

10-701: Introduction to Machine Learning

Lecture 17: Unsupervised Learning

Henry Chai & Zack Lipton

11/1/23

Front Matter

- Announcements
 - Project Proposals due 11/3 (Friday!)
 - Each group should only submit one PDF to Gradescope (see Piazza for instructions on making group submissions)
- Recommended Readings
 - Murphy, [Chapters 12.2.1 - 12.2.3](#)
 - Murphy, [Chapters 25.5.1 - 25.5.2](#)
 - Daumé III, [Chapter 15: Unsupervised Learning](#)

Learning Paradigms

- Supervised learning - $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N$
 - Regression - $y^{(n)} \in \mathbb{R}$
 - Classification - $y^{(n)} \in \{1, \dots, C\}$
- Reinforcement learning - $\mathcal{D} = \{(\mathbf{s}^{(n)}, \mathbf{a}^{(n)}, r^{(n)})\}_{n=1}^N$
- Unsupervised learning - $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^N$
 - Clustering
 - Dimensionality reduction

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

Recall: Similarity for k NN

- Intuition: ~~predict the label of a data point to be the label of the “most similar” training point~~ two points are “similar” if the distance between them is small
 - Euclidean distance: $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2$
- Partition-based clustering: Given a desired number of clusters, K , return a partition of the data set into K groups or clusters, $\{C_1, \dots, C_K\}$, that optimize some objective function

Recipe for K -means

- Define a model and model parameters
 - Assume K clusters and use the Euclidean distance
 - Parameters: $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ and $z^{(1)}, \dots, z^{(N)}$

- Write down an objective function

$$\sum_{n=1}^N \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_{z^{(n)}}\|_2$$

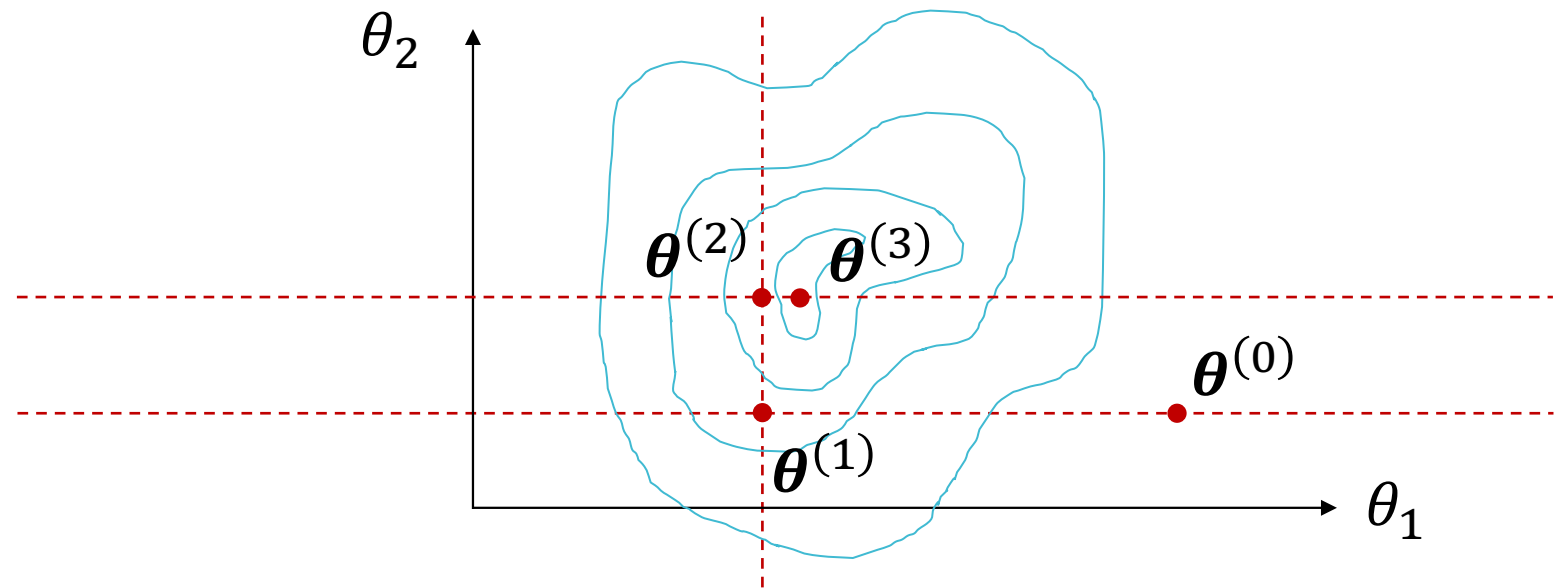
- Optimize the objective w.r.t. the model parameters
 - Use (block) coordinate descent

Coordinate Descent

- Goal: minimize some objective

$$\hat{\theta} = \operatorname{argmin} J(\theta)$$

- Idea: iteratively pick one variable and minimize the objective w.r.t. just that variable, *keeping all others fixed*.



Block Coordinate Descent

- Goal: minimize some objective

$$\hat{\alpha}, \hat{\beta} = \operatorname{argmin} J(\alpha, \beta)$$

- Idea: iteratively pick one *block* of variables (α or β) and minimize the objective w.r.t. that block, keeping the other(s) fixed.
 - Ideally, blocks should be the largest possible set of variables *that can be efficiently optimized simultaneously*

Optimizing the K -means objective

$$\hat{\boldsymbol{\mu}}_1, \dots, \hat{\boldsymbol{\mu}}_K, z^{(1)}, \dots, z^{(N)} = \operatorname{argmin} \sum_{n=1}^N \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_{z^{(n)}}\|_2$$

- If $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ are fixed

$$\hat{z}^{(n)} = \operatorname{argmin}_{k \in \{1, \dots, K\}} \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\|_2$$

- If $z^{(1)}, \dots, z^{(N)}$ are fixed

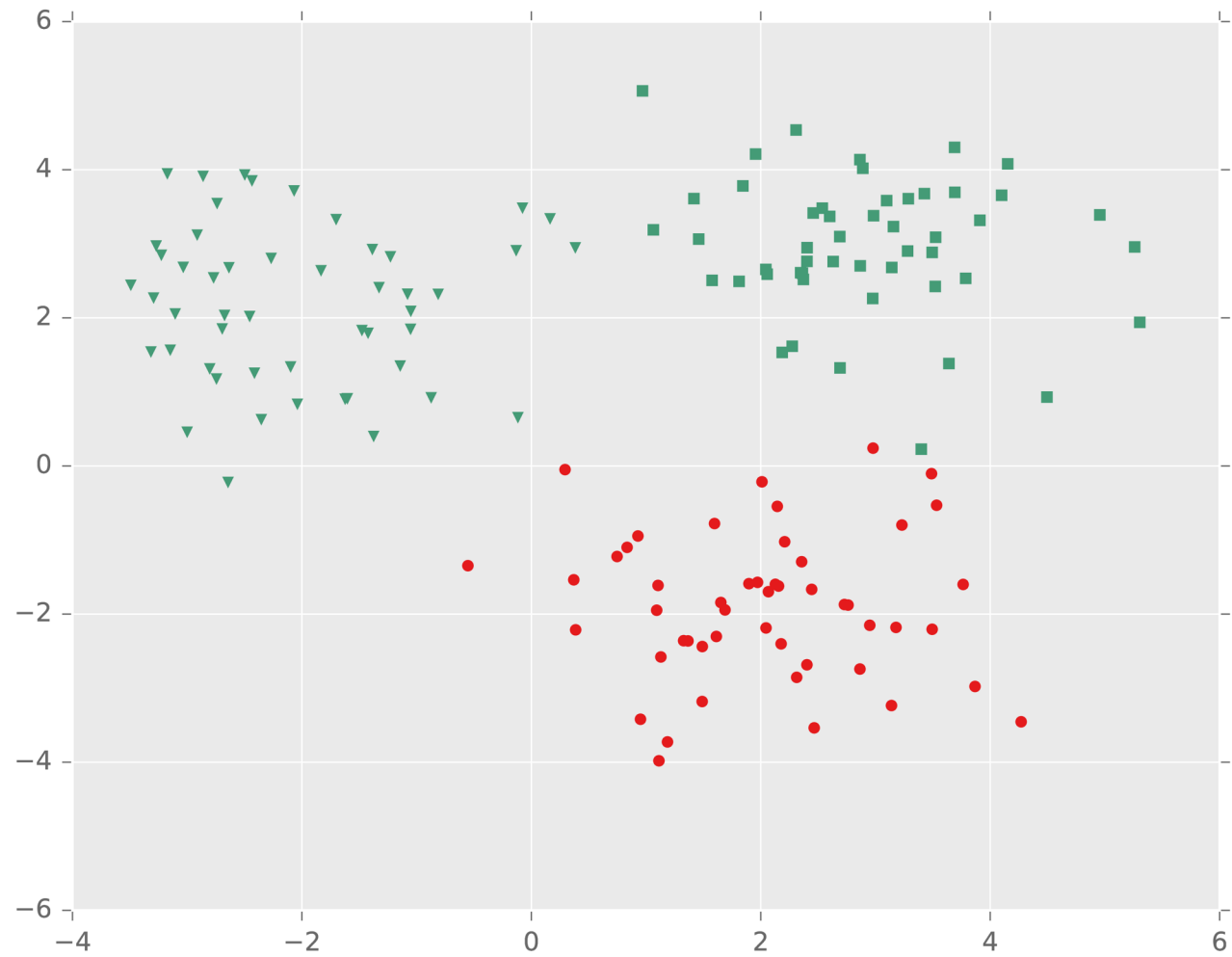
$$\hat{\boldsymbol{\mu}}_k = \operatorname{argmin}_{\boldsymbol{\mu}} \sum_{n: z^{(n)} = k} \|\mathbf{x}^{(n)} - \boldsymbol{\mu}\|_2$$

$$= \frac{1}{N_k} \sum_{n: z^{(n)} = k} \mathbf{x}^{(n)}$$

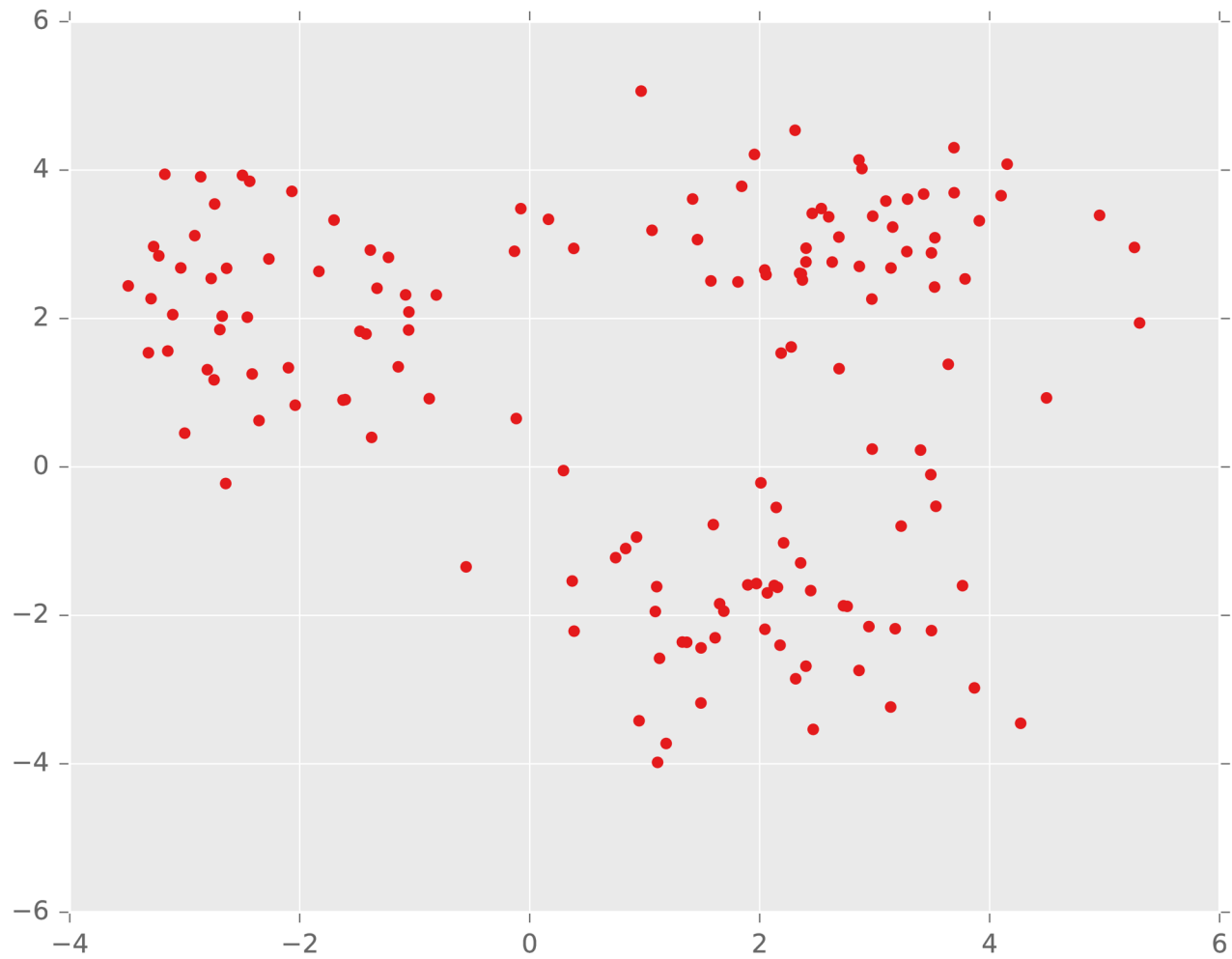
K-means Algorithm

- Input: $\mathcal{D} = \{(\mathbf{x}^{(n)})\}_{n=1}^N, K$
 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:
$$z^{(n)} = \underset{k}{\operatorname{argmin}} \|\mathbf{x}^{(n)} - \boldsymbol{\mu}_k\|_2$$
 - b. Recompute the cluster centers:
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n: z^{(n)}=k} \mathbf{x}^{(n)}$$
where N_k is the number of data points in cluster k
- Output: cluster centers $\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_K$ and cluster assignments $z^{(1)}, \dots, z^{(N)}$

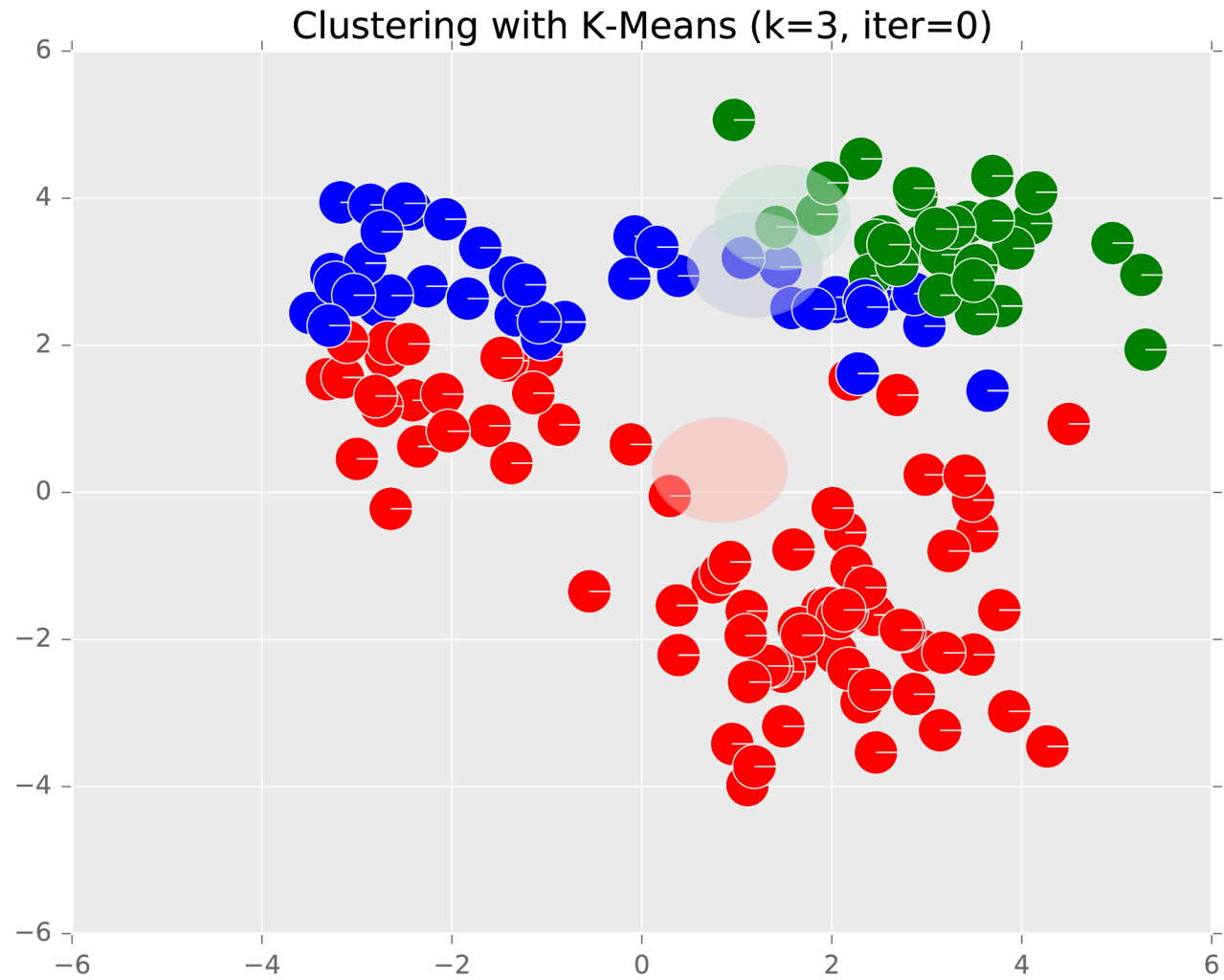
K -means: Example ($K = 3$)



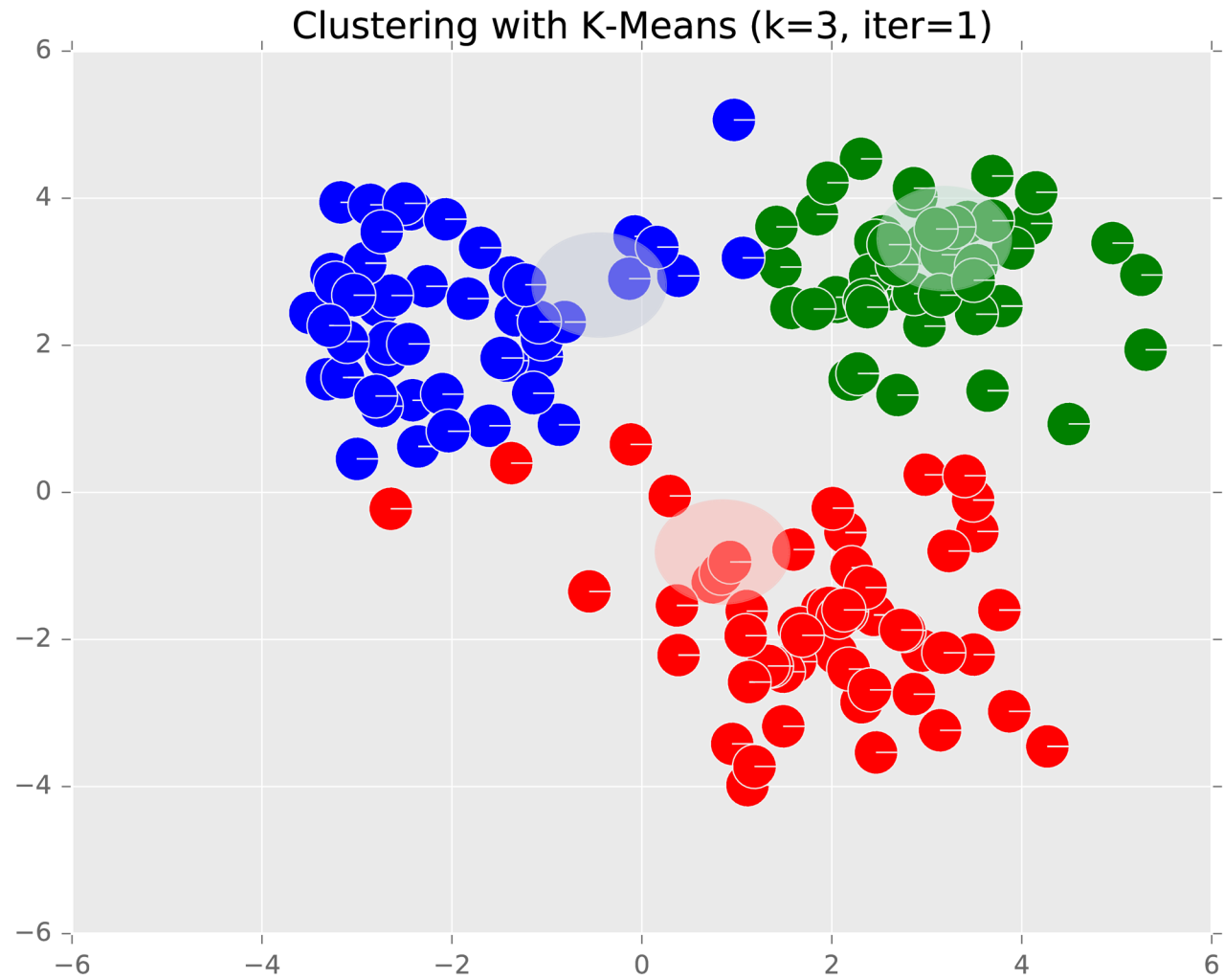
K -means: Example ($K = 3$)



K-means:
Example
($K = 3$)



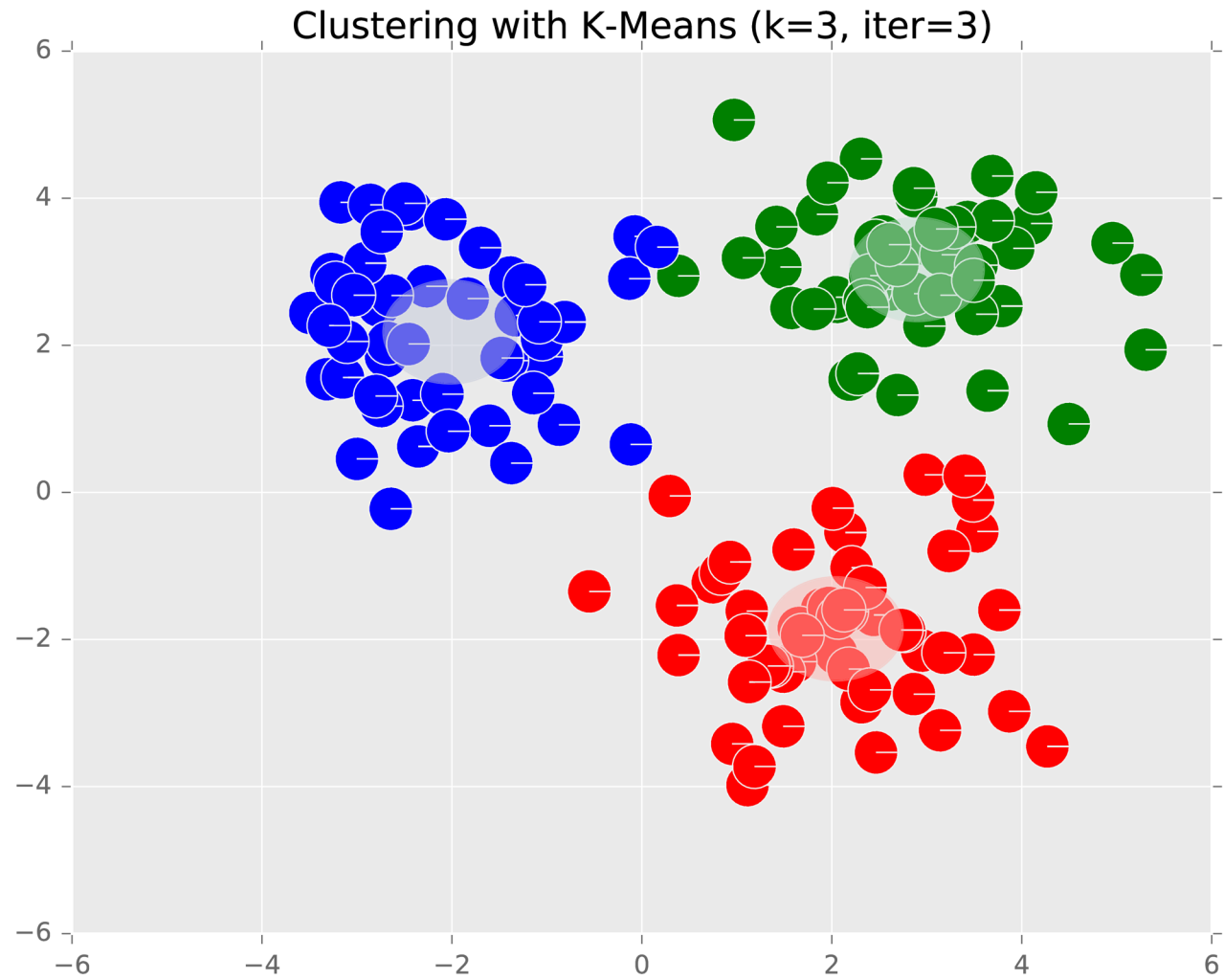
K-means:
Example
($K = 3$)



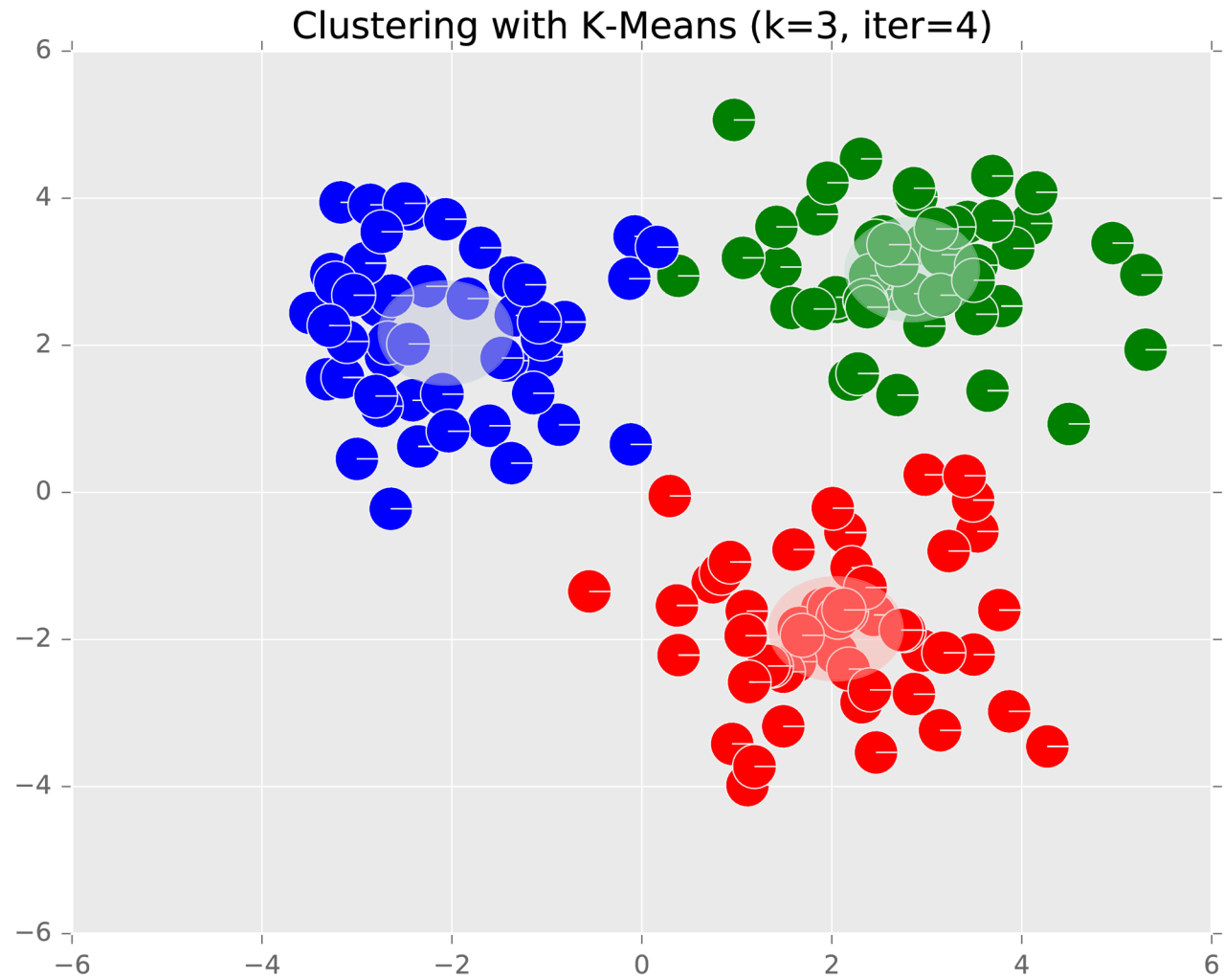
K-means:
Example
($K = 3$)



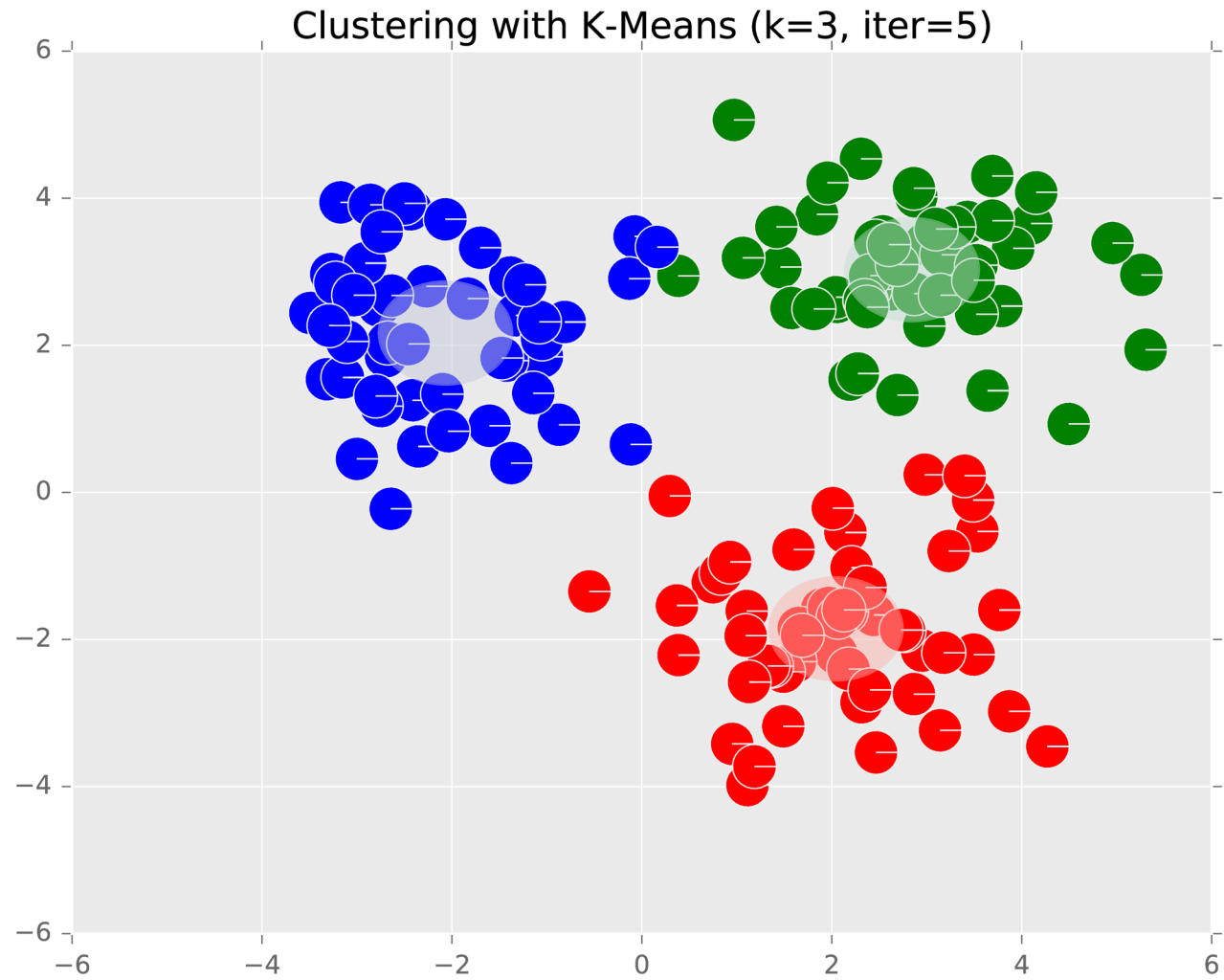
K-means:
Example
($K = 3$)



K-means:
Example
($K = 3$)

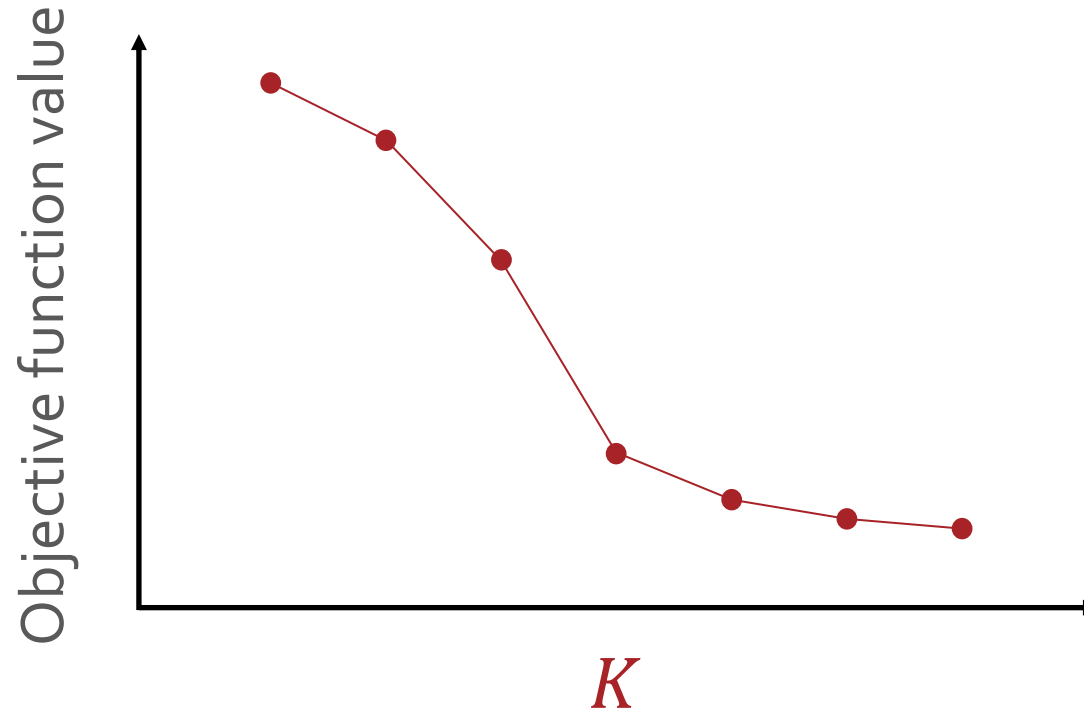


K-means:
Example
($K = 3$)



Setting K

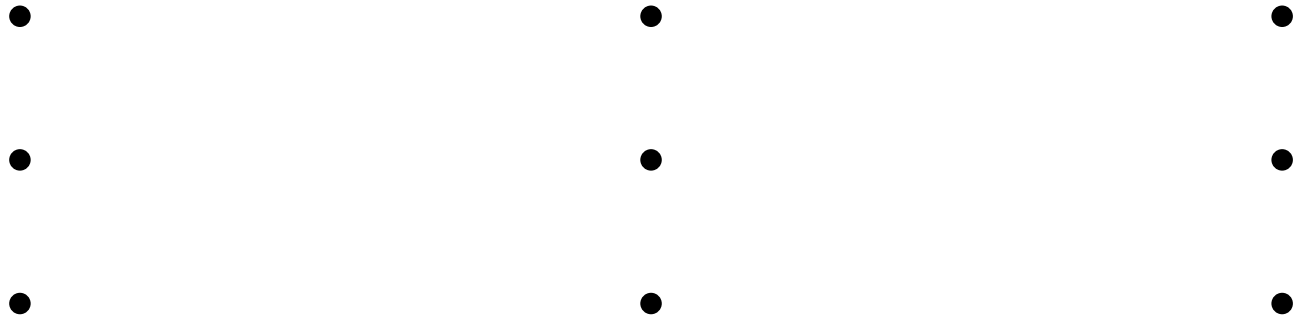
- Idea: choose the value of K that minimizes the objective function



- Look for the characteristic “elbow” or largest decrease when going from $K - 1$ to K

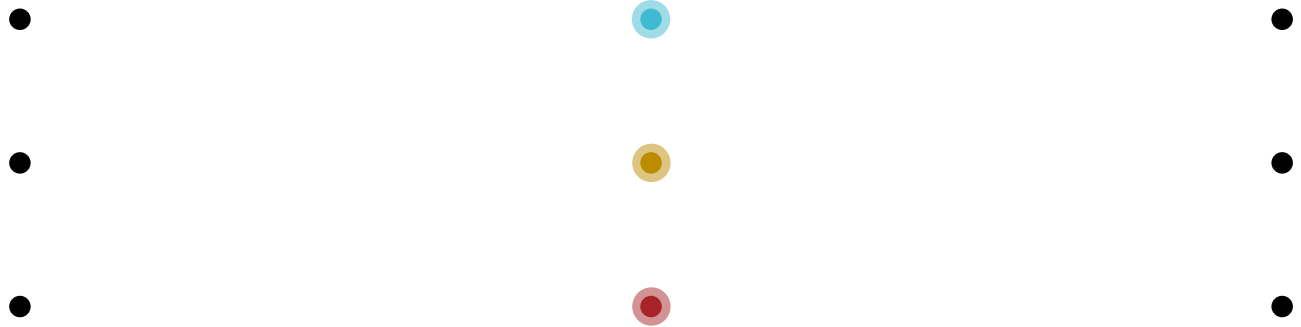
Initializing K -means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



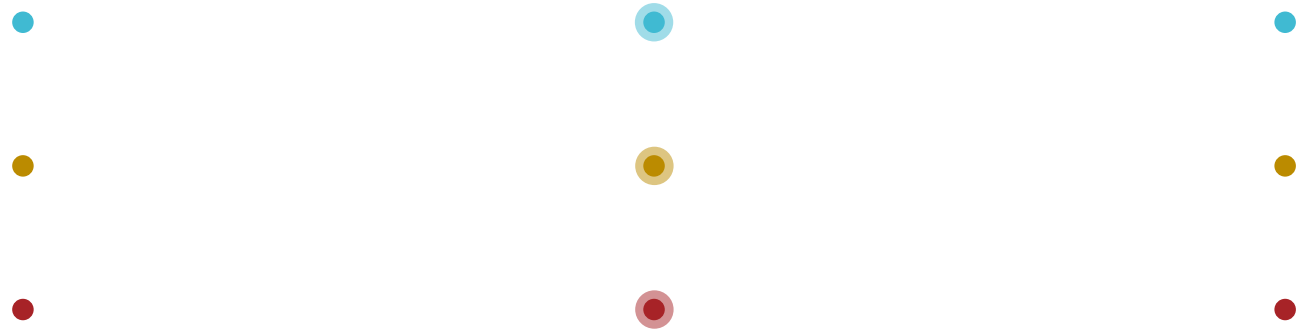
Initializing K -means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



Initializing K -means

- Common choice: choose K data points at random to be the initial cluster centers (Lloyd's method)



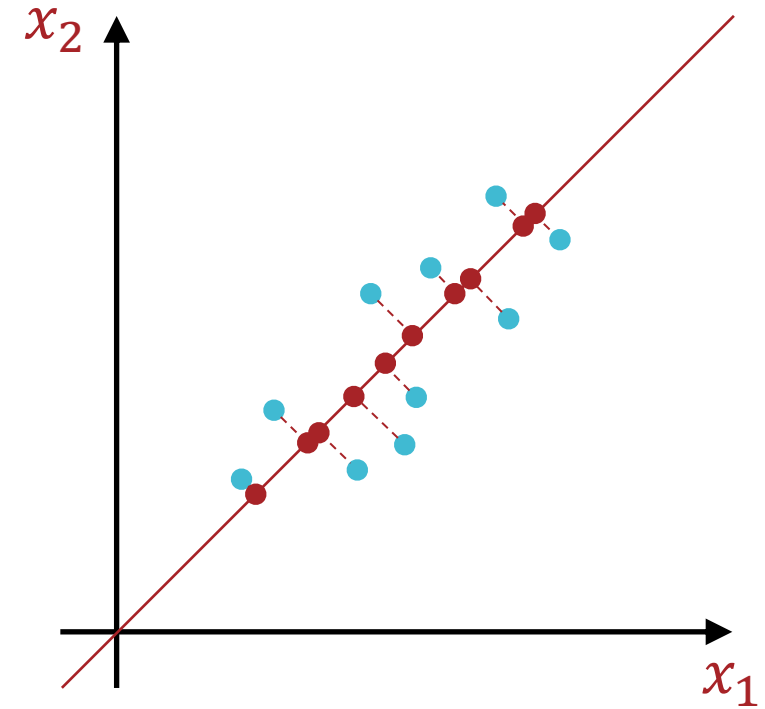
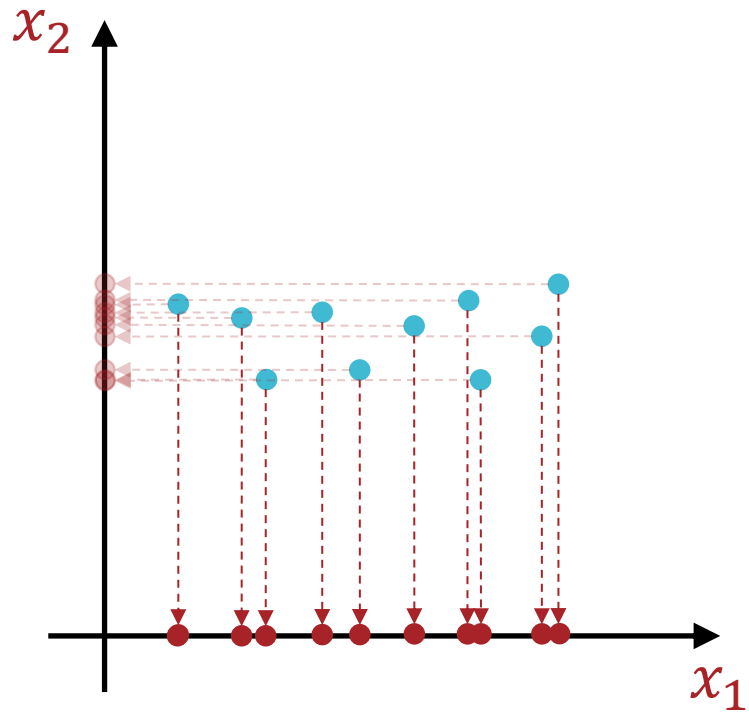
- Lloyd's method converges to a local minimum and that local minimum can be arbitrarily bad (relative to the optimal clusters)
- Intuition: want initial cluster centers to be far apart from one another

K -means++ (Arthur and Vassilvitskii, 2007)

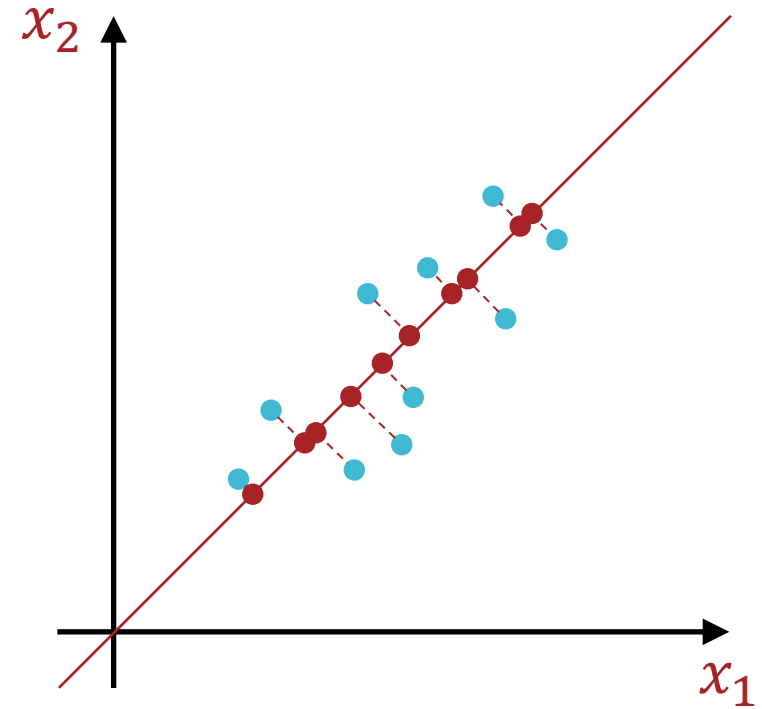
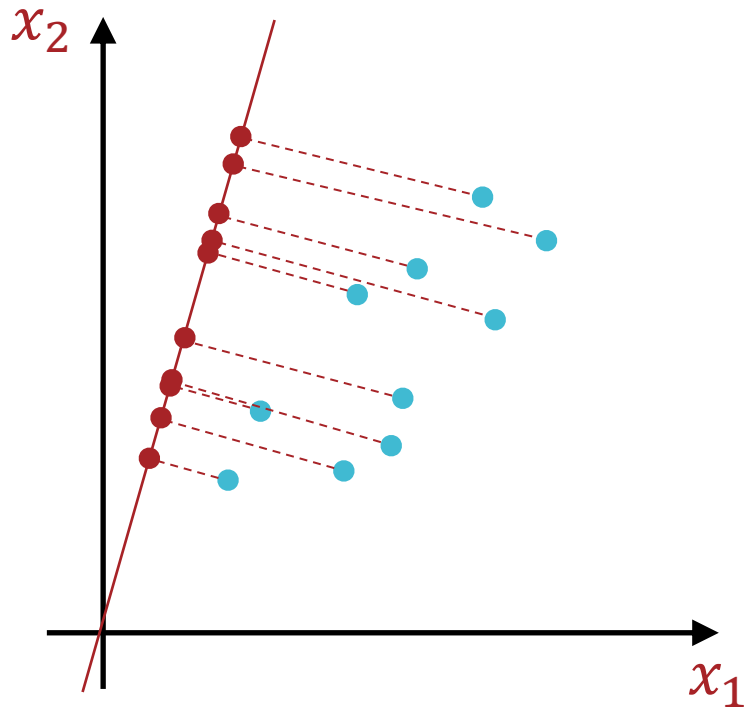
1. Choose the first cluster center randomly from the data points.
2. For each other data point \mathbf{x} , compute $D(\mathbf{x})$, the distance between \mathbf{x} and the closest cluster center.
3. Select the next cluster center proportional to $D(\mathbf{x})^2$.
4. Repeat 2 and 3 $K - 1$ times.
 - K -means++ achieves a $O(\log K)$ approximation to the optimal clustering in expectation
 - Both Lloyd's method and K -means++ can benefit from multiple random restarts.

Unsupervised Learning

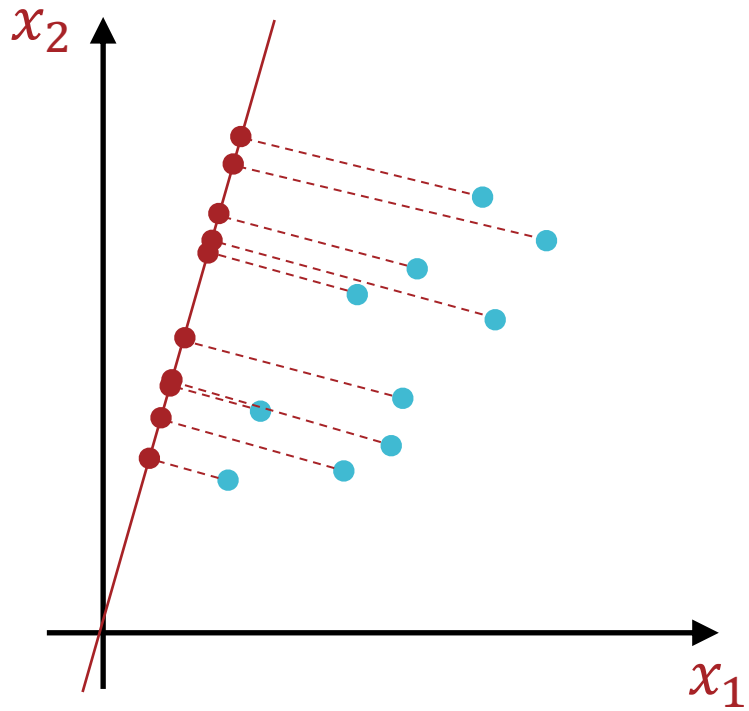
- Clustering: split an unlabeled data set into groups or partitions of “similar” data points
 - Use cases:
 - Organizing data
 - Discovering patterns or structure
 - Preprocessing for downstream tasks
- Dimensionality Reduction: given some unlabeled data set, learn a latent (typically lower-dimensional)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



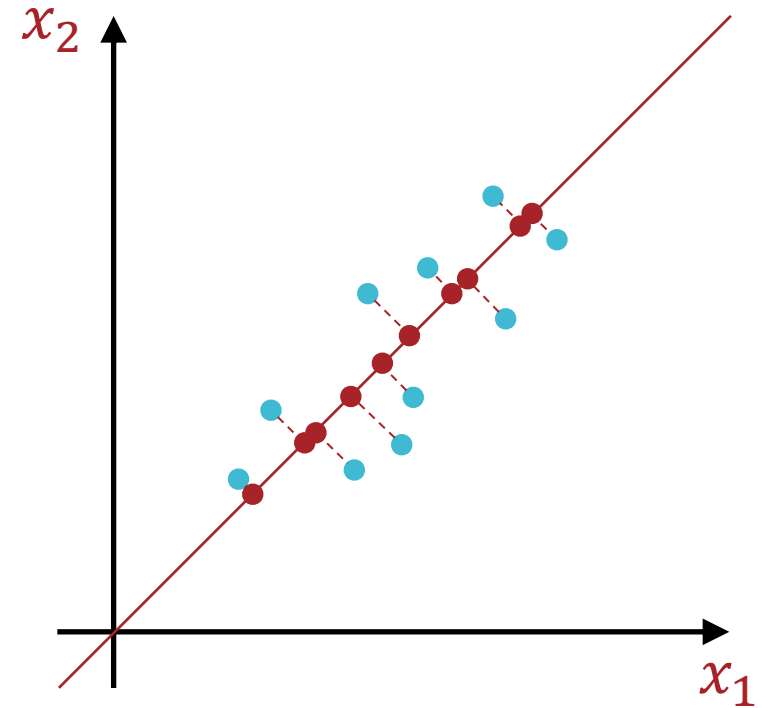
Feature Elimination



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:

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

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

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

Reconstruction Error

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

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

Length of projection

Direction of projection

Reconstruction Error

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

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

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

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

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

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

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

Minimizing the
Reconstruction
Error



Maximizing the
Variance

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

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

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

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

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

Maximizing the Variance

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

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

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

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

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

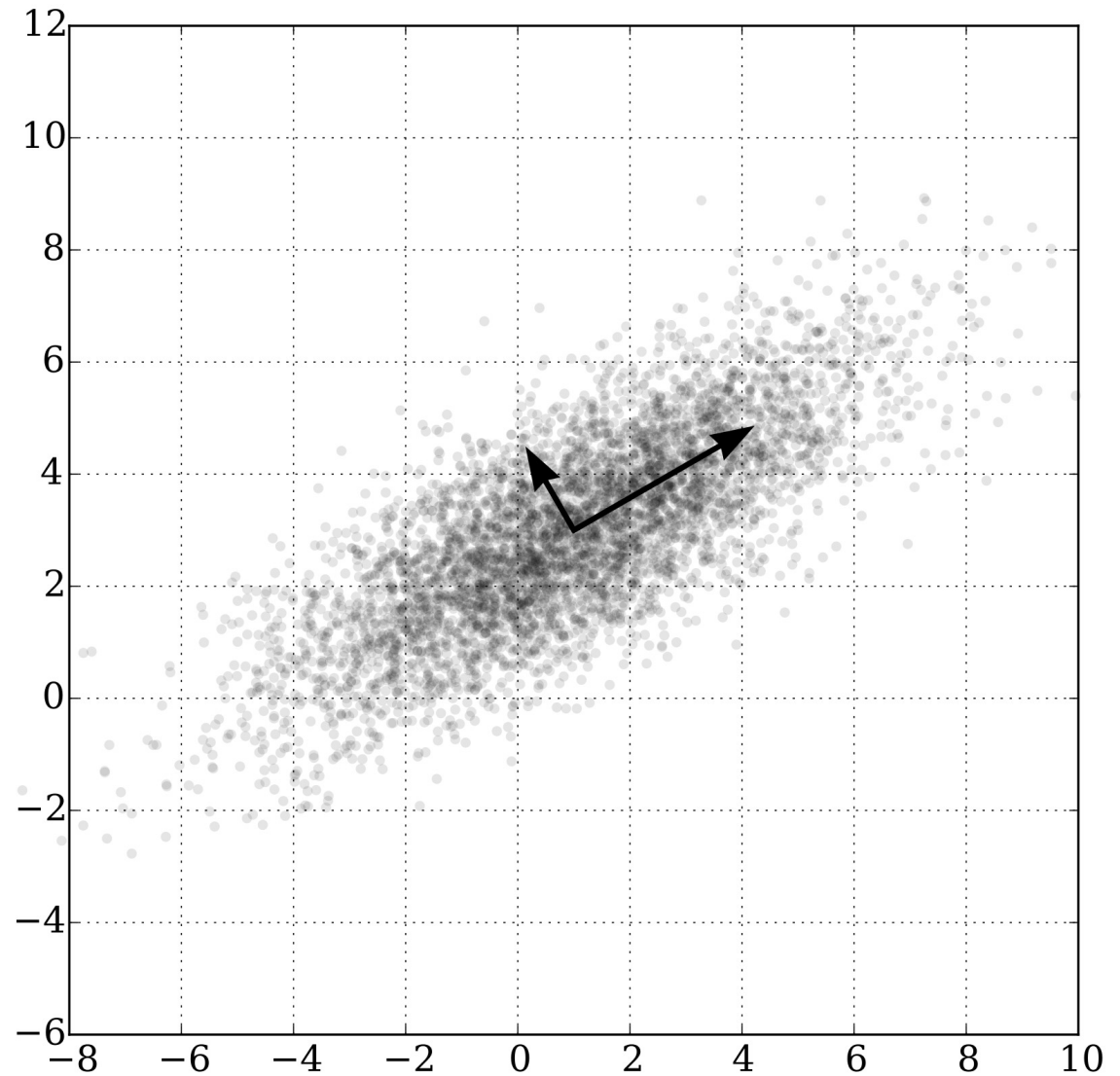
Maximizing the Variance

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

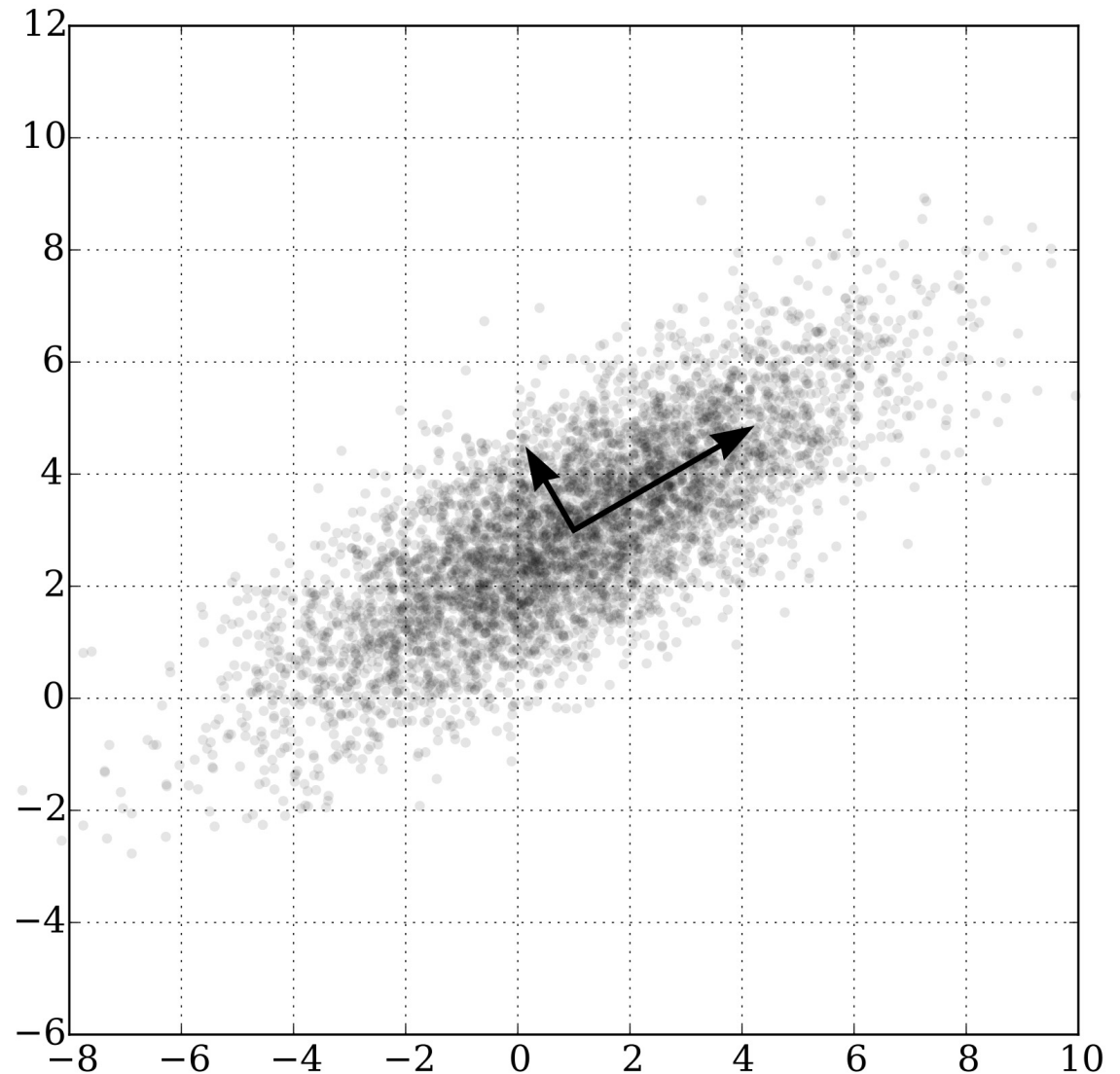
$$(X^T X) \hat{\mathbf{v}} = \lambda \hat{\mathbf{v}} \rightarrow \hat{\mathbf{v}}^T (X^T X) \hat{\mathbf{v}} = \lambda \hat{\mathbf{v}}^T \hat{\mathbf{v}} = \lambda$$

- The first principal component is the eigenvector $\hat{\mathbf{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\mathbf{v}}_2$ that corresponds to the second largest eigenvalue λ_2
 - $\hat{\mathbf{v}}_1$ and $\hat{\mathbf{v}}_2$ are orthogonal
- Etc ...
- λ_i is a measure of how much variance falls along $\hat{\mathbf{v}}_i$

Principal Components: Example



How can we efficiently find principal components (eigenvectors)?



Singular Value Decomposition (SVD) for PCA

- Every real-valued matrix $X \in \mathbb{R}^{N \times D}$ can be expressed as

$$X = USV^T$$

where:

1. $U \in \mathbb{R}^{N \times N}$ - columns of U are eigenvectors of XX^T
2. $V \in \mathbb{R}^{D \times D}$ - columns of V are eigenvectors of $X^T X$
3. $S \in \mathbb{R}^{N \times D}$ - diagonal matrix whose entries are the eigenvalues of $X \rightarrow$ squared entries are the eigenvalues of XX^T and $X^T X$

PCA Algorithm

- Input: $\mathcal{D} = \{(\mathbf{x}^{(n)})\}_{n=1}^N, \rho$
 1. Center the data
 2. Use SVD to 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 , $Z = X V_\rho$
- 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. Use SVD to 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

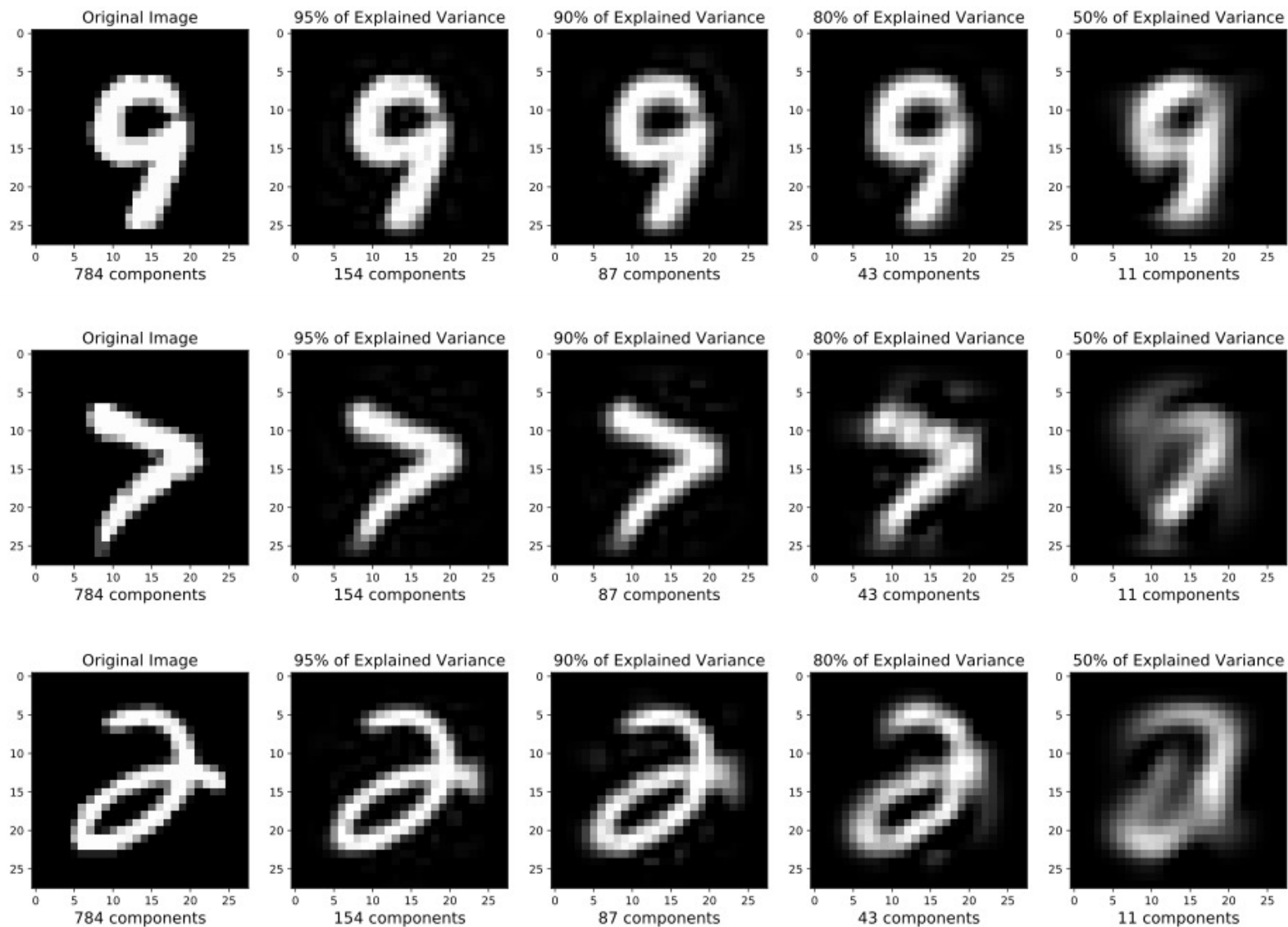
Choosing the number of PCs

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

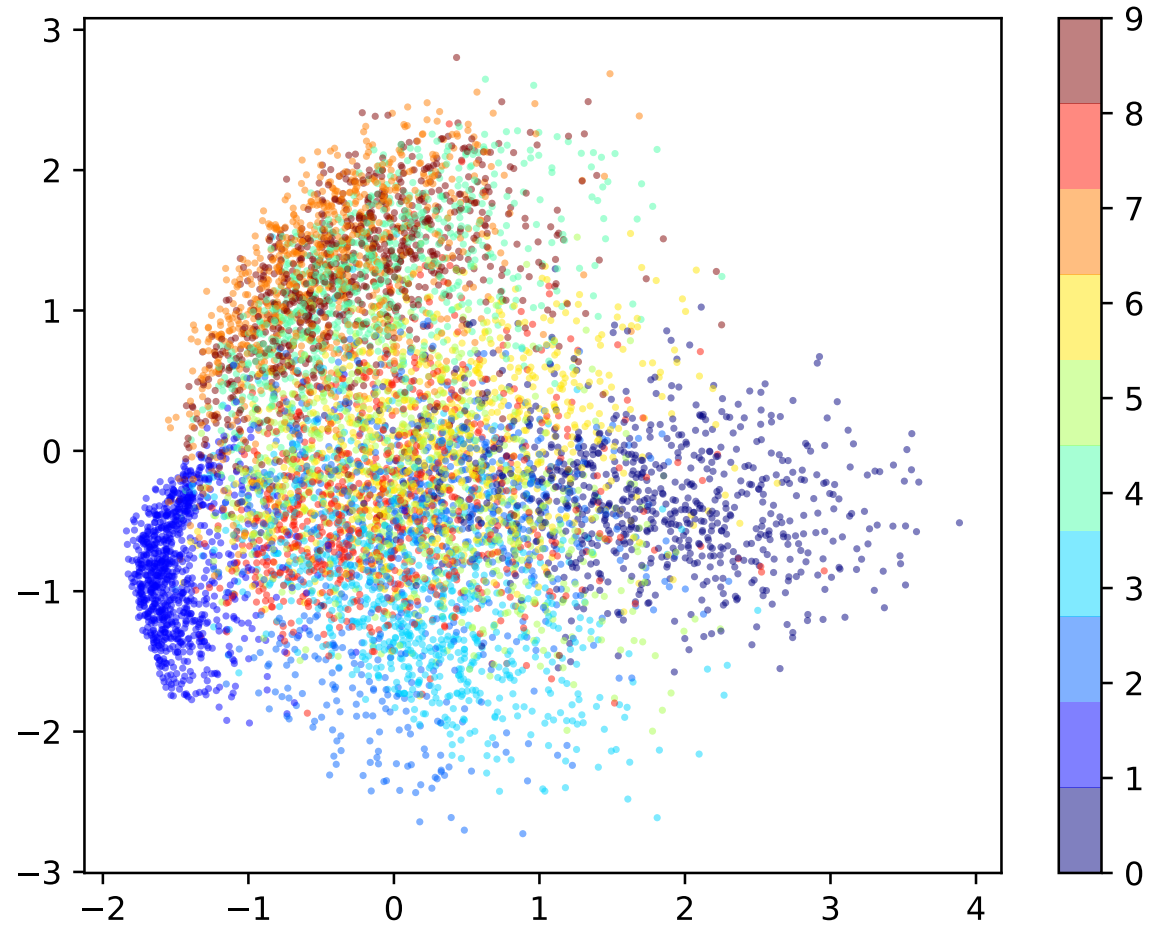
$$\lambda_i / \sum \lambda_j$$

- 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

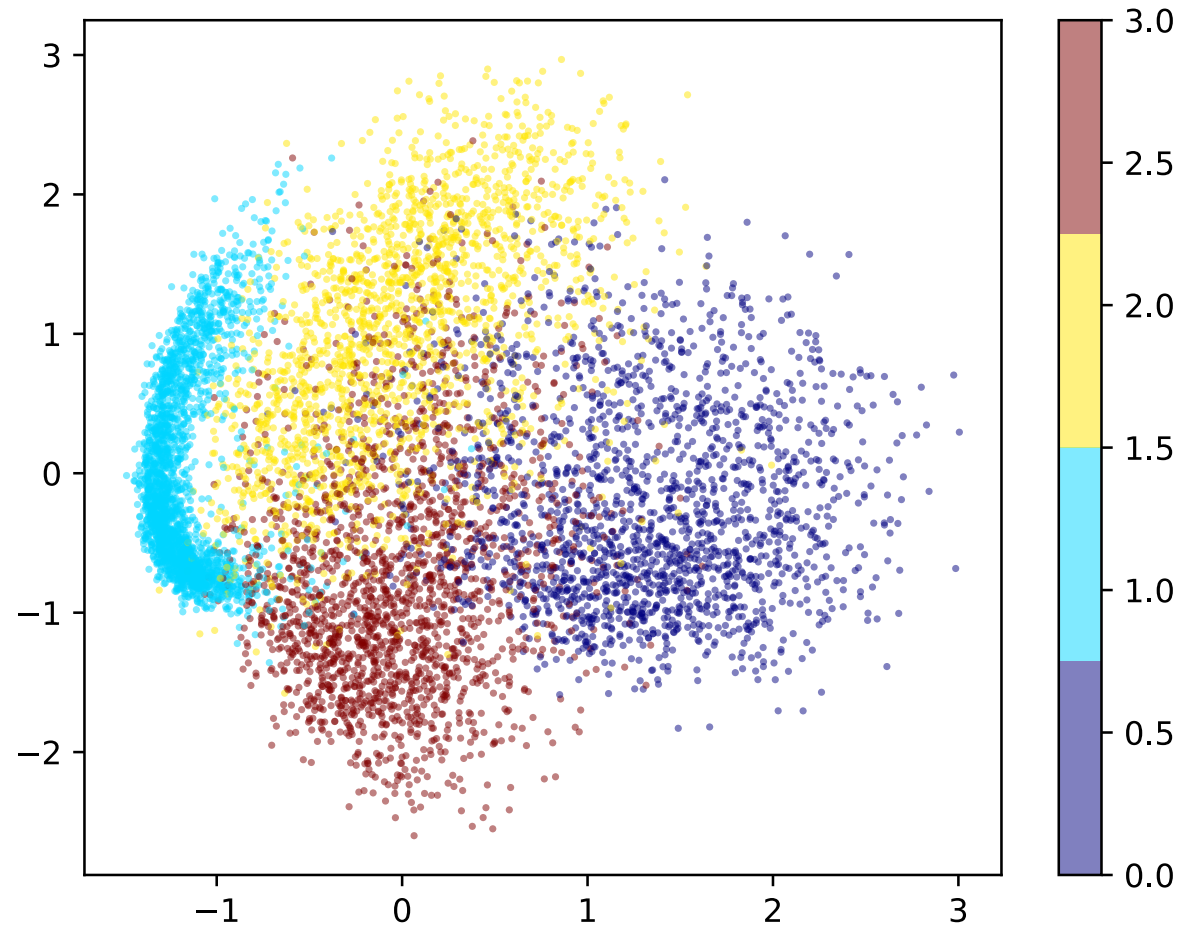
PCA Example: MNIST Digits



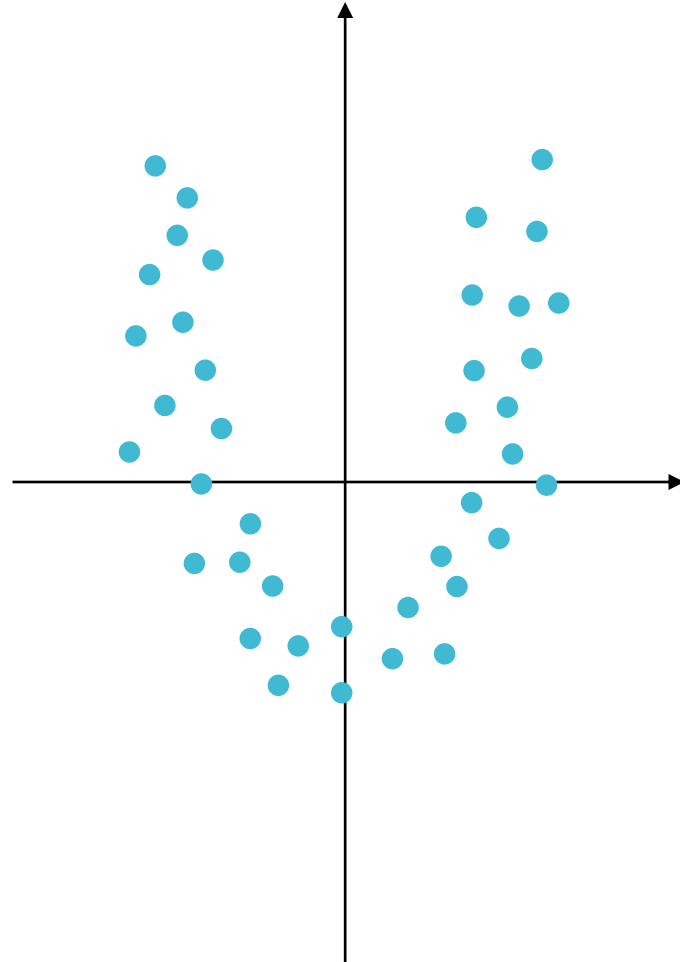
PCA Example: MNIST Digits



PCA Example: MNIST Digits



Shortcomings of PCA



Kernel PCA

- Claim: Principal components can be expressed as linear combinations of the (centered) data
- Proof:

$$\begin{aligned}(X^T X)\hat{\mathbf{v}} &= \lambda\hat{\mathbf{v}} \rightarrow \left(\sum_{i=1}^N \tilde{\mathbf{x}}^{(i)} \tilde{\mathbf{x}}^{(i)T} \right) \hat{\mathbf{v}} = \lambda\hat{\mathbf{v}} \\ &\rightarrow \frac{1}{\lambda} \sum_{i=1}^N \tilde{\mathbf{x}}^{(i)} \left(\tilde{\mathbf{x}}^{(i)T} \hat{\mathbf{v}} \right) = \hat{\mathbf{v}} \\ &\rightarrow \hat{\mathbf{v}} = \frac{1}{\lambda} \sum_{i=1}^N \hat{a}_i \tilde{\mathbf{x}}^{(i)} = \frac{1}{\lambda} X^T \hat{\mathbf{a}}\end{aligned}$$

Kernel PCA

- Claim: Principal components can be expressed as linear combinations of the (centered) data
- Consequence:

$$\begin{aligned}(X^T X)\hat{\mathbf{v}} &= \lambda\hat{\mathbf{v}} \rightarrow (X^T X)\left(\frac{1}{\lambda}X^T\hat{\mathbf{a}}\right) = \lambda\left(\frac{1}{\lambda}X^T\hat{\mathbf{a}}\right) \\ &\rightarrow X^T X X^T\hat{\mathbf{a}} = \lambda X^T\hat{\mathbf{a}} \\ &\rightarrow X X^T X X^T\hat{\mathbf{a}} = \lambda X X^T\hat{\mathbf{a}}\end{aligned}$$

$X X^T$ only consists of inner products between data points!

Recall: The Kernel Trick

- Approach: instead of computing $\Phi(\mathbf{x})$, find some function K_{Φ} s.t. $K_{\Phi}(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^T \Phi(\mathbf{x}') \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$
 - $K_{\Phi}(\mathbf{x}, \mathbf{x}')$ should be cheaper to compute than $\Phi(\mathbf{x})$
- Given some data set $\mathcal{D} = \{(\mathbf{x}^{(i)})\}_{i=1}^N$, define the Gram matrix of a kernel K as

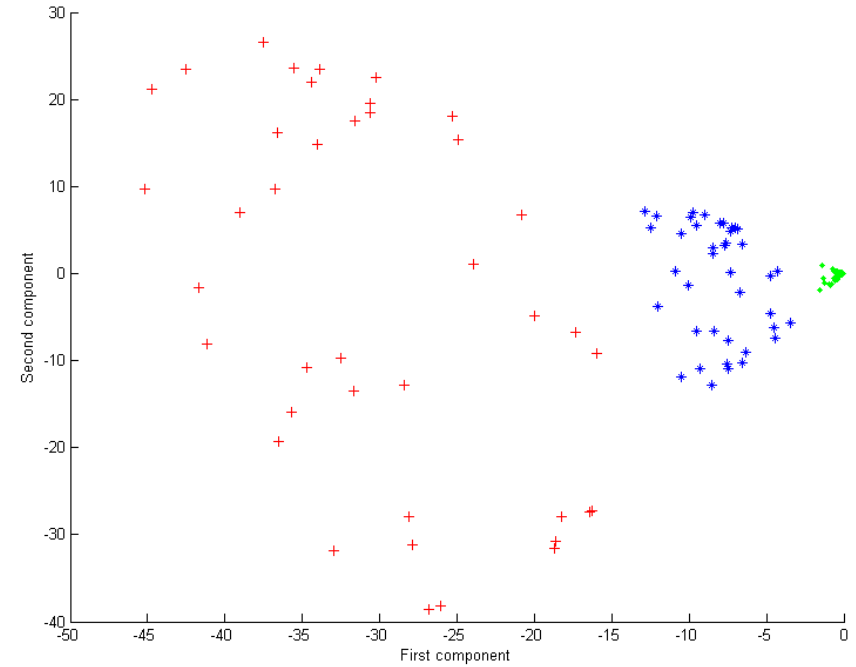
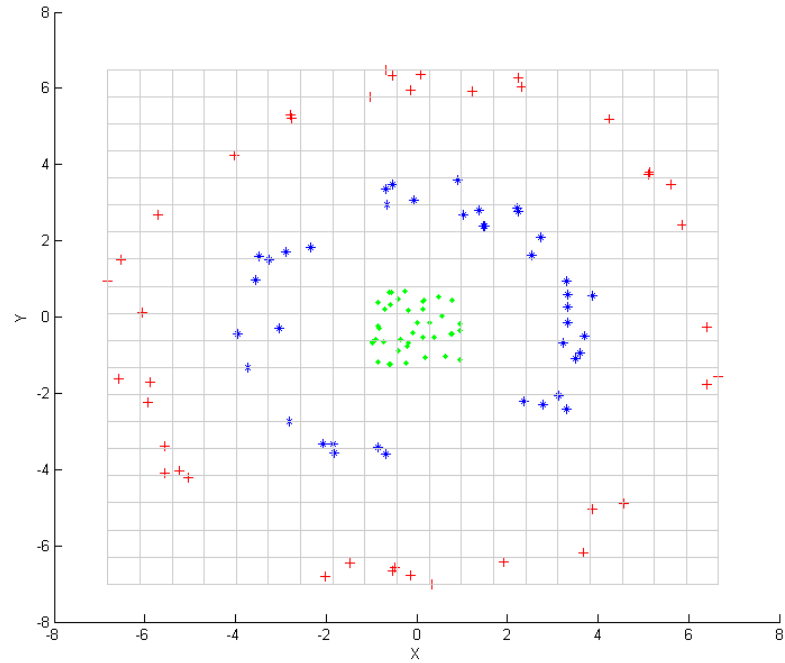
$$\mathbf{K} = \begin{bmatrix} K(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & K(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) & \dots & K(\mathbf{x}^{(1)}, \mathbf{x}^{(N)}) \\ K(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) & K(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) & \dots & K(\mathbf{x}^{(2)}, \mathbf{x}^{(N)}) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}^{(N)}, \mathbf{x}^{(1)}) & K(\mathbf{x}^{(N)}, \mathbf{x}^{(2)}) & \dots & K(\mathbf{x}^{(N)}, \mathbf{x}^{(N)}) \end{bmatrix}$$

Kernel PCA

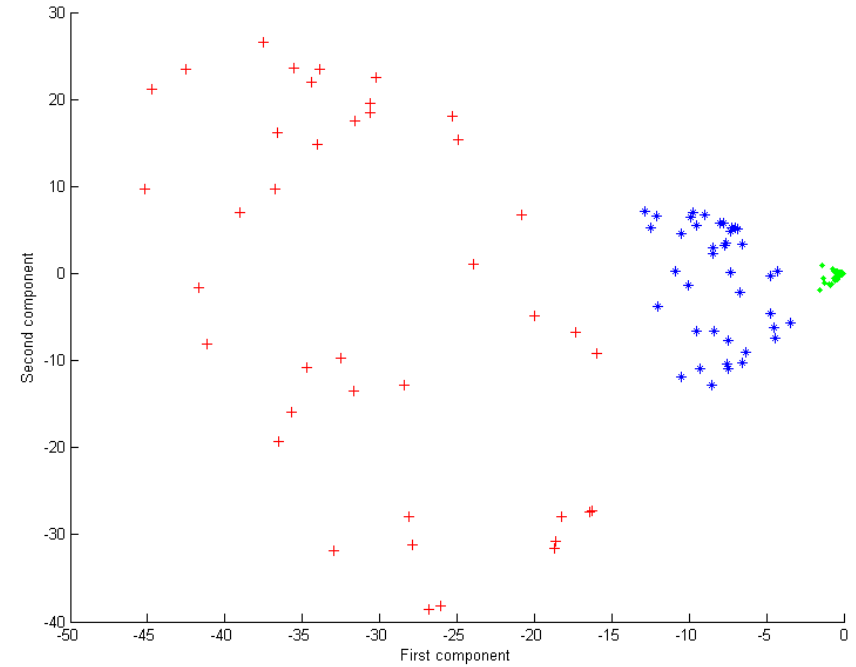
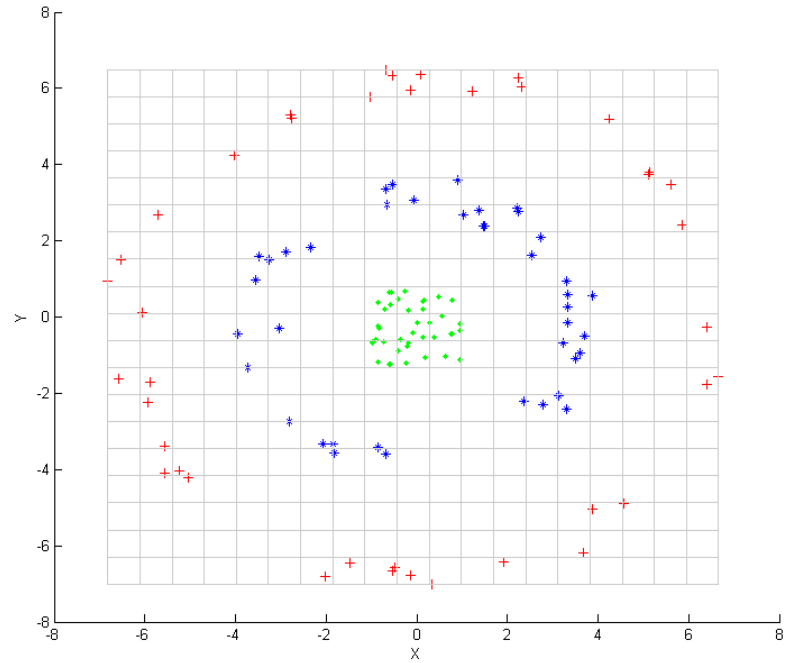
$$\mathbf{K}\mathbf{K}\hat{\mathbf{a}} = \lambda\mathbf{K}\hat{\mathbf{a}} \rightarrow \mathbf{K}\hat{\mathbf{a}} = \lambda\hat{\mathbf{a}}$$

- Principal components are the eigenvectors of the Gram matrix \mathbf{K} if the data is centered in the transformed space
- We can center the transformed data *without explicitly computing the transformations* (see Bishop 12.3 for complete details):

$$\tilde{\mathbf{K}} = \mathbf{K} - \frac{1}{N}\mathbf{1}\mathbf{K} - \frac{1}{N}\mathbf{K}\mathbf{1} + \frac{1}{N^2}\mathbf{1}\mathbf{K}\mathbf{1}$$



Kernel PCA: Example



PCs are still orthogonal...

Independent Component Analysis (ICA)

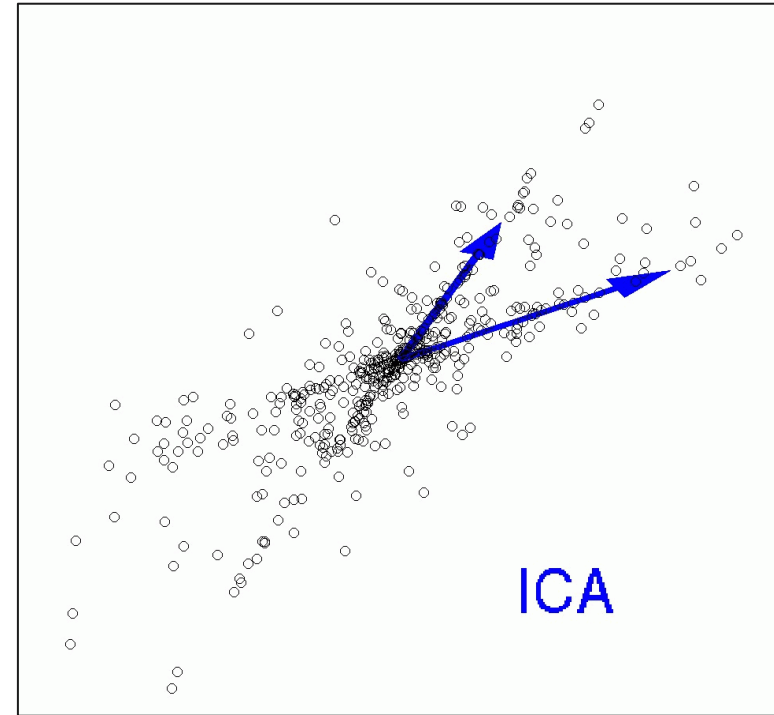
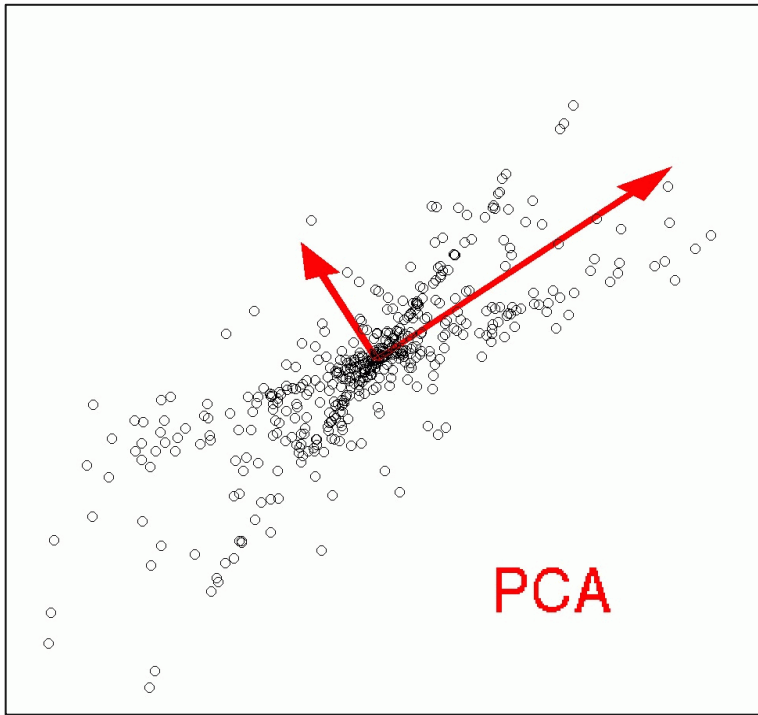
- Assume our data is a linear transformation of arbitrary (not necessarily orthogonal) components (“signals”)

$$\mathbf{x}^{(i)} = A\mathbf{s}^{(i)}$$

- Typically assume $\mathbf{s}^{(i)}$ is the same size as $\mathbf{x}^{(i)}$
- Goal: find components that are as statistically independent as possible:

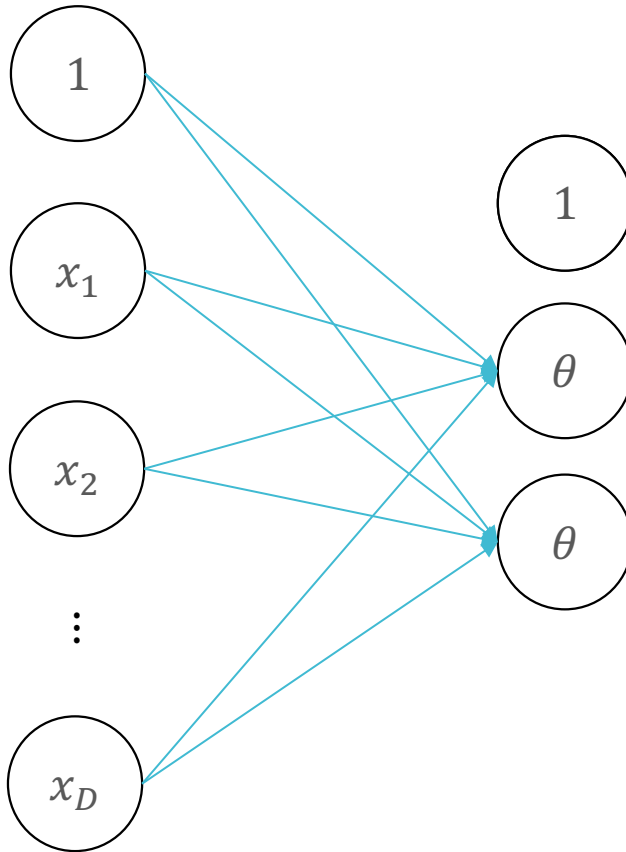
$$p\left(s_1^{(i)}, \dots, s_D^{(i)}\right) \approx p\left(s_1^{(i)}\right) \dots p\left(s_D^{(i)}\right)$$

- Common approach: minimize the mutual information between $s_1^{(i)}, \dots, s_D^{(i)}$



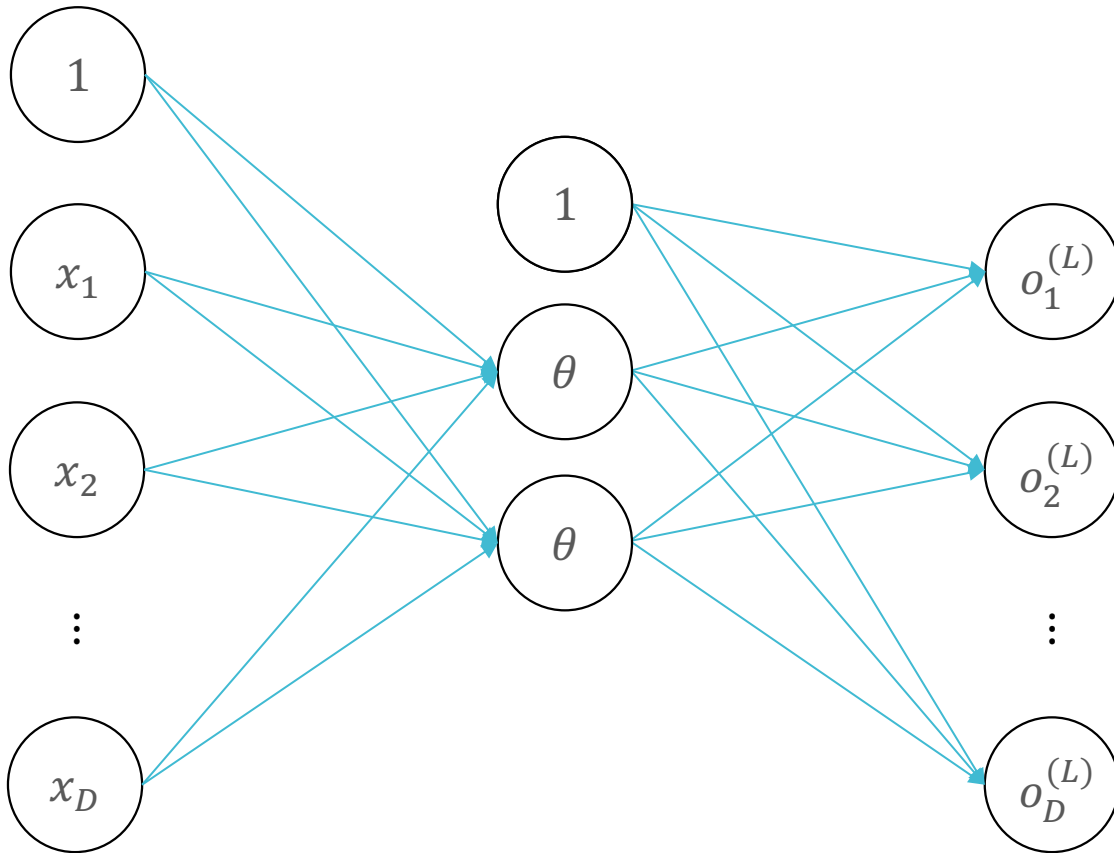
PCA vs. ICA

Autoencoders



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

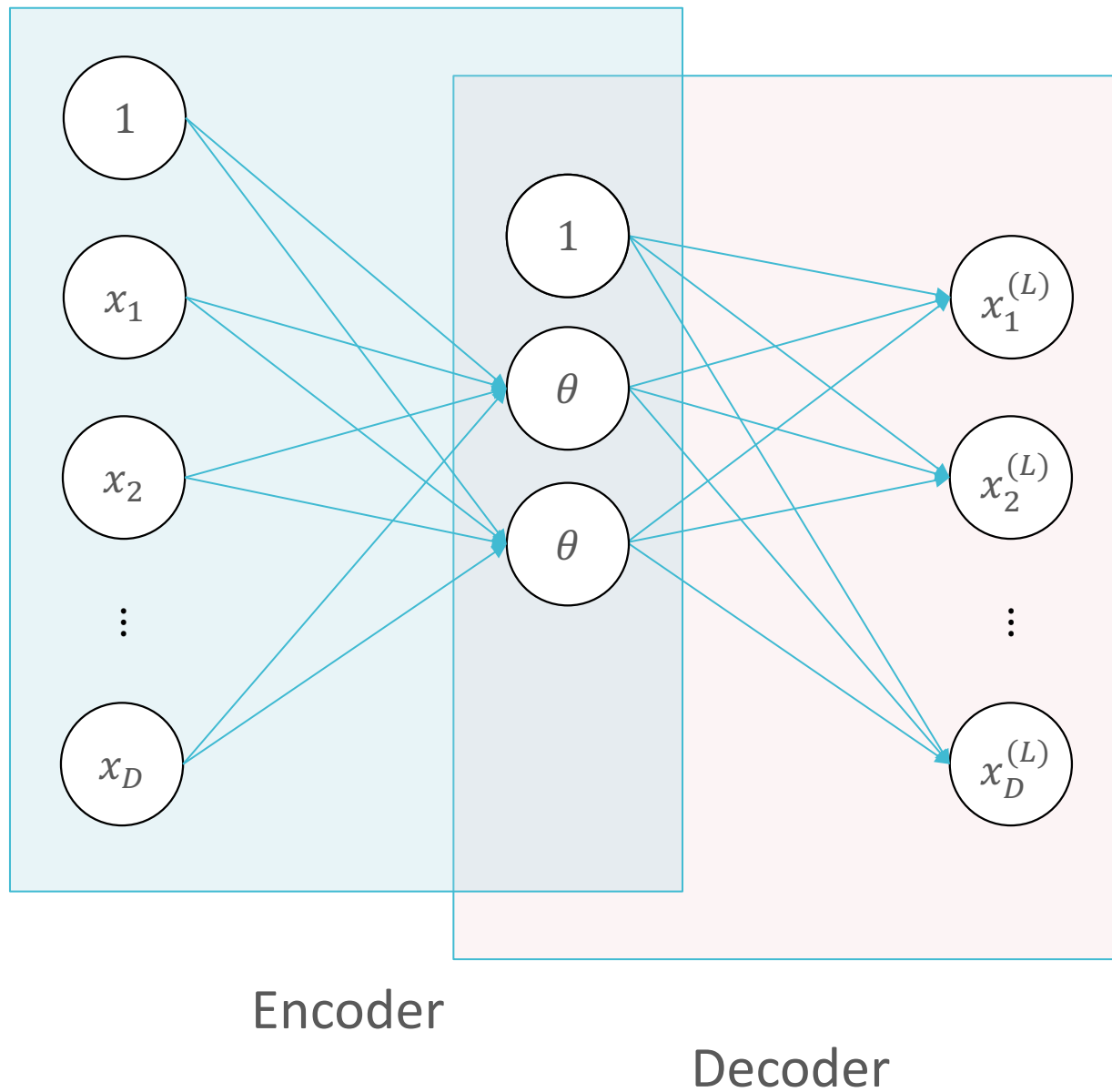
Autoencoders



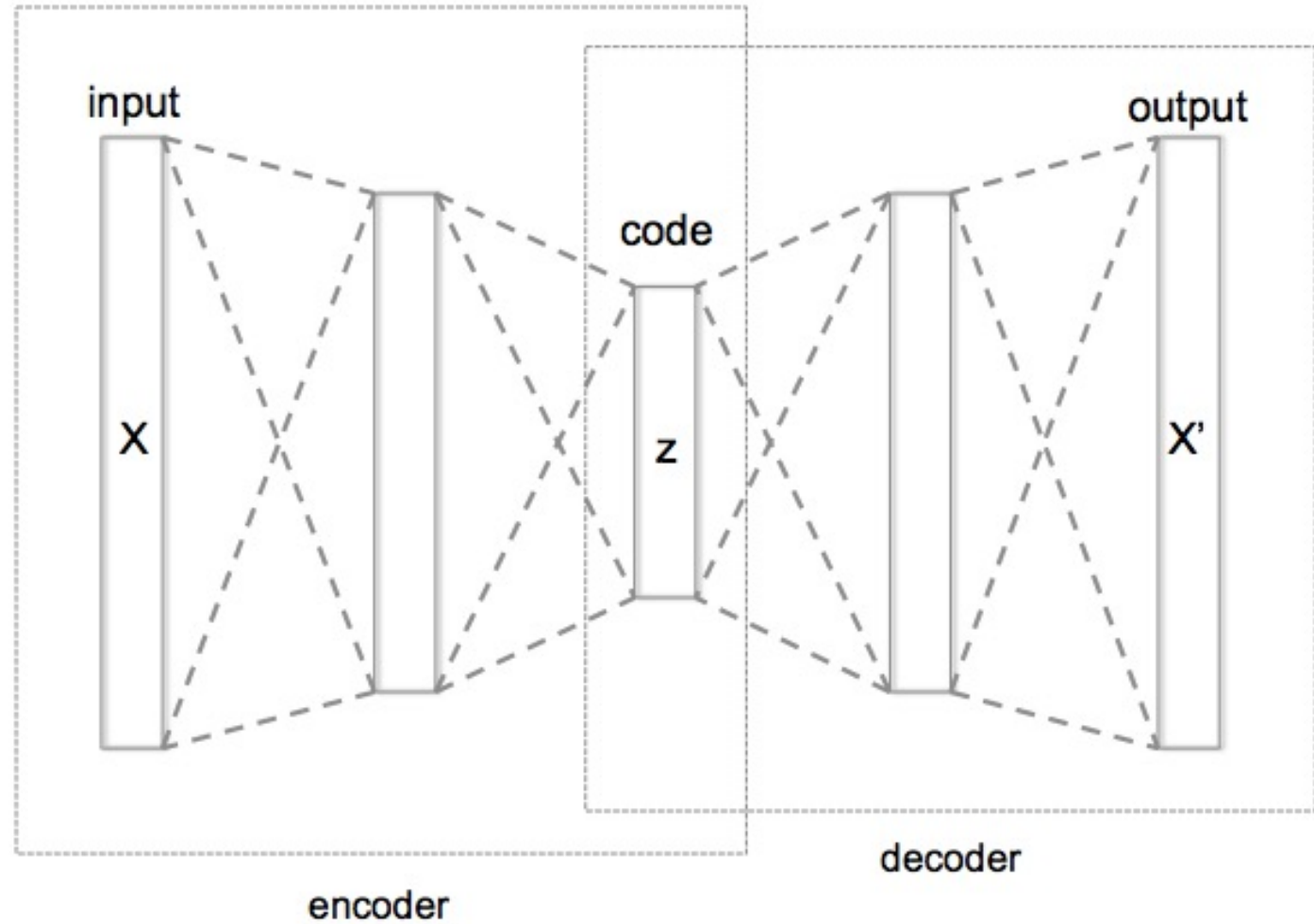
- Learn the weights by minimizing the reconstruction loss:

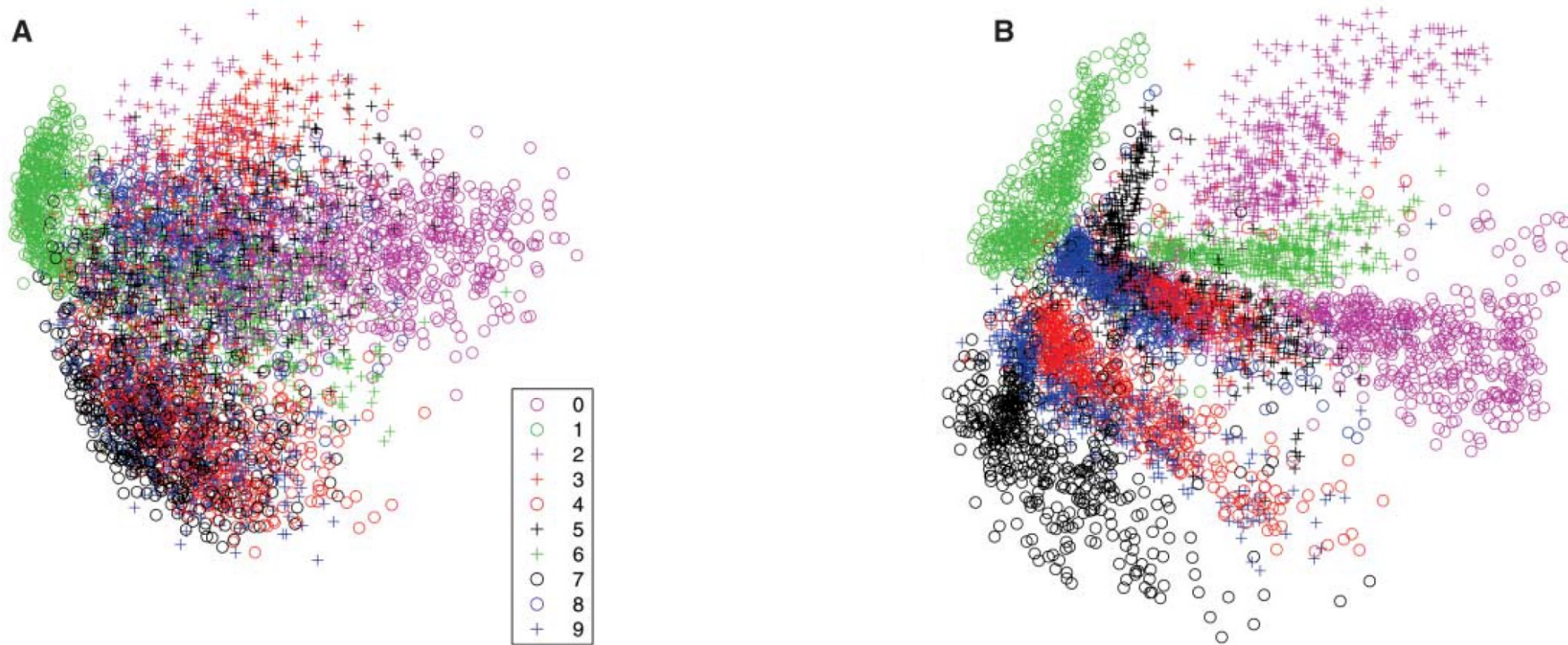
$$e(\mathbf{x}) = \|\mathbf{x} - \mathbf{o}^{(L)}\|_2^2$$

Autoencoders



Deep Autoencoders





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

Key Takeaways

- K-means partitions the dataset into K groups using block coordinate descent
 - The K-means objective function is non-convex
 - K-means++ can help avoid poor initializations
- PCA finds an orthonormal basis where the first principal component maximizes the variance \Leftrightarrow minimizes the reconstruction error
 - Can be kernelized
- ICA finds statistically independent, not orthogonal components
- Autoencoders use neural networks to automatically learn a latent representation that minimizes the reconstruction error