij} = e^{-\frac{\|x_i - x_j\|_2^2}{\sigma^2}}$$

  - Example 2: k-nearest neighbor graph

$$w_{ij} = 1 \text{ if } j \text{ is one of } k\text{-nearest neighbors of } i \text{ or}$$
$$i \text{ is one of } k\text{-nearest neighbors of } j$$

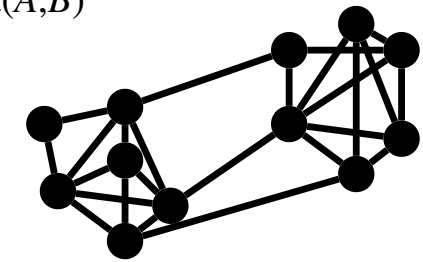




Step 2. Graph partitioning

- Once we have a similarity graph, how do we partition it?
- Can we use **minimum cut** for a graph $G(V, E)$?
  - Set of nodes $V = \{1, \ldots, n\}$
  - Set of edges $E = \{(i, j)\}$
  - If it is a weighted graph we have weights $\{w_{ij}\}_{(i,j) \in E}$
- **Minimum cut** of a graph is a partition $A \cup B = V$ and $A \cap B = \emptyset$ such that

$$\arg\min_{A,B} \underbrace{\sum_{i \in A} \sum_{j \in B} w_{i,j}}_{\text{cut}(A,B)}$$



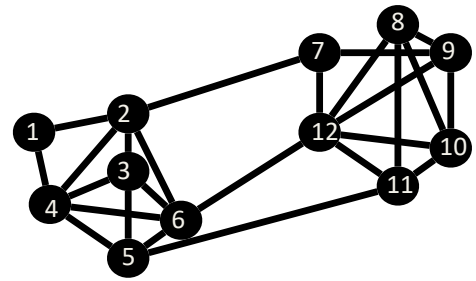
Step 2. Graph partition using Graph Laplacian

- Definitions (we will define it for unweighted graphs, but everything naturally generalizes to weighted graphs)
  - **Adjacency matrix** of a graph $A \in \mathbb{R}^{n \times n}$

    $A_{ij} = 1$ if $(i, j) \in E$

    0 otherwise
  - **Degree** of a node i, is $d_i = \sum_{j=1}^{n} A_{ij}$, which is number of edges connected to node i
  - Define $D \in \mathbb{R}^{n \times n}$ as a diagonal matrix with the degrees of each node in the diagonal
  - The **Graph Laplacian** of a graph is defined as

$$L_G = D - A$$

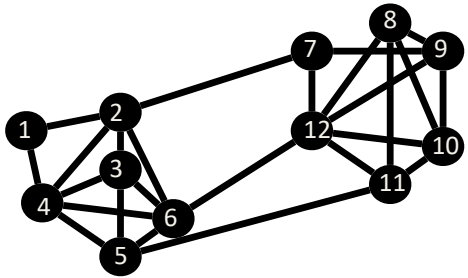


$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 \end{bmatrix}$$

Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line
  at positions $x = [x_1, x_2, \ldots, x_2]$

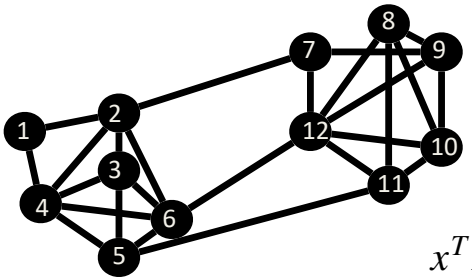


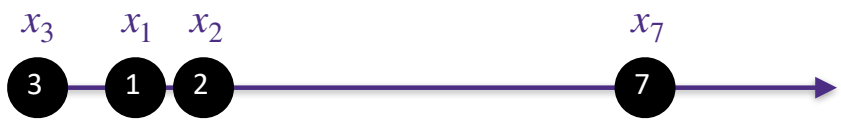
quadratic form of L_G is useful in capturing the structure of the graph:

$$
\begin{aligned}
x^T L_G x &= \sum_i d_i x_i^2 - \sum_{(i,j)\in E} 2x_i x_j \\
&= \sum_i \sum_{j:(i,j)\in E} x_i^2 - \sum_{(i,j)\in E} 2x_i x_j \\
&= \sum_{(i,j)\in E} 2x_i^2 - 2x_i x_j \\
&= \sum_{(i,j)\in E} x_i^2 + x_j^2 - 2x_i x_j \\
&= \sum_{(i,j)\in E} (x_i - x_j)^2
\end{aligned}
$$

Step 2. Graph partition using Graph Laplacian

- Graph Laplacian $L_G = D - A$ can capture some structure of the graph
- Consider placing each node in 1-dim line at positions $x = [x_1, x_2, \ldots, x_2]$





$$x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

- If we want a good graph partition, we want to place nodes such that the distance between connected nodes are smaller
- This naturally leads to the following problem:

$$\arg\min_{x\in\mathbb{R}^n} x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2$$

- There is a trivial solution to this problem: $x_i = 1$ for all i, which achieves the minimum value of zero, so we change it to

$$\arg\min_{x\in\mathbb{R}^n} x^T L_G x = \sum_{(i,j)\in E} (x_i - x_j)^2 \qquad \text{subject to } x^T\mathbf{1} = 0$$

Step 2. Graph partition using Graph Laplacian

- To solve graph partitioning, we solve

$$\arg\min_{x \in \mathbb{R}^n} \; x^T L_G x = \sum_{(i,j) \in E} (x_i - x_j)^2$$

$$\text{subject to } x^T \mathbf{1} = 0$$
$$\|x\|_2 = 1$$

  and place nodes as per x, and find a partition using simple algorithms like k-means
- It turns out that the above optimization has a efficient solver, because The optimal x turns out to be the second smallest eigen vector of the graph Laplacian L_G
- Since, eigen values of a matrix is also called a spectrum, this is called a spectral clustering algorithm

Spectral clustering

- Step 1. Define a similarity graph $G(V, E, W)$
- Step 2. Compute the Graph Laplacian

$$L_G = D - W$$

  where $D$is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} w_{ij}$

  - let x be the Eigen vector corresponding to the second smallest Eigen value
  - Place samples according to x and apply k-means clustering

- instead of using just the second smallest Eigen pair, you can use multiple smallest Eigen pairs

Questions?

Deep Generative Models

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

Deep generative model

- traditional parametric generative model
  - Gaussian:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

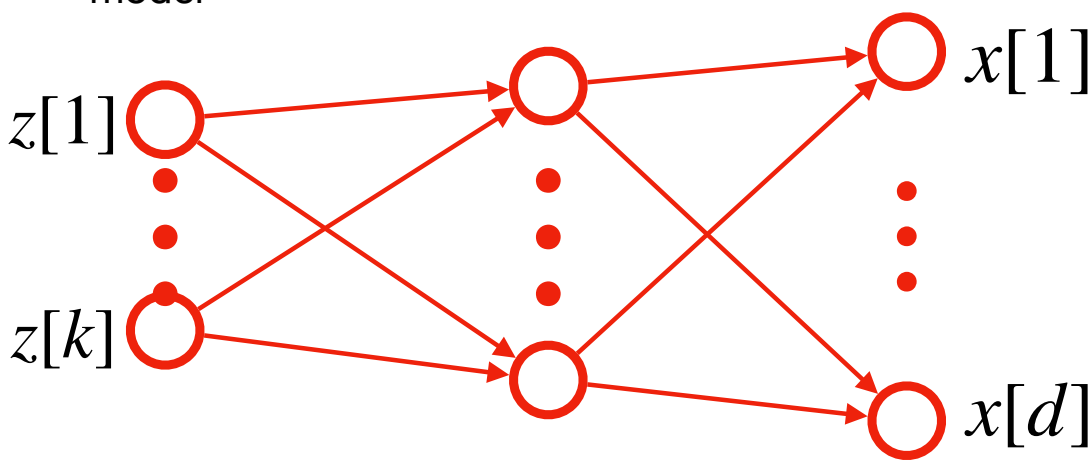
  - Gaussian Mixture Models (GMM)

$$f_{\{\mu_i\},\{\sigma_i\},\{\pi_i\}}(x) = \sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

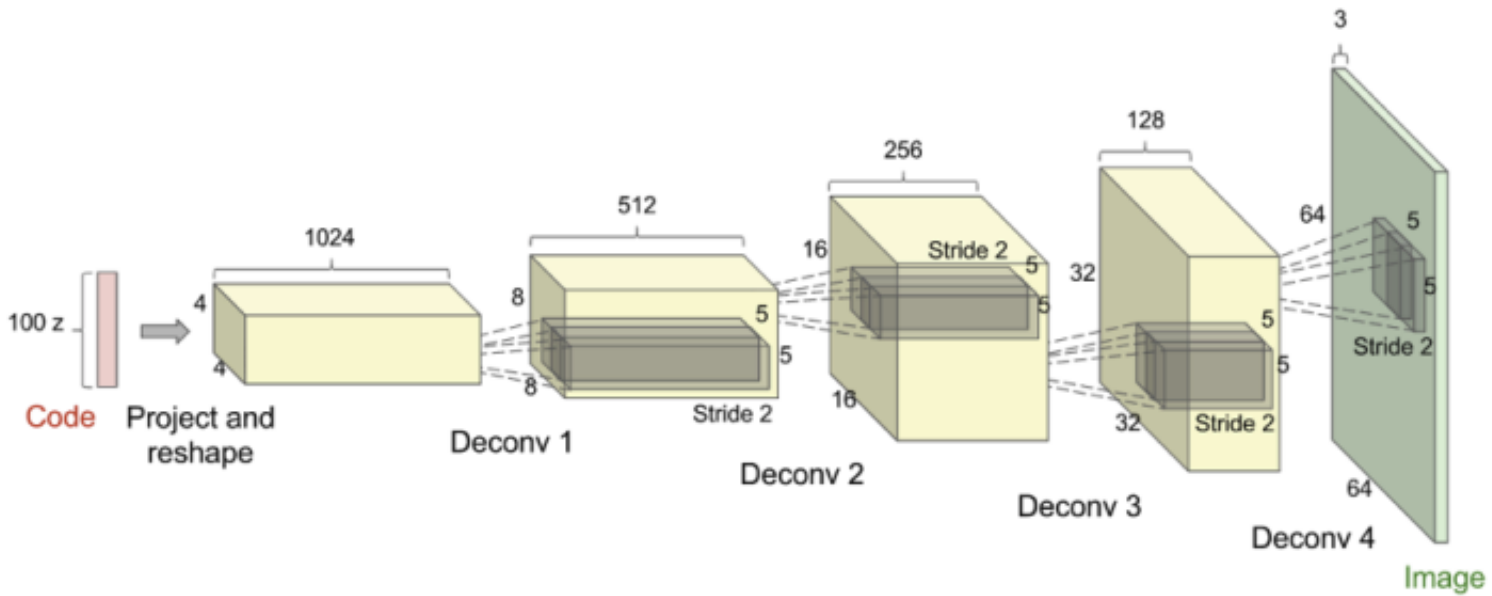
- deep generative model
  - easy to sample
  - high representation power
  - but no tractable evaluation of the density (i.e. p.d.f.)

Deep generative model

- sampling from a deep generative model, parametrized by w
  - first sample a **latent code** $z \in \mathbb{R}^k$ of small dimension $k \ll d$, from a simple distribution like standard Gaussian $N(0, \mathbf{I}_{k \times k})$
  - pass the code through a neural network of your choice, with parameter w
  - the output sample $x \in \mathbb{R}^d$ is the sample of this deep generative model



Deep generative model



Generative model

- a task of importance in unsupervised learning is fitting a generative model
- classically, if we fit a parametric model like mixture of Gaussians, we write the likelihood function explicitly in terms of the model parameters, and maximize it using some algorithms
- $$\text{maximize}_w \sum_{i=1}^{n} \underbrace{\log\left(P_w(x_i)\right)}_{\text{p.d.f.}}$$
- deep generative models use neural networks, but the likelihood of deep generative models cannot be evaluated easily, so we use alternative methods

Goal

- Given examples $\{x_i\}_{i=1}^{n}$ coming i.i.d from an unknown distribution $P(x)$, train a generative model that can

These are computer generated images from the "bigGAN".

Adversarial training

- Classification
  - Consider the example of SPAM detection
  - Each sample x_i is an email
  - Distribution of **true email** is $P(x)$
  - Suppose spammers generate **spams** with distribution $Q(x)$
  - Spam detection: Typical classification task
    - Generate samples from true emails and label them $y_i = 1$
    - Generate samples from spams and label them $y_i = 0$
    - Using these as training data, train a classifier that outputs

$$\mathbb{P}(y_i = 1 \,|\, x_i) \simeq \frac{1}{1 + e^{-f_\theta(x)}}$$

for some neural network $f_\theta(\,\cdot\,)$ with parameter θ
(this is the **logistic model** for binary classification)

Adversarial training

- Applying logistic regression, we want to solve

$$\max_{\theta} \sum_{i:y_i=1} \log\left(\frac{1}{1+e^{-f_\theta(x_i)}}\right) + \sum_{i:y_i=0} \log\left(1-\frac{1}{1+e^{-f_\theta(x_i)}}\right)$$

- in **adversarial training**, it is customary to write

$$D_\theta(x) = \frac{1}{1+e^{-f_\theta(x)}}$$

which is called a **discriminator**

- and find the "best" discriminator by solving for

$$\max_{\theta} \mathscr{L}(\theta) = \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{x_i \sim Q(\cdot)} \log(1 - D_\theta(x_i))$$

as 1 labelled examples come from real distribution $P(\cdot)$

and 0 labelled examples come from spam distribution $Q(\cdot)$

Adversarial training

- Suppose now that the **spam detector (i.e. the discriminator)** is fixed, then the spammer's job is to generate spams that can fool the detector by making the likelihood of the spams being classified as spams **small**:

$$\min_{Q(\cdot)} \mathscr{L}(\theta) = \underbrace{\sum_{x_i \sim P(\cdot)} \log D_\theta(x_i)}_{\text{does not depend on } Q(\cdot)} + \sum_{x_i \sim Q(\cdot)} \log(1 - D_\theta(x_i))$$

- where 0 labelled examples are coming from the distribution $Q(\cdot)$, which is modeled by a **deep neural network generative model,** i.e. $x_i = G_w(z_i)$ where $z_i \sim N(0, \mathbf{I}_{k \times k})$.
- The minimization can be solved by finding. The "best" generative model that can fool the discriminator

$$\min_{w} \mathscr{L}(w, \theta) = \underbrace{\sum_{x_i \sim P(\cdot)} \log D_\theta(x_i)}_{\text{does not depend on } Q(\cdot)} \boxed{+ \sum_{x_i \sim Q(\cdot)} \log\Big(1 - D_\theta\big(G_w(z_i)\big)\Big) \quad z_i \sim N(0, \mathbb{I}_{k \times k})}$$

Adversarial training

- Now we have a game between the spammer and the spam detector:

$$\min_{w} \max_{\theta} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} \log(1 - D_\theta(G_W(z_i)))$$

- Where $P(\cdot)$ is the distribution of real data (true emails), and $Q(\cdot)$ is the distribution of the generated data (spams) that we want to train with a **deep generative model**
- jointly training the discriminator and the generator is called **adversarial training**
- Alternating method is used to find the solution

Alternating gradient descent for adversarial training

- Gradient update for the **discriminator** (for fixed w)

$$\max_{\theta} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{x_i \sim Q(\cdot)} \log(1 - D_\theta(x_i))$$

- First sample n examples from real data (in the training set) and the generator data $x_i \sim G_w(z_i)$

  (for the current iterate of the generator weight w)

- compute the gradient for those $2n$ samples using back-propagation

- Update the discriminator weight θ by subtracting the gradient with a choice of a step size

Alternating gradient descent for adversarial training

- gradient update for the generator (for fixed θ)

$$\min_{w} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} \log(1 - D_\theta(G_w(z_i)))$$

- Consider the gradient update on a single sample

$$\min_{w} \mathscr{L}(w, z_i) \;=\; \log(1 - D_\theta(G_w(z_i)))$$

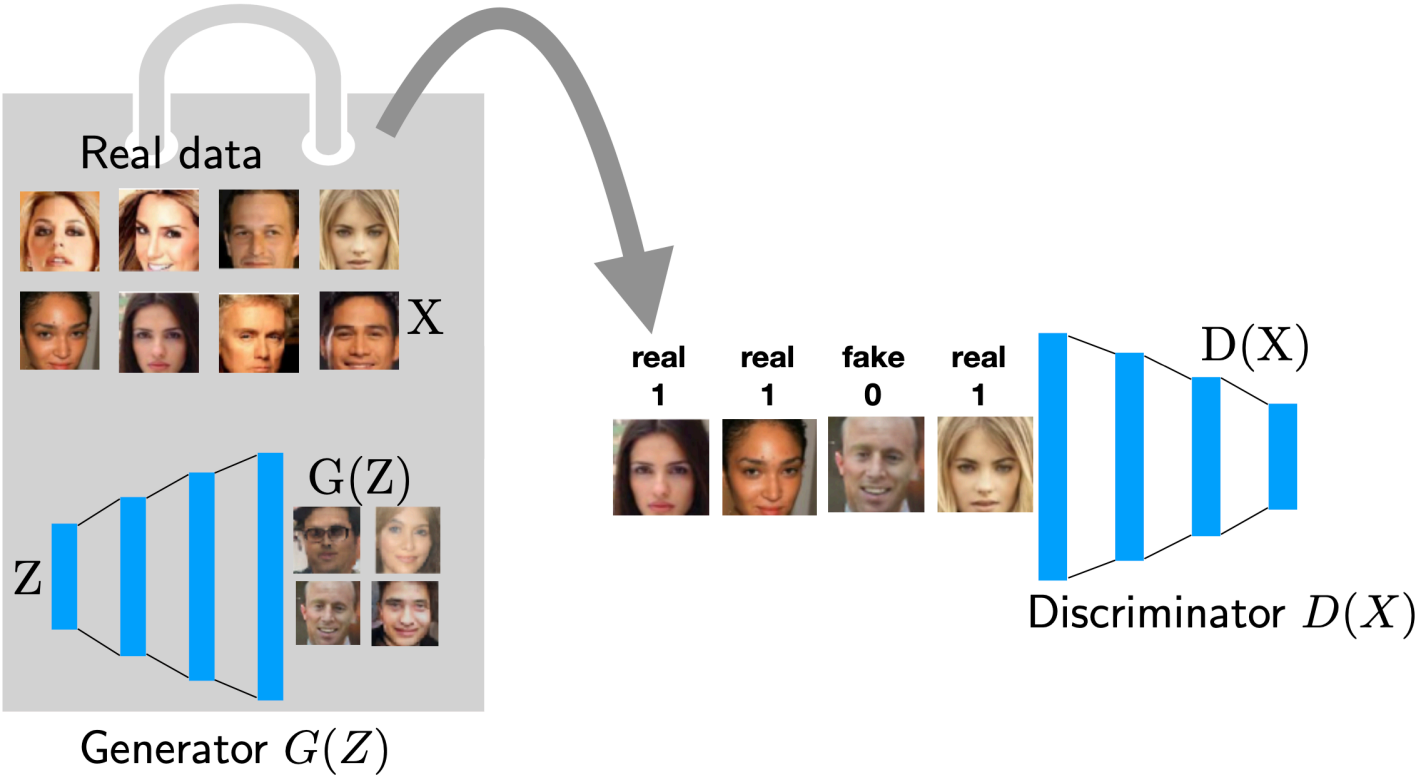
for a single $z_i \sim N(0,\mathbf{I})$ sampled from a Gaussian

- The gradient update is

$$\begin{aligned} w \;&=\; w - \eta \nabla_w \mathscr{L}(w, z_i) \\ &=\; w - \eta \; \nabla_w G_w(z_i) \; \nabla_x D_\theta(x) \; \frac{-1}{1 - D_\theta(x)} \end{aligned}$$

with $x = G_w(z_i)$

This gives a new way to train a deep generative model



$$\min_G \max_D V(G, D)$$

Not only is GAN amazing in generating realistic samples

http://whichfaceisreal.com

It opens new doors to exciting applications

- Cvcle-GAN

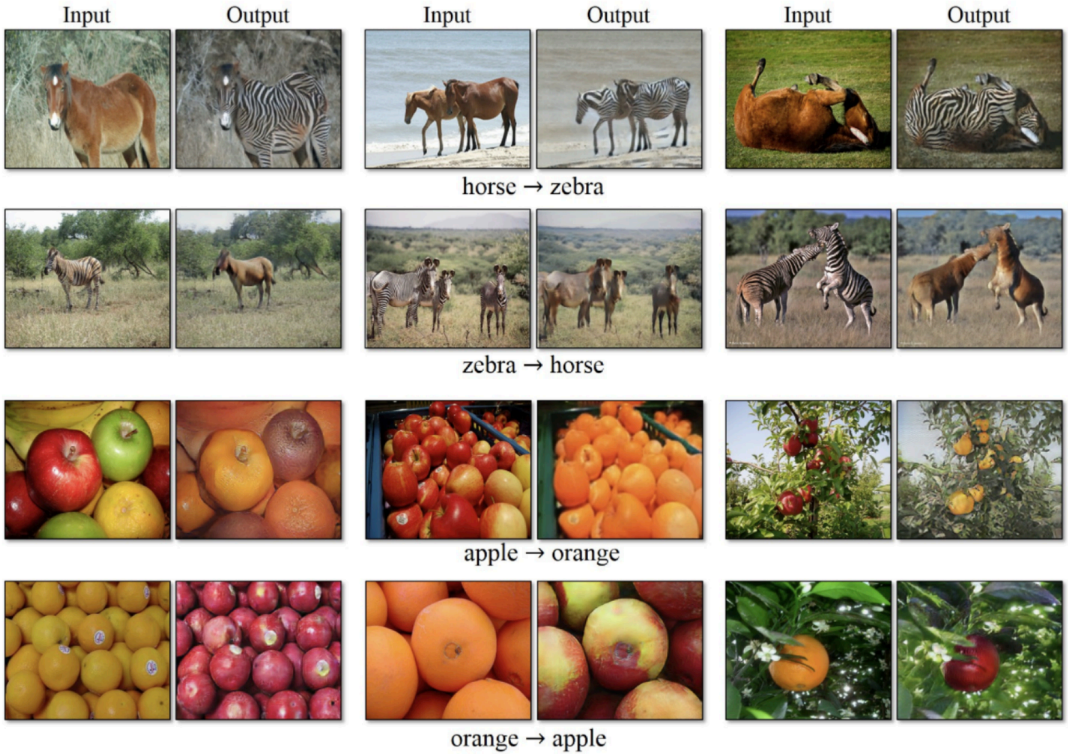


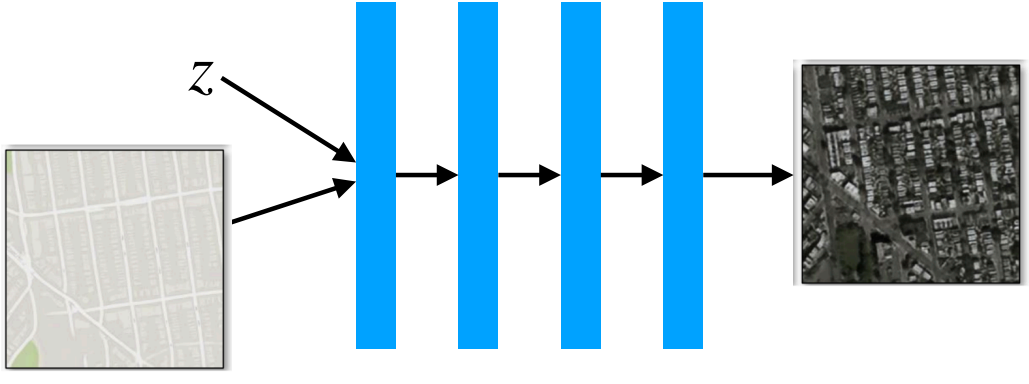
Figure 3: Street scene image translation results. For each pair, left is input and right is the translated image.

Style transfer with generative model

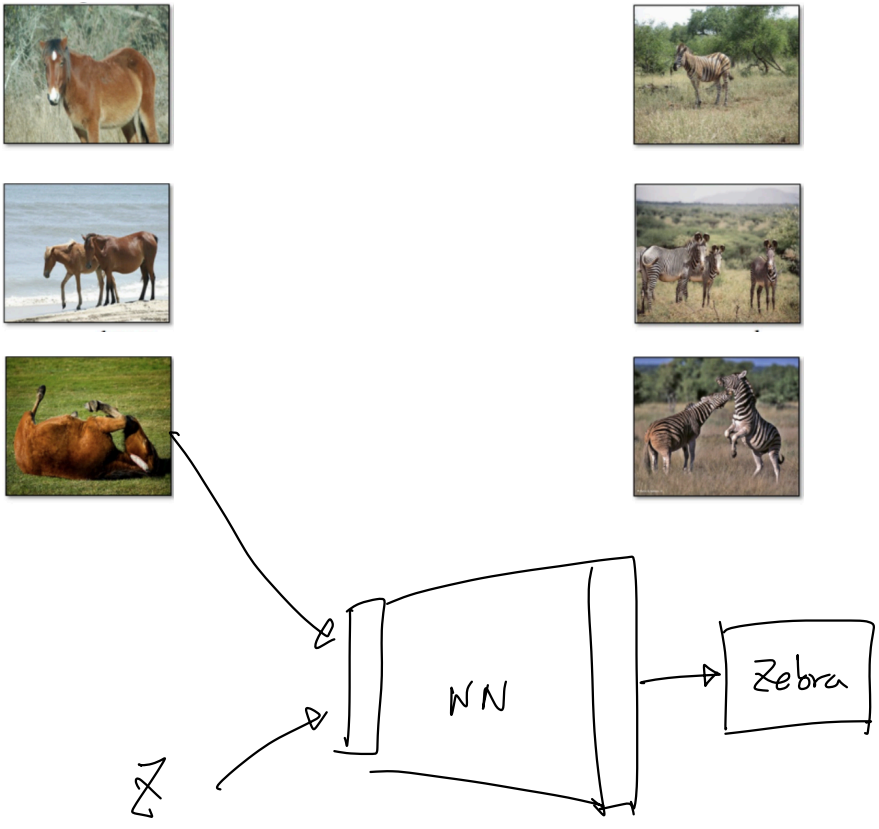
- If we have paired training data,

- And want to train a generative model G(x,z)=y,

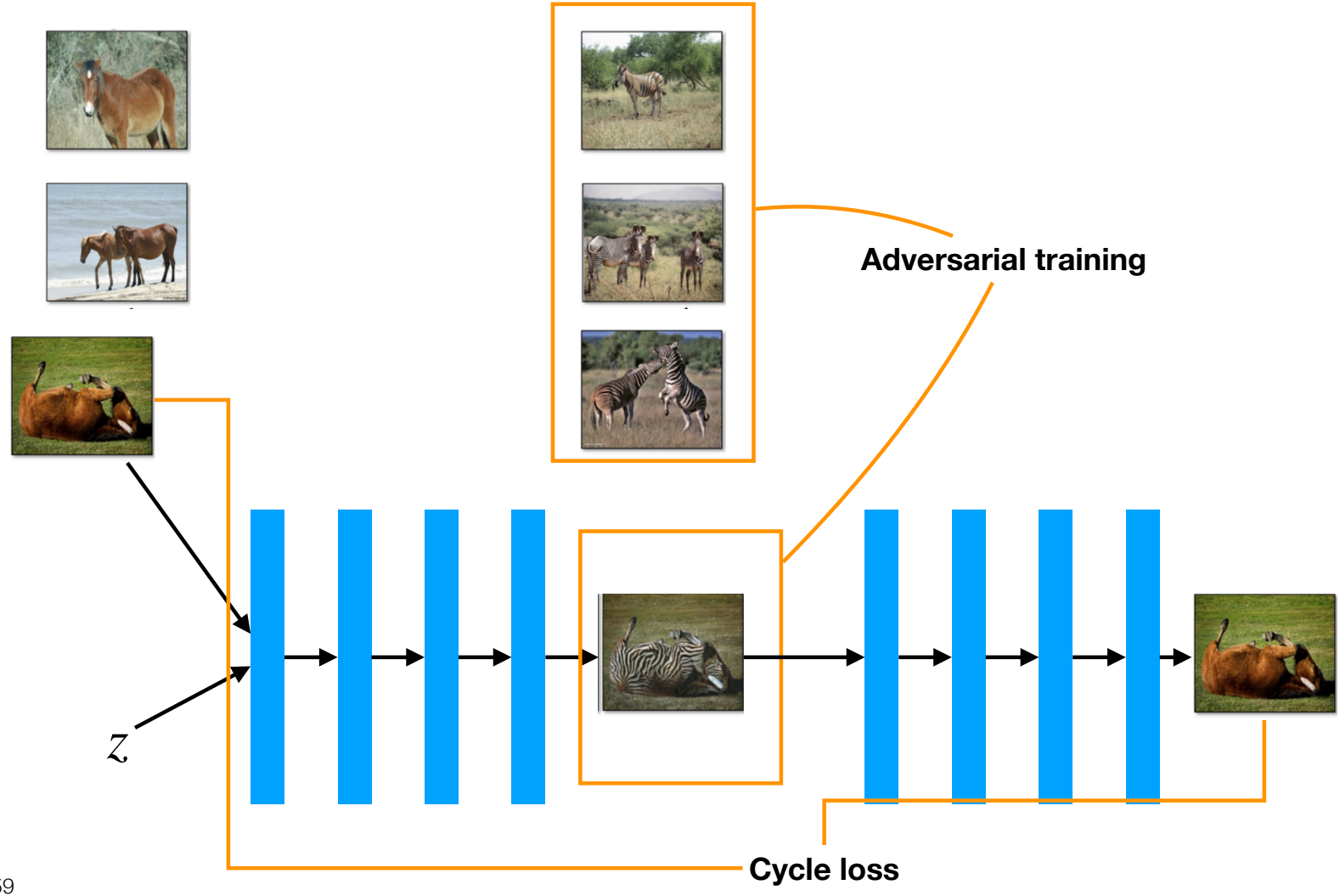
- This can be posed as a regression problem



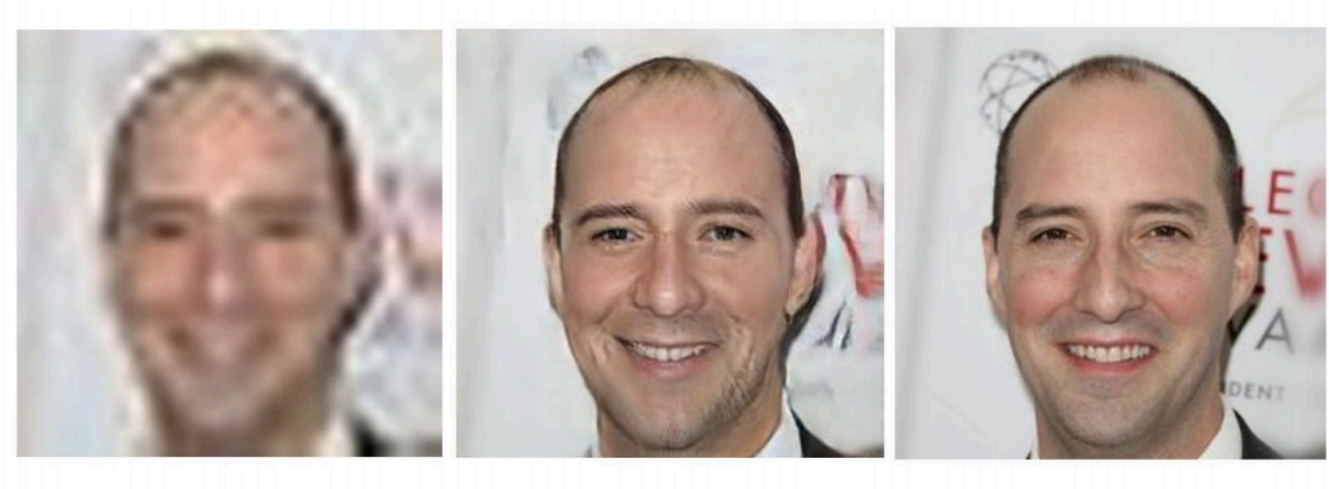
How do we do style transfer without paired data? Cycle-GAN



How do we do style transfer without paired data? Cycle-GAN

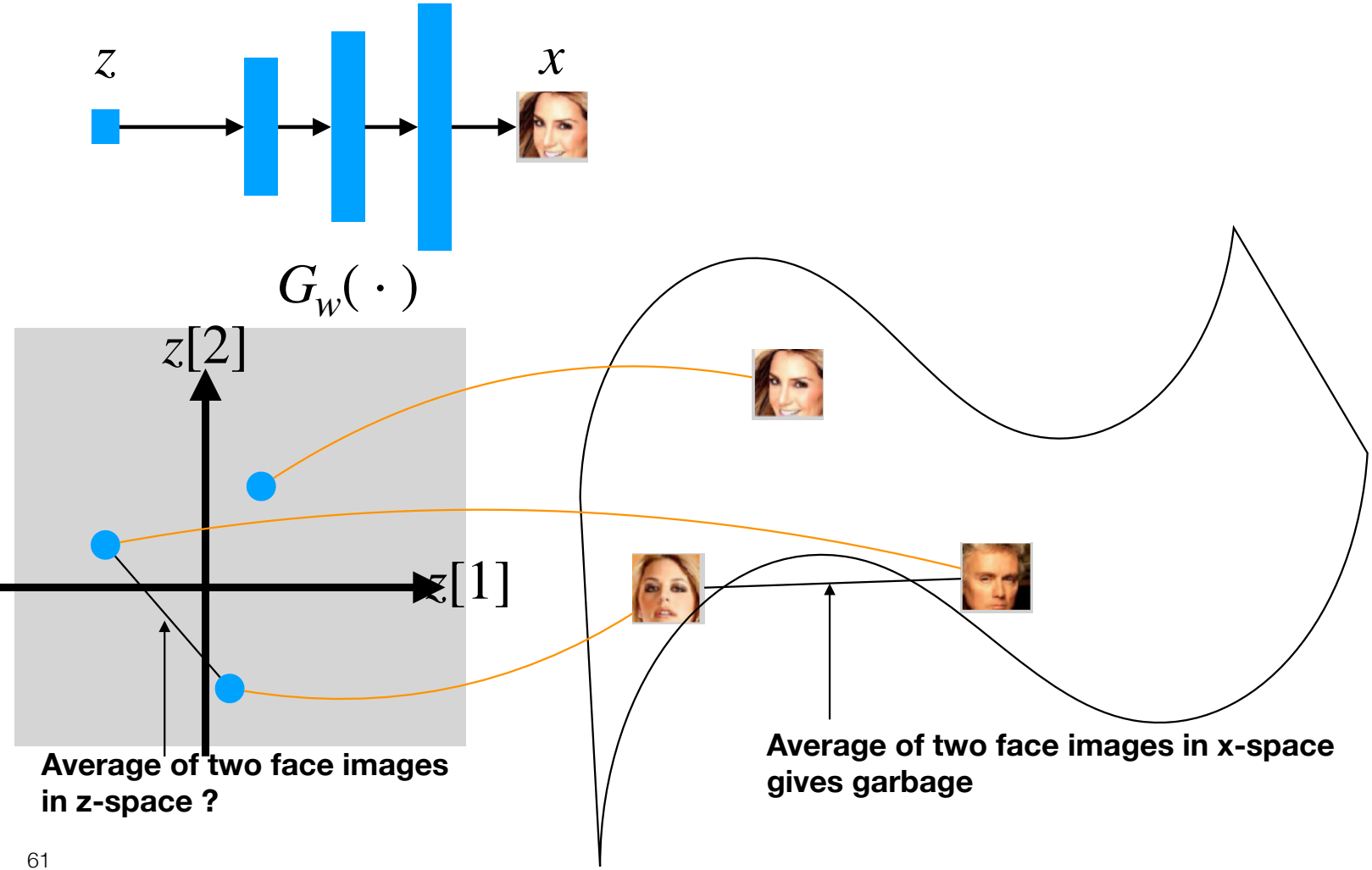


Super resolution



https://www.youtube.com/watch?v=PCBTZh41Ris

The learned latent space is important

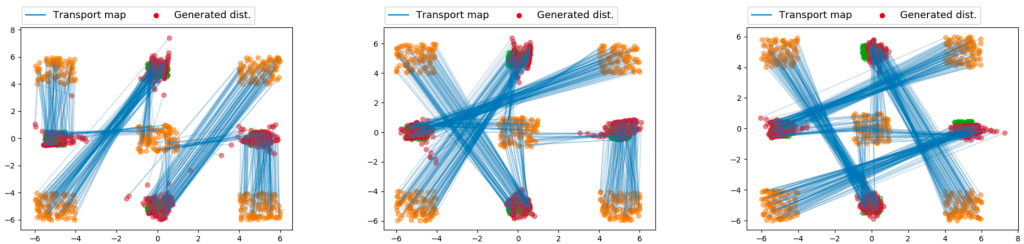


How do we check if we found the right manifold (of faces)?

- **latent traversal**

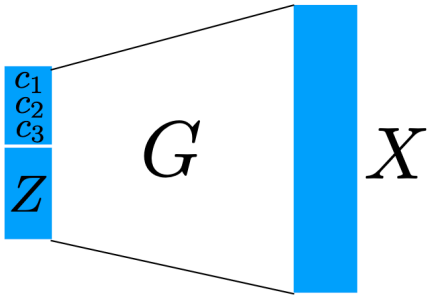
Can we make the relation between the latent space and the image space more meaningful?

- **Disentangling**
  - **GANs learn arbitrary mapping from z to x**
  - **As the loss only depends on the marginal distribution of x and not the conditional distribution of x given z (how z is mapped to x)**

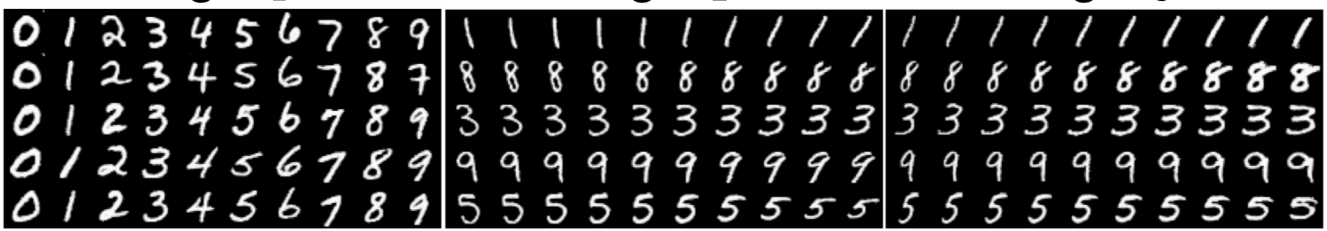


Disentangling seeks meaningful mapping from $\mathcal{Z}$ to $\mathcal{X}$

- there is no formal (mathematical) universally agreed upon definition of disentangling

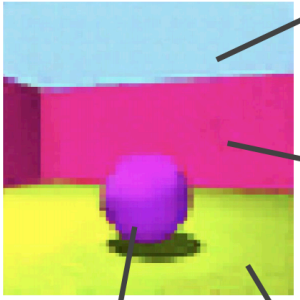


$\Rightarrow$ change c_1 $\Rightarrow$ change c_2 $\Rightarrow$ change c_3



- informally, we seek latent codes that
  - are "informative" or make "noticeable" changes
  - are "uncorrelated" or make "distinct" changes

Decompose data into a set of underlying **human-interpretable** factors of variation



Explainable models

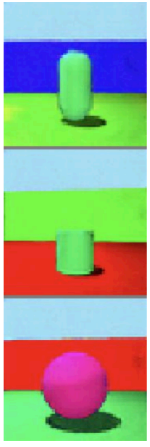
What is in the scene?

Controllable generation

Generate a red ball instead

Fully-supervised case

Strategy: Label everything



c_1 c_2 c_3

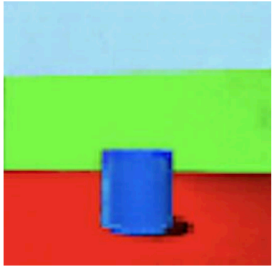
{dark blue wall, green floor, green oval}

{green wall, red floor, green cylinder}

{red wall, green floor, pink ball}

Controllable generation as **label-conditional generative modeling**

green wall, red floor, blue cylinder



$Q(x) \sim P(x)$

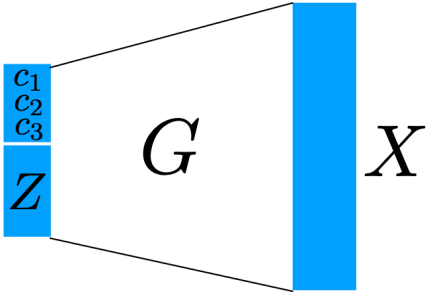
Fake Real

Train a **conditional GAN**, where

(c_1, c_2, c_3) is a numerical representation of the **labels**

given in the training data, and z is drawn from Gaussian

$Q(x|c) \sim P(x|c)$



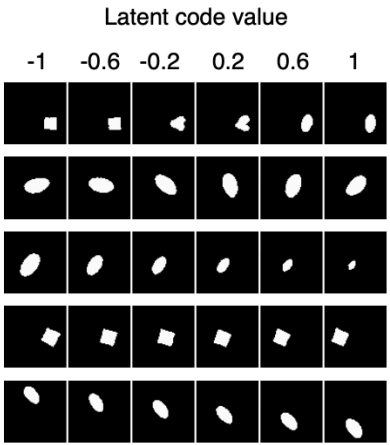
However, some properties are hard to represent numerically

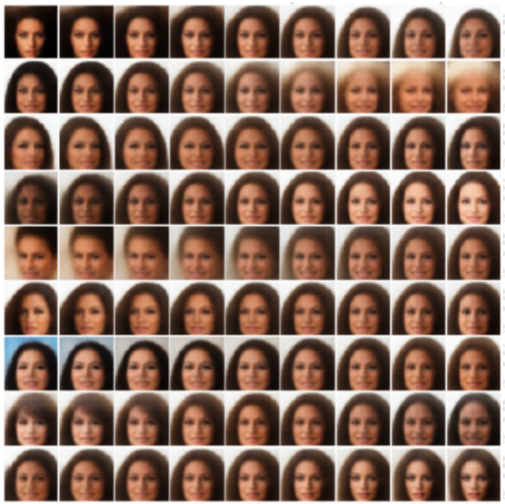
What kind of hairstyle?

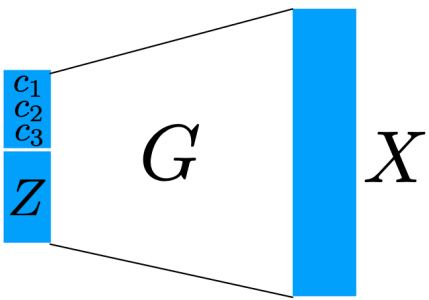
What kind of glasses?

Generate this guy with this hair

Unsupervised training of Disentangled GAN



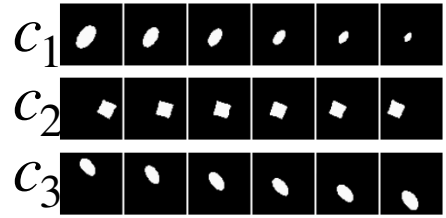




Disentangled GAN training: InfoGAN-CR, 2019

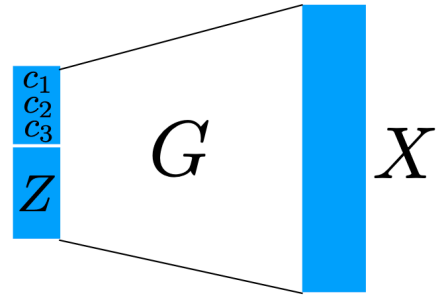
- 1. As in standard GAN training, we want $G_w(z)$ to look like training data (which is achieved by adversarial loss provided by a discriminator)

  D() = {real,fake} $\Rightarrow Q(x) \sim P(x)$



- 2. We also want the controllable latent code c to be predictable from the image
  - add a NN regressor that predicts $\hat{c}(x)$, and train the generator that makes the prediction accuracy high (note that both this predictor and the generator works to make the prediction accurate, unlike adversarial loss).

    NN predictor



$$\text{minimize} \quad \| \hat{c}(\quad) - c \|^2$$

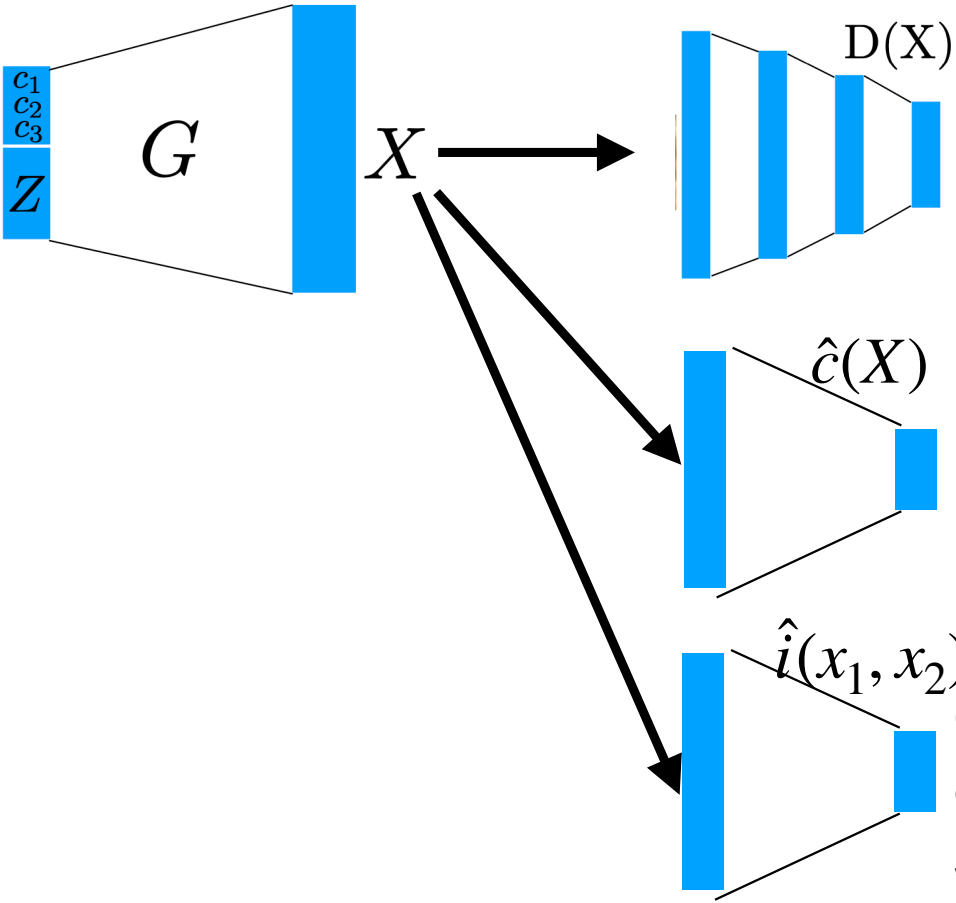
- 3. We also want each code to control distinct properties
  - add a NN that predicts which code was changed

    NN predictor

$$\hat{i}(\quad) \simeq i$$

Disentangling with contrastive regularizer

- To train a disentangled GAN, we use contrastive regularizer

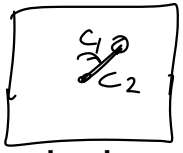


Discriminator encourages output x to be realistic

Predictor makes sure that the changes in c make noticeable changes in x

Contrastive regularizer detects which latent code c_i was the same in a paired $(x_1, x_2$

But is still challenging

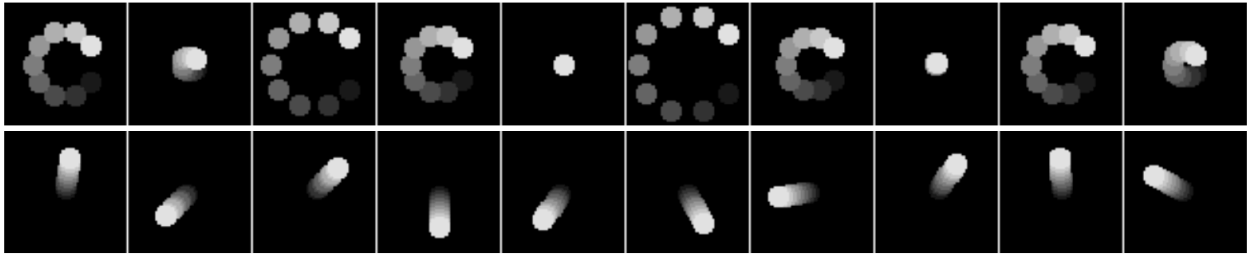


- Synthetic training data (with planted disentangled representation)

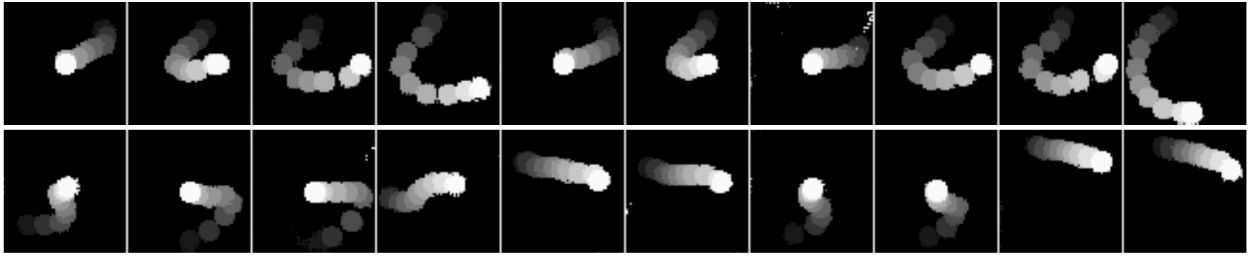
Synthetic data with two attributes (angle,radius)

change c_1 fix c_2

change c_2

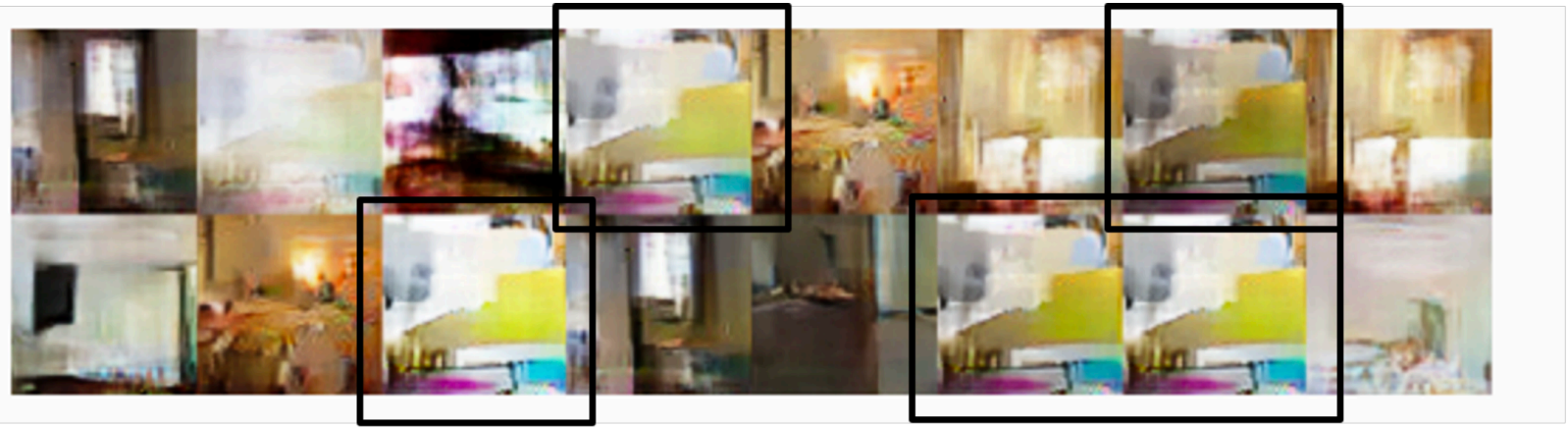


- Trained Disentangled GAN (latent traversal)



Challenges in training GANs

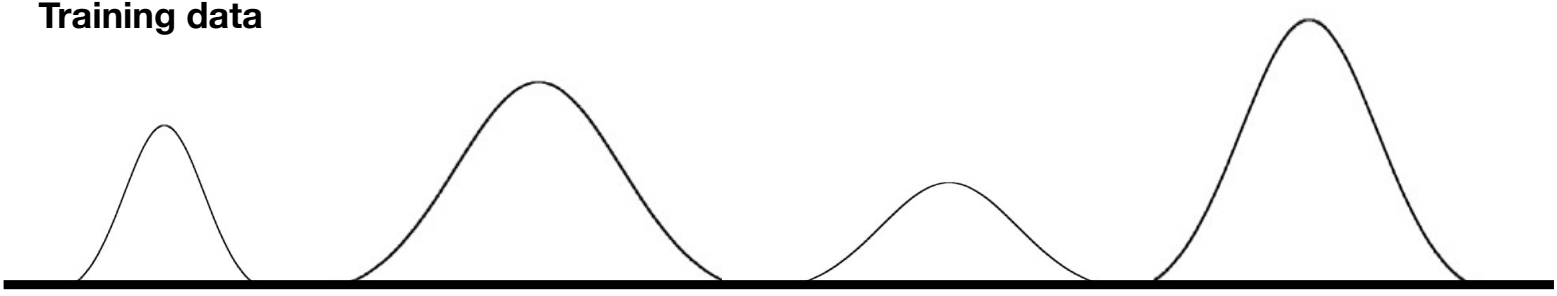
- GAN training suffers from **mode collapse**
- this refers to the phenomenon where the generated samples are not as diverse as the training samples



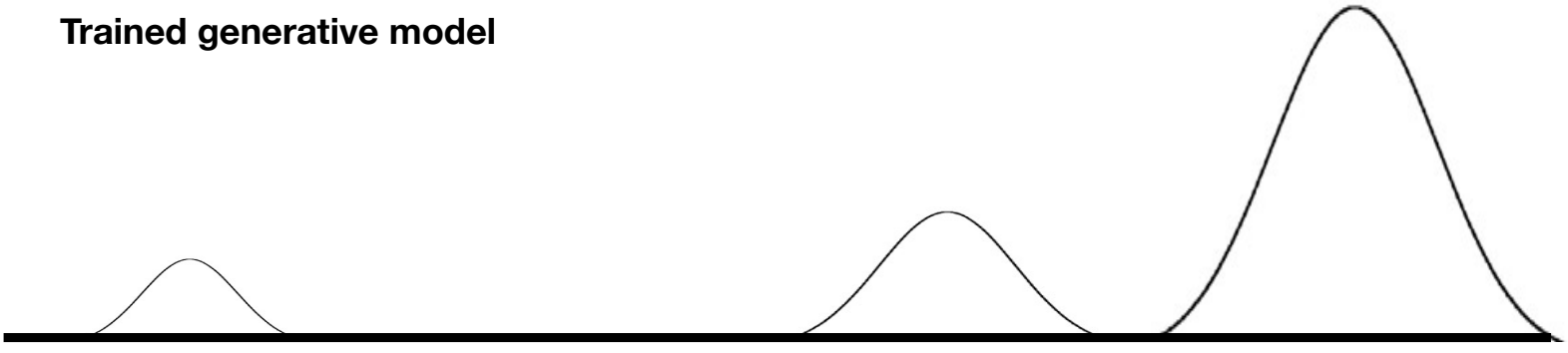
Arjovsky et al., 2017

Mode collapse

Training data

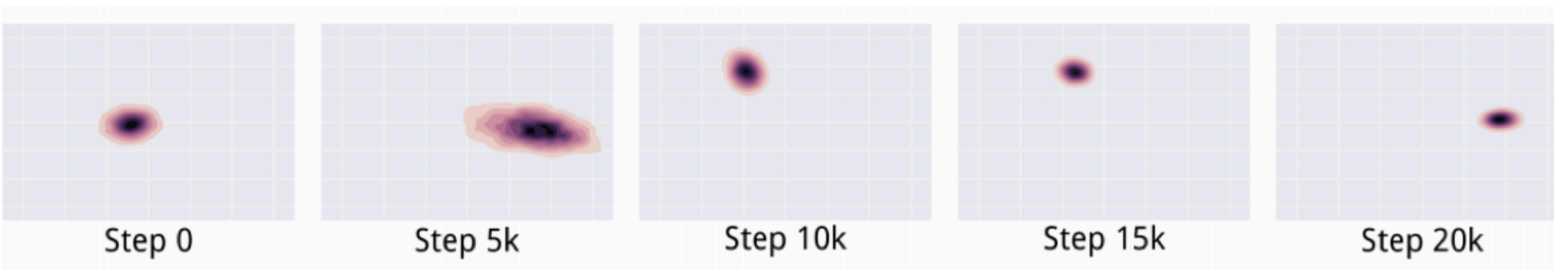


Trained generative model



Mode collapse

- True distribution is a mixture of Gaussians



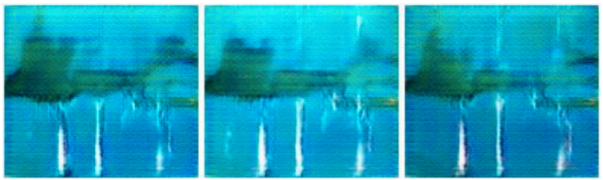
Source: Metz et al., 2017

- The generator distribution keeps oscillating between different modes

Mode collapse

- "A man in a orange jacket with sunglasses and a hat ski down a hill."

- "This guy is in black trunks and swimming underwater."



- "A tennis player in a blue polo shirt is looking down at the green court."

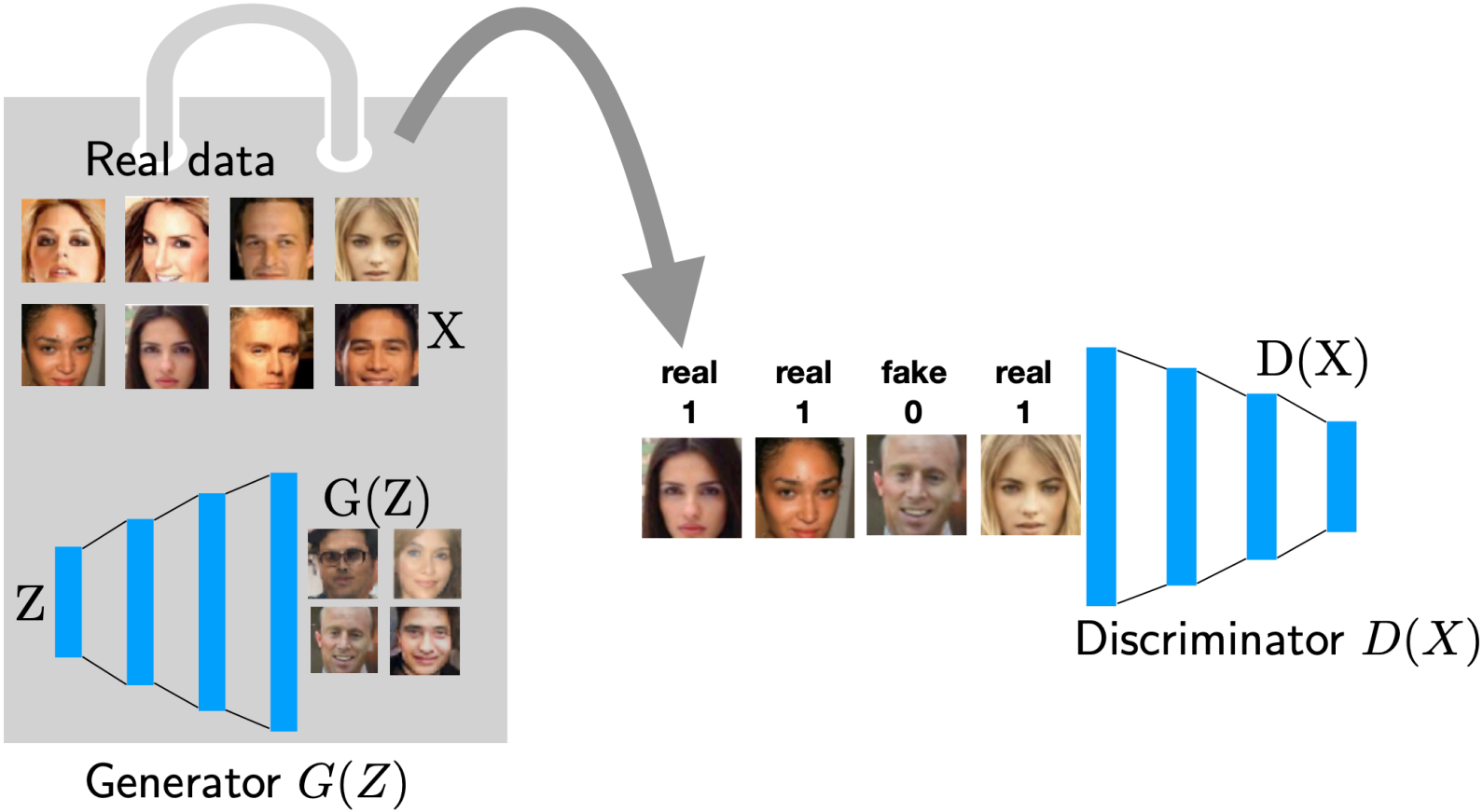
["Generating interpretable images with controllable structure", by Reed et al., 2016]

Principled approach to mode collapse

- Lack of diversity is easier to detect if we see multiple samples
- Consider MNIST hand-written digits
  - If we have a generator that generates 1,3,5,7 perfectly, it is hard to tell from a single sample that mode collapse has happened
  - But easier to tell from a collection of, say, 5 samples all from wither training data or all from generated data

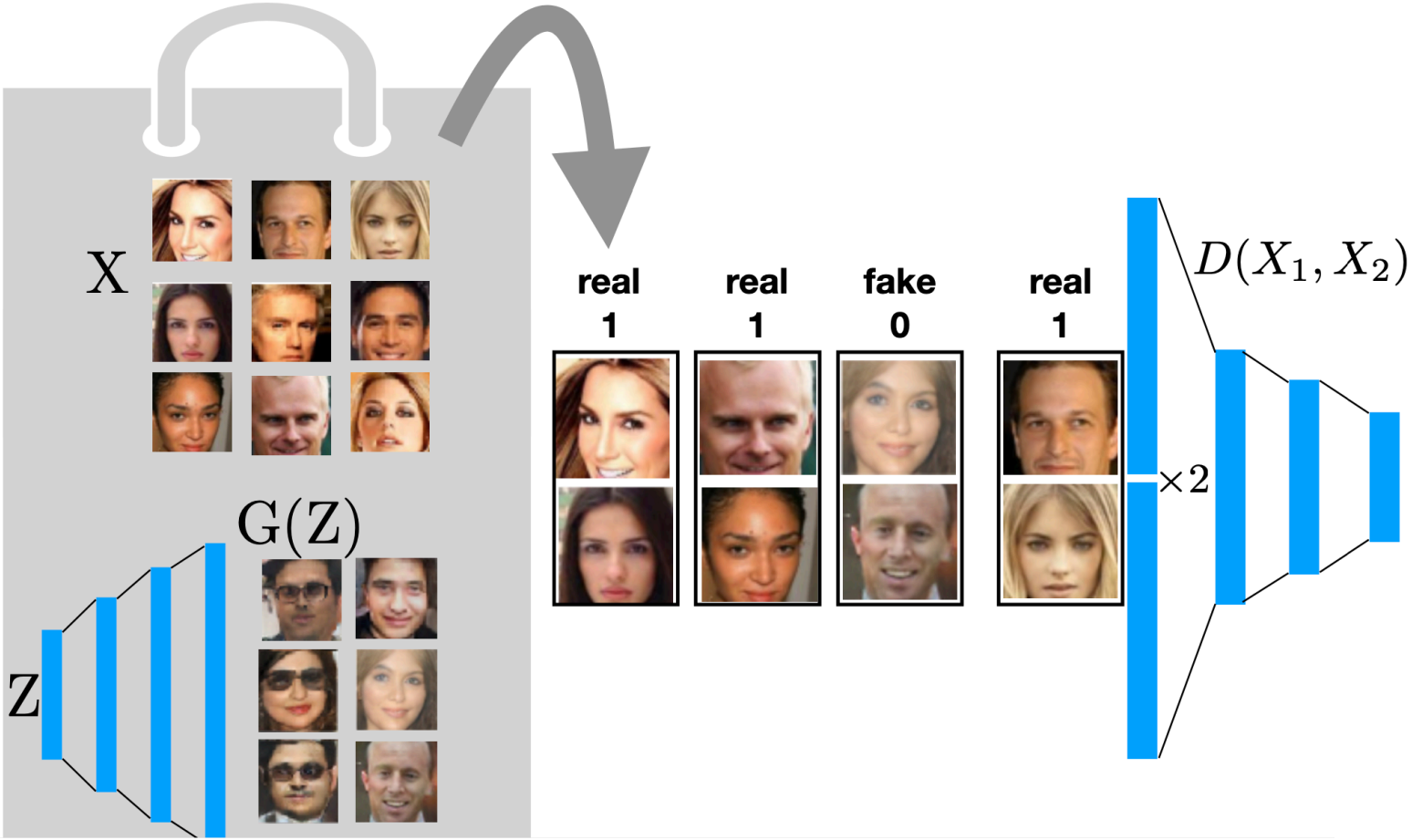
Principled approach to mode collapse

- Turning this intuition into a training algorithm:

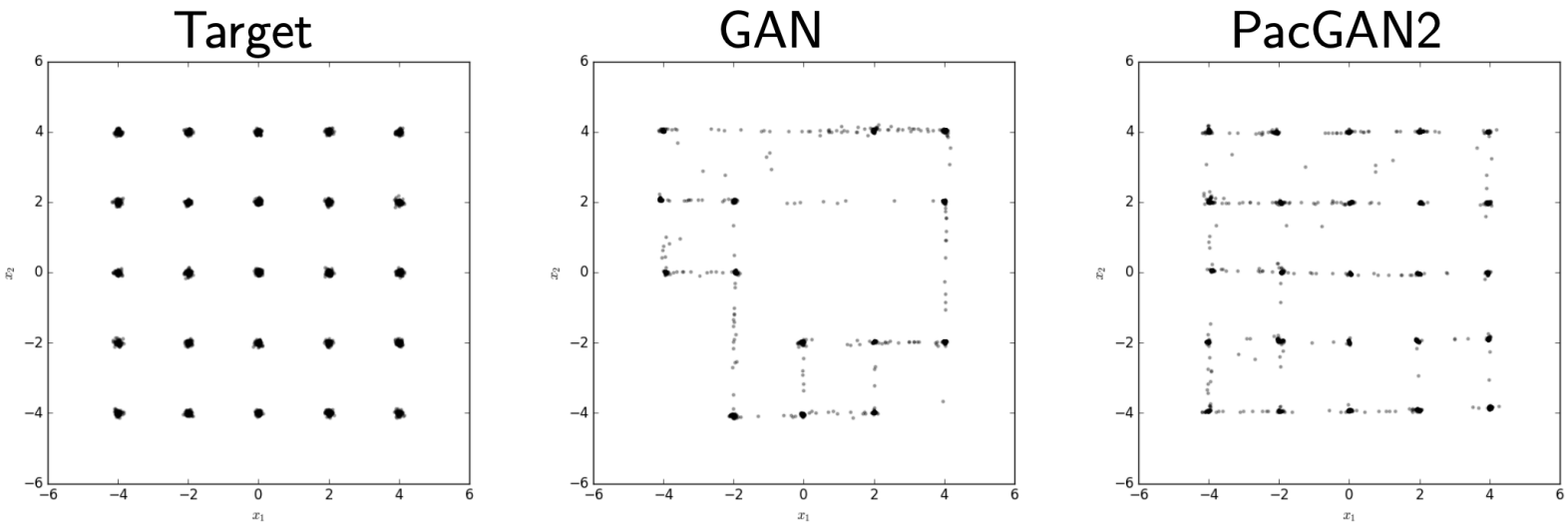


Principled approach to mode collapse: PacGAN, 2018

- Turning this intuition into a training algorithm:



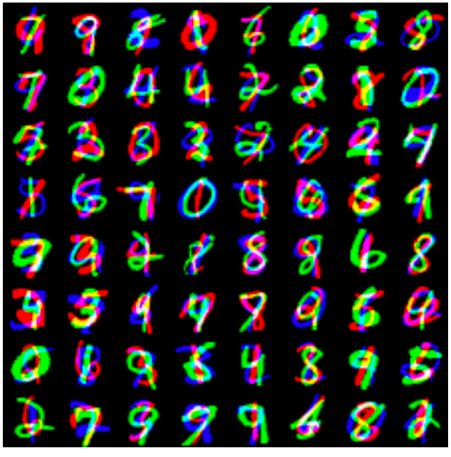
Principled approach to mode collapse



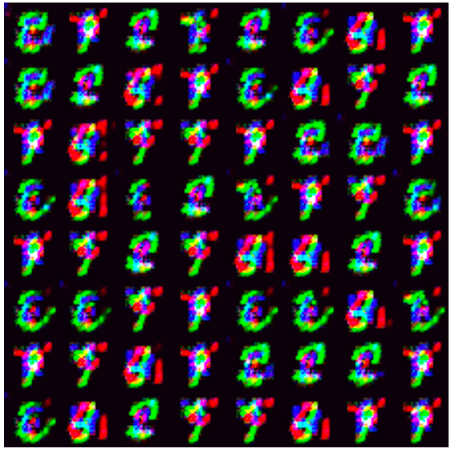
	Modes (Max 25)
GAN	17.3
PacGAN2	23.8
PacGAN3	24.6
PacGAN4	24.8

Principled approach to mode collapse

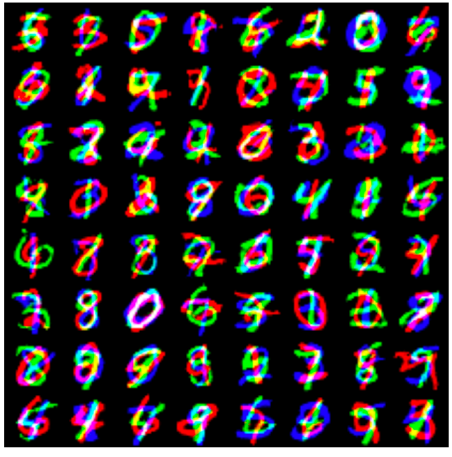
Real data



DCGAN



PacDCGAN2

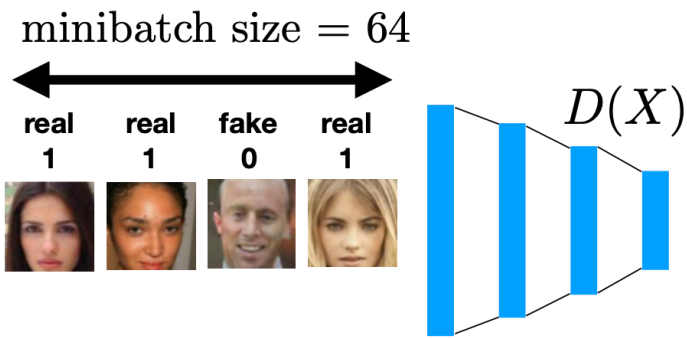


	Modes (Max 1000)
DCGAN	99.0
ALI	16.0
Unrolled GAN	48.7
VEEGAN	150.0
PacDCGAN2	1000.0
PacDCGAN3	1000.0
PacDCGAN4	1000.0

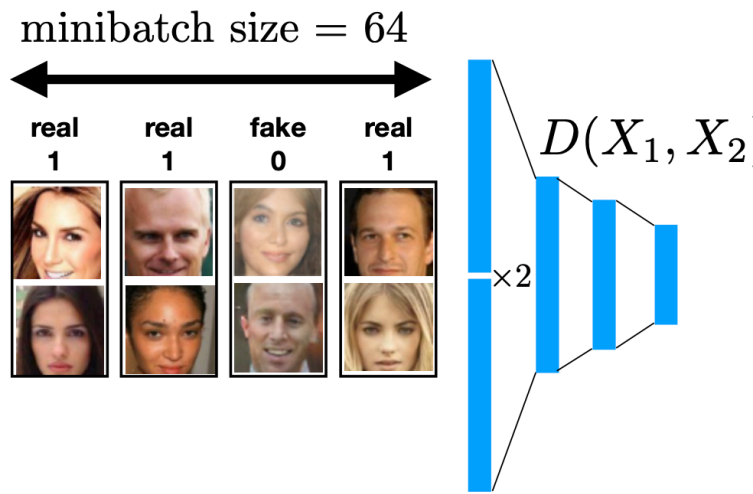
Principled approach to mode collapse

- Could PacGAN be cheating, as it is a larger discriminator network?

1. Discriminator size



GAN

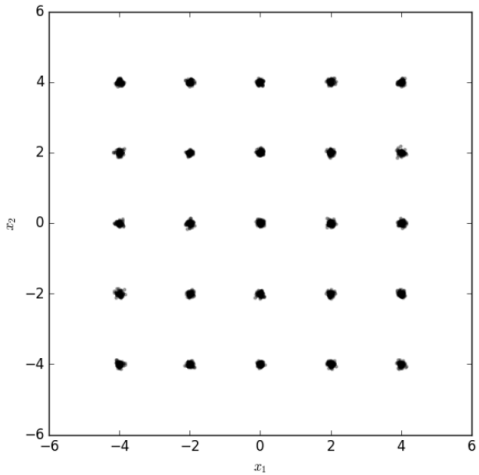


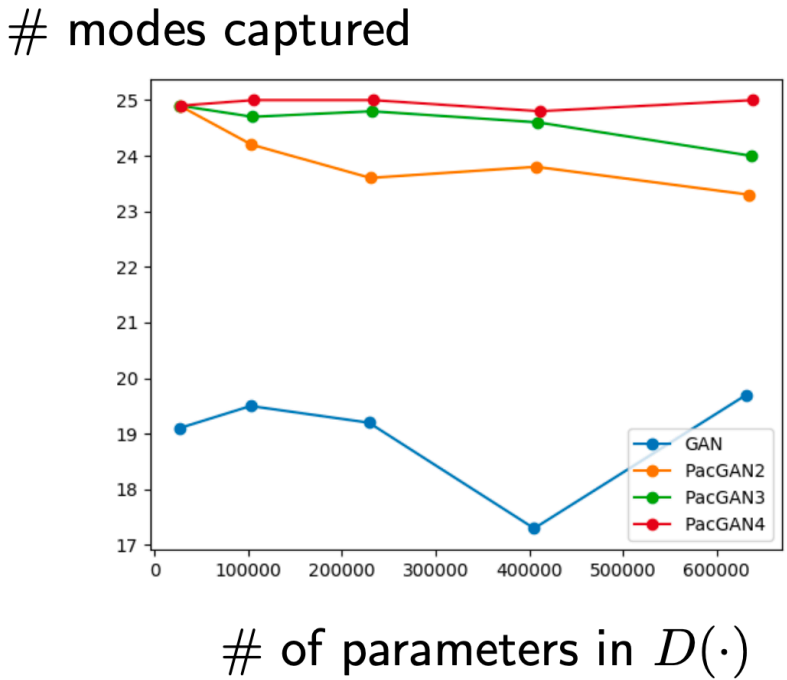
PacGAN2

Principled approach to mode collapse

- Could PacGAN be cheating, as it is a larger discriminator network?

1. Discriminator size

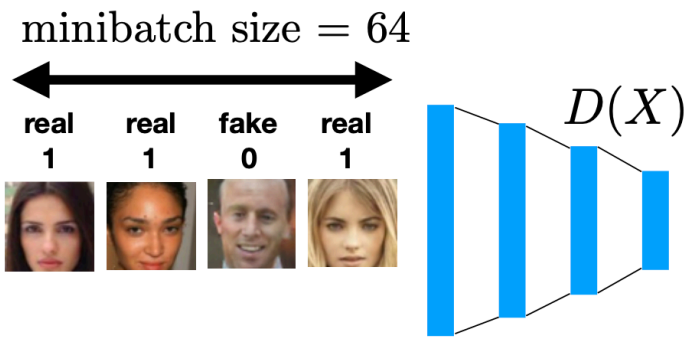




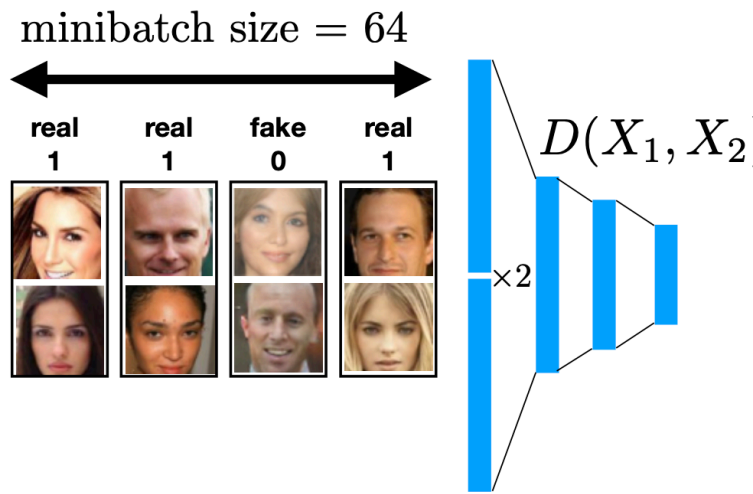
Principled approach to mode collapse

- Could PacGAN be cheating, as it uses more samples at each mini-batch?

1. Discriminator size



GAN

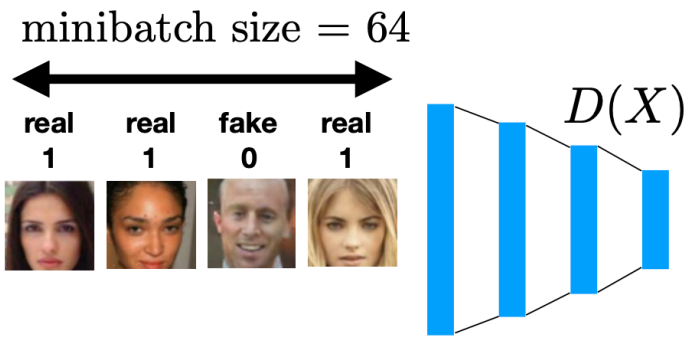


PacGAN2

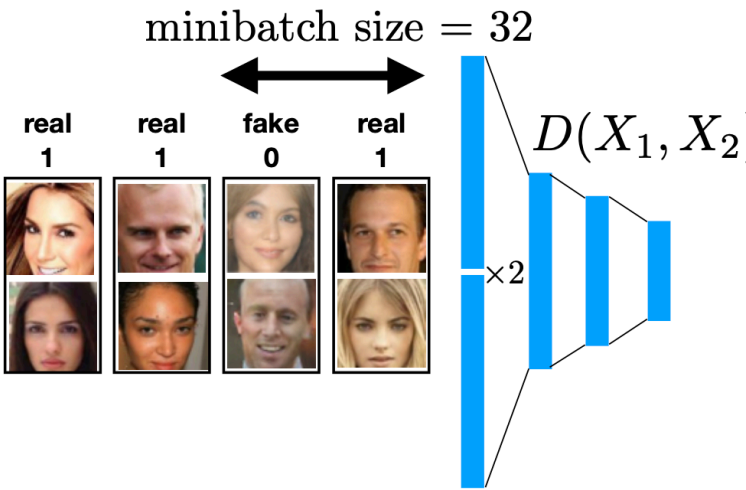
Principled approach to mode collapse

- Could PacGAN be cheating, as it uses more samples at each mini-batch?

2. Minibatch size



GAN

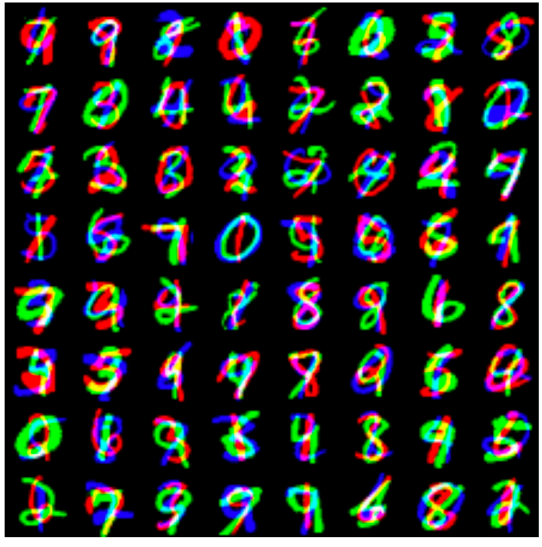


PacGAN2

Principled approach to mode collapse

- Could PacGAN be cheating, as it uses more samples at each mini-batch?

2. Minibatch size



	Modes
DCGAN	99.0
PacDCGAN2	1000.0

Theoretical intuition behind PacGAN

- Typical Gan training loss is

$$\min_{w} \max_{\theta} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} \log(1 - D_\theta(G_W(z_i)))$$

- We will consider

$$\min_{w} \max_{\theta} \sum_{x_i \sim P(\cdot)} D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} (1 - D_\theta(G_W(z_i)))$$

$$\text{subject to} \quad |D_\theta(x)| \leq 1 \,, \qquad \text{for all } x$$

Theoretical intuition behind PacGAN

- We will consider

$$\min_{w} \max_{\theta} \sum_{x_i \sim P(\cdot)} D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} (1 - D_\theta(G_W(z_i)))$$

$$\text{subject to} \quad |D_\theta(x)| \leq 1 \,, \qquad \text{for all } x$$

- this is a finite sample approximation of the following expectation

$$\min_{w} \max_{\theta} \; \mathbb{E}_{x \sim P(\cdot)}\big[\, D_\theta(x) \,\big] + \mathbb{E}_{z \sim N(0,\mathbf{I})}\big[\, 1 - D_\theta(G_W(z)) \,\big]$$

- let $Q(\,\cdot\,)$ denote the distribution of the generator $G_w(z_i)$

$$\min_{Q(\cdot)} \max_{\theta} \; \mathbb{E}_{x \sim P(\cdot)}\big[\, D_\theta(x) \,\big] + \mathbb{E}_{x \sim Q(\cdot)}\big[\, 1 - D_\theta(x) \,\big]$$

$$\text{subject to} \quad |D_\theta(x)| \leq 1 \,, \qquad \text{for all } x$$

- at this point, we can solve the maximization w.r.t. D_θ assuming it can represent any functions (for the purpose of theoretical analysis)
  - the optimal solution is

$$D_\theta(x) \;=\; \begin{cases} +1 & \text{if } P(x) \geq Q(x) \\ -1 & \text{if } P(x) < Q(x) \end{cases}$$

Theoretical intuition behind PacGAN

$$\min_{Q(\cdot)} \max_{\theta} \quad \mathbb{E}_{x\sim P(\cdot)}\big[\, D_\theta(x) \,\big] + \mathbb{E}_{x\sim Q(\cdot)}\big[\, 1 - D_\theta(x) \,\big]$$

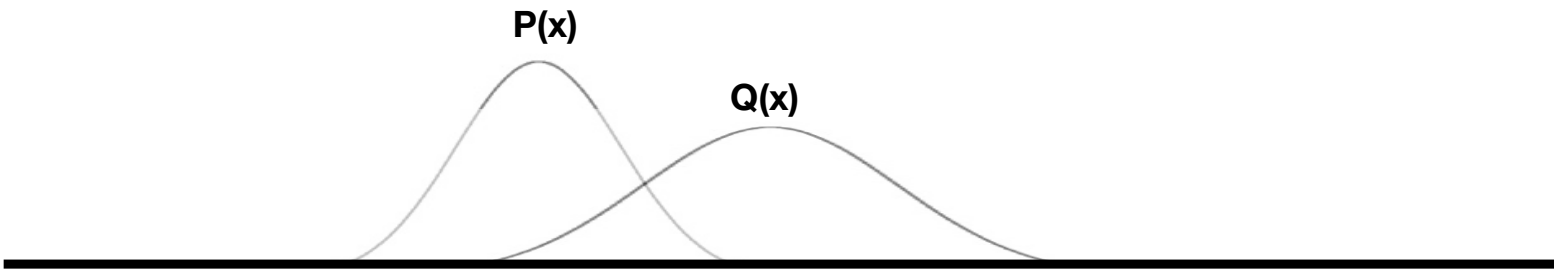
$$\text{subject to} \quad |\, D_\theta(x) \,| \leq 1 \,, \qquad \text{for all } x$$

- at this point, we can solve the maximization w.r.t. D_θ assuming it can represent any functions (for the purpose of theoretical analysis)
  - the optimal solution is

$$D_\theta(x) \;=\; \begin{cases} +1 & \text{if } P(x) \geq Q(x) \\ -1 & \text{if } P(x) < Q(x) \end{cases}$$

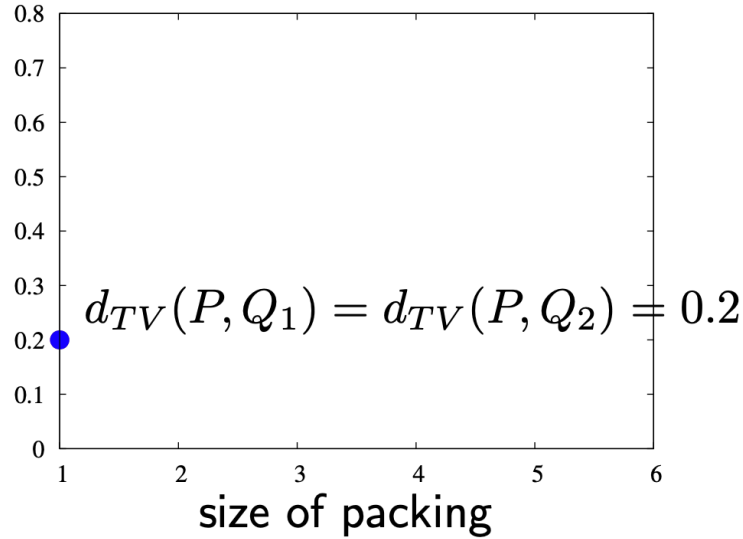
- Plugging this back in to the loss, we get

$$\min_{Q(\cdot)} \; D_{\mathrm{TV}}(P, Q) \;=\; \mathbb{E}_{x\sim P(\cdot)}\left[\, \left| 1 - \frac{Q(x)}{P(x)} \right| \,\right]$$



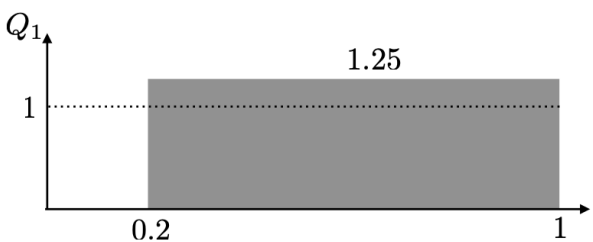
Theoretical intuition behind PacGAN

Target distribution P

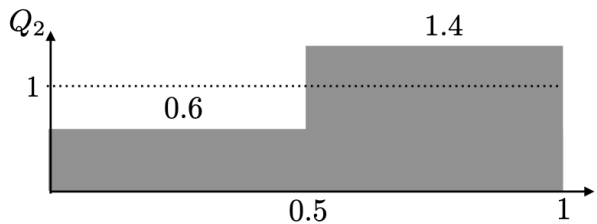


Generator Q_1
with mode collapse

Generator Q_2
without mode collapse

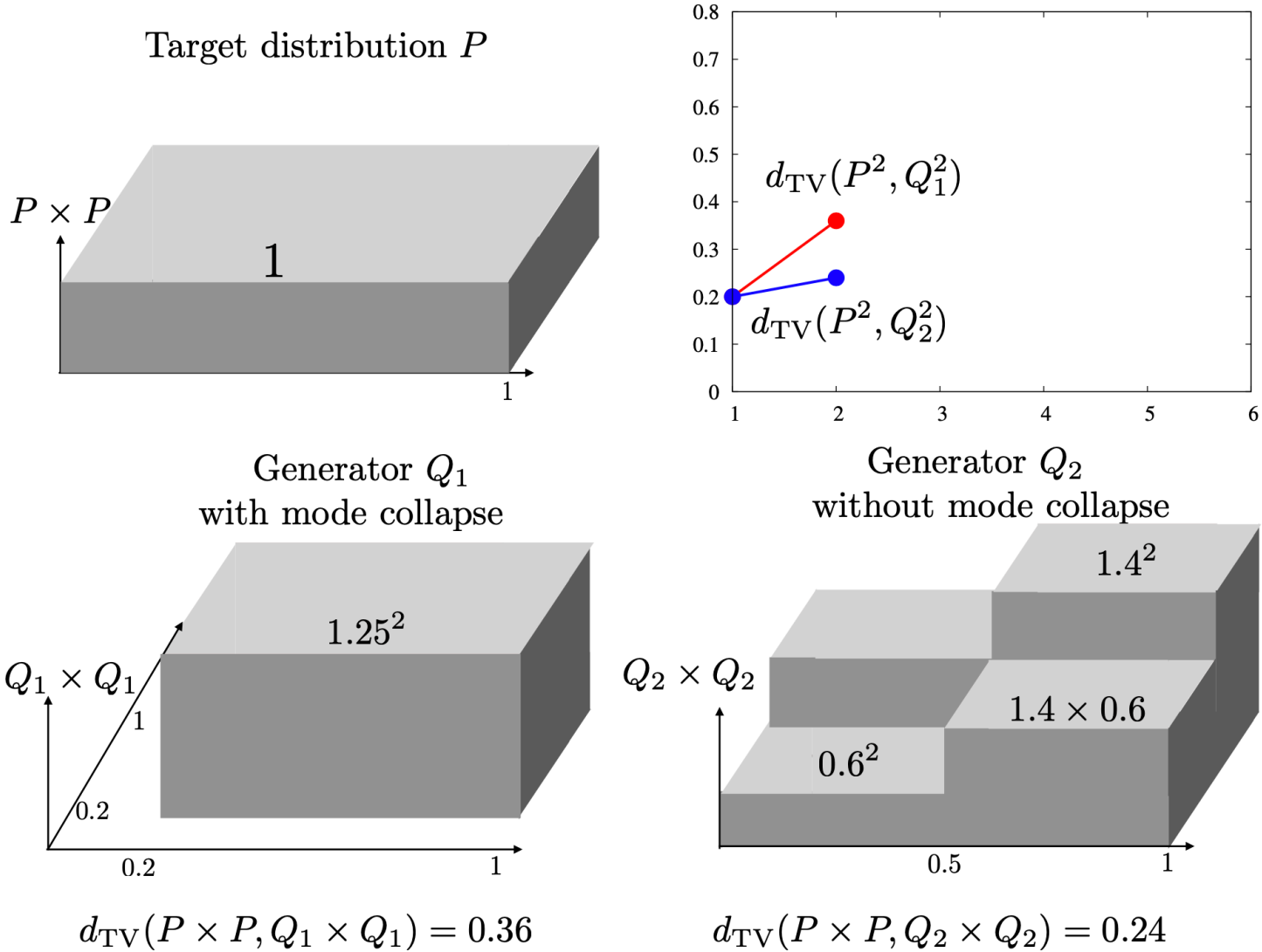


$d_{\mathrm{TV}}(P, Q_1) = 0.2$

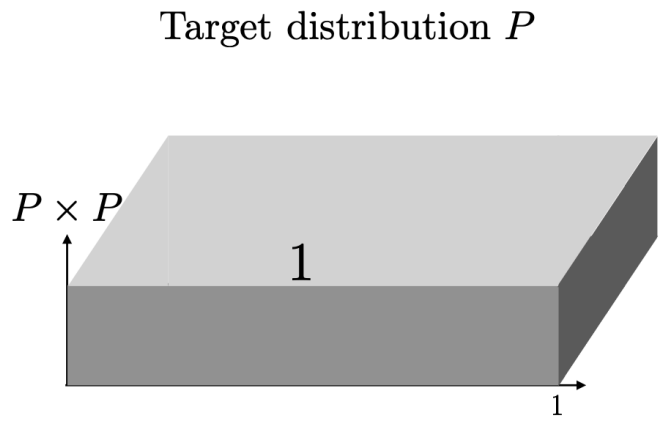


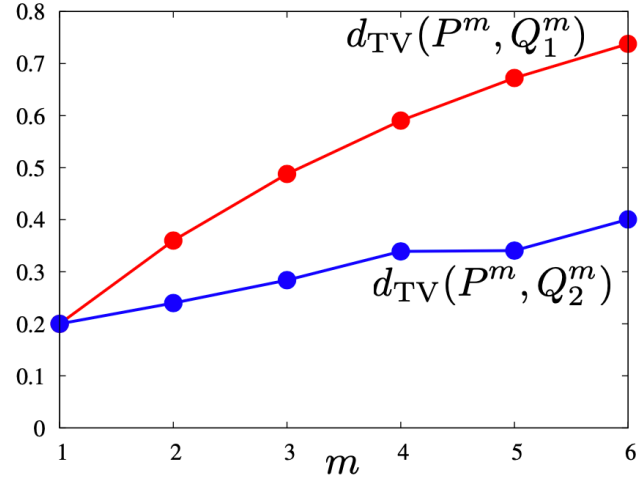
$d_{\mathrm{TV}}(P, Q_2) = 0.2$

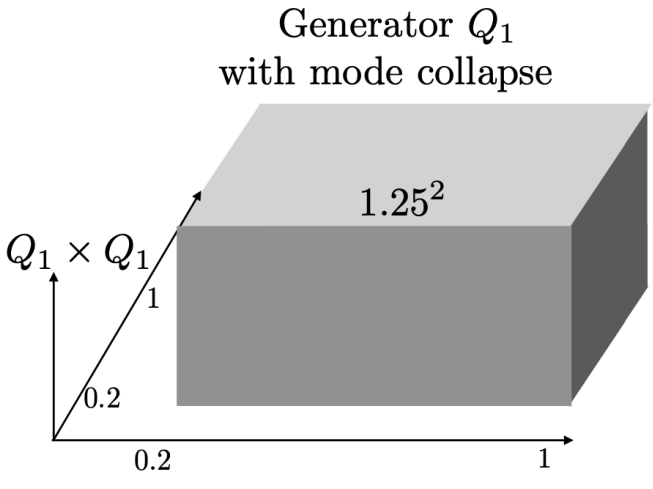
Theoretical intuition behind PacGAN

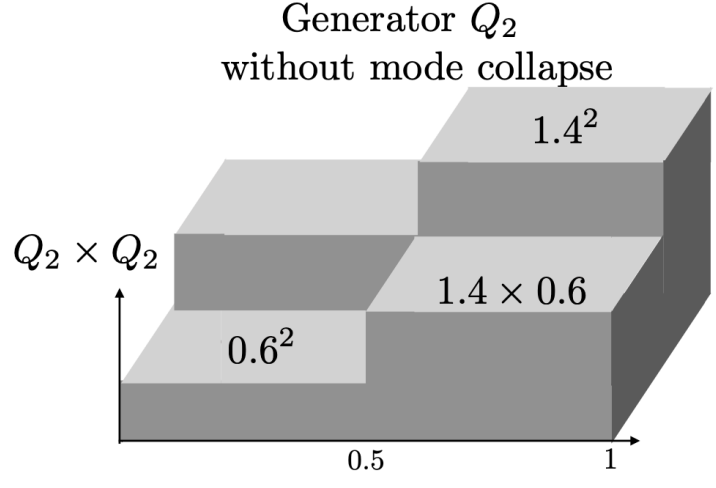


Theoretical intuition behind PacGAN









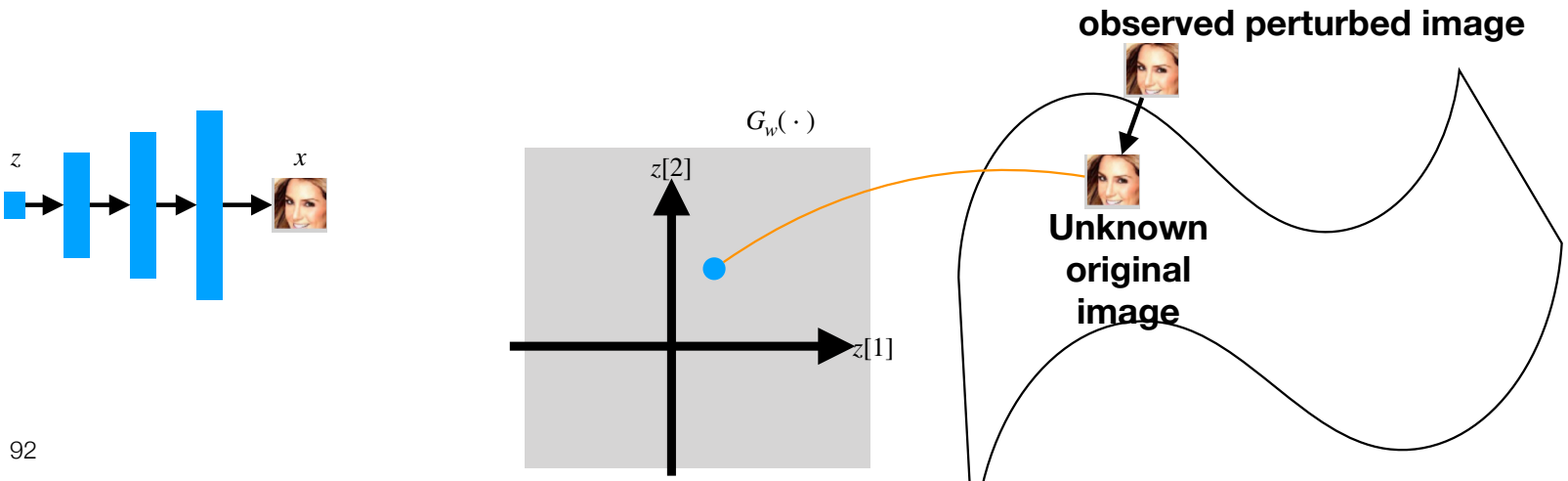
$d_{\mathrm{TV}}(P \times P, Q_1 \times Q_1) = 0.36$

$d_{\mathrm{TV}}(P \times P, Q_2 \times Q_2) = 0.24$

Deep Image prior

- in standard de-noising/inpainting with **trained** GAN
  we want to recover original image from some distortion

- if we have a GAN trained on similar class of images, then we can use the latent space and the manifold of natural images to recover the image as follows



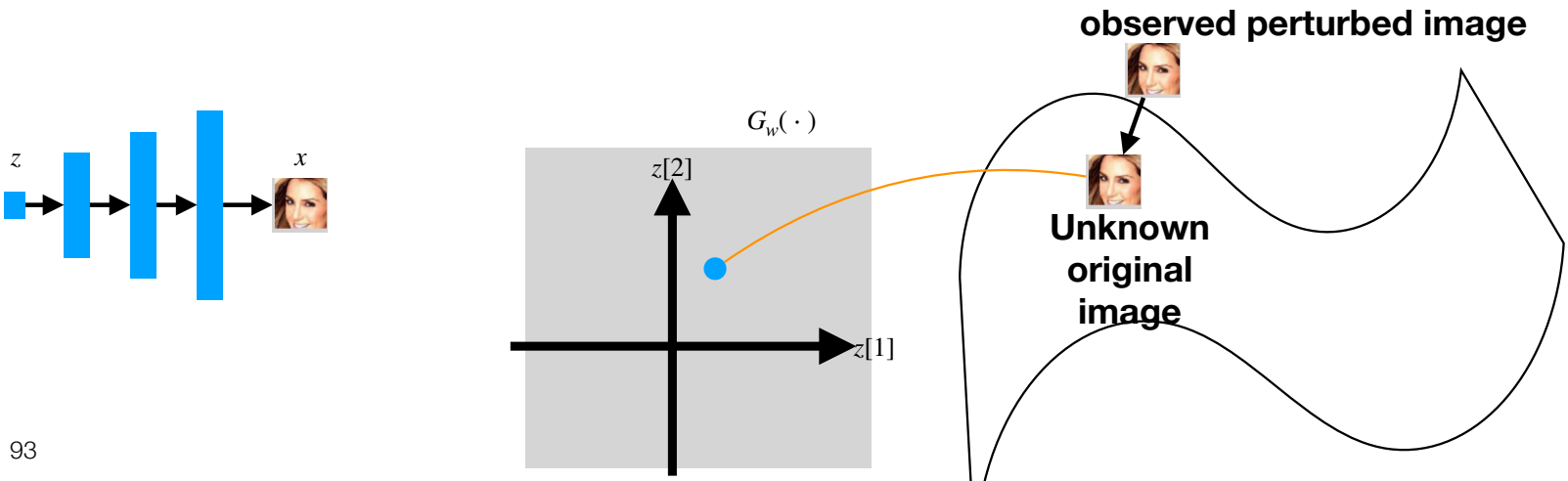
Deep Image prior

- Given a trained generator W that knows the manifold of natural images, find the latent vector Z that

$$\text{minimize}_z \quad \ell\Big(G_w(z$$

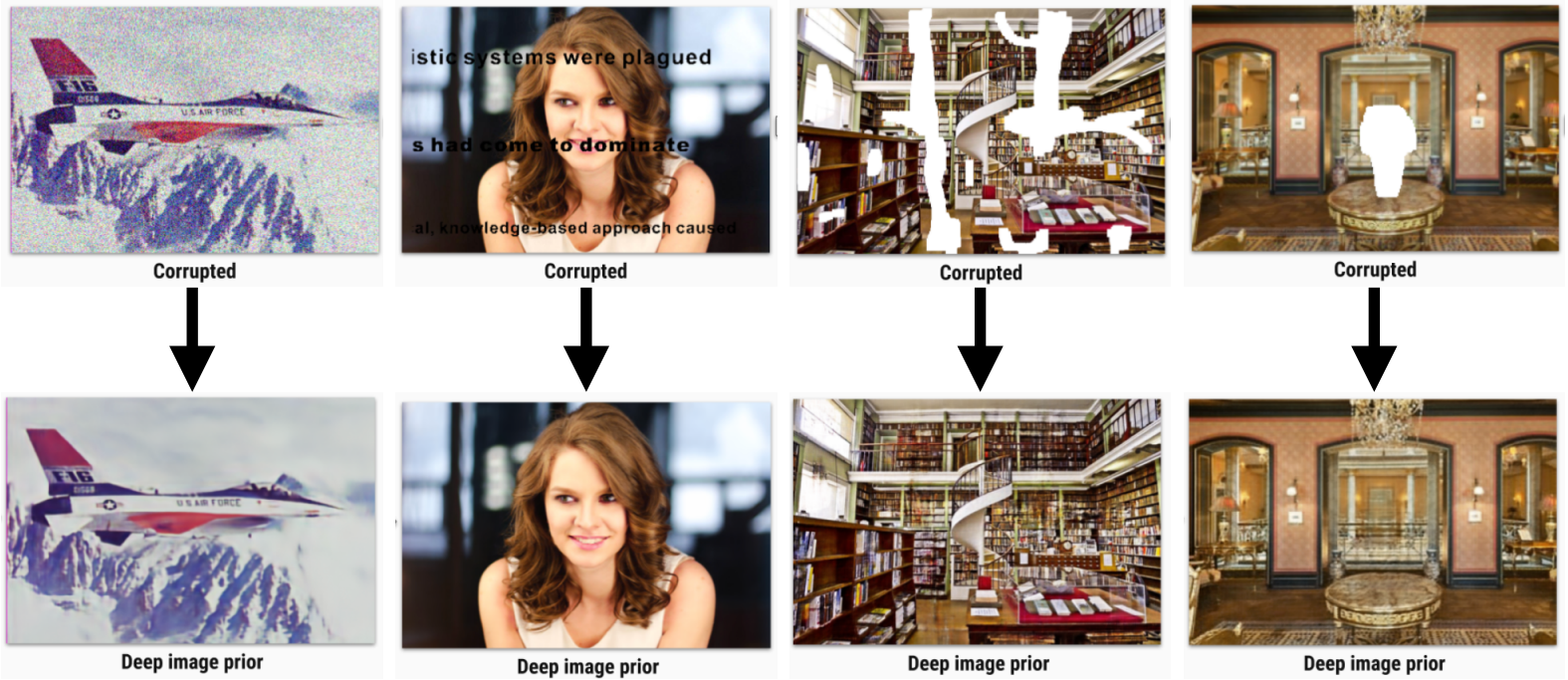
$$\Big)$$

- let $G_w(z)$ be the recovered image



Deep image prior

- **deep image prior** does amazing recovery, **without training**



Deep image prior

- fix z to be something random and find w that

$$\underset{w}{\text{minimize}}_{z} \quad \ell\Big(C$$

$$\Big)$$

and let $G_w(z)$ be the recovered image

https://www.youtube.com/watch?v=kSLJriaOumA&feature=youtu.be

Questions?

Questions?