

10-701: Introduction to Machine Learning

# Lecture 15 – Unsupervised Learning: Dimensionality Reduction

Hoda Heidari

\* Slides adopted from F24 offering of 10701 by Henry Chai.

# Recipe for $K$ -means

- Define a model and model parameters
  - Assume  $K$  clusters and use the Euclidean distance
  - Parameters:  $\mu_1, \dots, \mu_K$  and  $z^{(1)}, \dots, z^{(N)}$
- Write down an objective function
  - within-cluster sum of squares (WCSS)

$$\sum_{i=1}^N \|x^{(i)} - \mu_{z^{(i)}}\|_2$$

- Optimize the objective w.r.t. the model parameters
  - Use (block) coordinate descent

# K-means Algorithm

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(i)})\}_{i=1}^N, K$
- 1. Initialize cluster centers  $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$
- 2. While NOT CONVERGED
  - a. Assign each data point to the cluster with the nearest cluster center:

$$z^{(i)} = \operatorname{argmin}_k \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|_2$$

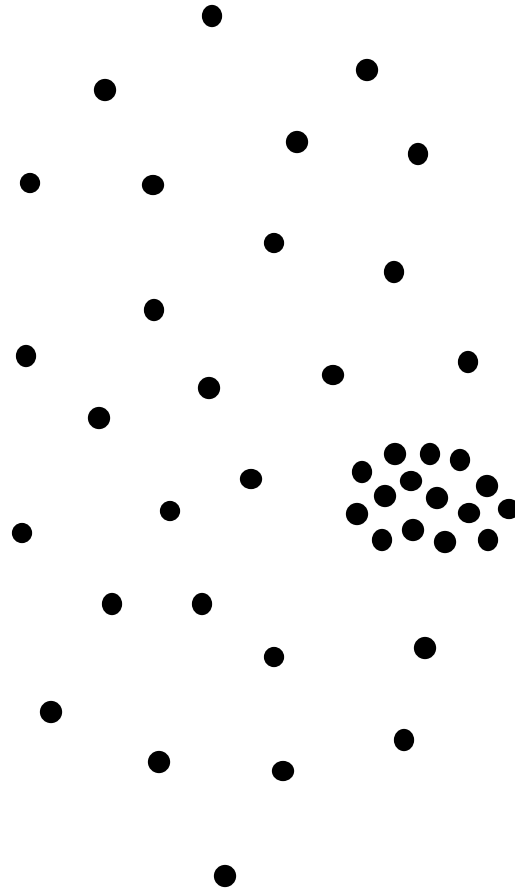
- b. Recompute the cluster centers:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i: z^{(i)}=k} \mathbf{x}^{(i)}$$

where  $N_k$  is the number of data points in cluster  $k$

- Output: cluster assignments  $z^{(1)}, \dots, z^{(N)}$

# Shortcomings of $K$ -means



- Clusters cannot overlap
- Clusters must all be of the same “width”
- Clusters must be linearly separable

# Probabilistic or “Soft” Assignments

- Instead of  $z^{(i)}$  being a deterministic scalar, let  $\mathbf{z}^{(i)}$  be a 1-of- $K$  vector indicating cluster membership
  - For example,  $\mathbf{z}^{(1)} = [0, 1, 0, \dots, 0]$  indicates that the first data point belongs to the second cluster
- Let  $\pi_k := p\left(z_k^{(i)} = 1\right)$

# Gaussian Mixture Models (GMMs)

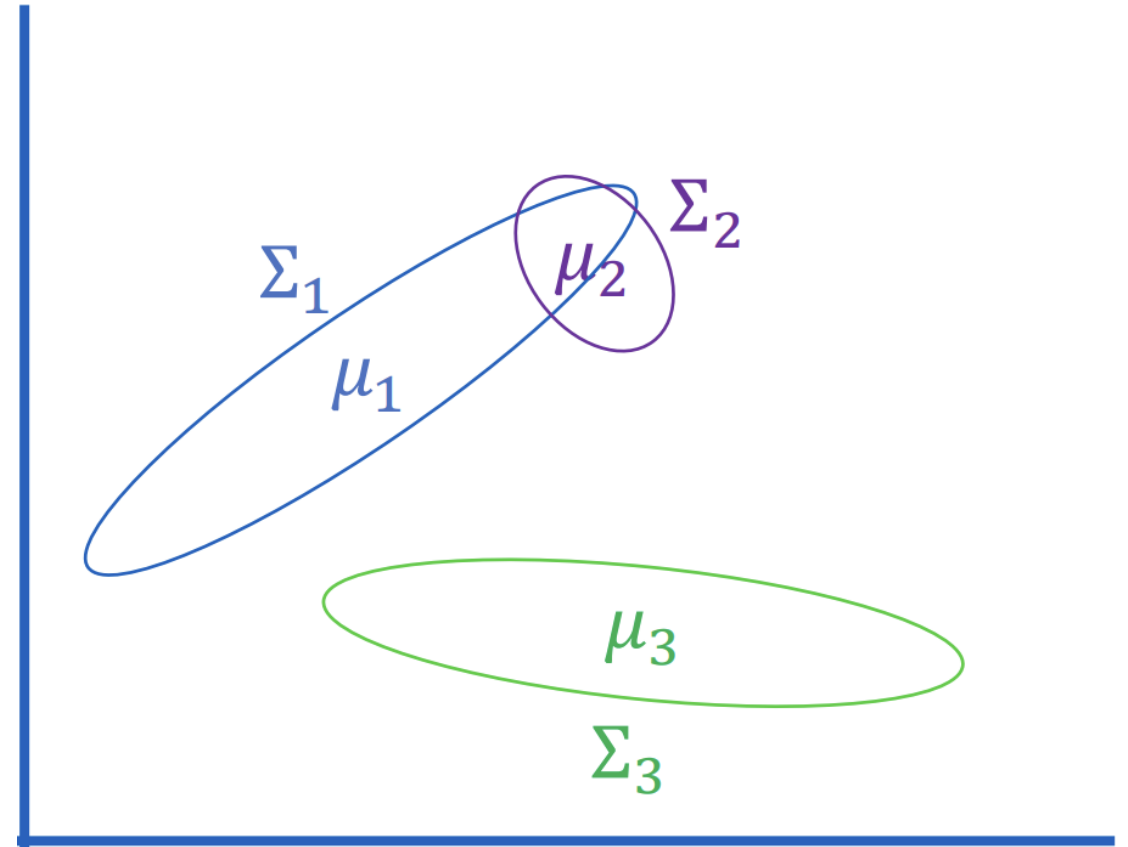
Assume the following data-generating model for our dataset,  $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$

1. Sample a cluster at random:

$$p(z_k^{(i)} = 1) = \pi_k$$

2. Sample a data point from the chosen cluster:

$$p(\mathbf{x}^{(i)} | z_k^{(i)} = 1) \sim N(\mu_k, \Sigma_k)$$



# Gaussian Mixture Models (GMMs)

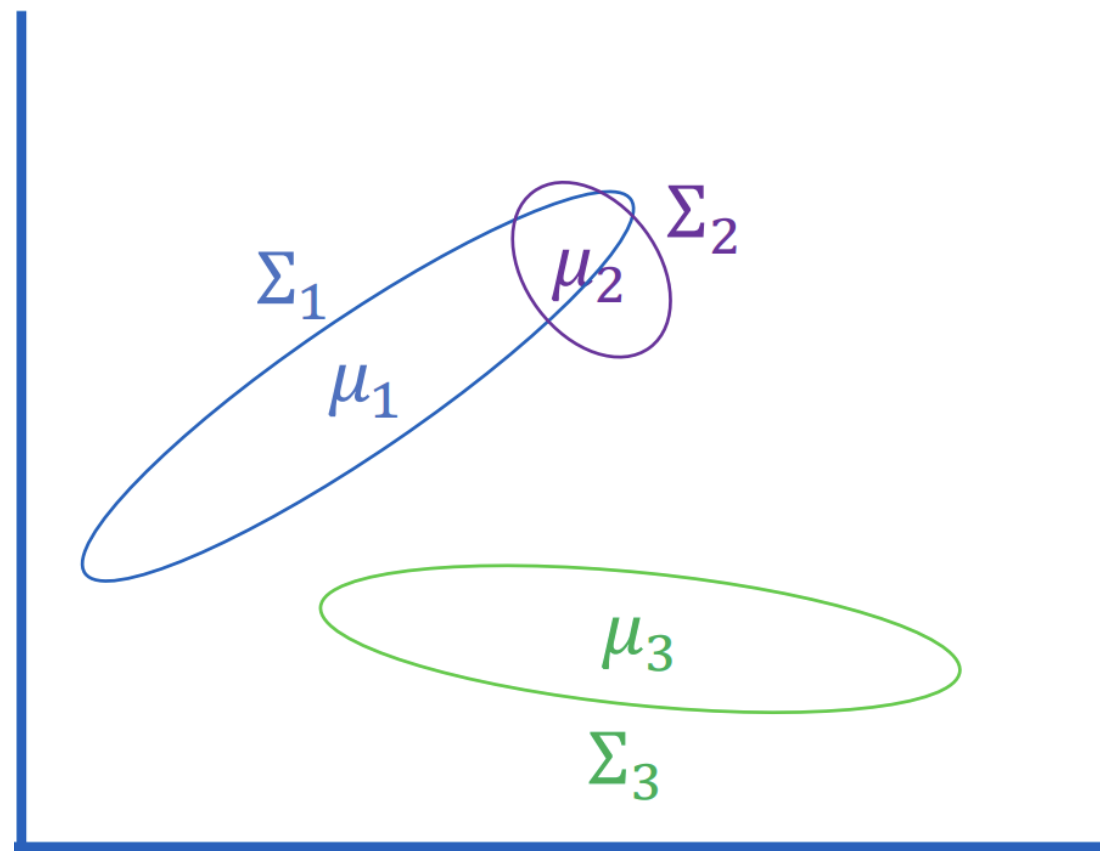
Assume the following data-generating model for our dataset,  $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$

1. Sample a cluster at random:

$$p(z_k^{(i)} = 1) = \pi_k$$

2. Sample a data point from the chosen cluster:

$$p(\mathbf{x}^{(i)} | z_k^{(i)} = 1) \sim N(\mu_k, \Sigma_k)$$



Let  $\theta = \{\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \pi_1, \dots, \pi_K\}$

# Maximizing the Likelihood?

- The log likelihood of  $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\}_{i=1}^N$  is

$$\begin{aligned}
 \ell_{\mathcal{D}}(\theta) &= \log \prod_{i=1}^N p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} | \theta) = \log \prod_{i=1}^N p(\mathbf{x}^{(i)} | \mathbf{z}^{(i)}, \theta) p(\mathbf{z}^{(i)} | \theta) \\
 &= \sum_{i=1}^N \log \underbrace{p(\mathbf{x}^{(i)} | \mathbf{z}^{(i)}, \theta)}_{\pi} + \log \underbrace{p(\mathbf{z}^{(i)} | \theta)}_{\pi} \\
 &= \sum_{i=1}^N \log \prod_{k=1}^K p(\mathbf{x}^{(i)} | \mathbf{z}_k^{(i)} = 1, \theta)^{\mathbf{z}_k^{(i)}} + \log \prod_{k=1}^K p(\mathbf{z}_k^{(i)} = 1 | \theta)^{\mathbf{z}_k^{(i)}} \\
 &= \sum_{i=1}^N \sum_{k=1}^K \mathbf{z}_k^{(i)} \log \underbrace{p(\mathbf{x}^{(i)} | \mathbf{z}_k^{(i)} = 1, \theta)}_{N(\mathbf{x}^{(i)}; \mu_k, \Sigma_k)} + \sum_{k=1}^K \mathbf{z}_k^{(i)} \log \underbrace{p(\mathbf{z}_k^{(i)} = 1 | \theta)}_{\pi_k}
 \end{aligned}$$



# Maximizing the Likelihood?

- The log likelihood of  $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\}_{i=1}^N$  is

$$\begin{aligned}\ell_{\mathcal{D}}(\theta) &= \log \prod_{i=1}^N p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} | \theta) = \sum_{i=1}^N \log p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} | \theta) \\&= \sum_{i=1}^N \log p(\mathbf{x}^{(i)} | \mathbf{z}^{(i)}, \theta) + \log p(\mathbf{z}^{(i)} | \theta) \\&= \sum_{i=1}^N \log \prod_{k=1}^K p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \theta)^{z_k^{(i)}} + \log \prod_{k=1}^K p(z_k^{(i)} = 1 | \theta)^{z_k^{(i)}} \\&= \sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} \log p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \theta) + \sum_{k=1}^K z_k^{(i)} \log p(z_k^{(i)} = 1 | \theta) \\&= \sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} (\log N(\mathbf{x}^{(i)}; \mu_k, \Sigma_k) + \log \pi_k)\end{aligned}$$

Maximizing  
the  
Complete  
Likelihood is  
easy but  
requires  $z^{(i)}$ !

- The log complete likelihood of  $\mathcal{D} = \{\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\}_{i=1}^N$  is

$$\begin{aligned}\ell_{\mathcal{D}}(\theta) &= \log \prod_{i=1}^N p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} | \theta) = \sum_{i=1}^N \log p(\mathbf{x}^{(i)}, \mathbf{z}^{(i)} | \theta) \\&= \sum_{i=1}^N \log p(\mathbf{x}^{(i)} | \mathbf{z}^{(i)}, \theta) + \log p(\mathbf{z}^{(i)} | \theta) \\&= \sum_{i=1}^N \log \prod_{k=1}^K p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \theta)^{z_k^{(i)}} + \log \prod_{k=1}^K p(z_k^{(i)} = 1 | \theta)^{z_k^{(i)}} \\&= \sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} \log p(\mathbf{x}^{(i)} | z_k^{(i)} = 1, \theta) + \sum_{k=1}^K z_k^{(i)} \log p(z_k^{(i)} = 1 | \theta) \\&= \sum_{i=1}^N \sum_{k=1}^K z_k^{(i)} (\log N(\mathbf{x}^{(i)}; \mu_k, \Sigma_k) + \log \pi_k)\end{aligned}$$

- Parameters decoupled  $\rightarrow$  set partial derivatives equal to 0

# Maximizing the Marginal Likelihood

- The log *marginal* likelihood of  $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$  is *low of total probability*

$$\ell(\theta|\mathcal{D}) = \log \prod_{i=1}^N p(\mathbf{x}^{(i)}|\theta) = \sum_{i=1}^N \log p(\mathbf{x}^{(i)}|\theta)$$

$$= \sum_{i=1}^N \log \sum_{\mathbf{z}^{(i)}} p(\mathbf{x}^{(i)}|\mathbf{z}^{(i)}, \theta) p(\mathbf{z}^{(i)}|\theta)$$

*marginalizing over z*

$$= \sum_{i=1}^N \log \sum_{\mathbf{z}^{(i)}} \prod_{k=1}^K \left( p(\mathbf{x}^{(i)}|z_k^{(i)} = 1, \theta) p(z_k^{(i)} = 1|\theta) \right)^{z_k^{(i)}}$$

$$= \sum_{i=1}^N \log \sum_{\mathbf{z}^{(i)}} \prod_{k=1}^K \left( N(\mathbf{x}^{(i)}; \mu_k, \Sigma_k) \pi_k \right)^{z_k^{(i)}}$$

- Parameters coupled and *constrained*  $\rightarrow$  gradient ascent is possible but complicated and slow to converge

# Recipe for GMMs

- Define a model and model parameters
    - Assume  $K$  Gaussian clusters
    - Parameters:  $\theta = \{\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \pi_1, \dots, \pi_K\}$
  - Write down an objective function
    - Maximize the log marginal likelihood
- $$\ell_{\mathcal{D}}(\theta) = \log \prod_{i=1}^N p(\mathbf{x}^{(i)} | \theta)$$
- Optimize the objective w.r.t. the model parameters
    - Expectation-maximization  $EM$

# Expectation- Maximization for GMMs: Intuition

- Insight: if we knew the cluster assignments,  $\mathbf{z}^{(i)}$ , we could maximize the log complete likelihood instead of the log marginal likelihood
- Idea: replace  $\mathbf{z}^{(i)}$  in the log complete likelihood with our “best guess” for  $\mathbf{z}^{(i)}$  given the parameters and the data
- Observation: changing the parameters changes our “best guess” and vice versa
- Approach: iterate between updating our “best guess” and updating the parameters

# Expectation- Maximization for GMMs

- Iterative algorithm that alternates between two steps
  - **Expectation or E-step:** for fixed parameters  $\theta$ , compute the expected assignment vectors conditioned on  $\theta$  and the data set  $\mathcal{D}$ 
$$E \left[ \underbrace{z_k^{(i)}}_{\text{assignment vector}} \mid \underline{x^{(i)}}, \underline{\theta} \right] = \underbrace{p \left( z_k^{(i)} = 1 \mid x^{(i)}, \theta \right)}_{\text{probability}} \quad \forall i \text{ and } k$$
  - **Maximization or M-step:** for fixed assignment vectors  $\mathbf{z}^{(i)}$ , set the parameters  $\theta$  to *maximize* the complete log likelihood of the data set  $\mathcal{D}$
- Under the hood: EM performs block-coordinate ascent on a lower bound (ELBO) of the log marginal likelihood

↓  
Evidence Lower Bound

## E-Step for GMMs

$$p(z_k^{(i)} = 1 | x^{(i)}, \theta) = \frac{p(z_k^{(i)} = 1, x^{(i)} | \theta)}{P(x^{(i)} | \theta)} \rightarrow \frac{p(z_k^{(i)} = 1, x^{(i)} | \theta)}{\sum_{j=1}^K p(z_j^{(i)} = 1, x^{(i)} | \theta)}$$
$$= \frac{\pi_k \mathcal{N}(x^{(i)}; \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x^{(i)}; \mu_j, \Sigma_j)}$$

## E-Step for GMMs

$$\begin{aligned} p\left(z_k^{(i)} = 1 \mid \mathbf{x}^{(i)}, \theta\right) &= \frac{p\left(z_k^{(i)} = 1, \mathbf{x}^{(i)} \mid \theta\right)}{p\left(\mathbf{x}^{(i)} \mid \theta\right)} \\ &= \frac{p\left(z_k^{(i)} = 1, \mathbf{x}^{(i)} \mid \theta\right)}{\sum_{j=1}^K p\left(z_j^{(i)} = 1, \mathbf{x}^{(i)} \mid \theta\right)} \\ &= \frac{\pi_k N\left(\mathbf{x}^{(i)} ; \mu_k, \Sigma_k\right)}{\sum_{j=1}^K \pi_j N\left(\mathbf{x}^{(i)} ; \mu_j, \Sigma_j\right)} \quad \forall i \text{ and } k \end{aligned}$$



## M-Step for GMMs

$$\text{Let } \underbrace{N_k}_{\text{red}} = \sum_{i=1}^N p(\underbrace{z_k^{(i)} = 1}_{\text{red}} | \mathbf{x}^{(i)}, \theta)$$

$$\pi_k = \frac{N_k}{N}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^N \underbrace{p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta)}_{\text{red}} \underbrace{\mathbf{x}^{(i)}}_{\text{red}}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^N p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta) (\mathbf{x}^{(i)} - \mu_k) (\mathbf{x}^{(i)} - \mu_k)^T$$

## M-Step for GMMs

$$\text{Let } N_k = \sum_{i=1}^N p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta)$$

$$\pi_k = \frac{N_k}{N}$$

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^N p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta) \mathbf{x}^{(i)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{i=1}^N p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta) (\mathbf{x}^{(i)} - \mu_k)(\mathbf{x}^{(i)} - \mu_k)^T$$

# GMM Algorithm

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(i)})\}_{i=1}^N, K$   
 $\theta$
- 1. Initialize all parameters  $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \pi_1, \dots, \pi_K$
- 2. While NOT CONVERGED
  - a. E-step: compute  $p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta) \forall i$  and  $k$
  - b. M-step: update the parameters
- Output: parameters  $\mu_1, \dots, \mu_K, \Sigma_1, \dots, \Sigma_K, \pi_1, \dots, \pi_K$  and assignments probabilities  $p(z_k^{(i)} = 1 | \mathbf{x}^{(i)}, \theta) \forall i$  and  $k$

# Initializing EM for GMMs

- Common heuristics for initialization

$\pi_k$ 's • Cluster proportions typically initialized to be uniform

$\mu_k$ 's • Cluster means

- Randomly select data points to be cluster centers
- Randomly sample locations in the range spanned by the data

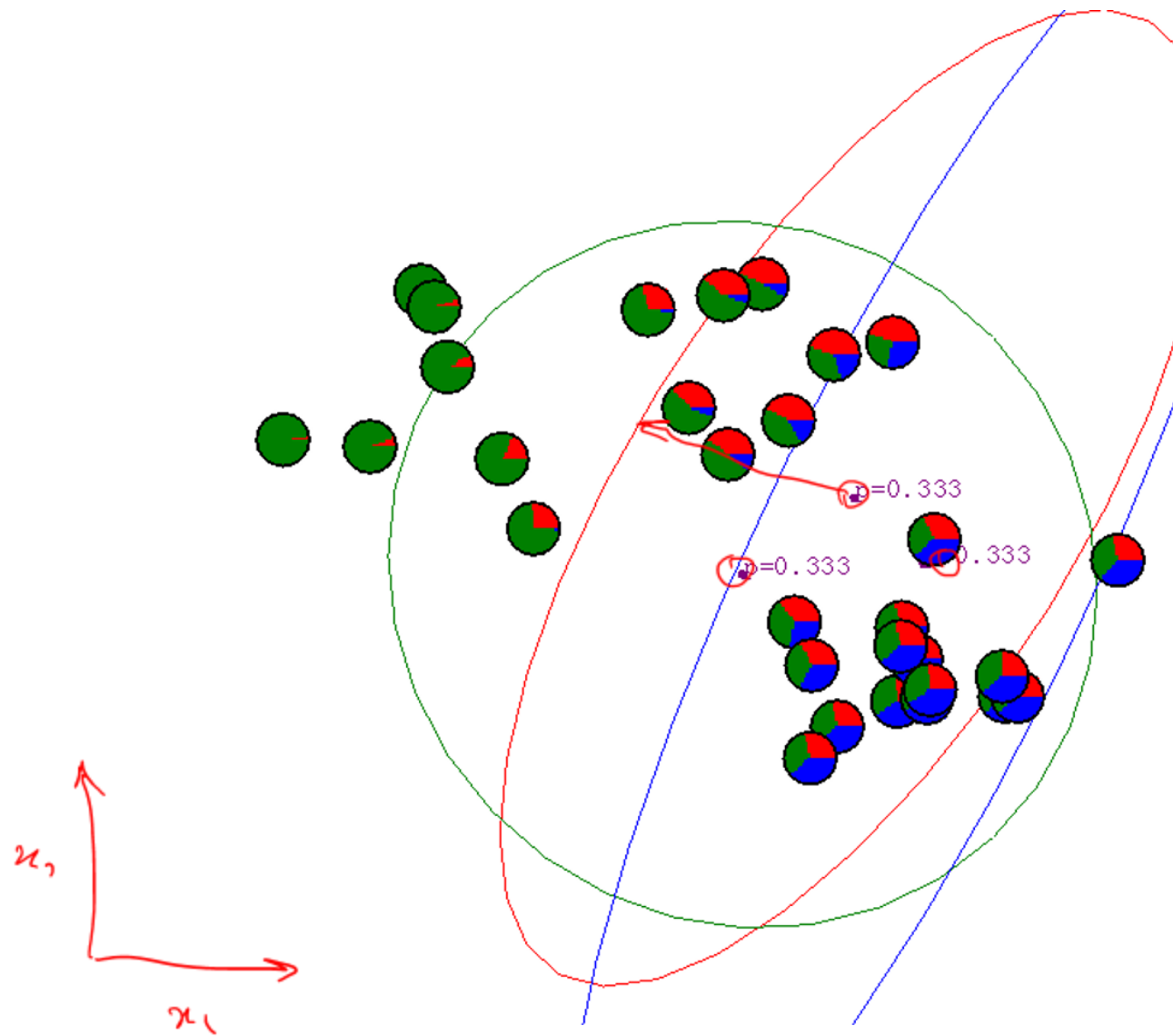
$\Sigma_k$ 's • Cluster covariances

- Identity (or scaled identity) matrix  $I$
- Random positive diagonal matrix
- • Randomly sample  $L$ , a lower triangular matrix with positive diagonal entries, and set to  $LL^T$
- Set to the empirical covariance of the data
- Use multiple random restarts

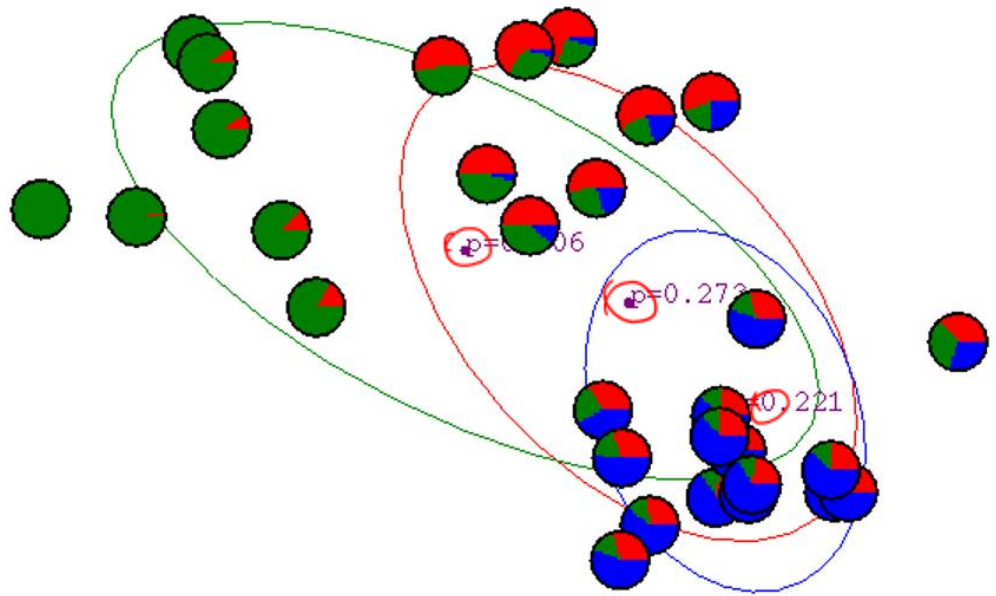
# Terminating EM for GMMs

- Common heuristics for termination
  - Stop if the log complete likelihood changes by less than some tolerance
  - Stop if the parameters and assignment probabilities change by less than some tolerance
  - Stop after a fixed number of iterations

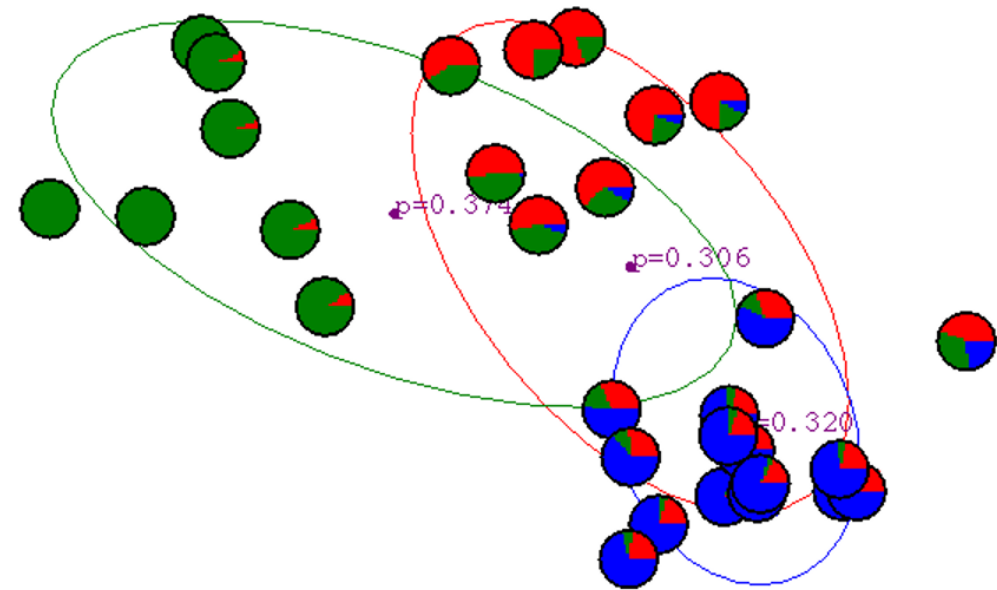
# GMMs: Example (Initial)



# GMMs: Example (1 Iteration)

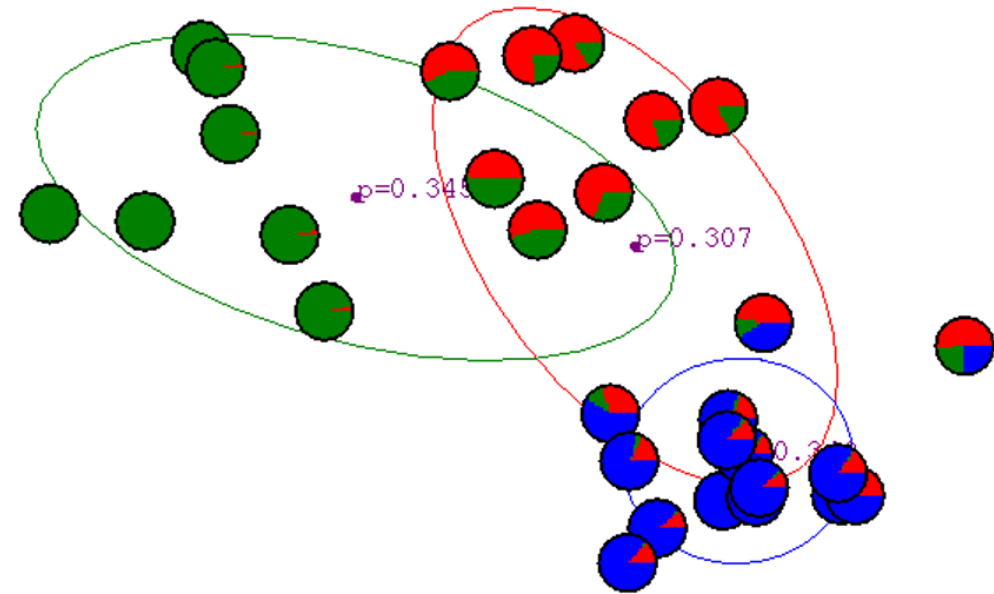


# GMMs: Example (2 Iterations)

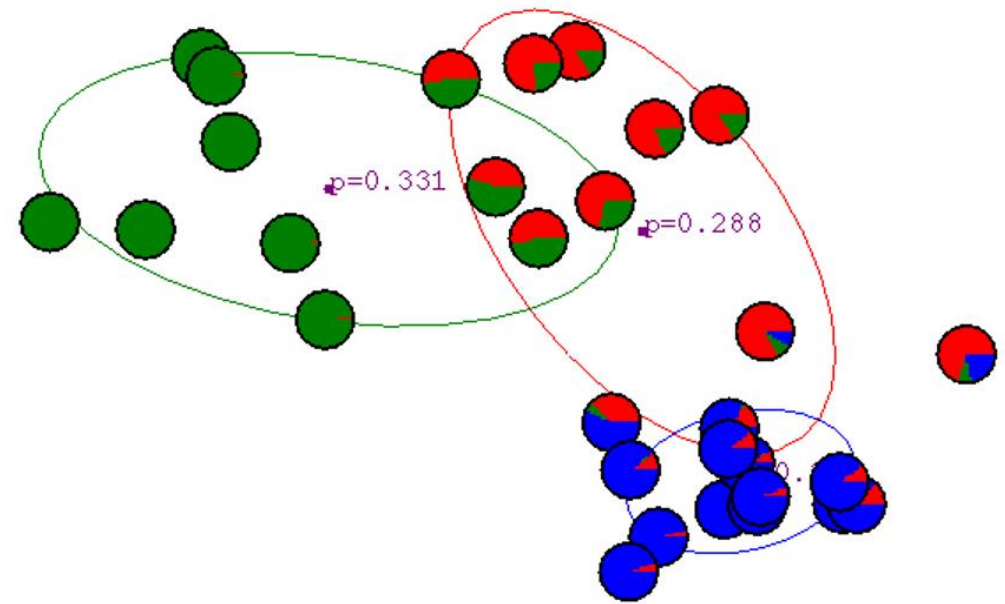




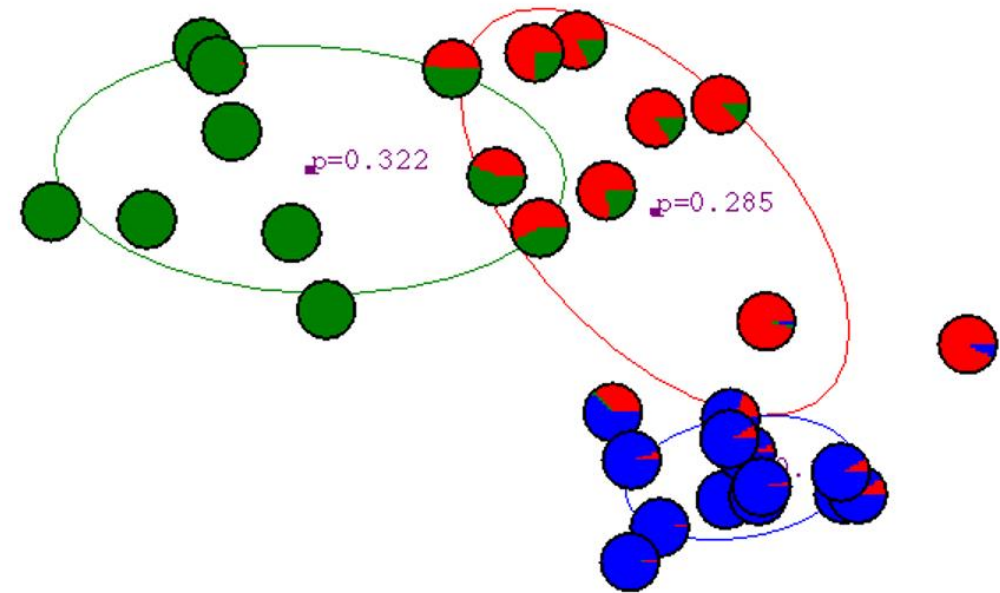
# GMMs: Example (3 Iterations)



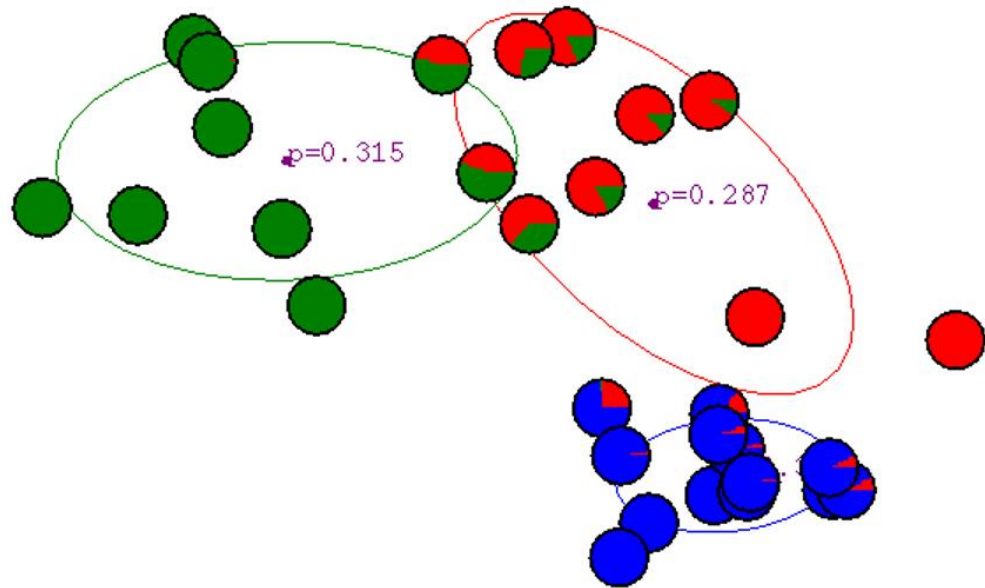
# GMMs: Example (4 Iterations)



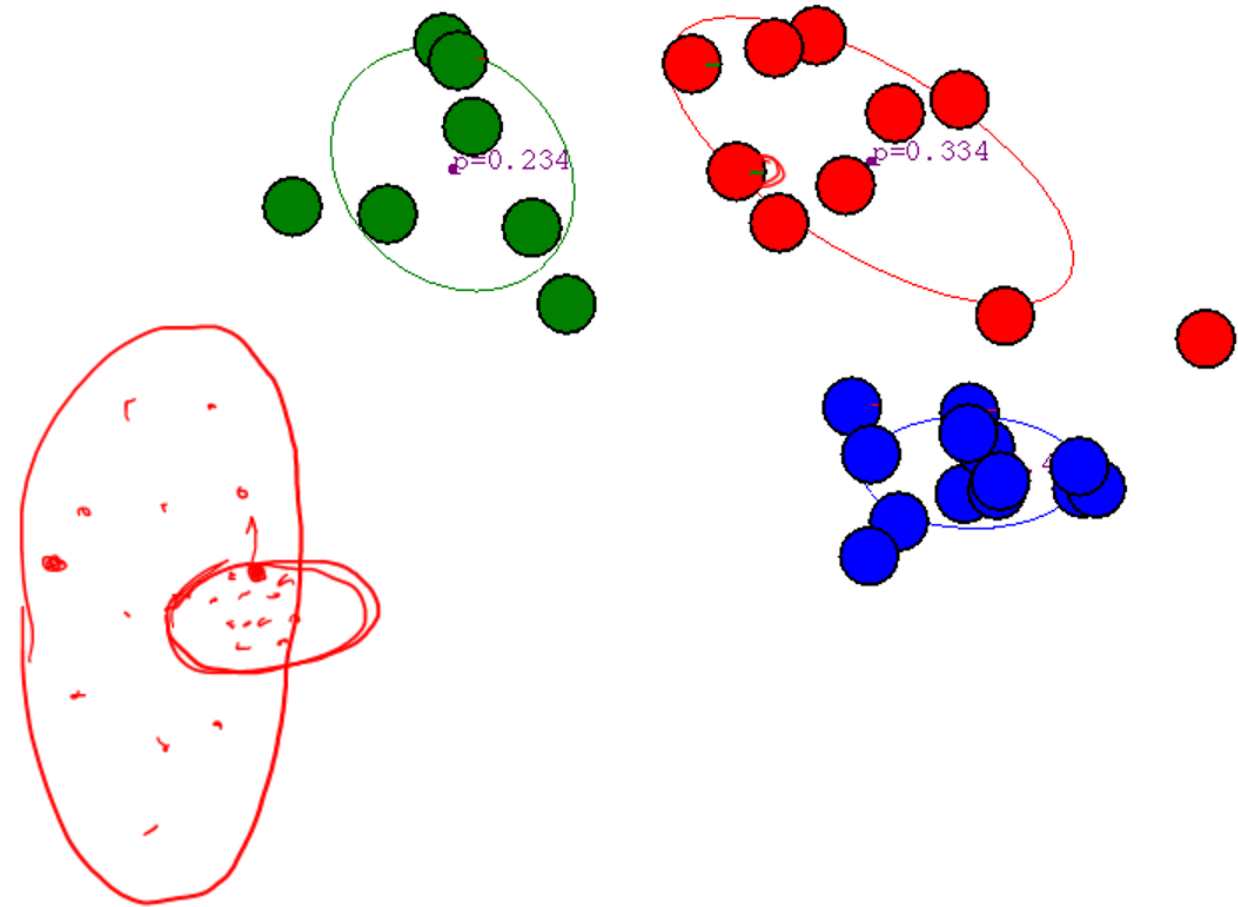
# GMMs: Example (5 Iterations)



# GMMs: Example (6 Iterations)



# GMMs: Example (20 Iterations)

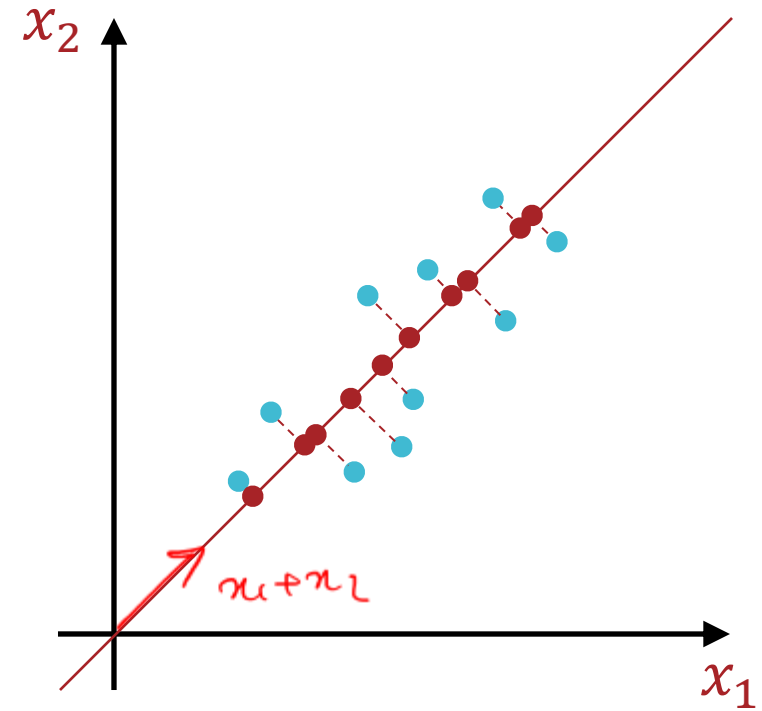
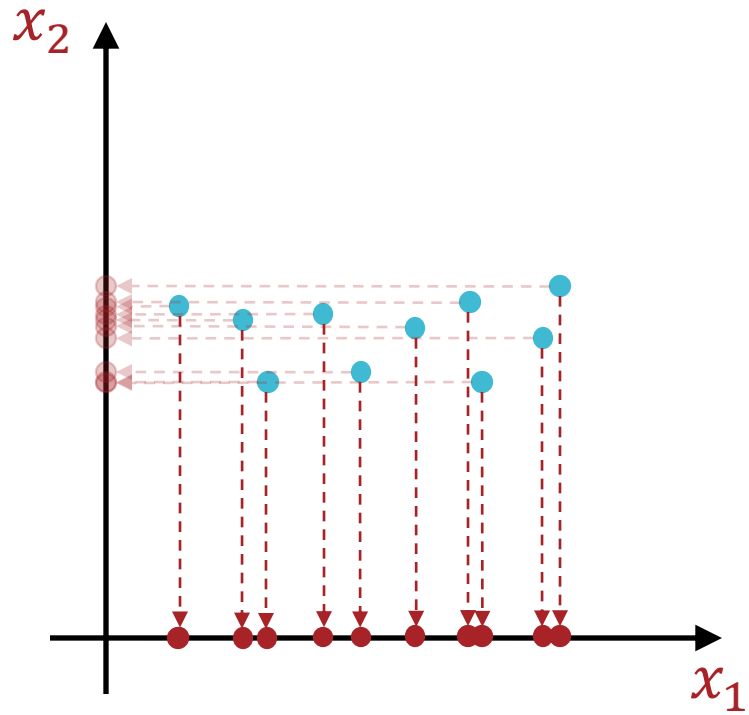


# Key Takeaways

- Partition-based clustering
  - $K$ -means (hard assignments)
    - Block-coordinate descent
    - Setting  $K$
    - Initializing  $K$  means
- Model-based clustering
  - Gaussian mixture models (probabilistic assignments)
    - Complete vs. marginal likelihood
    - Expectation-maximization for GMMs
    - Initializing EM for GMMs

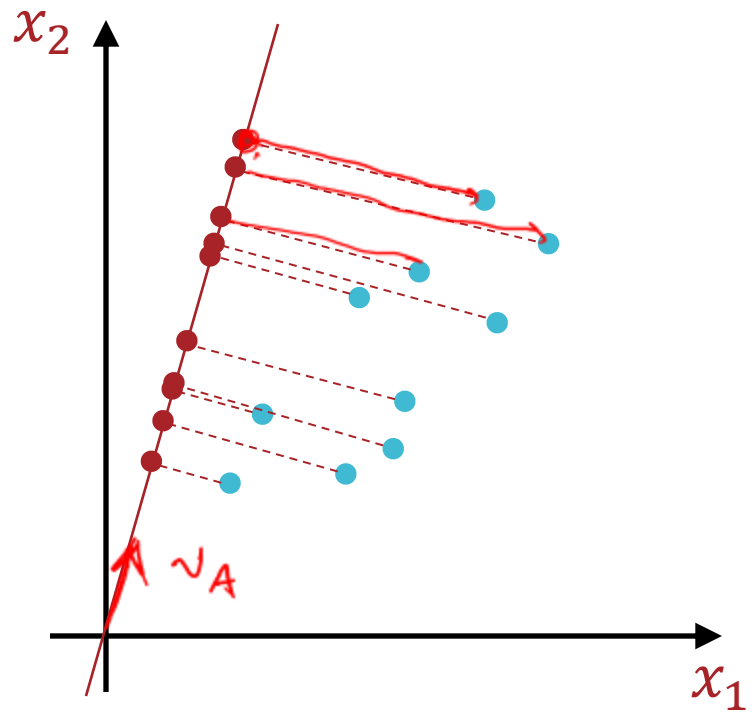
# Unsupervised Learning

- Clustering: split an unlabeled data set into groups or partitions of “similar” data points
  - Use cases:
    - Organizing data
    - Discovering patterns or structure
    - Preprocessing for downstream tasks
- Dimensionality Reduction: given some unlabeled data set, learn a latent (typically lower-dimensional) representation
  - Use cases:
    - Decreasing computational costs
    - Improving generalization
    - Visualizing data

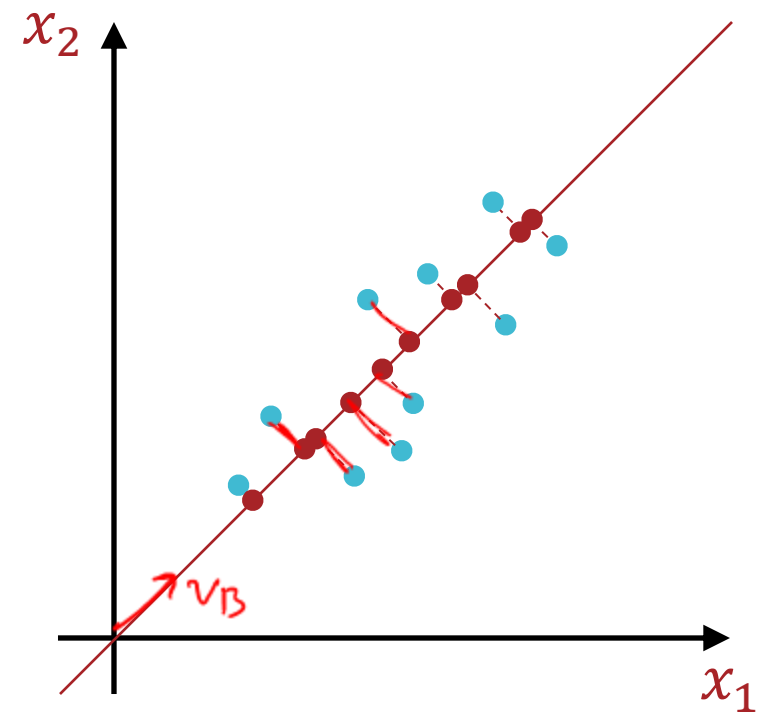


# Feature Elimination



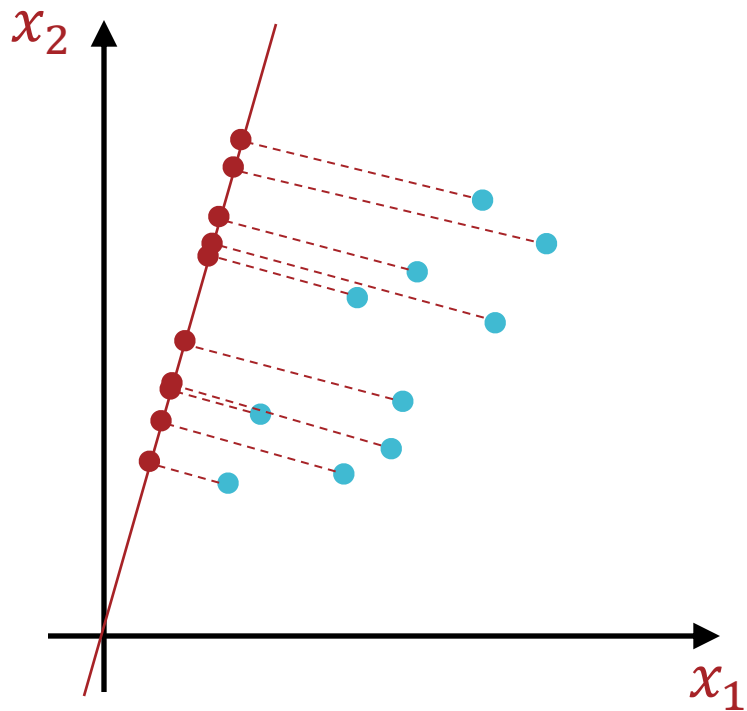


(A)

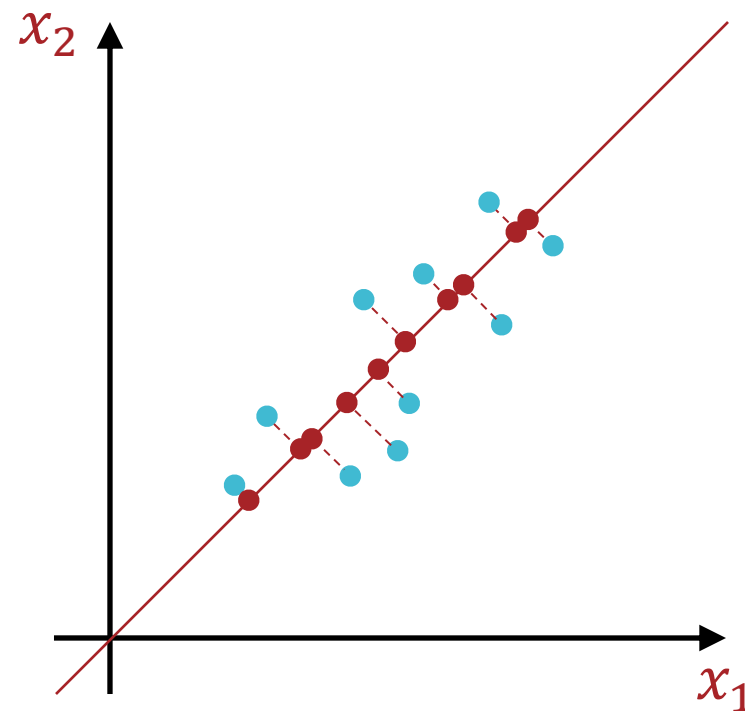


(B)

# Feature Reduction




Option A



Option B

Which projection do you prefer?

# Centering the Data

- To be consistent, we will constrain principal components to be *orthogonal unit vectors* that begin at the origin 
- Preprocess data to be centered around the origin:

1.  $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}^{(n)}$

2.  $\underbrace{\tilde{\mathbf{x}}^{(n)}} = \mathbf{x}^{(n)} - \boldsymbol{\mu} \forall n$

3.  $\tilde{X} = \begin{bmatrix} \tilde{\mathbf{x}}^{(1)T} \\ \tilde{\mathbf{x}}^{(2)T} \\ \vdots \\ \tilde{\mathbf{x}}^{(N)T} \end{bmatrix}$

# Computing Projections

- The projection of  $\tilde{\mathbf{x}}^{(n)}$  onto a vector  $\mathbf{v}$  is

$$\mathbf{z}^{(n)} = \underbrace{\left( \frac{\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}}{\|\mathbf{v}\|_2} \right)}_{\text{Length of projection}} \underbrace{\left( \frac{\mathbf{v}}{\|\mathbf{v}\|_2} \right)}_{\text{Direction of projection}}$$

Length of projection

Direction of projection

# Reconstruction Error

- The projection of  $\tilde{\mathbf{x}}^{(n)}$  onto a unit vector  $\mathbf{v}$  is

$$\mathbf{z}^{(n)} = \underbrace{(\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})}_{\text{scalar}} \underbrace{\mathbf{v}}_{\text{unit vector}}$$

Reconstruction error for dataset  $\mathcal{D}$

$$\hat{\mathbf{v}} = \operatorname{argmin}_{\mathbf{v}: \underbrace{\|\mathbf{v}\|_2^2}_{=1}} \sum_{n=1}^N \underbrace{\left\| \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v} \right\|_2^2}_{\text{reconstruction error for } \tilde{\mathbf{x}}^{(n)}}$$

$$\begin{aligned} \rightarrow & \left\| \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v} \right\|_2^2 \\ &= \left( \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v} \right)^T \left( \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v} \right) \\ &= \underbrace{\tilde{\mathbf{x}}^{(n)T} \tilde{\mathbf{x}}^{(n)}}_{\text{norm of } \tilde{\mathbf{x}}^{(n)}} - 2 \underbrace{(\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})}_{\text{projection coefficient squared}} + \underbrace{(\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})^2}_{\text{projection coefficient squared}} \cancel{\mathbf{v}^T \mathbf{v}} \\ &= \tilde{\mathbf{x}}^{(n)T} \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})^2 \end{aligned}$$

# Reconstruction Error

- The projection of  $\tilde{\mathbf{x}}^{(n)}$  onto a unit vector  $\mathbf{v}$  is

$$\mathbf{z}^{(n)} = (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}$$

$$\hat{\mathbf{v}} = \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \sum_{n=1}^N \|\tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}\|_2^2$$

$$\|\tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}\|_2^2$$

$$= \tilde{\mathbf{x}}^{(n)T} \tilde{\mathbf{x}}^{(n)} - 2(\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}^T \tilde{\mathbf{x}}^{(n)} + (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}^T \mathbf{v}$$

$$= \tilde{\mathbf{x}}^{(n)T} \tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}^T \tilde{\mathbf{x}}^{(n)}$$

$$= \|\tilde{\mathbf{x}}^{(n)}\|_2^2 - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})^2$$

Minimizing the  
Reconstruction  
Error



Maximizing the  
Variance

$$\hat{\mathbf{v}} = \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \sum_{n=1}^N \underbrace{\|\tilde{\mathbf{x}}^{(n)} - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}) \mathbf{v}\|_2^2}$$

$$= \operatorname{argmin}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \sum_{n=1}^N \underbrace{\|\tilde{\mathbf{x}}^{(n)}\|_2^2 - (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})^2} \quad \leftarrow$$

$$= \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \sum_{n=1}^N (\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})^2 \quad \leftarrow \begin{array}{l} \text{Variance of projections} \\ (\tilde{\mathbf{x}}^{(n)} \text{ are centered}) \end{array}$$

$$\rightarrow = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \mathbf{v}^T \left( \sum_{n=1}^N \tilde{\mathbf{x}}^{(n)} \tilde{\mathbf{x}}^{(n)T} \right) \mathbf{v}$$

$$= \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \mathbf{v}^T (\mathbf{X}^T \mathbf{X}) \mathbf{v}$$

$$\underbrace{(\mathbf{v}^T \tilde{\mathbf{x}}^{(n)})}_{\mathbf{v}^T \tilde{\mathbf{x}}^{(n)}} \underbrace{(\tilde{\mathbf{x}}^{(n)T} \mathbf{v})}_{\tilde{\mathbf{x}}^{(n)T} \mathbf{v}} = \mathbf{v}^T \left( \tilde{\mathbf{x}}^{(n)} \tilde{\mathbf{x}}^{(n)T} \right) \mathbf{v}$$

# Maximizing the Variance

$$\hat{\mathbf{v}} = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2 = 1} \mathbf{v}^T (X^T X) \mathbf{v}$$

$$\begin{aligned} \mathcal{L}(\mathbf{v}, \lambda) &= \mathbf{v}^T (X^T X) \mathbf{v} - \lambda (\|\mathbf{v}\|_2^2 - 1) \\ &= \mathbf{v}^T (X^T X) \mathbf{v} - \lambda (\mathbf{v}^T \mathbf{v} - 1) \end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (X^T X) \mathbf{v} - \lambda \mathbf{v}$$

$$\text{at } \hat{\mathbf{v}} : (X^T X) \hat{\mathbf{v}} - \lambda \hat{\mathbf{v}} = 0 \Rightarrow (X^T X) \hat{\mathbf{v}} = \lambda \hat{\mathbf{v}}$$

eigenvalue  
↖  
eigenvector



## Maximizing the Variance

$$\hat{\mathbf{v}} = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2 = 1} \mathbf{v}^T (X^T X) \mathbf{v}$$

$$\begin{aligned} \mathcal{L}(\mathbf{v}, \lambda) &= \mathbf{v}^T (X^T X) \mathbf{v} - \lambda (\|\mathbf{v}\|_2^2 - 1) \\ &= \mathbf{v}^T (X^T X) \mathbf{v} - \lambda (\mathbf{v}^T \mathbf{v} - 1) \end{aligned}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (X^T X) \mathbf{v} - \lambda \mathbf{v}$$

$$\rightarrow (X^T X) \hat{\mathbf{v}} - \lambda \hat{\mathbf{v}} = 0 \rightarrow (X^T X) \hat{\mathbf{v}} = \lambda \hat{\mathbf{v}}$$

- $\hat{\mathbf{v}}$  is an eigenvector of  $X^T X$  and  $\lambda$  is the corresponding eigenvalue! But which one?

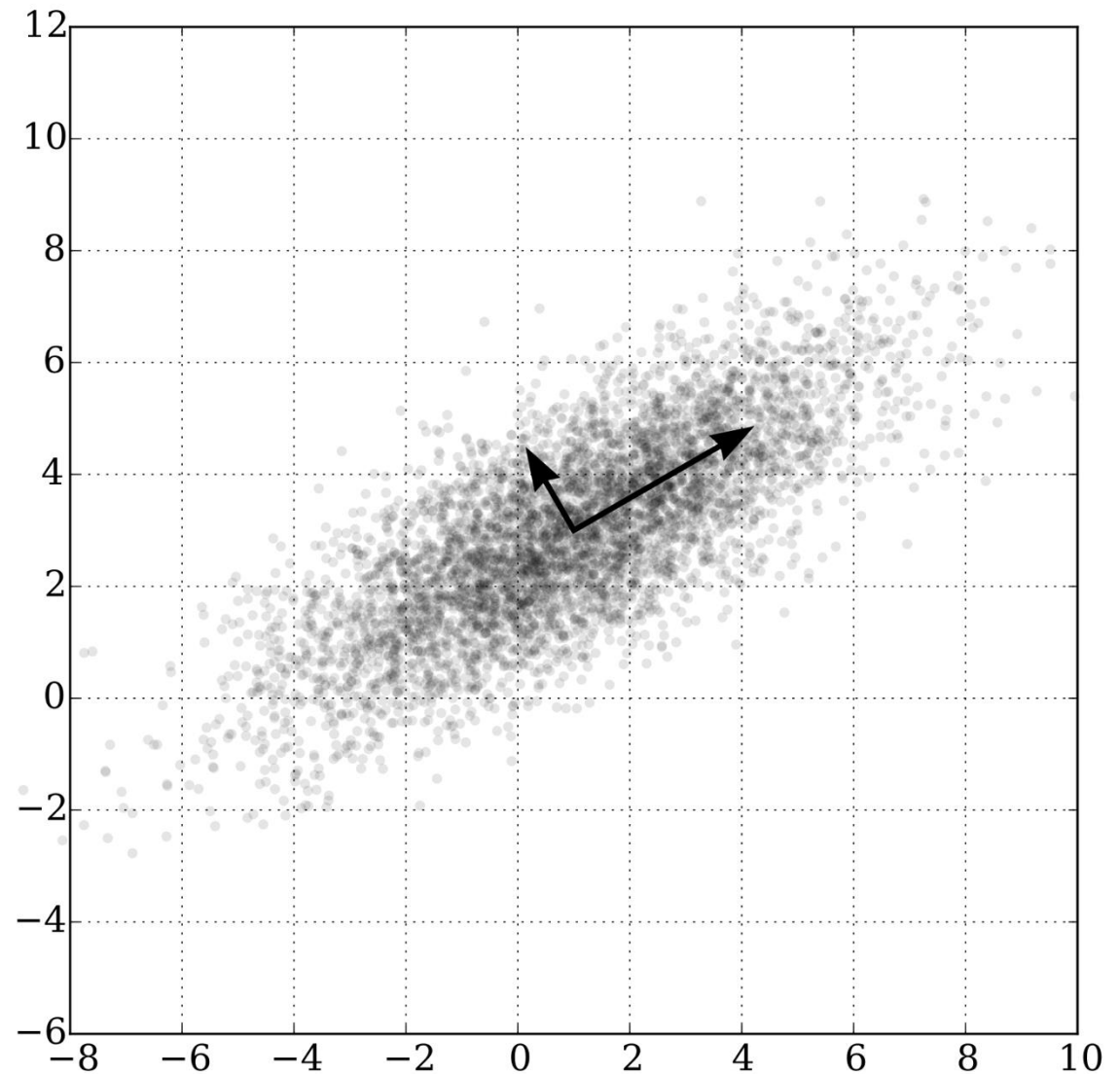
# Maximizing the Variance

$$\hat{\mathbf{v}} = \operatorname{argmax}_{\mathbf{v}: \|\mathbf{v}\|_2^2=1} \mathbf{v}^T \underbrace{(X^T X) \mathbf{v}}_{\lambda \hat{\mathbf{v}}}$$

$$\underbrace{(X^T X) \hat{\mathbf{v}}}_{\lambda \hat{\mathbf{v}}} = \lambda \hat{\mathbf{v}} \rightarrow \hat{\mathbf{v}}^T (X^T X) \hat{\mathbf{v}} = \underbrace{\lambda \hat{\mathbf{v}}^T \hat{\mathbf{v}}}_{\lambda}$$

- The first principal component is the eigenvector  $\hat{\mathbf{v}}_1$  that corresponds to the largest eigenvalue  $\lambda_1$
- The second principal component is the eigenvector  $\hat{\mathbf{v}}_2$  that corresponds to the second largest eigenvalue  $\lambda_1$ 
  - $\hat{\mathbf{v}}_1$  and  $\hat{\mathbf{v}}_2$  are orthogonal
- Etc ...
- $\lambda_i$  is a measure of how much variance falls along  $\hat{\mathbf{v}}_i$

# Principal Components: Example



# PCA Algorithm

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)})\}_{n=1}^N, \rho$
- ✓ 1. Center the data
- 2. Compute the eigenvalues and eigenvectors of  $(\tilde{X}^T \tilde{X})$
- 3. Collect the top  $\rho$  eigenvectors (corresponding to the  $\rho$  largest eigenvalues),  $\underbrace{V_\rho}_{\substack{\text{matrix of} \\ \text{eigenvectors}}} \in \mathbb{R}^{D \times \rho}$
- 4. Project the data into the space defined by  $\underbrace{V_\rho}_{\substack{\text{matrix of} \\ \text{eigenvectors}}}, \underbrace{Z}_{\substack{\text{transformed} \\ \text{data}}} = X V_\rho$
- Output:  $Z$ , the transformed (potentially lower-dimensional) data

## How many PCs should we use?

- Input:  $\mathcal{D} = \{(\mathbf{x}^{(n)})\}_{n=1}^N, \rho$ 
  1. Center the data
  2. Compute the eigenvalues and eigenvectors of  $\mathbf{X}^T \mathbf{X}$
  3. Collect the top  $\rho$  eigenvectors (corresponding to the  $\rho$  largest eigenvalues),  $\mathbf{V}_\rho \in \mathbb{R}^{D \times \rho}$
  4. Project the data into the space defined by  $\mathbf{V}_\rho$ ,  $\mathbf{Z} = \mathbf{X}\mathbf{V}_\rho$
- Output:  $\mathbf{Z}$ , the transformed (potentially lower-dimensional) data

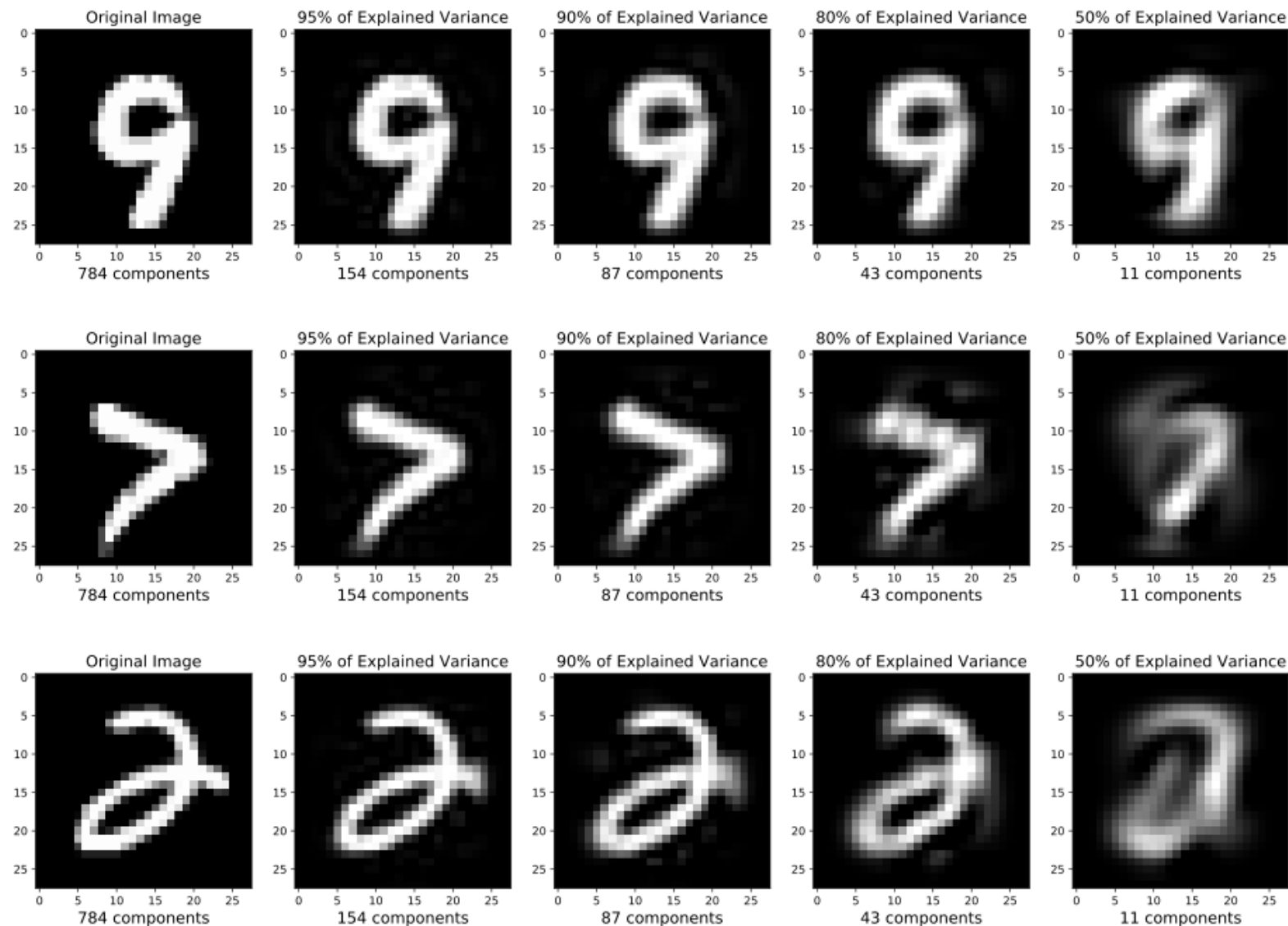
## Choosing the number of PCs

- Define a percentage of explained variance for the  $i^{\text{th}}$  PC:

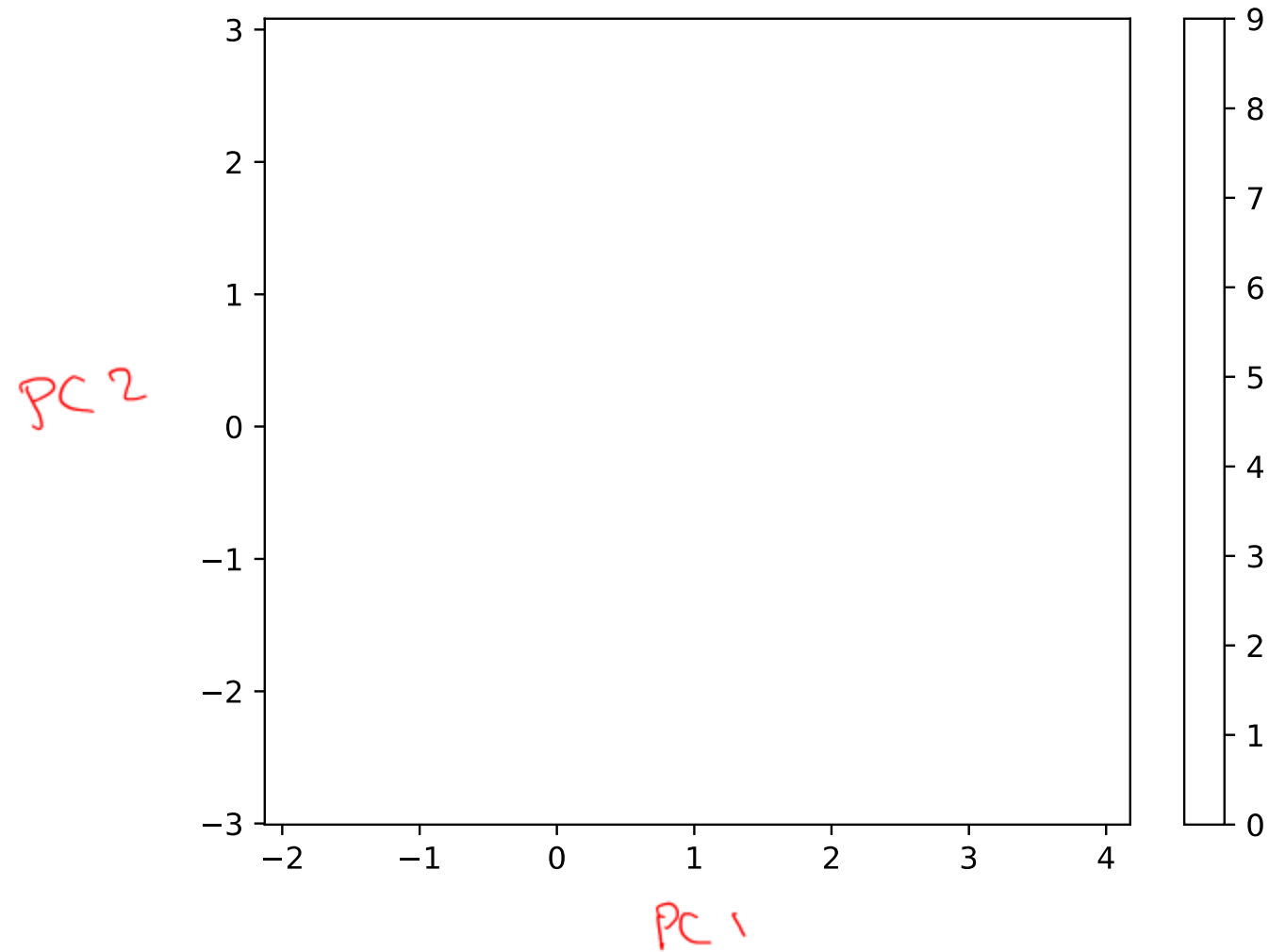
$$\underbrace{\lambda_i / \sum \lambda_j}$$

- Select all PCs above some threshold of explained variance, e.g., 5%
- Keep selecting PCs until the total explained variance exceeds some threshold, e.g., 90%
- Evaluate on some downstream metric

# PCA Example: MNIST Digits

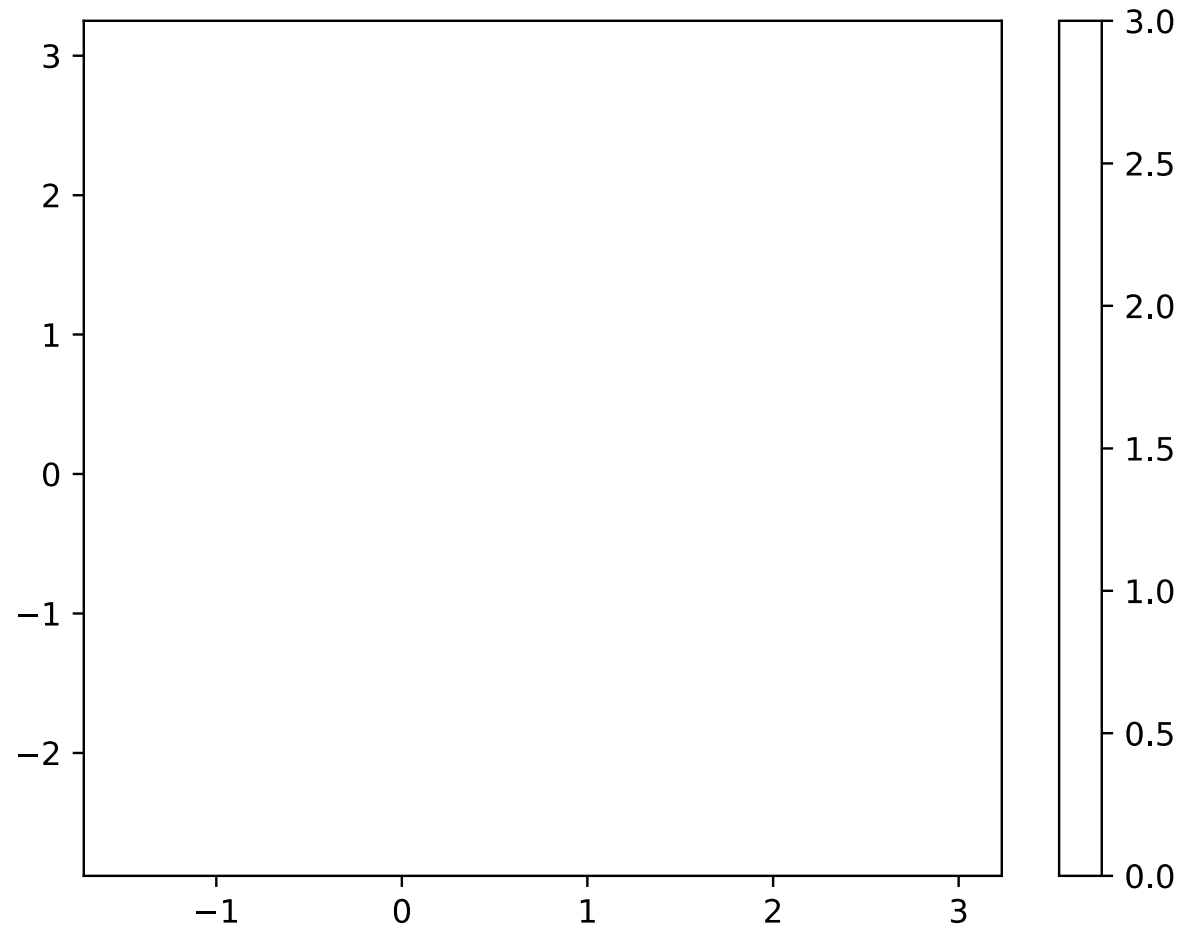


# PCA Example: MNIST Digits

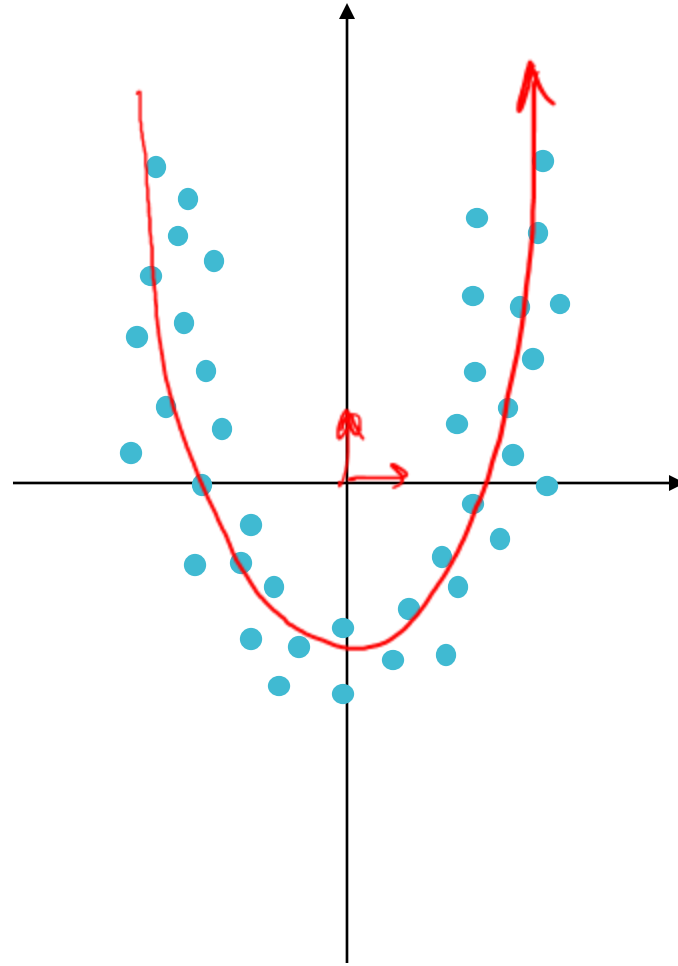




# PCA Example: MNIST Digits

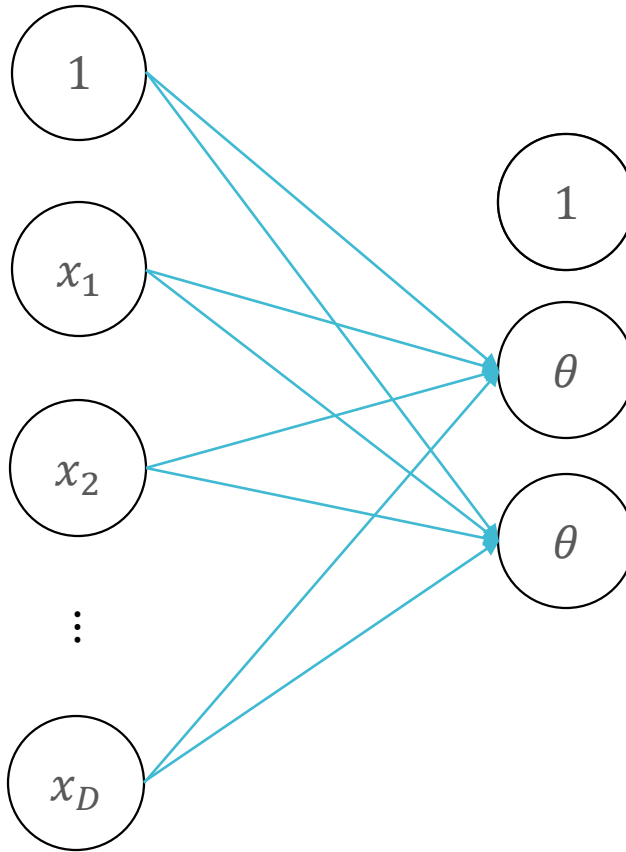


# Shortcomings of PCA



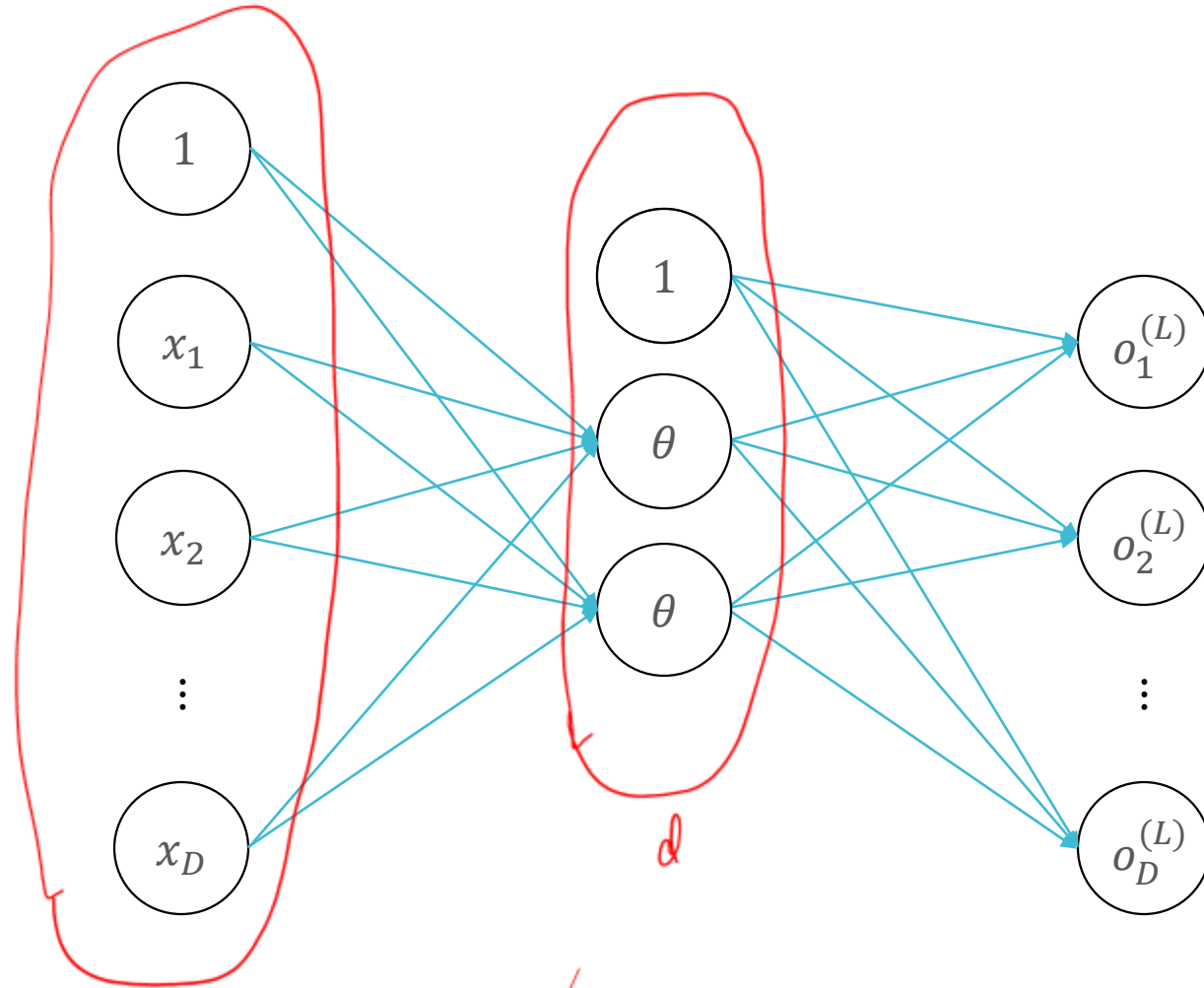
- Principal components are orthogonal (unit) vectors
- Principal components are expressed as **linear** combinations of the data

# Autoencoders



Insight: neural networks implicitly learn low-dimensional representations of inputs in hidden layers

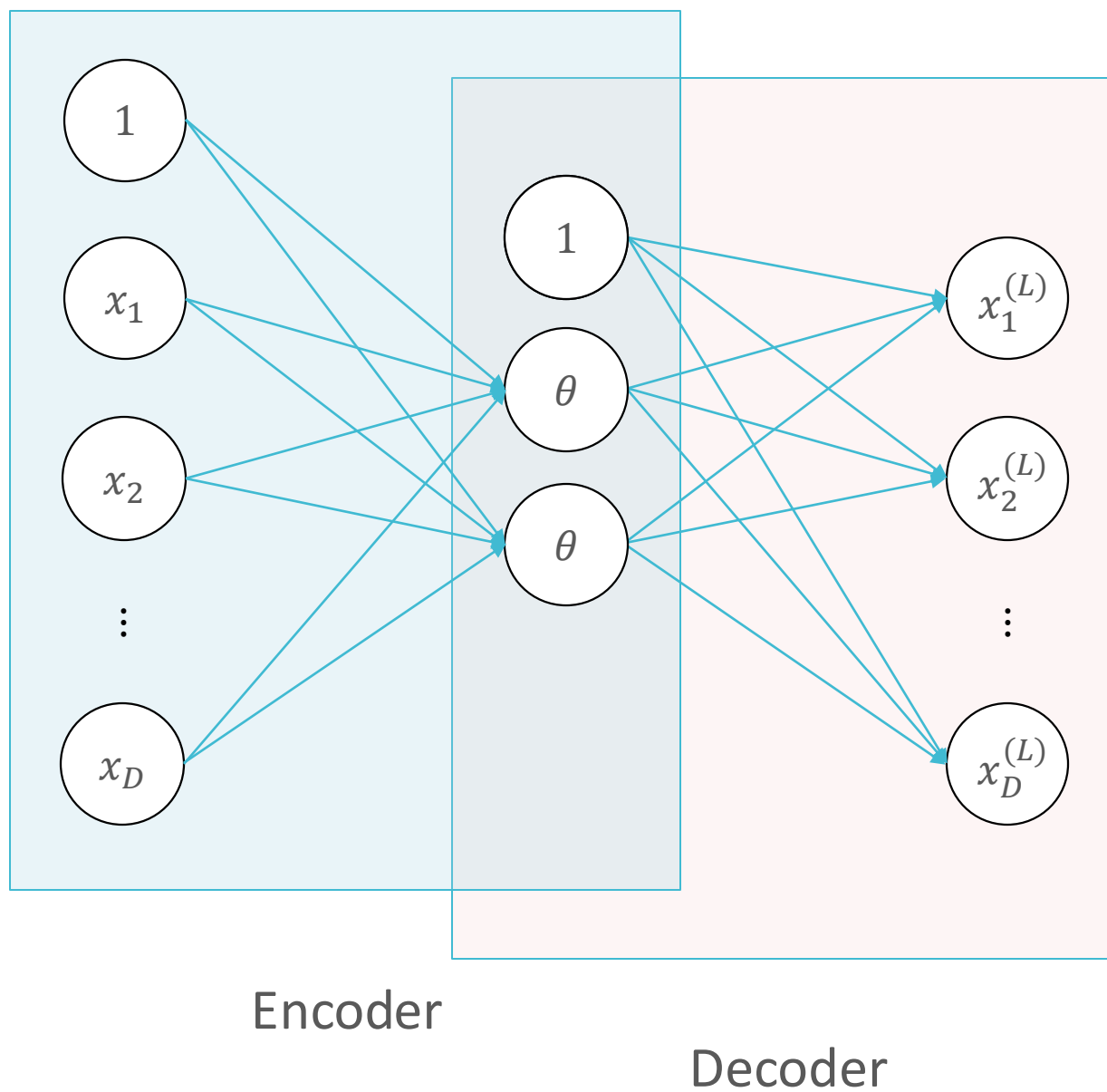
# Autoencoders



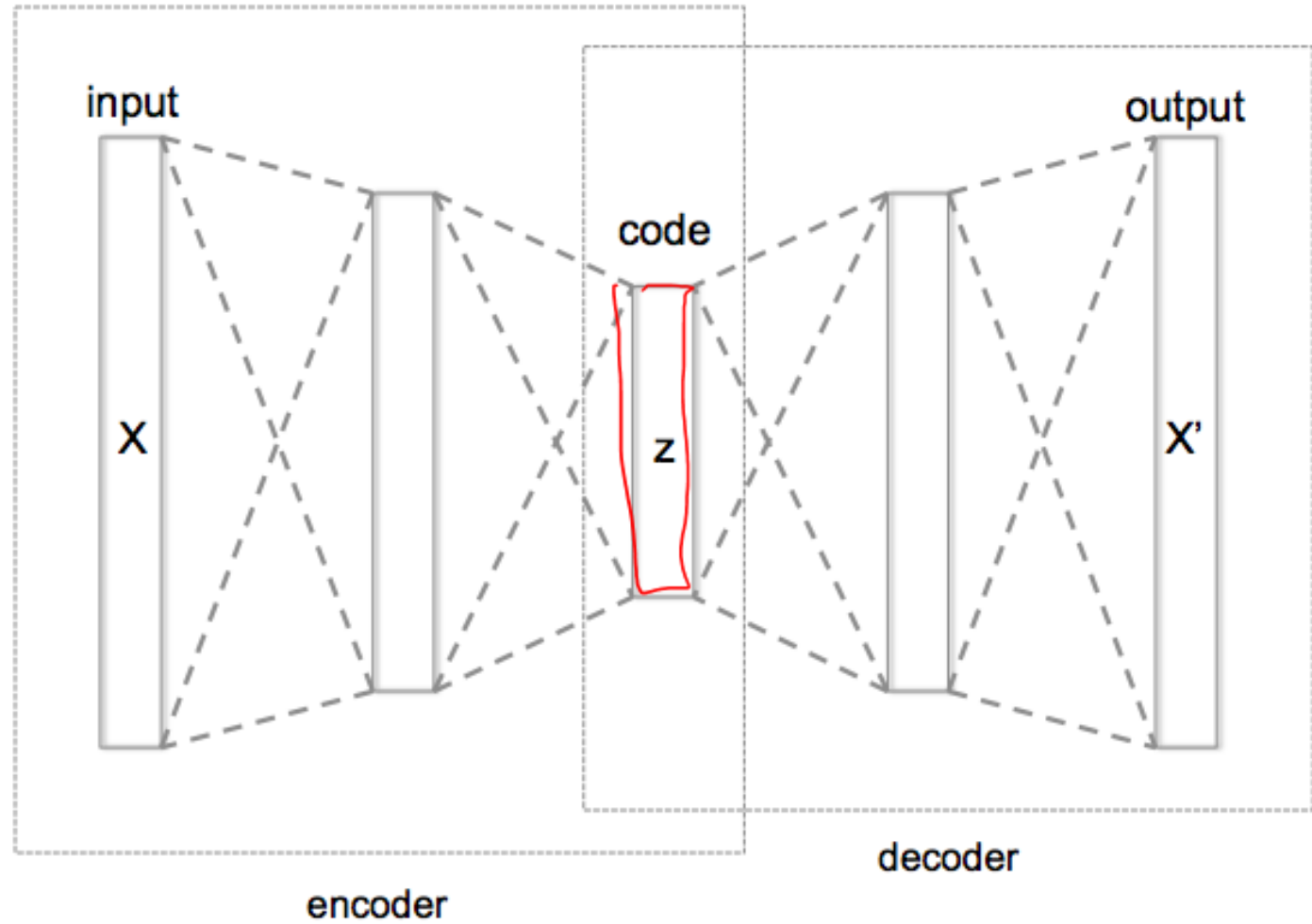
- Learn the weights by minimizing the reconstruction loss:

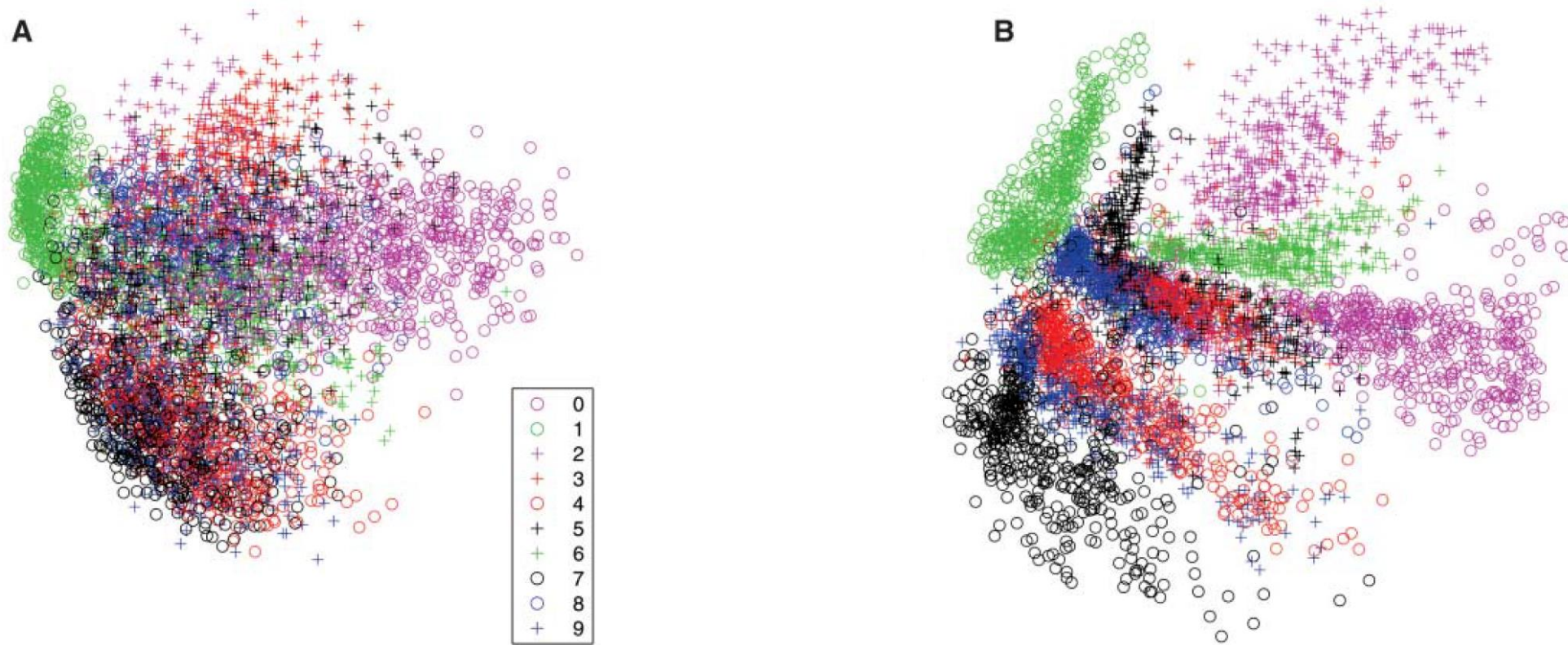
$$e(\mathbf{x}) = \|\mathbf{x} - \mathbf{o}^{(L)}\|_2^2$$

# Autoencoders



# Deep Autoencoders





# PCA (A) vs. Autoencoders (B) (Hinton and Salakhutdinov, 2014)

# Key Takeaways

- PCA finds an orthonormal basis where the first principal component maximizes the variance  $\Leftrightarrow$  minimizes the reconstruction error
- Autoencoders use neural networks to automatically learn a latent representation that minimizes the reconstruction error