# 10-301/601: Introduction to Machine Learning Lecture 25: Dimensionality Reduction

Matt Gormley & Henry Chai

11/25/24

# Front Matter

- Announcements
  - HW8 released 11/17, due 11/25 (today!) at 11:59 PM
    - Please be mindful of your grace day usage (see the course syllabus for the policy)
  - HW9 released 11/25 (today!), due 12/5 at 11:59 PM
    - You are not expected to work on HW9 over Thanksgiving break
      - Relatedly, there are no OH over break
    - **You may only take at most 2 grace days on HW9**
    - Recitation for HW9 on Monday, 12/2 (after break)

# Recall: $K$-means Algorithm

How do we set these hyperparameters?

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

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:

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

   b. Recompute the cluster centers:

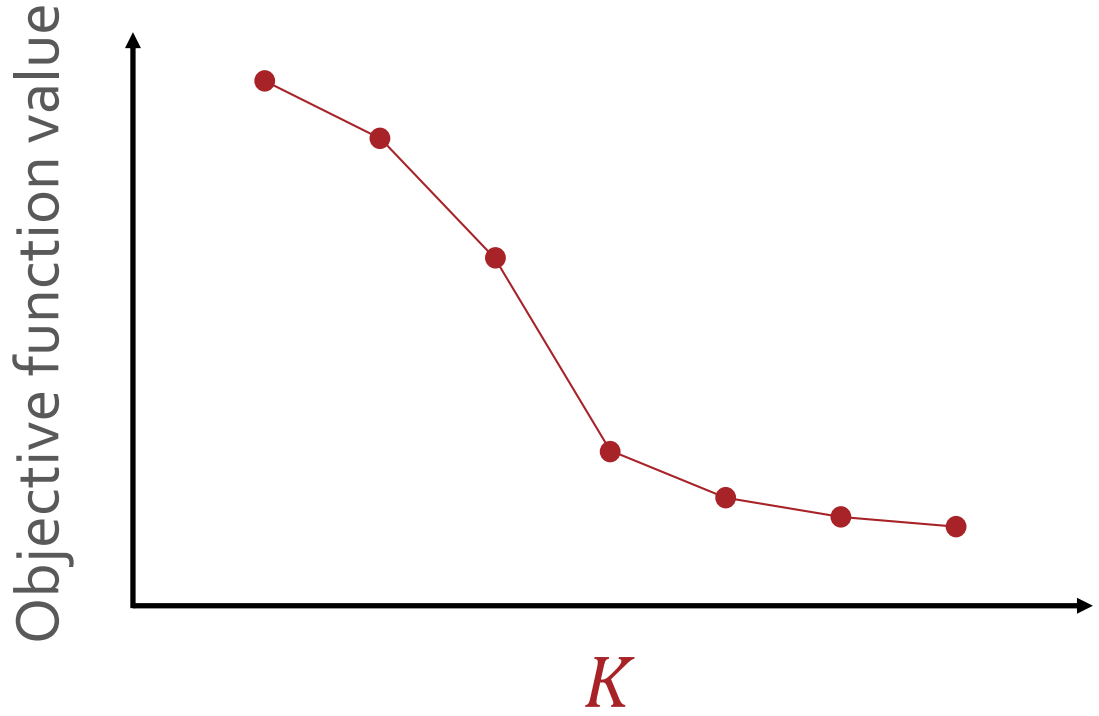
$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n\,:z^{(n)}=k} \boldsymbol{x}^{(n)}$$

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

- Output: cluster centers $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and cluster assignments $z^{(1)}, \ldots, z^{(N)}$

# Setting $K$

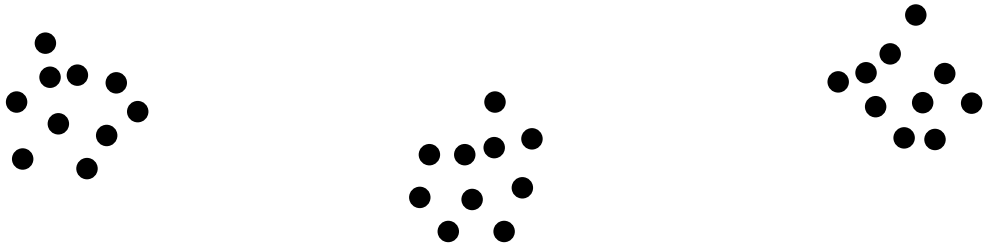
- Idea: choose the value of $K$ that minimizes the objective function




- Better Idea: look for the characteristic "elbow" or largest decrease when going from $K-1$ to $K$

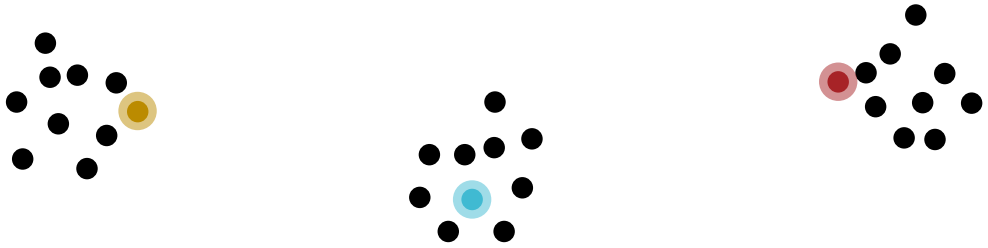
# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers



# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers

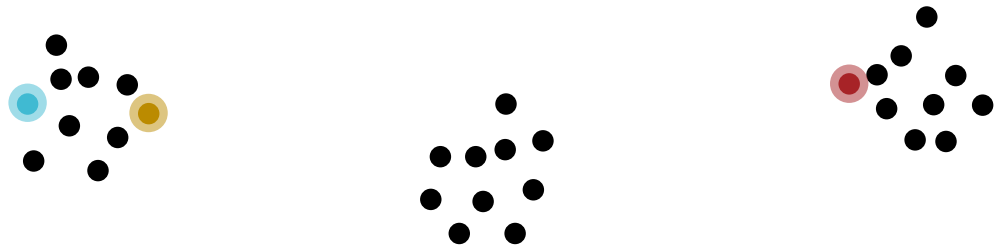


# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers

# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers

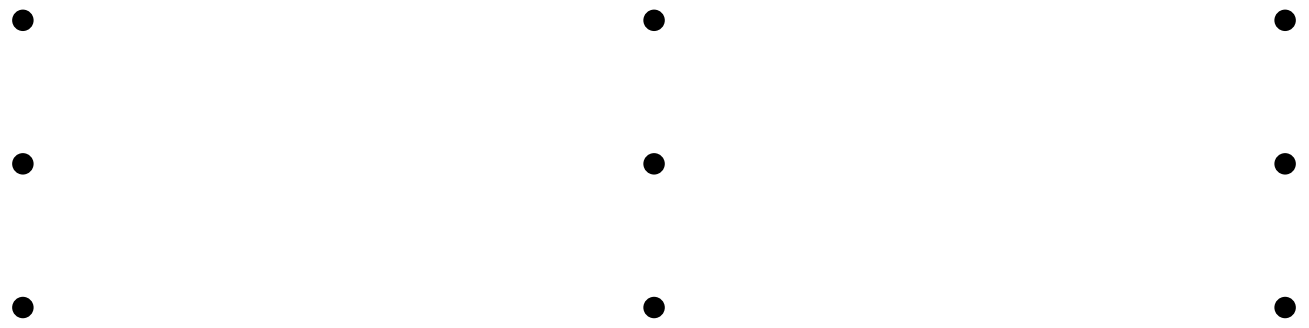


# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers

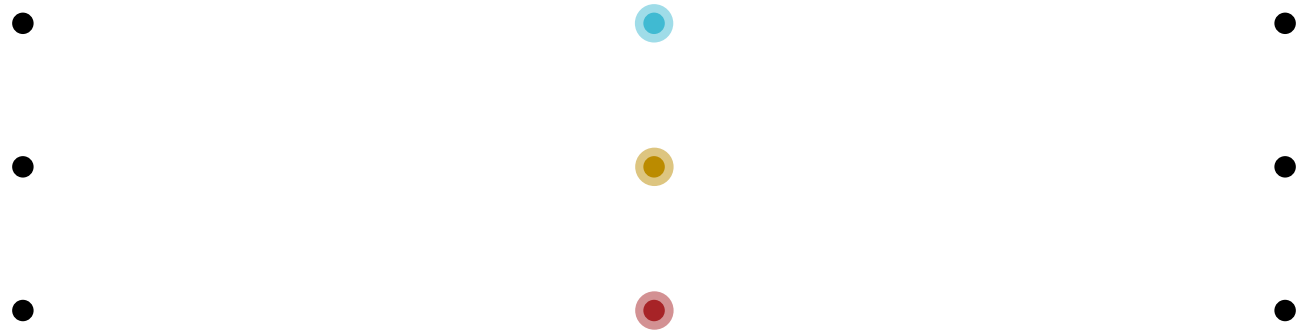
# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers



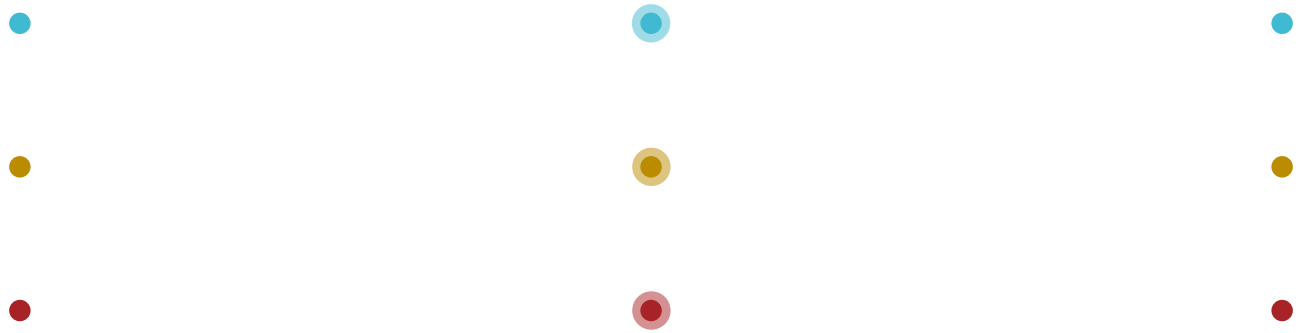
# Initializing $K$-means: Lloyd's Method

- Randomly choose $K$ data points to be the initial cluster centers



# Initializing $K$-means: Lloyd's Method

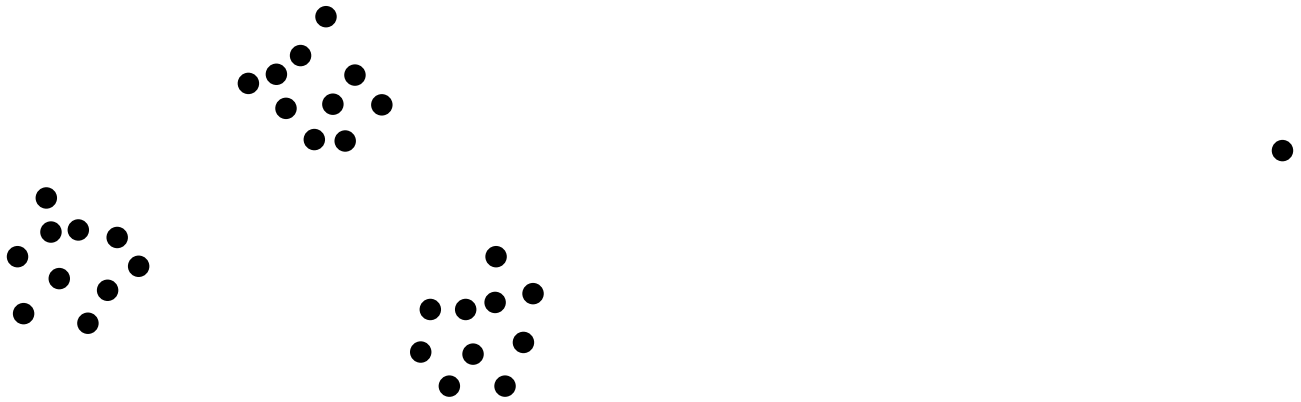
- Randomly choose $K$ data points to be the initial cluster centers



- 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

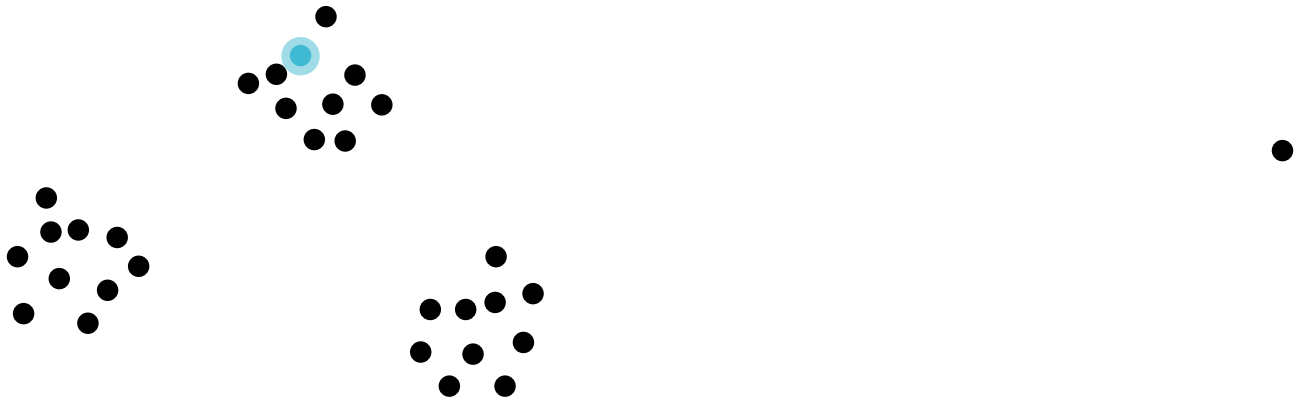
# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times



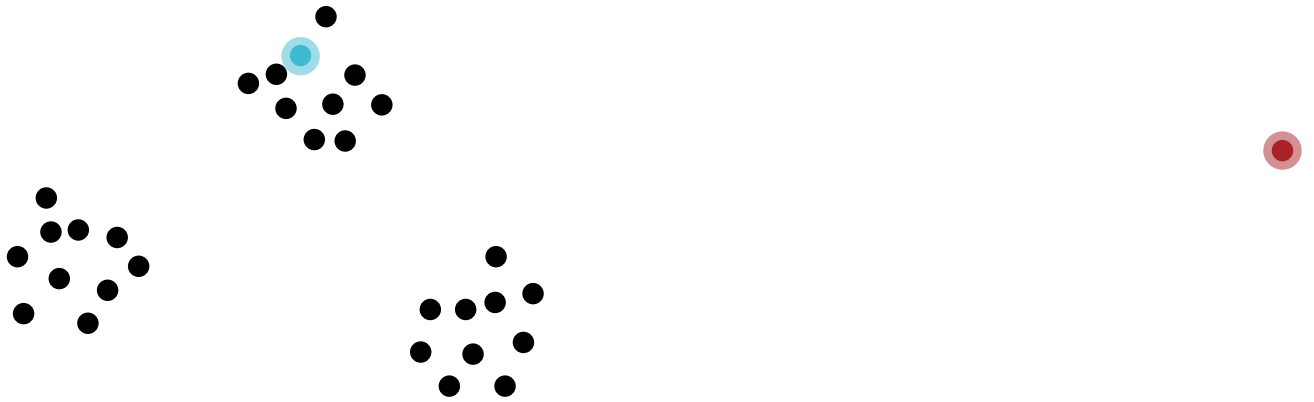
# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times



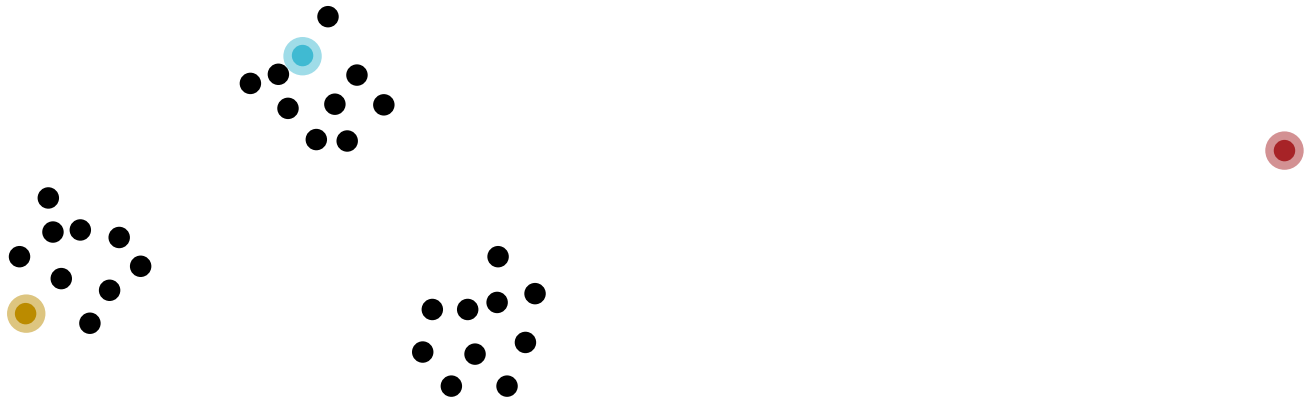
# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times



# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times



# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times

# Initializing $K$-means: Furthest Point

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. Select the data point with the largest $D(\boldsymbol{x})$ as the next cluster center
4. Repeat 2 and 3 $K - 1$ times

- Works great in the case of well-clustered data!
- Can struggle with outliers...

# Initializing $K$-means: $K$-means++

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. *Sample* the next cluster center with probability proportional to $D(\boldsymbol{x})^2$
4. Repeat 2 and 3 $K-1$ times

| $i$ | $D(\boldsymbol{x}^{(i)})$ | $D(\boldsymbol{x}^{(i)})^2$ | Probability of Being Selected |
|---|---|---|---|
| 1 | 4 | 16 | $16/123$ |
| 2 | 7 | 49 | $49/123$ |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $N$ | 1 | 1 | $1/123$ |
| **Total** | | 123 | $123/123 = 1$ |

# Initializing $K$-means: $K$-means++

1. Choose the first cluster center randomly from the data points
2. For each other data point $\boldsymbol{x}$, compute $D(\boldsymbol{x})$, the distance between $\boldsymbol{x}$ and the closest cluster center to $\boldsymbol{x}$
3. *Sample* the next cluster center with probability proportional to $D(\boldsymbol{x})^2$
4. Repeat 2 and 3 $K-1$ times

- $K$-means++ achieves a $O(\log K)$ approximation to the optimal clustering in expectation!
- All initialization methods can benefit from multiple random restarts

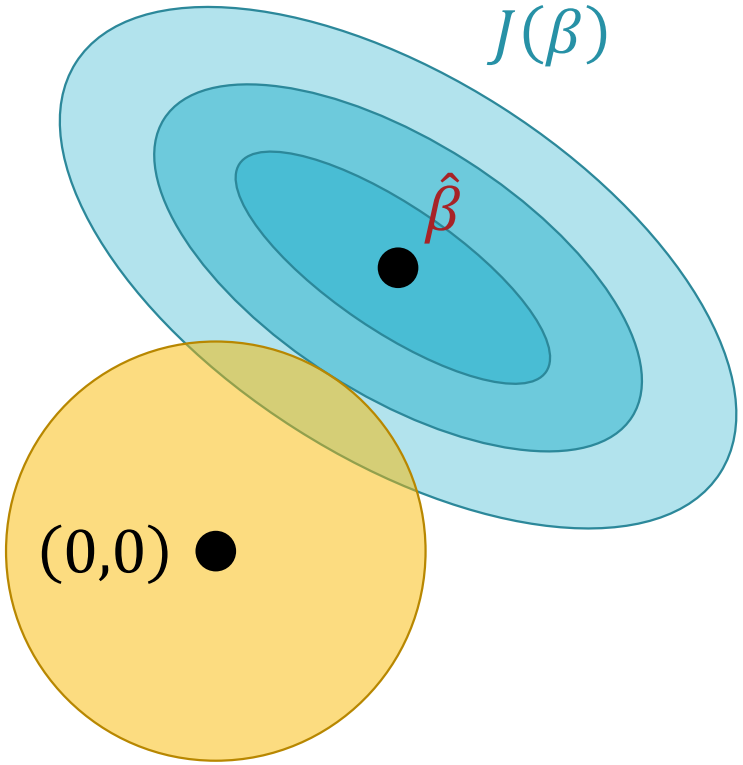
# $K$-means Learning Objectives

You should be able to…

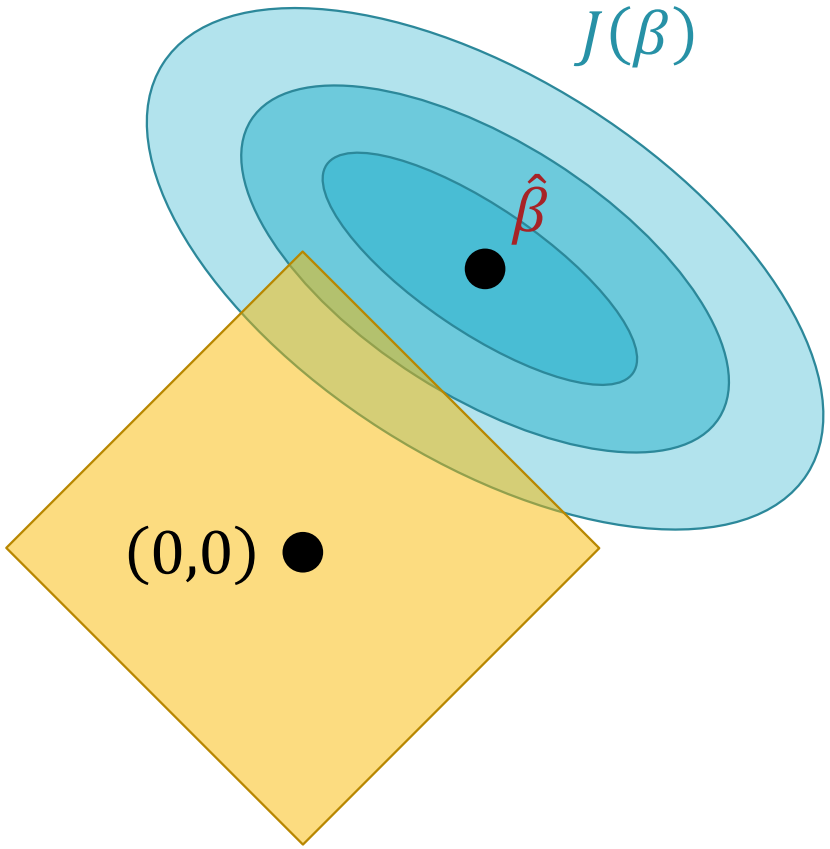
- Distinguish between coordinate descent and block coordinate descent
- Define an objective function that gives rise to a "good" clustering
- Apply block coordinate descent to an objective function preferring each point to be close to its nearest objective function to obtain the K-Means algorithm
- Implement the K-Means algorithm
- Connect the nonconvexity of the K-Means objective function with the (possibly) poor performance of random initialization

# Dimensionality Reduction

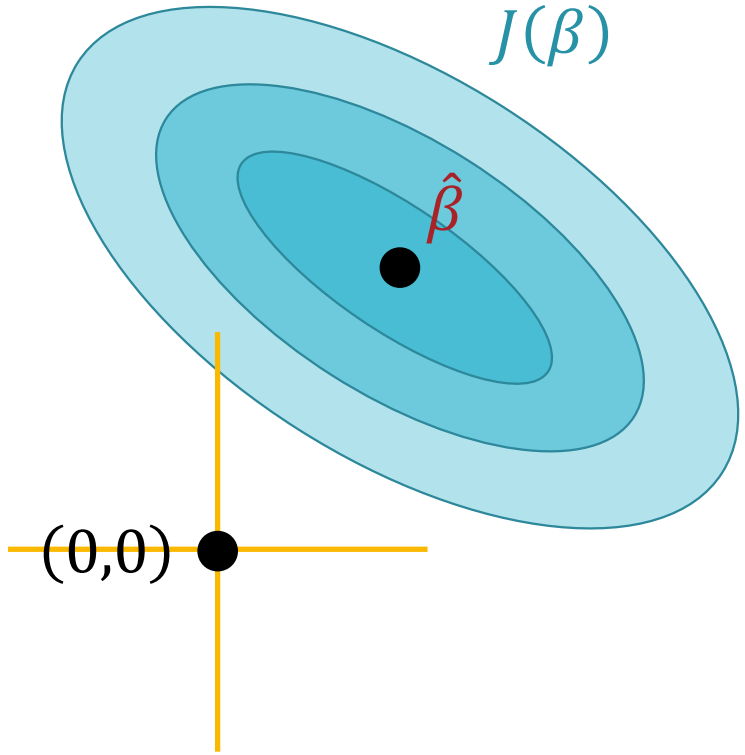
- Goal: given some unlabeled data set, learn a latent (typically lower-dimensional) representation
- Use cases:
  - Reducing computational cost (runtime, storage, etc…)
  - Improving generalization
  - Visualizing data
- Applications:
  - High-resolution images/videos
  - Text data
  - Financial or transaction data




Ridge or $L2$

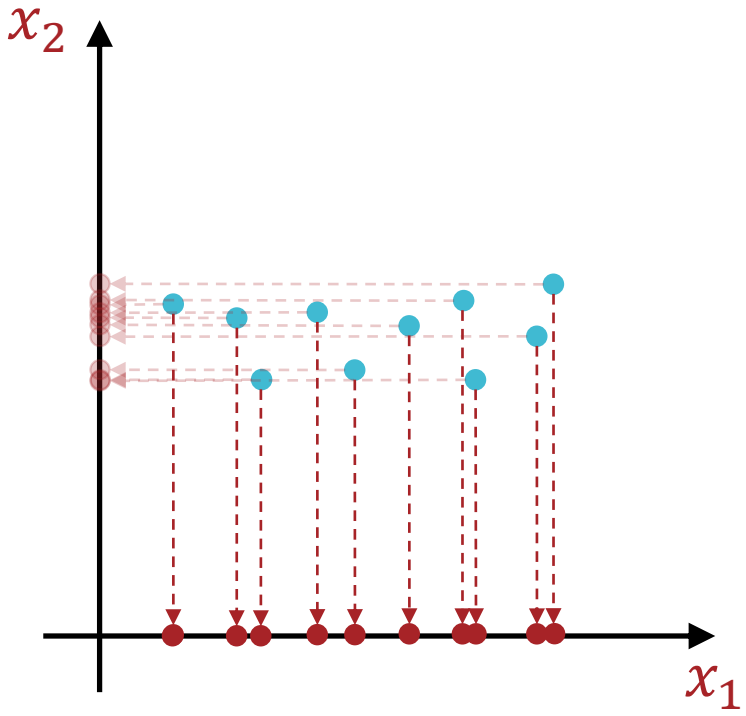


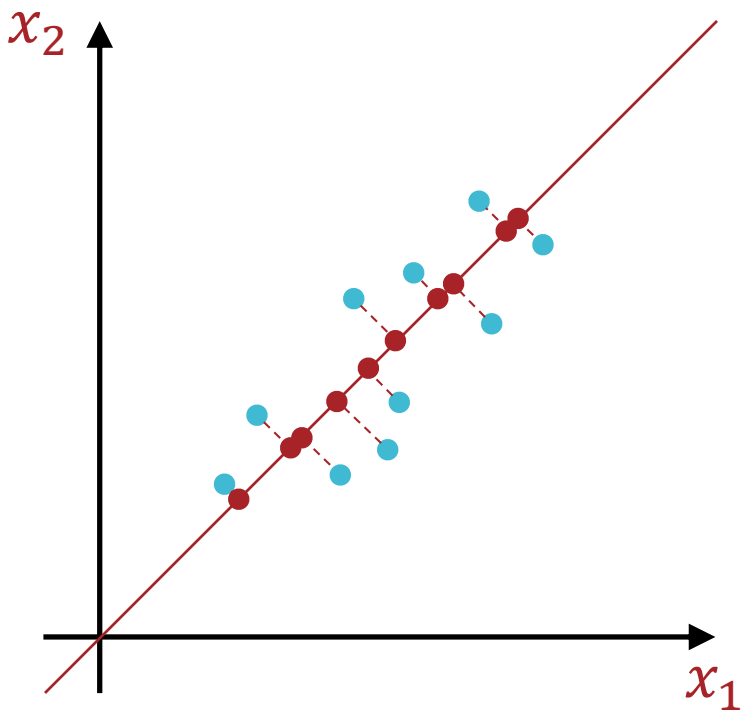

Lasso or $L1$



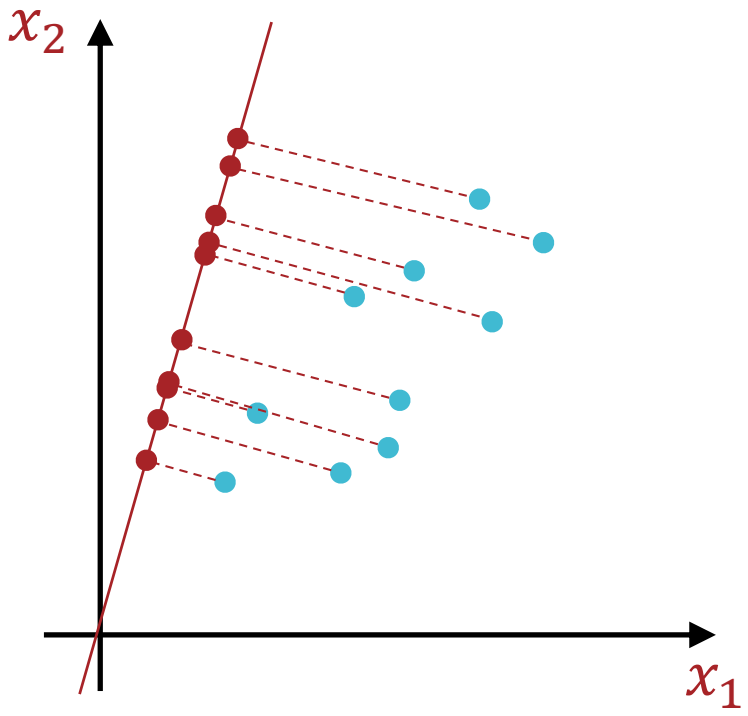

$L0$

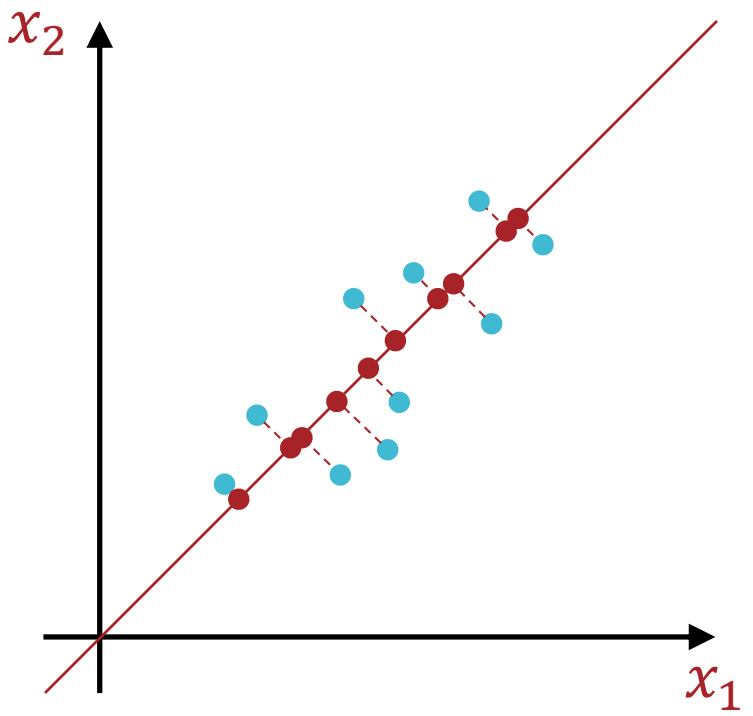
# Recall: $L1$ (or $L0$) Regularization



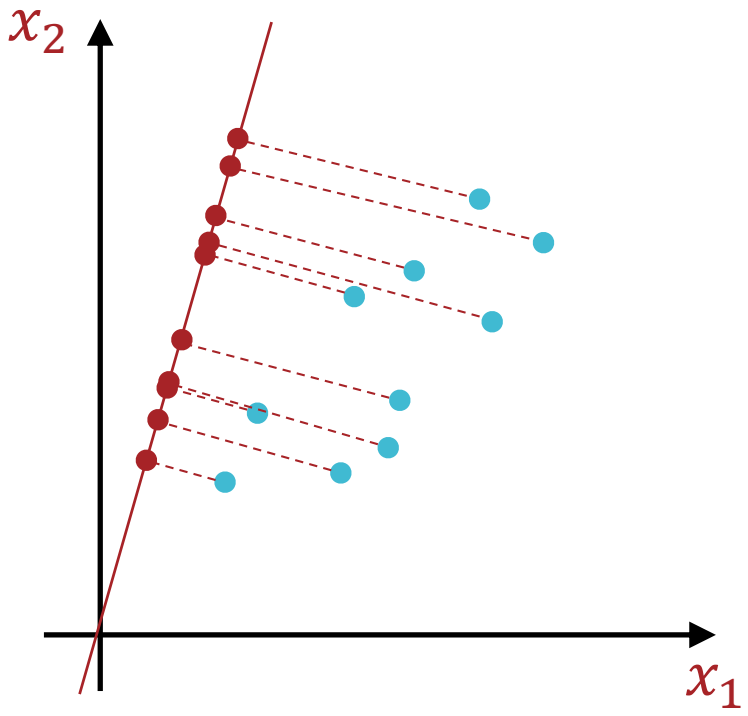




# Feature Elimination

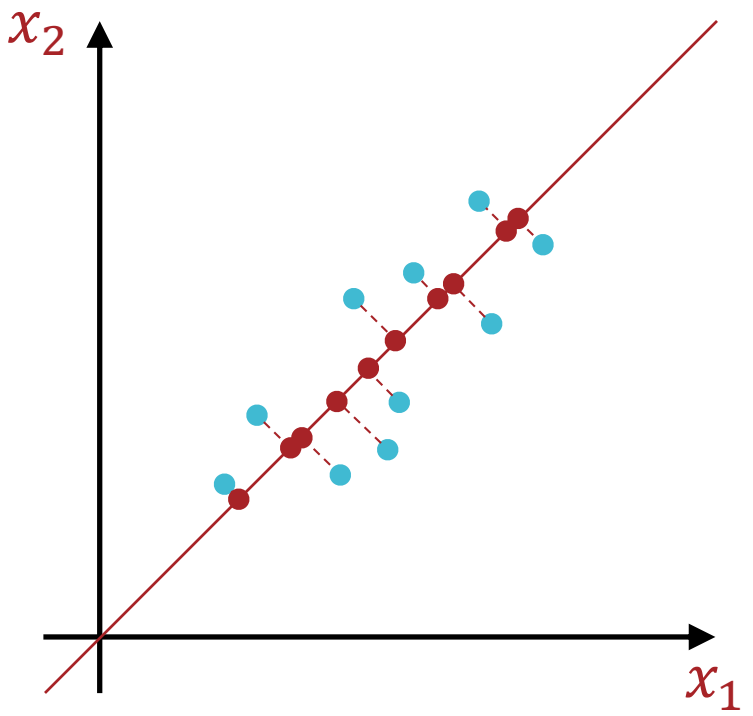



# Feature Reduction




Option A




Option B (TOXIC)

Option C

# Which projection do you prefer (Q1) and why (Q2)?

# Background: Sample Variance and Covariance

- Given a collection of $N$ 1-dimensional samples $\left[x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right]$ from some random variable, the **sample variance** is

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \hat{\mu}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\right)^2$$

- Given a collection of $N$ $D$-dimensional samples $\left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\right]$ from some random variable, the **sample covariance** between dimension $j$ and $k$ is

$$\Sigma_{jk} = \frac{1}{N}\sum_{i=1}^{N}\left(x_j^{(i)} - \hat{\mu}_j\right)\left(x_k^{(i)} - \hat{\mu}_k\right) \text{ where } \hat{\mu}_d = \frac{1}{N}\sum_{n=1}^{N} x_d^{(n)}$$

# Background: Sample Variance and Covariance

- Given a collection of $N$ 1-dimensional samples $\left[x^{(1)}, x^{(2)}, \ldots, x^{(N)}\right]$ from some random variable, the **sample variance** is

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \hat{\mu}\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(x^{(i)} - \frac{1}{N}\sum_{n=1}^{N} x^{(n)}\right)^2$$

- Given a collection of $N$ $D$-dimensional samples $\left[\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\right]$ from some random variable, the **sample covariance matrix** is

$$\Sigma = \frac{1}{N}X^T X \text{ where } X = \begin{bmatrix} (\boldsymbol{x}^{(1)} - \boldsymbol{\mu})^T \\ (\boldsymbol{x}^{(2)} - \boldsymbol{\mu})^T \\ \vdots \\ (\boldsymbol{x}^{(N)} - \boldsymbol{\mu})^T \end{bmatrix}$$

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

# Reconstruction Error

- 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

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

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

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

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

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

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

# 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

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

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

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = 2(X^T X)\boldsymbol{v} - 2\lambda \boldsymbol{v}$$

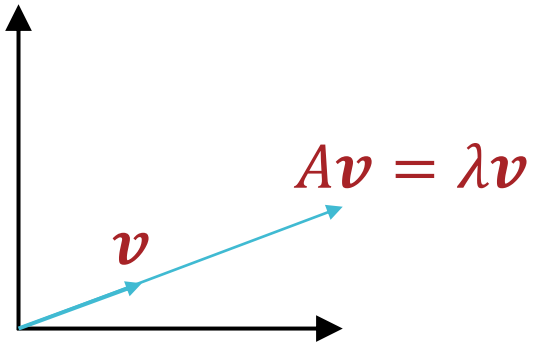
$$\rightarrow 2(X^T X)\hat{\boldsymbol{v}} - 2\lambda\hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}$$

- $\hat{\boldsymbol{v}}$ is an eigenvector of $X^T X$ and $\lambda$ is the corresponding eigenvalue!

# Background: Eigenvectors & Eigenvalues

- Given a square matrix $A \in \mathbb{R}^{N \times N}$, a vector $\boldsymbol{v} \in \mathbb{R}^{N \times 1}$ is an **eigenvector** of $A$ iff there exists some scalar $\lambda$ such that

$$A\boldsymbol{v} = \lambda\boldsymbol{v}$$




Intuition: $A$ scales or stretches $\boldsymbol{v}$ but does not rotate it

- Key property: the eigenvectors of symmetric matrices (e.g., the covariance matrix of a data set) are orthogonal!

# Maximizing the Variance

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

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

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = 2(X^T X)\boldsymbol{v} - 2\lambda \boldsymbol{v}$$

$$\rightarrow 2(X^T X)\hat{\boldsymbol{v}} - 2\lambda\hat{\boldsymbol{v}} = 0 \rightarrow (X^T X)\hat{\boldsymbol{v}} = \lambda\hat{\boldsymbol{v}}$$

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

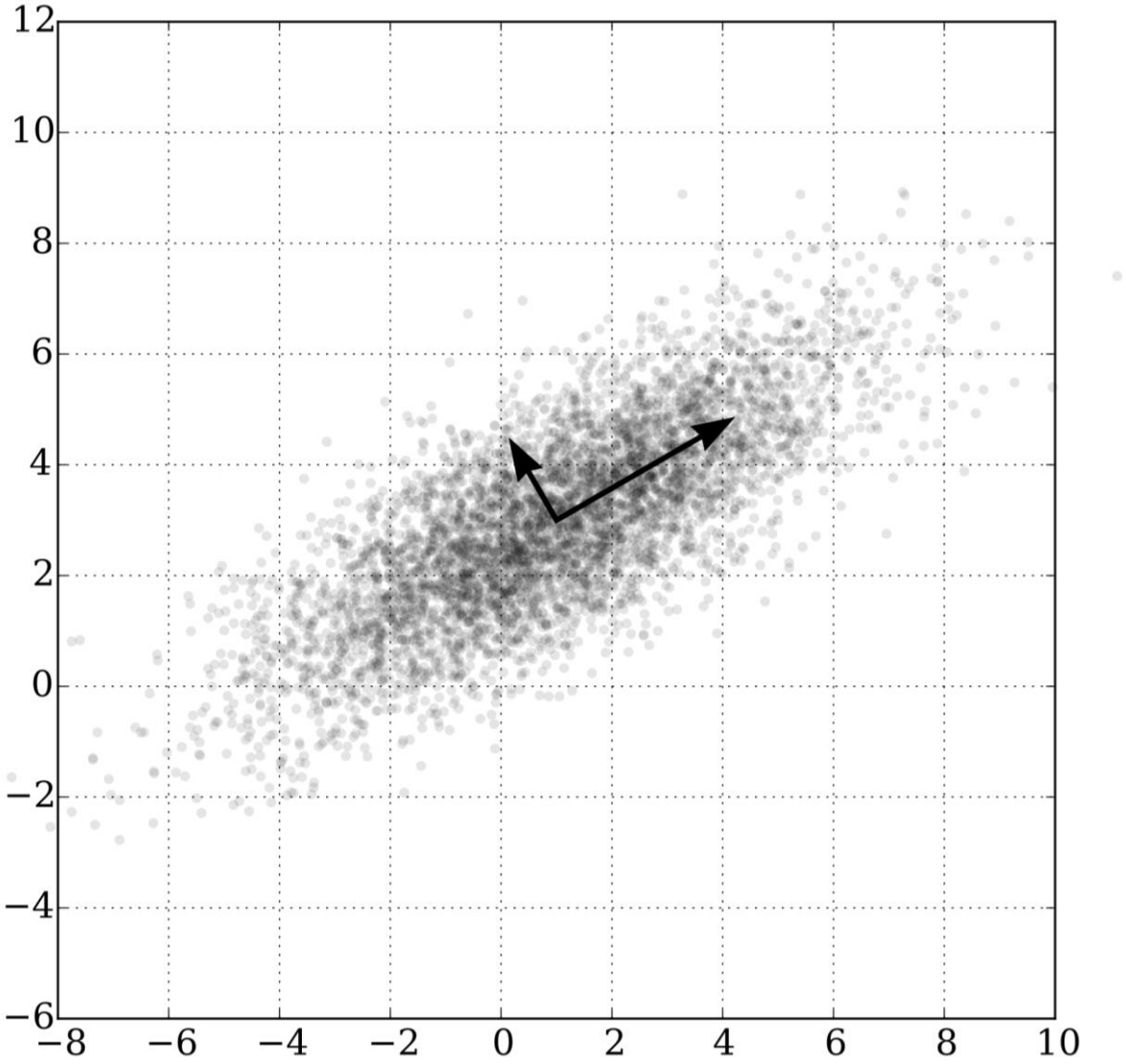
# Maximizing the Variance

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

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

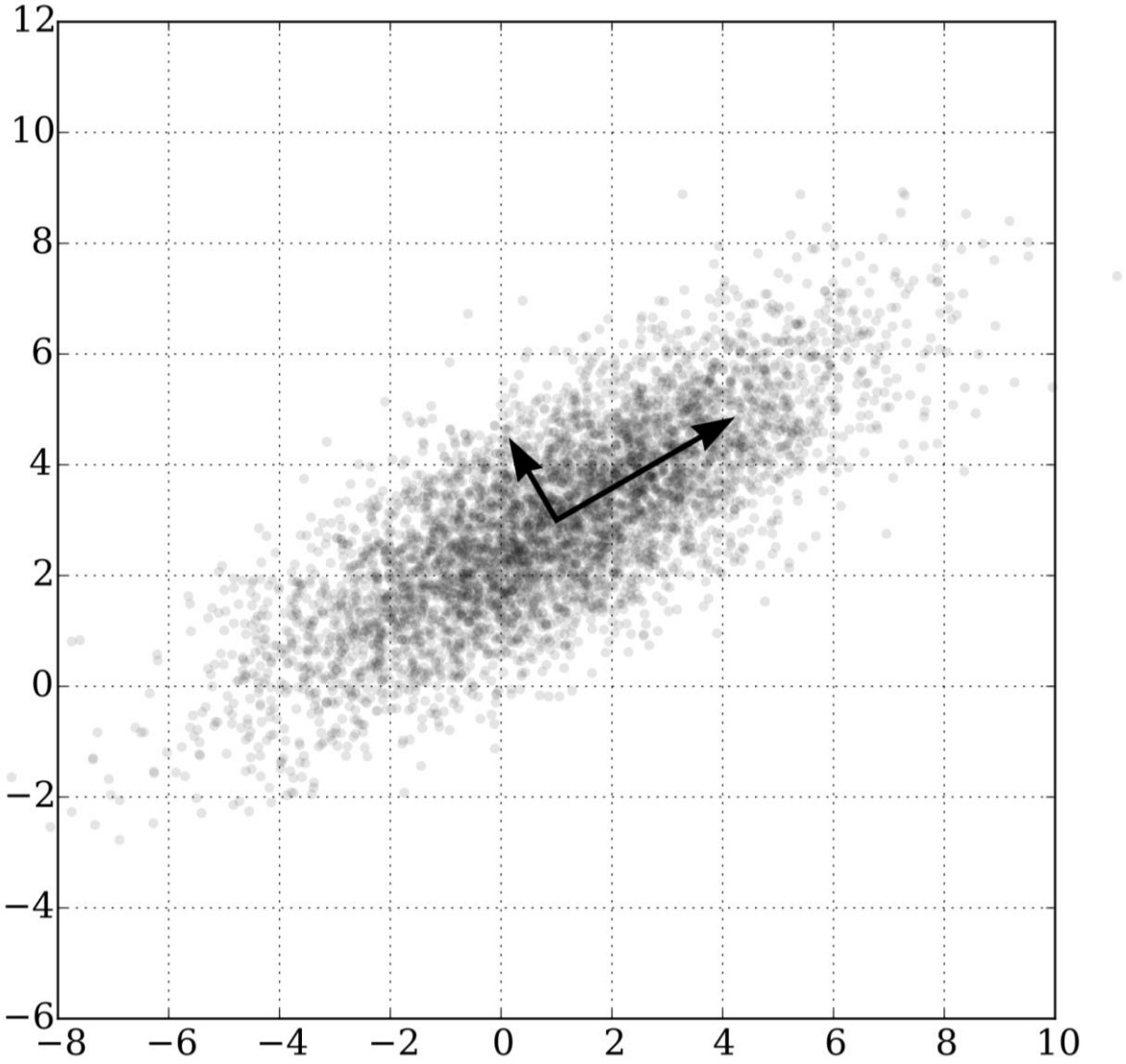
- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue $\lambda_1$
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue $\lambda_2$
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ are orthogonal
- Etc …
- $\lambda_i$ 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

# How can we efficiently find principal components (eigenvectors)?




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

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

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

- 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. Use SVD to compute the eigenvalues and eigenvectors of $X^T X$
3. Collect the top $\rho$ eigenvectors (corresponding to the $\rho$ largest eigenvalues), $V_\rho \in \mathbb{R}^{D \times \rho}$
4. Project the data into the space defined by $V_\rho$, $Z = XV_\rho$

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

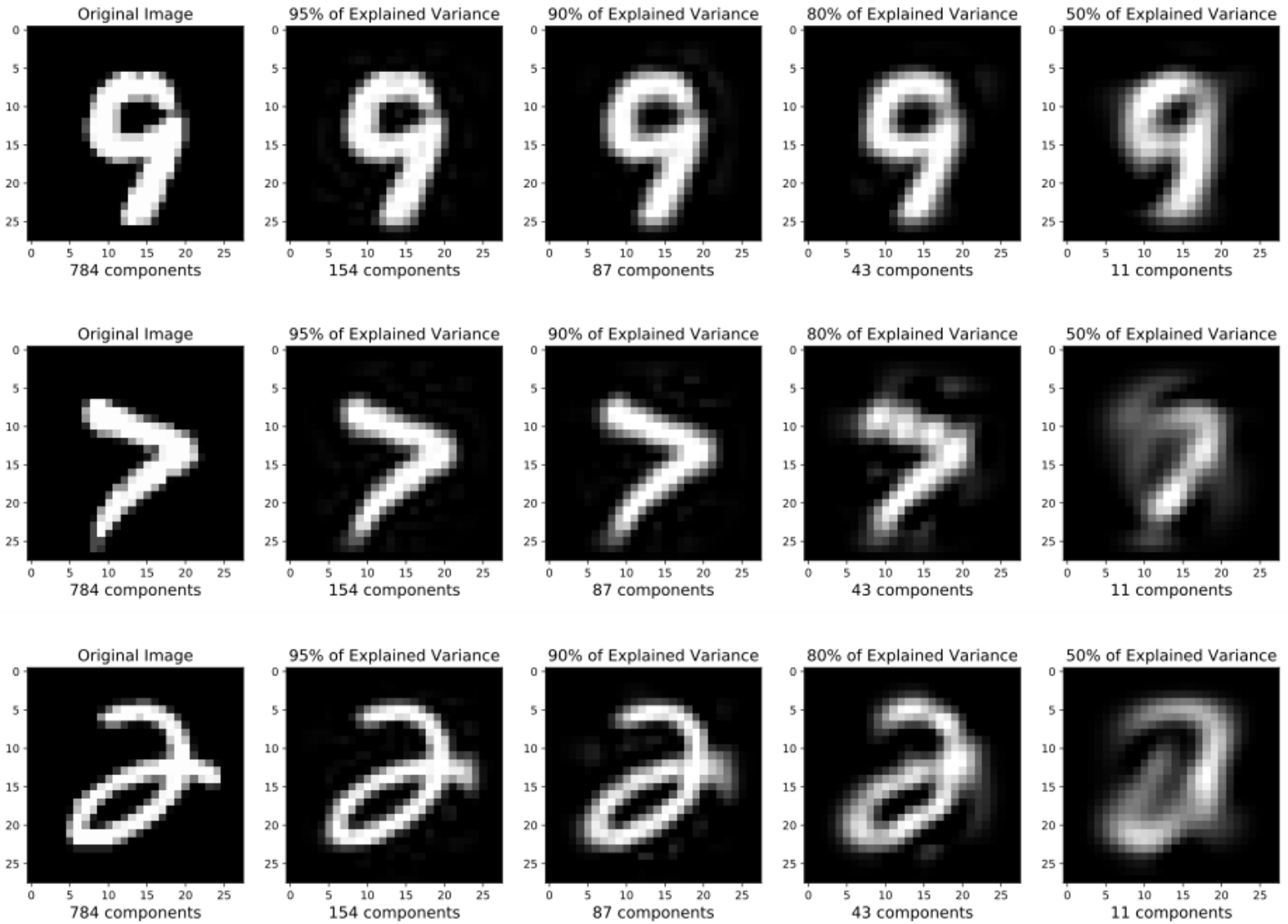
# Choosing the number of PCs

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

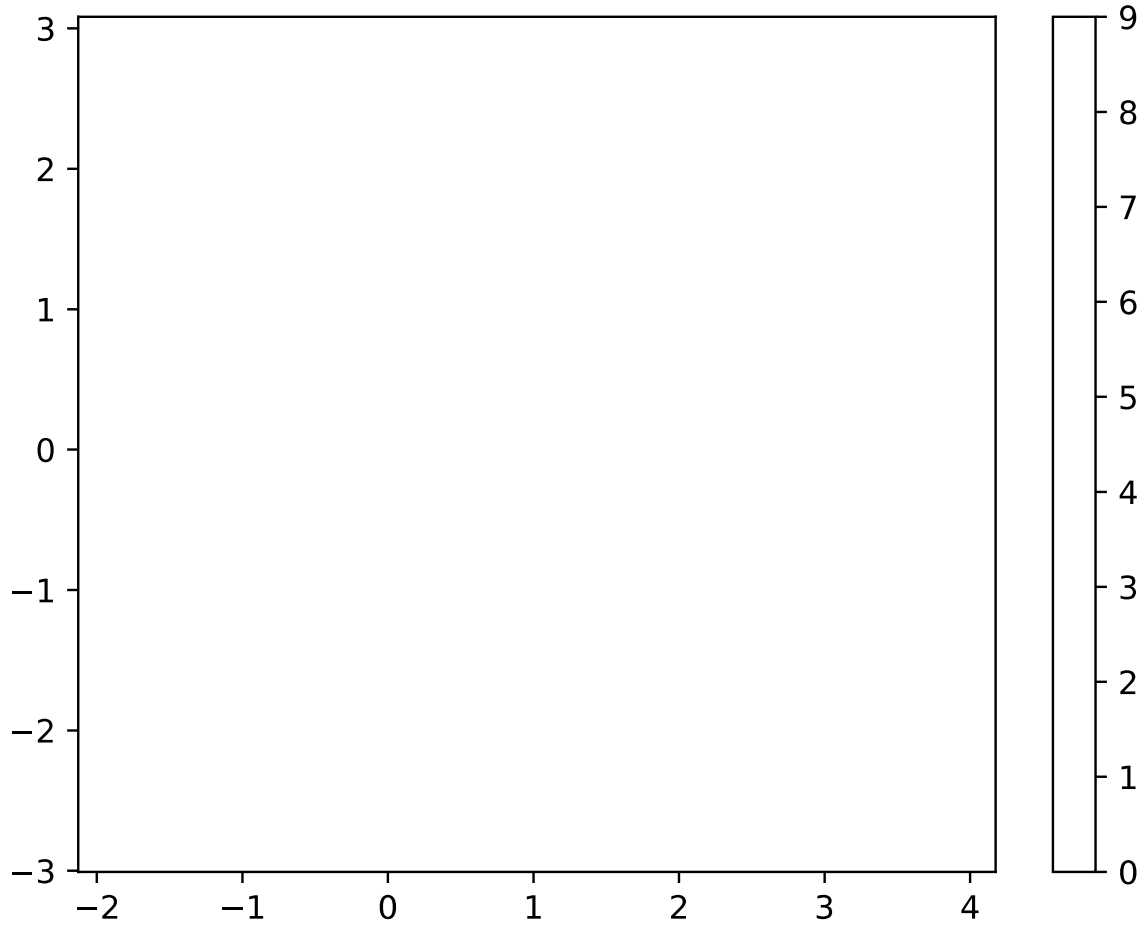



Figure courtesy of Matt Gormley

# PCA Example: MNIST Digits

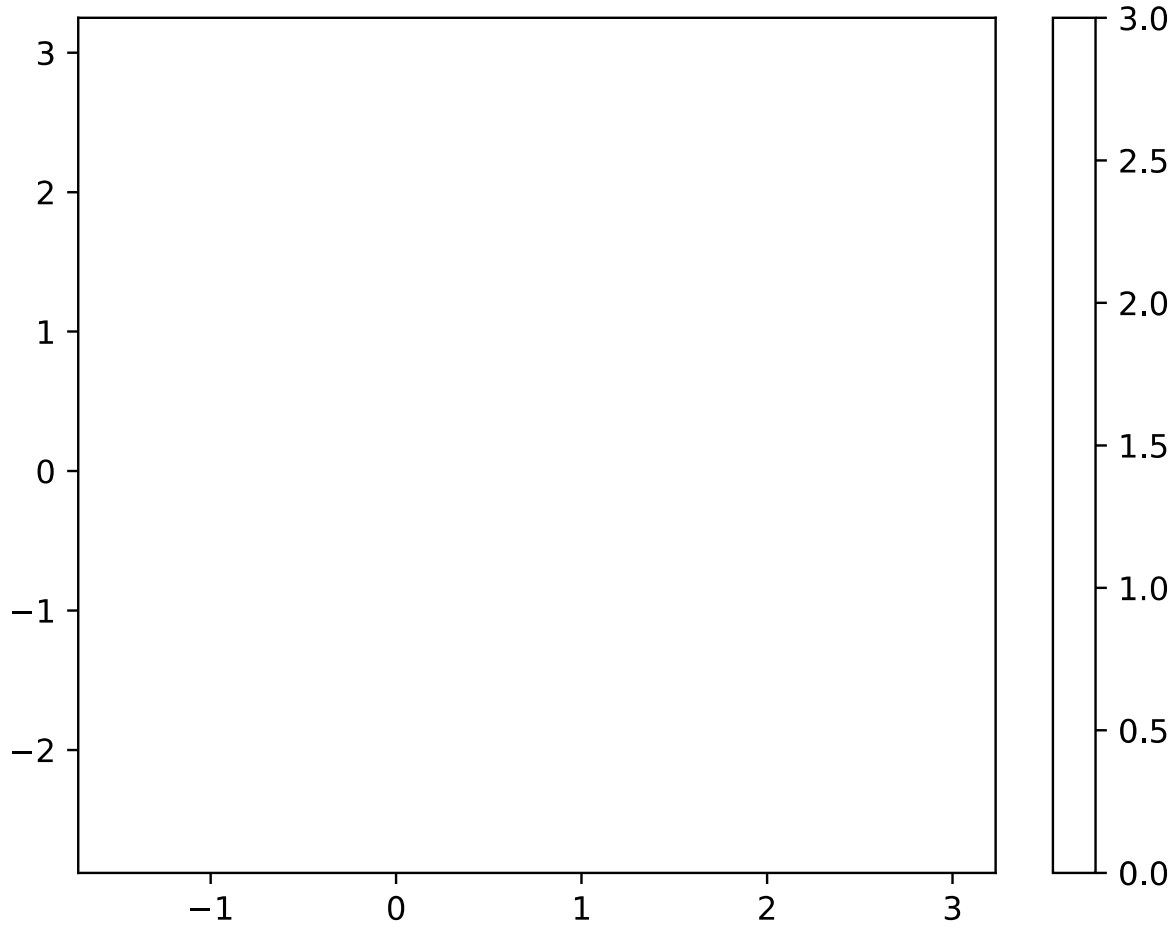
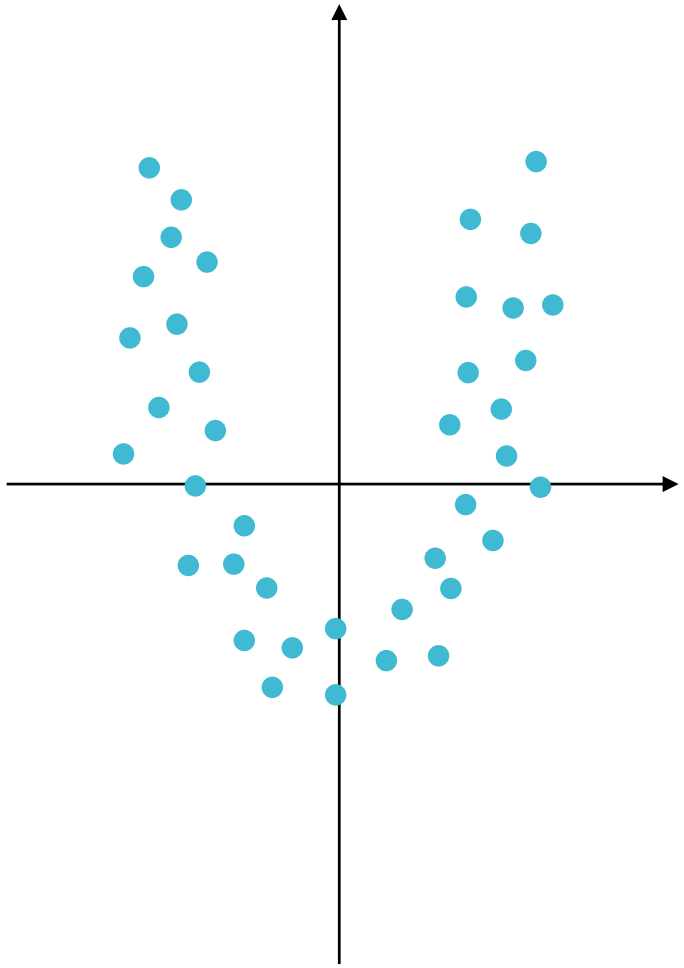


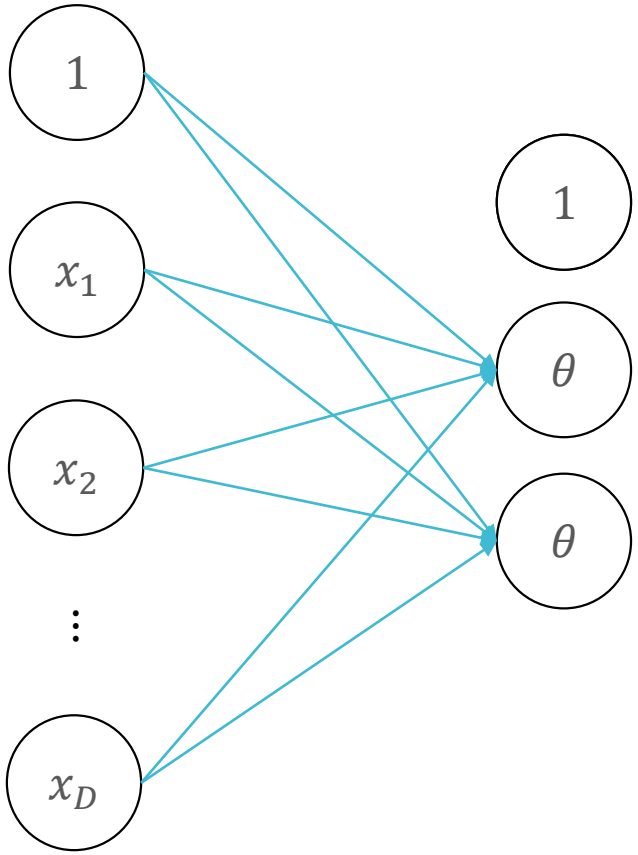

Figure courtesy of Matt Gormley

# Shortcomings of PCA



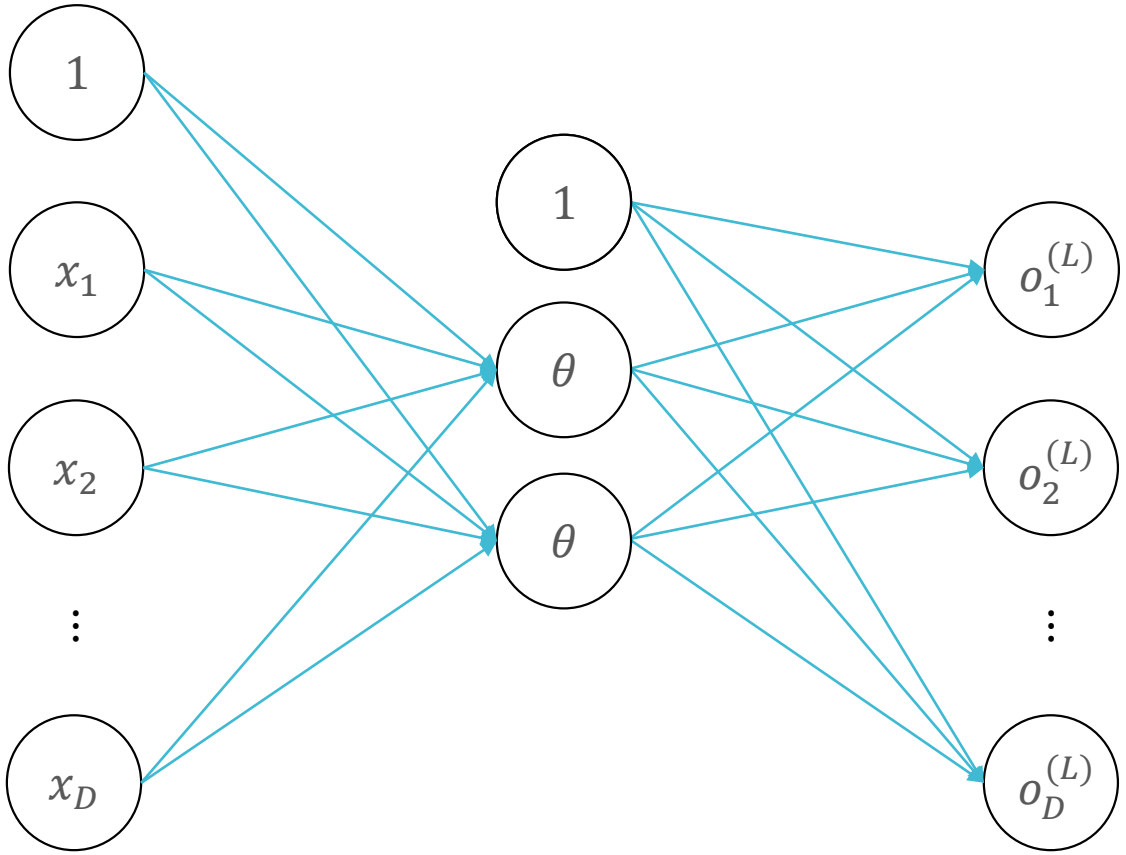
- Principal components are orthogonal (unit) vectors
- Principal components can be expressed as linear combinations of the data

# Autoencoders




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

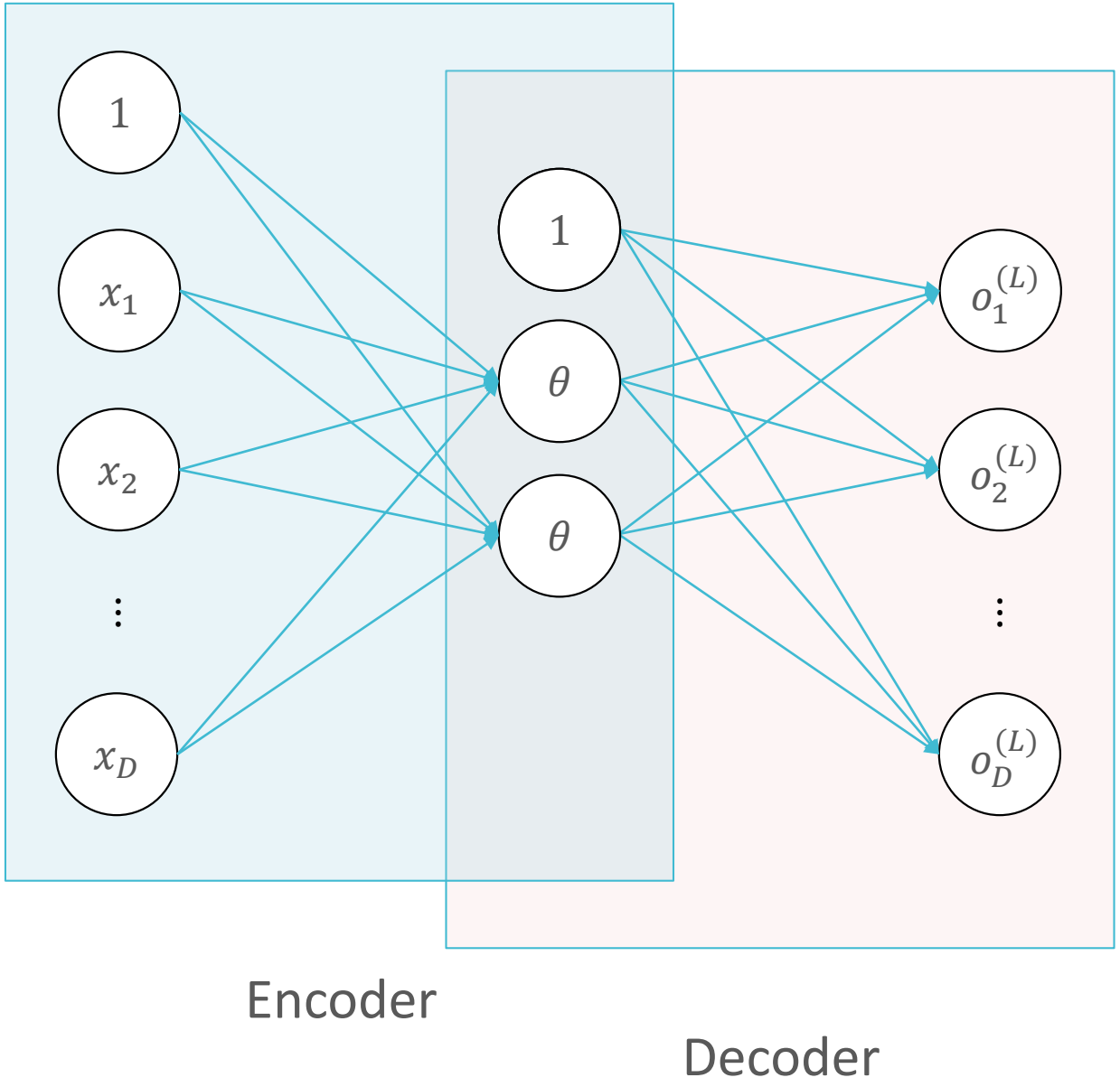
# Autoencoders



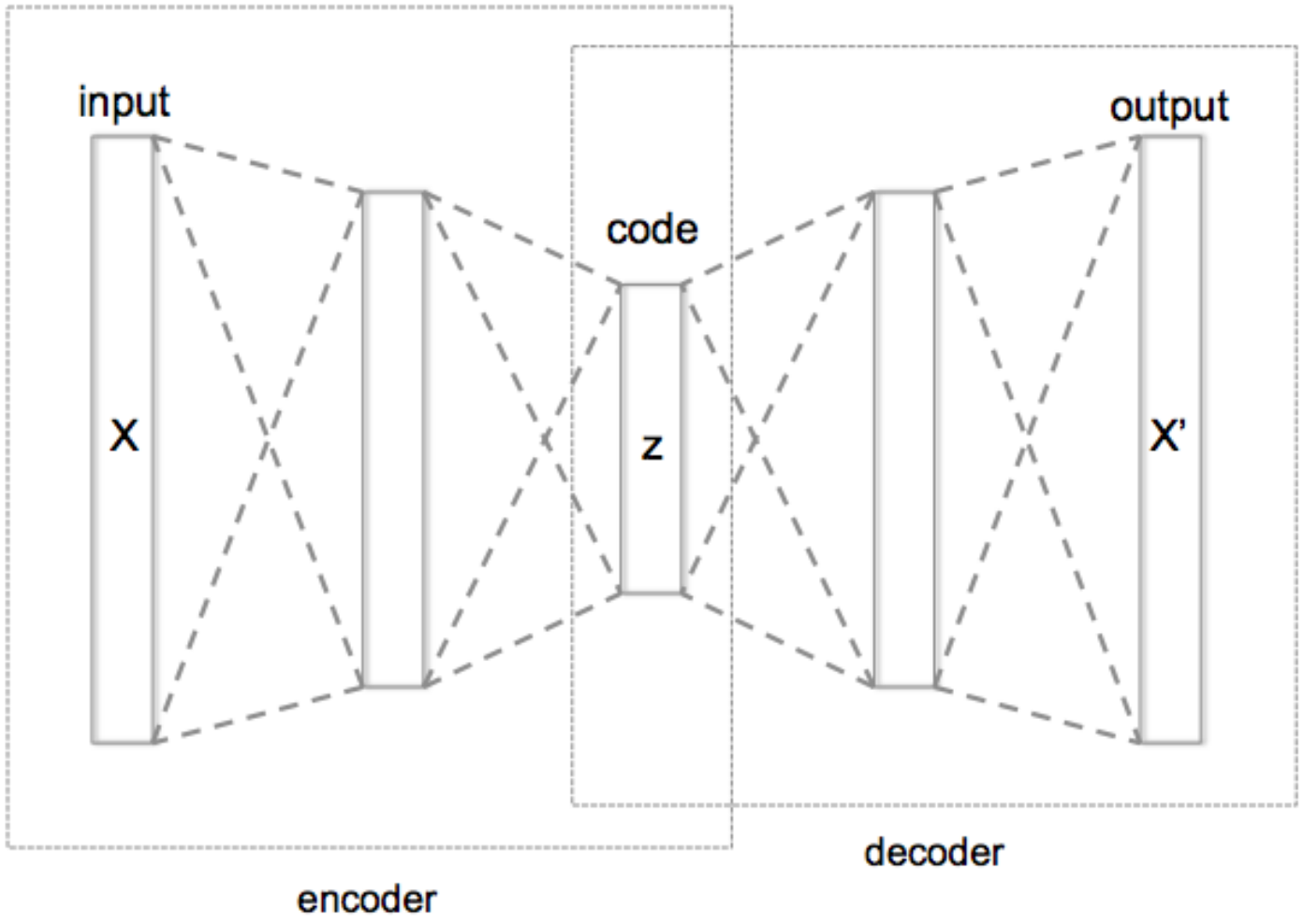

- Learn the weights by minimizing the reconstruction loss:

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

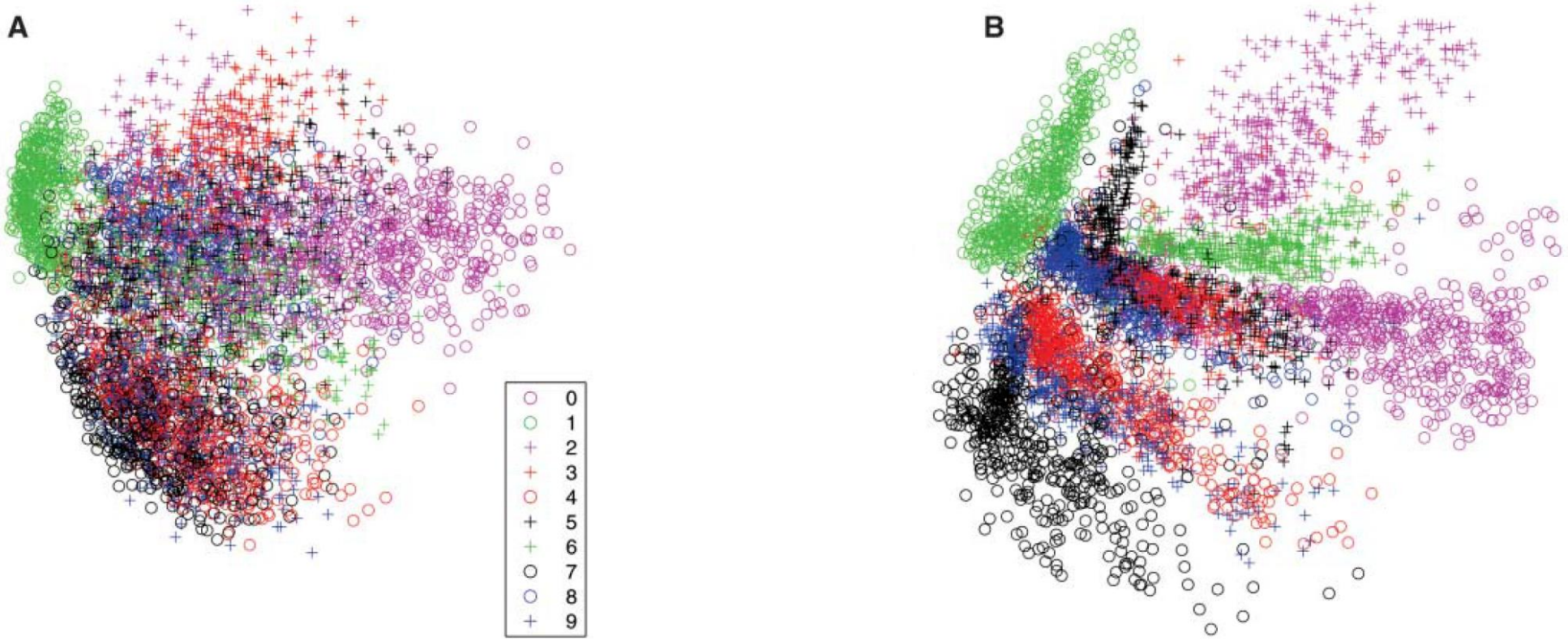
# Autoencoders



# Deep Autoencoders




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




# PCA (A) vs. Autoencoders (B) (Hinton and Salakhutdinov, 2006)



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

# PCA Learning Objectives

You should be able to…

- Define the sample mean, sample variance, and sample covariance of a vector-valued dataset
- Identify examples of high dimensional data and common use cases for dimensionality reduction
- Draw the principal components of a given toy dataset
- Establish the equivalence of minimization of reconstruction error with maximization of variance
- Given a set of principal components, project from high to low dimensional space and do the reverse to produce a reconstruction
- Explain the connection between PCA, eigenvectors, eigenvalues, and covariance matrix
- Use common methods in linear algebra to obtain the principal components