Imperial College London

AI in Medicine I

Unsupervised and self-supervised learning

Felix Meissen
I31 – Chair for AI in Medicine and Healthcare
Faculty of Informatics and Medicine

Some slides created by B. Glocker, L. Le Folgoc

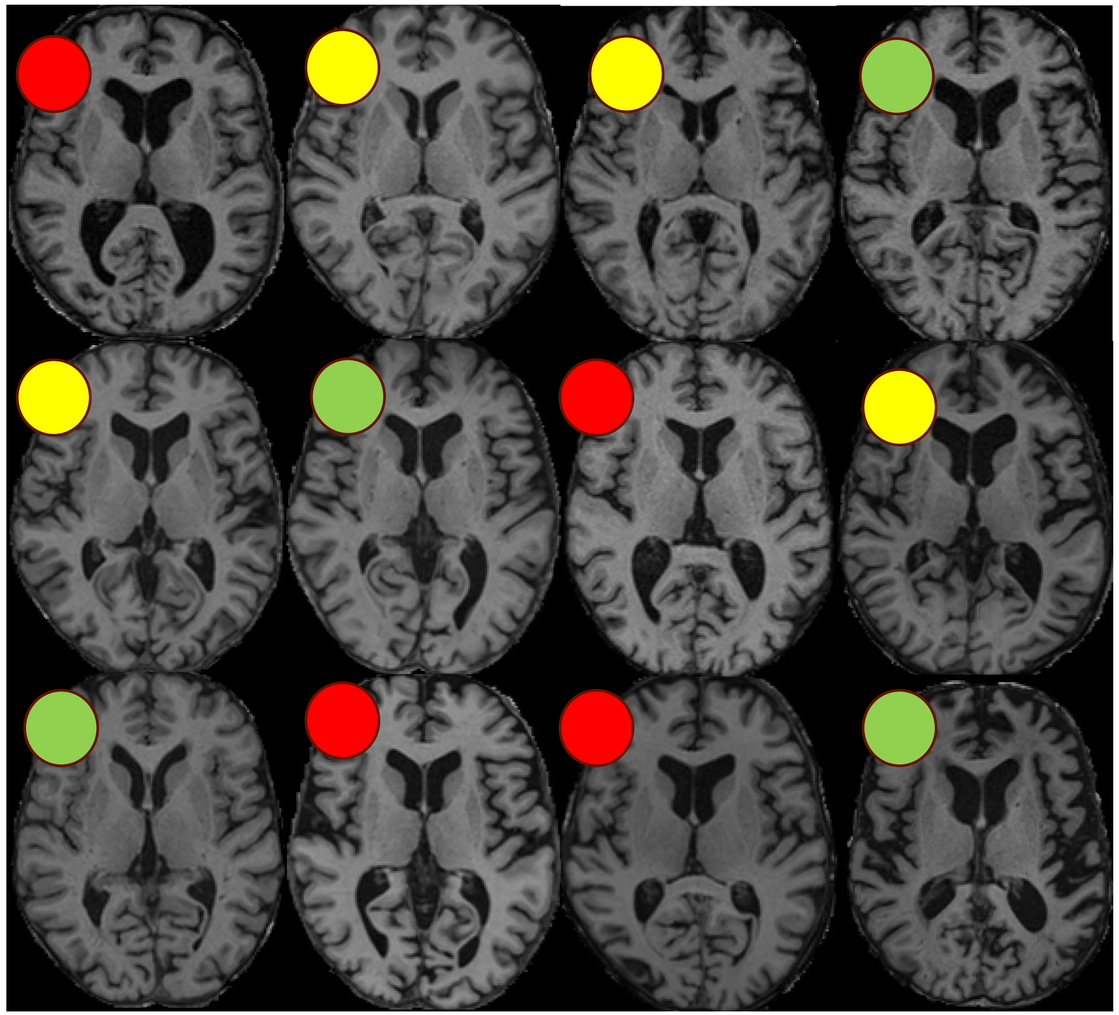
Today

- Clustering:
  - k-Means
  - Gaussian Mixture Model
- Dimensionality Reduction:
  - PCA
  - Autoencoders
- Generative Modelling:
  - VAEs
  - GANs
- Self-supervised learning
  - Pretext tasks
  - Contrastive Learning

Imperial College London

Clustering

Clustering – Example: Population stratification

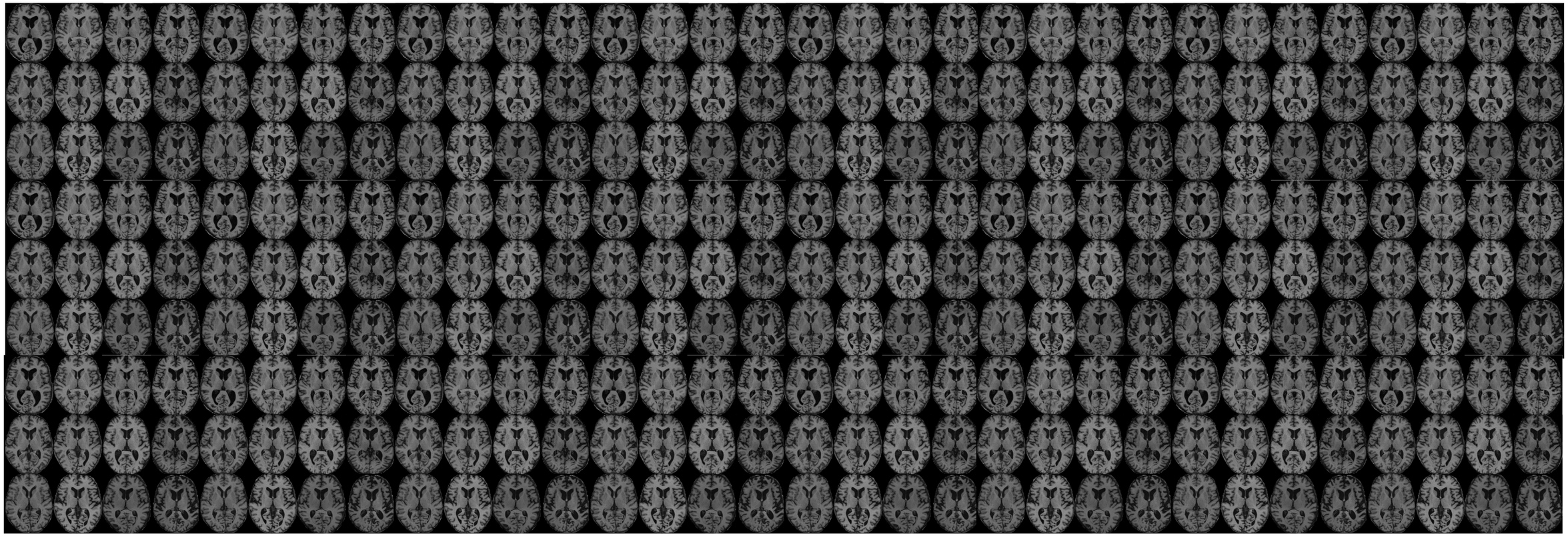


Normal

Mild Cognitive Impairment

Alzheimer’s

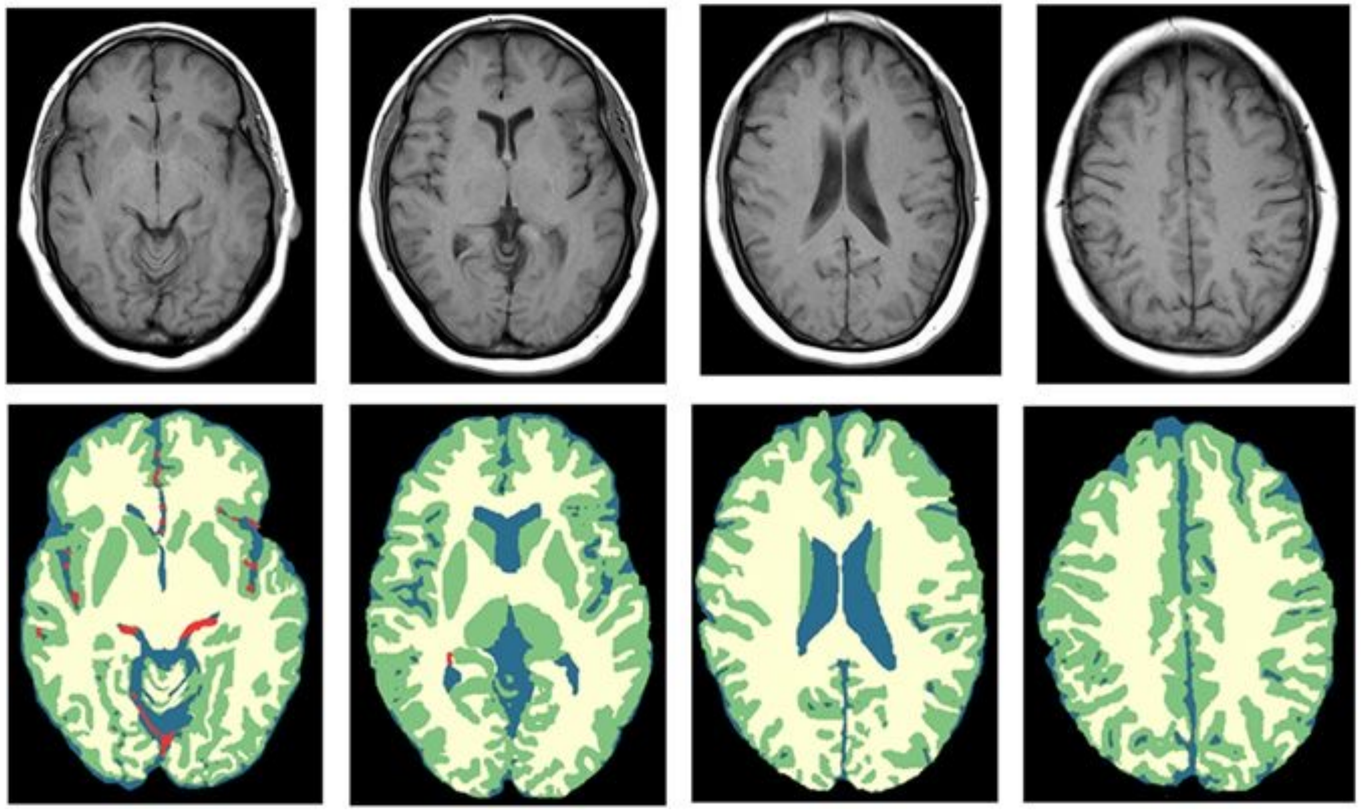
Clustering – Example:
Population stratification, e.g. discovery of pathologies



- Sometimes gathering annotations / “ground truth” can be costly or cumbersome
- Clustering could also identify novel ways to group subsets of data semantically

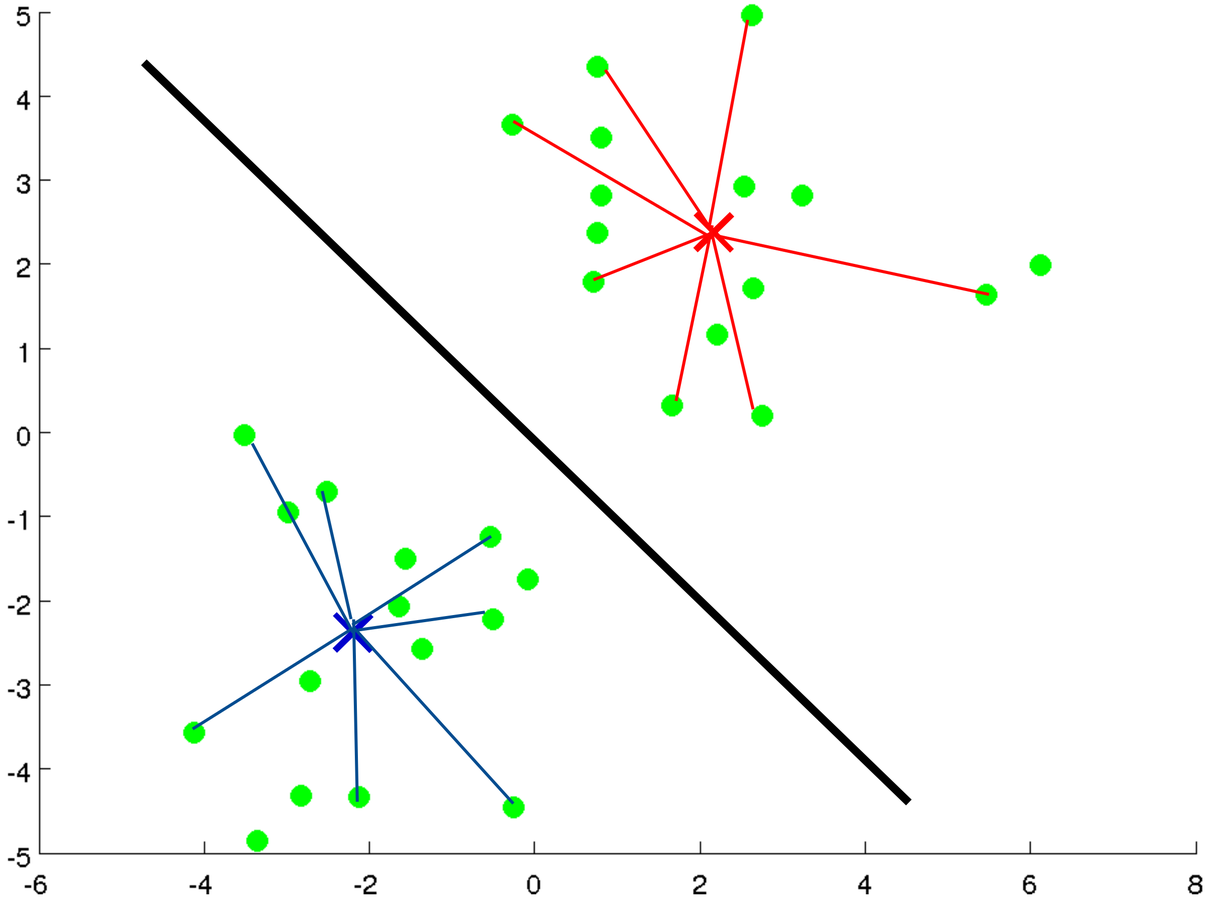
Clustering – Example: Unsupervised image segmentation

- Goal: MR soft tissue segmentation



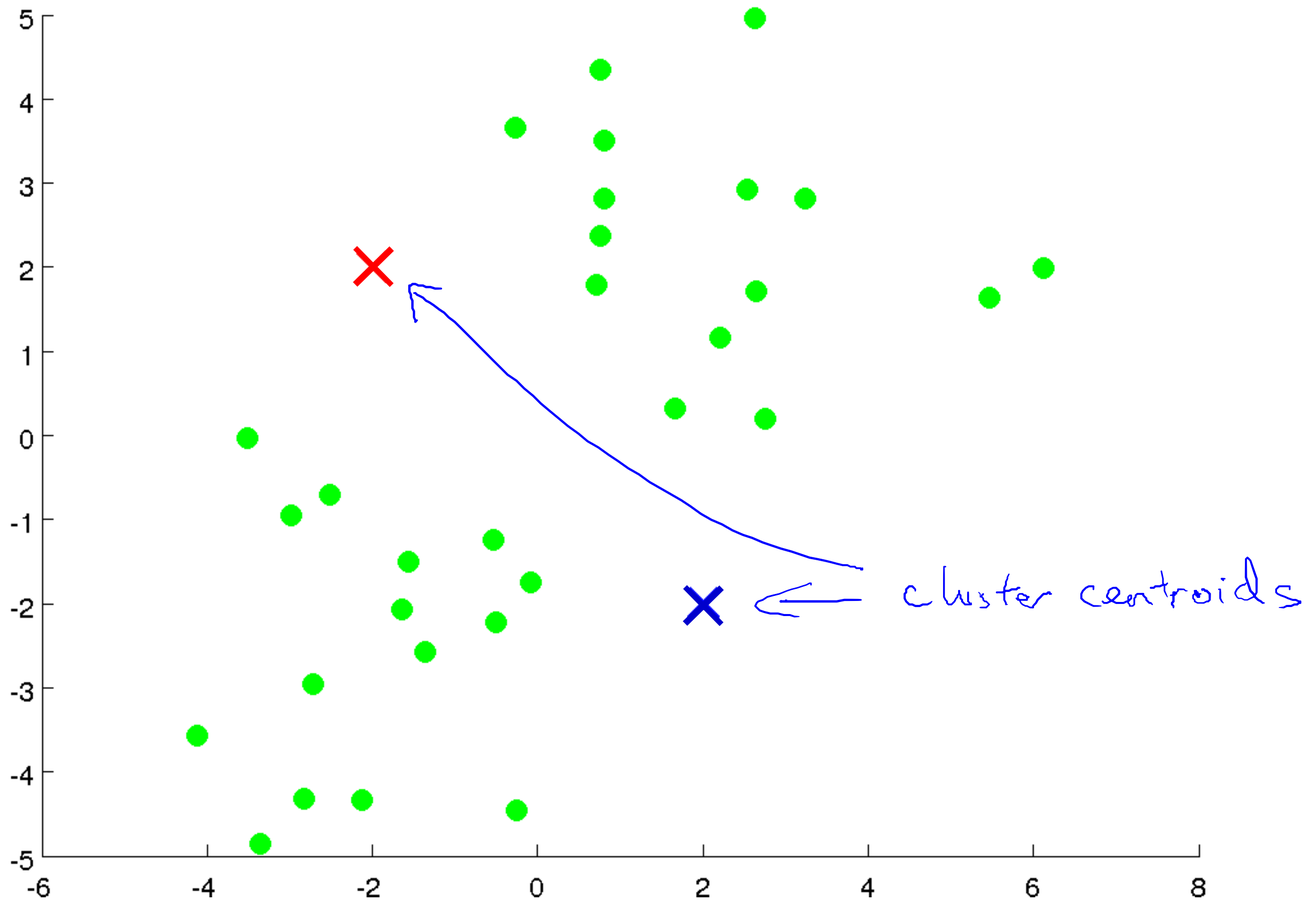
Green ≡ Gray Matter, Yellow ≡ White Matter, Blue ≡ Cerebro-Spinal Fluid
Adapted from Nature's *Scientific Reports* **volume 7**, Article number: 119 (March 2017)

K-Means Clustering



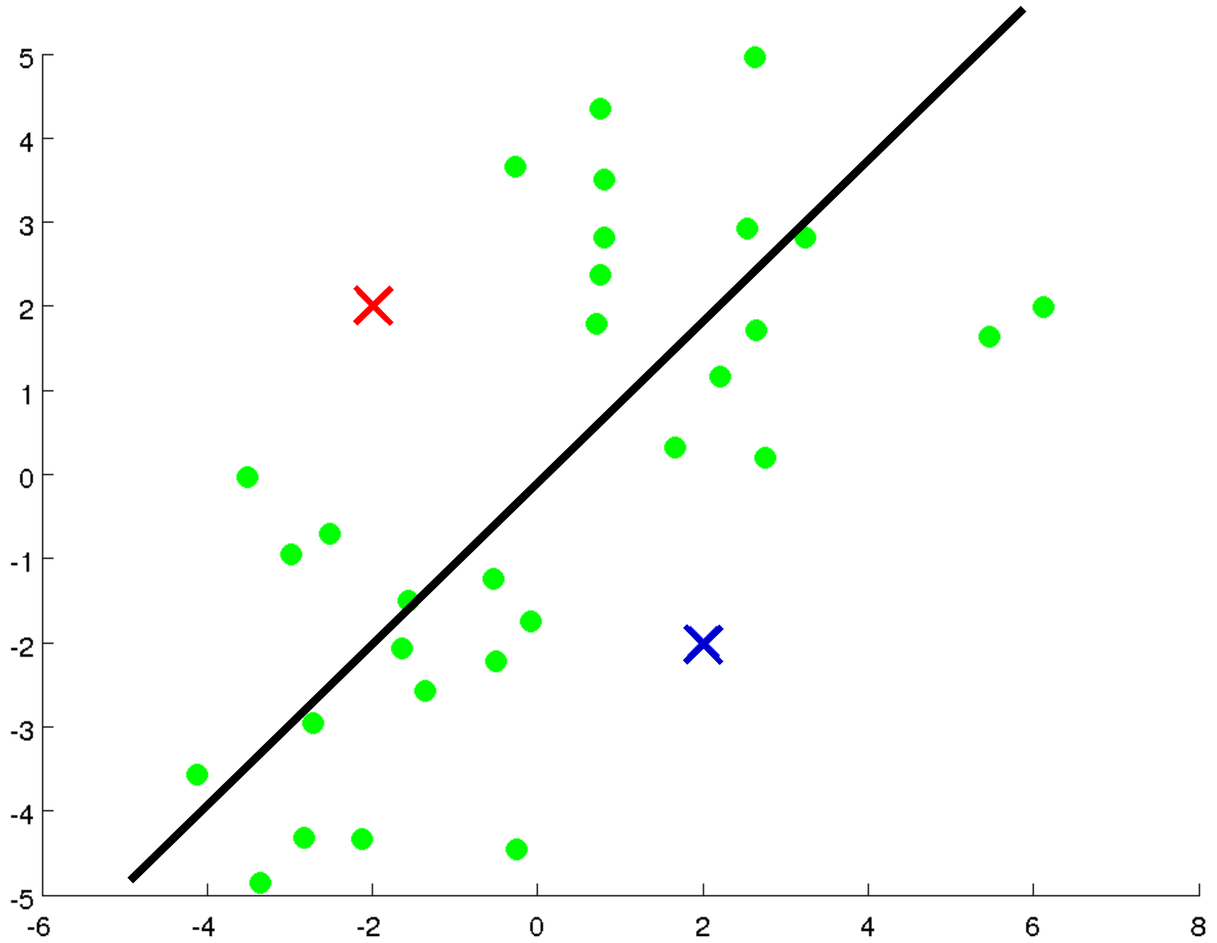
Adapted from Andrew Ng

K-Means Clustering



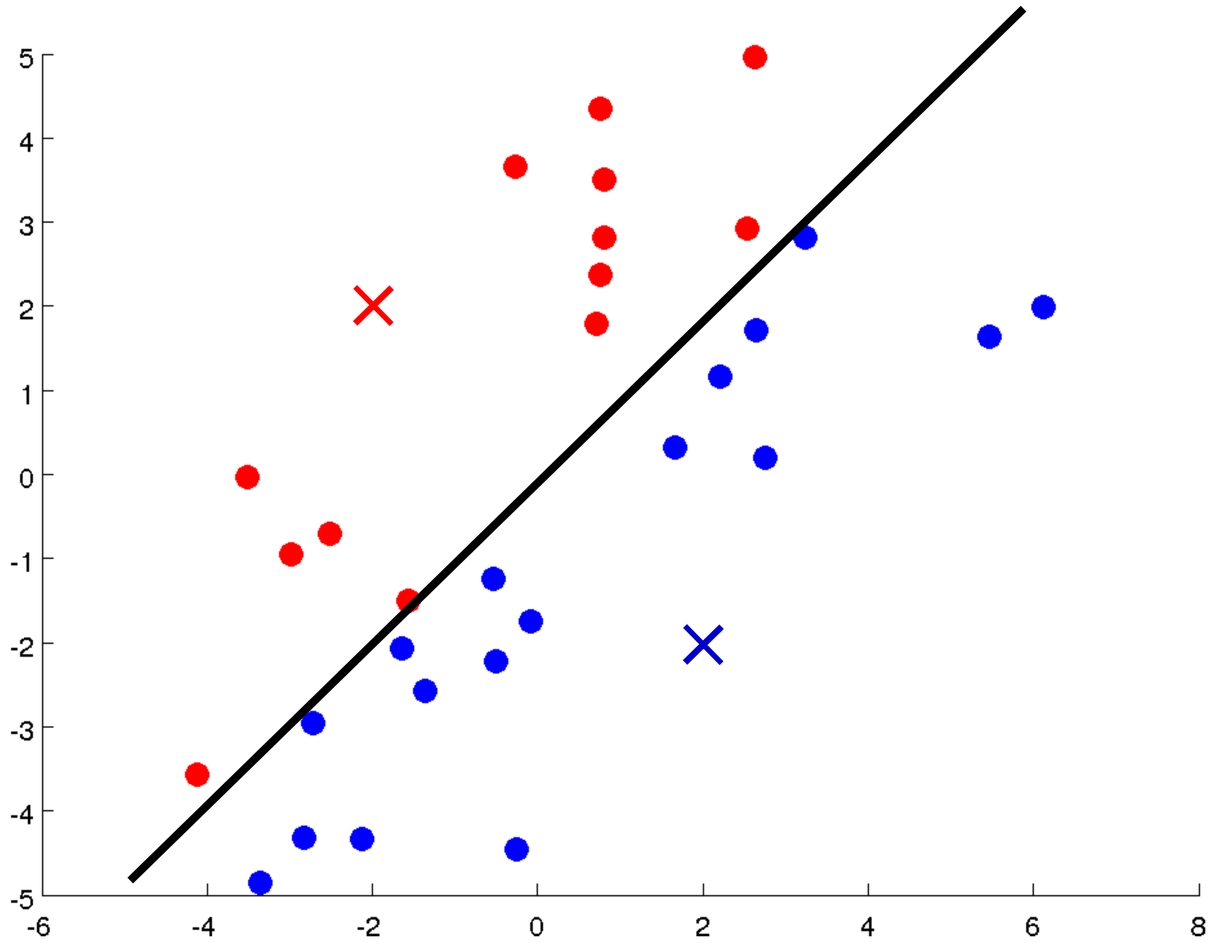
Adapted from Andrew Ng

K-Means Clustering



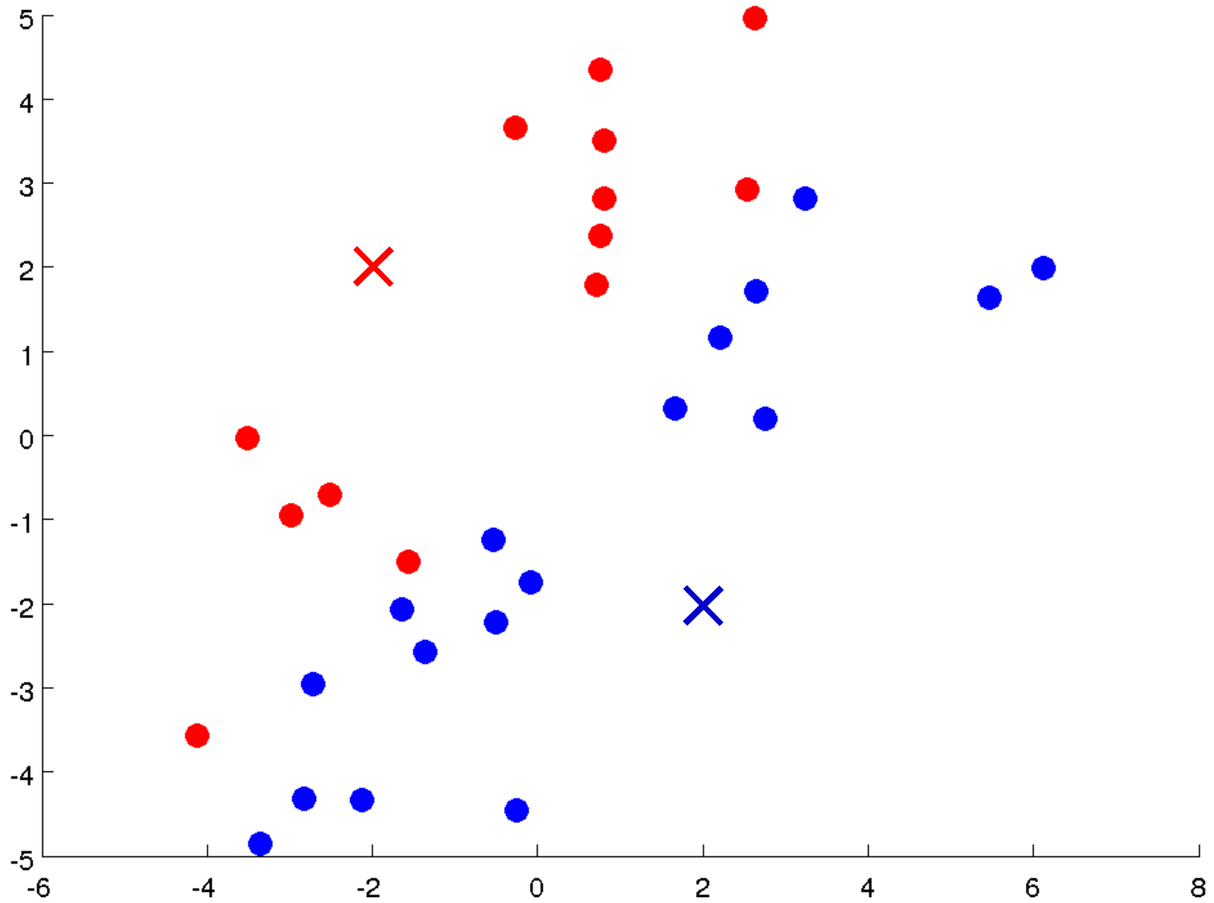
Adapted from Andrew Ng

K-Means Clustering



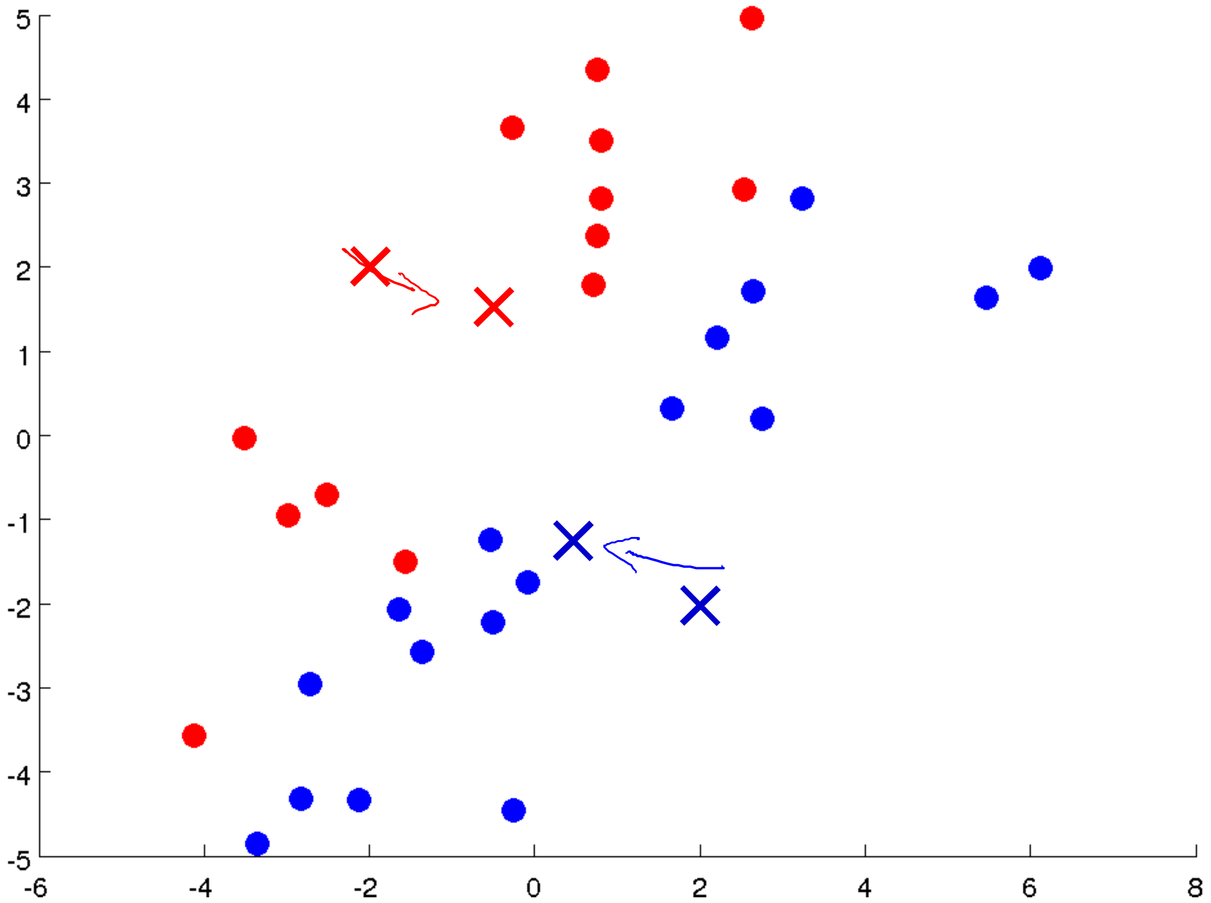
Adapted from Andrew Ng

K-Means Clustering



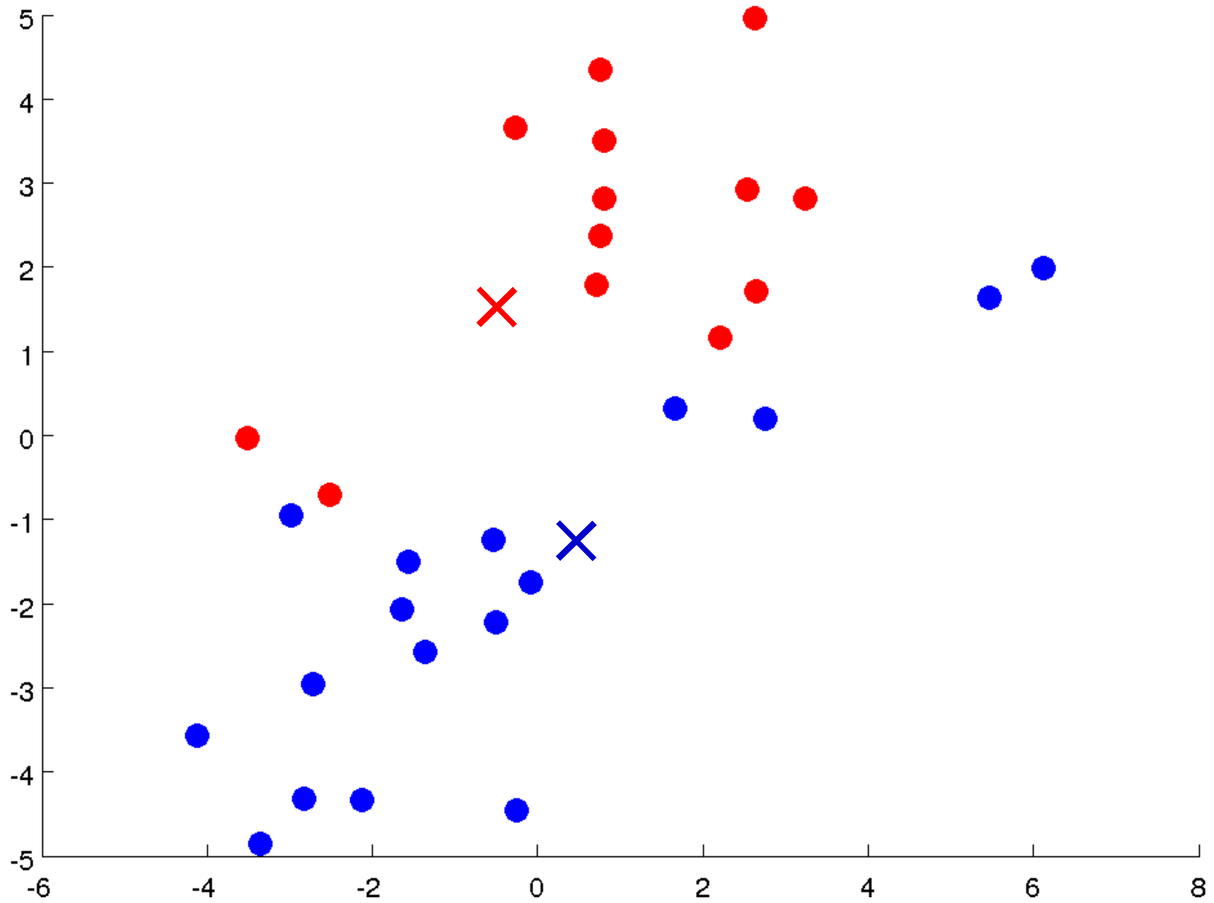
Adapted from Andrew Ng

K-Means Clustering



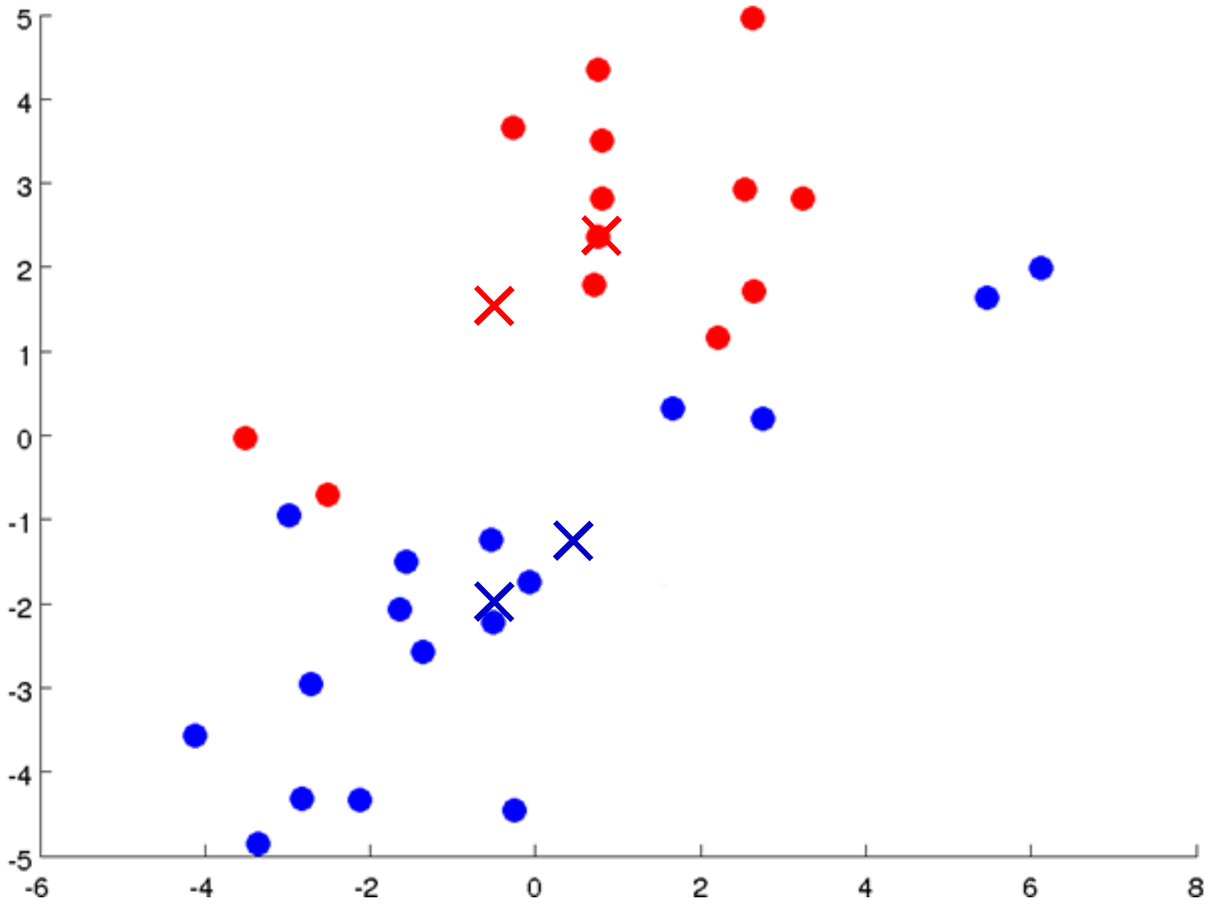
Adapted from Andrew Ng

K-Means Clustering



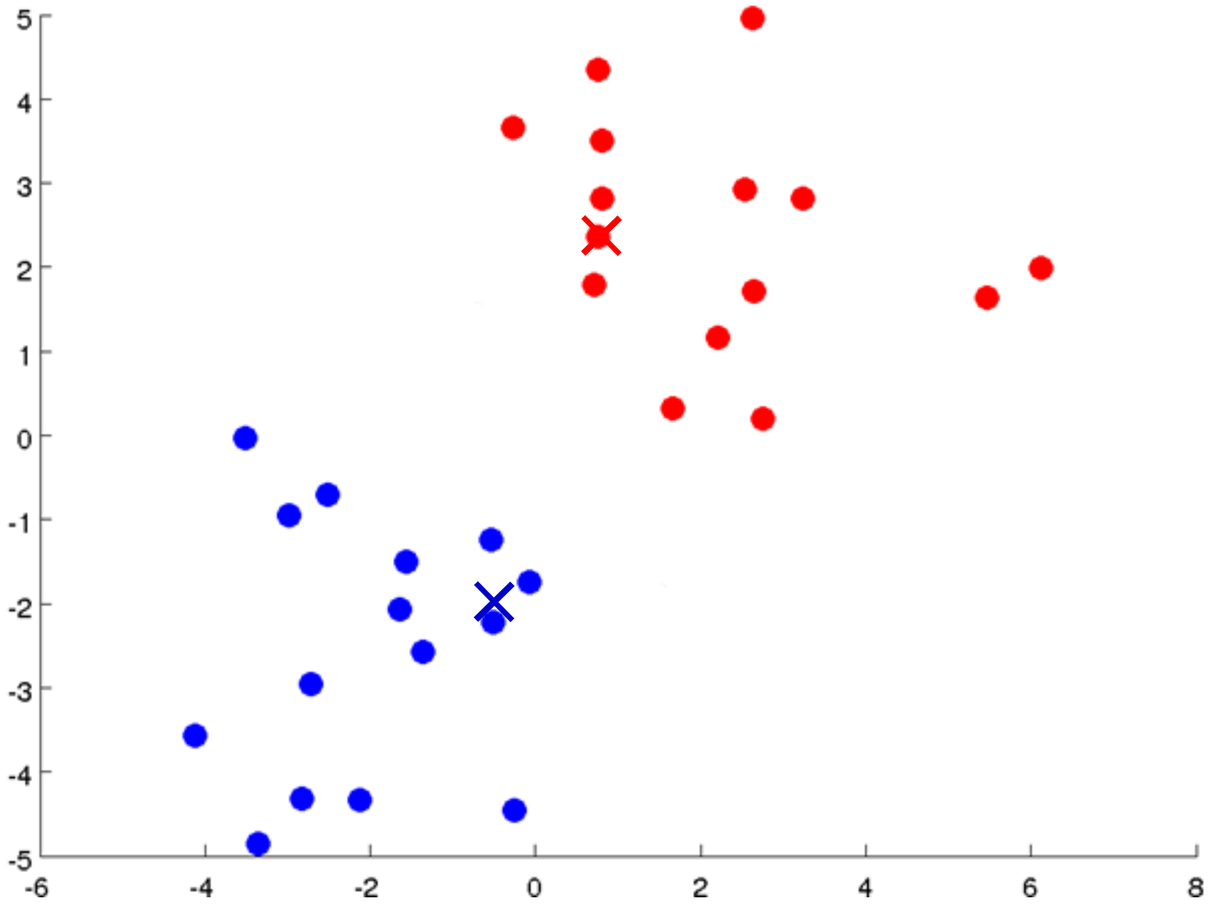
Adapted from Andrew Ng

K-Means Clustering



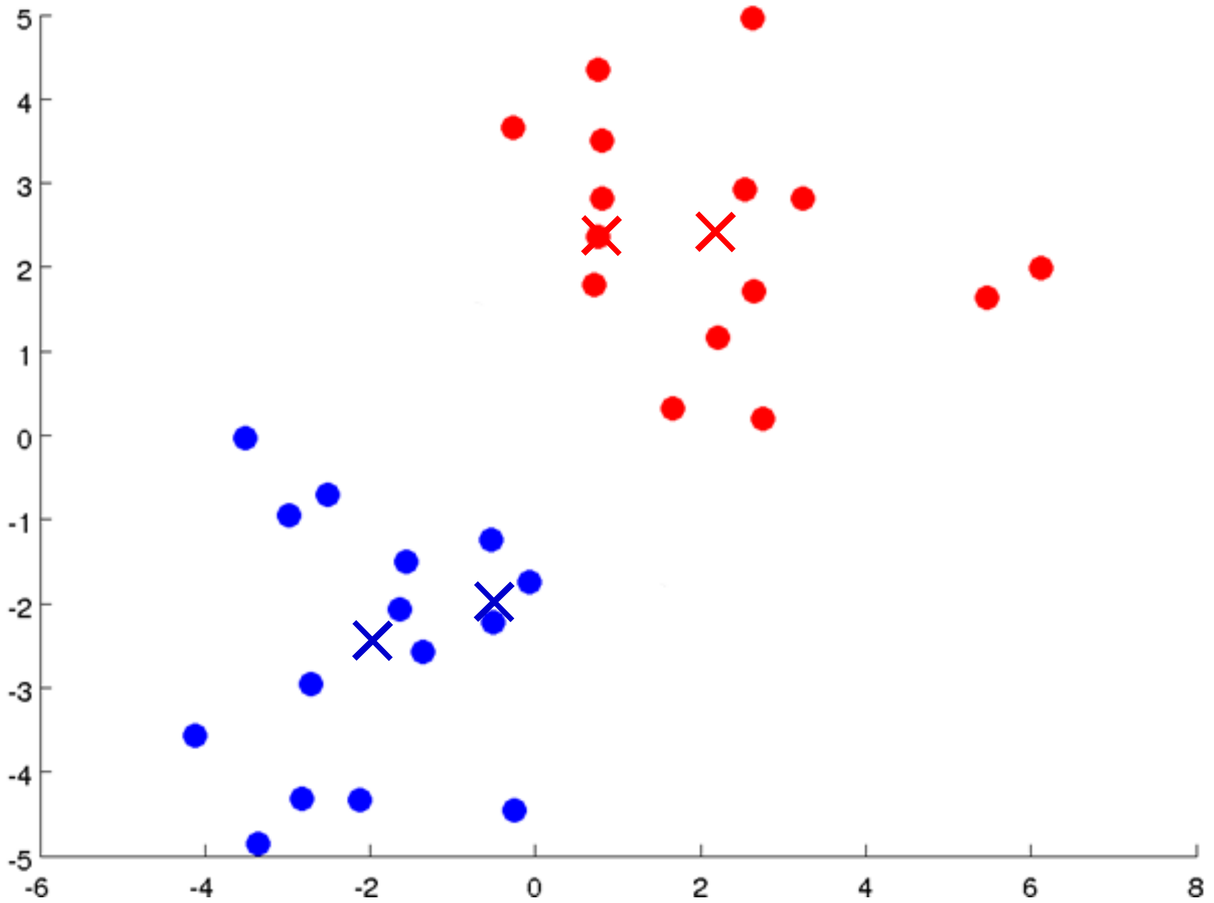
Adapted from Andrew Ng

K-Means Clustering



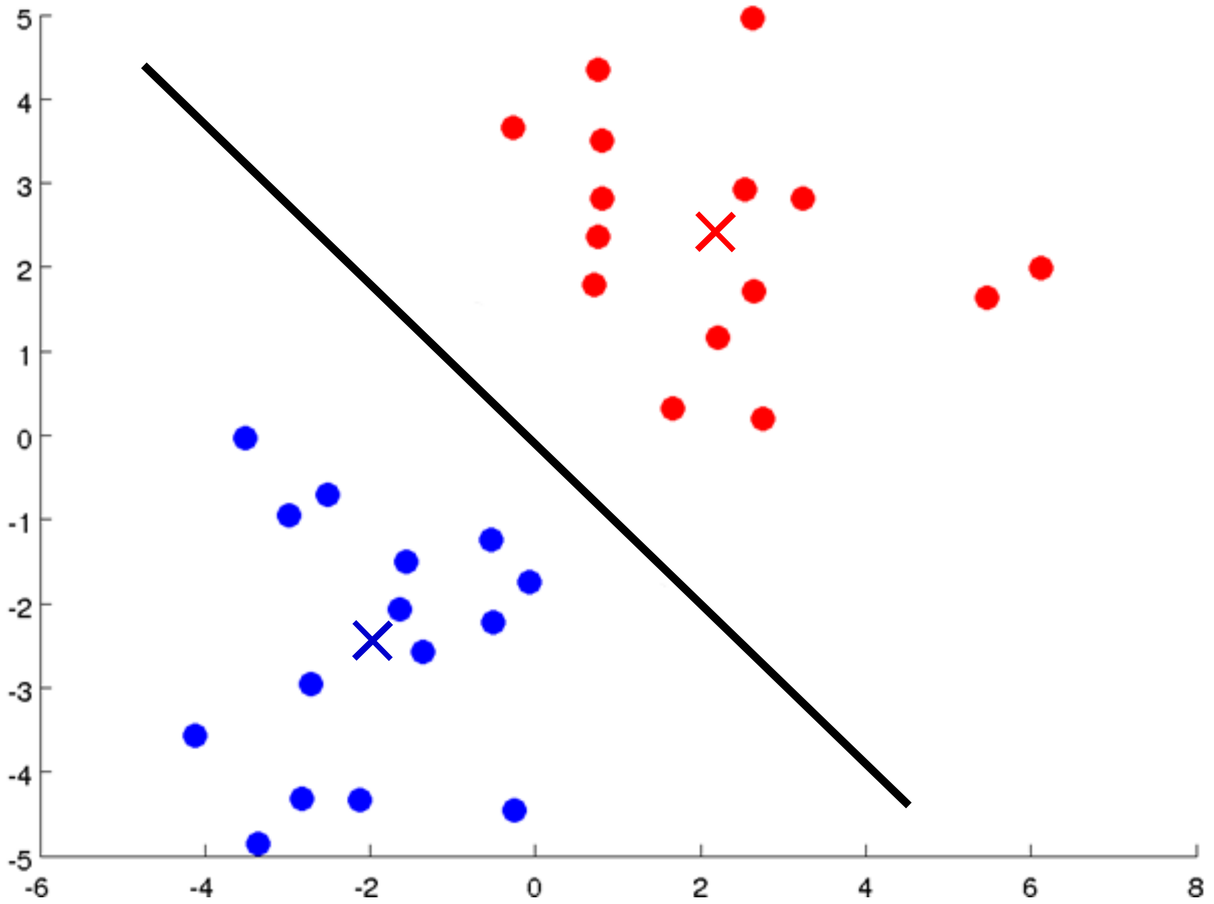
Adapted from Andrew Ng

K-Means Clustering



Adapted from Andrew Ng

K-Means Clustering



Adapted from Andrew Ng

K-Means Clustering

Input: $x_n, 1 \leq n \leq N$

Algorithm:

1. Initialize centroids $\mu_k, 1 \leq k \leq K$: e.g. randomly choose K distinct input points
2. Until stopping criterion met (e.g. cluster assignments converge):
3. For each point n, $c_n :=$ index $\in \{1, \ldots, K\}$ of the centroid closest to x_n
4. For each cluster k, $\mu_k :=$ average of the points assigned to cluster k

Output: Cluster centroids and/or cluster assignments

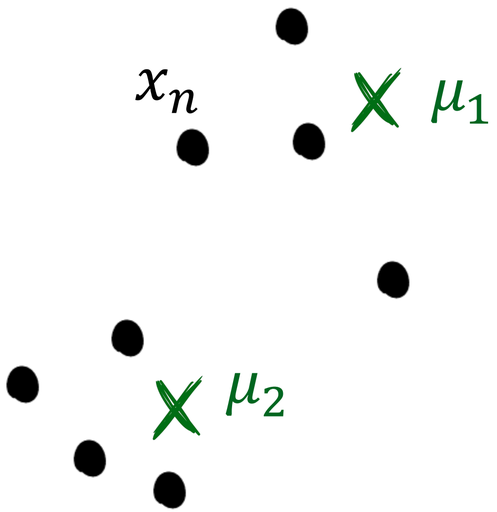
Does K-Means optimize a cost function?

- Solves initialization problem:
  1. Multiple reinitializations
  2. Pick the best cost

- Explains some properties of K-Means → new ways to improve it!

K-Means cost function

- Recall the algorithm:
  - For each point n, c_n := index $\in \{1, \ldots, K\}$ of the centroid closest to x_n
  - For each cluster k, μ_k := average of the points assigned to cluster k



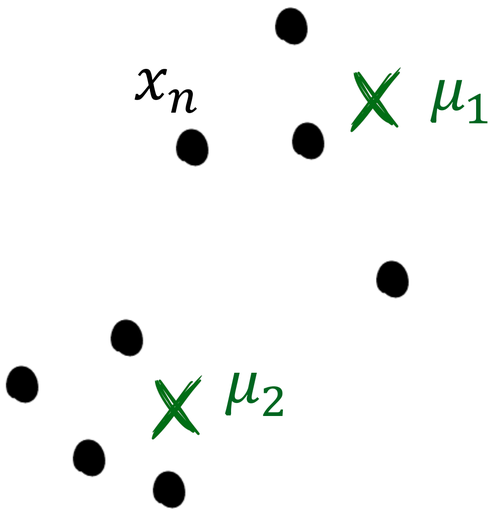
$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_k \sum_n z_{nk} \|x_n - \mu_k\|^2$$

$$s.t. \;\; \forall n, k, z_{nk} \in \{0,1\}$$

$$\forall n, \sum_k z_{nk} = 1$$

K-Means cost function

- Recall the algorithm:
  - For each point n, $c_n :=$ index $\in \{1, \ldots, K\}$ of the centroid closest to x_n
  - For each cluster k, $\mu_k :=$ average of the points assigned to cluster k



$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_n \sum_k z_{nk} \|x_n - \mu_k\|^2$$

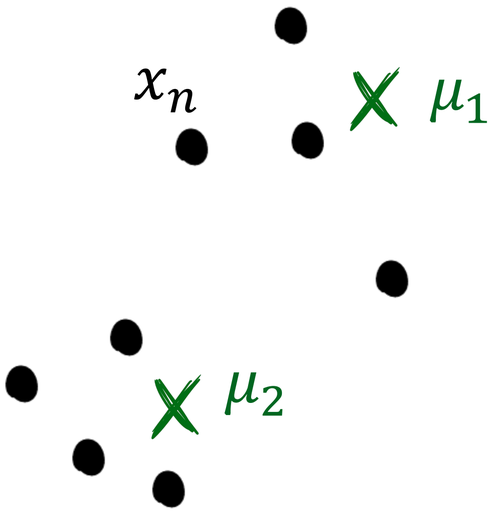
$$s.t. \quad \forall n, k, z_{nk} \in \{0,1\}$$

$$\forall n, \sum_k z_{nk} = 1$$

(Decouples over each point when centroids fixed)

K-Means cost function

- Recall the algorithm:
  - For each point n, $c_n :=$ index $\in \{1, \ldots, K\}$ of the centroid closest to x_n
  - For each cluster k, $\mu_k :=$ average of the points assigned to cluster k



$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_k \sum_n z_{nk} \|x_n - \mu_k\|^2$$

$$s.t. \;\; \forall n, k, z_{nk} \in \{0,1\}$$

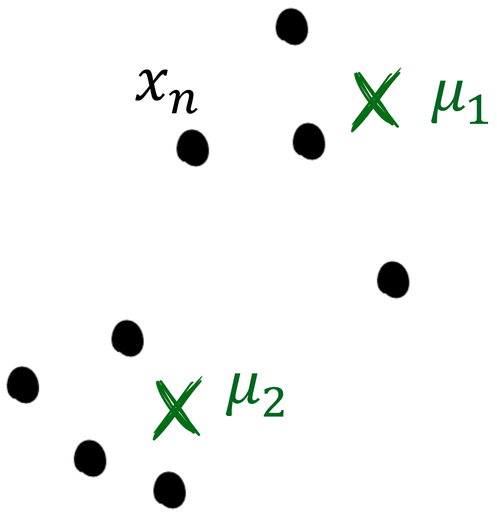
$$\forall n, \sum_k z_{nk} = 1$$

(Decoupled over each component when assignments fixed)

K-Means cost function

- Recall the algorithm:
  - For each point n, c_n := index $\in \{1, \dots, K\}$ of the centroid closest to x_n
  - For each cluster k, μ_k := average of the points assigned to cluster k

Cost monotonically ↓



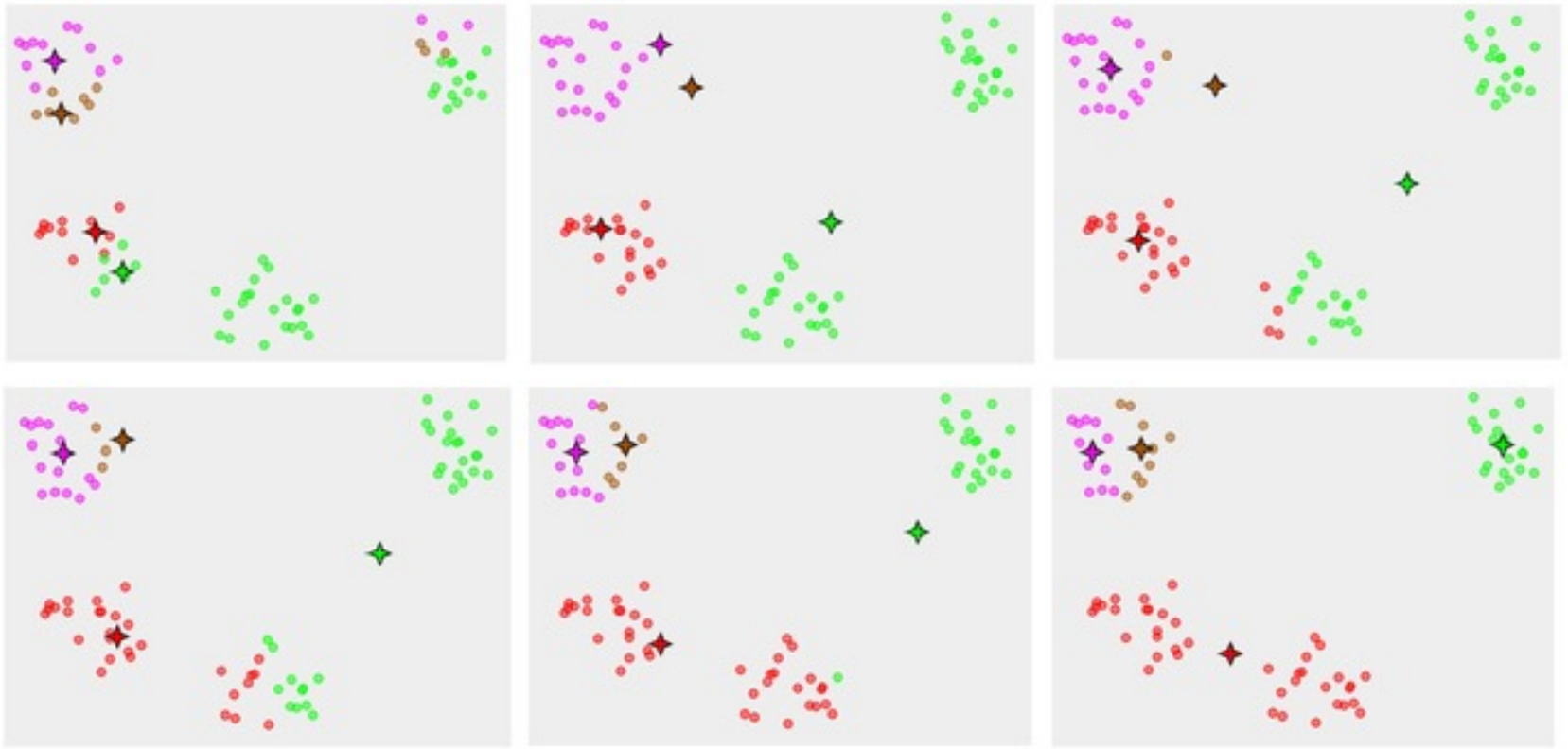
$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_n \sum_k z_{nk} \|x_n - \mu_k\|^2$$

$$s.t. \;\; \forall n, k, z_{nk} \in \{0,1\}$$

$$\forall n, \sum_k z_{nk} = 1$$

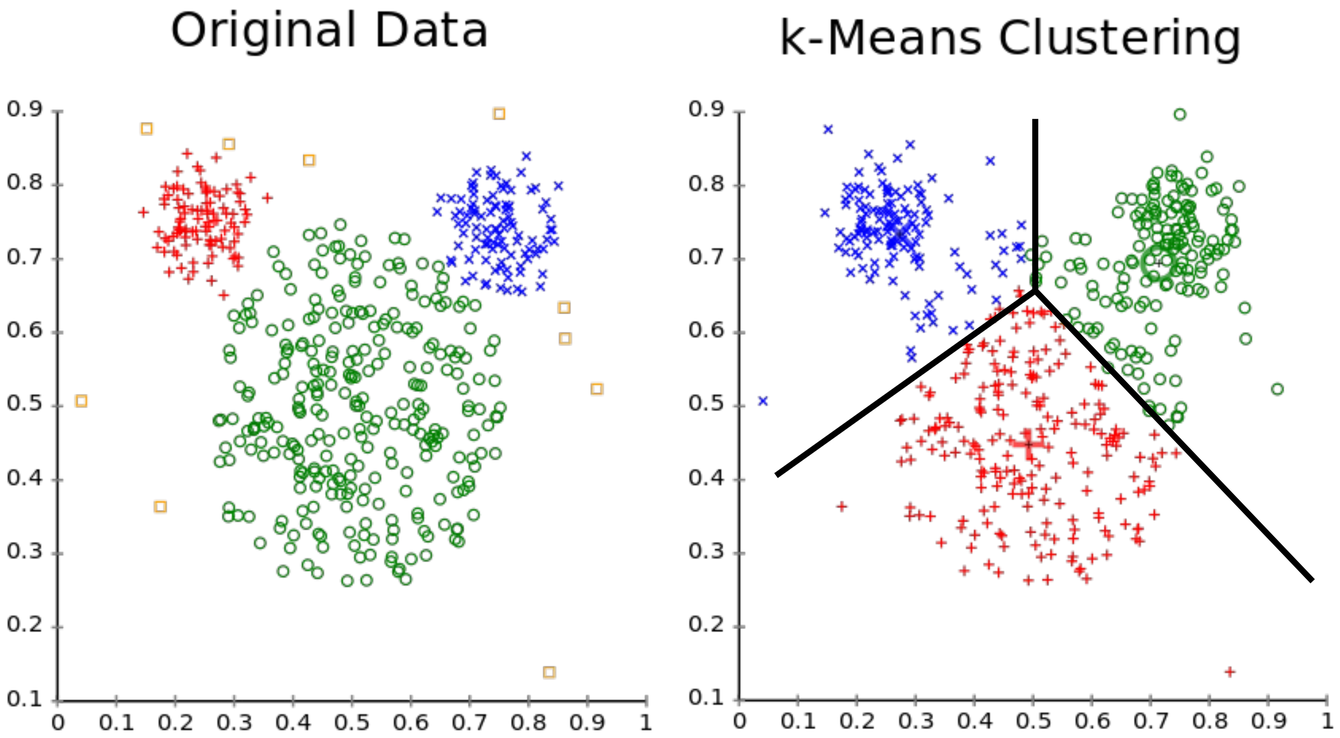
K-Means caveats

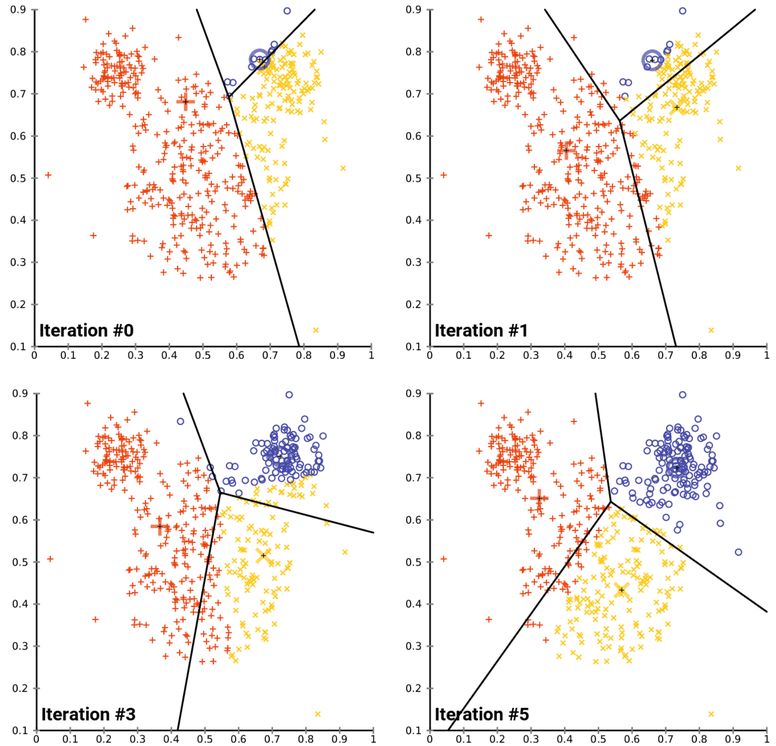
Bad initialization can yield bad clustering



from Wikipedia

K-Means caveats

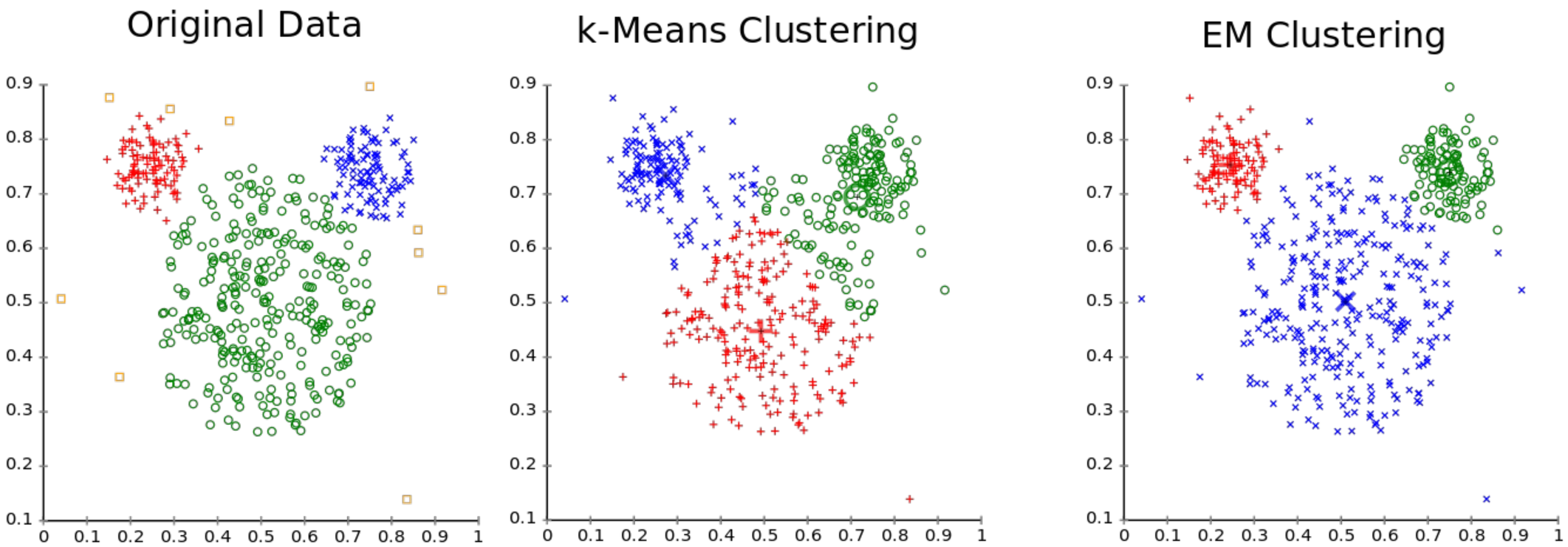




- Prefers ≃ same size clusters
- Crosses intuitive low-density splits

from Wikipedia

Mixture Model fit by EM



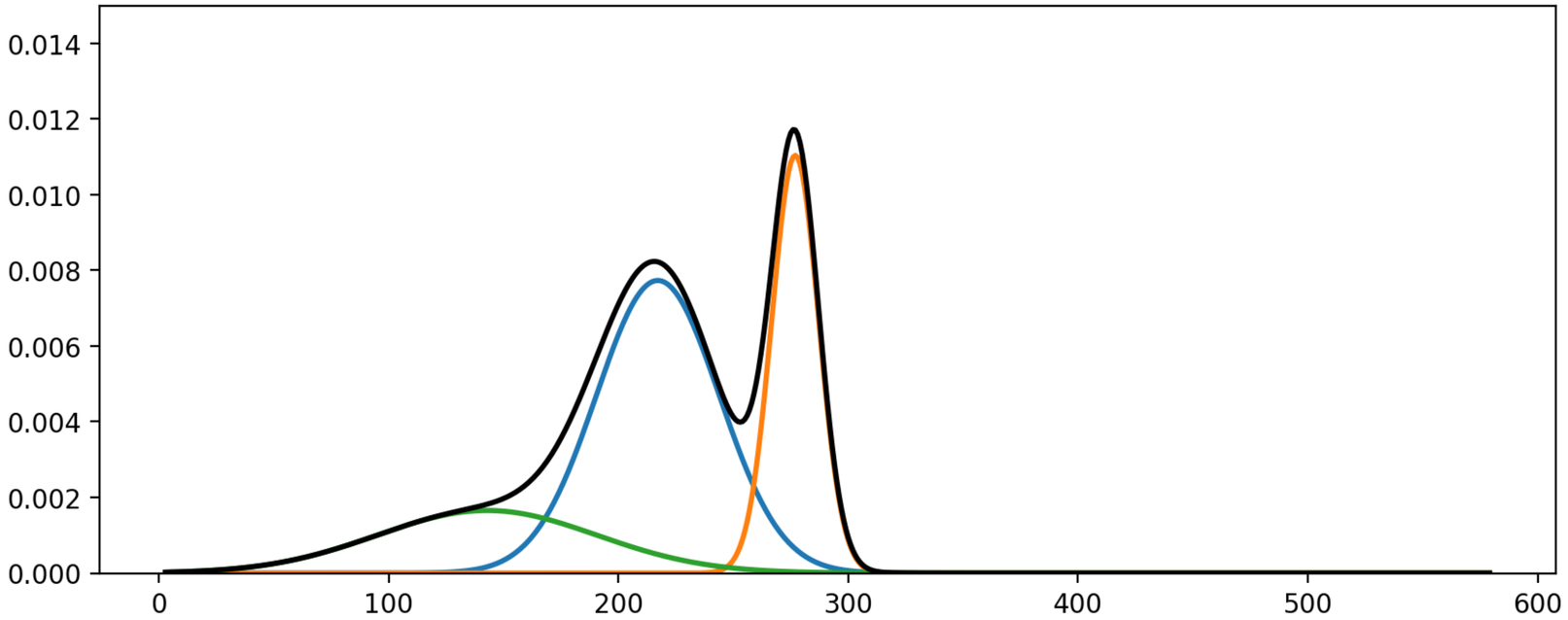
- To represent one cluster, use a statistical model (e.g. multivariate Gaussian)
- Use a mixture thereof for several clusters / the data!

from Wikipedia

Gaussian Mixture Model (GMM)

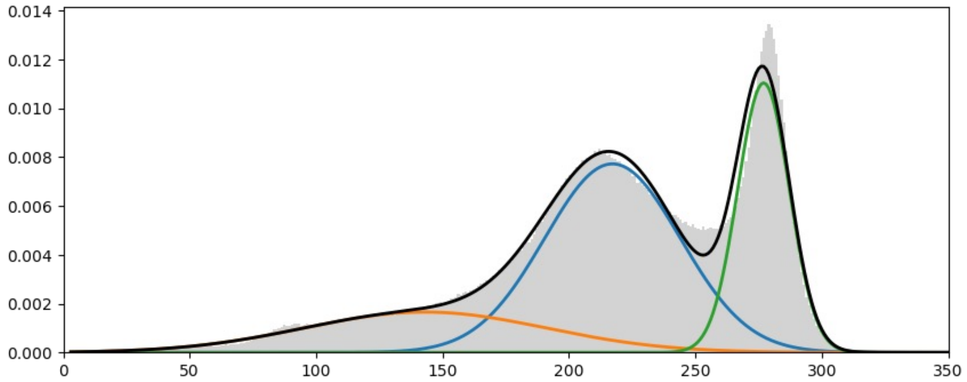
- Weighted sum of normal distributions:

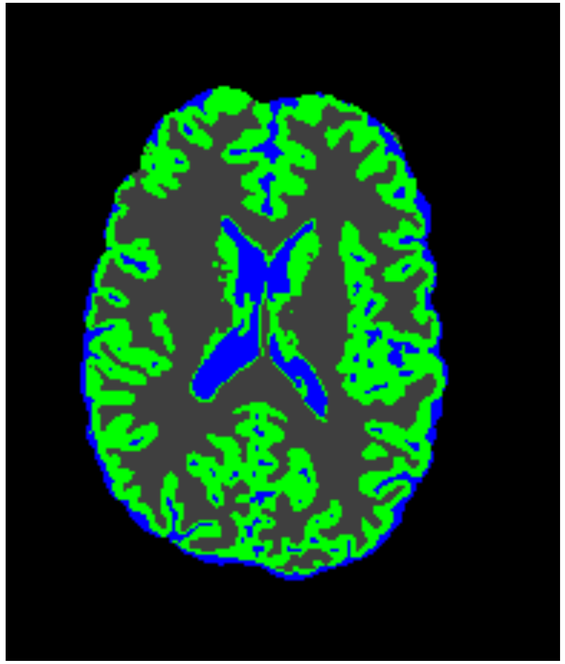
$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x|\mu_k, \sigma_k^2\right)$$



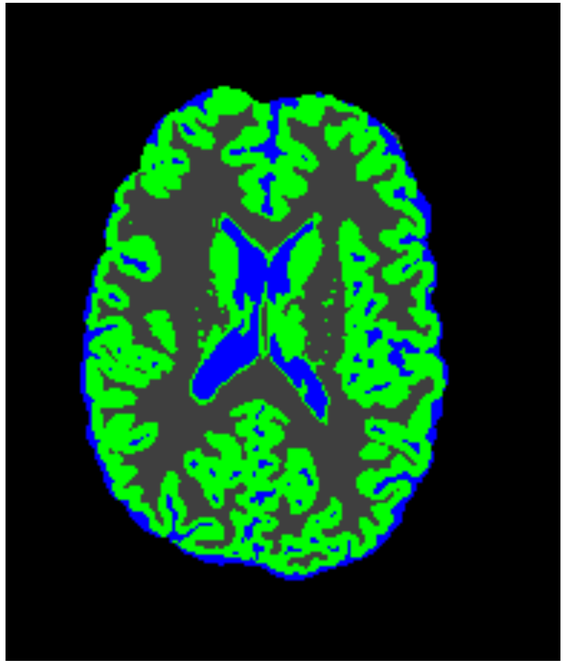
K-Means vs. GMMs

- 3 components





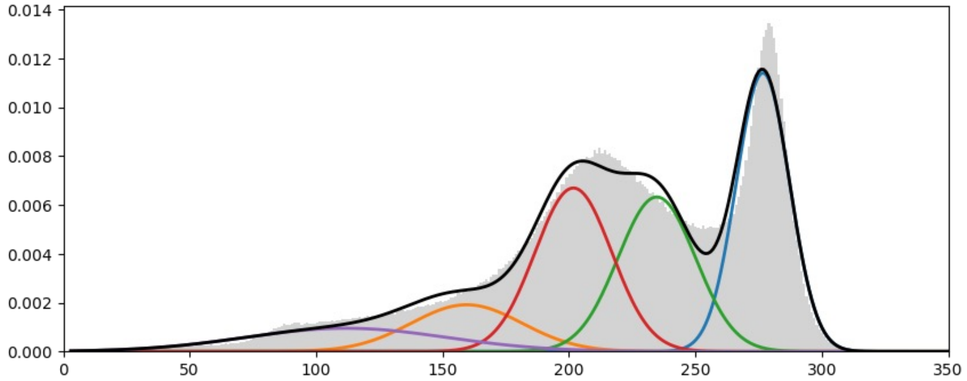
K-Means

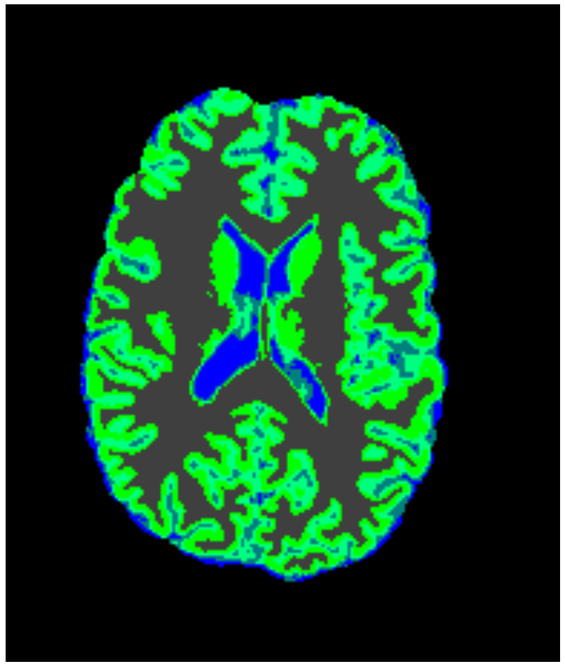


GMM

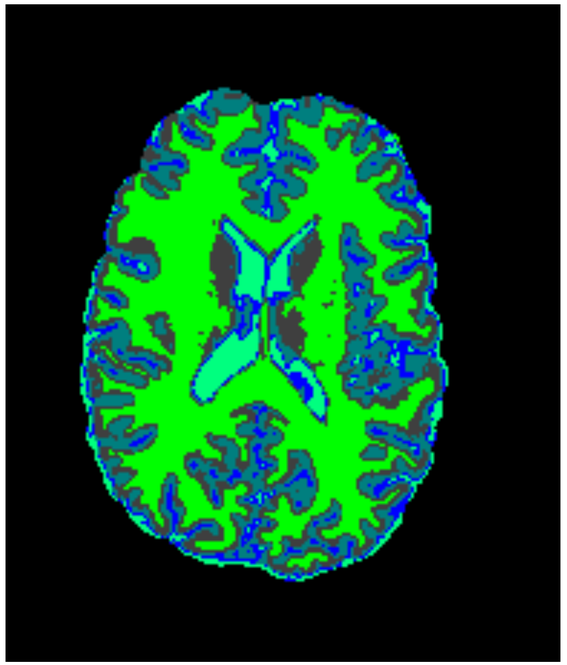
K-Means vs. GMMs

- 5 components





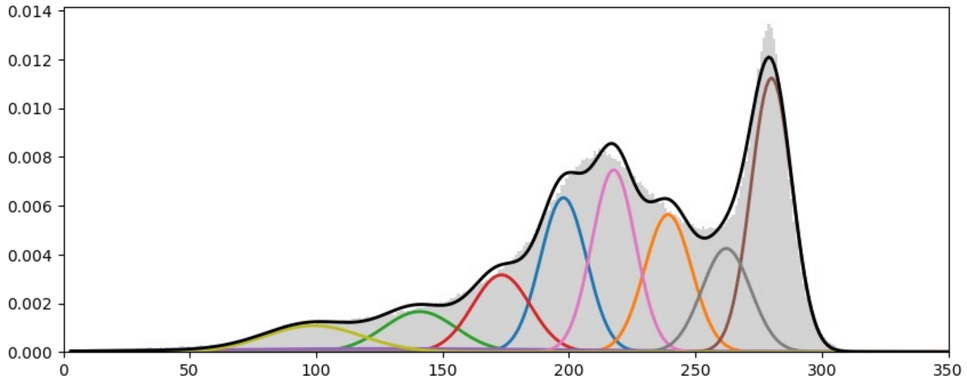
K-Means

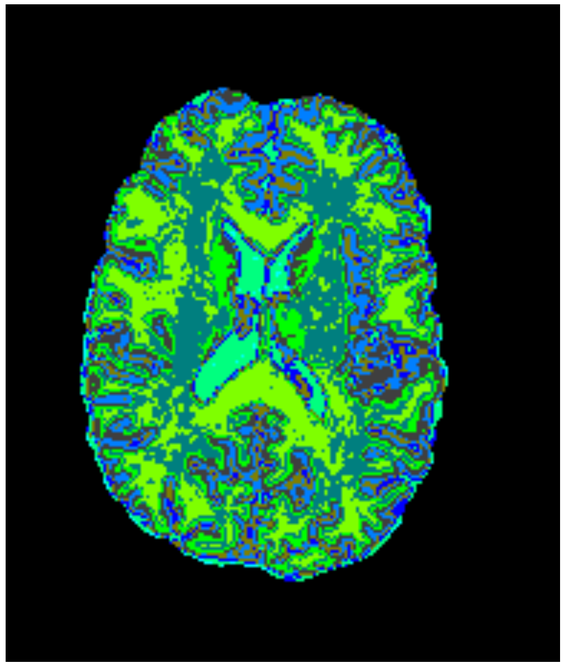


GMM

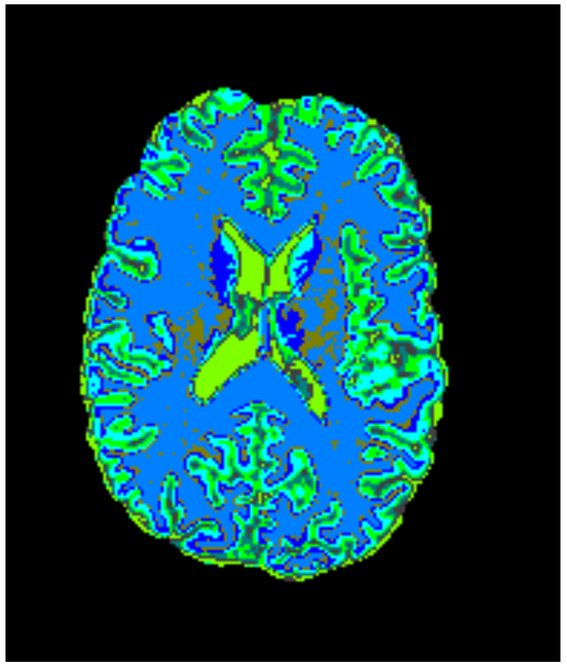
K-Means vs. GMMs

- 9 components





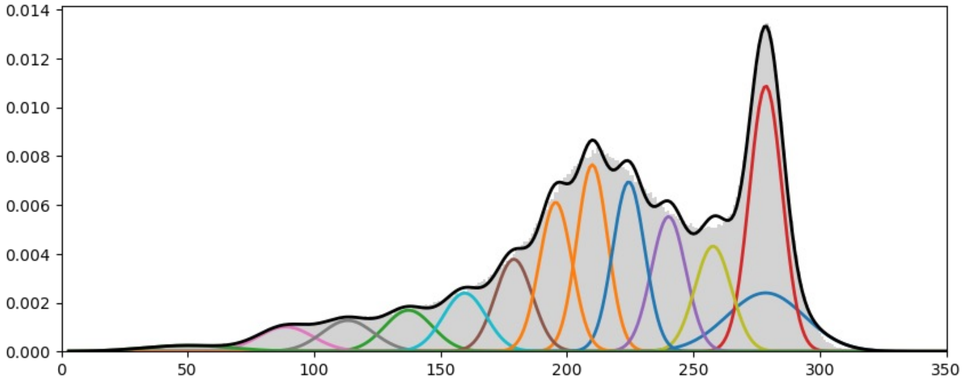
K-Means

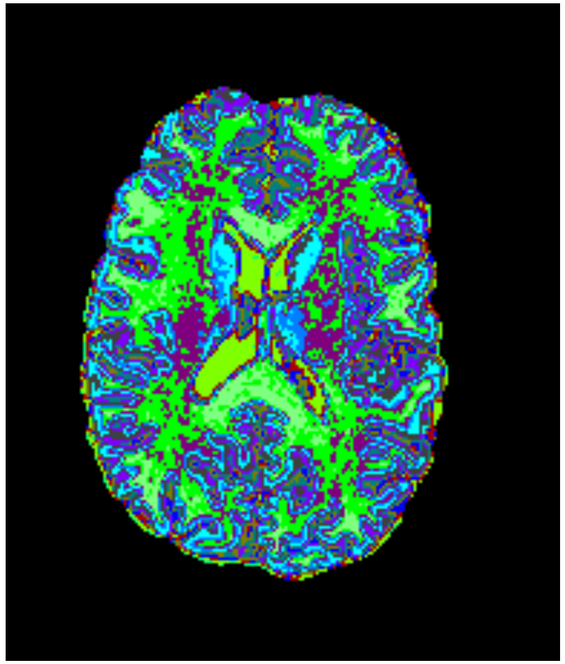


GMM

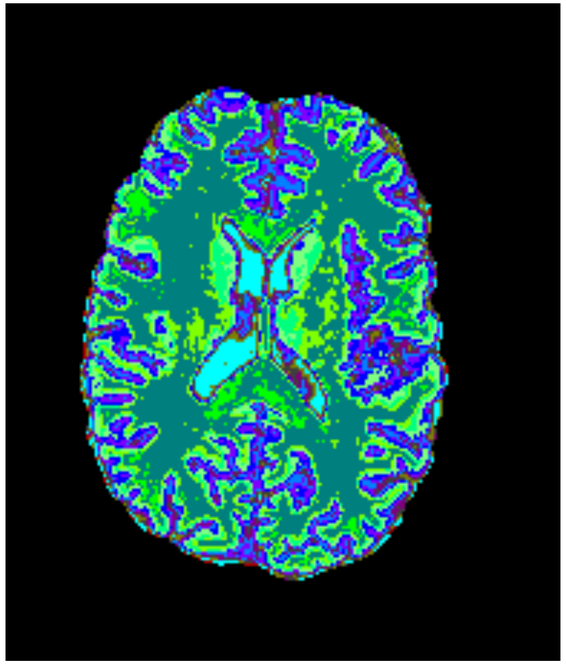
K-Means vs. GMMs

- 13 components





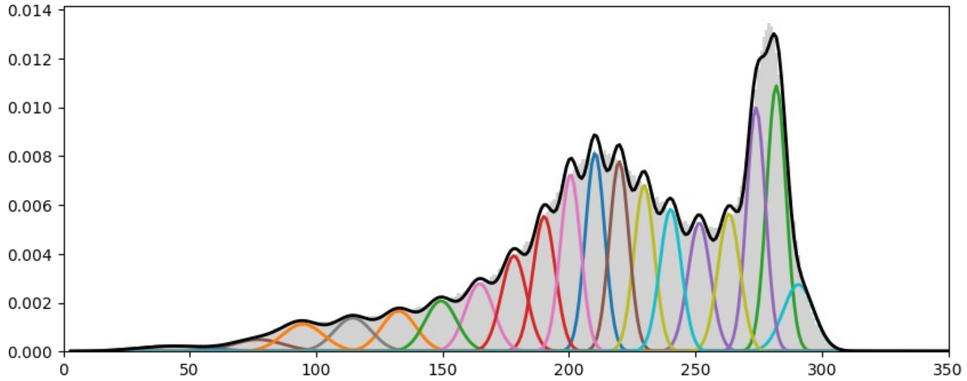
K-Means

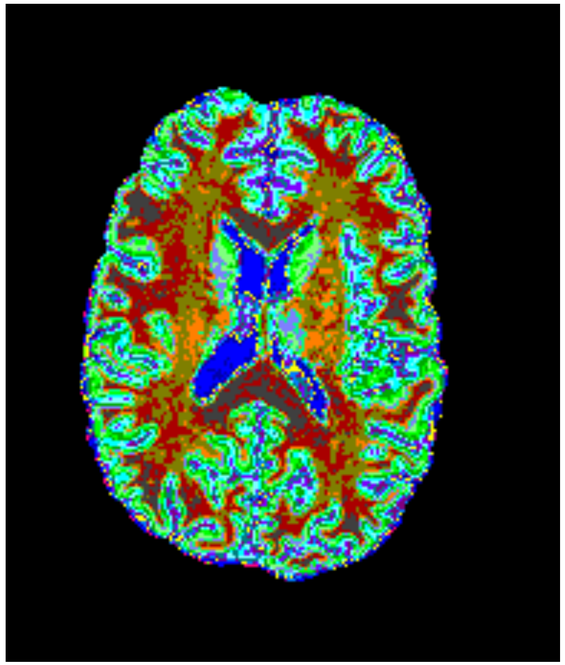


GMM

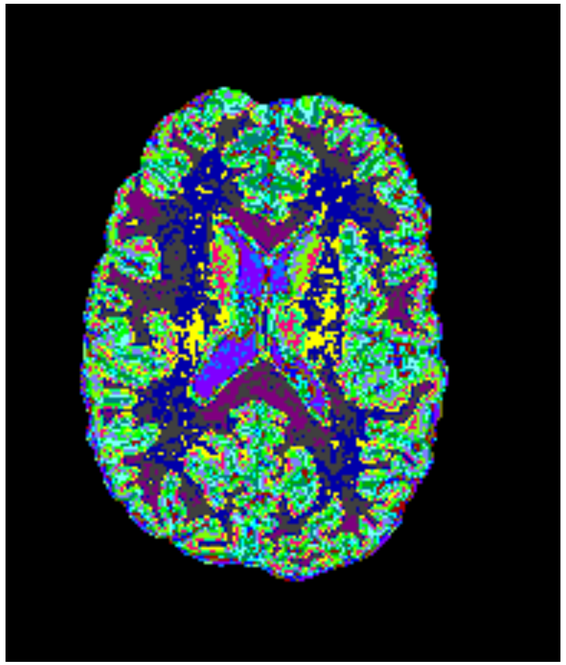
K-Means vs. GMMs

- 20 components





K-Means



GMM

Expectation-Maximization (optimization)

- Iterative method to estimate parameters of a statistical model, where the model depends on unobserved latent variables

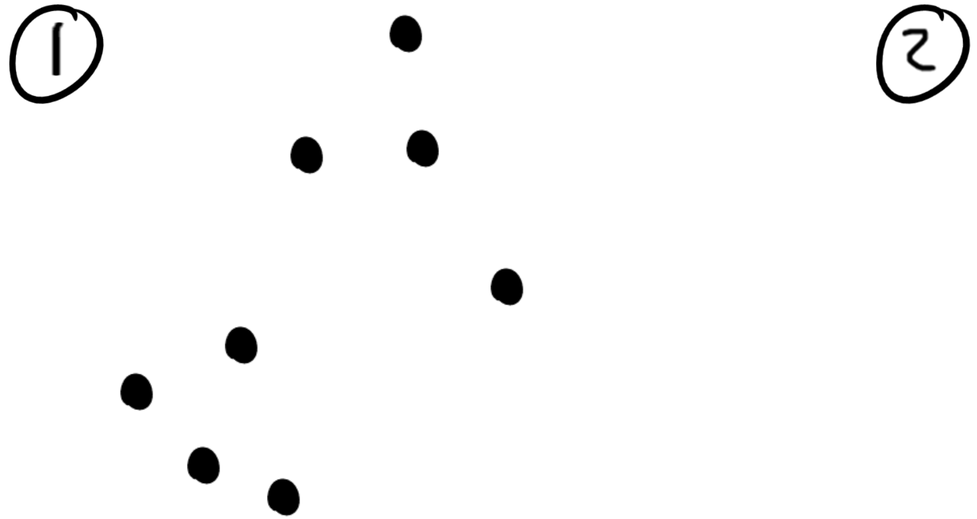
- Example: Gaussian Mixture
  - Given some data: $X = \{x_n\}_{n=1}^{N}$
  - Parameters: $\theta = \{\pi_k, \mu_k, \sigma_k\}$
  - Latent variables: $Z = \{z_{nk}\}_{n \leq N, k \leq K}$ with $z_{nk} = 1$ iff n belongs to cluster k

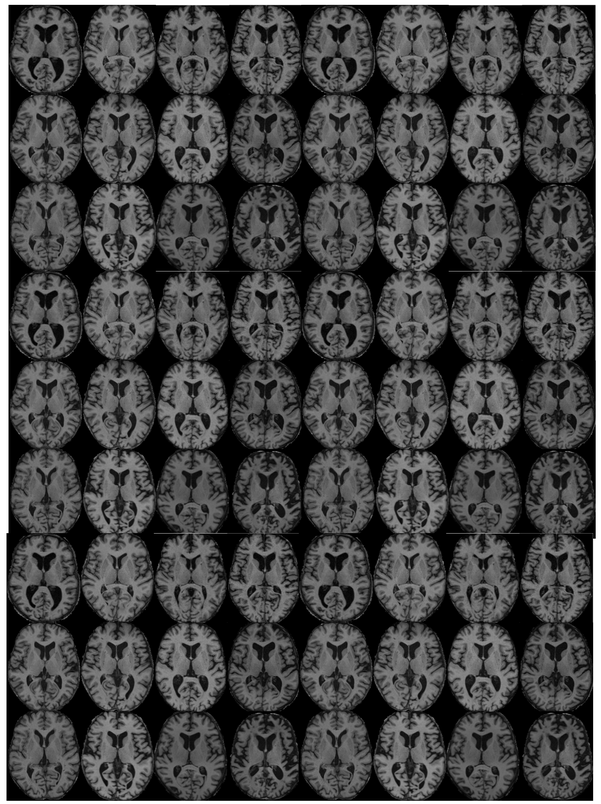
$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x | \mu_k, \sigma_k^2\right)$$

- EM algorithm alternates between
  - E-step: determine $p(Z|X, \theta)$
  - M-step: update θ
- EM greedily maximizes a lower-bound of the (marginal) likelihood $\mathcal{L}(\theta) = \log p(X|\theta)$

Choosing the number of clusters

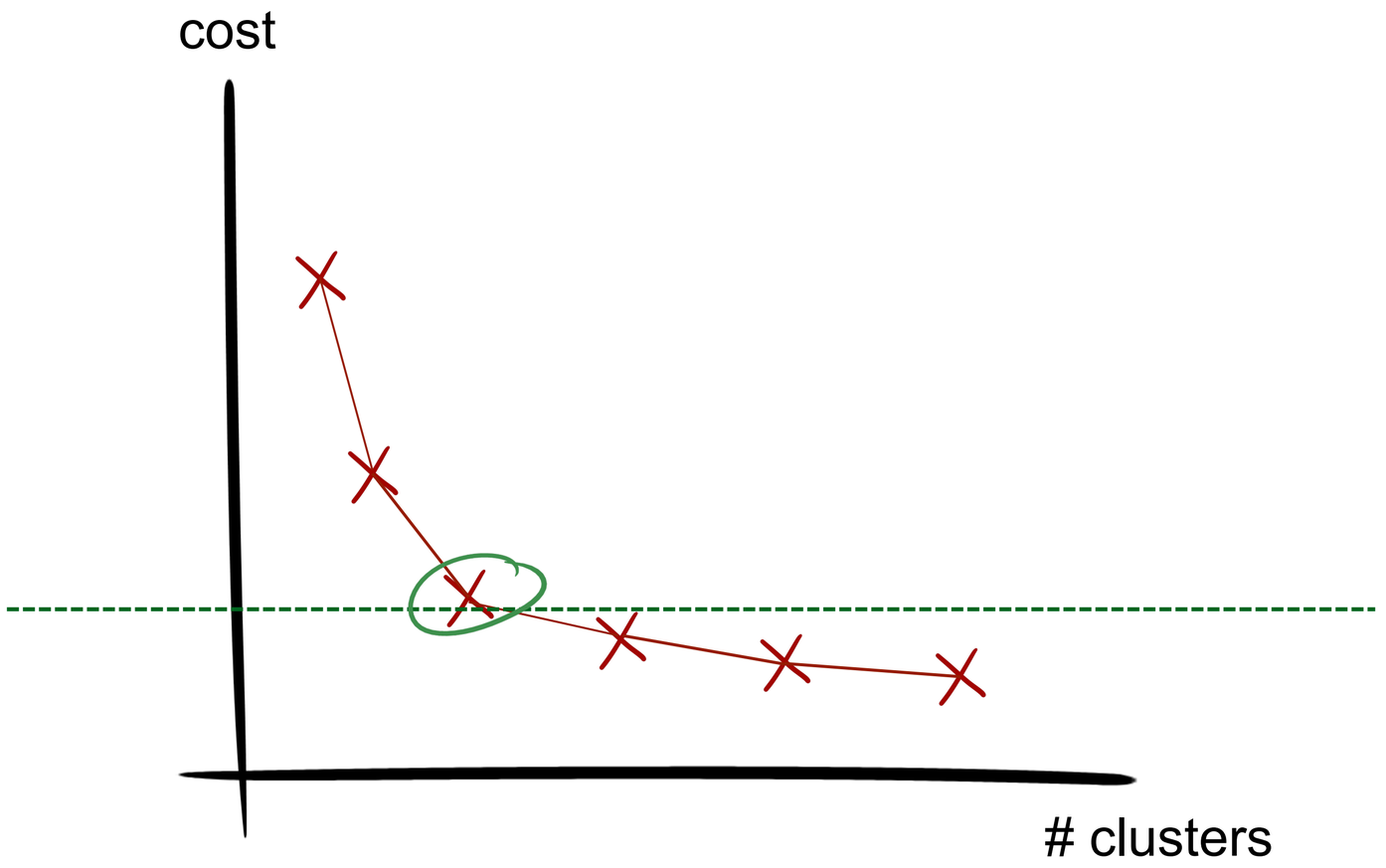
- How many clusters in the data?





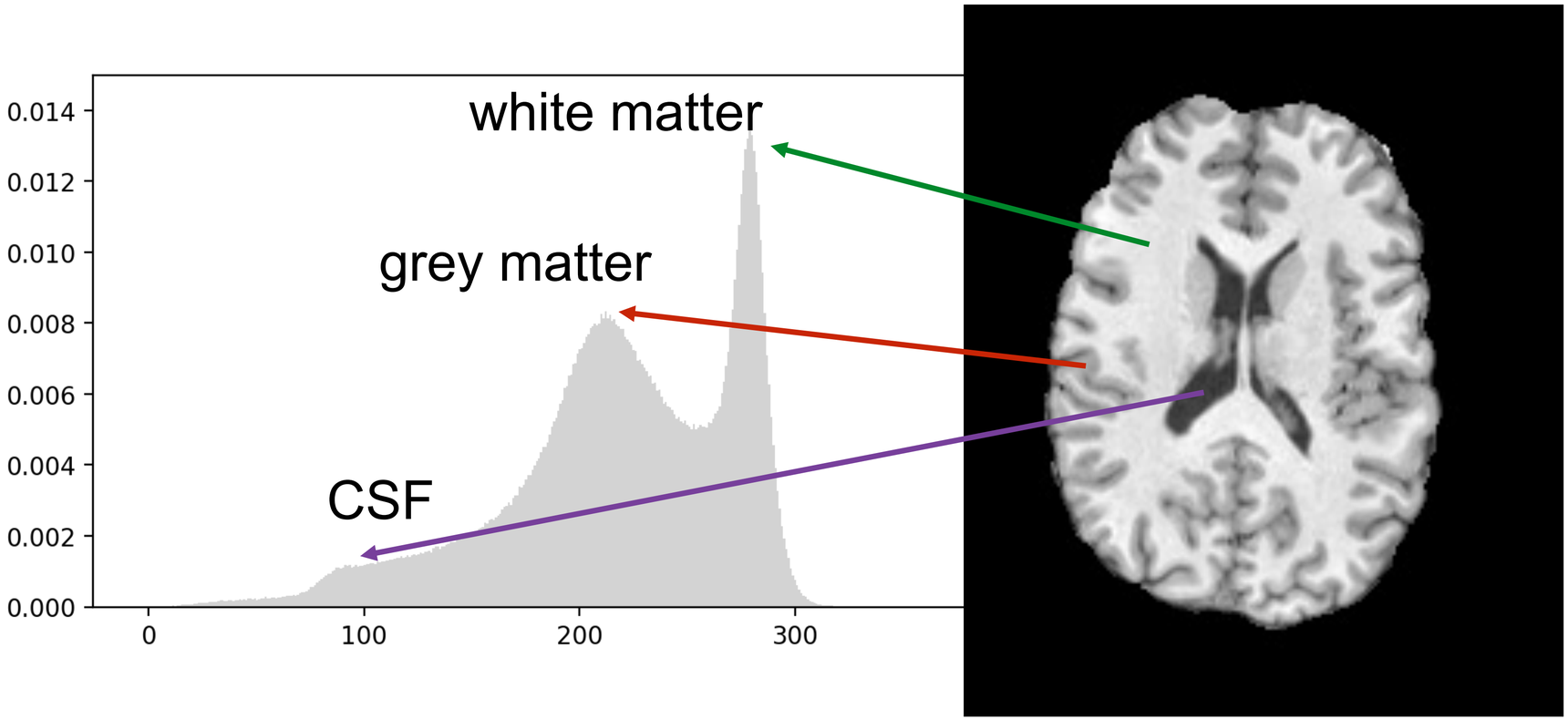
Number of clusters

1. “Elbow” method



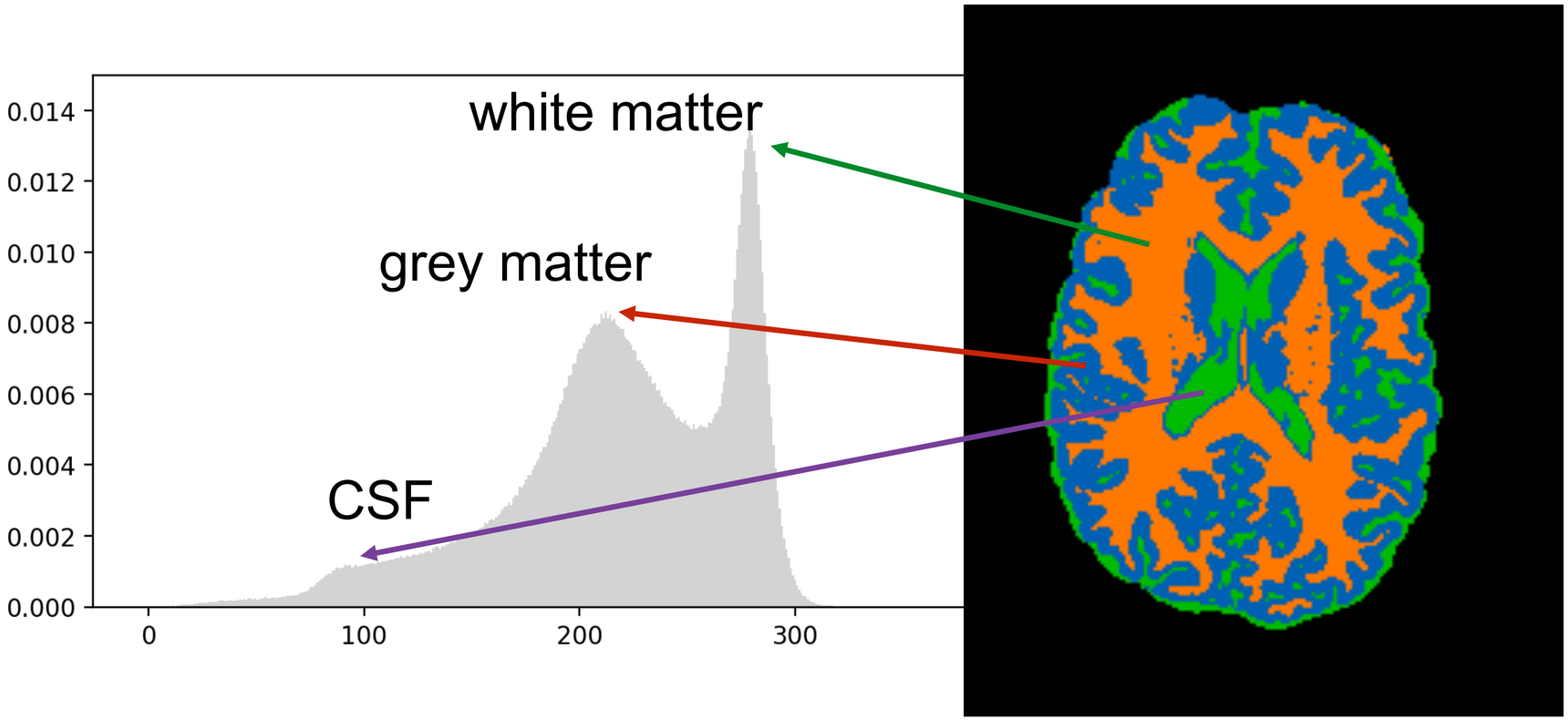
Number of clusters

1. “Elbow” method
2. Application-based rationale



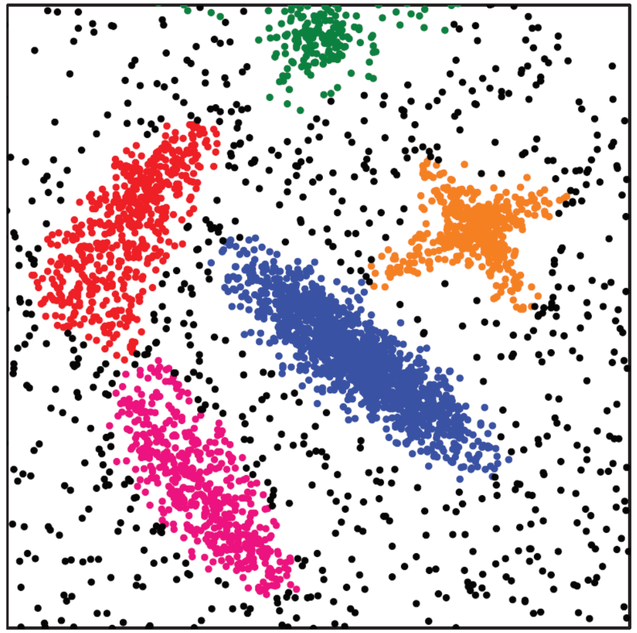
Number of clusters

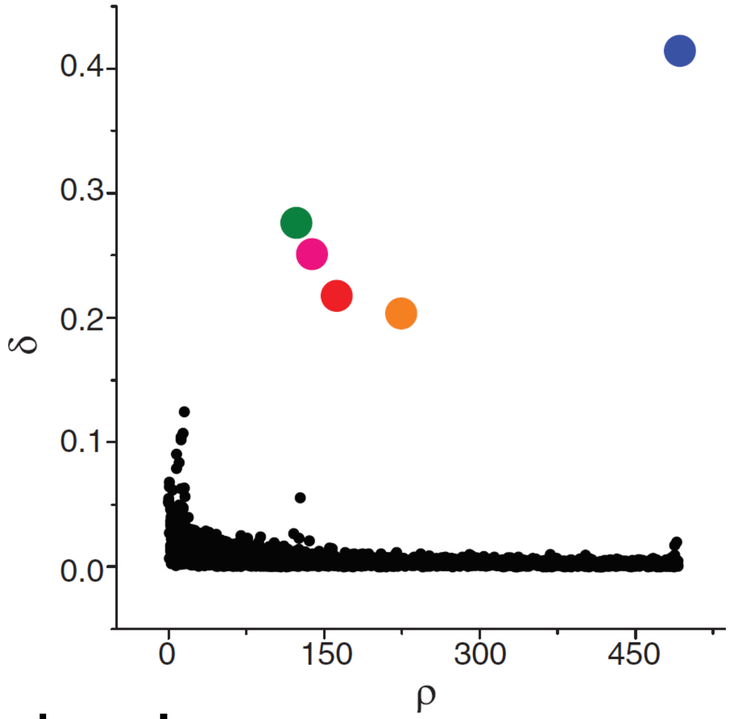
1. “Elbow” method
2. Application-based rationale



Number of clusters

1. “Elbow” method
2. Application-based rationale
3. And more



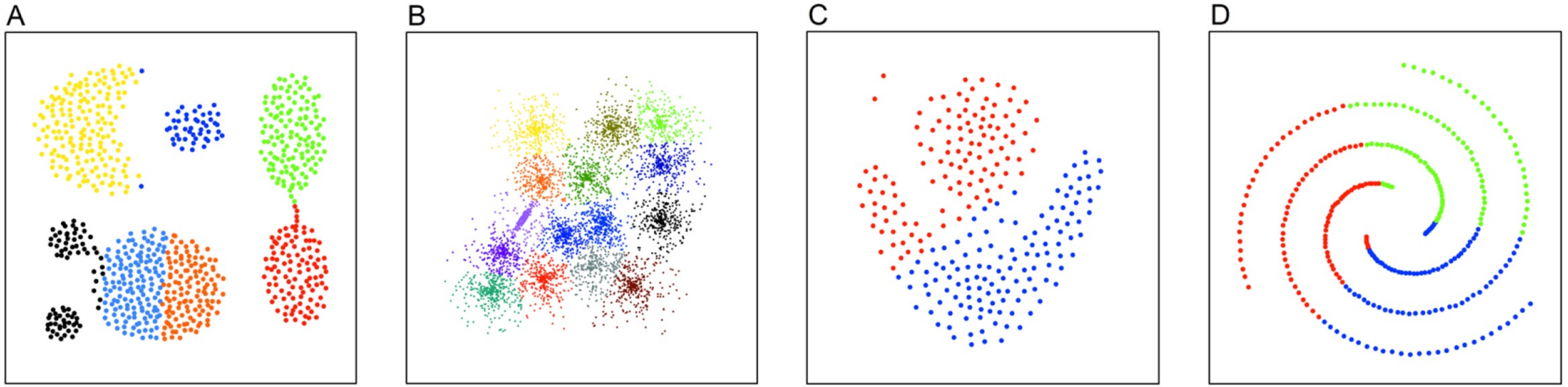


- Bayesian Mixture Models: don’t choose, use priors!
- Persistence diagrams / persistent homology

From Rodriguez et Laio, *Clustering by fast search and find of density peaks*, Science 2014

Data geometry: non-linearities

- K-Means handles anisotropic data and non-linearities poorly

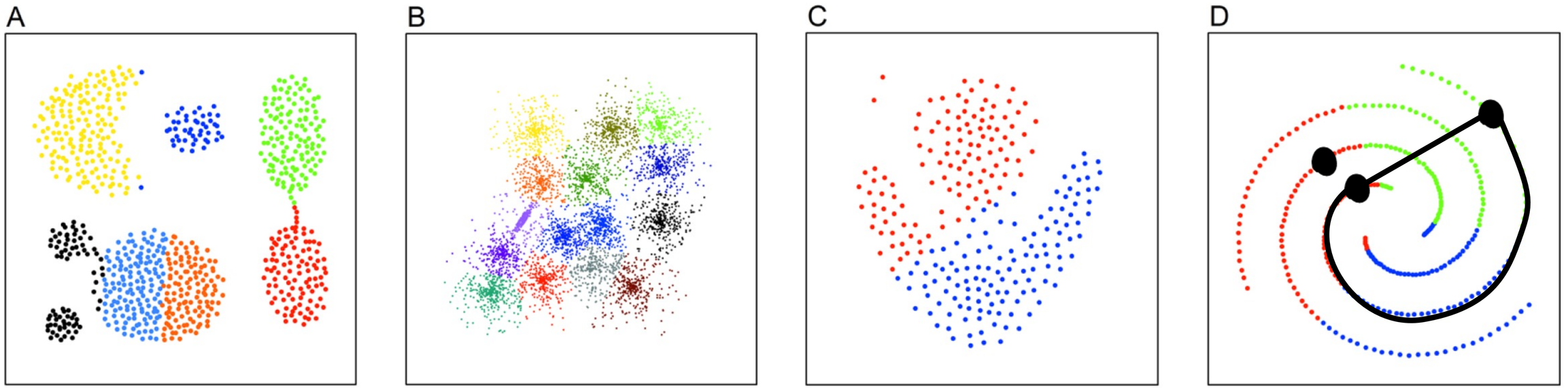


$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_n \sum_k z_{nk} \|x_n - \mu_k\|^2$$

From Rodriguez et Laio, *Clustering by fast search and find of density peaks*, Science 2014

Data geometry: non-linearities

- K-Means handles anisotropic data and non-linearities poorly



$$\min_{\{z_{nk}\},\{\mu_k\}} \sum_n \sum_k z_{nk} \|x_n - \mu_k\|^2$$

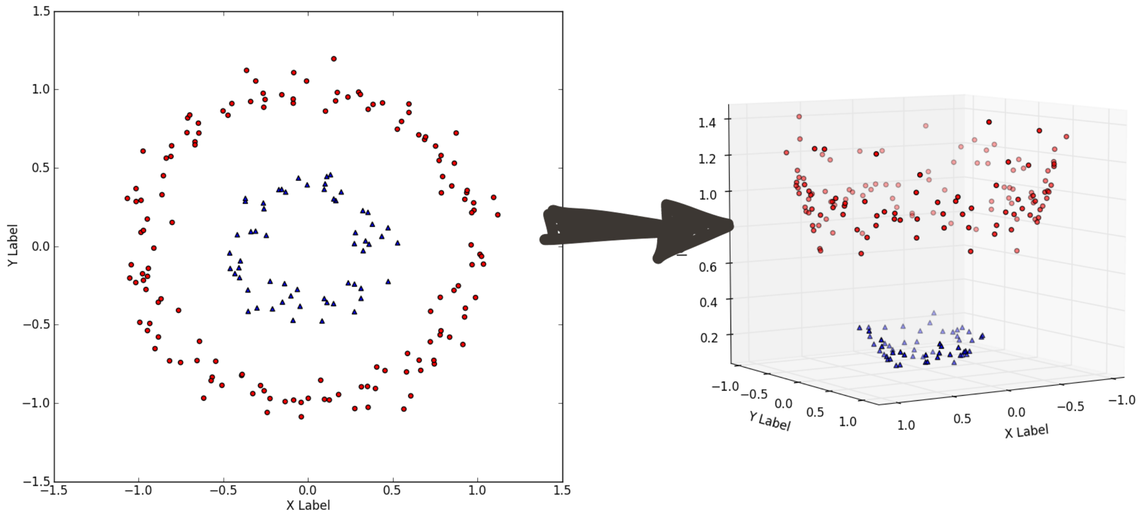
Euclidean distance

1. geodesic / graph-based

2. kernel-based

Data geometry is often complex

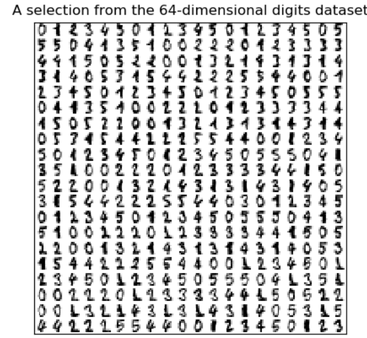
Idea 1. Lift data to a feature space where it can be described in linear terms

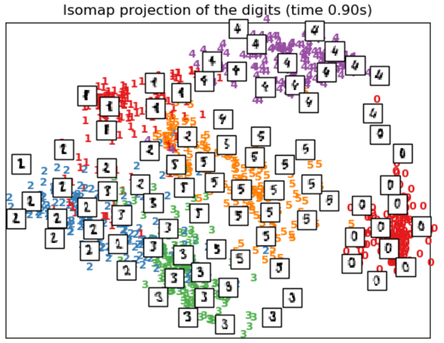


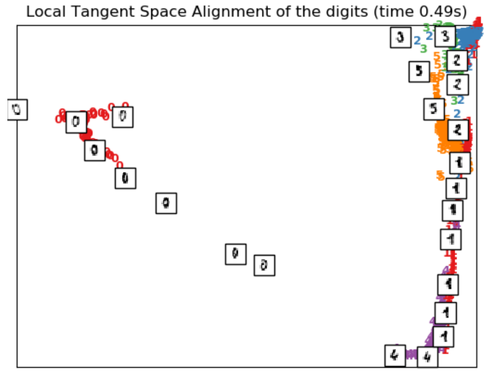
- Kernel methods
- e.g., support vector clustering, kernel k-Means (for clustering)
- Neural Networks

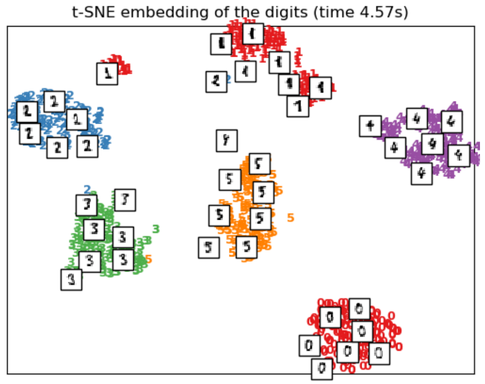
Idea 2. Geometry-inspired formulation, “manifold of data”

- “Manifold Learning”, “nonlinear dimensionality reduction” (e.g. MDS, Isomap, LTSA, t-SNE, UMAP)







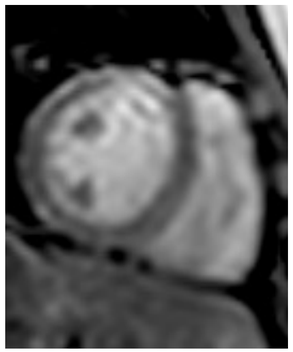


Well-suited to data visualisation

Dimensionality reduction / data encoding

Dimensionality reduction

- Real data, e.g. images, is often high-dimensional

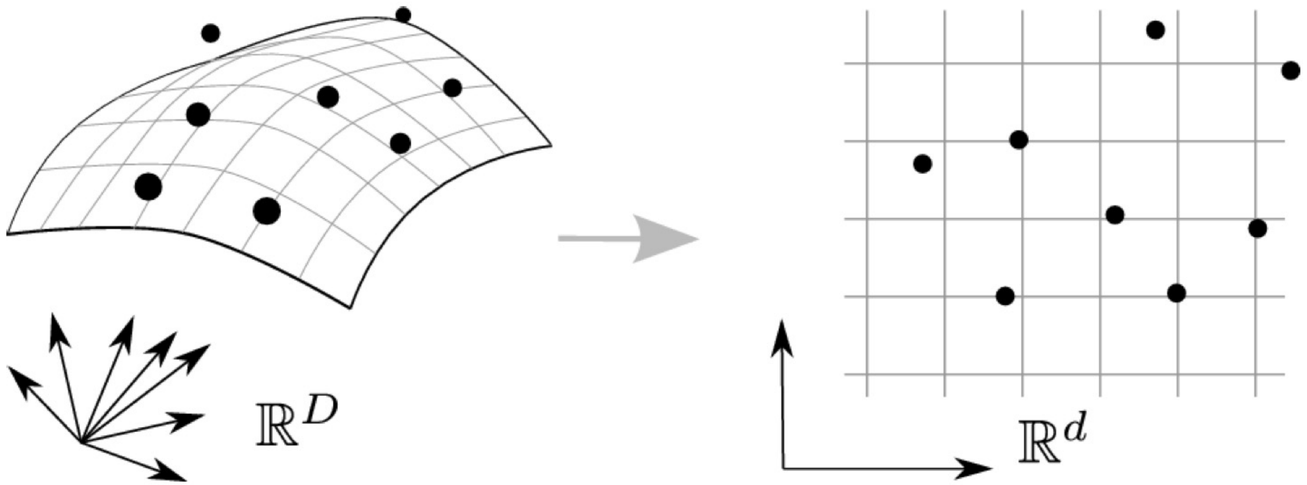


$$= \begin{pmatrix} 150 & 152 & \dots & 151 \\ 153 & 154 & \dots & 120 \\ \vdots & \vdots & \vdots & \vdots \\ 80 & 132 & 140 & 122 \end{pmatrix} \rightarrow \begin{pmatrix} 150 \\ 152 \\ \vdots \\ 151 \\ 153 \\ \vdots \\ 122 \end{pmatrix}$$

n x *m* matrix

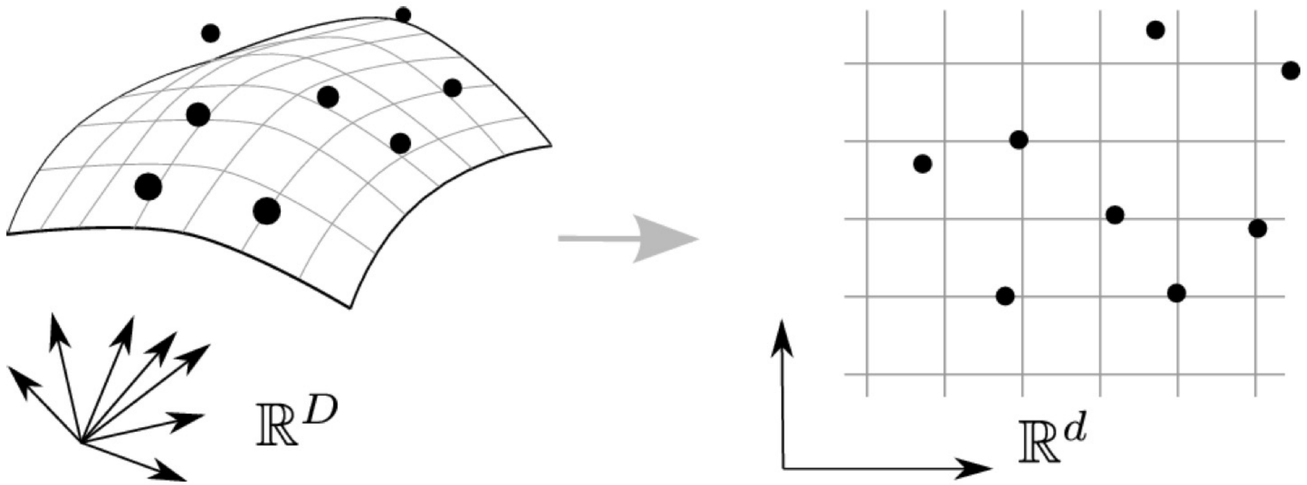
nm vector

Dimensionality reduction: Basic idea



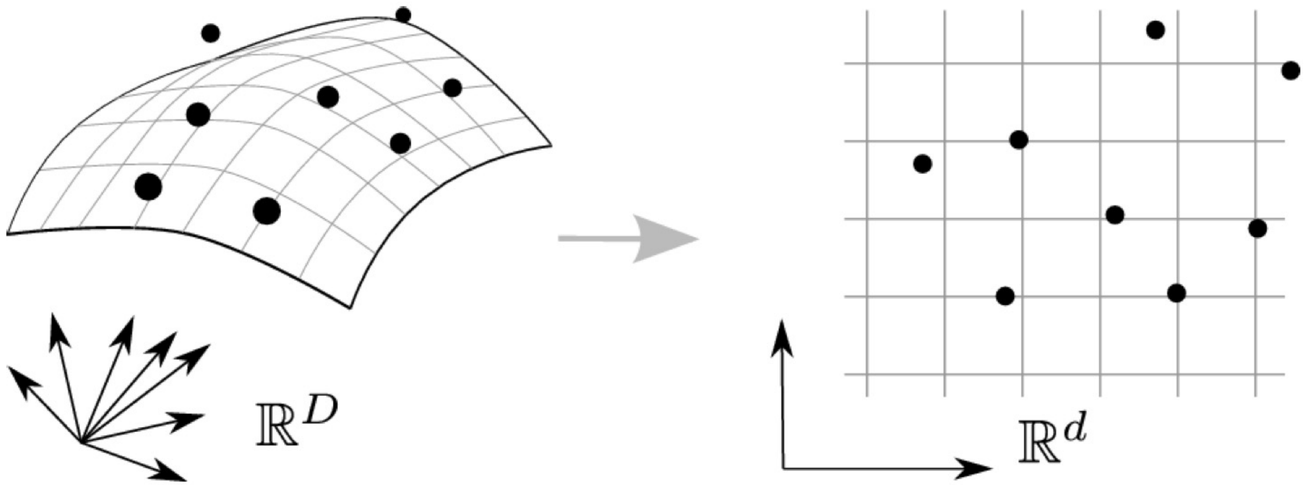
- High dimensional data set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
- Each $\mathbf{x}_j \in \mathbb{R}^D$ for large, possibly unknown, dimension D

Dimensionality reduction: Basic idea



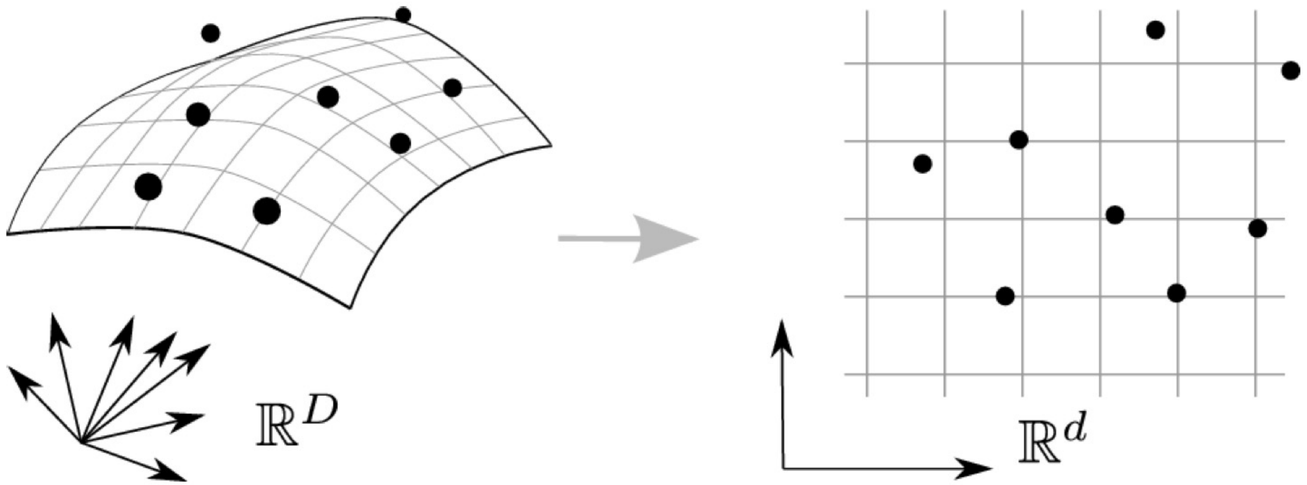
- Assume data lie on (or near) a manifold $\mathcal{M}$ embedded in $\mathbb{R}^D$ where
  - Intrinsic dimension of $\mathcal{M}$: $d \ll D$
  - Seek a low dimensional representation $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n\}$ for our data
  - The $\mathbf{y}_j$ provide a ‘chart’ for the manifold

Dimensionality reduction: Basic idea



- The input data can be $\mathbf{x}_j,\ 1 \leq j \leq n$
  - The original measurements
  - A set of measurements on pairs of data items $(\mathbf{x}_i, \mathbf{x}_j),\ \forall i, j$ similarities, distances, etc.

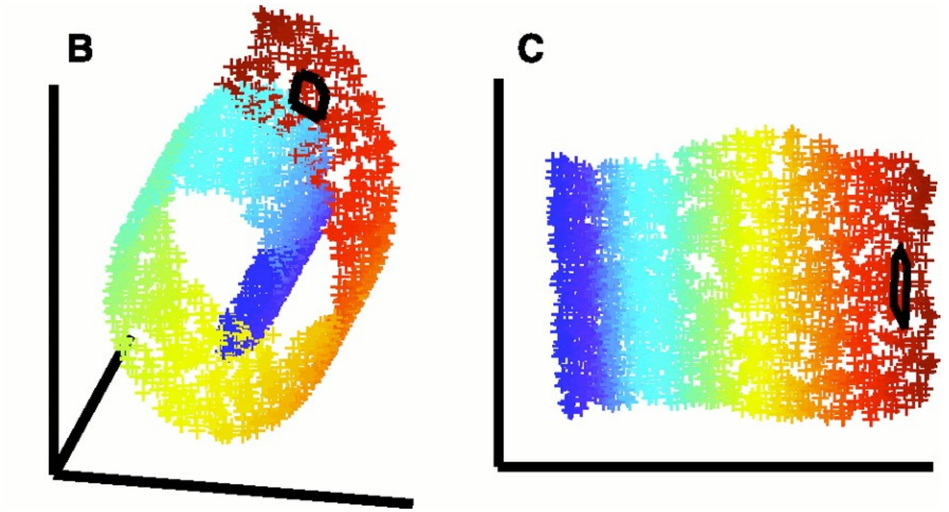
Dimensionality reduction: Basic idea



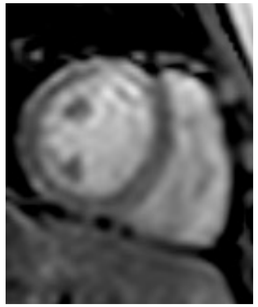
- The output data $\mathbf{y}_j$
  - Are low-dimensional and estimated using the objective function on the input data
  - Hopefully capture some useful or important aspects of our original data

Dimensionality reduction: Motivation

- Why do we need dimension reduction?
  - Dimension reduction aspect, often we may have unwieldy high dimensional data
  - Reduced dimension of data may be better suited to further analysis tasks such as clustering, classification or visualisation
  - Data may reside in some manifold which would like to discover
  - Measuring distances along the manifold is often more meaningful than measuring distances in the original space

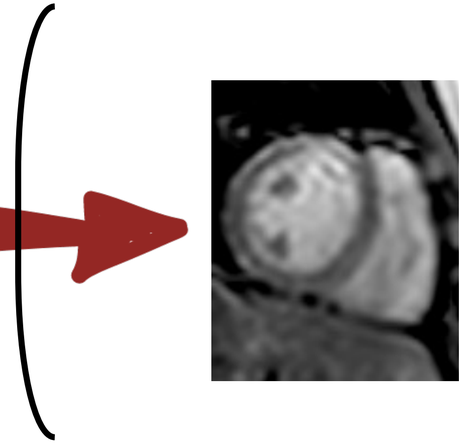


Dimensionality reduction



$m = 128 \times 128 = 16384$ pixels

$v \in \mathbb{R}^d$,
$d \ll 16386$



- Idea 1. Exploit redundancies in image
- Idea 2. Exploit redundancies in dataset

Applications

- Data visualisation
- Image Compression
- Modelling
- …

“MNIST” digit dataset

(a) Varying c_1 on InfoGAN (Digit type)

(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Figure adapted from Chen et al. InfoGAN (2016)

Principal Component Analysis (PCA) – Variant 1

Step 1. Build the design matrix

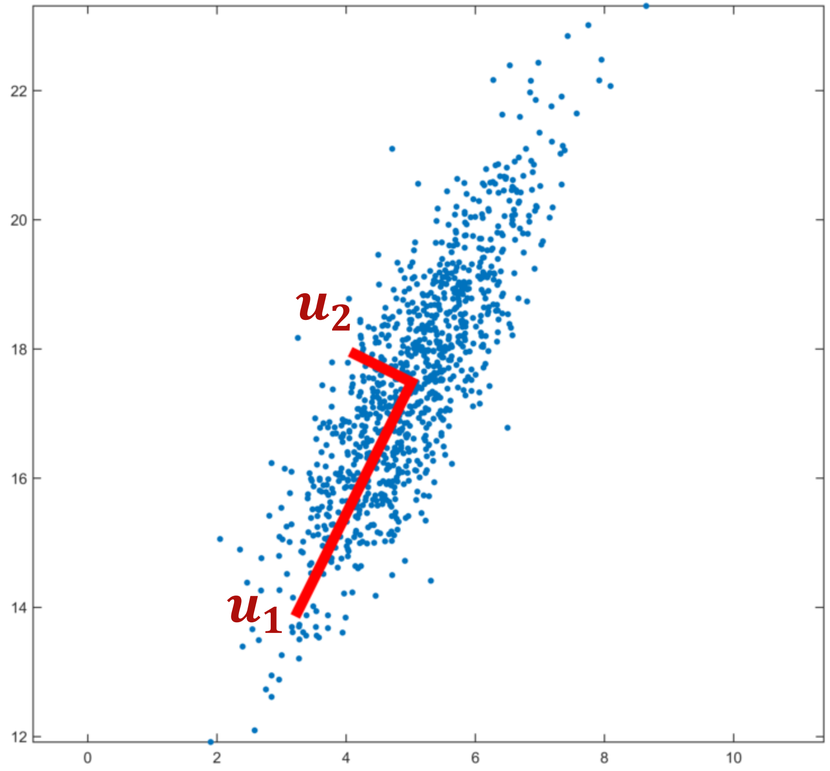
$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(n)} \\ \vdots & \ddots & \vdots \\ x_m^{(1)} & \cdots & x_m^{(n)} \end{bmatrix}$$

Step 2. Center the design matrix

$$X' = \frac{1}{\sqrt{n-1}}(X - \mu_X)$$

Step 3. Decomposition of the $m \times m$ covariance matrix

$$UDV^\top = SVD(X'X'^\top)$$



$U = (u_1 | \cdots | u_m)$: principal directions

$D \triangleq \mathrm{diag}(\lambda_i)$: variances along u_i's

Principal Component Analysis (PCA) – Variant 2

Step 1. Build the design matrix

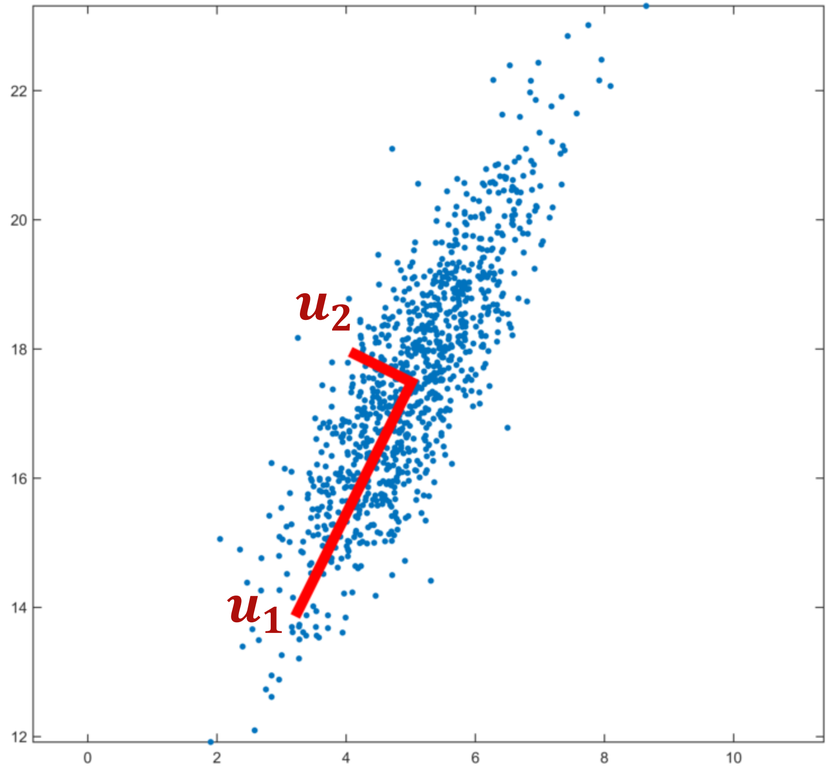
$$X = \begin{bmatrix} x_1^{(1)} & \cdots & x_1^{(n)} \\ \vdots & \ddots & \vdots \\ x_m^{(1)} & \cdots & x_m^{(n)} \end{bmatrix}$$

Step 2. Center the design matrix

$$X' = \frac{1}{\sqrt{n-1}}(X - \mu_X)$$

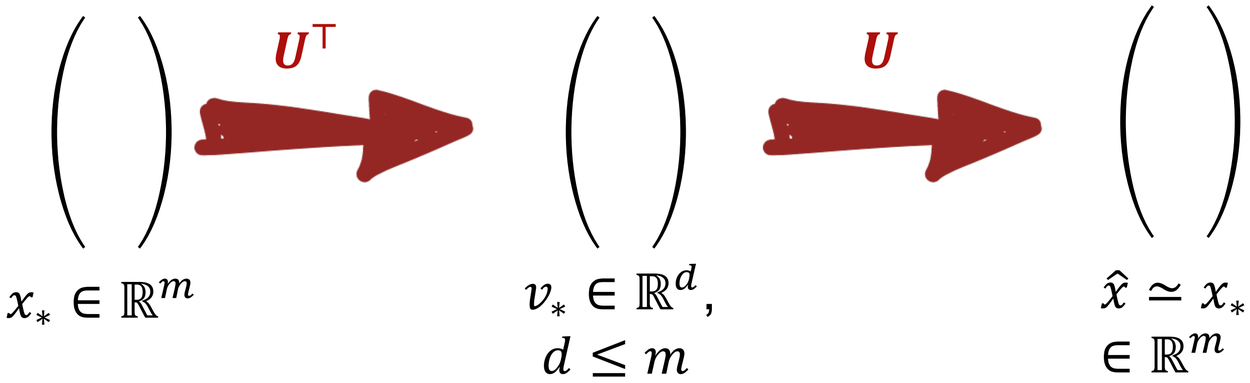
Step 3. Decomposition of the $m \times n$ design matrix

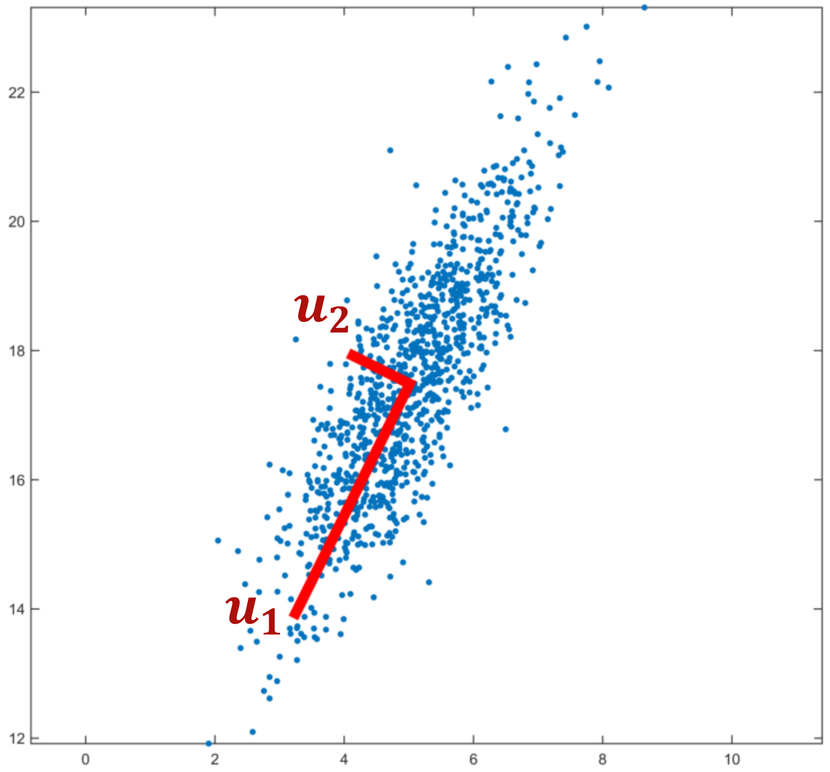
$$UDV^\top = SVD(X')$$



$U = (u_1 | \cdots | u_m)$: principal directions $\quad D \triangleq \mathrm{diag}(\lambda_i)$: std along u_i's

PCA for linear data encoding/decoding





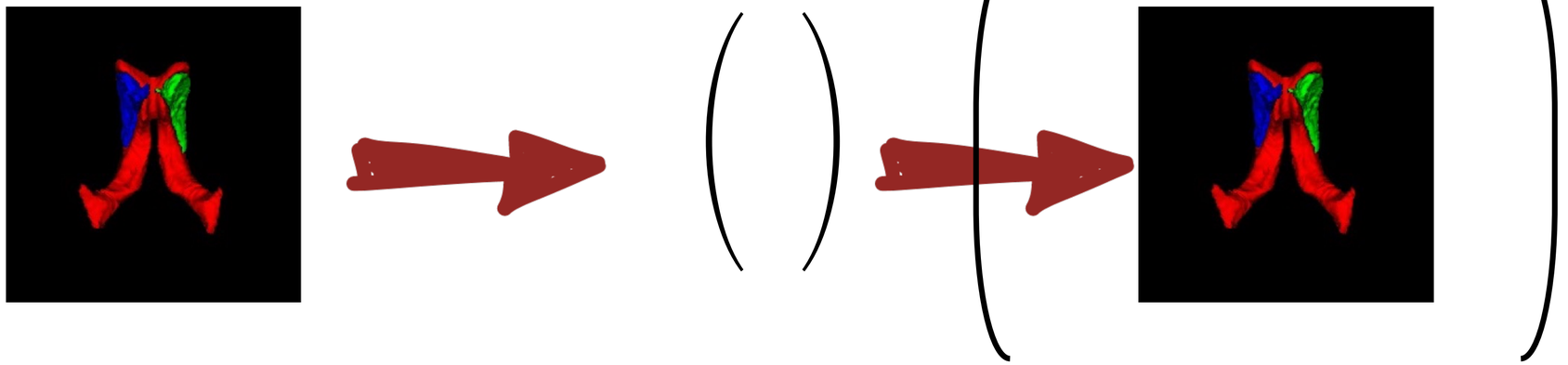
- Projection of x_* on principal directions:

$$v_* = U^\top x_*$$

- Reconstruction from the reduced coordinates v_*:

$$\hat{x} = U v_*$$

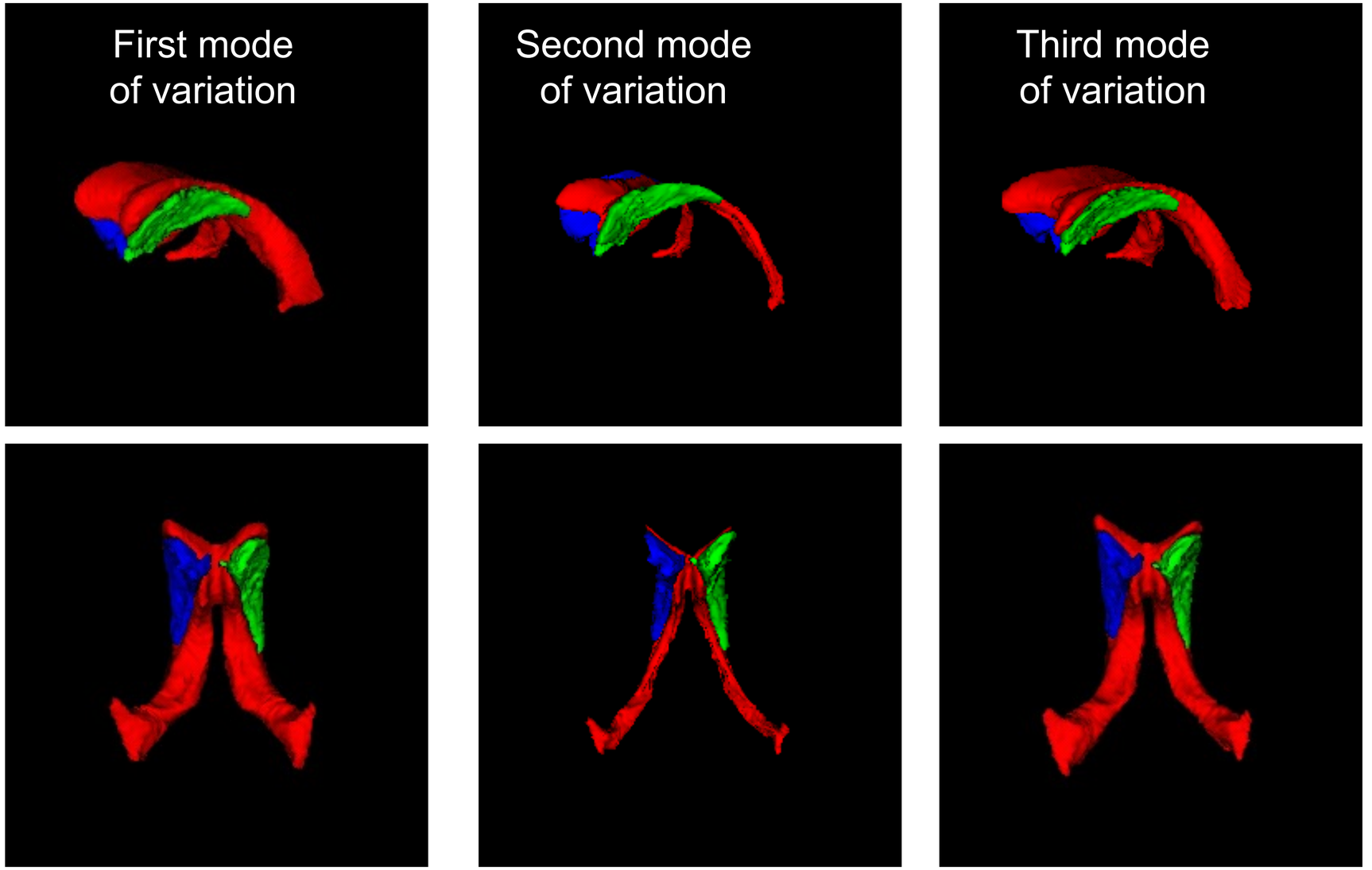
Dimensionality reduction



$m = no.\ of\ points \times 3$

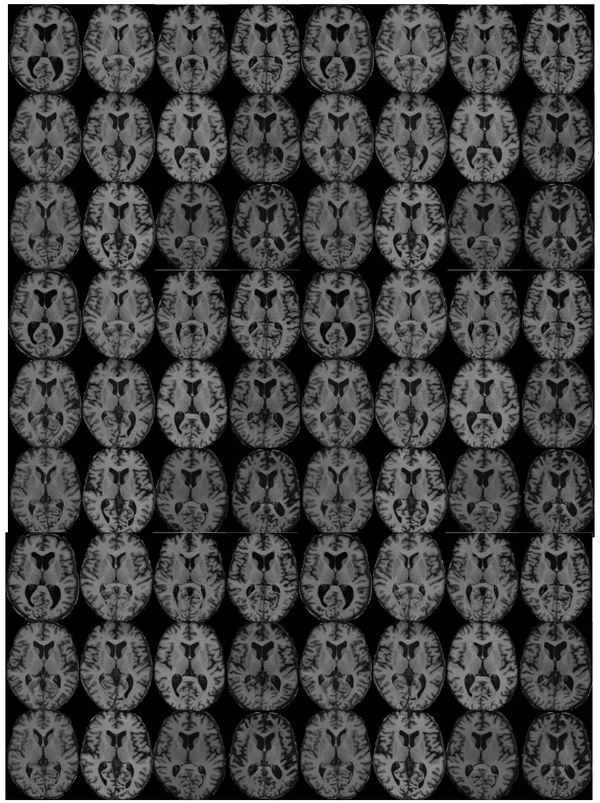
$v \in \mathbb{R}^d$

Example: PCA Shape Model



PCA image model?

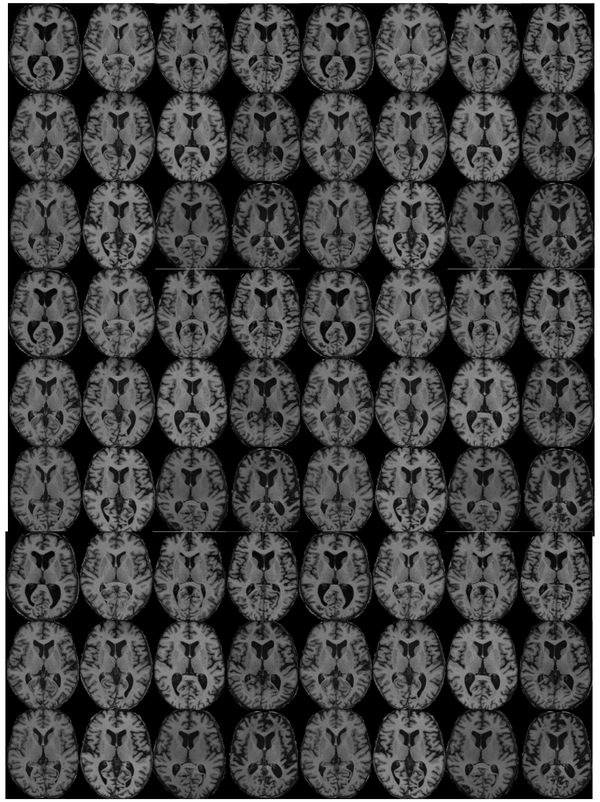
Apply PCA to “vectorized” images



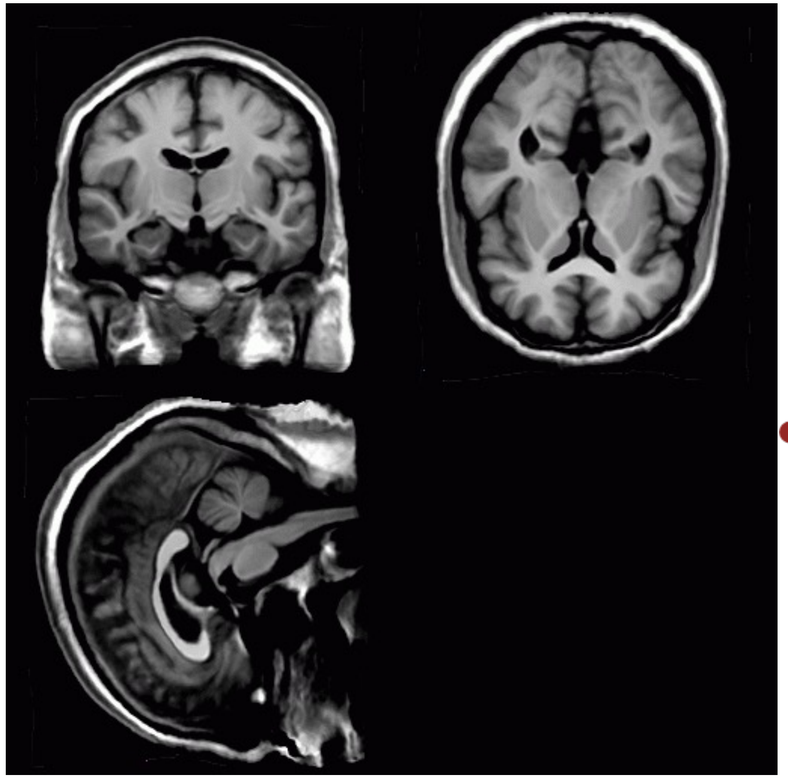
Dataset (subset)

PCA image model?

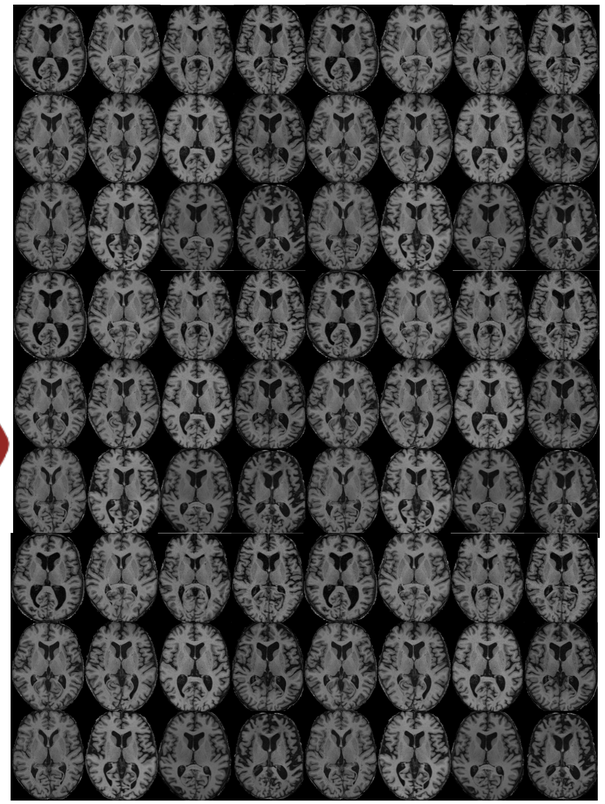
Apply PCA to “vectorized” images



Dataset (subset)



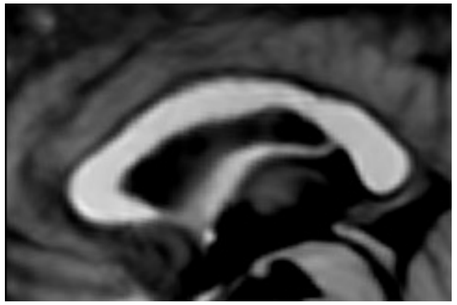
First eigendirections



PCA reconstruction

PCA image model?

First mode of variation



Second mode of variation

Third mode of variation

PCA Interpretations

1. kth eigenvector maximizes the explained variance in the subspace orthogonal to all previous eigenvectors

$$\text{e.g.,} \quad u_1 = \underset{\|u\|=1}{\operatorname{argmax}}\, u^T X'^T X' u$$

2. PCA fits an "ellipsoid" to the data, more formally a Gaussian distribution (e.g. Max. Likelihood estimate):

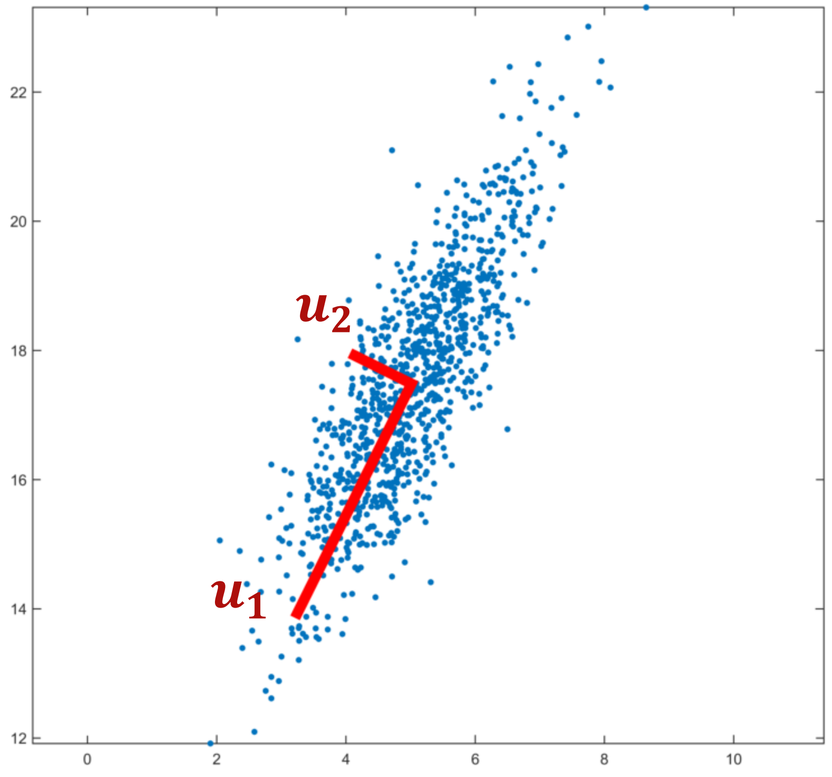
$$p(x;\mu,\Sigma) = |2\pi\Sigma|^{-1/2} \exp -\tfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$$

3. The first K PCA eigendirections yield optimal L_2 reconstruction among all linear models of rank $\leq K$

$$\underset{U}{\operatorname{argmin}} \sum_n \|x_n - \hat{x}_n(U;x_n)\|^2$$

4. PCA ↔ linear probabilistic model (cf. Factor analysis / probabilistic PCA)

$$x = Uw + \mu + \epsilon,$$
$$w \sim \mathcal{N}(0, I), \quad \Leftrightarrow \quad \mathrm{x} \sim \mathcal{N}(\mu, UU^T + \sigma^2 I)$$
$$\epsilon \text{ i.i.d.} \sim \mathcal{N}(0, \sigma^2 I)$$



PCA Interpretations

1. kth eigenvector maximizes the explained variance in the subspace orthogonal to all previous eigenvectors

$$\text{e.g.,} \quad u_1 = \underset{\|u\|=1}{\operatorname{argmax}}\, u^T X'^T X' u$$

2. PCA fits an "ellipsoid" to the data, more formally a Gaussian distribution (e.g. Max. Likelihood estimate):

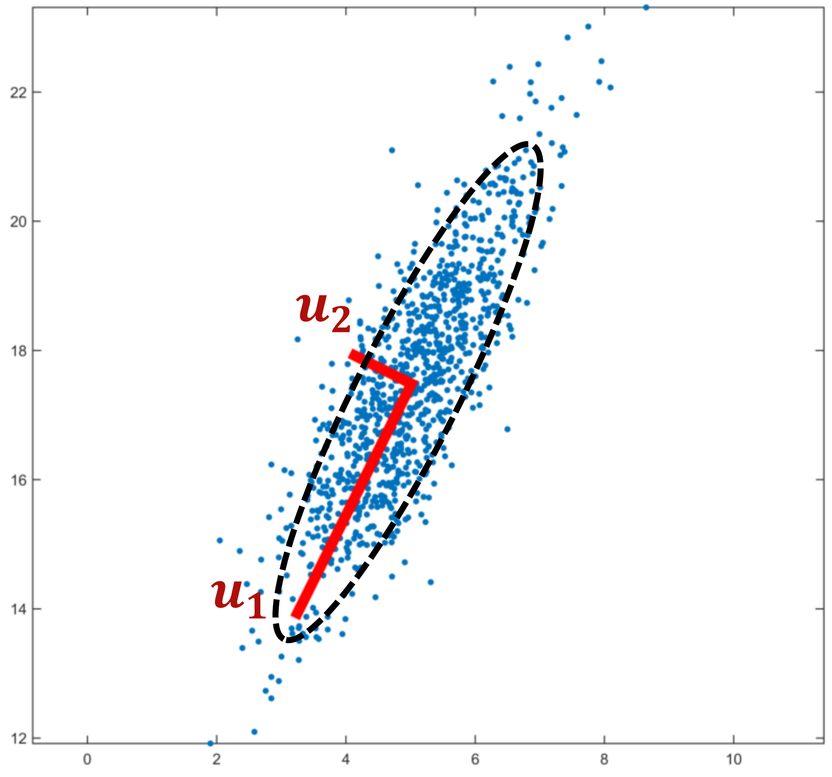
$$p(x;\mu,\Sigma) = |2\pi\Sigma|^{-1/2} \exp -\tfrac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$$

3. The first K PCA eigendirections yield optimal L_2 reconstruction among all linear models of rank $\leq K$

$$\underset{U}{\operatorname{argmin}} \sum_n \|x_n - \hat{x}_n(U;x_n)\|^2$$

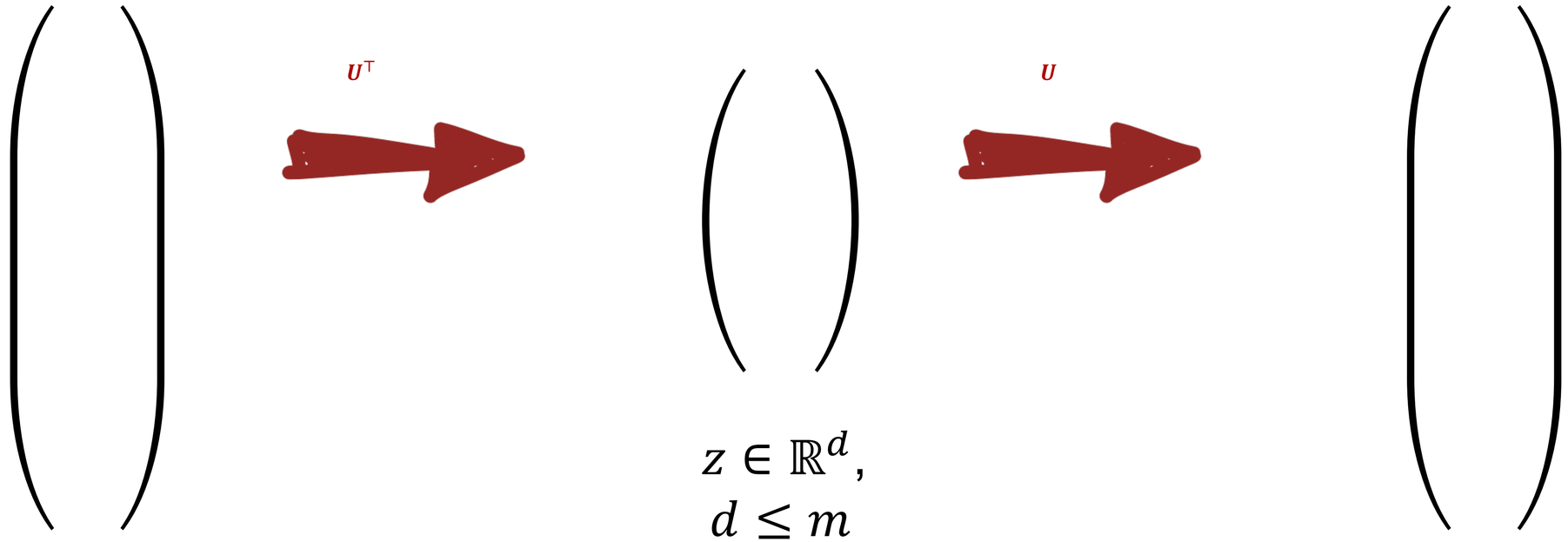
4. PCA ↔ linear probabilistic model (cf. Factor analysis / probabilistic PCA)

$$x = Uw + \mu + \epsilon,$$
$$w \sim \mathcal{N}(0, I), \quad \Leftrightarrow \quad \mathrm{x} \sim \mathcal{N}(\mu, UU^T + \sigma^2 I)$$
$$\epsilon \text{ i.i.d.} \sim \mathcal{N}(0, \sigma^2 I)$$



Autoencoders

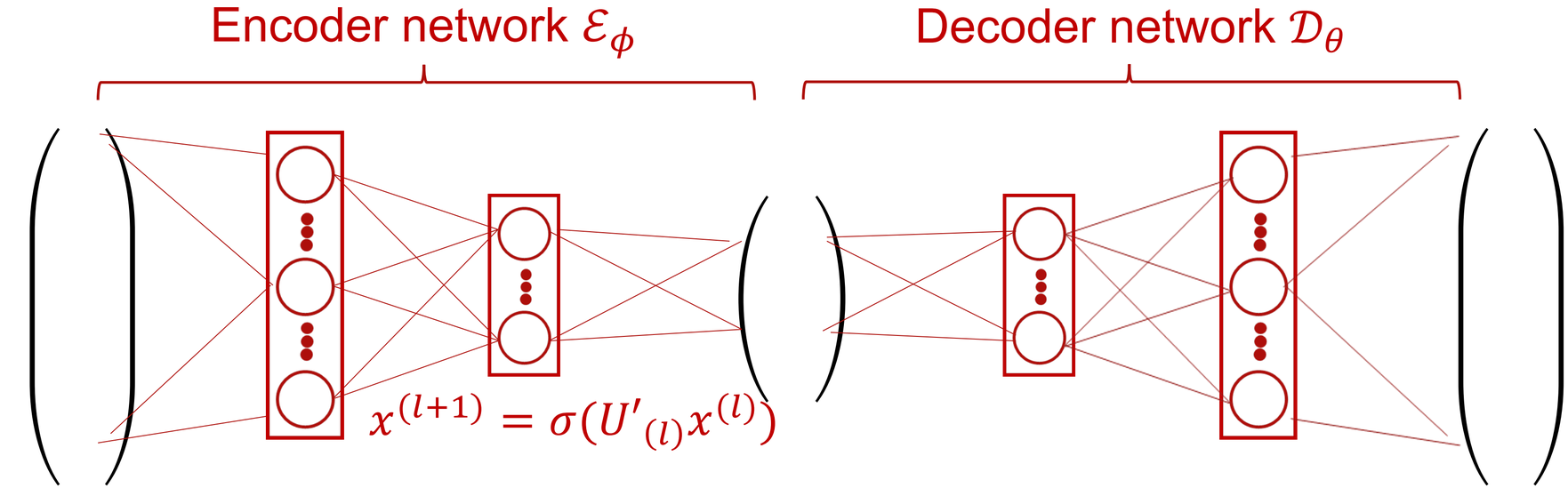
- PCA is a linear encoder/decoder



Reconstruction loss $\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2, \qquad \hat{x} = U(U^T x)$

Autoencoders

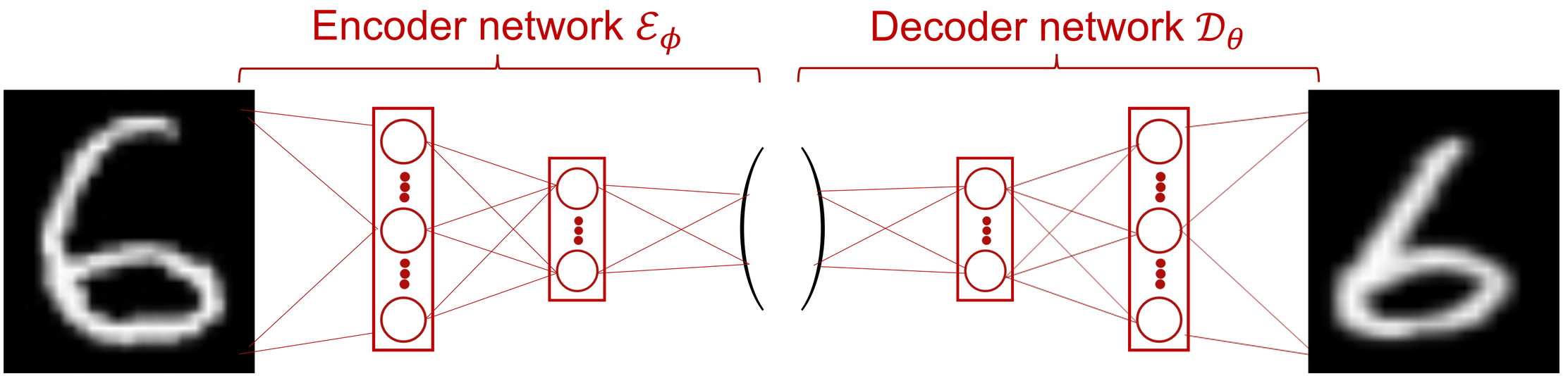
- Auto-Encoders are general encoder/decoder architectures with non-linearities



(not always, e.g. sparse autoencoders)

Reconstruction loss $\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2, \qquad \hat{x} = \mathcal{D}_\theta(\mathcal{E}_\phi(x))$

Autoencoders for Imaging



I
$\in \mathbb{R}^{H \times W}$

$z \in \mathbb{R}^d,$
$d \leq H \times \mathrm{W}$

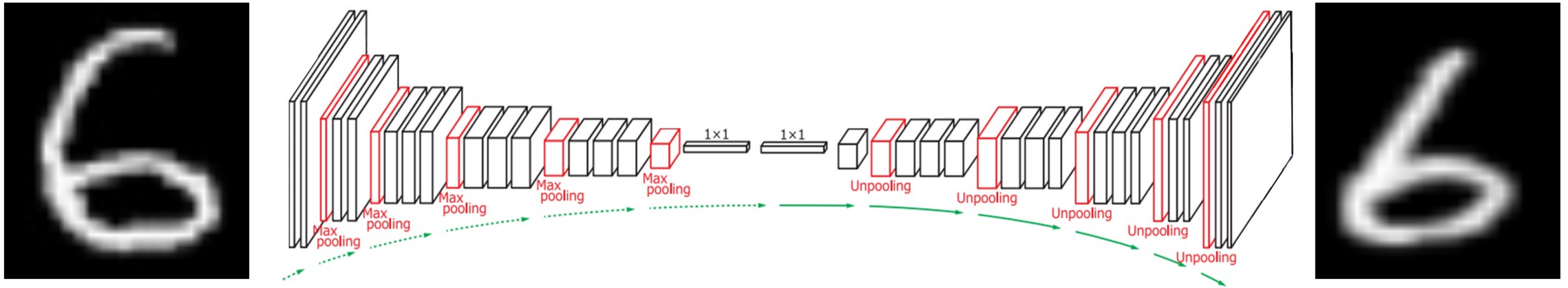
$\hat{I} \simeq I$
$\in \mathbb{R}^{H \times W}$

Reconstruction loss $\mathcal{L}(I, \hat{I}) = \left\| I - \hat{I} \right\|^2,$ $\hat{I} = \mathcal{D}_\theta(\mathcal{E}_\phi(I))$

Autoencoders for Imaging

- Many possible architectures (e.g. Convolutional Autoencoder)

Encoder network $\mathcal{E}_\phi$ Decoder network $\mathcal{D}_\theta$



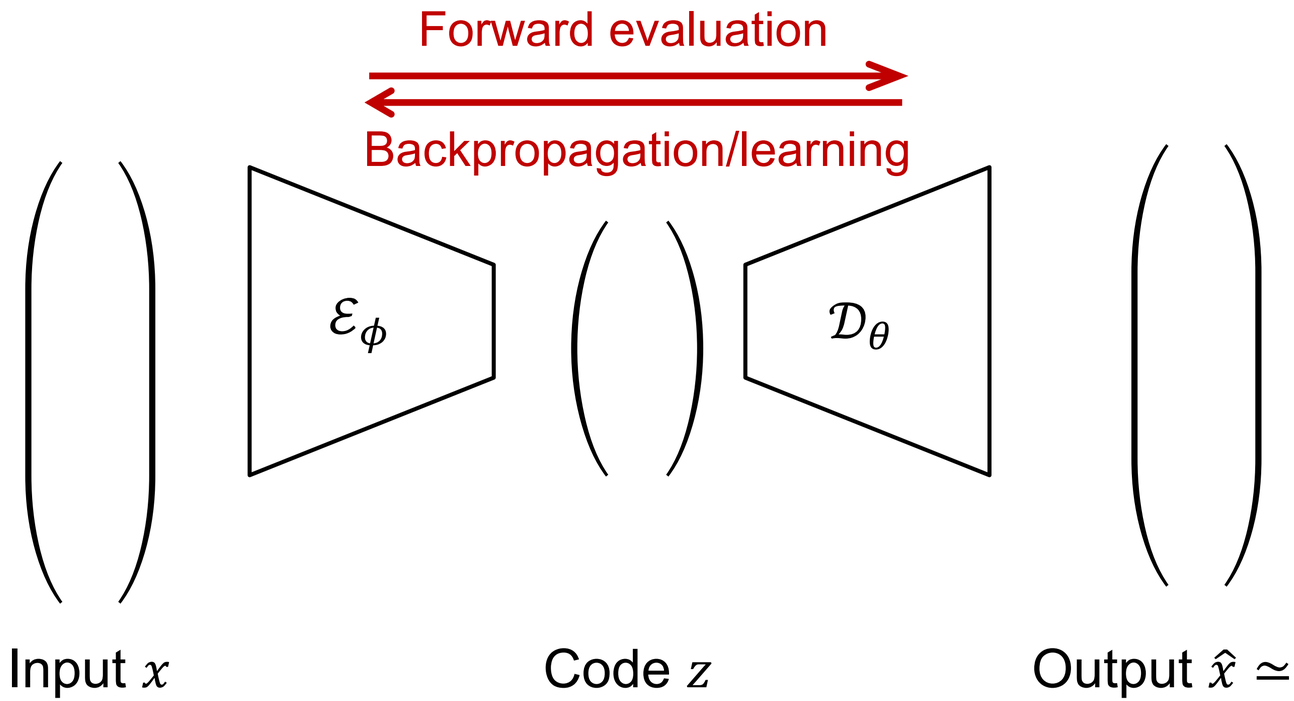
$I \in \mathbb{R}^{H \times W}$

$z \in \mathbb{R}^d$,
$d \leq H \times \mathrm{W}$

$\hat{I} \simeq I \in \mathbb{R}^{H \times W}$

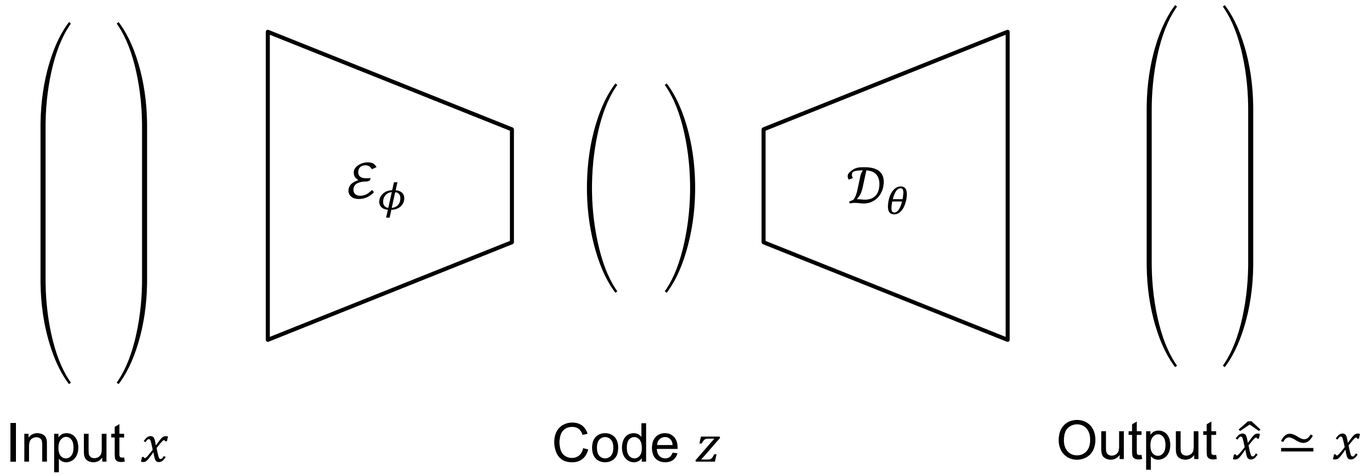
Reconstruction loss $\mathcal{L}(I, \hat{I}) = \left\| I - \hat{I} \right\|^2, \quad \hat{I} = \mathcal{D}_\theta(\mathcal{E}_\phi(I))$

Autoencoders 101



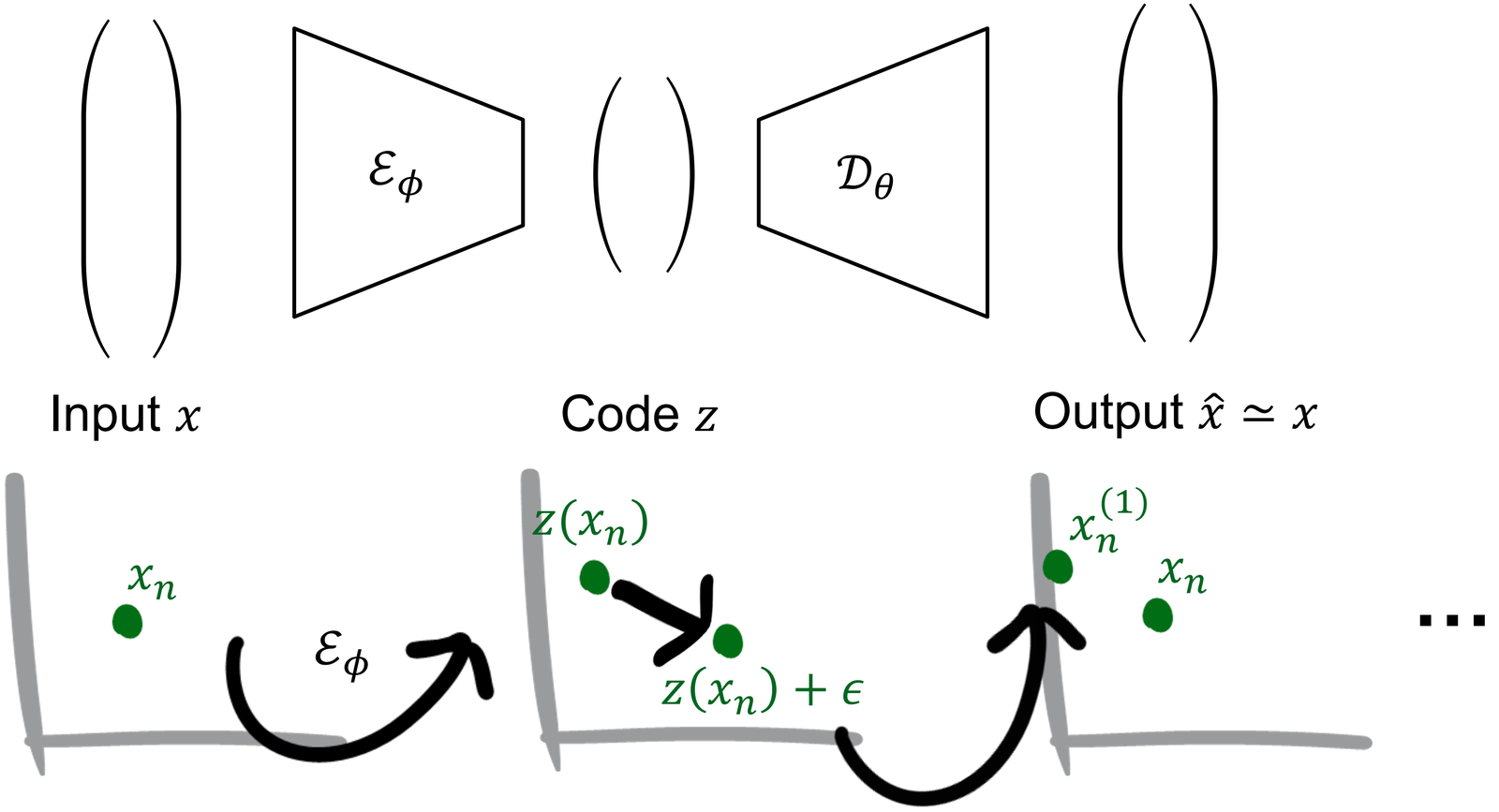
- Reconstruction loss $\mathcal{L}_{\phi,\theta}(x, \hat{x}) = \|x - \hat{x}(\phi, \theta)\|^2$
- How do autoencoders not just learn the identity map?
  - Bottleneck in the code: $\dim(z) \ll \dim(x)$
  - Add regularization: $\mathcal{L}_{\phi,\theta} + \lambda \cdot \mathcal{R}(\phi, \theta)$ e.g. Regularized AE, Sparse AE, Contractive AE
  - By training on corrupted input $\tilde{x}$ (Denoising AE):
    ex. Gaussian noise, “dropout” (randomly set p% network activations to 0)

Autoencoders: data visualization, generative modelling?



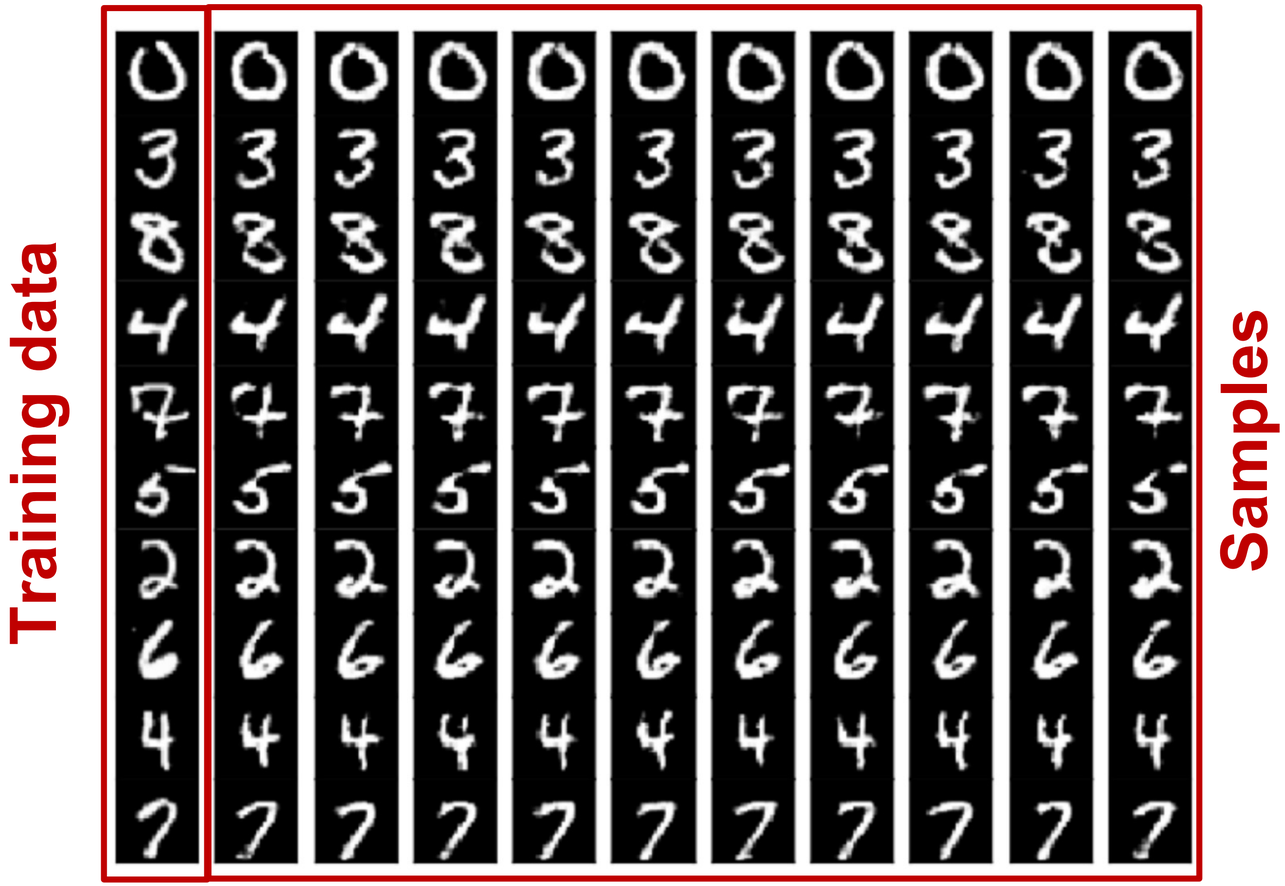
- Unlike PCA, non trivial to:
  - visualize data in the latent space of Autoencoders
  - generate new samples

Autoencoders: data visualization, generative modelling?



- Unlike PCA, non trivial to:
  - visualize data in the latent space of Autoencoders
  - generate new samples: perturb latent code of training example and iterate

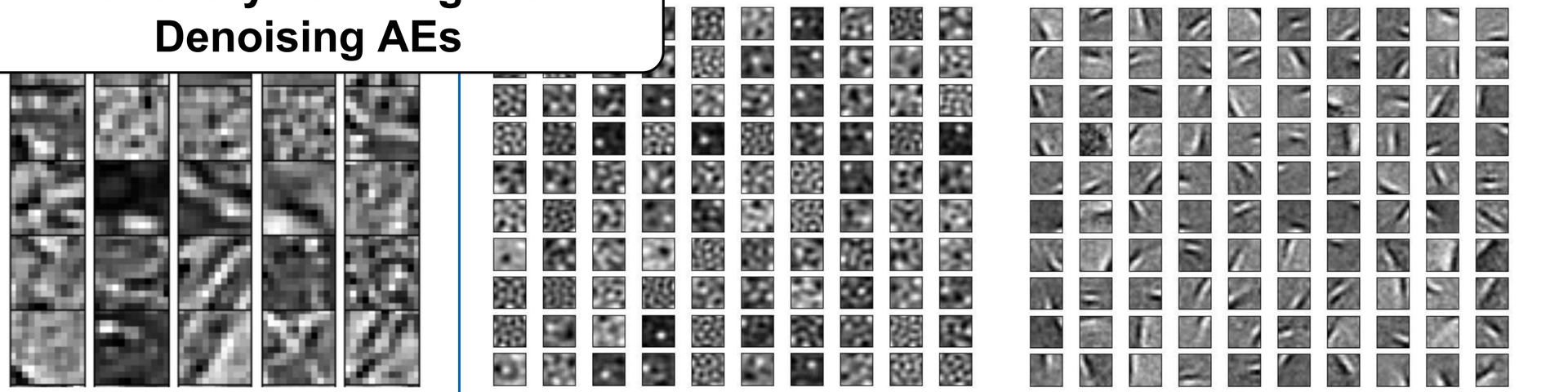
Autoencoders: data visualization, generative modelling?



- Unlike PCA, non trivial to:
  - visualize data in the latent space of Autoencoders
  - generate new samples: perturb latent code of training example and iterate

Autoencoders: Applications

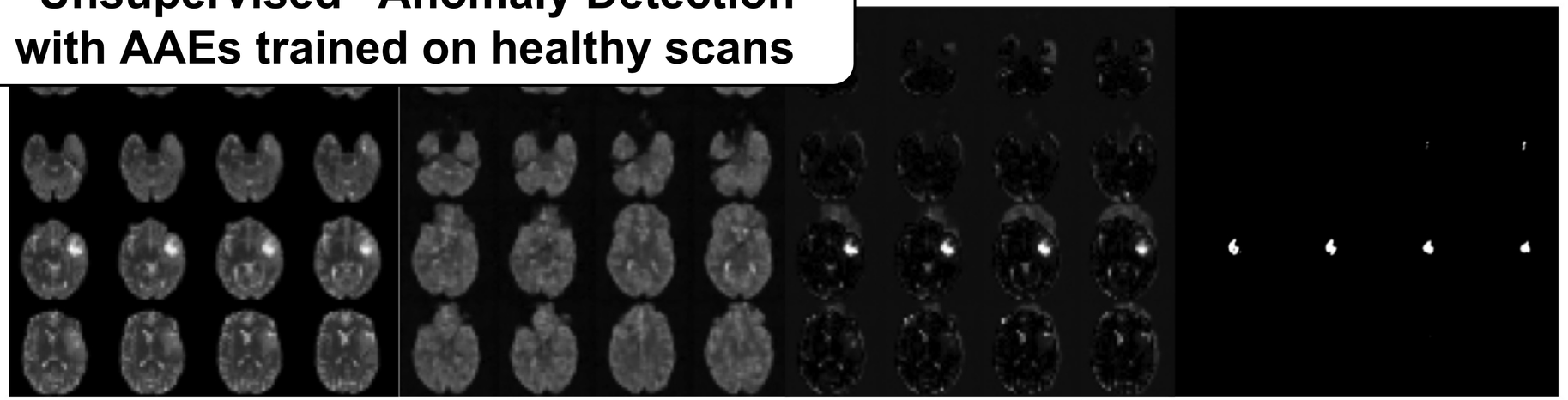
Dictionary Learning with Denoising AEs



Natural Image Patches | Regularized AEs | Denoising AEs (Gaussian noise)

Vincent et al. Stacked Denoising Autoencoders, JMLR 2010

“Unsupervised” Anomaly Detection with AAEs trained on healthy scans



Brain MRI images | AE reconstruction | Difference $I - \hat{I}$ | Expert lesion mask

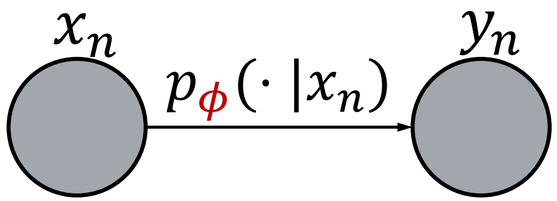
Chen et Konukoglu. MIDL 2018

Generative Modelling

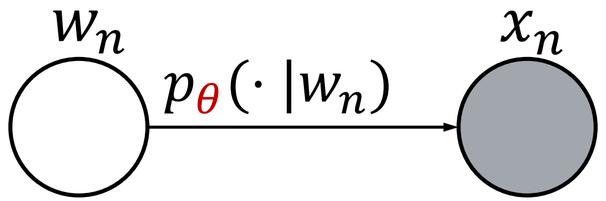
Generative Modelling

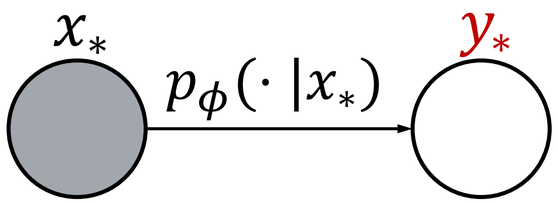
- Model the process through which data is generated
  – Capture the structure in the data e.g., dependencies, causal links…
- Mostly subsumes other tasks: clustering, dimensionality reduction…
- Introduce latent (≙ unobserved, unknown) parameters if necessary. Since they are unknown, assign probability distributions to them
- Bayesian framework allows to
  – Model probabilistically, manipulate probabilistic variables (= distributions)
  – make optimal predictions in the presence of unknown

Supervised vs. Unsupervised

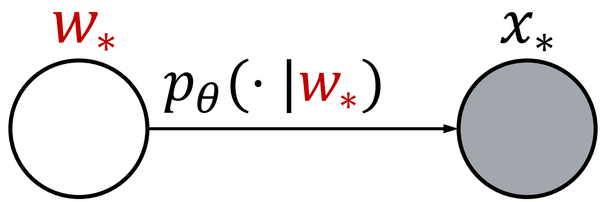


Training time





Test time



For encoding or clustering

Inputs

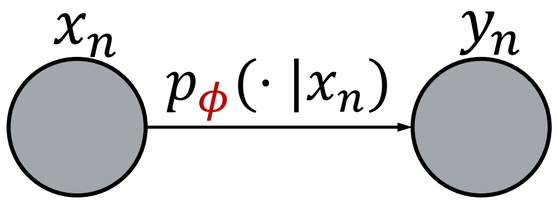
Dependent variable / label

Latent code

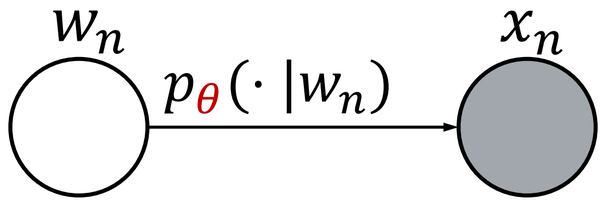
Inputs

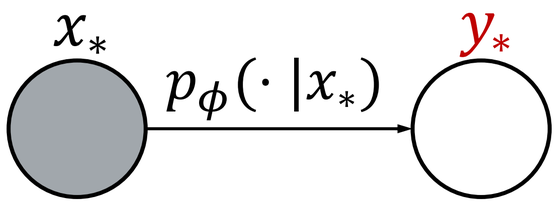
- In red, what we want to know
- Similar but for unsupervised training, neither latent codes nor θ known!

Supervised vs. Unsupervised

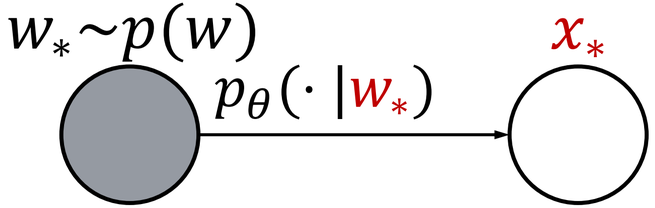


Training time





Test time



To generate new realistic samples

Inputs

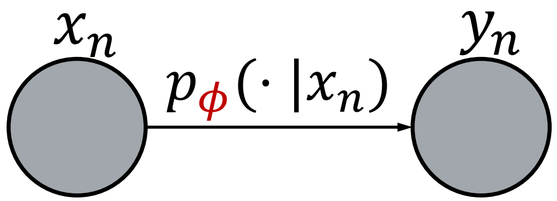
Dependent variable / label

Latent code

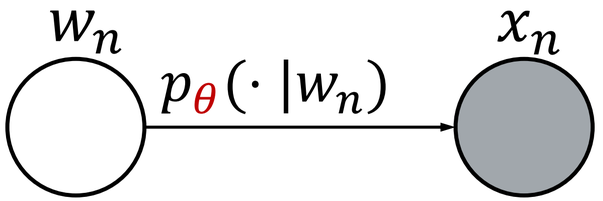
Inputs

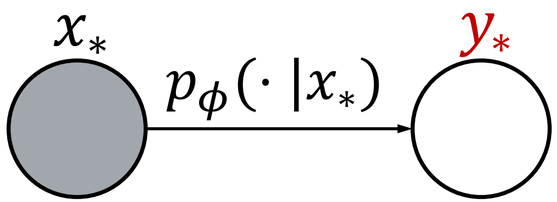
- In red, what we want to know
- Similar but for unsupervised training, neither latent codes nor θ known!

Supervised vs. Unsupervised

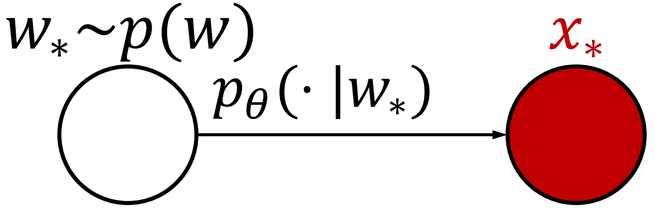


Training time





Test time



For outlier detection
$p_\theta(x_*) = ?$

Inputs

Dependent variable / label

Latent code

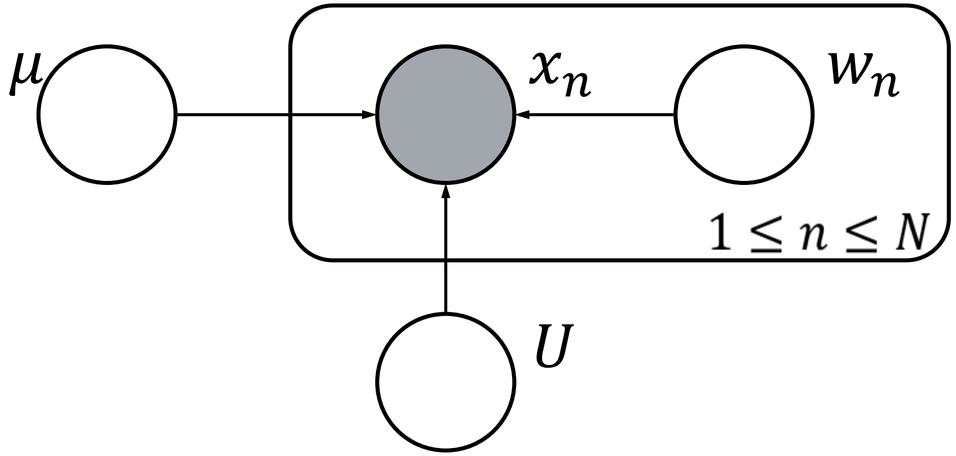
Inputs

- In red, what we want to know
- Similar but for unsupervised training, neither latent codes nor θ known!

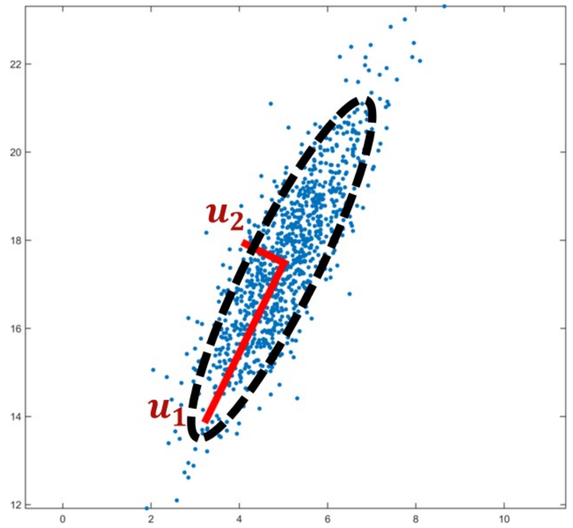
Example 1: PCA as a linear generative model

Latent code: $w_n \sim \mathcal{N}(0, I),$

Linear decoder: $x_n = Uw_n + \mu,$



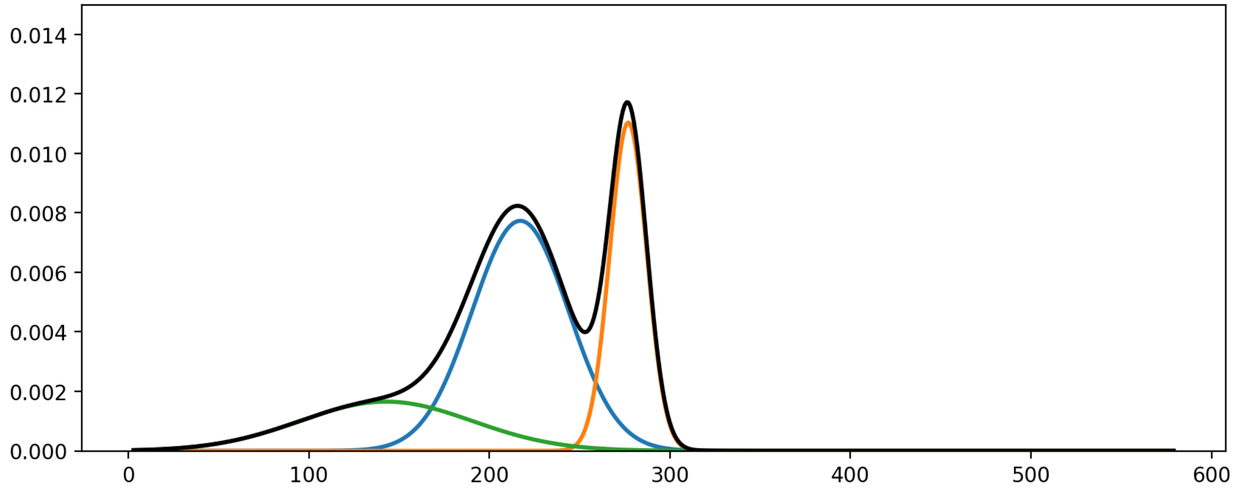
$$p_\theta(x_n) = \mathcal{N}(x_n|\mu, \Sigma)$$
$$\Sigma = UU^t$$



Example 2: GMM generative model (1D data)

Latent code: $w_n \sim \mathcal{N}(0,1)$,
$c_n \in \{1 \cdots K\} \sim \mathcal{C}(\{\pi_k\}_{k \leq K})$

Decoder: $x_n = \sigma_{c_n} w_n + \mu_{c_n}$

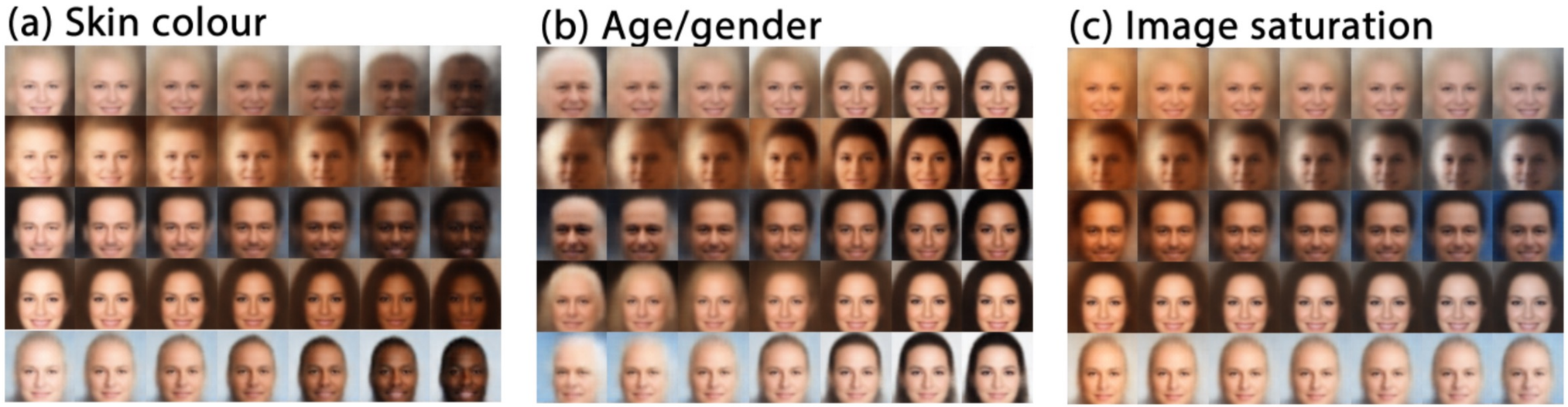


$$p_\theta(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x | \mu_k, \sigma_k^2\right)$$

$$\theta = \{\pi_k, \mu_k, \sigma_k\}_{k \leq K}$$

‘Black-box’ Generative Modelling

- Use generic architectures (NNs)
- Automatically capture the structure / learn variation in the data



From Higgins et al., beta-VAE (ICLR 2017)

‘Black-box’ Generative Modelling

Face interpolation

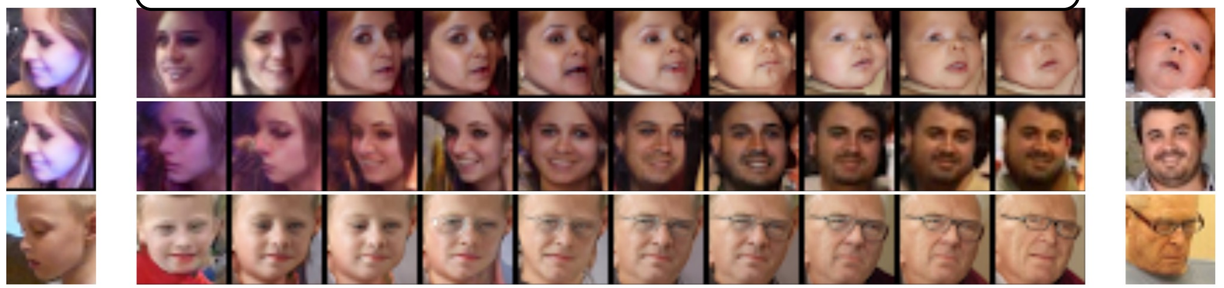


Figure 5: Linear interpolations in the embedding space decoded by the PixelCNN. Embeddings from leftmost and rightmost images are used for endpoints of the interpolation.

From van den Oord et al. *PixelCNN* (NIPS 2016)

Shape interpolation / arithmetic

Figure 6: Intra/inter-class interpolation between object vectors

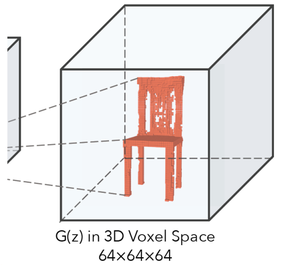
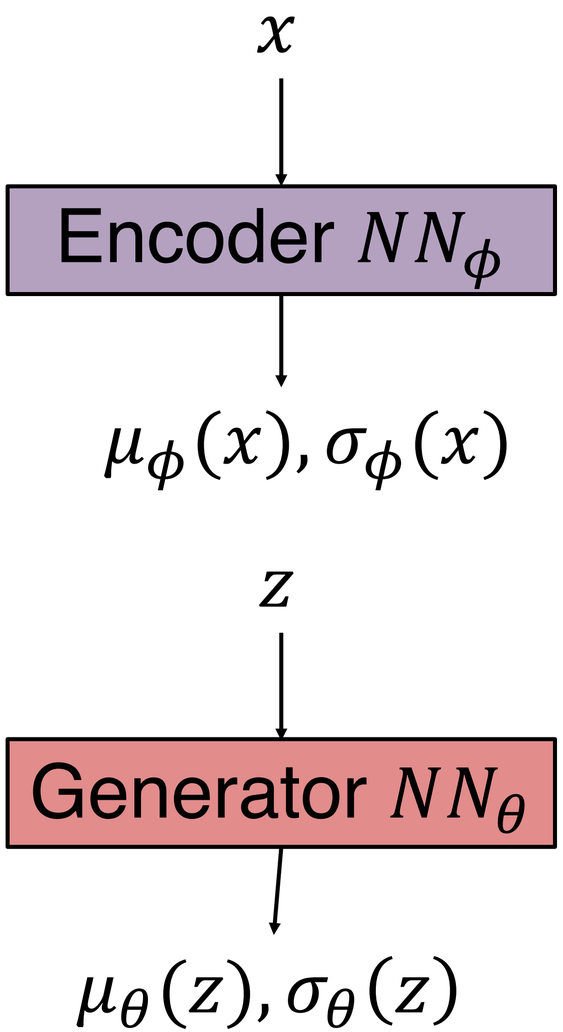


Figure 8: Shape arithmetic for chairs and tables. The left images show the obtained “arm” vector can be added to other chairs, and the right ones show the “layer” vector can be added to other tables. ostly mirrors the generator.

From Wu et al. *Learning a probabilistic latent space of object shapes via 3D Generative Adversarial Modeling* (NIPS 2016)

Gaussian Variational Autoencoder (VAE)

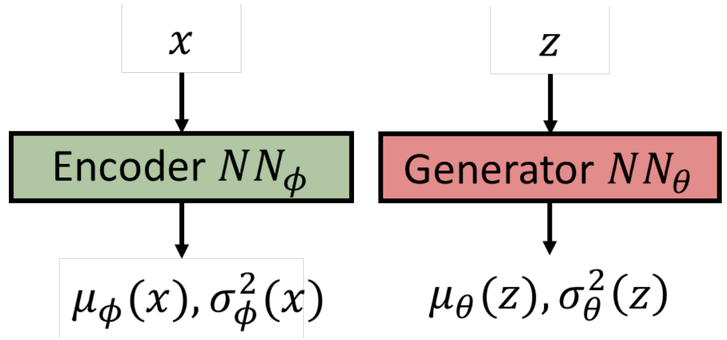


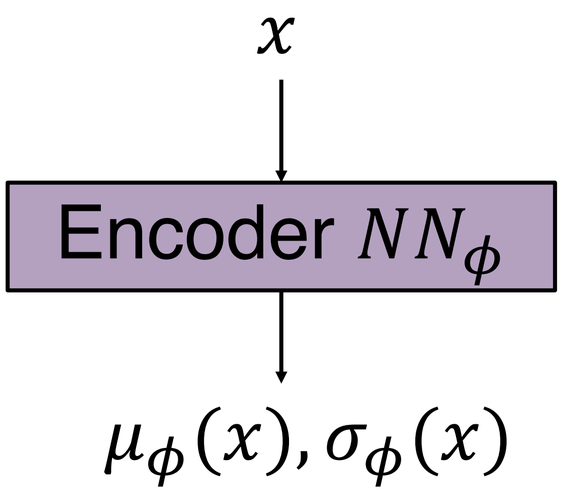
$$q_\phi(z|x) = \mathcal{N}(z|\mu_\phi(x), \sigma_\phi^2(x))$$

$$p_\theta(\tilde{x}|z) = \mathcal{N}(\tilde{x}|\mu_\theta(z), \sigma_\theta^2(z))$$

Kingma et Welling, *Auto-encoding Variational Bayes* (2013)

GVAE, test time: Encoding



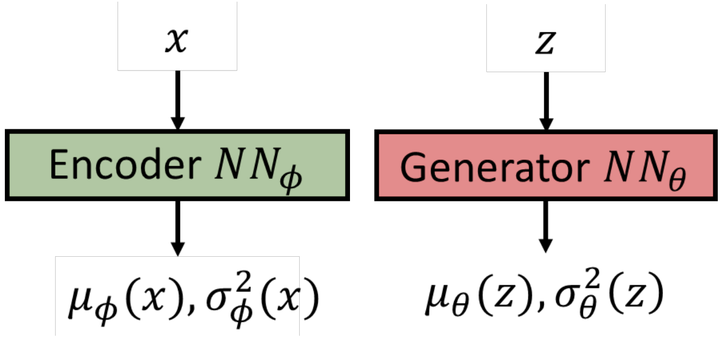


$q_\phi(z|x)$ is the distribution for the code of x

$\mu_\phi(x), \sigma^2_\phi(x) \in \mathbb{R}^d$ are elementwise the mean and variance for code $z \in \mathbb{R}^d$

Kingma et Welling, *Auto-encoding Variational Bayes* (2013)

GVAE, test time: Sample generation



1. Sample a latent code z

$$z \sim \mathcal{N}(0, I)$$

Generator NN_θ

2. Get component-wise mean and variance for x_*

$$\mu_\theta(z), \sigma_\theta(z)$$

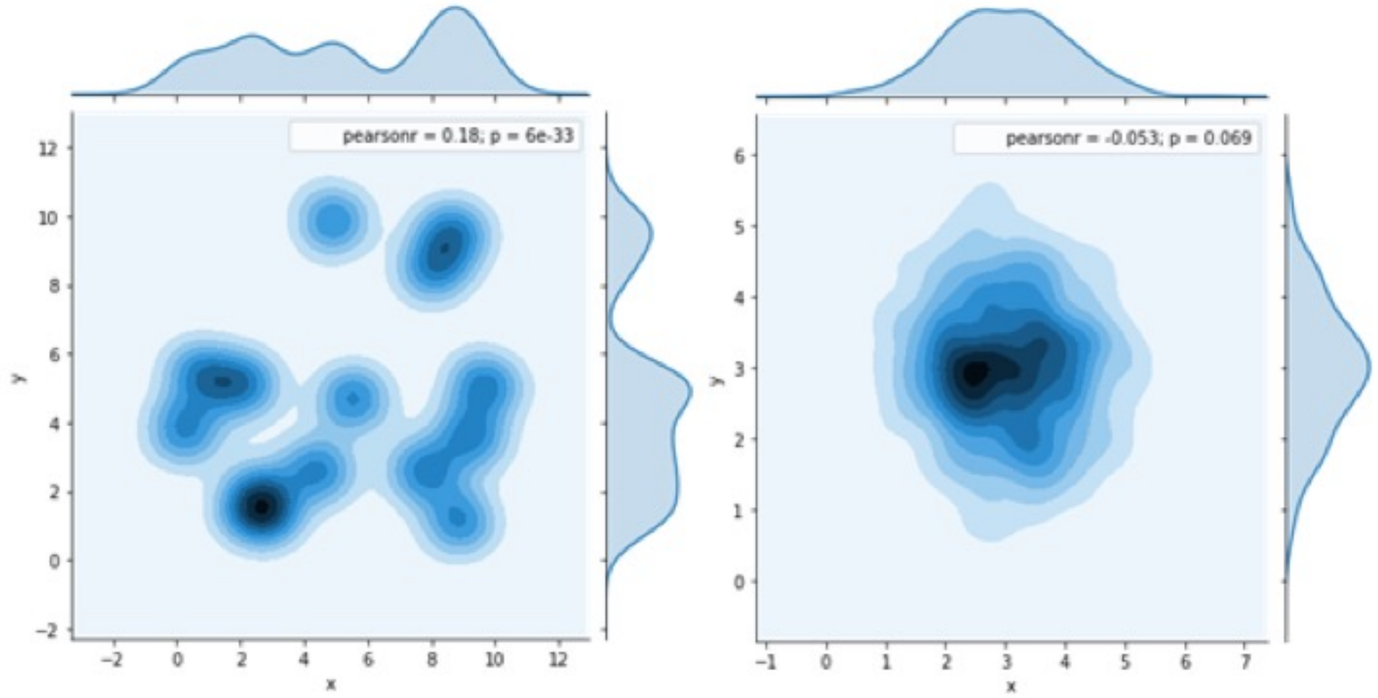
3. Sample a new x_*

$$\epsilon \sim \mathcal{N}(0, I)$$
$$x_* = \mu_\theta(z) + \epsilon \odot \sigma_\theta(z)$$

Kingma et Welling, *Auto-encoding Variational Bayes* (2013)

AE vs VAE

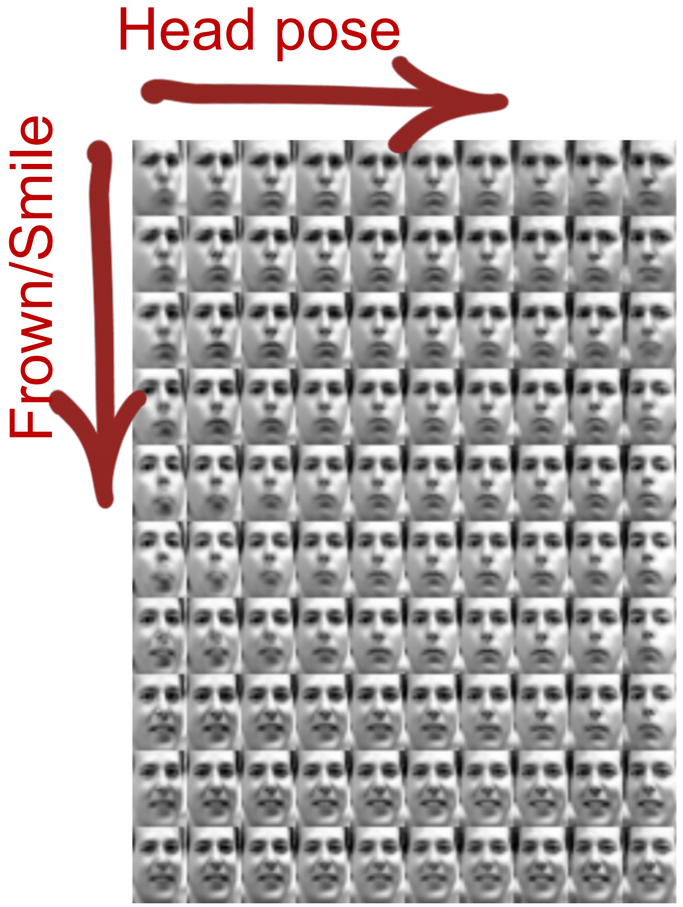
- Latent space of AEs is usually sparse with clusters around the support
- Latent space of VAEs is dense and behaves more like a Gaussian



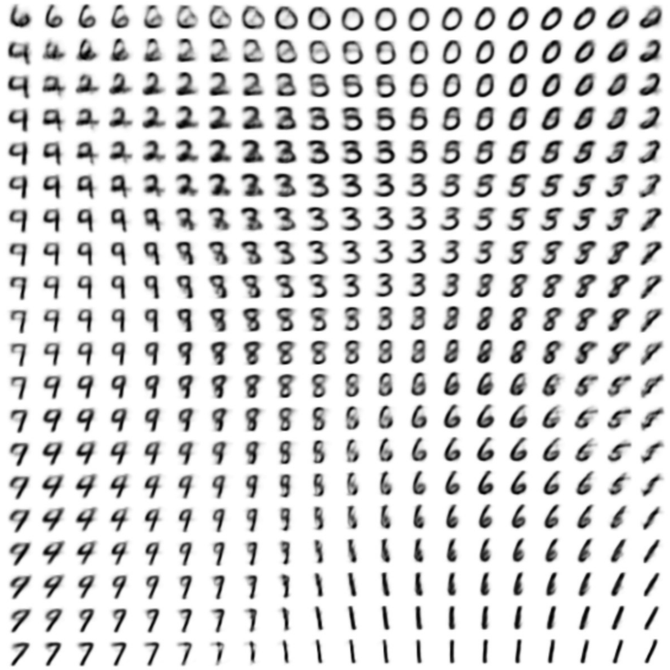
Source: https://news.sophos.com/en-us/2018/06/15/using-variational-autoencoders-to-learn-variations-in-data/

VAEs – learning the manifold of faces/digits:

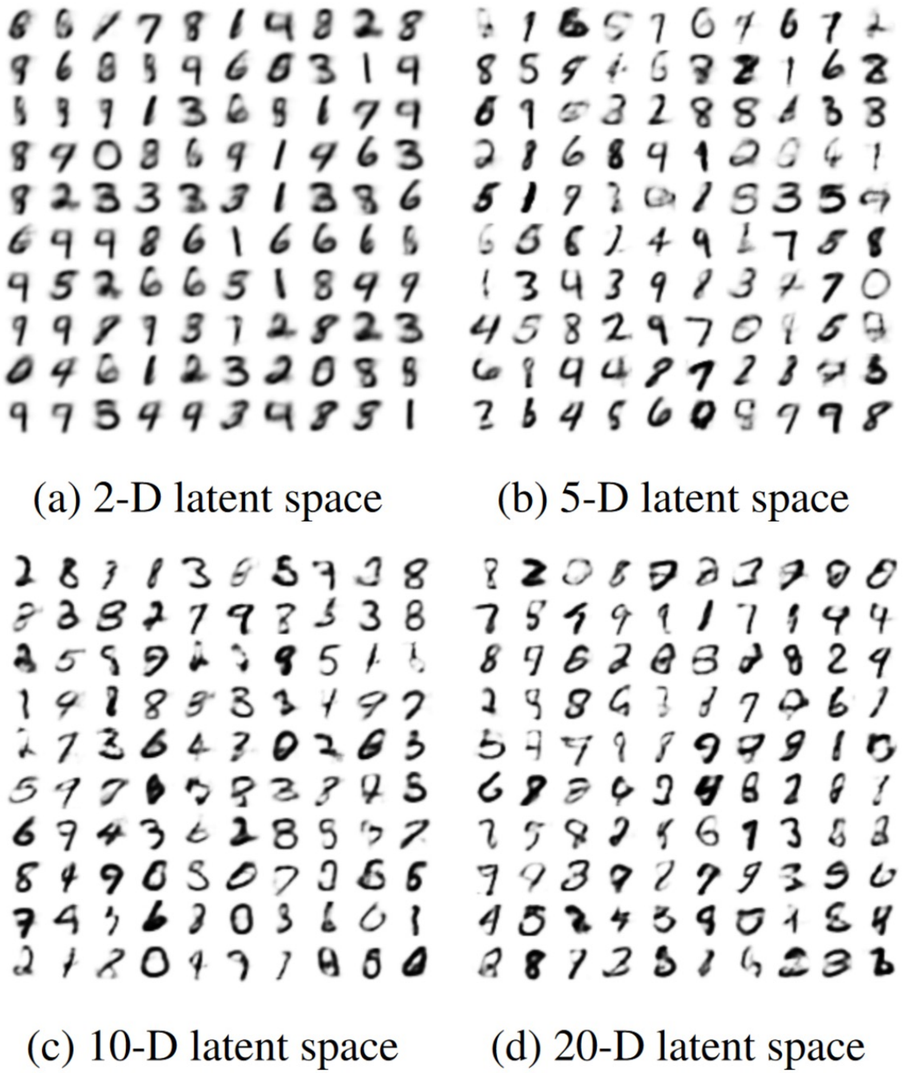
$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$ and $p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ are Multi-Layer Perceptron; $p_\theta(z) = \mathcal{N}(0, I)$



2D latent space, Frey faces



2D latent space, MNIST digits



(a) 2-D latent space (b) 5-D latent space

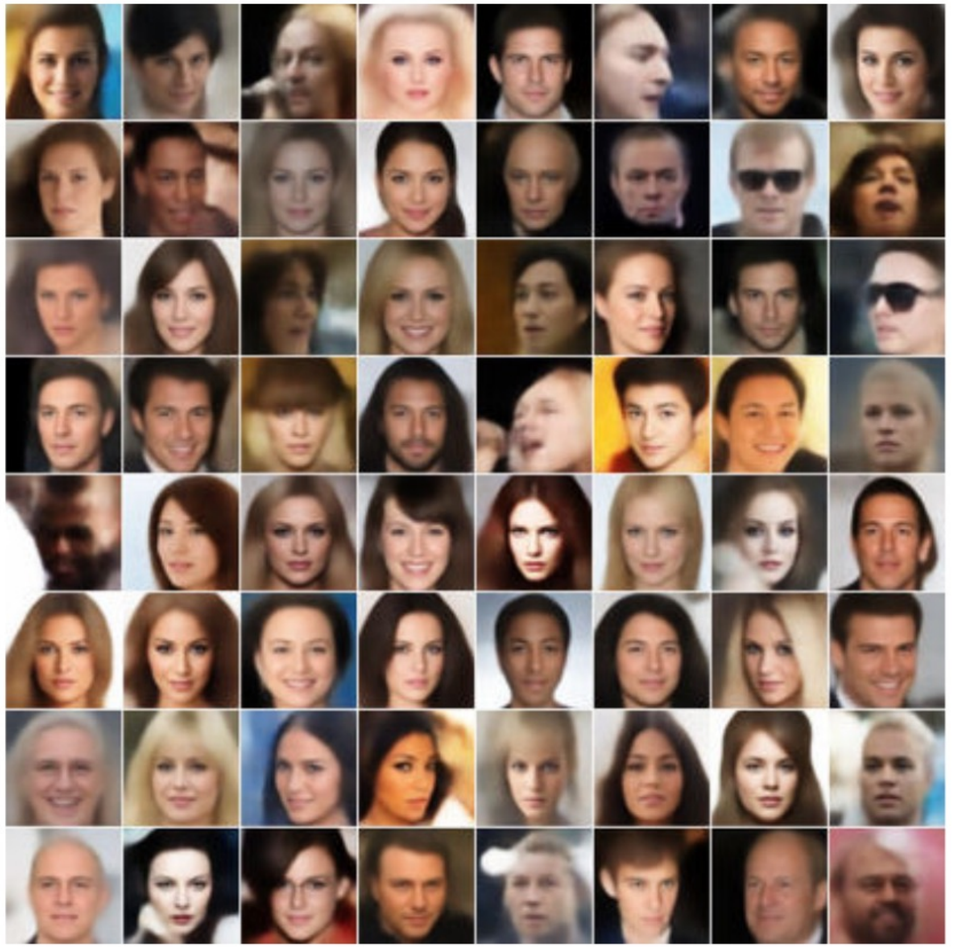
(c) 10-D latent space (d) 20-D latent space

Kingma et Welling, *Auto-encoding Variational Bayes* (2013)

VAEs – image encoding and reconstruction:

Input x

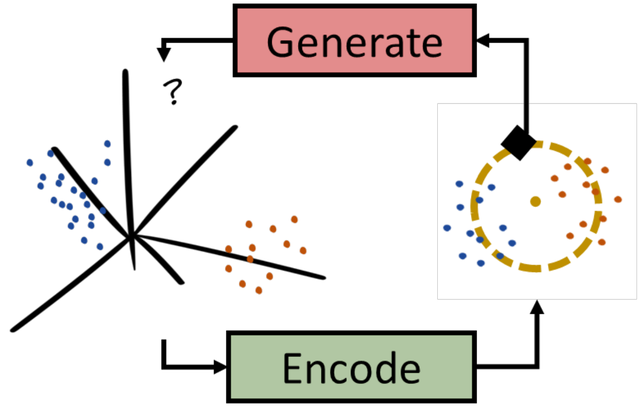
Reconstruction $\tilde{x} = \mu_\theta(\mu_\phi(x))$



Rosca et al., *Distribution Matching in Variational Inference* (2019)

VAEs: summary

- VAEs learn an **explicit encoding** q_ϕ from input to latent → **representation learning**
- VAEs **only indirectly** fit marginal distribution $p_\theta(x)$ → design **not** optimized for **sample realism**/quality
- Modelled data distribution depends on **latent prior** $p(z)$ → need rich learnable priors



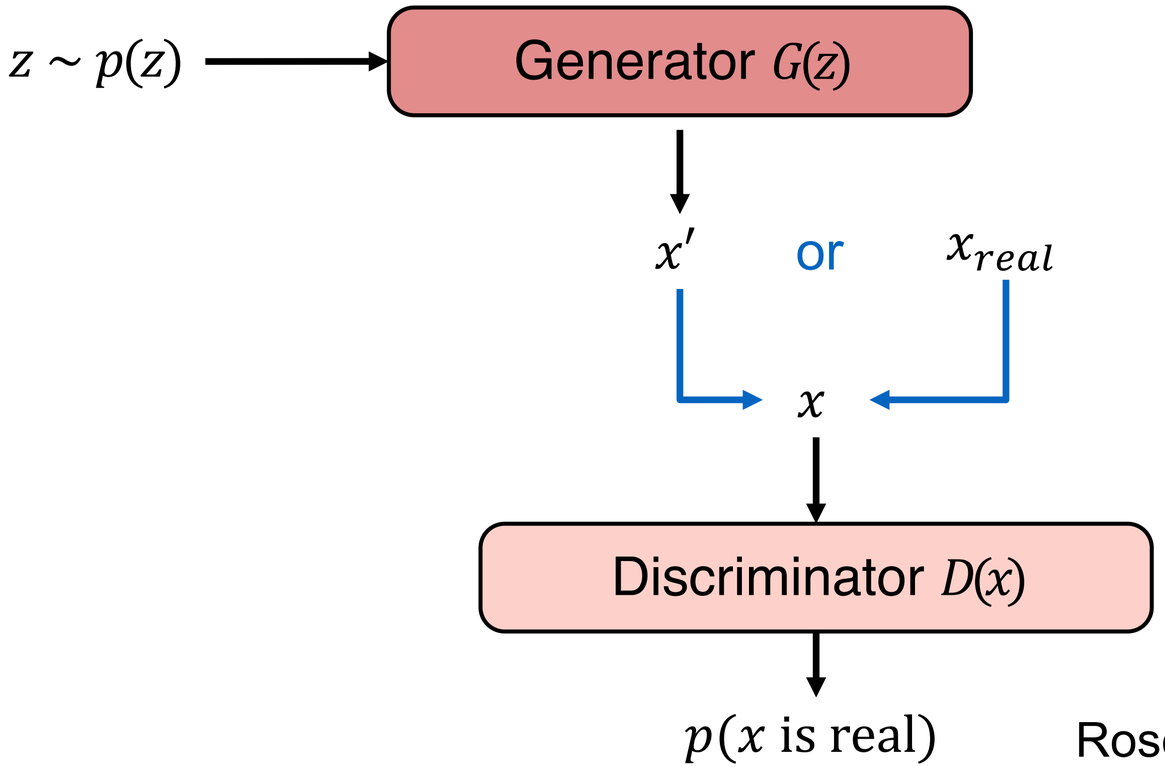
Unused parts of latent space → non-realistic samples

(a) Color-MNIST (b) CIFAR-10 (c) CelebA

Rosca et al., *Distribution Matching in Variational Inference* (2019)

Generative Adversarial Networks (GANs)

- GANs = Generator + Discriminator
- GANs optimize a minimax objective designed to encourage high-quality samples
- Generator tries to fool discriminator. Discriminator learns to classify generated vs. real data samples



$$\min_G \max_D V(G, D) = \mathcal{L}_{data}(D) + \mathcal{L}_{model}(D, G)$$

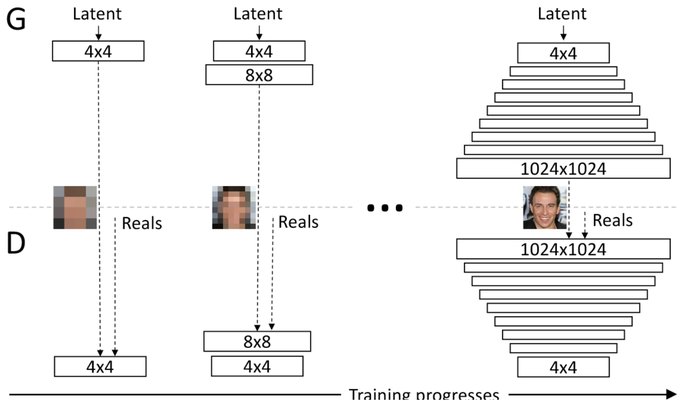
$$\mathcal{L}_{data}(D) = \mathbb{E}_{x \sim data}[\log D(x)]$$

$$\mathcal{L}_{model}(D) = \mathbb{E}_{x' \sim G(z)}[\log(1 - D(x'))]$$

Rosca et al., *Distribution Matching in Variational Inference* (2019)

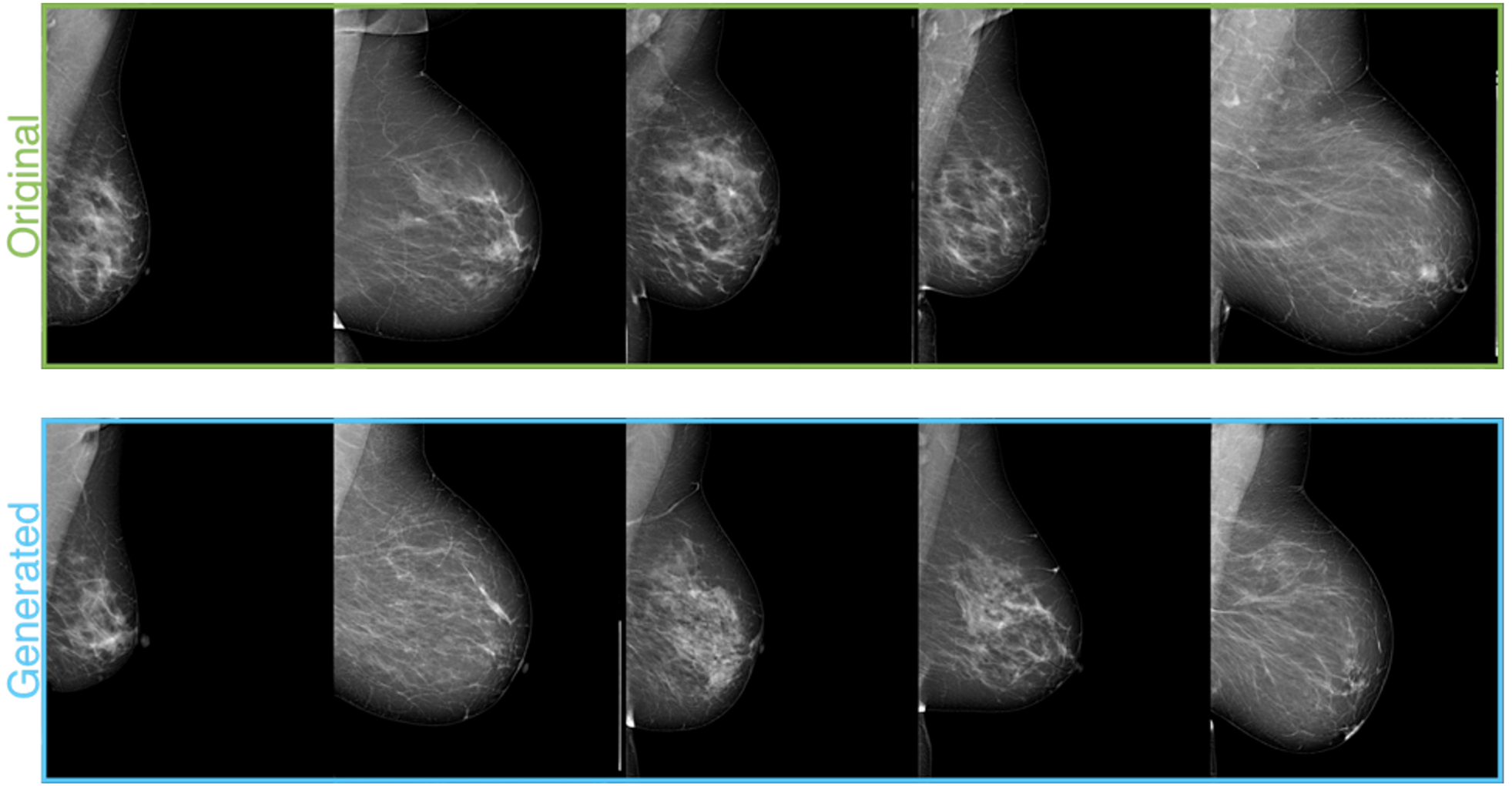
Example: GAN HQ sample generation (Karras et al.)

- Incrementally adds higher-res GAN layers during training to go from vignette-like to HQ fakes.



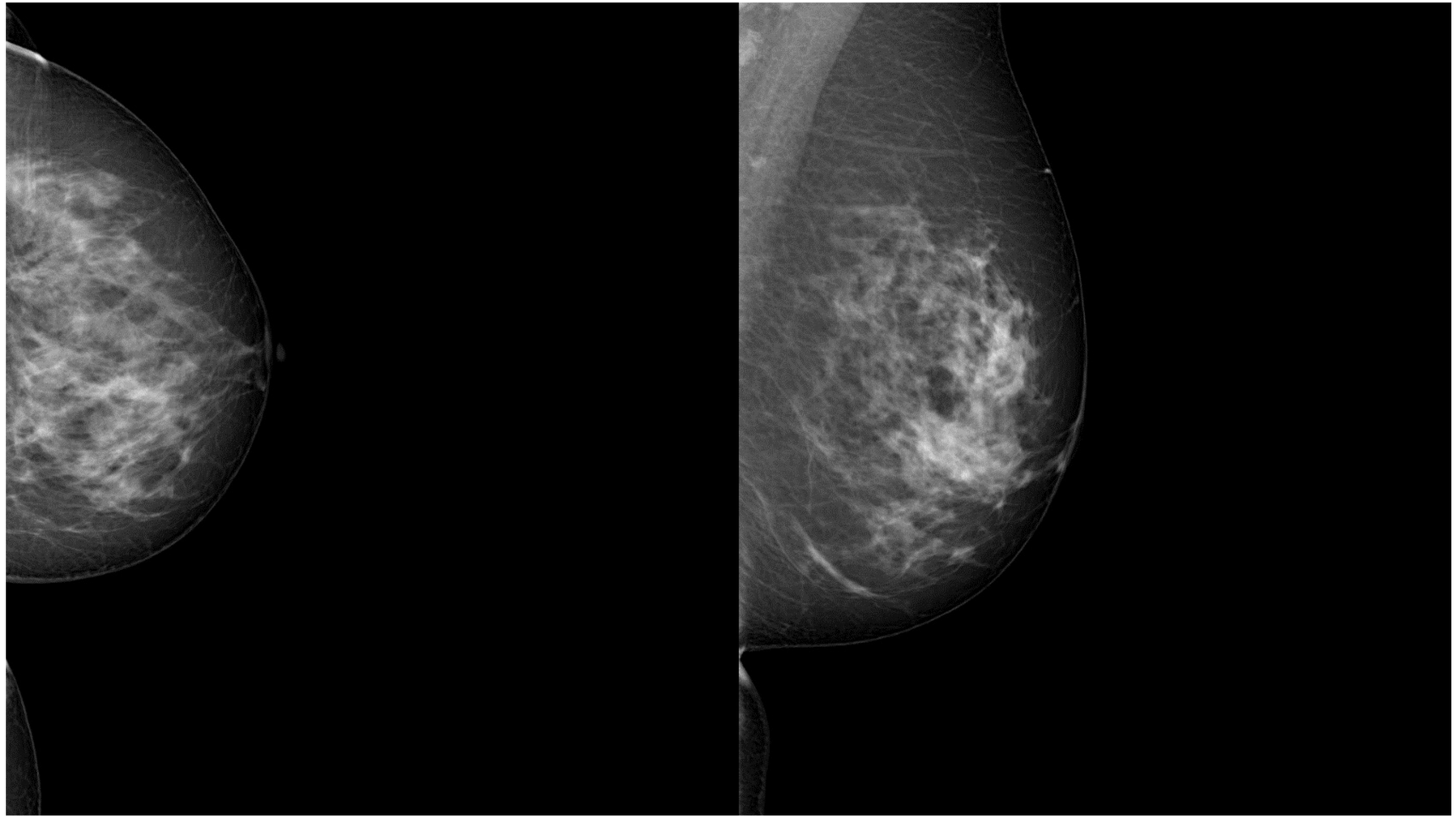
Karras, Aila, Laine, Lehtinen: *Progressive growing of GANs for improved quality, stability and variation*

Example: GAN to synthesize medical scans



Korkinof, Harvey, Heindl, Karpati, Williams, Rijken, Kecskemethy, Glocker: Perceived Realism of High Resolution Generative Adversarial Network Derived Synthetic Mammograms

Example: GAN to synthesize medical scans



Korkinof, Harvey, Heindl, Karpati, Williams, Rijken, Kecskemethy, Glocker: Perceived Realism of High Resolution Generative Adversarial Network Derived Synthetic Mammograms

GAN / VAE

GANs focus on sampling/ They learn to **produce realistic (new) samples**

Implicit p.d.f. $p(x|z)$ through $x = G(z)$.
Implicit $p(z|x)$ (**no** encoding/reconstruction)

Implicit p.d.f. $p(x)$ through $x = G(z)$ and $p(z)$

Trained by playing **minimax game**

Marginal $p(x)$ **explicitly** encouraged to **fit** $\boldsymbol{p_{data}}$
Latent space explicitly encouraged to **map to plausible samples**

Mode collapse: GANs may produce a limited variety of samples.

VAEs focus on **representation learning**
They learn to **encode and reconstruct**

Explicit p.d.f. $p_\theta(x|z)$
Explicit $q_\phi(z|x) \simeq p_\theta(z|x)$

Implicit p.d.f. $p(x)$ through $p_\theta(x|z)$ and $p_\theta(z)$

Trained by **ELBO maximization**

Distribution of **real data latent codes** $\int q_\phi(z|x)p_{data}(x)dx$ may be **mismatched with prior** $p_\theta(z)$. Then latent codes can map to **implausible samples**.

VAE objective encourages cycle consistency (reconstructed $\simeq$ original data)

Deep Generative Models

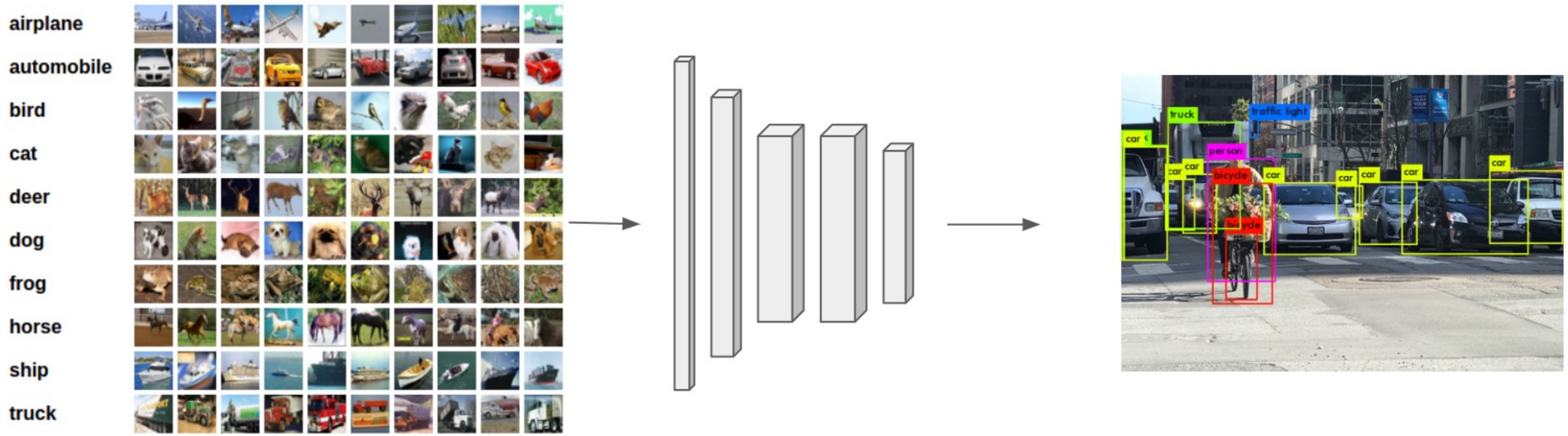
- Active area of research
  - Countless applications: style transfer, image-to-caption, fake news, image inpainting, NLP, sketch-to-photorealistic
  - Coupling VAE- and GAN-like designs, rich learnable priors on the latent space, better network architectures, etc.
  - Disentangling & Data invariances: decoupling and understanding appearance, shape, pose, etc. automatically
  - Adapting models to complex data: proteins, chemical molecules, graphs, shapes
- Other models available:
  - PixelCNN
  - Flow-Based DGM
  - etc.

Imperial College London

Self-supervised learning

Supervised learning: A Success Story

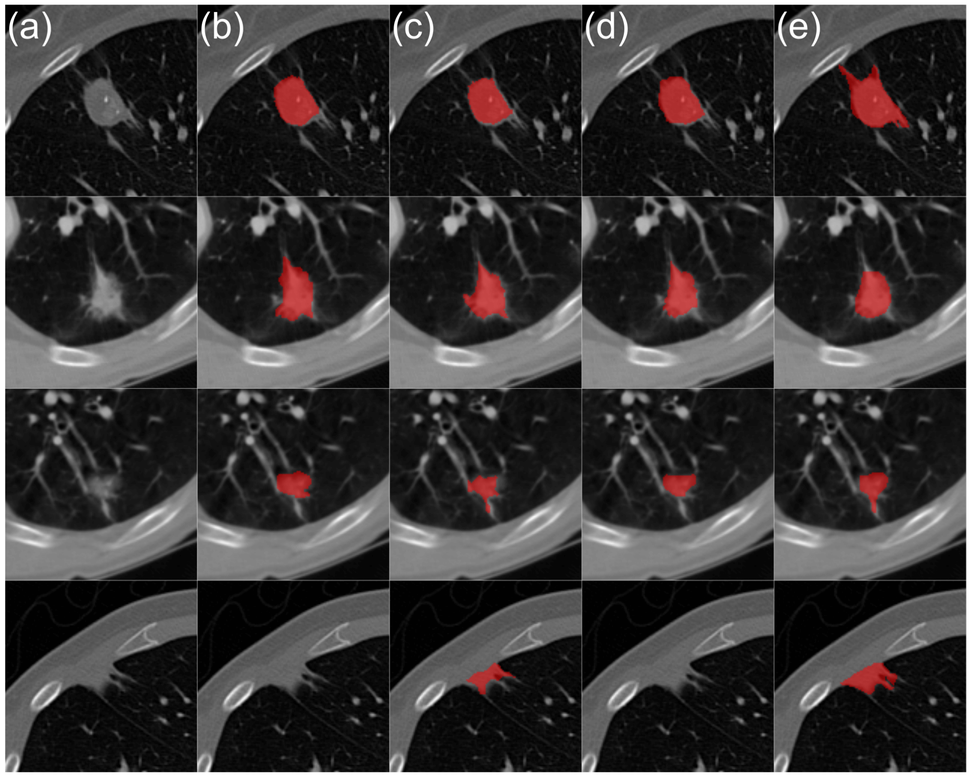
- Acquire a large-scale annotated dataset for that task.
- Specify a training loss and neural network architecture.
- Train the network and deploy.



Goal: Learn a representation!

Data annotation challenges

- Annotating images seems straightforward task, but it is not.
- Requires intense human labor (annotating + cleaning raw data).
- Humans are prone to errors.
- Time consuming and expensive.
- Human labelling is sparse.



Stochastic Segmentation Networks:
Modelling Spatially Correlated Aleatoric Uncertainty

Miguel Monteiro
Imperial College London
mm6818@ic.ac.uk

Loïc Le Folgoc
Imperial College London
llefolgo@ic.ac.uk

Daniel Coelho de Castro
Imperial College London
dc315@ic.ac.uk

Nick Pawlowski
Imperial College London
np716@ic.ac.uk

Bernardo Marques
Imperial College London
bgmarque@ic.ac.uk

Konstantinos Kamnitsas
Imperial College London
kk2412@ic.ac.uk

Mark van der Wilk
Imperial College London
m.vdwilk@ic.ac.uk

Ben Glocker
Imperial College London
b.glocker@ic.ac.uk

Can we exploit the raw data?

- Acquiring raw data is usually easy.
- Typical supervised methods cannot exploit and benefit from it.
- Rich signal in the data itself that cannot be grasped and annotated by humans.

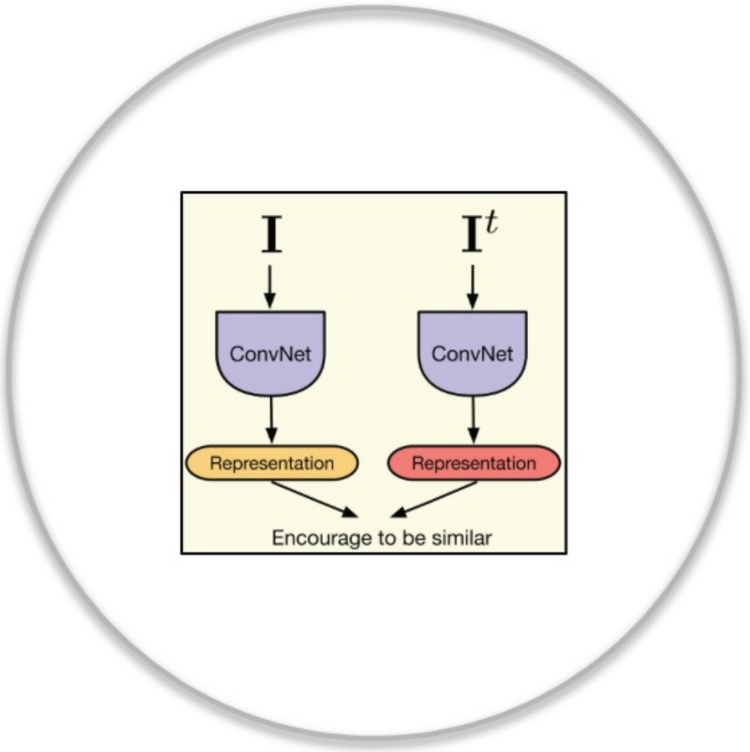
Deep learning requires large amounts of carefully labeled data which is difficult to acquire and expensive to annotate.

What is self-supervised learning?

- A form of unsupervised learning where the data (not the human) provides the supervision signal
- Usually, define a pretext task for which the network is forced to learn what we really care about
- For some pretext tasks, a part of the data is withheld and the network has to predict it
- Others try to detect transformations of the data
- The features/representations learned on the pretext task are subsequently used for a different downstream task, usually where some annotations are available.

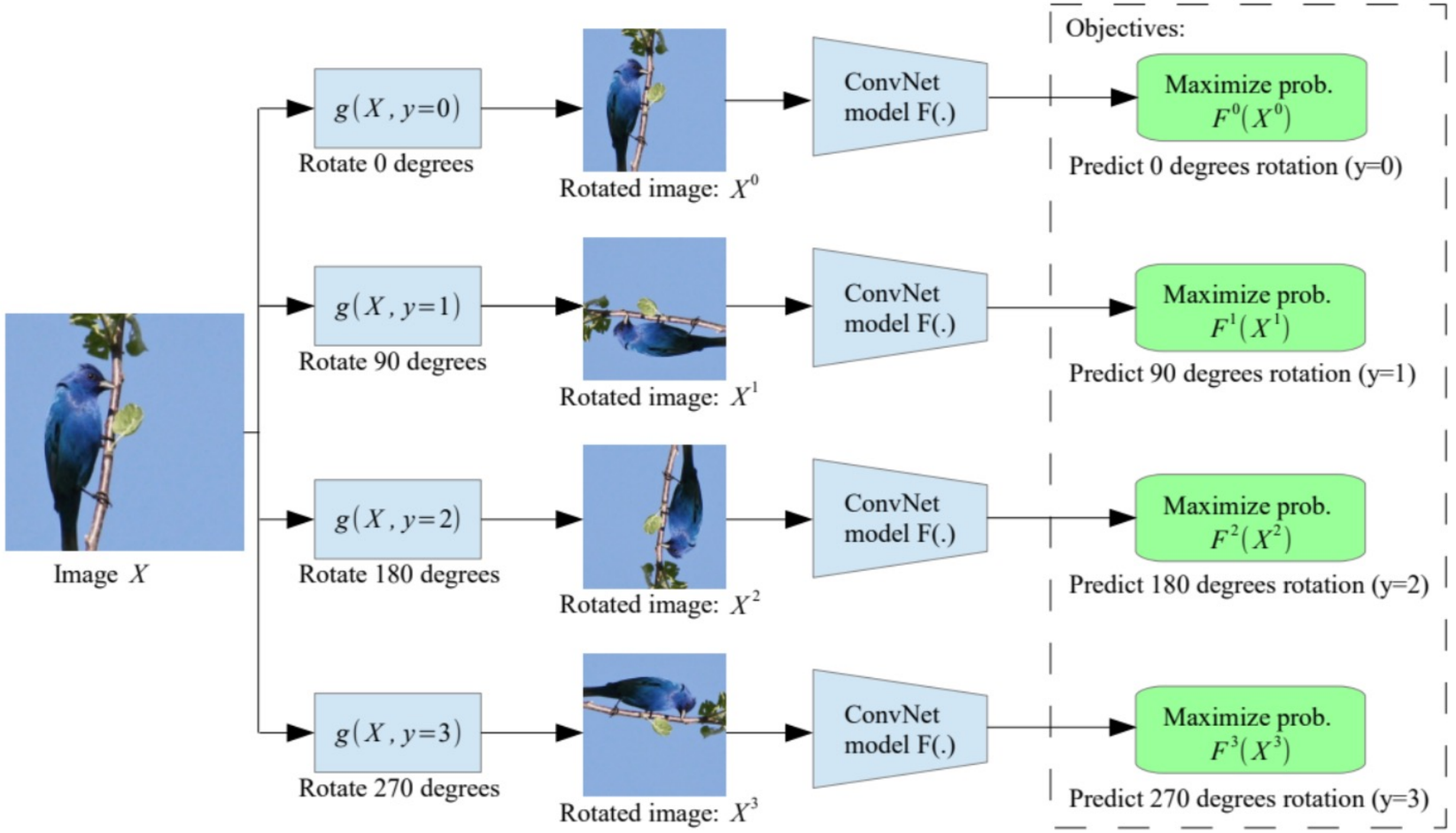
How can we actually do self-supervised learning?

Handcrafted Pretext Tasks



Contrastive Learning

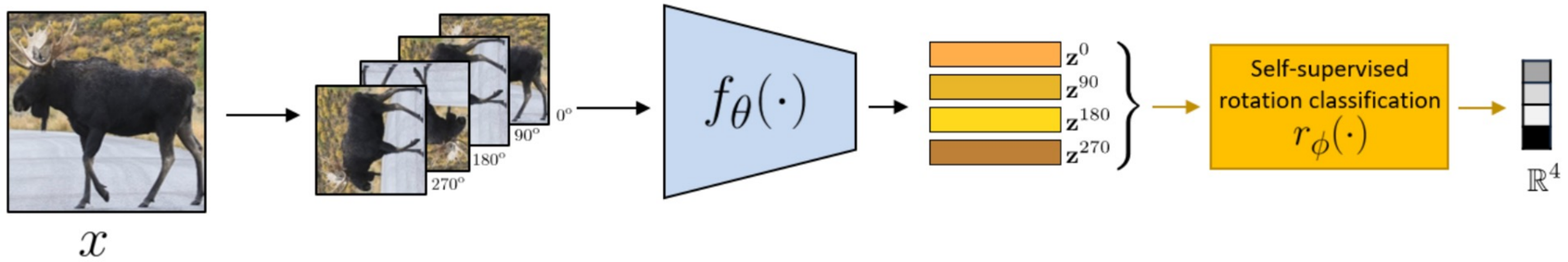
Example: Rotation prediction



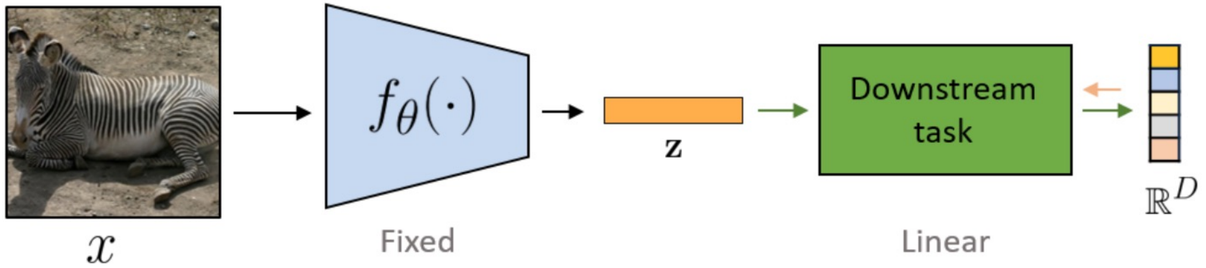
Predict the orientation of the image

Self-supervised learning pipeline

Stage 1: Train network on pretext task (without human labels)



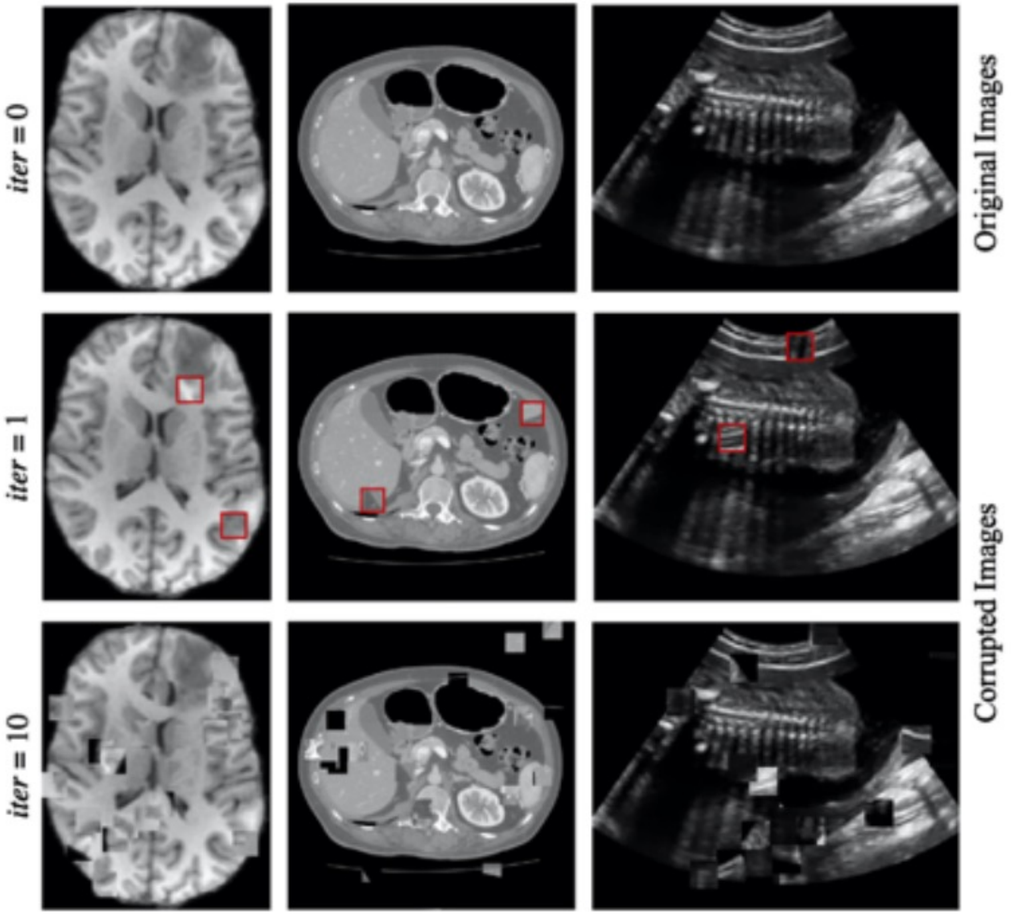
Stage 2: Train classifier on learned features for new task with fewer labels



Design of the pretext task

Example: Context restoration

- Randomly select two isolated small patches in given image and swap their contents.
- Repeat T times. (The intensity distribution is preserved, the spatial info is altered).
- Attempt to restore the corrupted images.

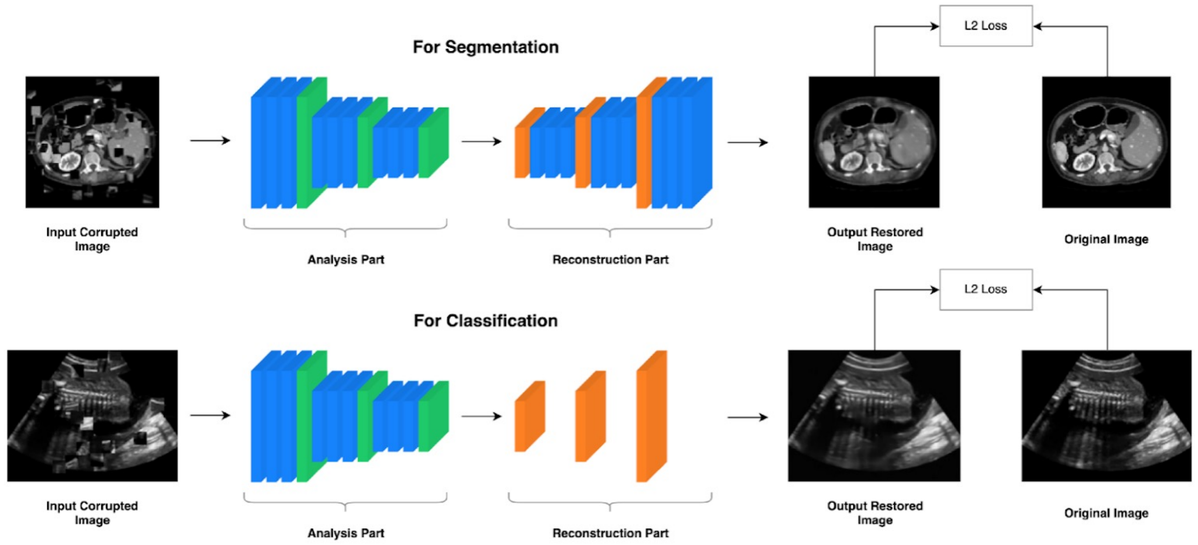


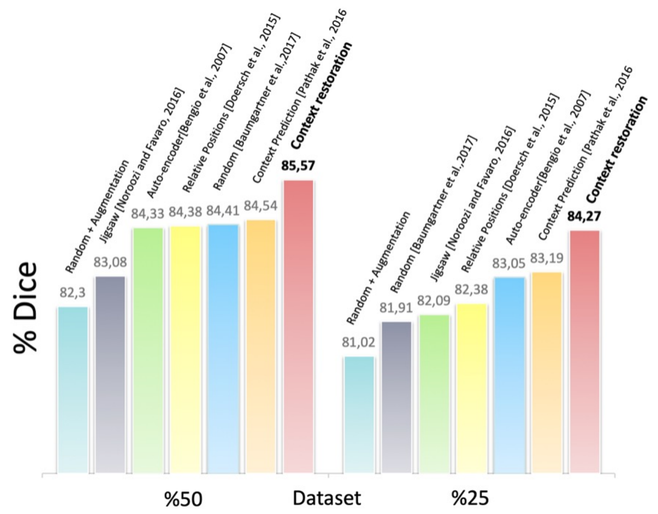
Chen, L., Bentley, P., Mori, K., Misawa, K., Fujiwara, M., & Rueckert, D. (2019). Self-supervised learning for medical image analysis using image context restoration. *Medical image analysis, 58*, 101539 .

Design of the pretext task

Example: Context restoration

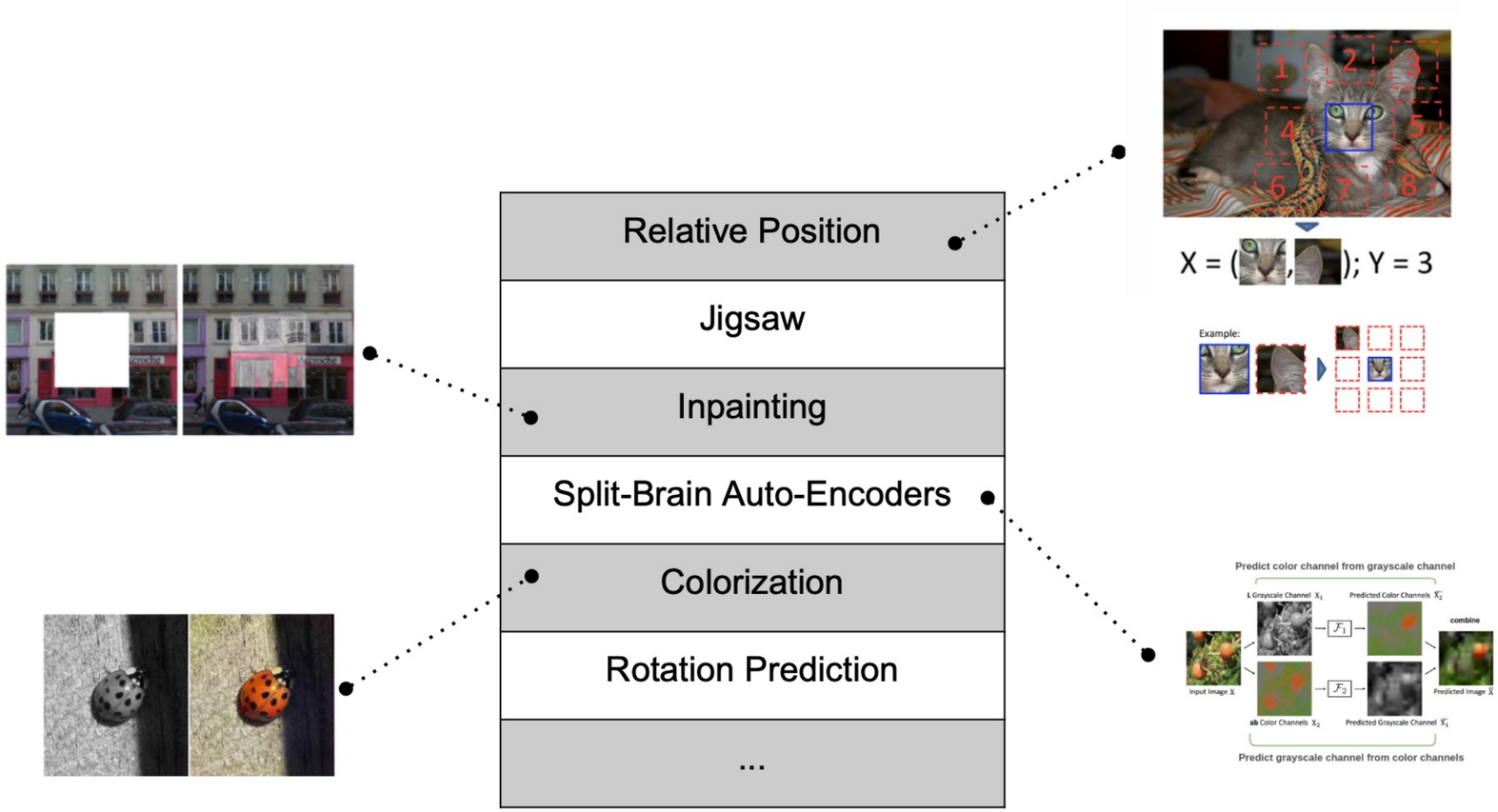
- We do not care to measure performance on the pretext task.
- We evaluate the performance on how well the learnt representations perform on downstream tasks.





Other examples of pretext tasks

Learn a data transformation to capture the image semantics.



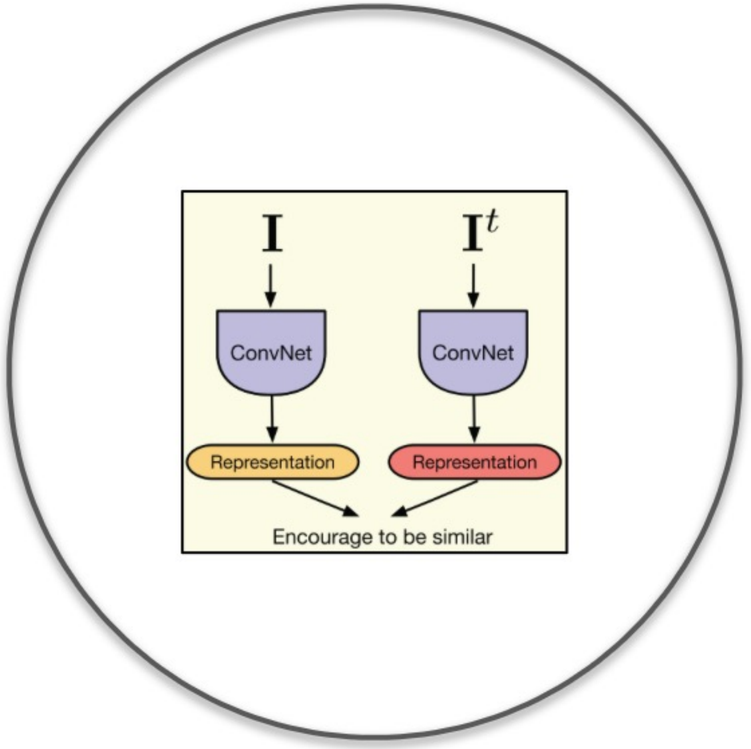
Summary on pretext task design

- Pretext tasks focus on common sense, e.g. patch rearrangement, predicting rotations, colorization, inpainting etc.
- By solving the pretext task, the models are forced to learn good features (semantic representation) about natural images.
- Attention is focused on the performance of the downstream tasks (classification , detection, segmentation) and not the performance of the pretext task.

- **Challenges:**
  - Designing a pretext task is not always as straightforward as it might seem.
  - The learnt representations may not be as general as we want.

How can we actually do self-supervised learning?

Handcrafted Pretext Tasks



Contrastive Learning

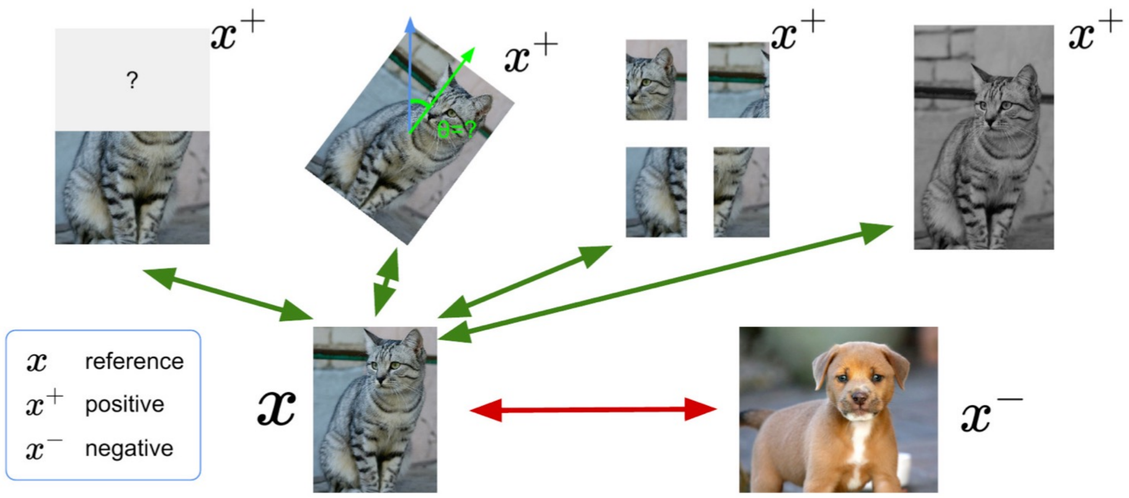
What is contrastive learning?

- From a given image, sample a “positive” image and “negative” images.
- Fit a scoring model such that:

$$score(x, x_{pos}) > score(x, x_{neg})$$

What is contrastive learning?

- Positive samples: sample 2 parts of the same image under some transformation.
- Negative samples: Sample parts of a random image or image at different location.



Common transformations:

- Crop, resize, flip
- Rotation, cutout
- Colour drop, jitter
- Gaussian noise/blur
- Sobel filter

Pairwise Contrastive Loss

Sample a pair of images and compute their distance:

$$D_i = ||x, x_i||_2$$

If positive sample:

$$L_i = D_i^2$$

If negative sample:

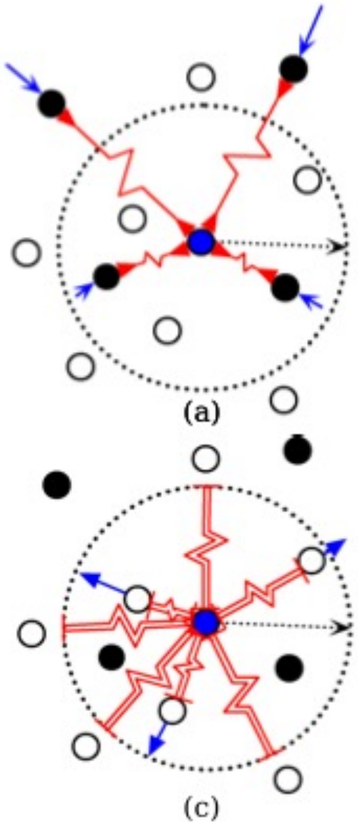
$$L_i = max(0, \varepsilon - D_I)^2$$

x

pos

x

neg



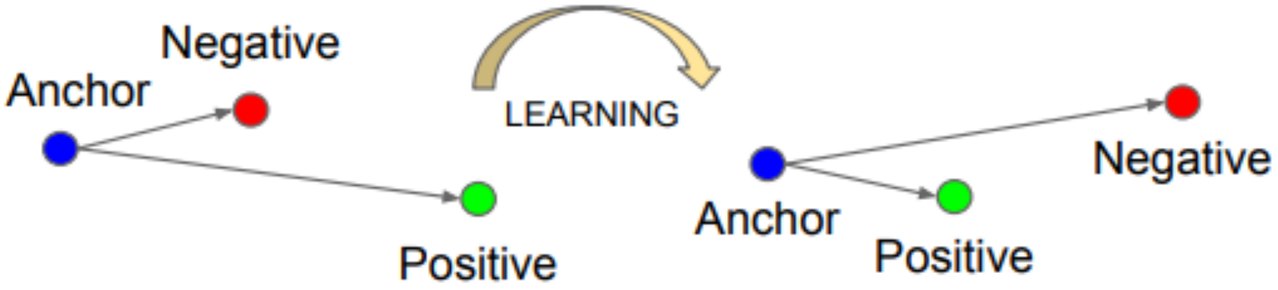
Triplet loss

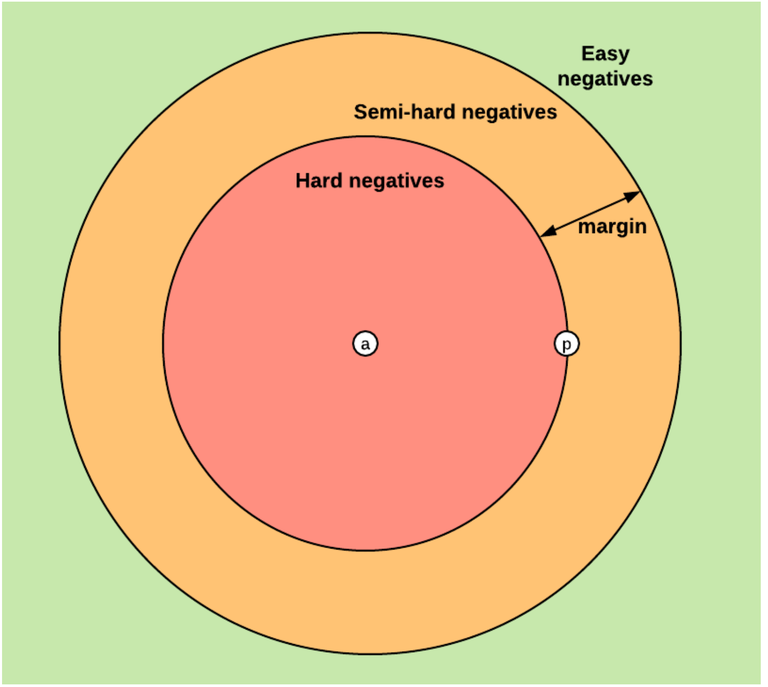
$$D_{pos} = ||x, x_{pos}||_2$$
$$D_{neg} = ||x, x_{neg}||_2$$
$$L = max(0, D_{pos}^2 - D_{neg}^2 - margin)$$

anchor

pos

neg





https://omoindrot.github.io/triplet-loss

InfoNCE Loss

Given a set of samples: $x \quad x^+ \quad x_j^-$

The loss is the log likelihood of predicting the positive sample correctly.

Multi-class cross entropy loss function.

$$\mathcal{L} = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

InfoNCE Loss

Cosine similarity

$$s(\mathbf{f}_1, \mathbf{f}_2) = \frac{\mathbf{f}_1^\top \mathbf{f}_2}{\|\mathbf{f}_1\| \, \|\mathbf{f}_2\|}$$

$$\mathcal{L} = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\exp(s(f(x), f(x^+))) + \sum_{j=1}^{N-1} \exp(s(f(x), f(x_j^-)))} \right]$$

- This is commonly known as the InfoNCE loss and its negative is a lower bound on the mutual information between $f(x)$ and $f(x^+)$.

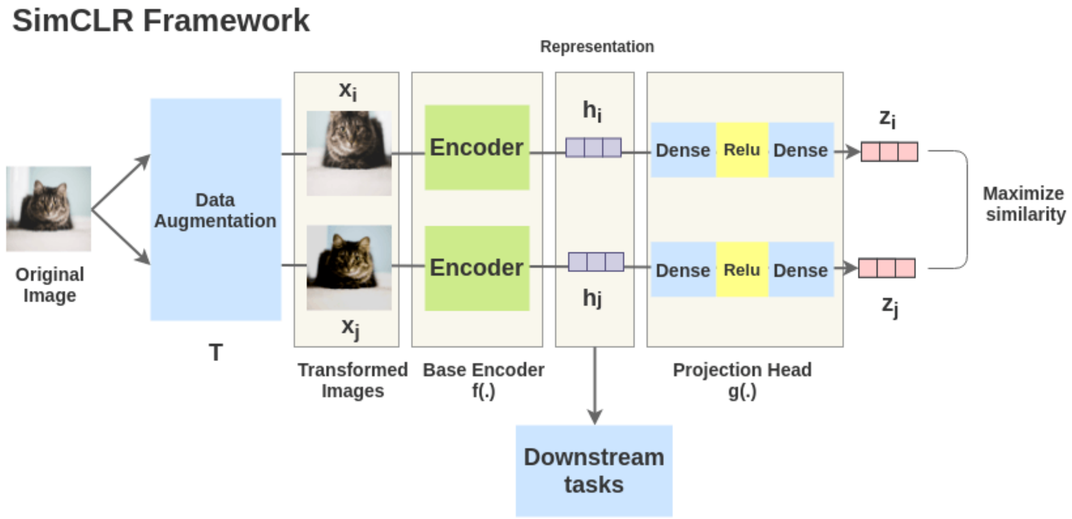
$$MI[f(x), f(x^+)] \geq \log(N) - \mathcal{L}$$

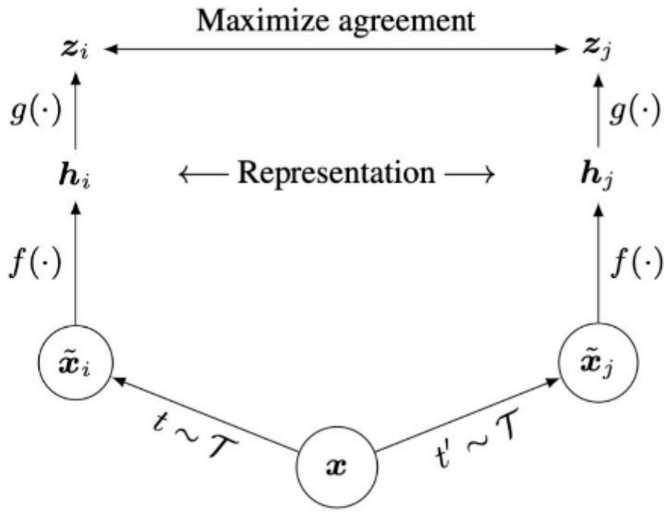
- Maximising mutual information between features extracted from multiple views forces the features to capture information about higher-level factors
- **Note:** The larger the negative sample size (N), the tighter the bound.

A Simple Framework for Contrastive Learning (SimCLR)

Siamese network architecture

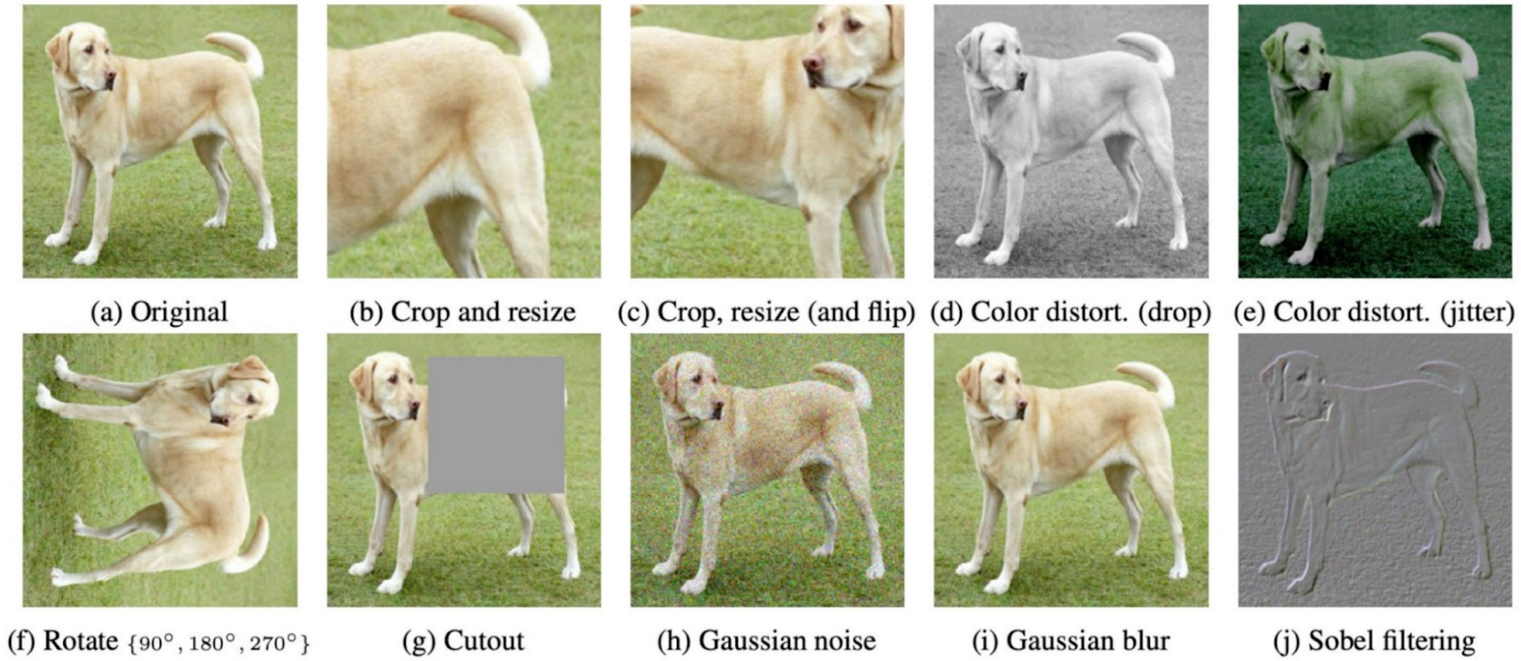
- The base encoder learns the representations.
- The projection head projects feature to an embedding space where contrastive learning is applied.
- Heavily depends on the batch size. (The larger, the better!)





A Simple Framework for Contrastive Learning

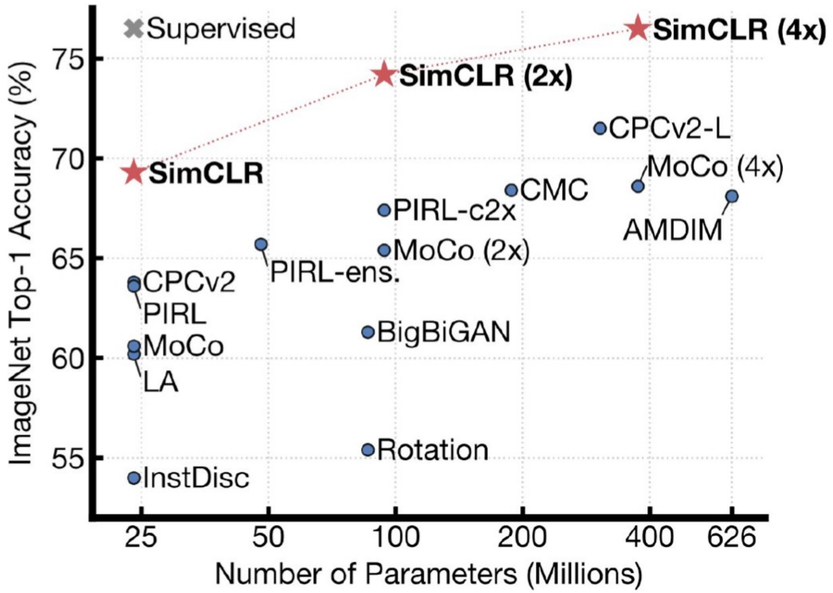
- The learnt representation quality is dependent on the augmentation techniques used.



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

A Simple Framework for Contrastive Learning



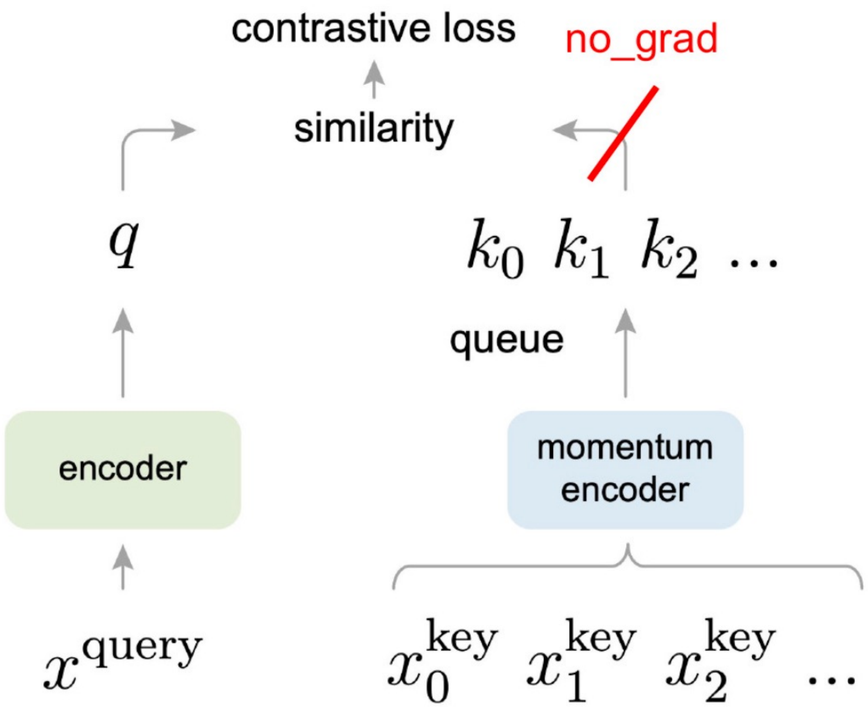
Method	Architecture	Label fraction 1%	Label fraction 10%
		Top 5	Top 5
Supervised baseline	ResNet-50	48.4	80.4
Methods using other label-propagation:			
Pseudo-label	ResNet-50	51.6	82.4
VAT+Entropy Min.	ResNet-50	47.0	83.4
UDA (w. RandAug)	ResNet-50	-	88.5
FixMatch (w. RandAug)	ResNet-50	-	89.1
S4L (Rot+VAT+En. M.)	ResNet-50 (4×)	-	91.2
Methods using representation learning only:			
InstDisc	ResNet-50	39.2	77.4
BigBiGAN	RevNet-50 (4×)	55.2	78.8
PIRL	ResNet-50	57.2	83.8
CPC v2	ResNet-161(∗)	77.9	91.2
SimCLR (ours)	ResNet-50	75.5	87.8
SimCLR (ours)	ResNet-50 (2×)	83.0	91.2
SimCLR (ours)	ResNet-50 (4×)	**85.8**	**92.6**

Train feature encoder on the entire training set of the ImageNet.

Freeze the feature encoder.
Train a linear classifier on top with labelled data.

Finetune the feature encoder
with 1%, 10% of labeled data on ImageNet.

Momentum Contrast (MoCo)



Differences to SimCLR:

- Queue (memory bank) of keys for negative samples.
- Back-propagate the gradients only to query encoder, not the queue.
- Memory bank can be much larger than the mini-batch.
- To ensure consistency of keys in queue use momentum update rule:

$$\theta_k = \beta\,\theta_k + (1-\beta)\,\theta_q$$

Improved Momentum Contrast

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

- Non-linear projection head.
- Heavy data augmentation.
- MoCo v2 outperforms SimCLR with much smaller mini-batch sizes.

Summary

- Data annotation can be difficult, expensive and time consuming.
- In self-supervised methods the data itself provides the supervisory signal.
  - Exploits a huge number of data that comes without annotations.
  - Goal is to learn more generic representations using pre-text tasks.
  - Learnt representations can be fine tuned to solve a downstream task.
  - Sometimes the pretext task cannot generate generic features.

- Contrastive learning produces state of the art results, closing the gap to fully supervised methods.

What we learned today

- Clustering:
  - k-Means
  - Mixture Model
- Dimensionality Reduction:
  - PCA
  - autoencoders
- Generative Modelling:
  - VAEs
  - GANs
- Self-supervised learning
  - Pretext tasks
  - Contrastive Learning

Imperial College London

That’s all for now