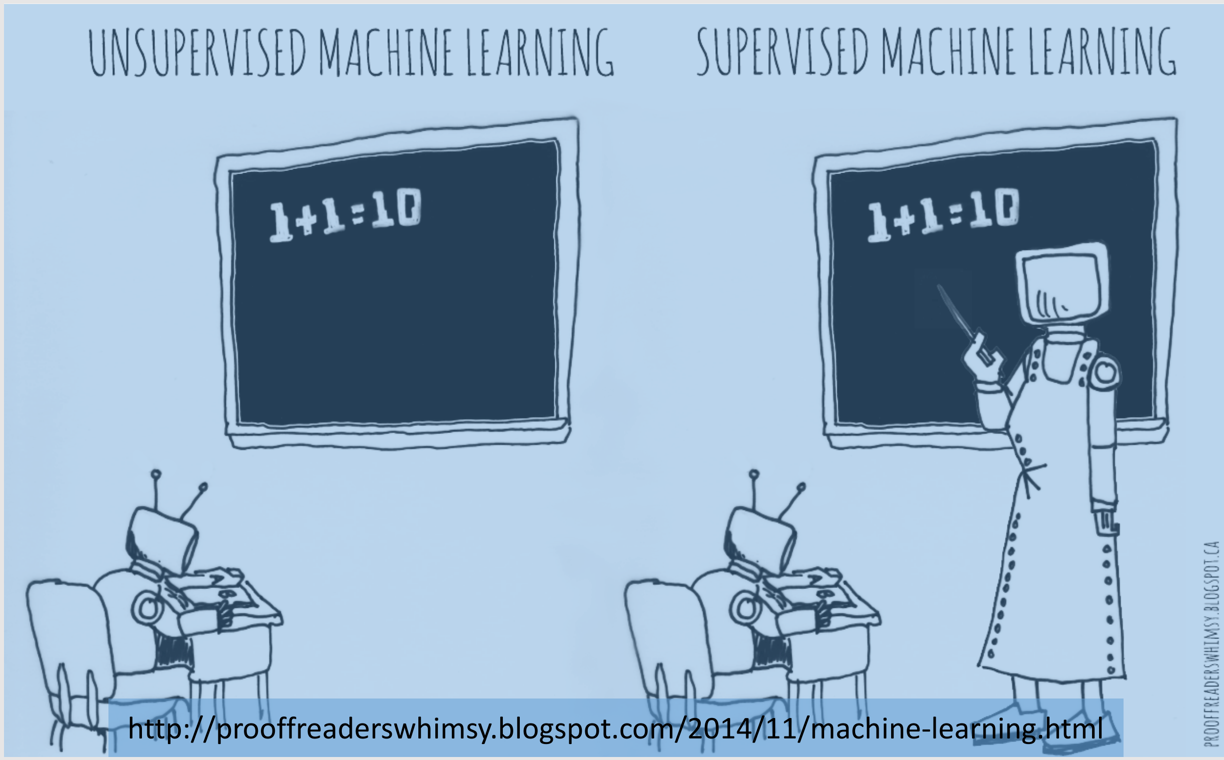
Lecture 8: Unsupervised Learning



http://prooffreaderswhimsy.blogspot.com/2014/11/machine-learning.html

Haiping Lu

YouTube Playlist: https://www.youtube.com/c/HaipingLu/playlists

COM4059/6059: MLAI21@The University of Sheffield

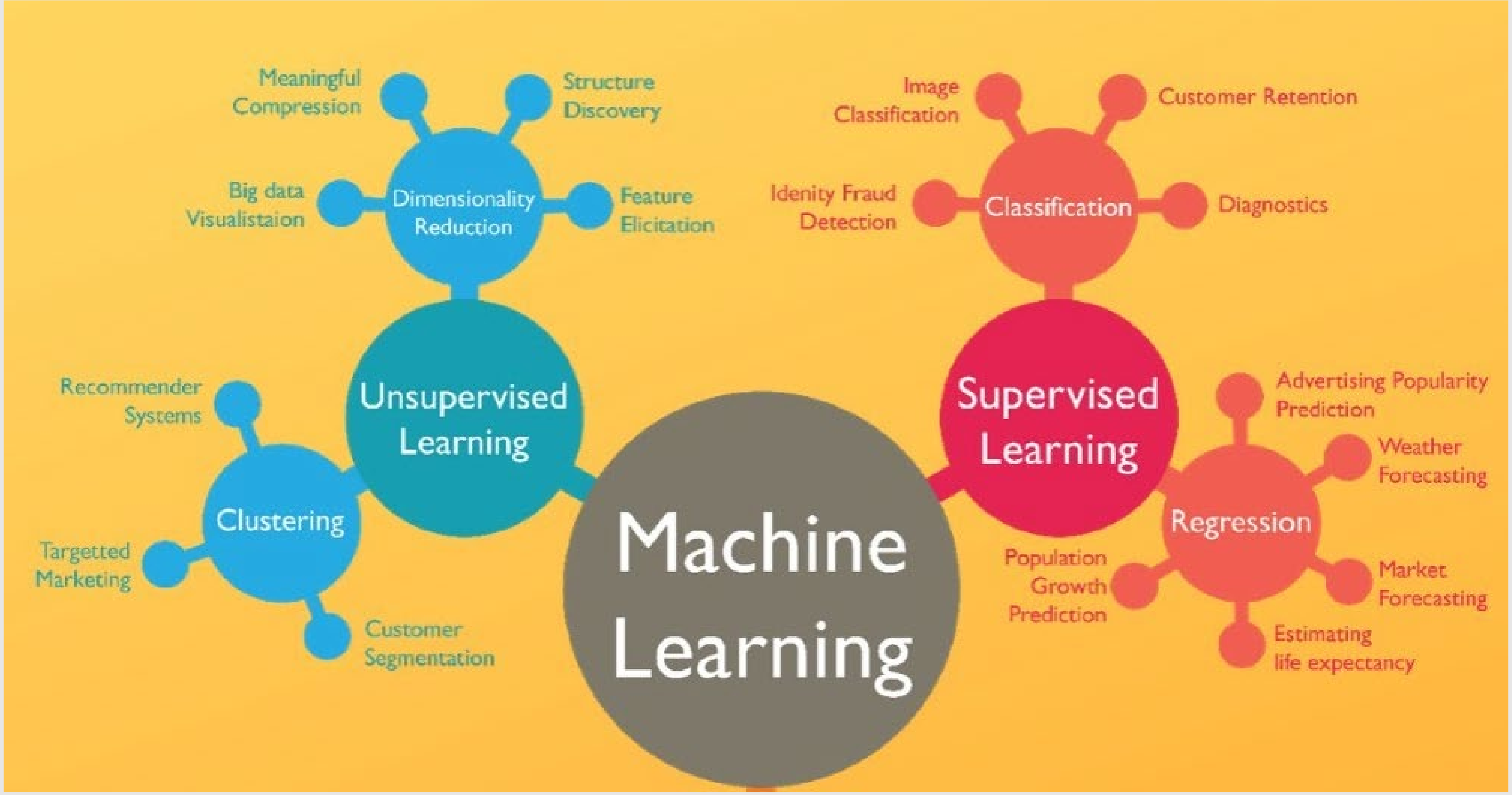
Week 8 Contents / Objectives

- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k-means to spectral
- Autoencoder

Week 8 Contents / Objectives

- **Why Unsupervised Learning?**
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k-means to spectral
- Autoencoder

Supervised vs Unsupervised



https://i.morioh.com/2020/04/14/ff897f322fed.jpg

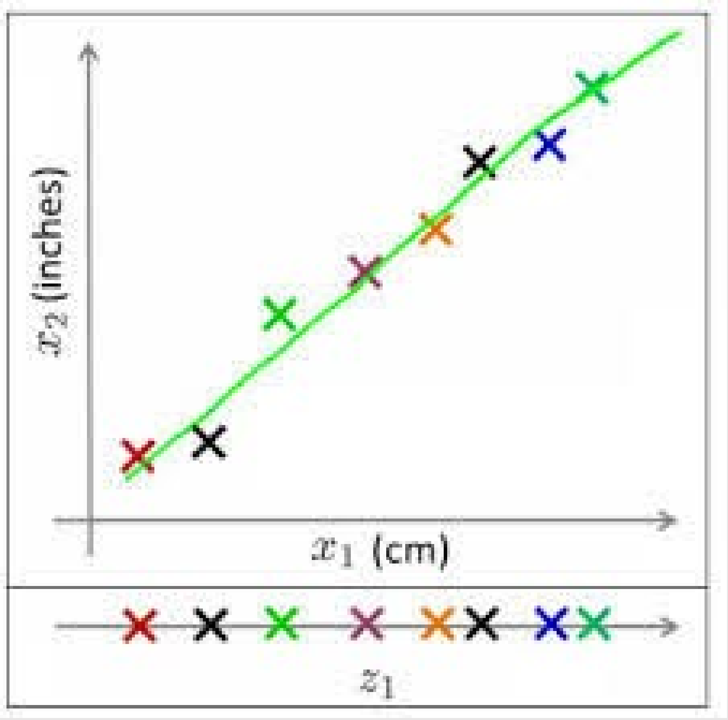
Unsupervised Learning

- Supervised learning: each data point has a label
- Unsupervised learning: no labels for the data
    - Dimensionality reduction
    - Clustering

Machine Learning	**Supervised**	**Unsupervised**
Discrete output	Classification	Clustering
Continuous output	Regression	Dimensionality Reduction

Dimensionality Reduction (DR)

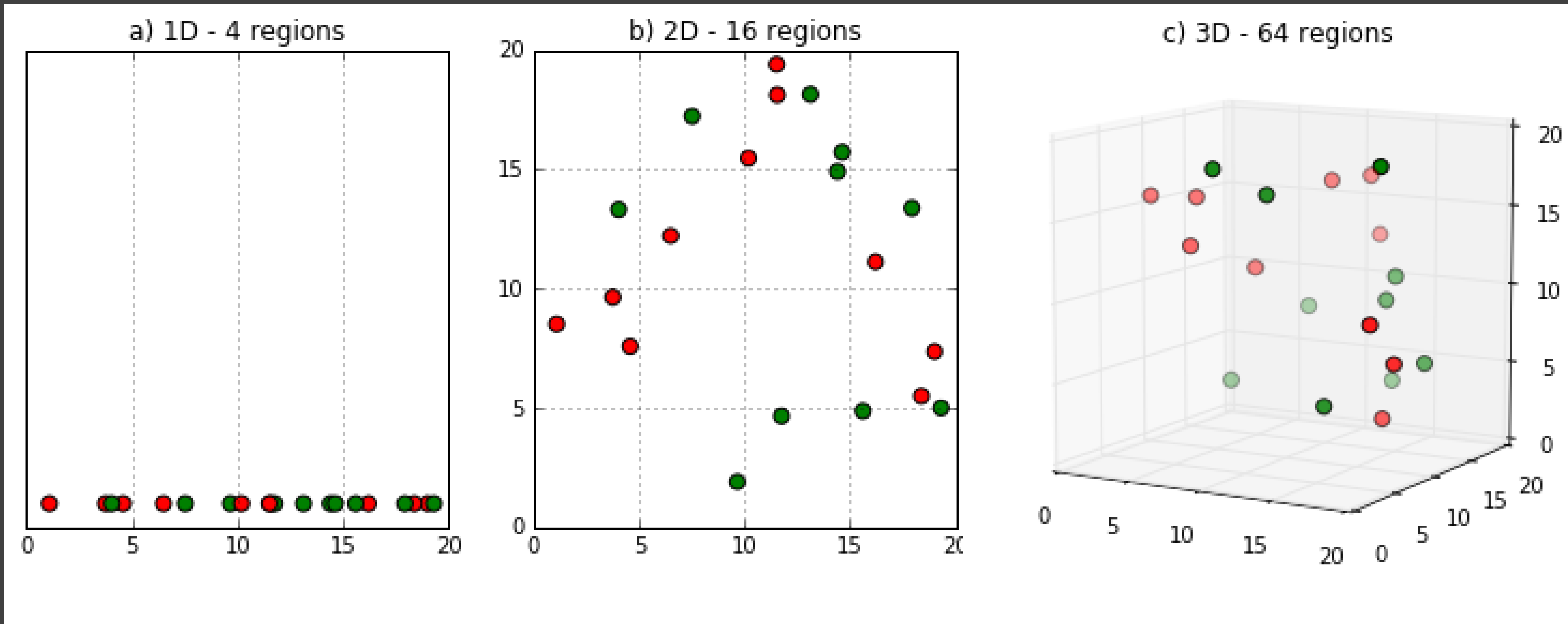
- 2 → 1



High Dimensional

Low Dimensional

Original data	Transformed
(1, 1.2)	1.15
(2, 2)	2
(3, 3.3)	3.1
…	…



Why DR?

High D → Low D

- **Curse of dimensionality**
  - **Nearest neighbours**
- Reduce redundancy (correlation)
- Visualisation

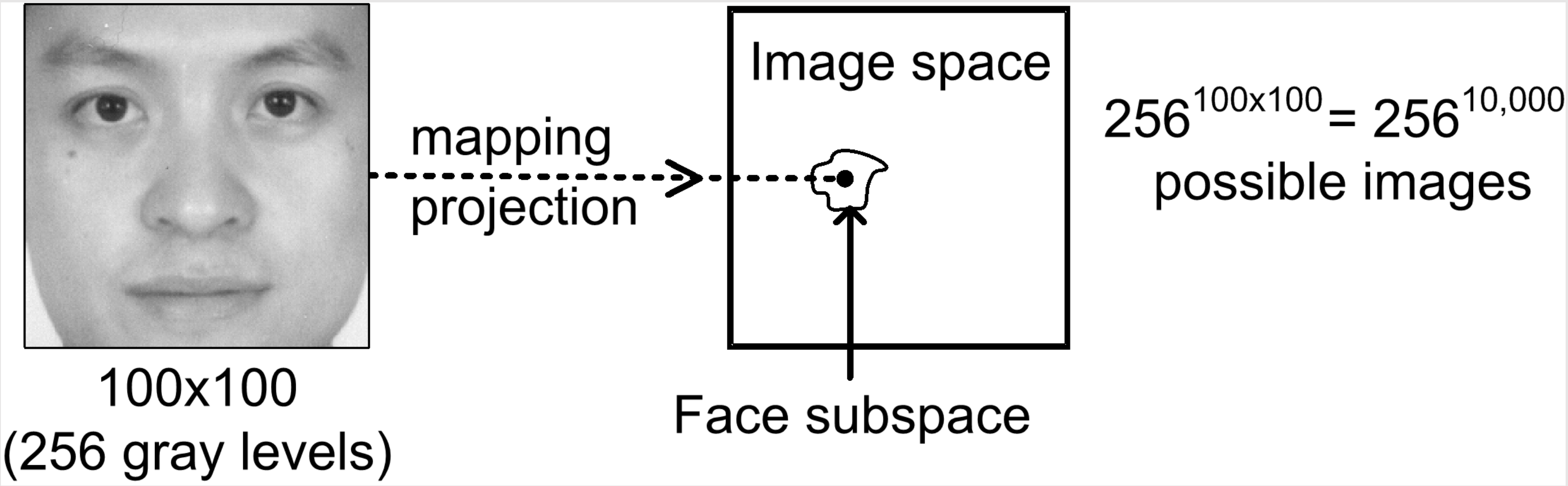
Question

- USPS dataset handwritten digit
- Size: 64 x 57; binary (1-bit, BW)
- The binary image space contains much more than just this digit.

How many possible images of this size and bit depth?

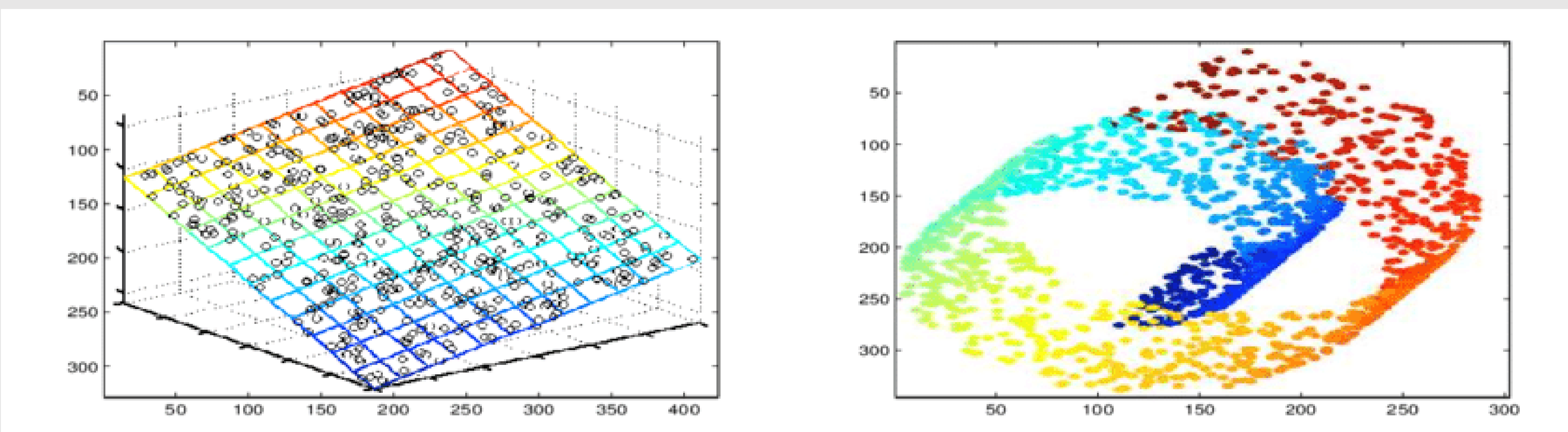
$$2^{64 \times 57} = 2^{3648} = ?$$

How About a Face?



Low-D Subspace/Manifolds

- For high dimensional data with **structure**:
  - Fewer variations than dimensions
  - Data to live on a lower dimensional manifold
  - → Deal with them by looking for a lower dimensional embedding (or projection, transformation)



Week 8 Contents / Objectives

- Why Unsupervised Learning?
- **Principal Component Analysis (PCA)**
- PCA Unboxing
- Clustering: from k-means to spectral
- Autoencoder

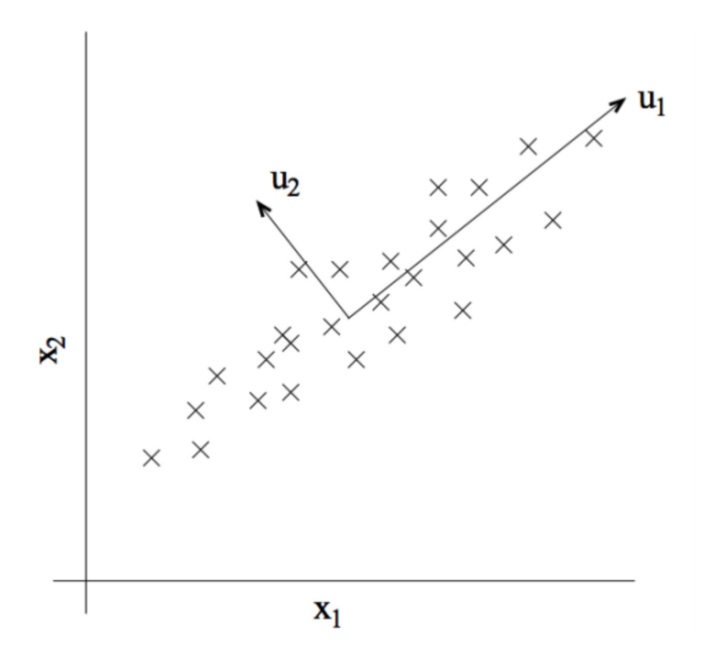
PCA?

Visualisation demo

Principal Component Analysis

The idea:

1. Rotate the data with some rotation matrix **R** (change of basis) so that the new features are **uncorrelated**
2. Keep the dimension with the highest **variance** for DR



Principal Component Analysis

- PCA (@Hotelling:analysis33): a linear embedding
- Rotate to find *directions* in data with **maximum variance**
- How do we find these directions?
  - Diagonalize the **sample covariance** (**scatter**) **matrix** of N samples $\{\mathbf{x}^{(i)}, i = 1, \cdots, N\}$

$$\mathbf{S} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right) \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^\top$$

Principal Component Analysis

- Given data $\{\mathbf{x}^{(i)}\}$, PCA finds **orthogonal** directions defined by a projection (rotation) $\mathbf{U}$ capturing the **maximum variance** in the data
- Solution: eigenvectors of $\mathbf{S}$ → $\mathbf{U}$
- PCA representation $\mathbf{y} = \mathbf{U}^\top \mathbf{x}$
- **Question**: Given the PCA representation $\mathbf{y}$, how to obtain an approximation/reconstruction of $\mathbf{x}$?

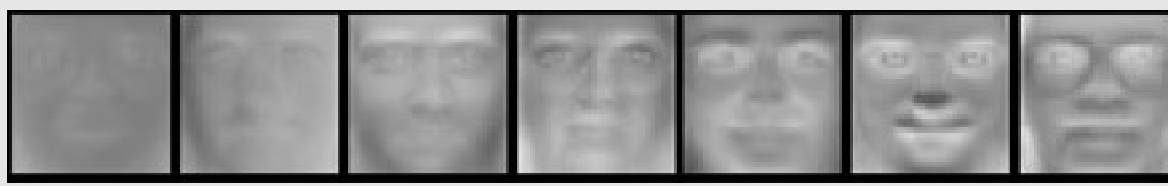
$$\hat{\mathbf{x}} = \mathbf{U}\mathbf{y}$$

Representation & Reconstruction

- Face $\mathbf{x}$ in k "face space" coordinates

$$\mathbf{x} \rightarrow [\mathbf{u}_1^\top(\mathbf{x}-\boldsymbol{\mu}), \cdots, \mathbf{u}_k^\top(\mathbf{x}-\boldsymbol{\mu})]$$
$$= [w_1, \cdots, w_k]$$

- Reconstruction: eigenvectors as orthonormal basis vectors

 =  + 

$$\hat{\mathbf{x}} \quad = \quad \boldsymbol{\mu} \quad + \quad w_1\mathbf{u}_1 + w_2\mathbf{u}_2 + w_3\mathbf{u}_3 + w_4\mathbf{u}_4 + \cdots$$

slide by Derek Hoiem

Reconstruction

$k = 4$

$k = 200$

$k = 400$

After computing eigenfaces using 400 face images from the ORL face database

slide by Derek Hoiem

PCA Ingredients

- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: linear projection $\mathbf{y} = \mathbf{U}^\top \mathbf{x}$
  - **Hyper-parameter**: lower dimension k
  - Parameters (theta): the principal components (eigenvectors)
- Evaluation metric: max variance
- Optimisation: eigen-decomposition

Week 8 Contents / Objectives

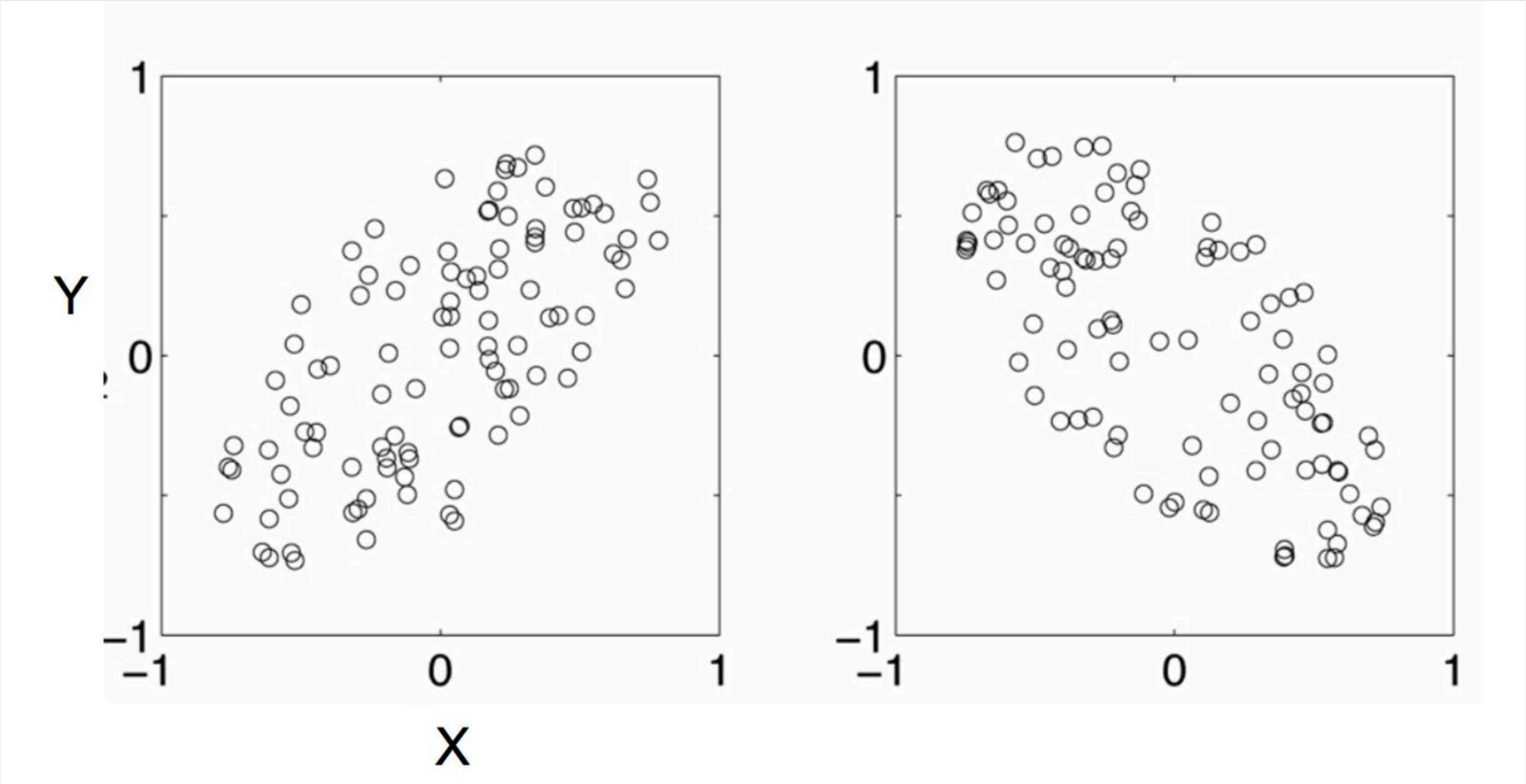
- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- **PCA Unboxing**
- Clustering: from k-means to spectral
- Autoencoder

Variance →Covariance (Matrix)

- Variance and covariance:
  - Measure of the “spread” of a set of points around their *center of mass* (mean)
- Variance (scalar):
  - Measure of the deviation from the mean for points in **one dimension**
- Covariance (**matrix**):
  - Measure of how much each of the dimensions vary from the mean with **respect to each other**

- Covariance is measured between two dimensions
- Covariance sees if there is a **relation** between the two dimensions
- Covariance between one dimension is the variance (degeneration)

Positive/Negative Covariance



Positive: Both dimensions increase or decrease together

Negative: While one increase the other decrease

Covariance

- Find relationships between dimensions in high dimensional data sets $\mathbf{X}$
- Scatter matrix $\mathbf{S}$: sample-based estimation of covariance matrix (i: sample; j/k: variable)

$$s_{jk} = \frac{1}{N}\sum_{i=1}^{N}(X_{ij} - \underbrace{E(X_j)}_{\text{The Sample mean}})(X_{ik} - E(X_k))$$

- Uncorrelated variables → covariance = 0
- Diagonal covariance mat → all variables are uncorrelated

PCA Derivation – Max Variance

- Scatter mat for the input $\mathbf{S} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}\right)^{\top}$
- **Question**: what is the scatter mat for the projections?

$$\mathbf{U}^{\top}\mathbf{S}\mathbf{U}$$

- First PC: **maximise the variance** in the **projected** space, i.e. the variance of $y = \mathbf{u}_1^{\top}\mathbf{x}$

$$\begin{aligned}\mathrm{var}(y) &= \frac{1}{N}\sum_{i=1}^{N}\left(y^{(i)} - \mu_y\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_1^{\top}\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{\mathbf{x}}\right)\left(\mathbf{x}^{(i)} - \boldsymbol{\mu}_{\mathbf{x}}\right)^{\top}\mathbf{u}_1 \\ &= \mathbf{u}_1^{\top}\mathbf{S}\mathbf{u}_1\end{aligned}$$

PCA: Max Variance → Eigenvalue

- Find the first direction $\mathbf{u}_1$ via a unit-norm constrained optimisation, using Lagrange multipliers:

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1 \left(1 - \mathbf{u}_1^\top \mathbf{u}_1\right)$$

- Gradient w.r.t. $\mathbf{u}_1$: $$\frac{\mathrm{d}L(\mathbf{u}_1, \lambda_1)}{\mathrm{d}\mathbf{u}_1} = 2\mathbf{S}\mathbf{u}_1 - 2\lambda_1 \mathbf{u}_1$$
- Set to 0 and rearrange: $\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$ → First PC
  - Eigenvalue problem
- **Question**: many solutions (eigen-pairs), which to choose?

$$\mathrm{var}(y) = \mathbf{u}^\top \mathbf{S}\mathbf{u} = \lambda \mathbf{u}^\top \mathbf{u} = \lambda$$

PCA Solution

- Further directions: **orthogonal** (uncorrelated) to the first and each others → top k eigenvectors of $\mathbf{S}$
  - Eigenvectors: basis functions, principal components
  - Eigenvalue: the **variance** captured respectively
- **Questions**
  - For $\mathbf{u}_1$, what if we do not have the unit-norm constraint?

$$L(\mathbf{u}_1, \lambda_1) = \mathbf{u}_1^\top \mathbf{S} \mathbf{u}_1 + \cancel{\lambda_1 \left(1 - \mathbf{u}_1^\top \mathbf{u}_1\right)}$$

→Trivial solution: infinity

  - For further directions: what if we do not require them to be orthogonal?

→ We will have the same $\mathbf{u}_1$, useless solution

PCA: Max Variance ←→ Min MSE

Consider the first PC with the projection vector **u**. We assume **zero-mean**, i.e. *centered* data

Maximum Variance Direction: 1st PC = a vector **u** such that projection on it captures max variance in the data

$$\frac{1}{N}\sum_{i=1}^{N}(\mathbf{u}^T\mathbf{x}^{(i)})^2 = \mathbf{u}^T\mathbf{X}\mathbf{X}^T\mathbf{u}$$

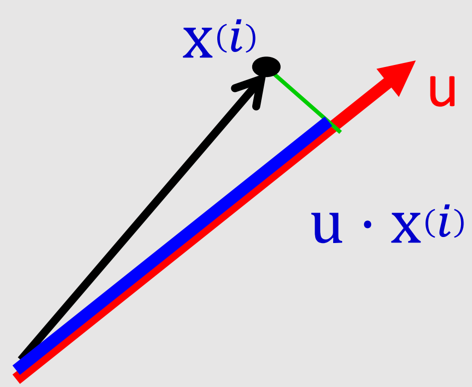
Minimum Reconstruction Error: 1st PC = a vector **u** such that projection on it yields minimum MSE reconstruction

$$\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)} - (\mathbf{u}^T\mathbf{x}^{(i)})\mathbf{u}\|^2$$

blue^2 + green^2 = black^2

black^2 is fixed (it's just the data)

So, maximizing blue^2 is equivalent to minimizing green^2

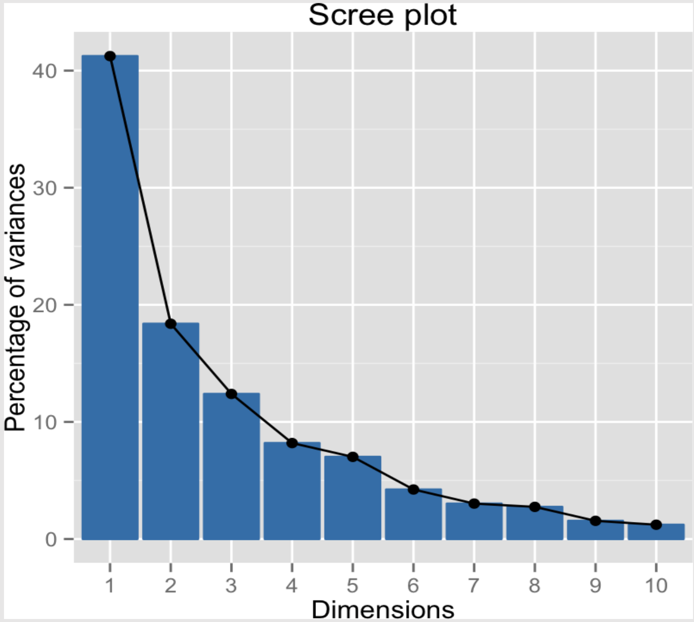


How many (k) PCs to keep?

- Pick based on percentage of variance captured / lost
  - Variance captured: the variance of the projected data
  - Pick smallest k that explains a certain percentage of variance

$$(\text{Sum of first } k \text{ EVs})/(\text{sum of all EVs})$$

- Look for an "elbow" in scree plot (plot of explained variance or eigenvalues)

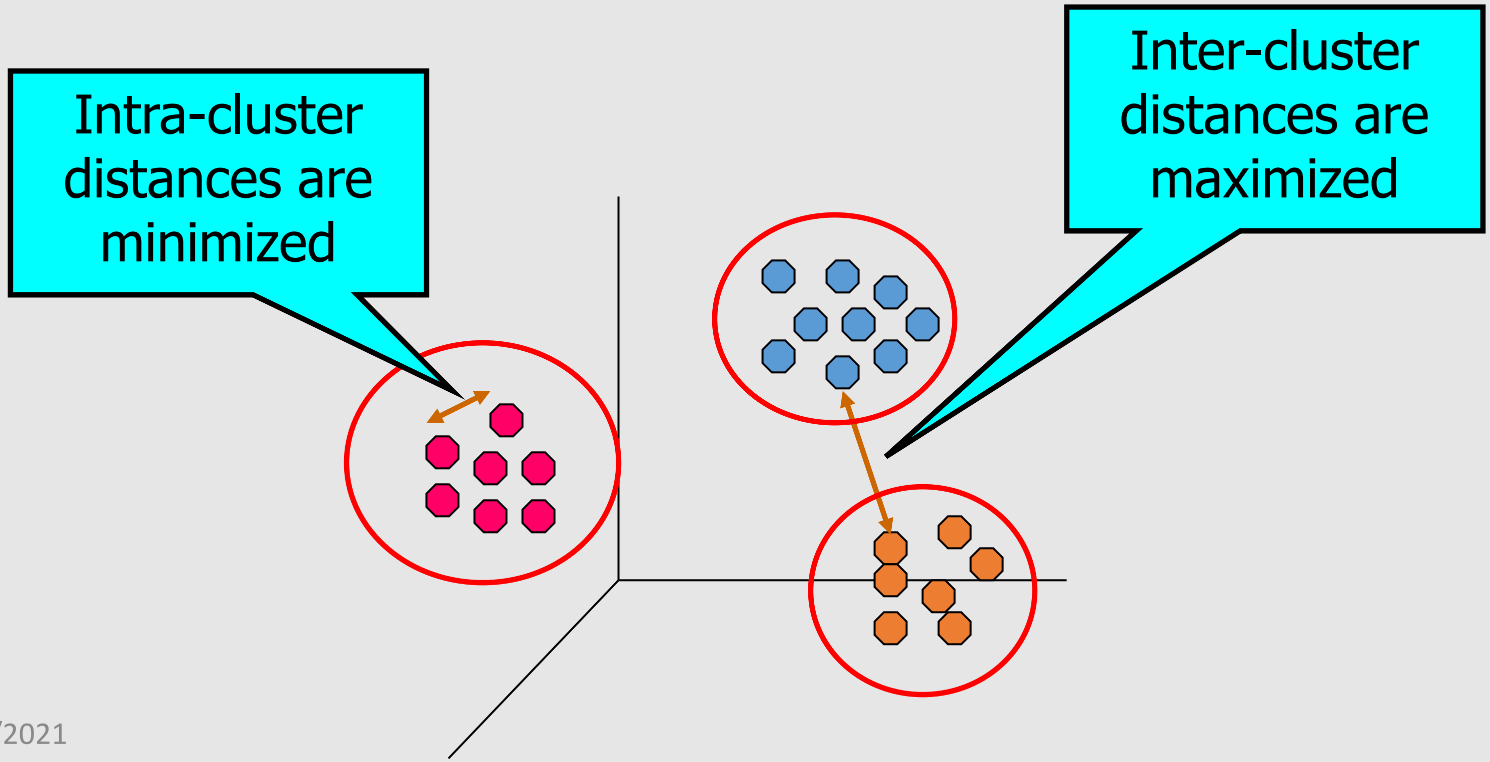


Week 8 Contents / Objectives

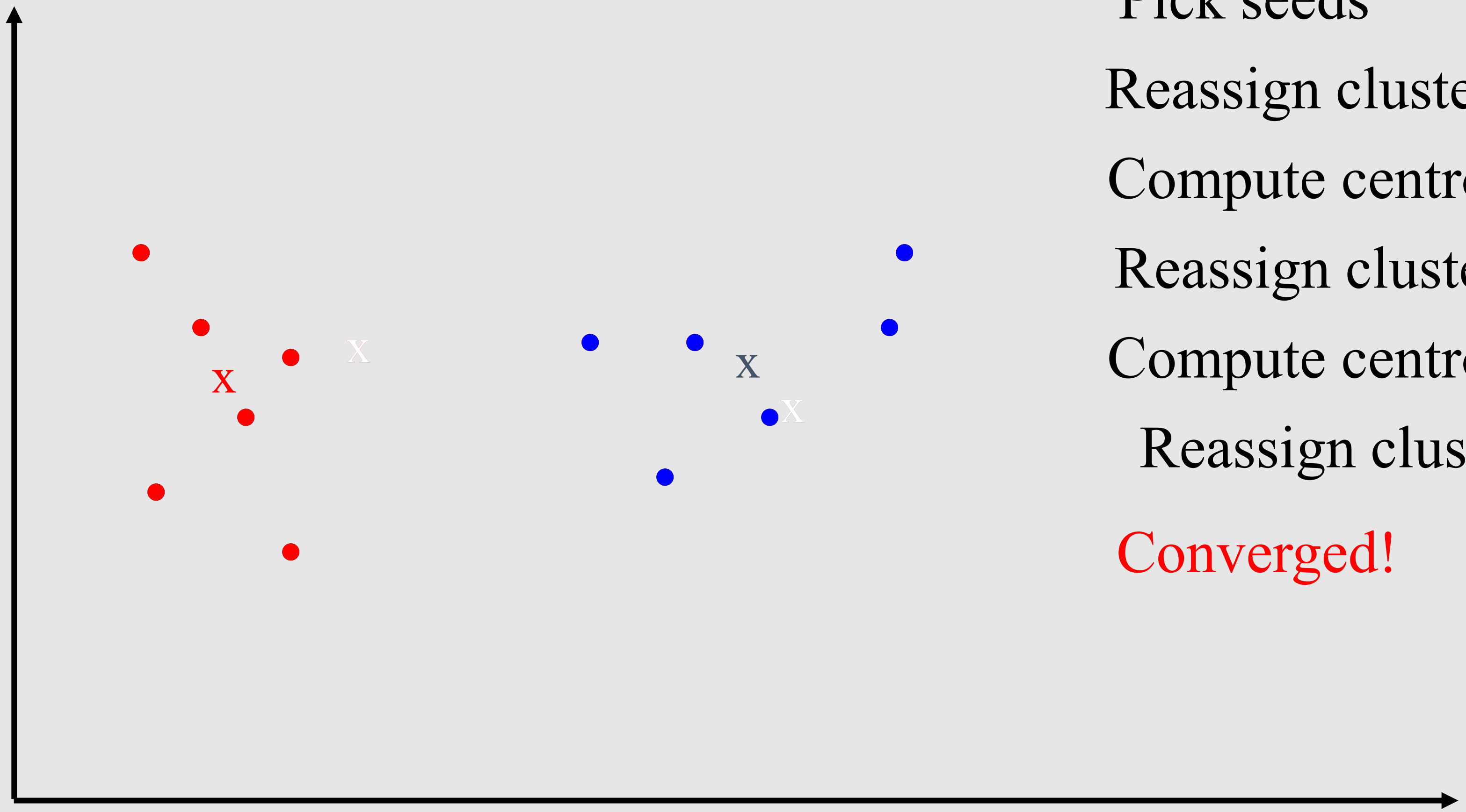
- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- **Clustering: from k-means to spectral**
- Autoencoder

What is Clustering?

- Find grouping of objects such that the objects in a group will be similar or more related to one another and those in different groups will be different or less related



k-means Example (*k*=2)



Pick seeds

Reassign clusters

Compute centroids

Reassign clusters

Compute centroids

Reassign clusters

Converged!

k-means & PCA

Machine Learning	Supervised	Unsupervised
Discrete output	Classification	Clustering
Continuous output	Regression	Dimensionality Reduction

- The objective of k-means clustering: to minimize the within-cluster scatter

$$\min \sum_{j=1}^{k} \sum_{i \text{ allocated to } j} \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right)^\top \left(\mathbf{x}^{(i)} - \boldsymbol{\mu}^{(j)}\right)$$

- Similarity with PCA: also measure of the "spread" of a set of points around their *center of mass* (mean)
- Clustering analogy
  - Classification w/o labelled training data
  - Extreme dimensionality reduction (to a **cluster label**)

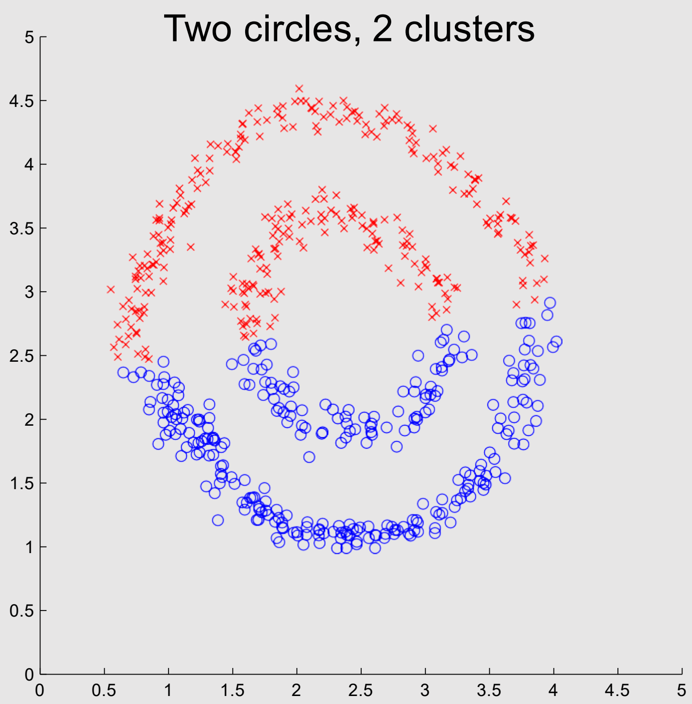
k-means Clustering Ingredients

- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: linear separation btw clusters
  - **Hyper-parameter**: #clusters k
  - Parameters (theta): the k cluster centroids
- Evaluation metric: within-cluster scatter
- Optimisation: expectation maximisation (EM), kind of gradient descent

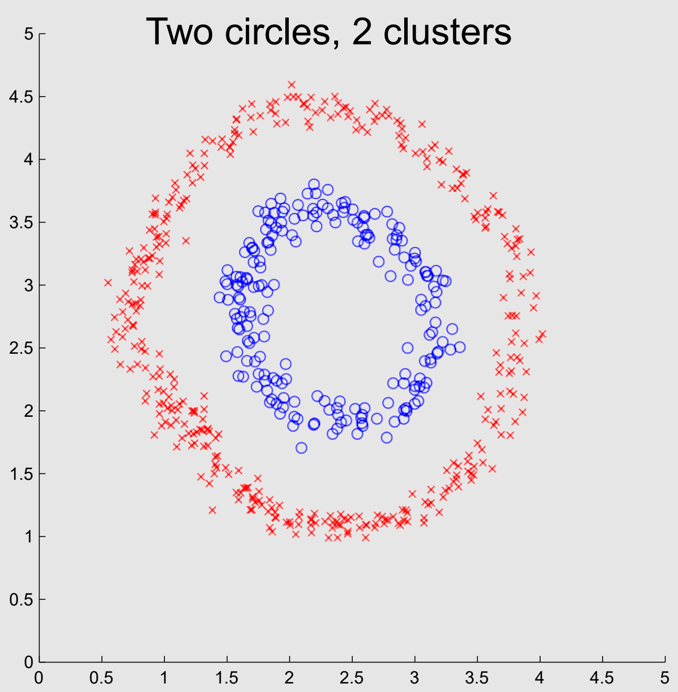
k-means Has a Problem

- k-means: centroid-based, for compactness (scatter)
- Spectral clustering: **connectivity**

k-means



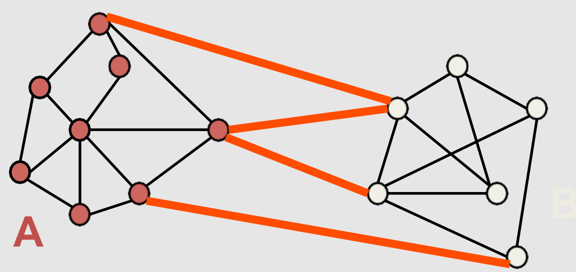
Spectral clustering

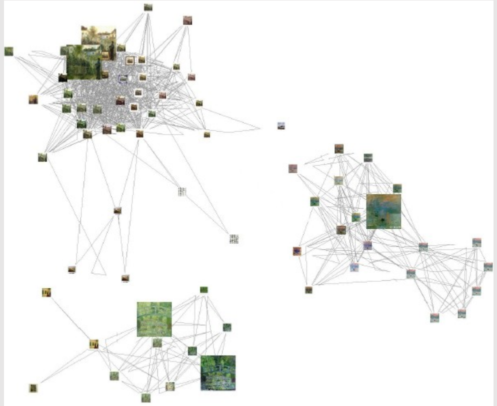


[Shi & Malik ‘00; Ng, Jordan, Weiss NeurIPS ‘01]

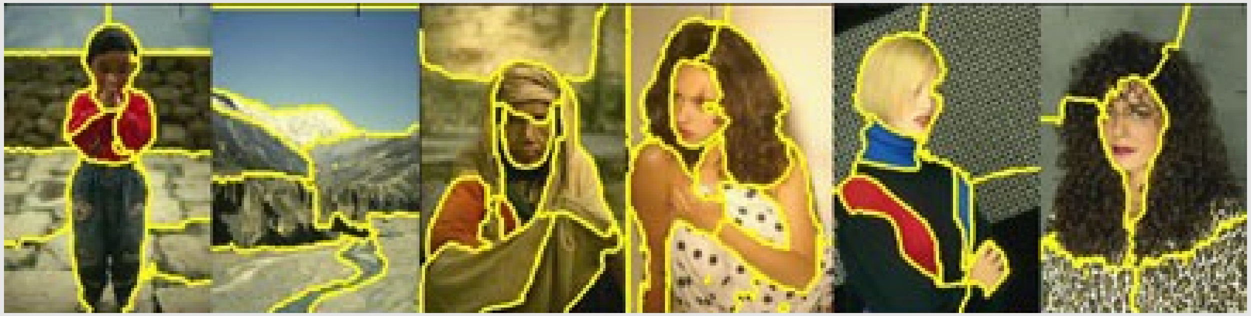
Spectral Clustering

- Group points based on **links** in a **graph**





- Image as graph: segmentation as clustering



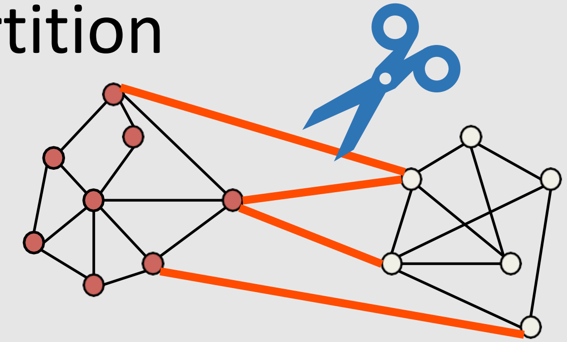
Slide from James Hays

How to Create the Graph?

- Gaussian kernel→ compute similarity between objects

$$\mathbf{W}(i,j) = \exp \frac{-|x_i - x_j|^2}{\sigma^2}$$

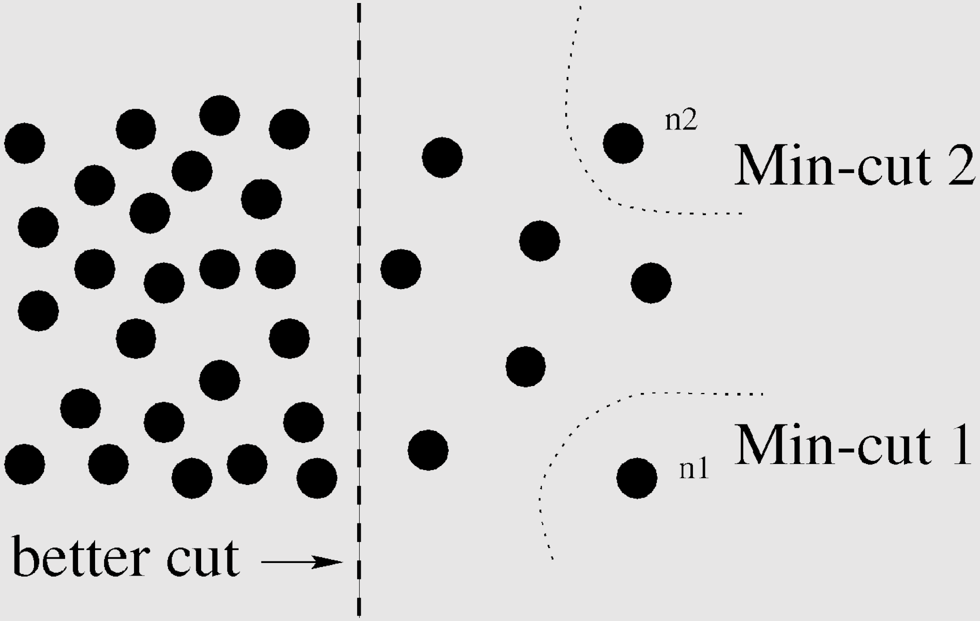
- One could create
  - A fully connected graph (~ FC layer)
  - K-nearest neighbour graph: each node is only connected to its K-nearest neighbours (~ convolutional layer, local connectivity)
- Clustering → Graph cut/partition
  - Objective: **minimise** cut



Slide from Alan Fern

Min Cut = Good Cut?

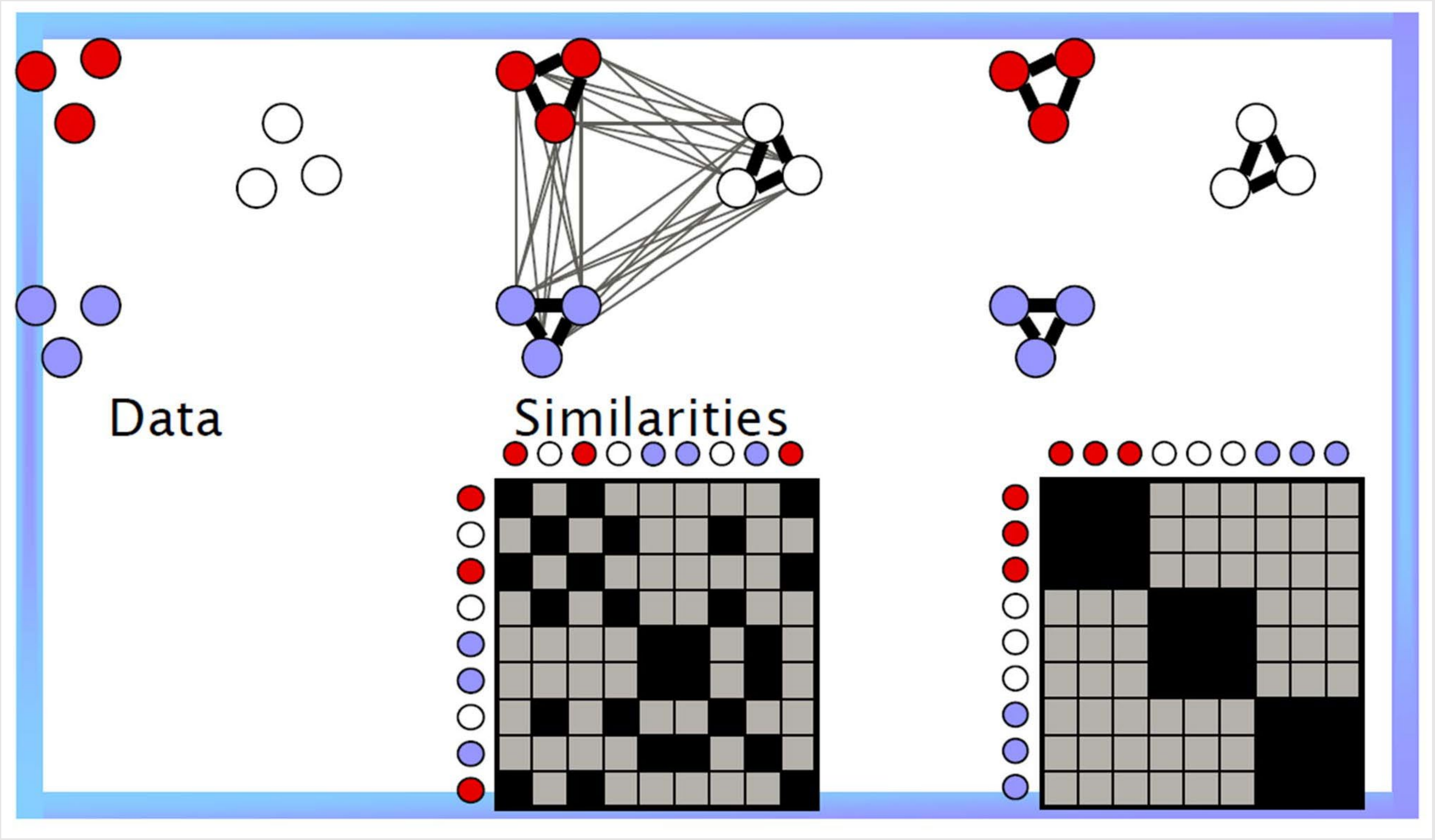
- A case where minimum cut gives a bad partition



- Solution: **Normalise** the cut

[Shi & Malik ‘00]

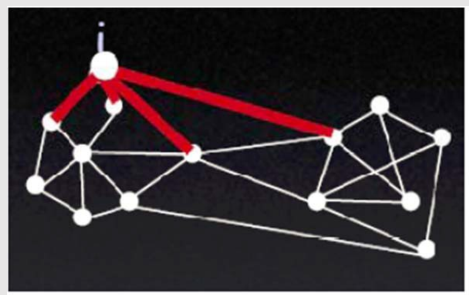
Graph Clustering Process

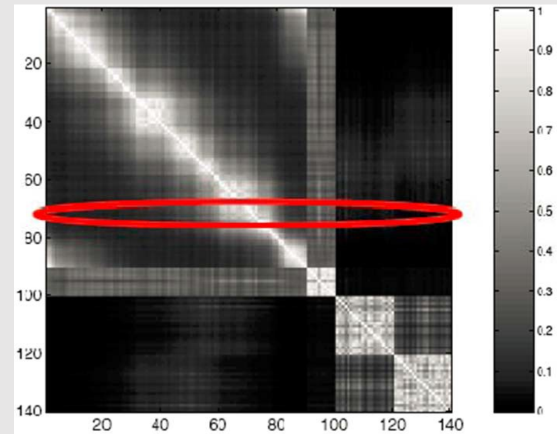


Graph Terminologies

- Degree of nodes

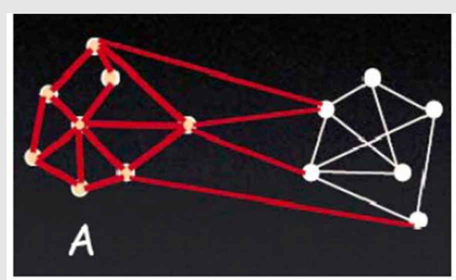
$$d_i = \sum_j w_{i,j}$$

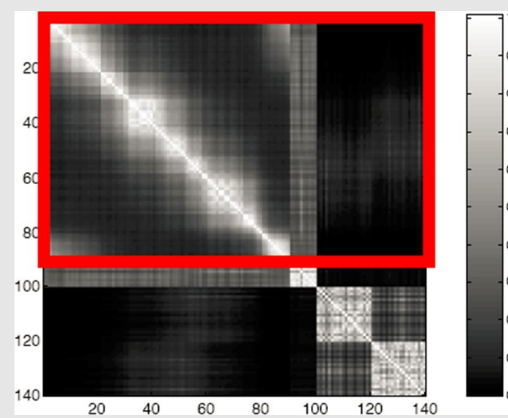




- Volume of a set

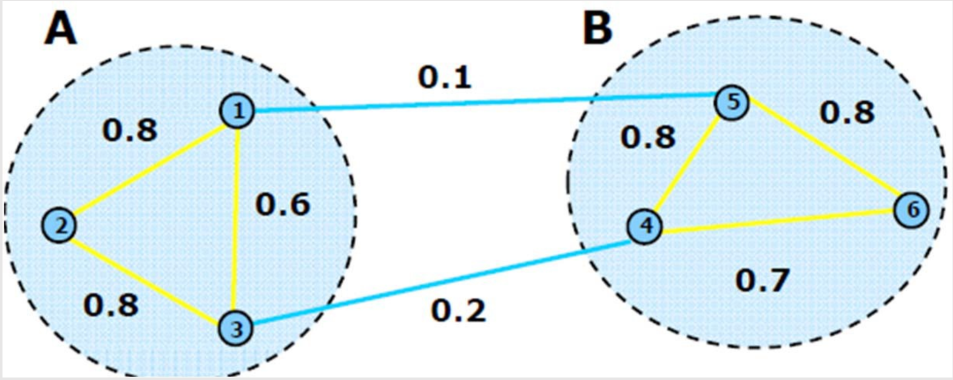
$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





Graph Cut

- Consider a partition of the graph into two parts *A* & *B*



Question

$$cut(A, B) =$$

- $cut(A, B)$: sum of the weights of the set of edges that connect the two groups

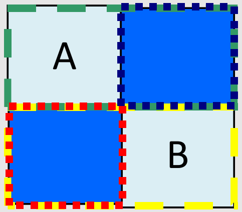
$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$

- Goal: find the partition that minimizes the cut

Normalized Cut (Ncut)

- Consider the **connectivity** between groups **relative** to the volume of each group

$$Ncut(A,B) = \frac{cut(A,B)}{Vol(A)} + \frac{cut(A,B)}{Vol(B)}$$



$$Ncut(A,B) = cut(A,B)\frac{Vol(A)+Vol(B)}{Vol(A)Vol(B)}$$

\usepackage{mathtools}

Solving/Minimising Ncut

- Compute the similarity matrix $\mathbf{W} : \mathbf{W}(i, j) = w_{i,j}$
- Compute the degree matrix $\mathbf{D} : \mathbf{D}(i, i) = \sum_j w_{i,j}$
- Solve a generalised **eigenvalue** problem (relaxed Ncut)

$$\min_{\mathbf{y}} \mathbf{y}^\top (\mathbf{D} - \mathbf{W})\mathbf{y} \text{ s.t. } \mathbf{y}^\top \mathbf{D}\mathbf{y} = 1 \Rightarrow (\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

(D-W) : Laplacian matrix

- Solution:
  - Bipartition: use the eigenvector with the second smallest eigenvalue to partition the graph into two parts
    - Splitting point: minimum Ncut (plot).
  - K-way partition: k-means clustering of multiple eigenvectors
    - Graph embedding (dimensionality reduction) → eigenvectors

Recap: Power of Transform

- Logistic regression **transforms** classification into linear regression of the log odds, modelling the probability of the predicted output rather than the output itself.
- Spectral clustering **transforms** non-linear clustering into (linear) k-means clustering of generalised eigenvectors based on the similarity graph, modelling the connectivity of data points rather than themselves.

https://i.kym-cdn.com/photos/images/newsfeed/001/030/448/b9f.gif

Spectral Clustering Ingredients

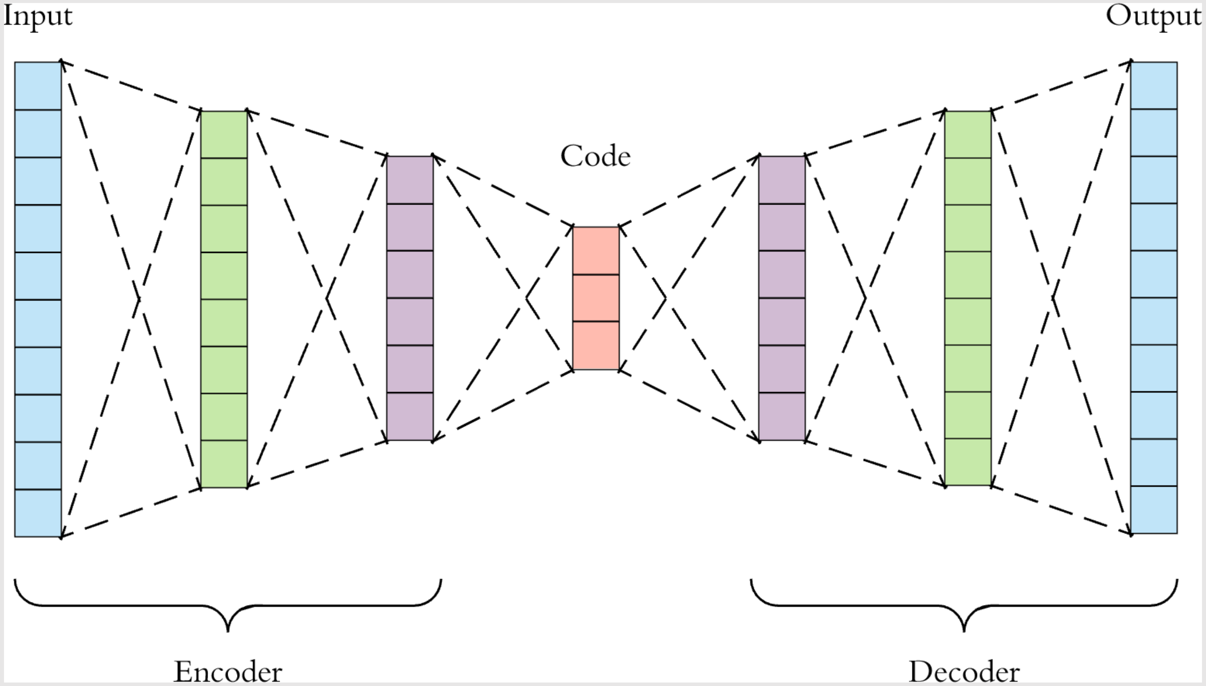
- **Data**: +pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: spectral (nonlinear) separation btw clusters
  - **Hyper-parameter**: #clusters k (+#eigenvectors), kernel bandwidth σ
  - Parameters (theta): the (generalised) eigenvectors
- Evaluation metric: normalised (graph) cut
- Optimisation: eigen-decomposition (and expectation maximisation in k-means)

Week 8 Contents / Objectives

- Why Unsupervised Learning?
- Principal Component Analysis (PCA)
- PCA Unboxing
- Clustering: from k-means to spectral
- **Autoencoder**

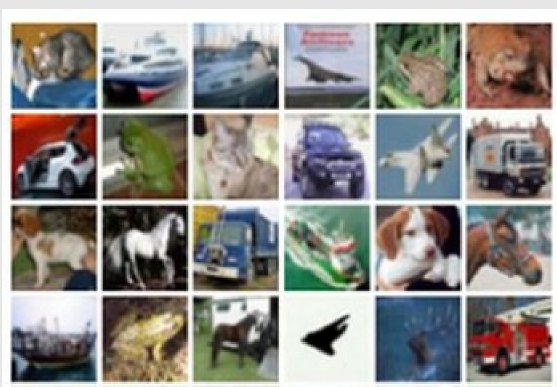
Autoencoder

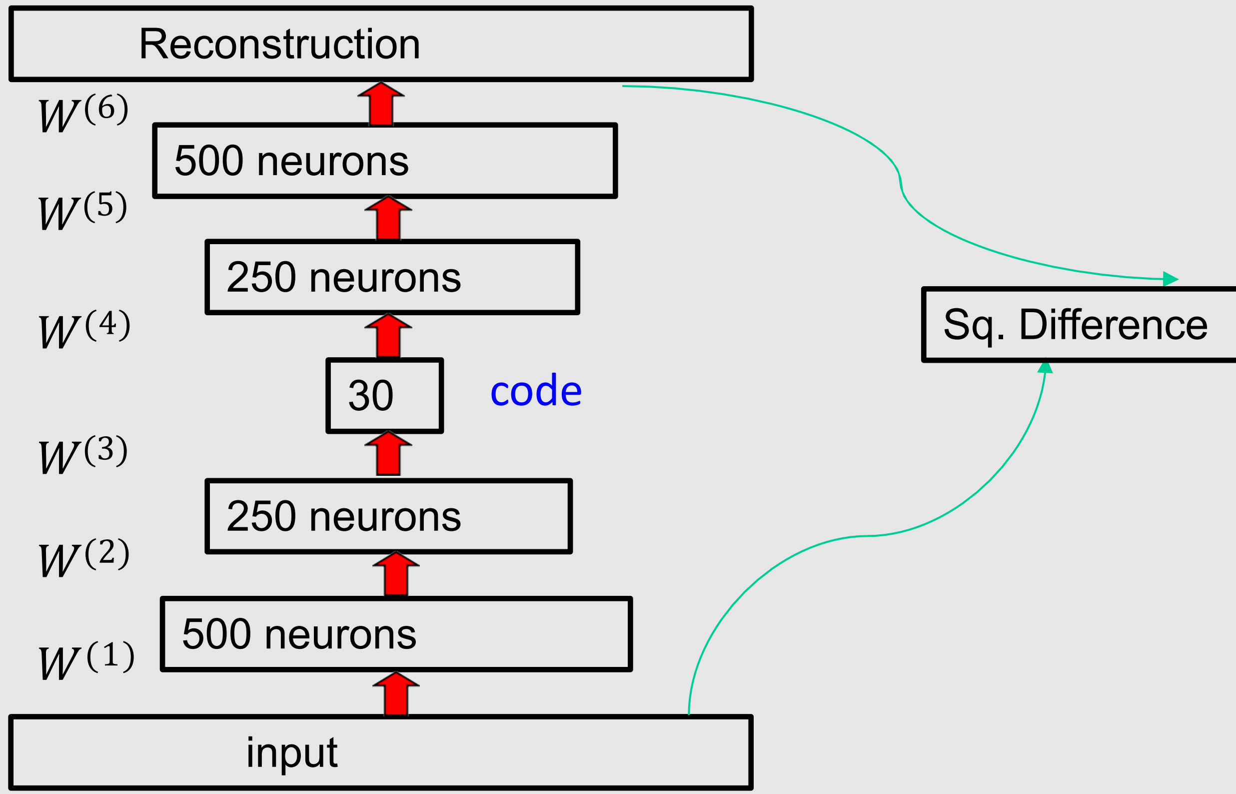
- **Encoder:** compress data or extract features
- **Decoder:** generate images given a new code
- **Bottleneck** (**code**): to make it non-trivial, much smaller dimension as the latent representation



https://miro.medium.com/max/3148/1*44eDEuZBEsmG_TCAKRI3Kw@2x.png

Autoencoder Example

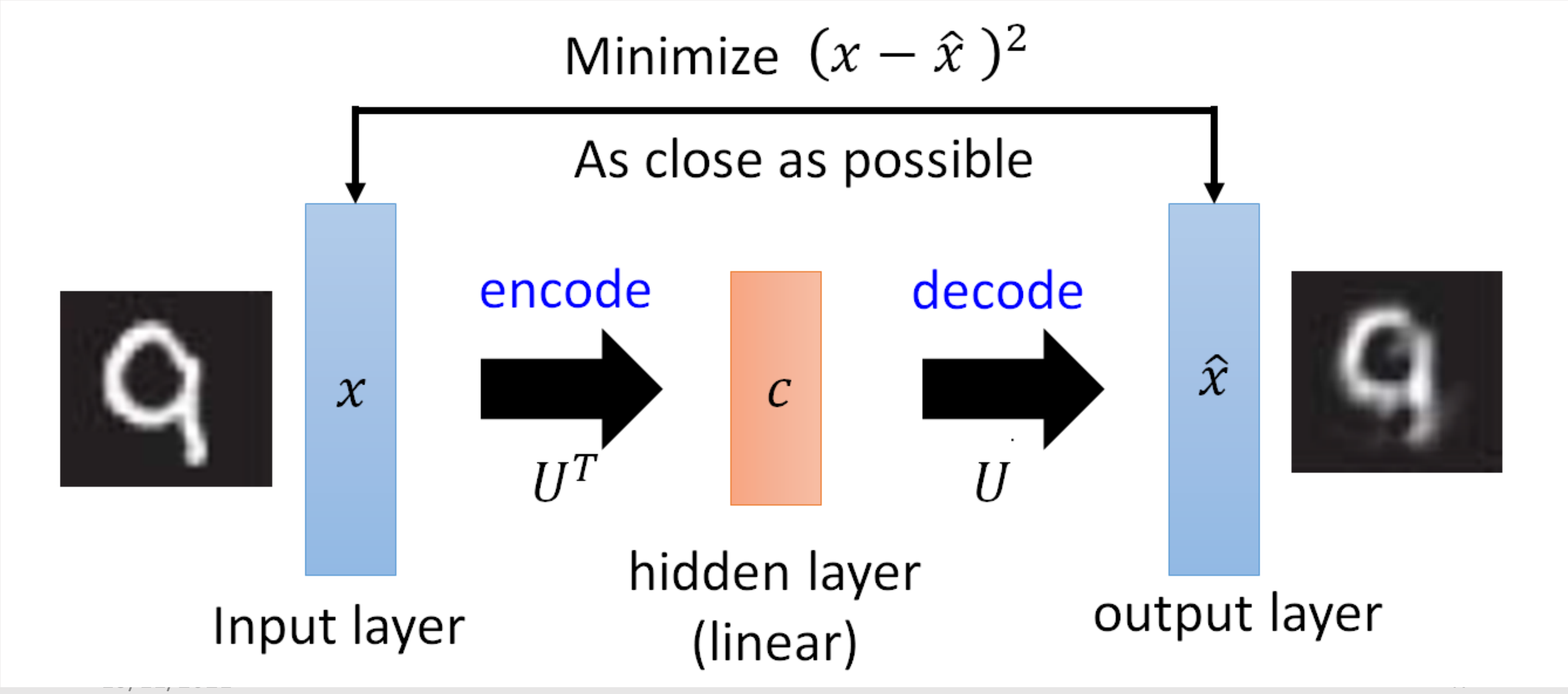




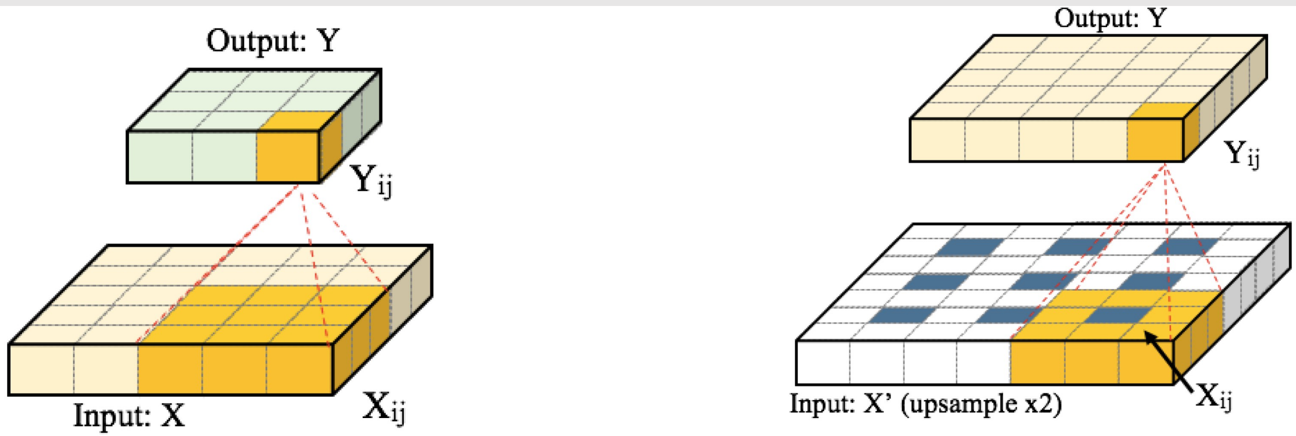
- Find the weights minimising the difference between the input and reconstruction
- The code layer (bottleneck) is a low--dimensional summary of the input

PCA as Linear Autoencoder

- PCA: autoencoder w/t single-layer encoder/decoder
- Weight sharing between the encoder and decoder



Transpose Convolution Layer



(**a**) Convolutional layer: the input size is $W_1 = H_1 = 5$; the receptive field $F = 3$; the convolution is performed with stride $S = 1$ and no padding ($P = 0$). The output Yis of size $W_2 = H_2 = 3$.

(**b**) Transposed convolutional layer: input size $W_1 = H_1 = 3$; transposed convolution with stride $S = 2$; padding with $P = 1$; and a receptive field of $F = 3$. The output Yis of size $W_2 = H_2 = 5$.

https://www.mdpi.com/2072-4292/9/6/522/htm

More at https://github.com/vdumoulin/conv_arithmetic

Convolutional Autoencoder (Lab)

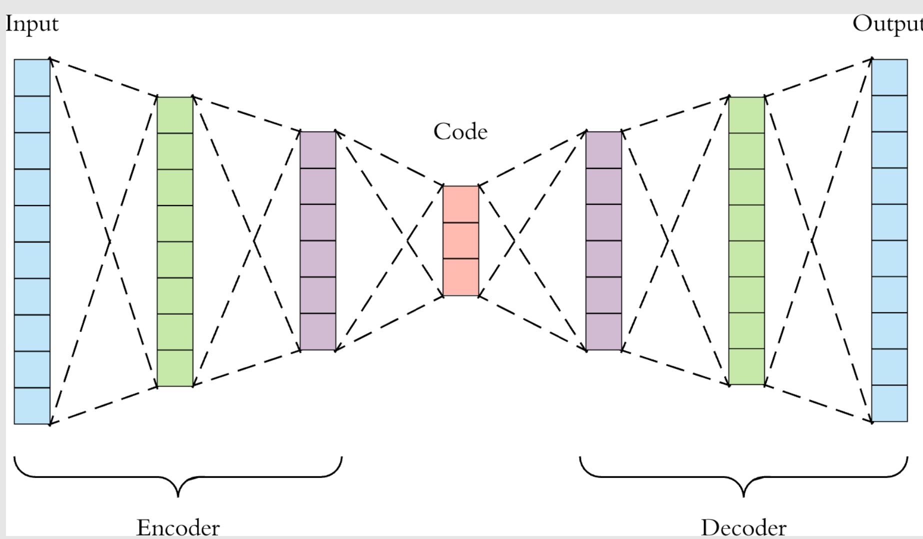
```python
class Autoencoder(nn.Module):
    def __init__(self):
        super(Autoencoder, self).__init__()
        self.encoder = nn.Sequential(
            # 1 input image channel, 16 output channel, 3x3 square convolution
            nn.Conv2d(1, 16, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(16, 32, 3, stride=2, padding=1),
            nn.ReLU(),
            nn.Conv2d(32, 64, 7)
        )
        self.decoder = nn.Sequential(
            nn.ConvTranspose2d(64, 32, 7),
            nn.ReLU(),
            nn.ConvTranspose2d(32, 16, 3, stride=2, padding=1, output_padding=1),
            nn.ReLU(),
            nn.ConvTranspose2d(16, 1, 3, stride=2, padding=1, output_padding=1),
            nn.Sigmoid()  #to range [0, 1]
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

Training

Sample labeled data (**batch**) → **Forward** it through the network, get predictions → **Back-propagate** the errors → **Update** the network weights

Data → Model → Metric →Optimisation



Autoencoder Ingredients

- **Data**: + pre-processing, e.g., $\mathcal{N}(0,1)$
- **Model**
  - Structure/Architecture: layers defined in nn.module
  - **Hyper-parameter**: layer specs, e.g. #layers #channels, kernel size
  - Parameters (theta): layer weights and biases
- Evaluation metric: MSE or other
- Optimisation: backprop, SGD or the like

Acknowledgement

- The slides used materials from: *Lisa Zhang, Michael Guerzhoy, Neil Lawrence, Derek Hoiem, Pandu Nayak, Prabhakar Raghavan, Fereshteh Sadeghi, Aarti Singh, David Sontag, James Hays, Alan Fern, Tommi Jaakkola, Jure Leskovec*

Recommended Reading

- The PCA book (from a UIUC link)
- Chapter on PCA in most machine learning books
- Chapter on clustering in most machine learning books
- The normalized cut paper in 2000
- Chapter on autoencoder in the Deep Learning Book

- Wikipedia entries on covered topics
- Scikit-learn/PyTorch documentations
- The lab notebook and references