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: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and $z^{(1)}, \ldots, z^{(N)}$
- Write down an objective function
  - within-cluster sum of squares (WCSS)

$$\sum_{i=1}^{N} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_{z^{(i)}} \right\|_2$$

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

K-means Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{N}, K$

1. Initialize cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$
2. While NOT CONVERGED
   a. Assign each data point to the cluster with the nearest cluster center:

$$z^{(i)} = \underset{k}{\operatorname{argmin}} \left\| \boldsymbol{x}^{(i)} - \boldsymbol{\mu}_k \right\|_2$$

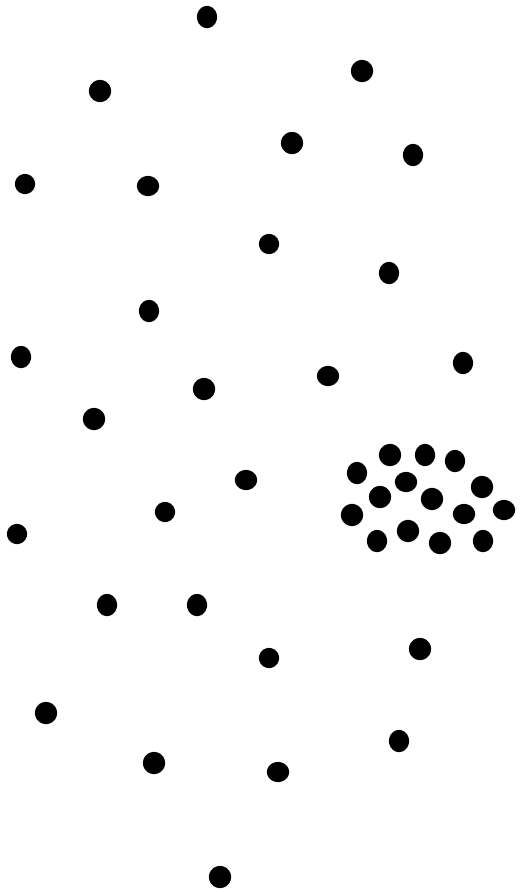
   b. Recompute the cluster centers:

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

   where N_k is the number of data points in cluster k

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

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

Gaussian Mixture Models (GMMs)

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

1. Sample a cluster at random:
   $p\left(z_k^{(i)} = 1\right) = \pi_k$
2. Sample a data point from the chosen cluster:
   $p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1\right) \sim N(\mu_k, \Sigma_k)$

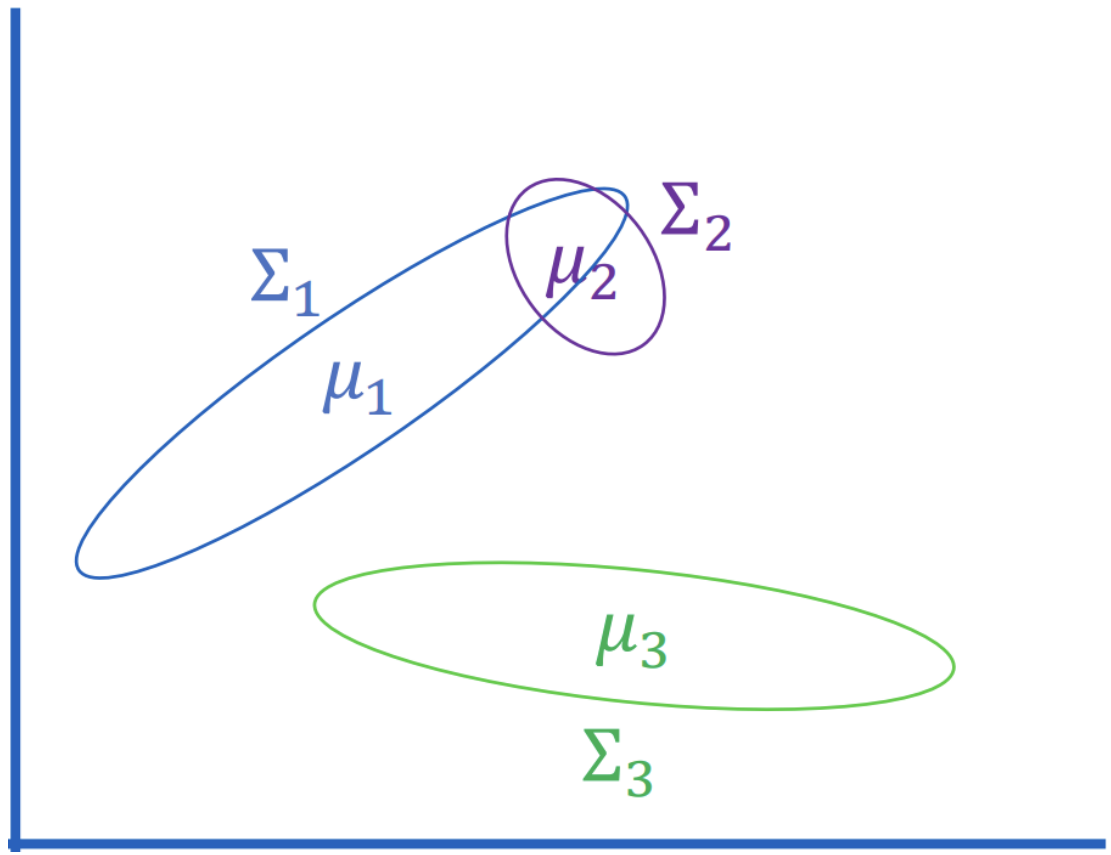
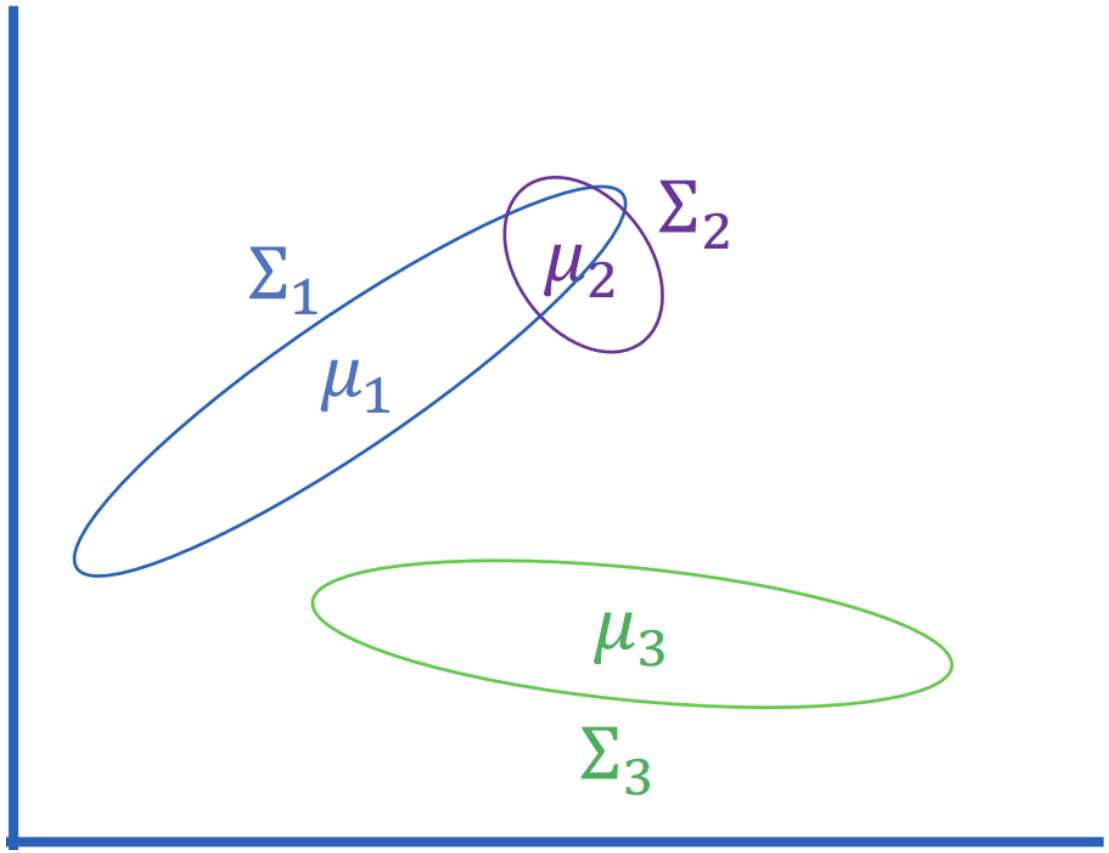


Figure courtesy of Pat Virtue

Gaussian Mixture Models (GMMs)

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

1. Sample a cluster at random:
   $p\left(z_k^{(i)} = 1\right) = \pi_k$
2. Sample a data point from the chosen cluster:
   $p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1\right) \sim N(\mu_k, \Sigma_k)$



Let $\theta = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K\}$

Figure courtesy of Pat Virtue

Maximizing the Likelihood?

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

$\ell_{\mathcal{D}}(\theta) =$

Maximizing the Likelihood?

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

$$\ell_{\mathcal{D}}(\theta) = \log \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} \middle| \theta\right) = \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} \middle| \theta\right)$$

$$= \sum_{i=1}^{N} \log p\left(\boldsymbol{x}^{(i)} \middle| \boldsymbol{z}^{(i)}, \theta\right) + \log p\left(\boldsymbol{z}^{(i)} \middle| \theta\right)$$

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

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \log p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1, \theta\right) + \sum_{k=1}^{K} z_k^{(i)} \log p\left(z_k^{(i)} = 1 \middle| \theta\right)$$

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

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

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

$$\ell_{\mathcal{D}}(\theta) = \log \prod_{i=1}^{N} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} | \theta) = \sum_{i=1}^{N} \log p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} | \theta)$$

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

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

$$= \sum_{i=1}^{N} \sum_{k=1}^{K} z_k^{(i)} \log p\left(\boldsymbol{x}^{(i)} \middle| z_k^{(i)} = 1, \theta\right) + \sum_{k=1}^{K} z_k^{(i)} \log p\left(z_k^{(i)} = 1 \middle| \theta\right)$$

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

- Parameters decoupled → set partial derivatives equal to 0

Maximizing the Marginal Likelihood

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

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

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

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

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

- Parameters coupled and *constrained* → gradient ascent is possible but complicated and slow to converge

Recipe for GMMs

- Define a model and model parameters
  - Assume K Gaussian clusters
  - Parameters: $\theta = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K\}$
- Write down an objective function
  - Maximize the log marginal likelihood

$$\ell_{\mathcal{D}}(\theta) = \log \prod_{i=1}^{N} p\left(\boldsymbol{x}^{(i)} \middle| \theta\right)$$

- Optimize the objective w.r.t. the model parameters
  - Expectation-maximization

Expectation-Maximization for GMMs: Intuition

- Insight: if we knew the cluster assignments, $\boldsymbol{z}^{(i)}$, we could maximize the log complete likelihood instead of the log marginal likelihood
- Idea: replace $\boldsymbol{z}^{(i)}$ in the log complete likelihood with our "best guess" for $\boldsymbol{z}^{(i)}$ 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 θ, compute the *expected* assignment vectors conditioned on θ and the data set $\mathcal{D}$

$$E\left[z_k^{(i)} | \boldsymbol{x}^{(i)}, \theta\right] = p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \; \forall \, i \text{ and } k$$

  - **Maximization or M-step:** for fixed assignment vectors $\boldsymbol{z}^{(i)}$, set the parameters θ 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

E-Step for GMMs

$$p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) =$$

E-Step for GMMs

$$p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) = \frac{p\left(z_k^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}{p(\boldsymbol{x}^{(i)} | \theta)}$$

$$= \frac{p\left(z_k^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}{\sum_{j=1}^{K} p\left(z_j^{(i)} = 1, \boldsymbol{x}^{(i)} | \theta\right)}$$

$$= \frac{\pi_k N\left(\boldsymbol{x}^{(i)}; \mu_k, \Sigma_k\right)}{\sum_{j=1}^{K} \pi_j N\left(\boldsymbol{x}^{(i)}; \mu_j, \Sigma_j\right)} \quad \forall\, i \text{ and } k$$

M-Step for GMMs

Let $N_k = \sum_{i=1}^{N} p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right)$

$\pi_k =$

$\mu_k =$

$\Sigma_k =$

M-Step for GMMs

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

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

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

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

GMM Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}\right)\right\}_{i=1}^{N}, K$

1. Initialize all parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K$
2. While NOT CONVERGED
   a. E-step: compute $p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \forall\ i \text{ and } k$
   b. M-step: update the parameters

- Output: parameters $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma_1, \ldots, \Sigma_K, \pi_1, \ldots, \pi_K$ and assignments probabilities $p\left(z_k^{(i)} = 1 \middle| \boldsymbol{x}^{(i)}, \theta\right) \forall\ i \text{ and } k$

Initializing EM for GMMs

- Common heuristics for initialization
  - Cluster proportions typically initialized to be uniform
  - Cluster means
    - Randomly select data points to be cluster centers
    - Randomly sample locations in the range spanned by the data
  - Cluster covariances
    - Identity (or scaled identity) matrix
    - 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)

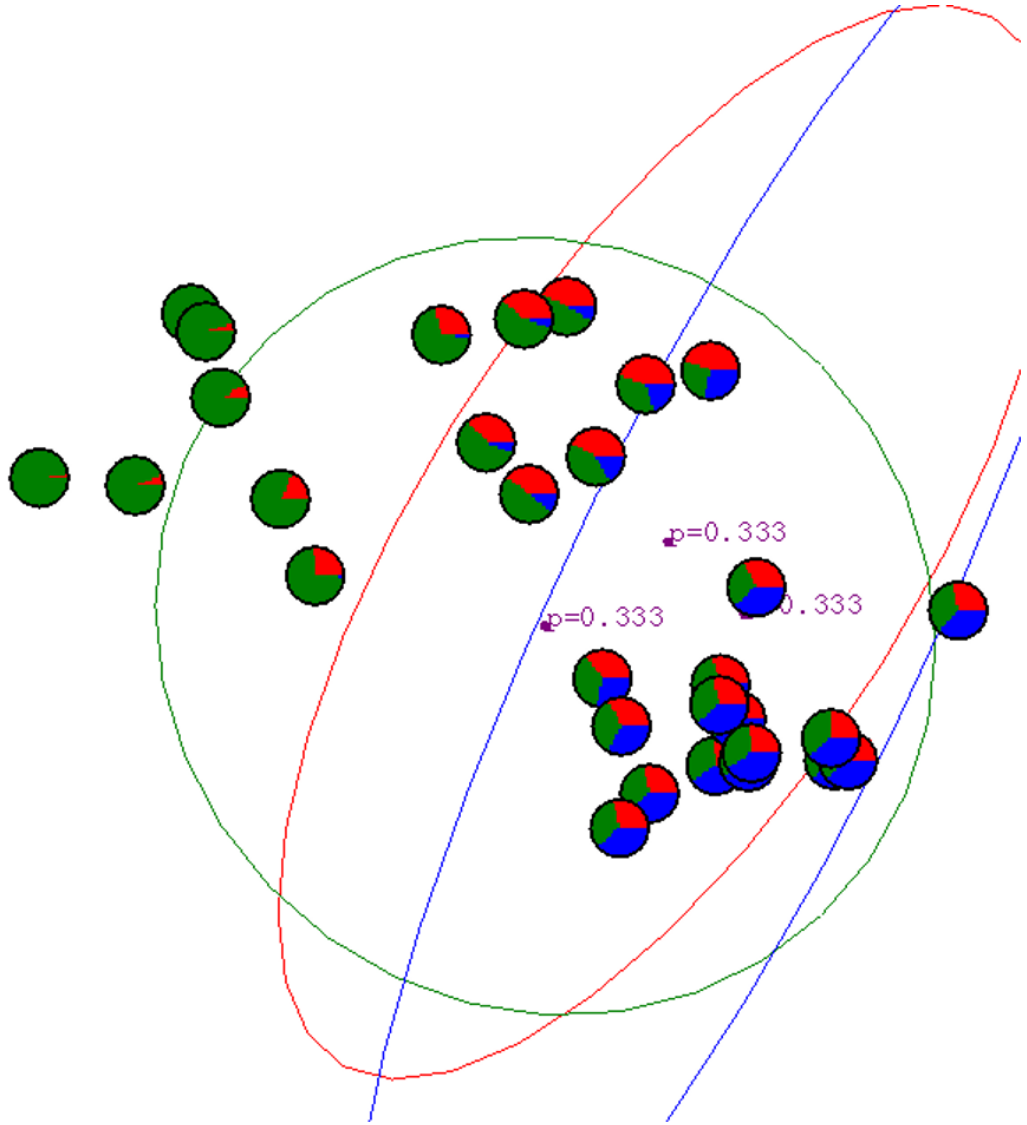


Figure courtesy of Pat Virtue

GMMs: Example (1 Iteration)

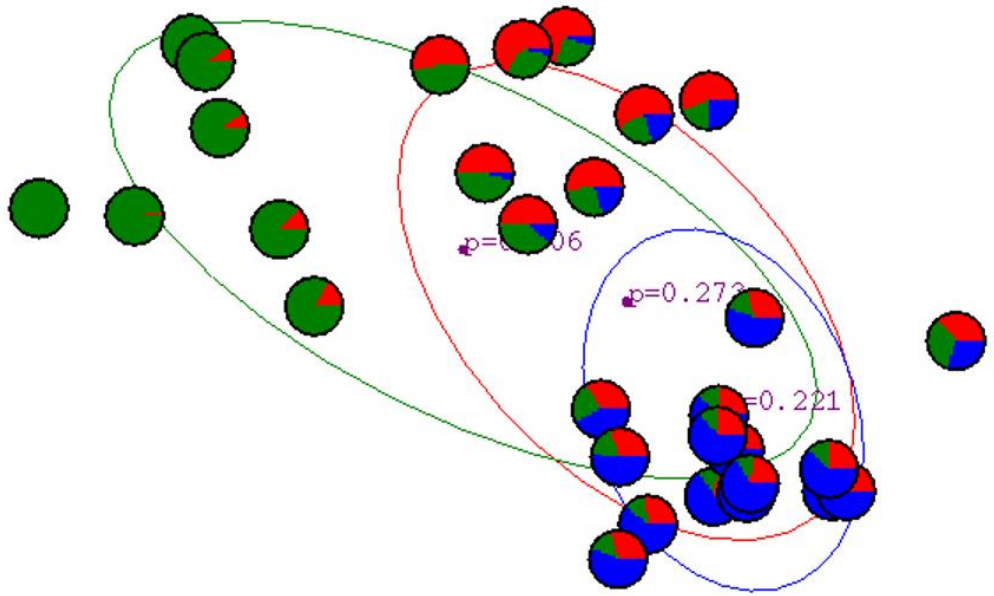


Figure courtesy of Pat Virtue

GMMs: Example (2 Iterations)

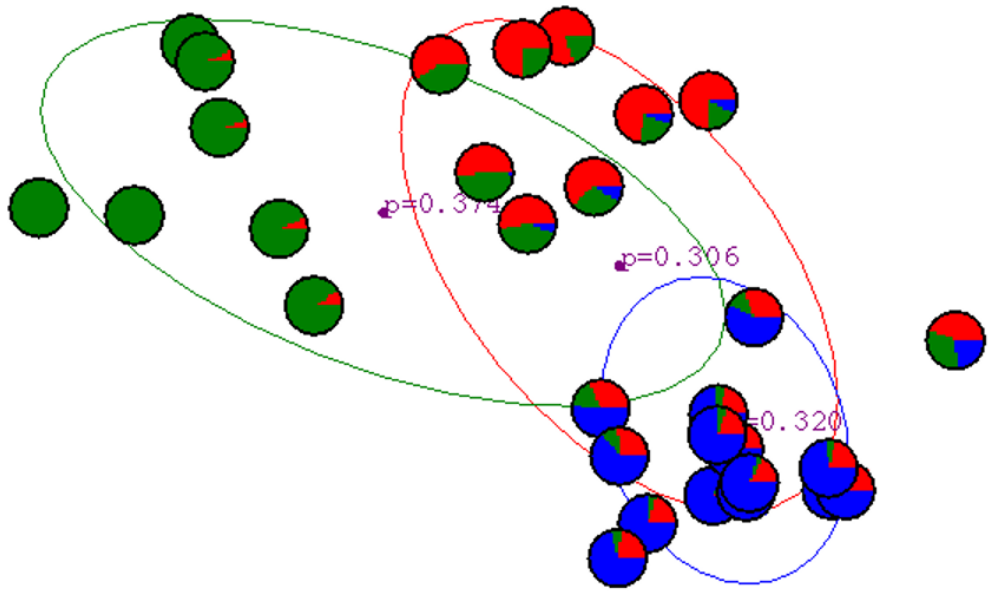


Figure courtesy of Pat Virtue

GMMs: Example (3 Iterations)

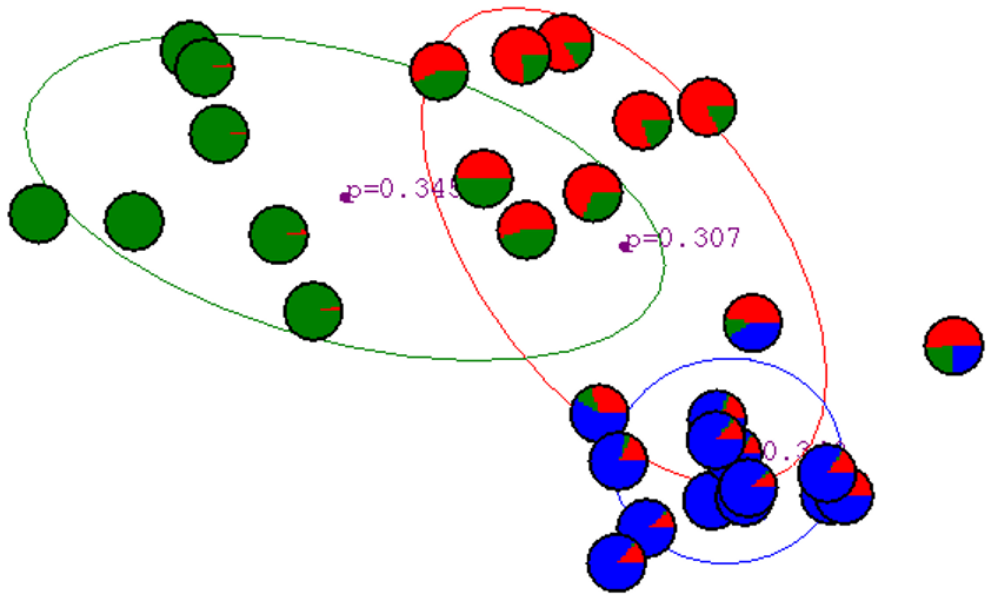


Figure courtesy of Pat Virtue

GMMs: Example (4 Iterations)

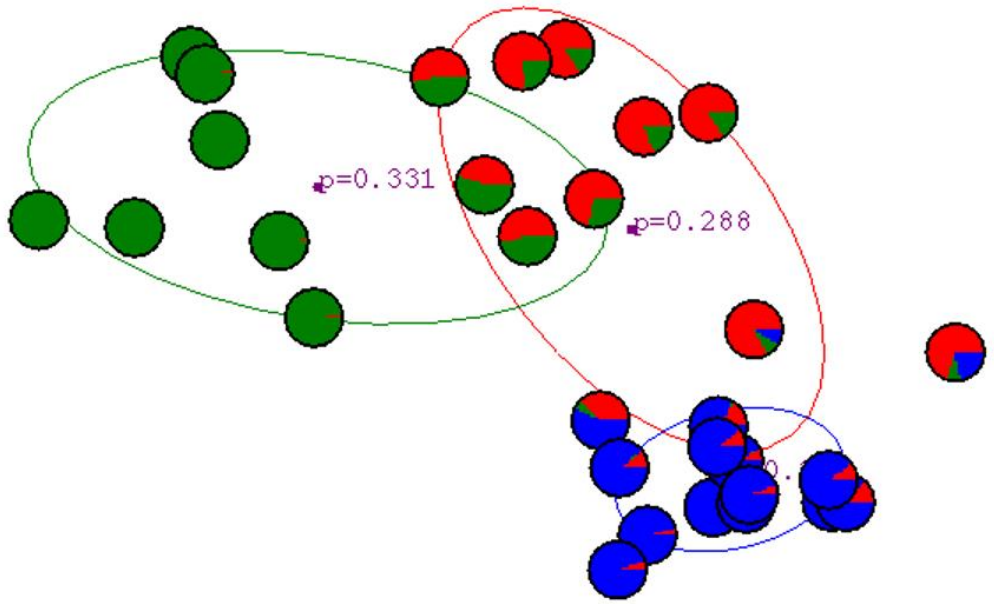


Figure courtesy of Pat Virtue

GMMs: Example (5 Iterations)

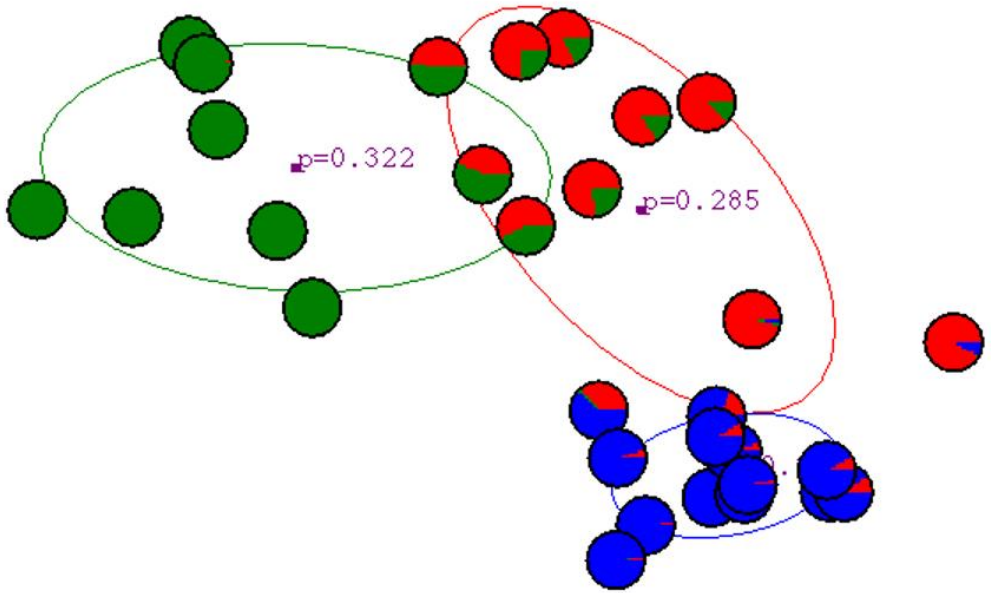


Figure courtesy of Pat Virtue

GMMs: Example (6 Iterations)

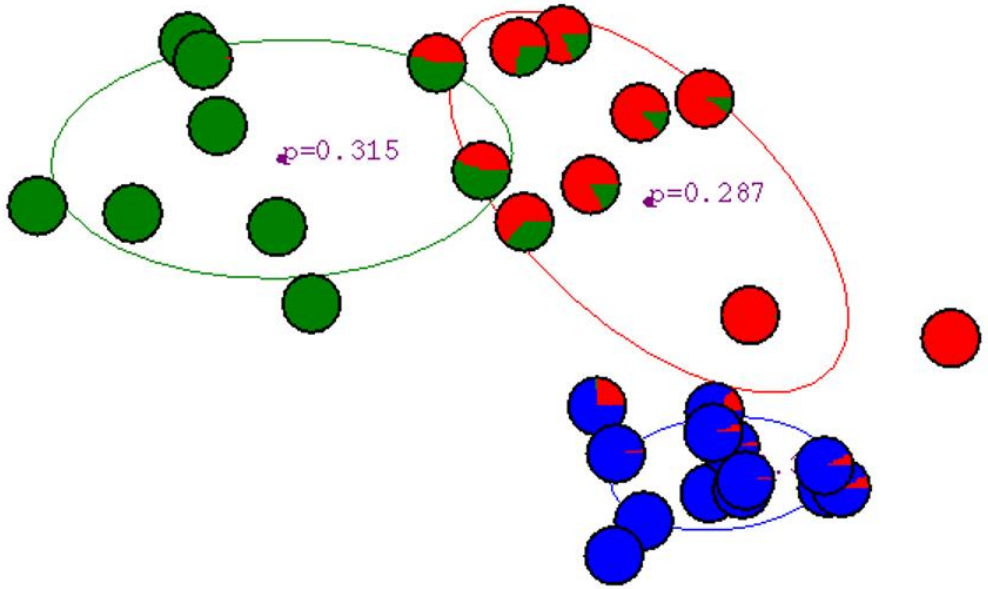


Figure courtesy of Pat Virtue

GMMs: Example (20 Iterations)

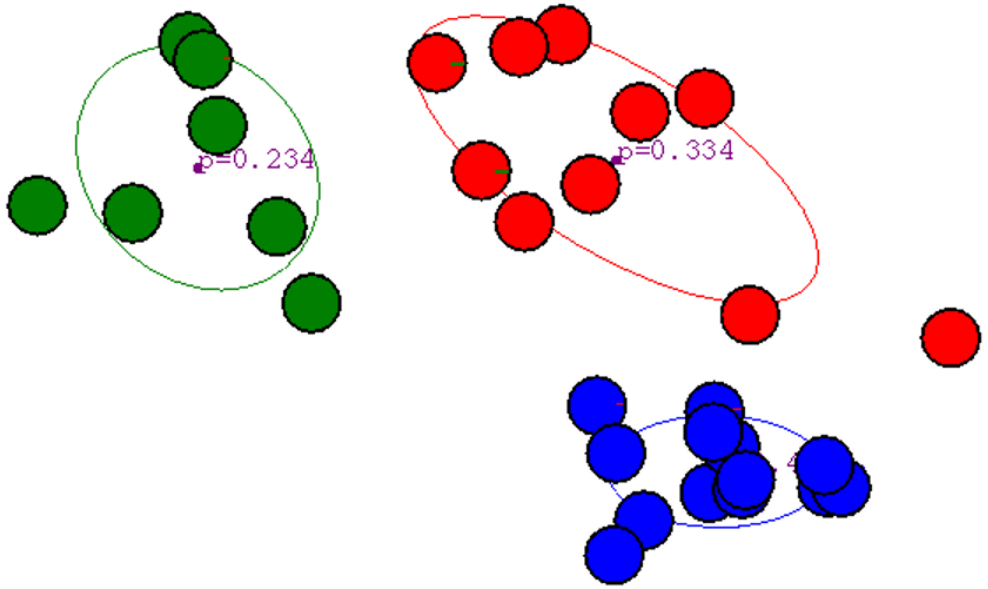


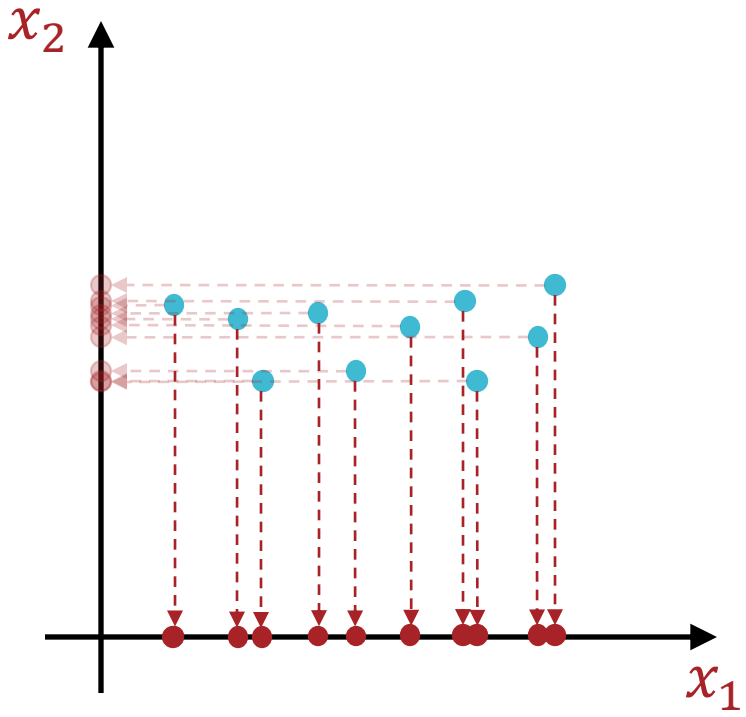
Figure courtesy of Pat Virtue

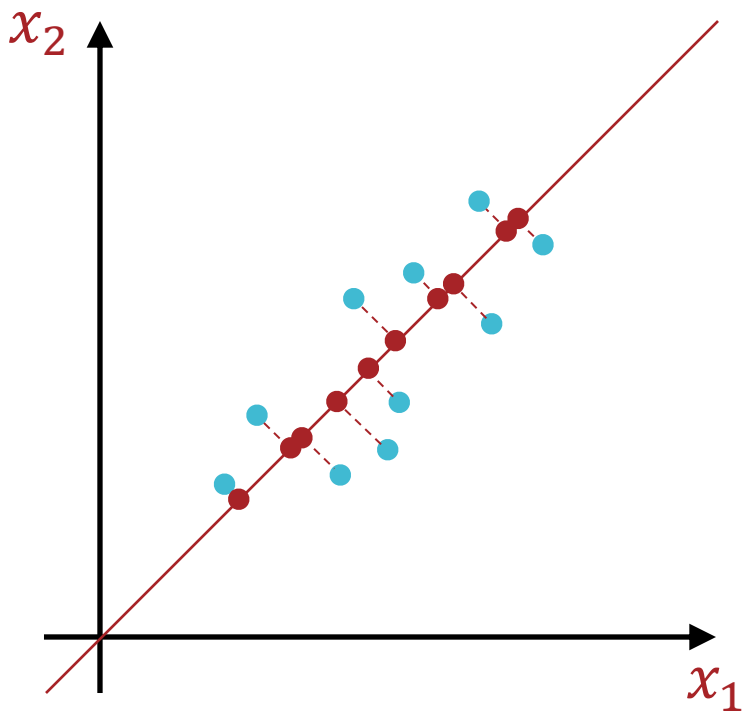
Key Takeaways

- 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

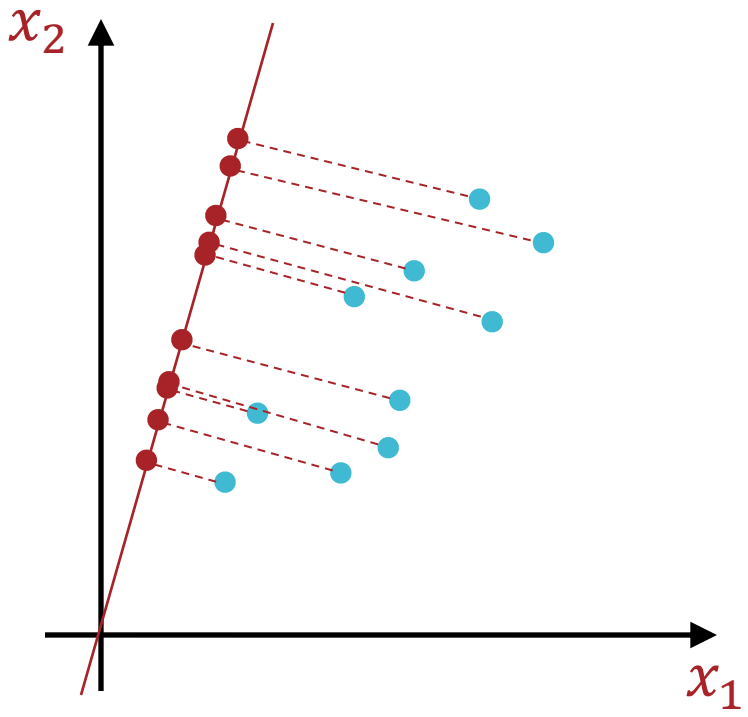
Unsupervised Learning

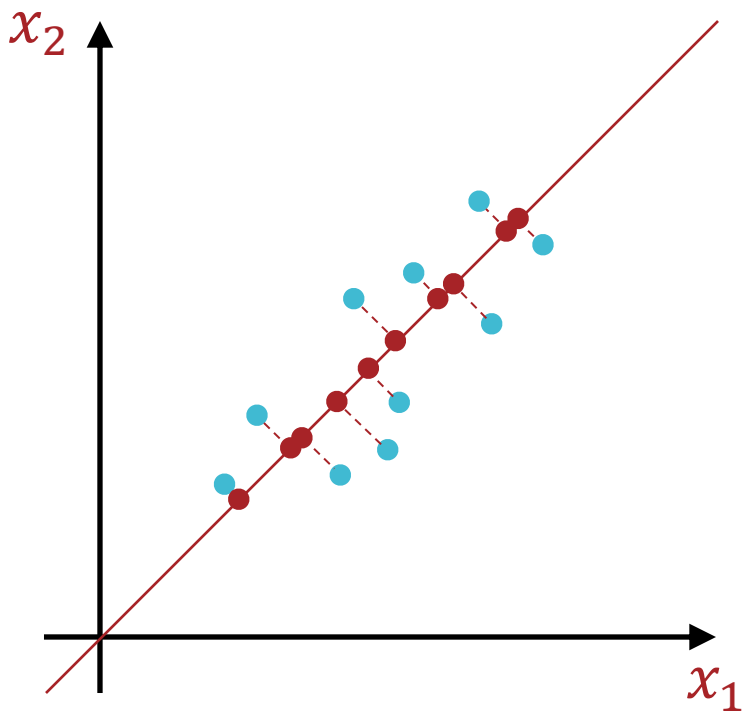
- Clustering: split an unlabeled data set into groups or partitions of “similar” data points
  - Use cases:
    - Organizing data
    - Discovering patterns or structure
    - Preprocessing for downstream tasks
- Dimensionality Reduction: given some unlabeled data set, learn a latent (typically lower-dimensional)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



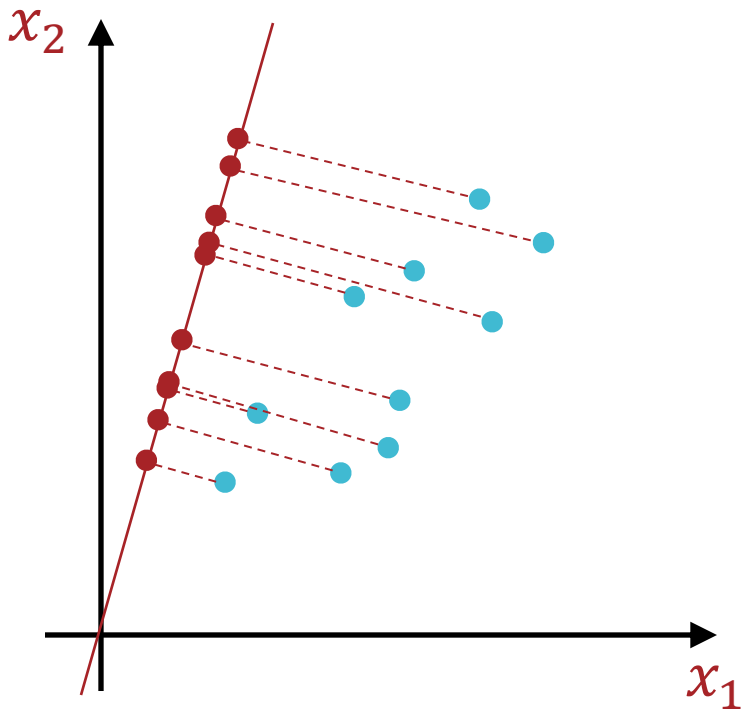


Feature Elimination

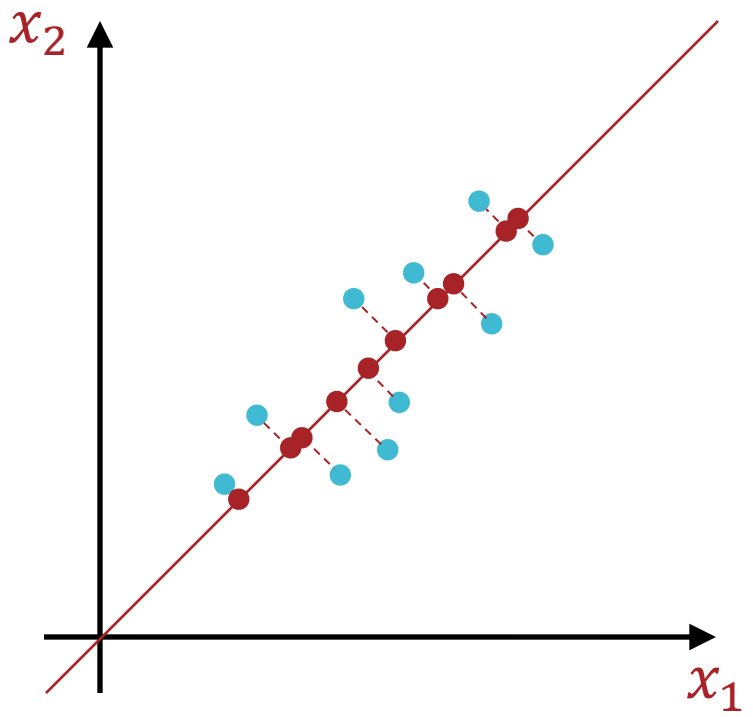




Feature Reduction



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. $\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}^{(n)}$
2. $\tilde{\boldsymbol{x}}^{(n)} = \boldsymbol{x}^{(n)} - \boldsymbol{\mu}\ \forall\ n$
3. $X = \begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)^T} \\ \tilde{\boldsymbol{x}}^{(2)^T} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(N)^T} \end{bmatrix}$

Computing Projections

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

$$\boldsymbol{z}^{(n)} = \left(\frac{\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}}{\|\boldsymbol{v}\|_2} \right) \frac{\boldsymbol{v}}{\|\boldsymbol{v}\|_2}$$

Length of projection

Direction of projection

Reconstruction Error

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

$$\boldsymbol{z}^{(n)} = \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}$$

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

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

$$=$$

Reconstruction Error

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

$$\boldsymbol{z}^{(n)} = \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}$$

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

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

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

$$= {\tilde{\boldsymbol{x}}^{(n)}}^T \tilde{\boldsymbol{x}}^{(n)} - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}$$

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

Minimizing the Reconstruction Error

⇕

Maximizing the Variance

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

$$= \underset{\boldsymbol{v}:\|\boldsymbol{v}\|_2^2=1}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| \tilde{\boldsymbol{x}}^{(n)} \right\|_2^2 - \left(\boldsymbol{v}^T \tilde{\boldsymbol{x}}^{(n)}\right)^2$$

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

← Variance of projections ($\tilde{\boldsymbol{x}}^{(n)}$ are centered)

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

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

Maximizing the Variance

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

$$\mathcal{L}(\boldsymbol{v}, \lambda) =$$

Maximizing the Variance

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

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

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

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

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

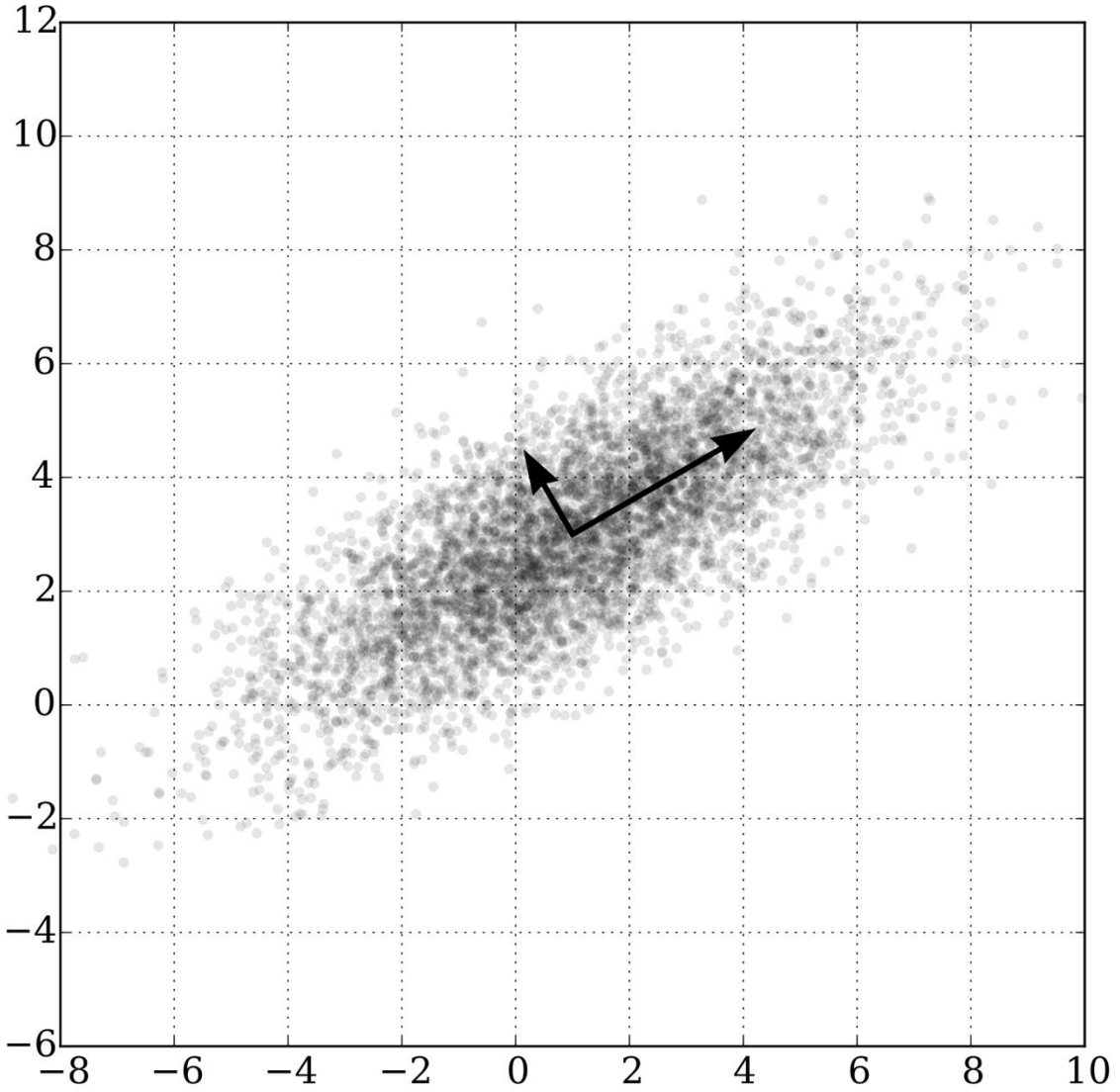
Maximizing the Variance

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

$$(X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}} \rightarrow \hat{\boldsymbol{v}}^T (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}^T\hat{\boldsymbol{v}} = \lambda$$

- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_1
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthogonal
- Etc …
- λ_i is a measure of how much variance falls along $\hat{\boldsymbol{v}}_i$

Principal Components: Example



Source: https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:GaussianScatterPCA.svg

PCA Algorithm

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}\right)\right\}_{n=1}^{N}, \rho$

1. Center the data
2. Compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = XV_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

How many PCs should we use?

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}\right)\right\}_{n=1}^{N}, \rho$

1. Center the data
2. Compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top ρ eigenvectors (corresponding to the ρ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by V_ρ, $Z = XV_\rho$

- Output: Z, the transformed (potentially lower-dimensional) data

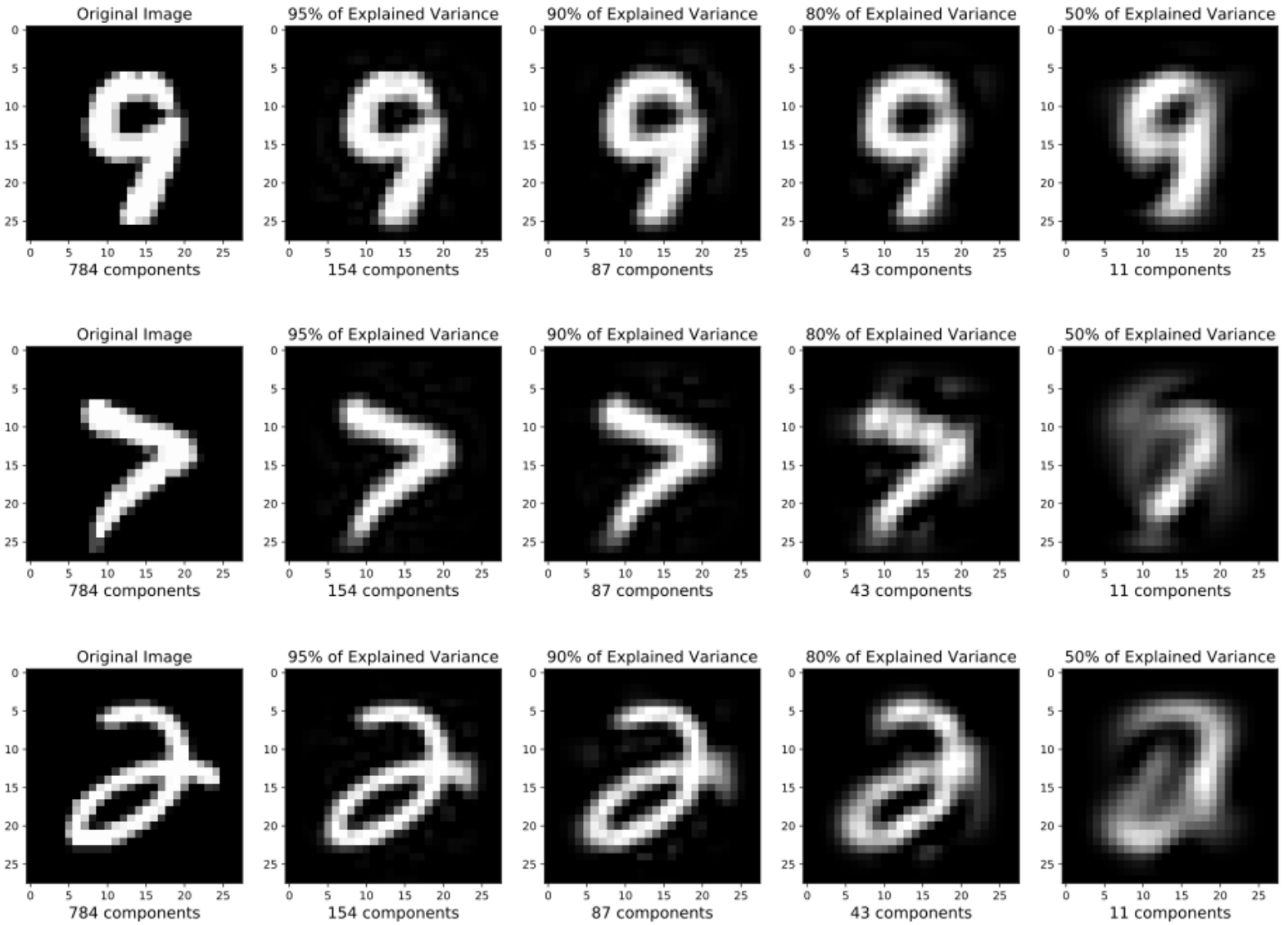
Choosing the number of PCs

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

$$\lambda_i \Big/ \sum \lambda_j$$

- 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



Figures courtesy of Matt Gormley

PCA Example: MNIST Digits

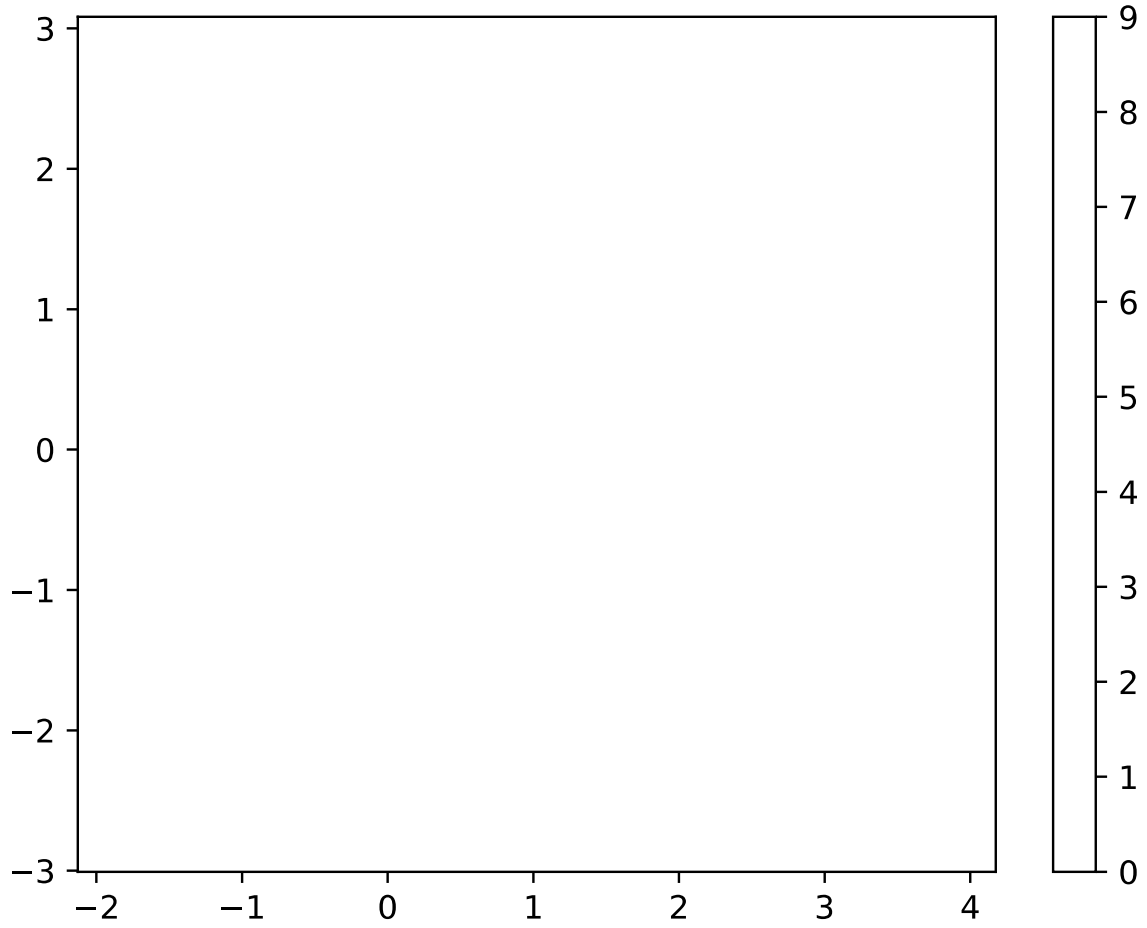


Figure courtesy of Matt Gormley

PCA Example: MNIST Digits

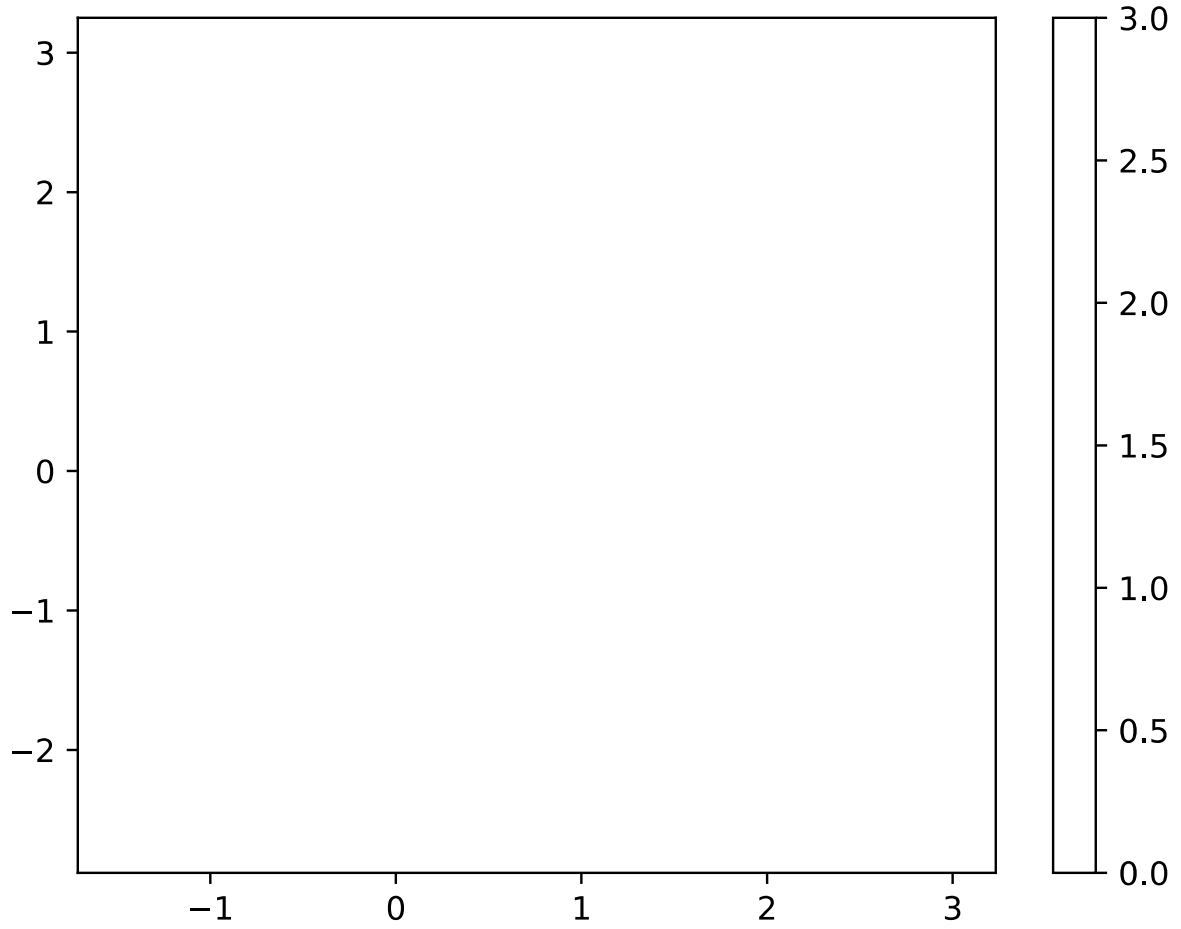
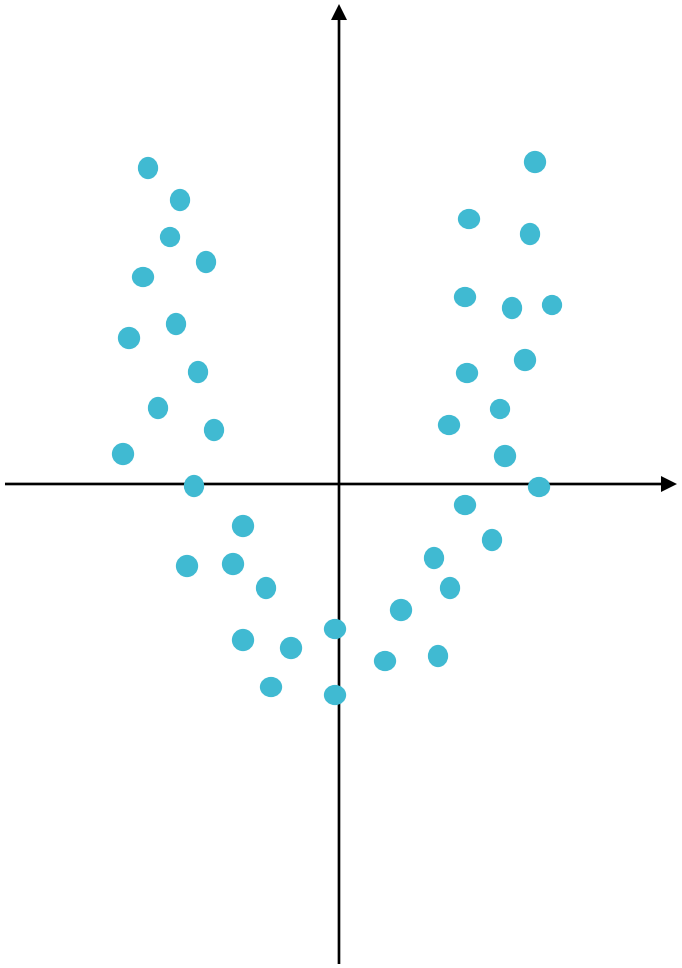


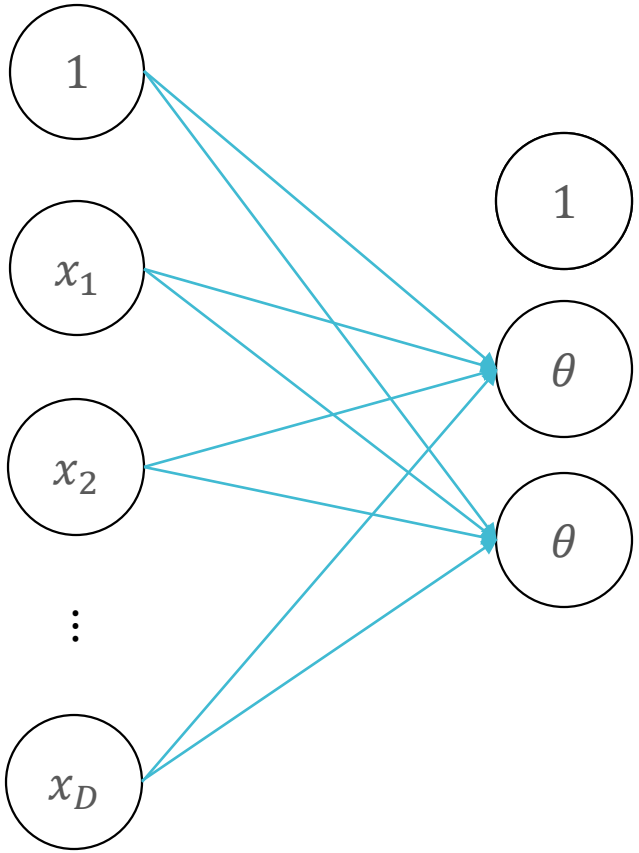
Figure courtesy of Matt Gormley

Shortcomings of PCA



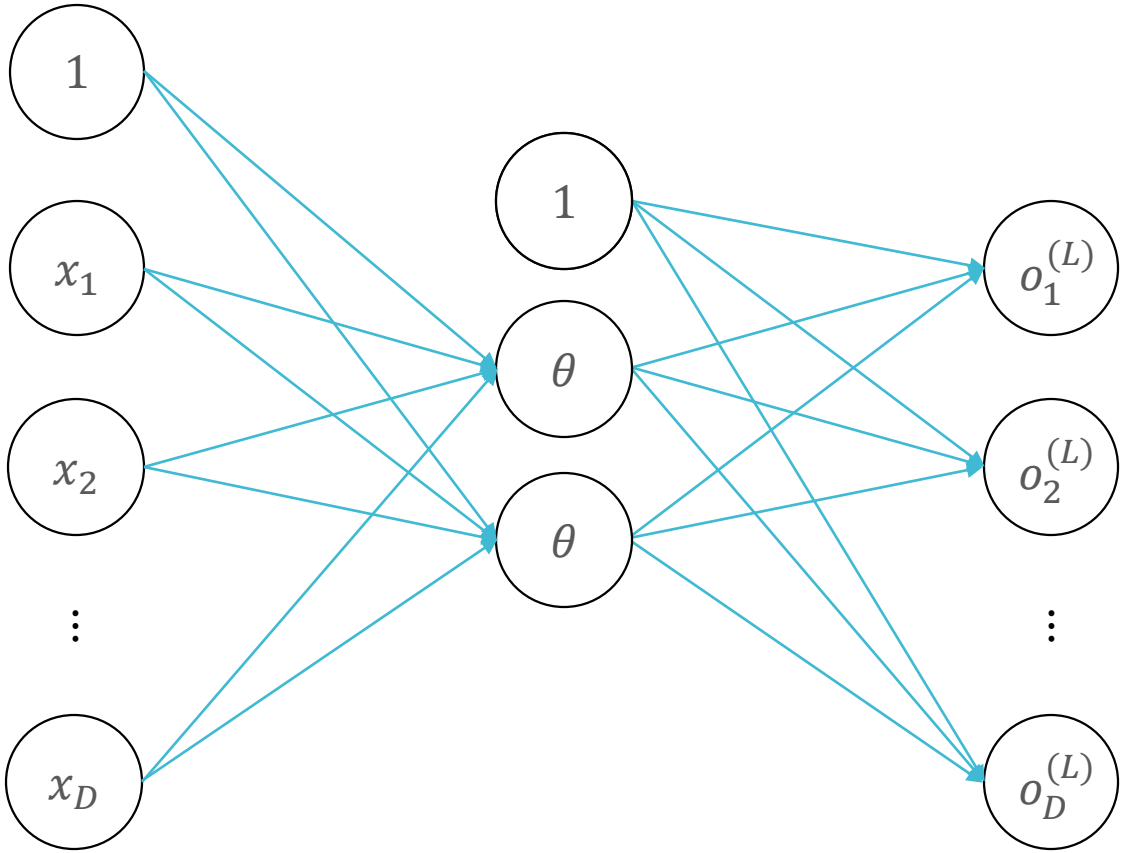
- Principal components are orthogonal (unit) vectors
- Principal components are expressed as **linear** combinations of the data

Autoencoders



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

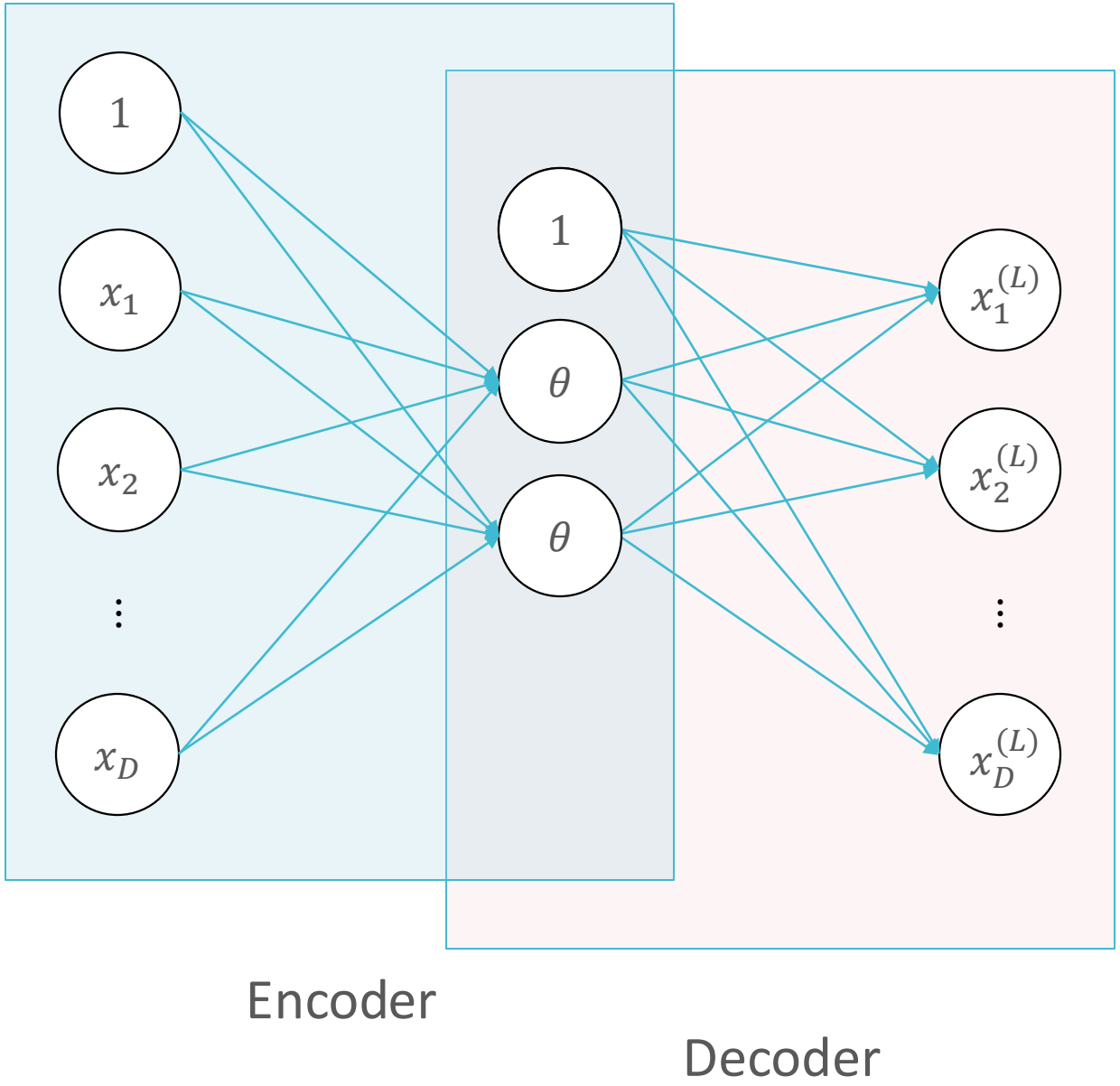
Autoencoders



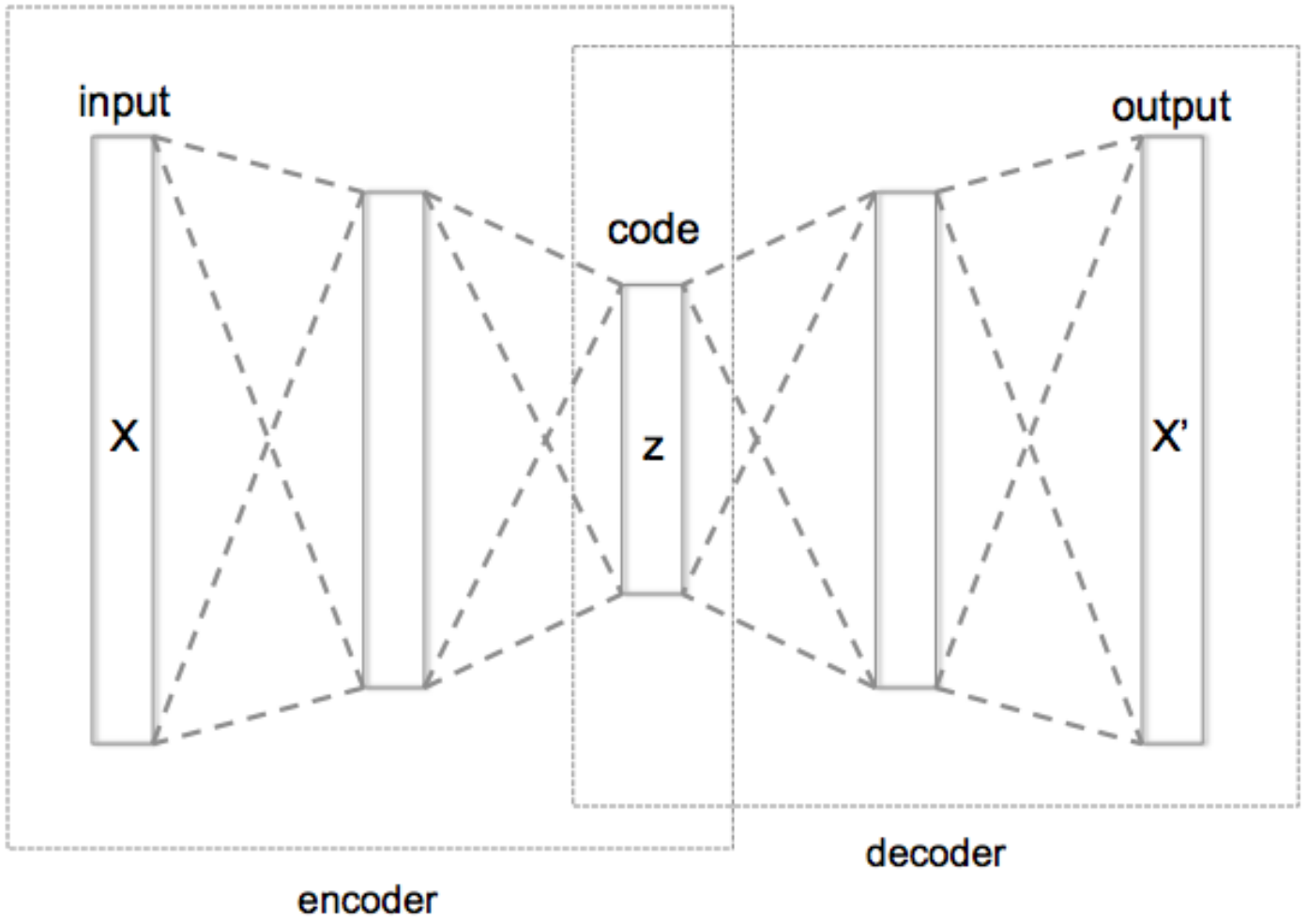
- Learn the weights by minimizing the reconstruction loss:

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

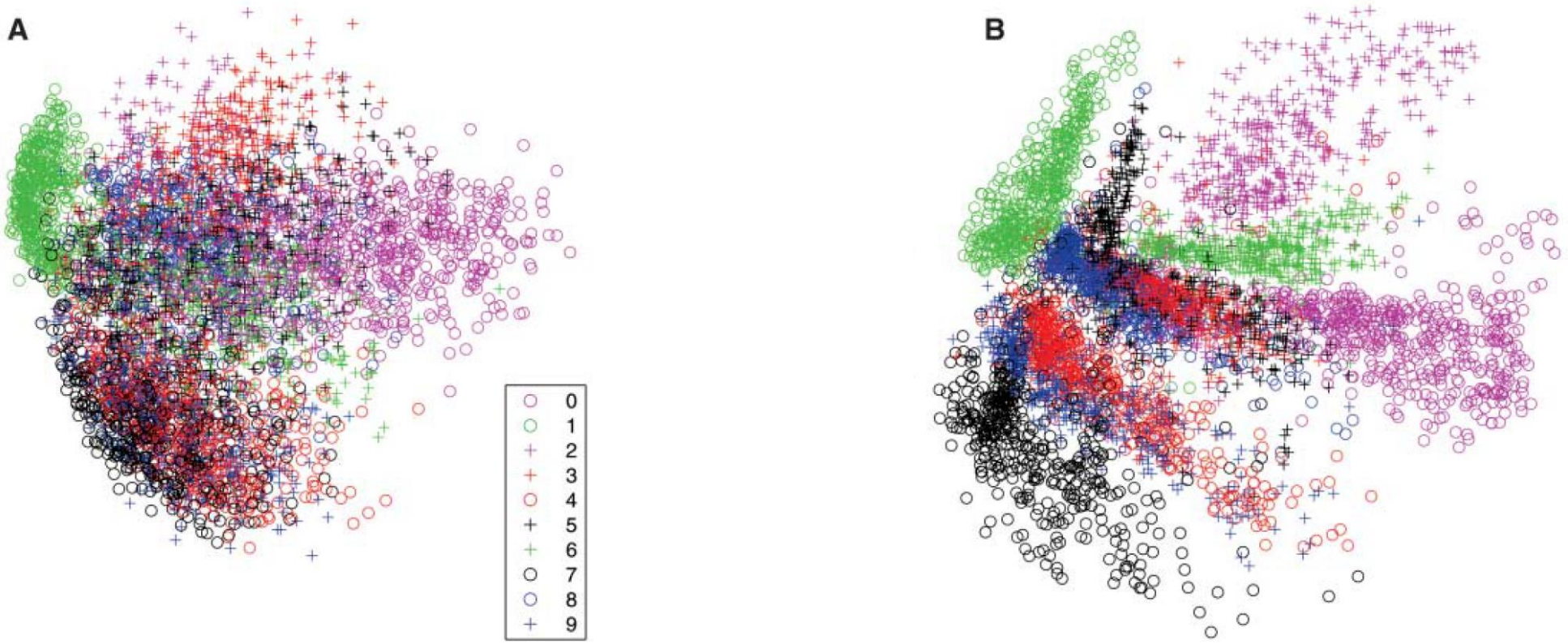
Autoencoders



Deep Autoencoders



Source: https://en.wikipedia.org/wiki/Autoencoder#/media/File:Autoencoder_structure.png



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

Source: https://www.science.org/doi/10.1126/science.1127647

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