10-301/601: Introduction to Machine Learning Lecture 18: Dimensionality Reduction

Henry Chai

7/10/23

Front Matter

- Announcements
  - PA4 released 6/15, due 7/13 at 11:59 PM
  - Quiz 6: Deep Learning & Learning Theory on 7/11 (tomorrow!)
- Recommended Readings
  - Murphy, Chapters 12.2.1 - 12.2.3
  - Daumé III, Chapter 15: Unsupervised Learning

Learning Paradigms

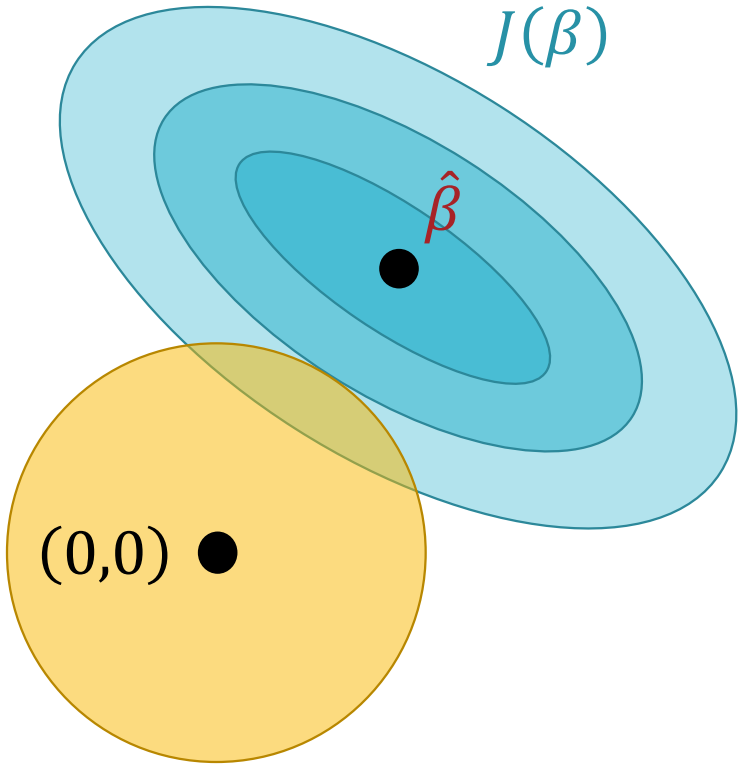
- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N}$
  - Clustering
  - Dimensionality reduction

Learning Paradigms

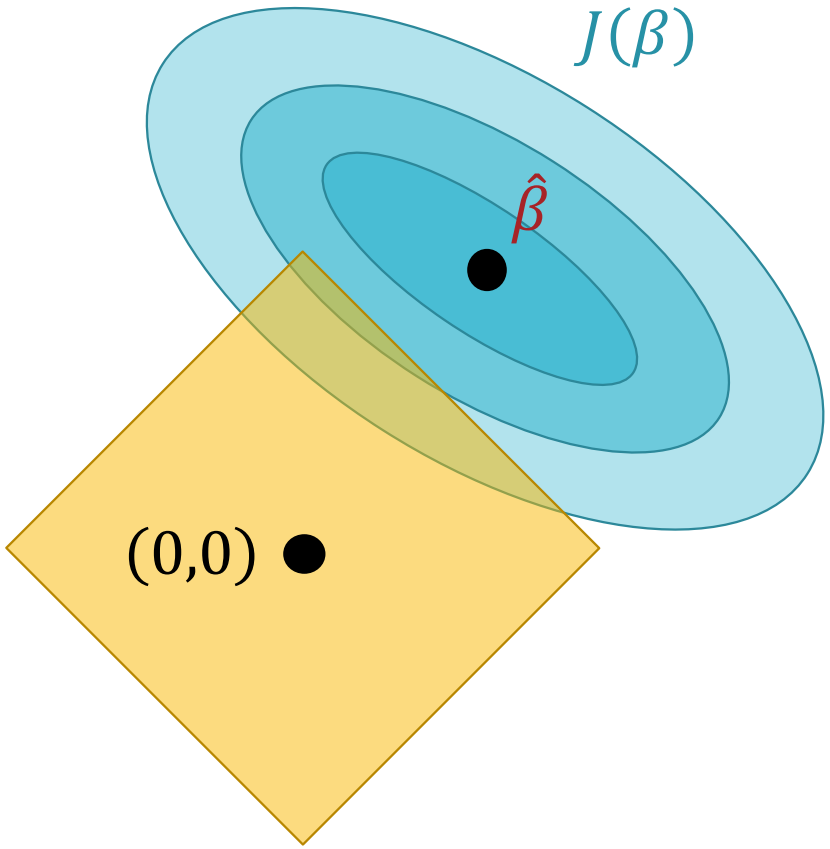
- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N}$
  - Clustering
  - **Dimensionality reduction**

Dimensionality Reduction

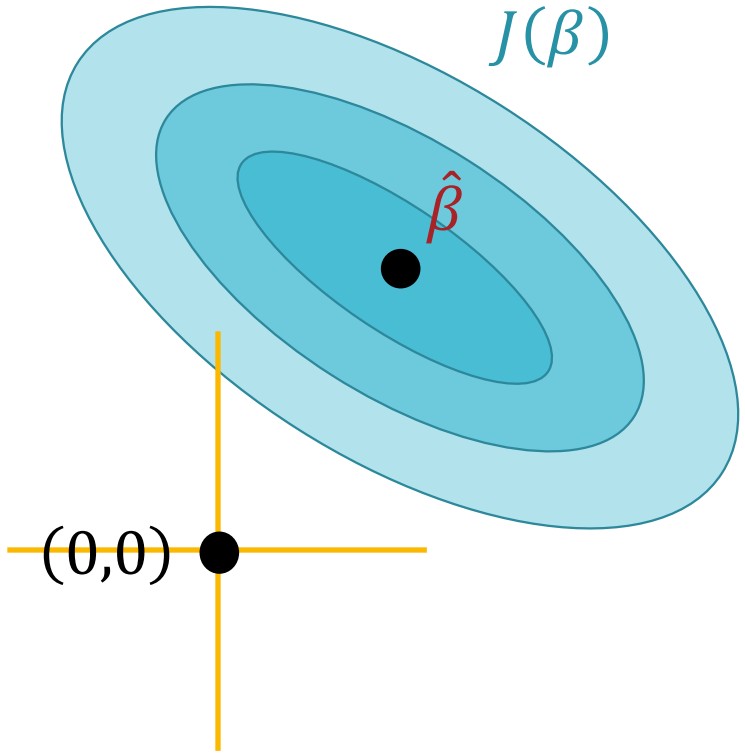
- Goal: given some unlabeled data set, learn a latent (typically lower-dimensional) representation
- Use cases:
  - Reducing computational cost (runtime, storage, etc…)
  - Improving generalization
  - Visualizing data
- Applications:
  - Recommender system: demographic info., usage behavior, current recommendations
  - Inspection of parts: physical measurements of components – most are typically irrelevant
  - Text & image/video



Ridge or $L2$

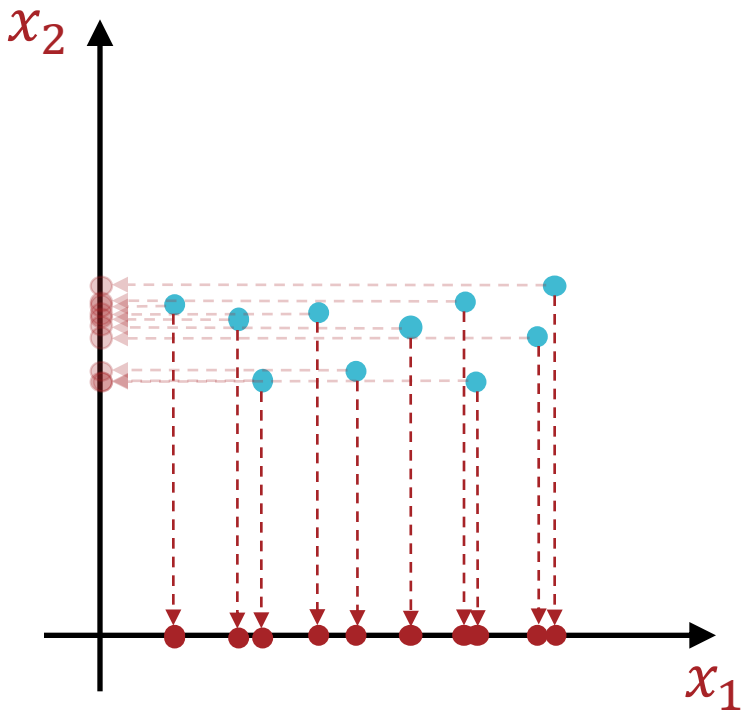


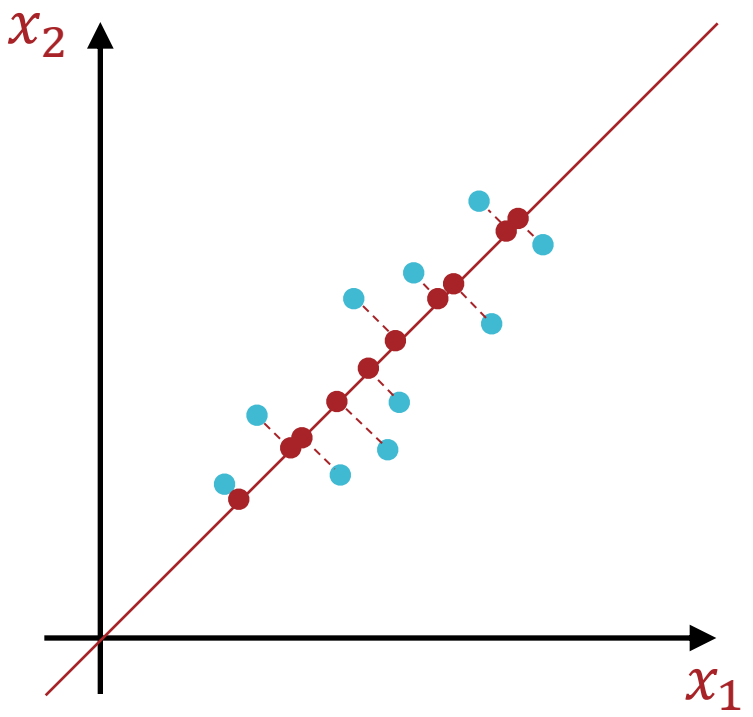
Lasso or $L1$



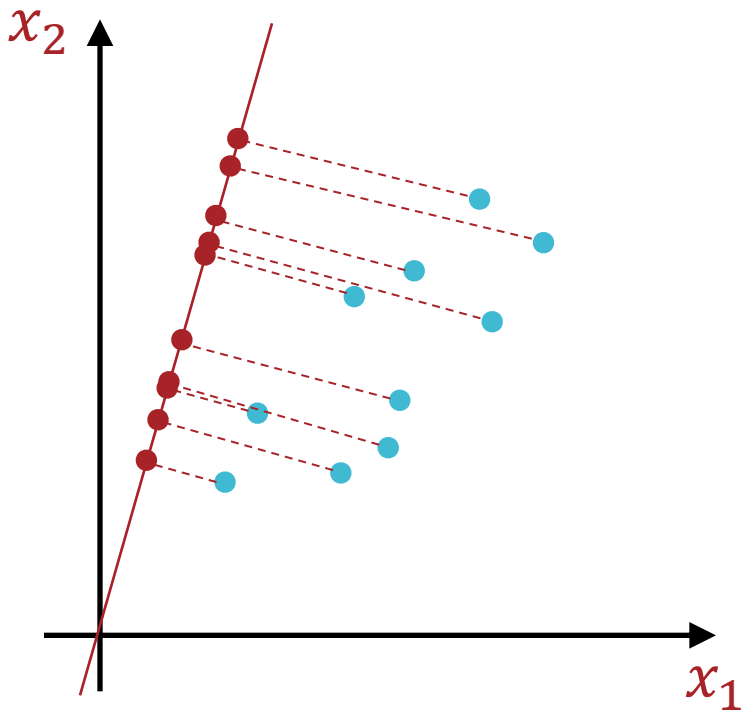
$L0$

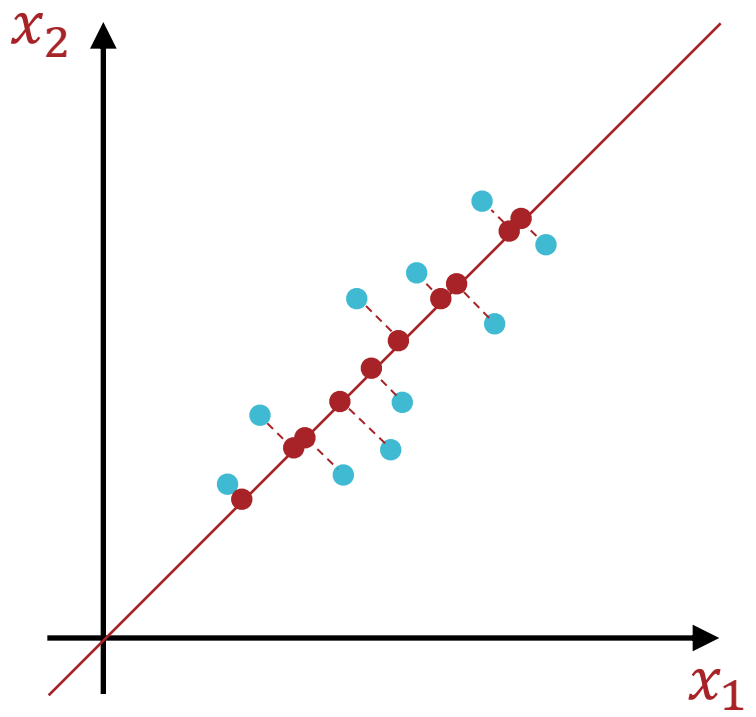
Recall: $L1$ (or $L0$) Regularization



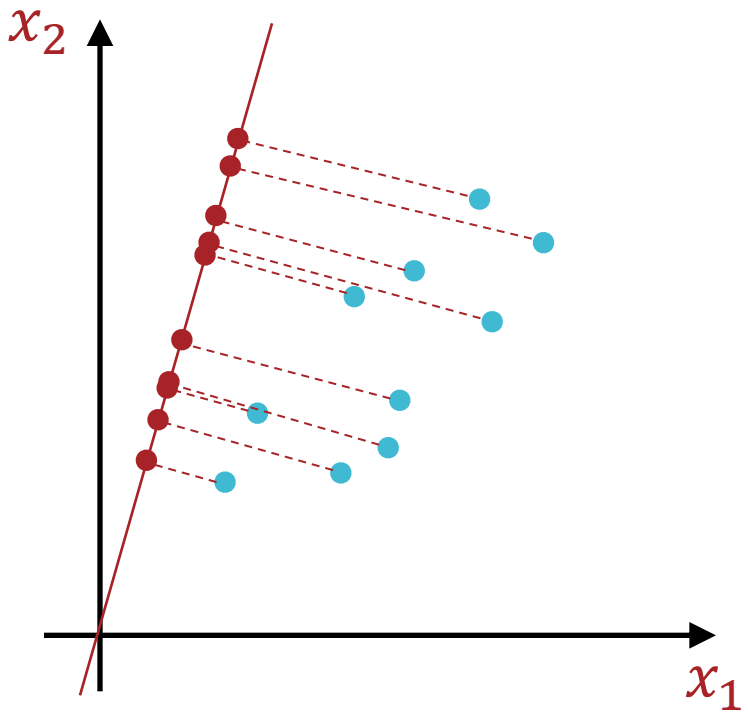


Feature Elimination

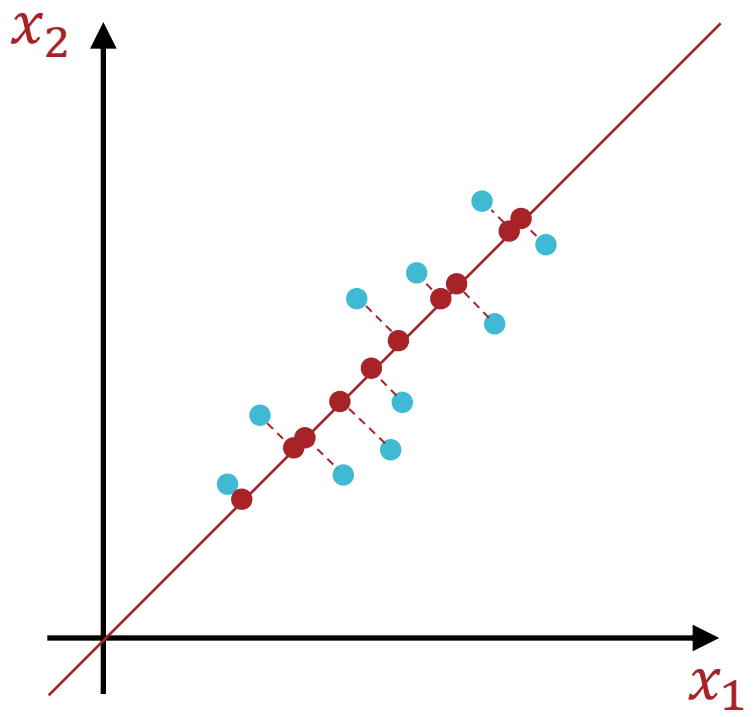




Feature Reduction



Option A



Option B

Which projection do you prefer?

Which projection do you prefer?

Option A

Option B

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

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

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

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

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

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

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

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

$v^T \tilde{x}^{(n)} \in \mathbb{R}$

Minimizing the Reconstruction Error

⇕

Maximizing the Variance

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

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

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

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

variance of a zero-mean random variable

$$\hat{v} = \underset{v:\|v\|_2=1}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} (v^T \tilde{x}^{(n)})^2 v^T v - 0$$

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

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

Maximizing the Variance

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

$\rightarrow$ s.t. $\|v\|_2^2 = 1$

$\rightarrow \|v\|_2^2 - 1 = 0$

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

$$= v^T (X^T X) v - \lambda(v^T v - 1)$$

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

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

$\rightarrow$ $\hat{v}$ is an ~~eigenvalue~~ eigenvector of $X^T X$!

but which one?

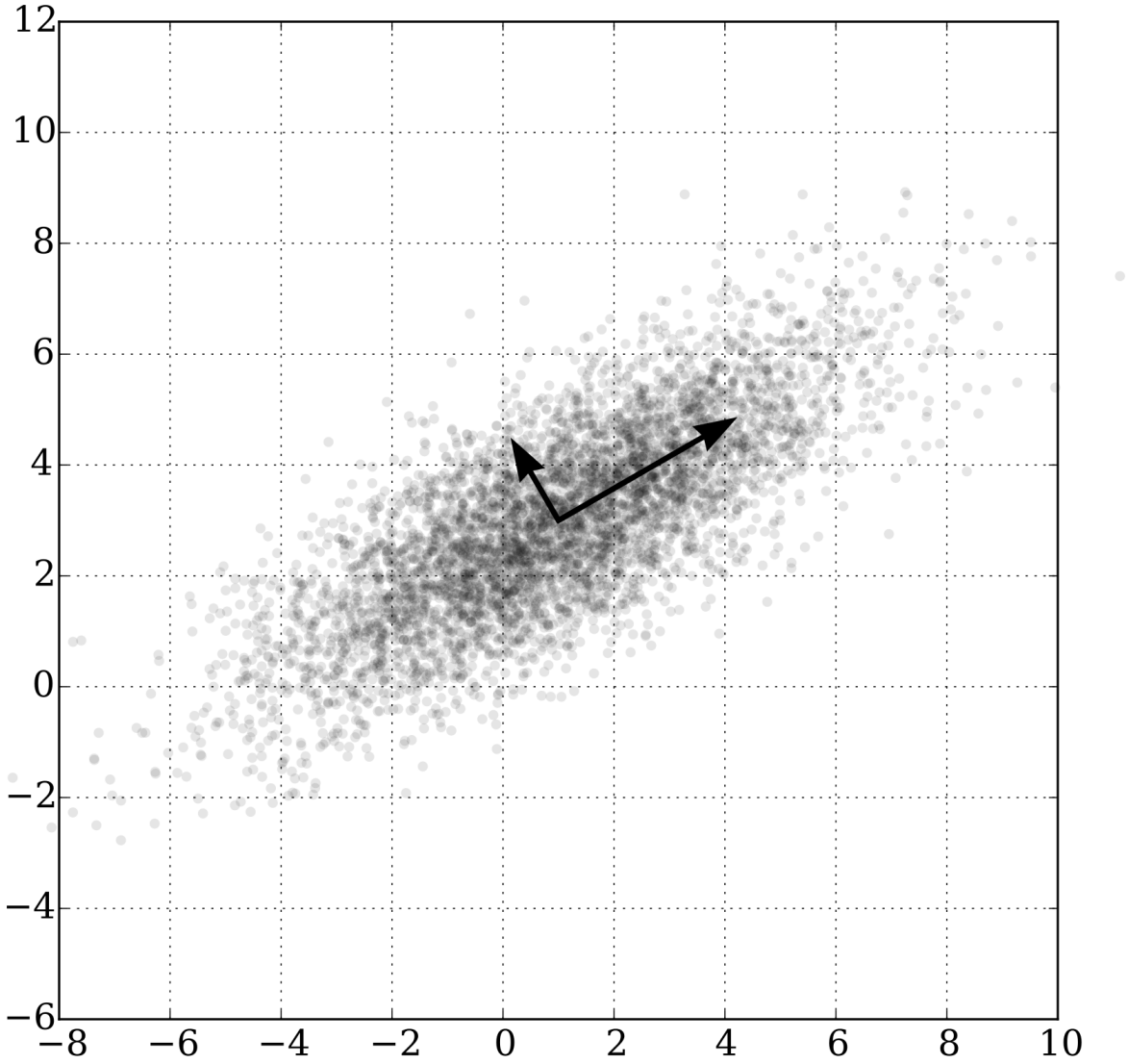
Maximizing the Variance

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

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

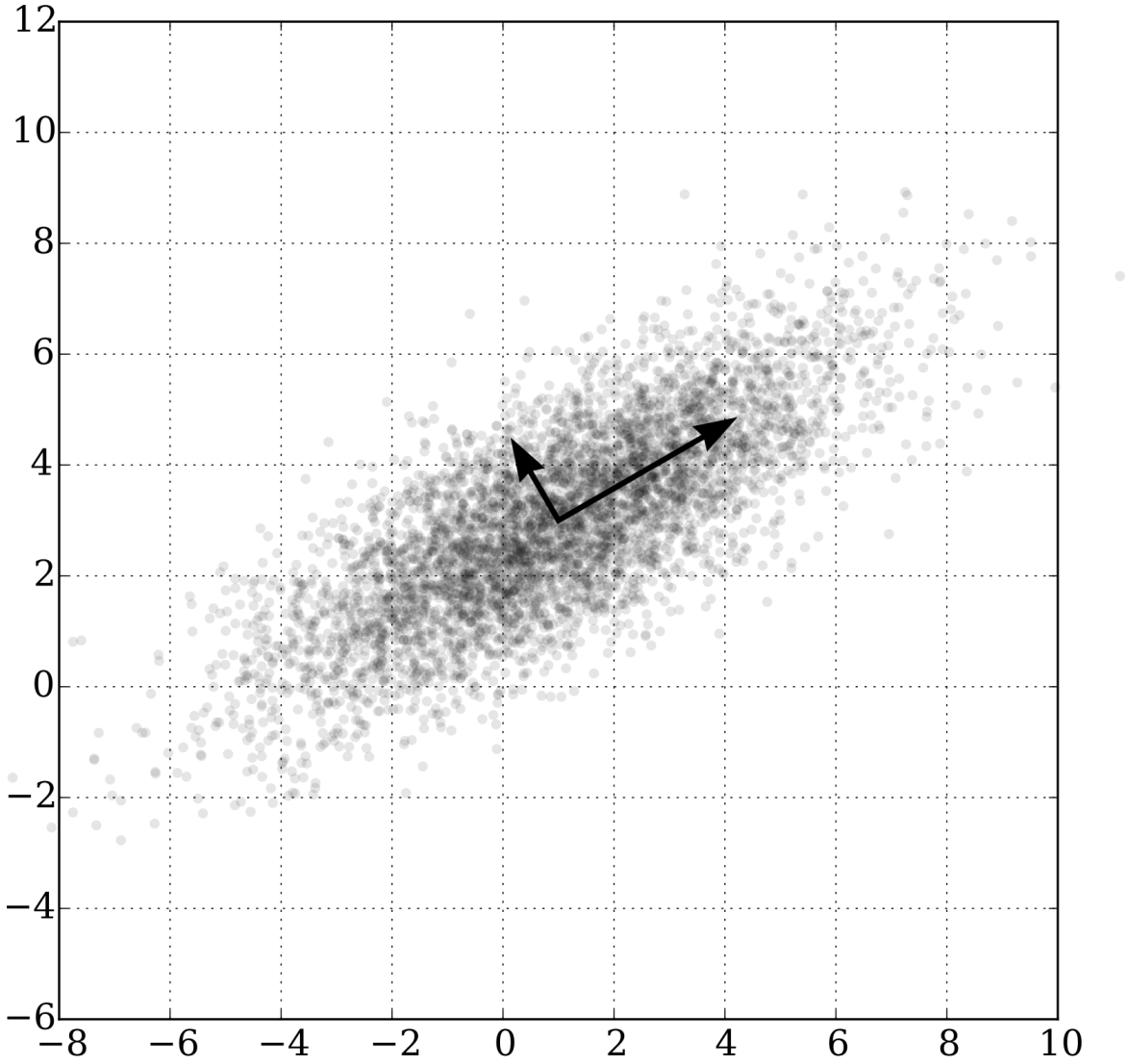
- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_2
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$ 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

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 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. 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

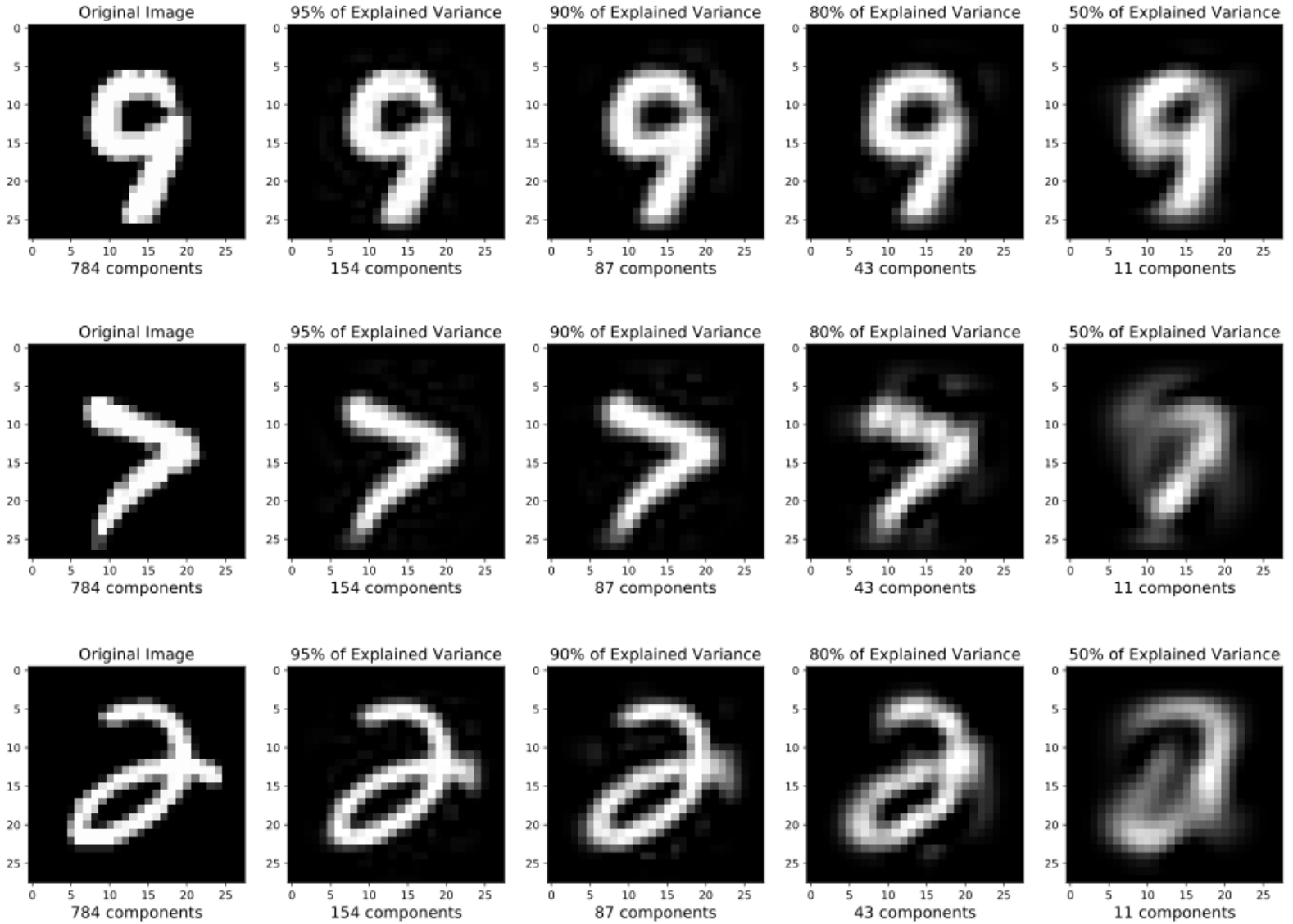
Choosing the number of PCs

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

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

- 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

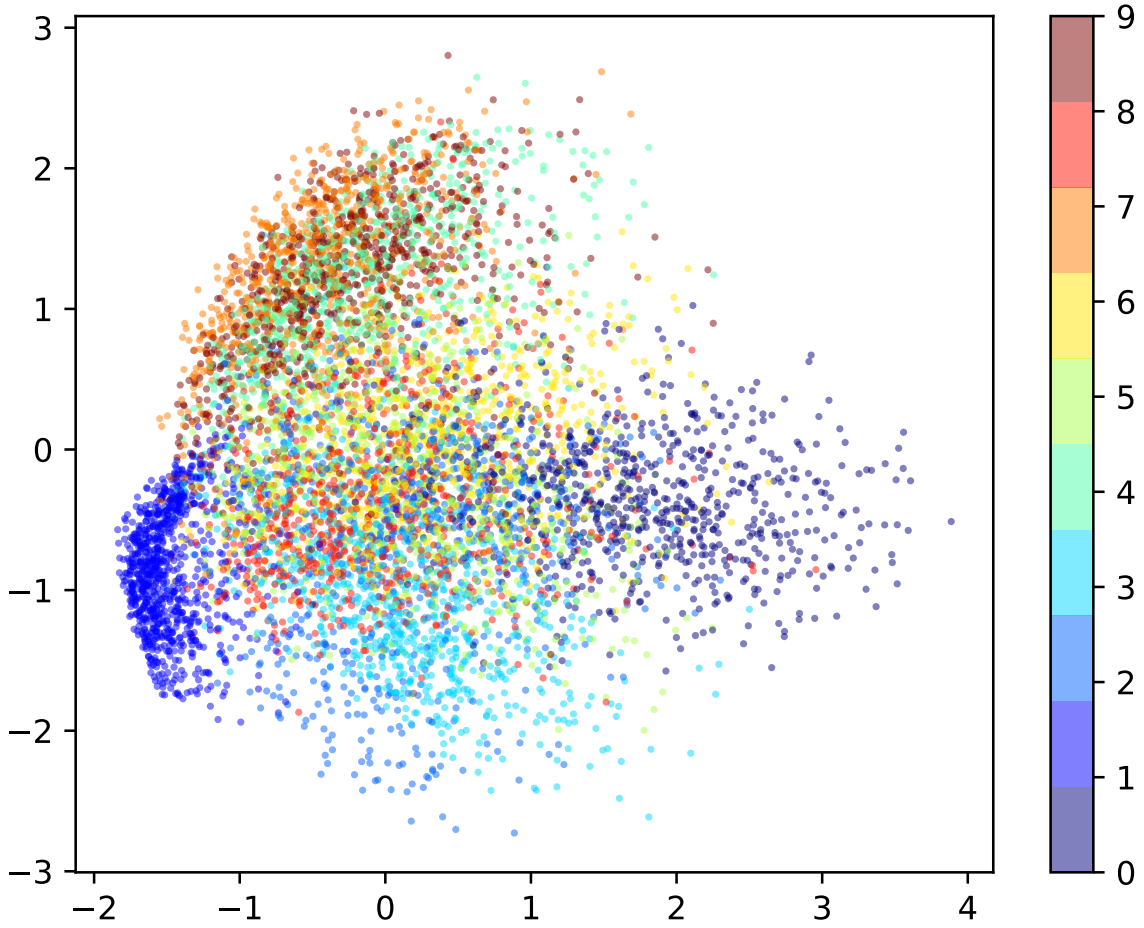


Figure courtesy of Matt Gormley

PCA Example: MNIST Digits

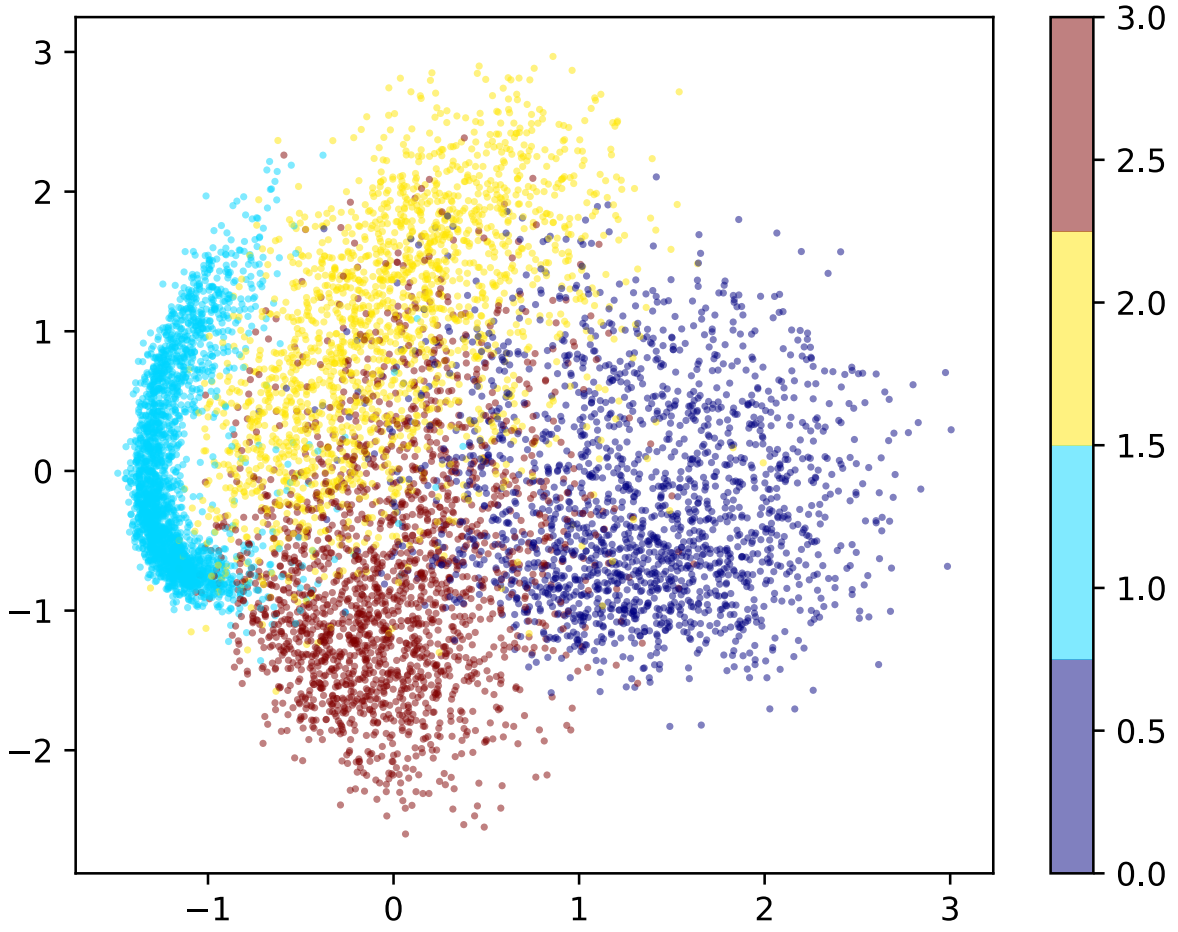
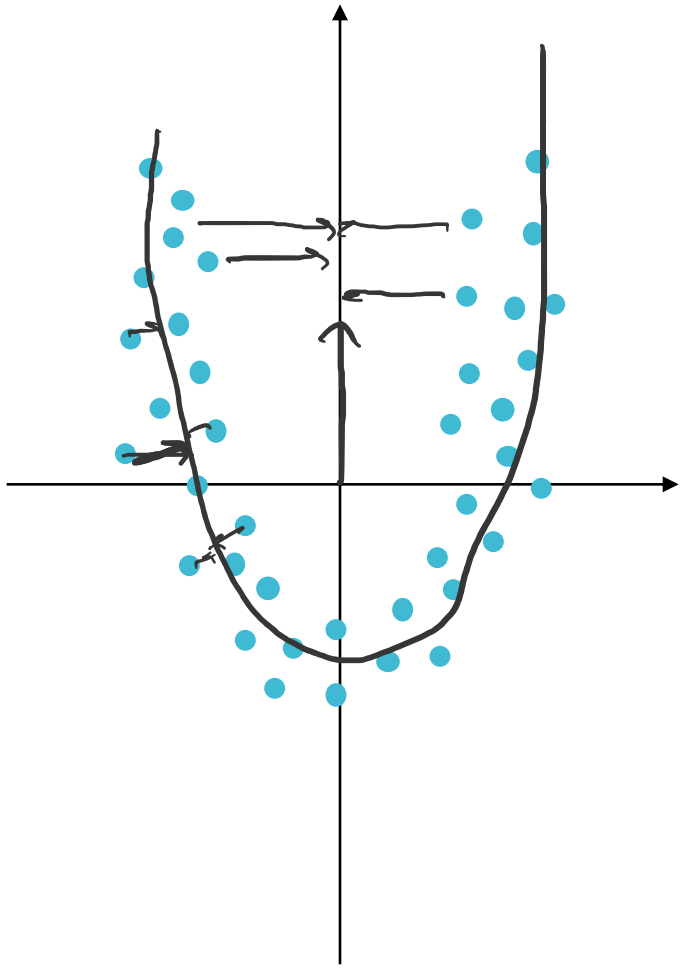


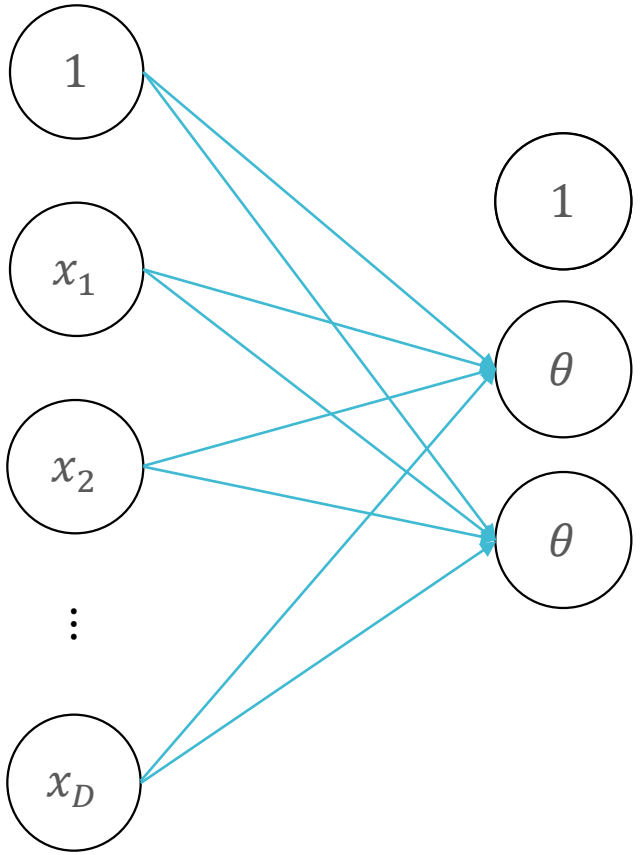
Figure courtesy of Matt Gormley

Shortcomings of PCA



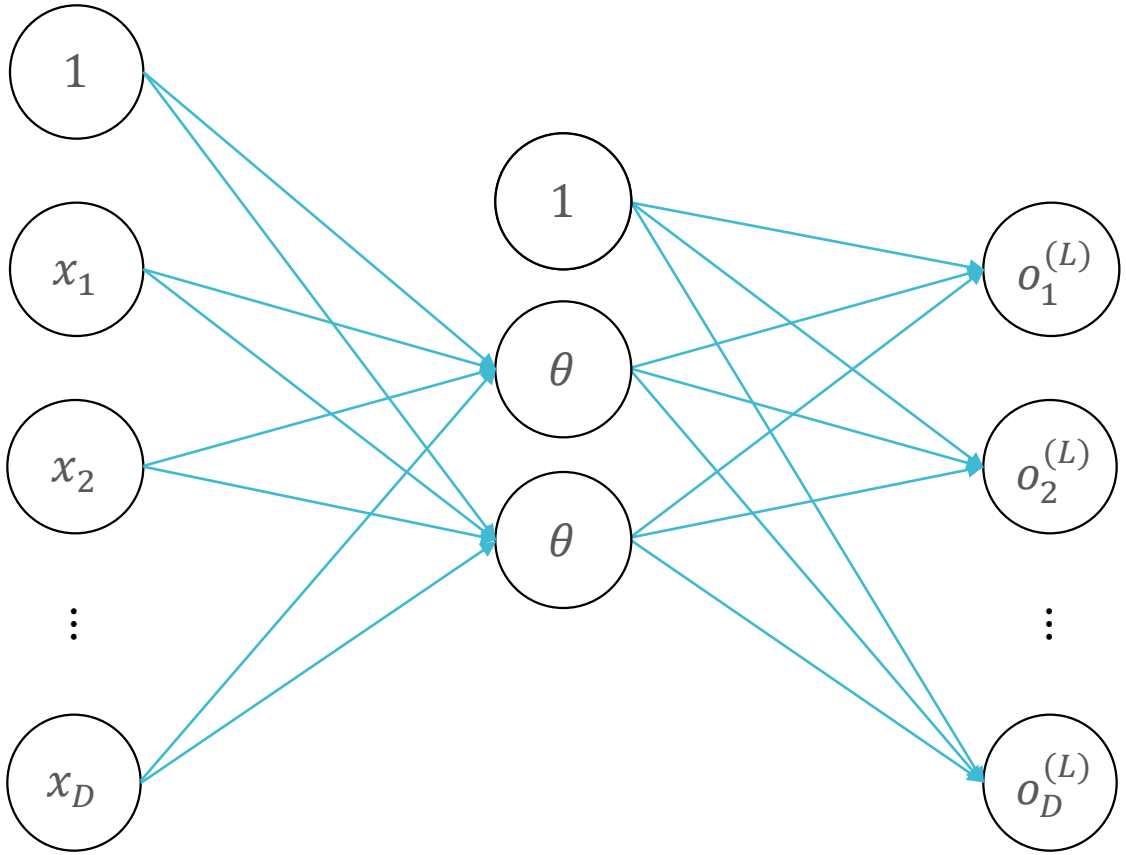
- PCs are constrained to be linear

Autoencoders



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

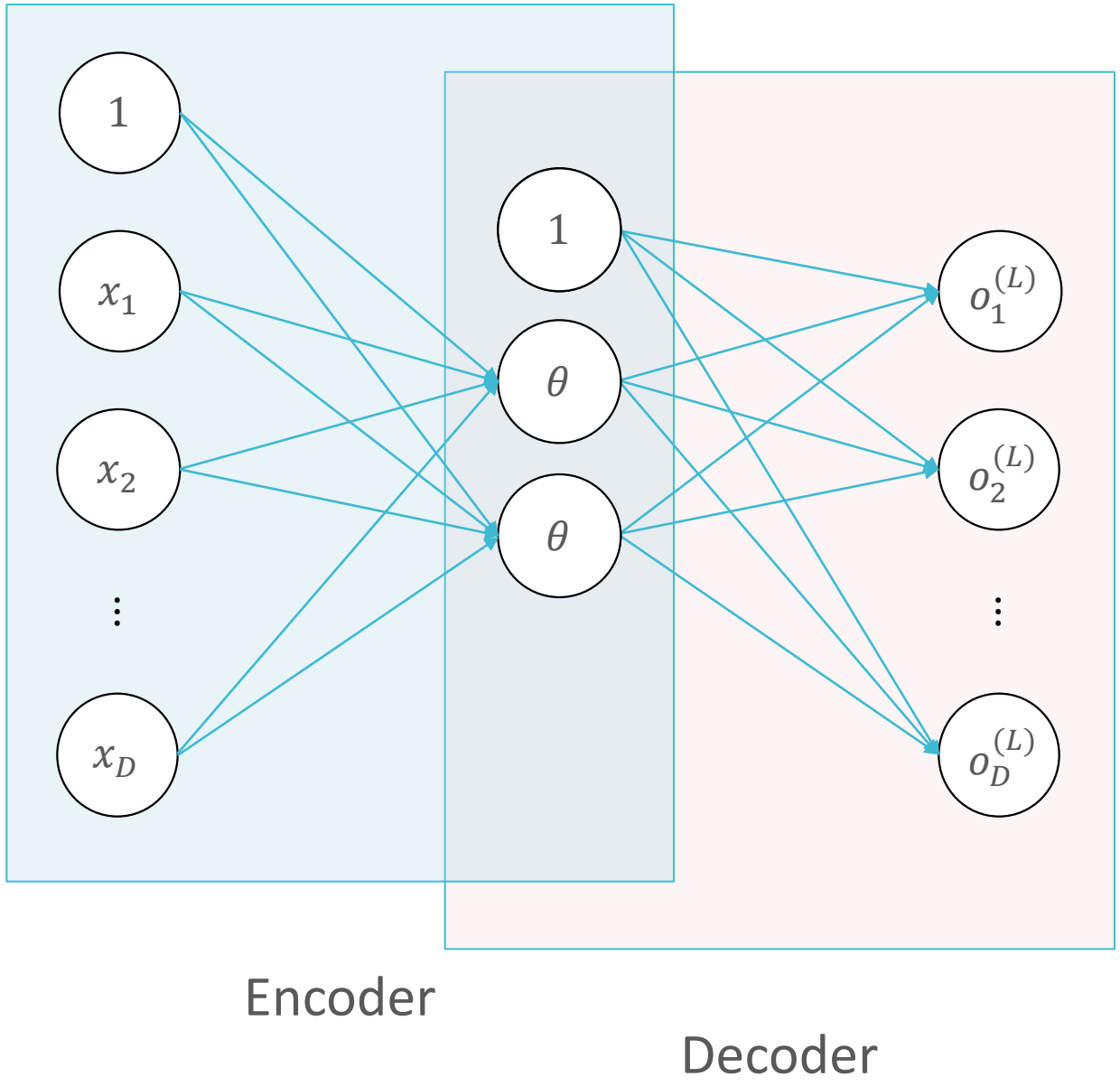
Autoencoders



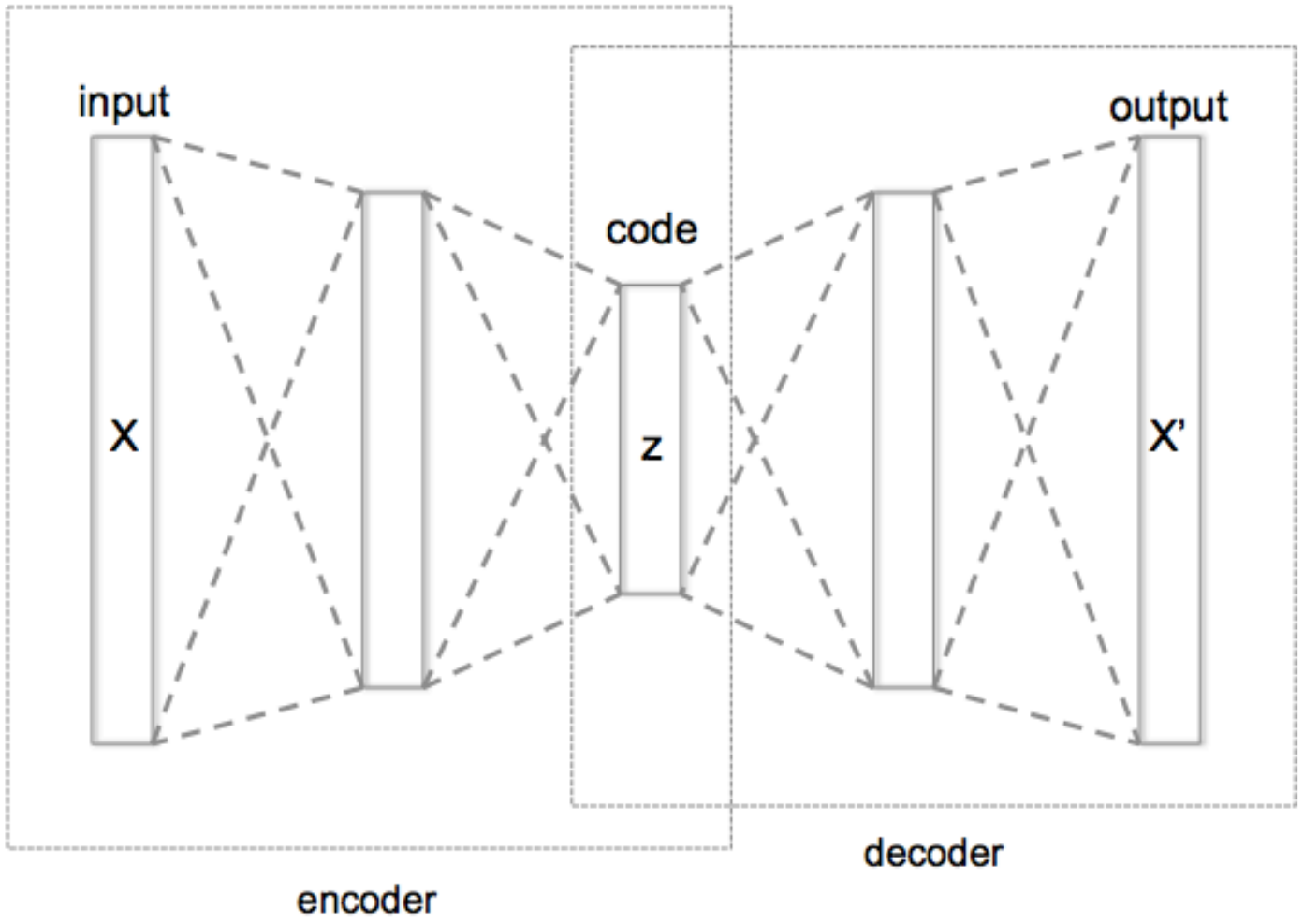
- Learn the weights by minimizing the reconstruction loss:

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

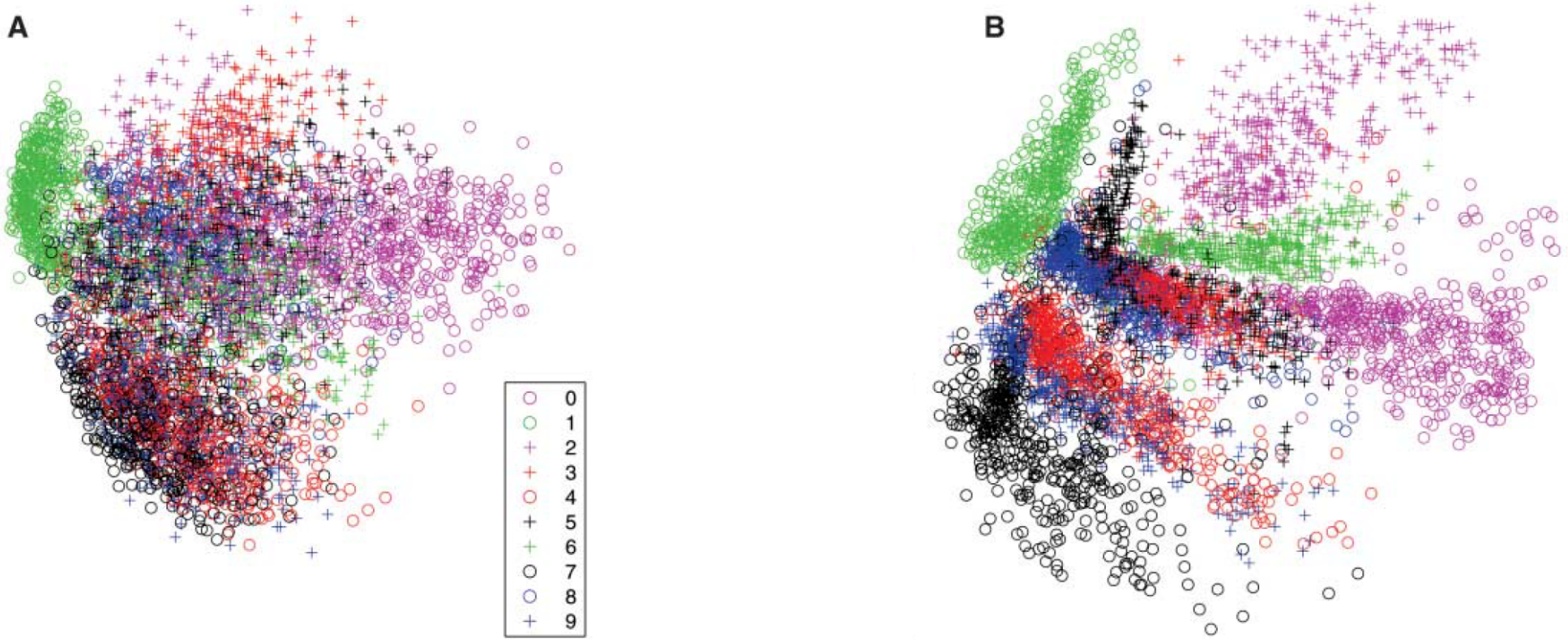
Autoencoders



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

Key Takeaways

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