10-301/601: Introduction to Machine Learning Lecture 24: Dimensionality Reduction

Henry Chai

6/5/25

Front Matter

- Announcements
  - HW5 released on 6/3, due 6/6 (tomorrow) at 11:59 PM
  - HW6 to be released on 6/6 (tomorrow), due 6/10 at 11:59 PM
  - Quiz 3 on 6/6 (tomorrow) at 11:00 AM in BH A36

Learning Paradigms

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

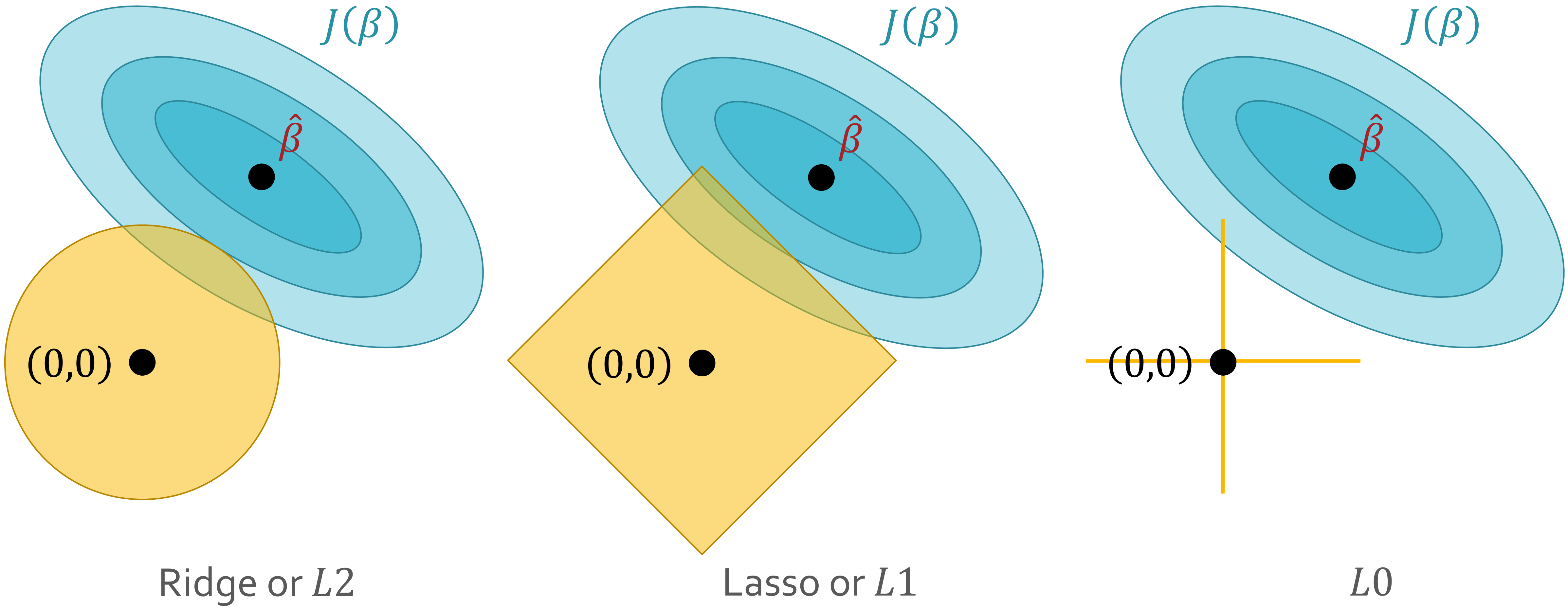
Learning Paradigms

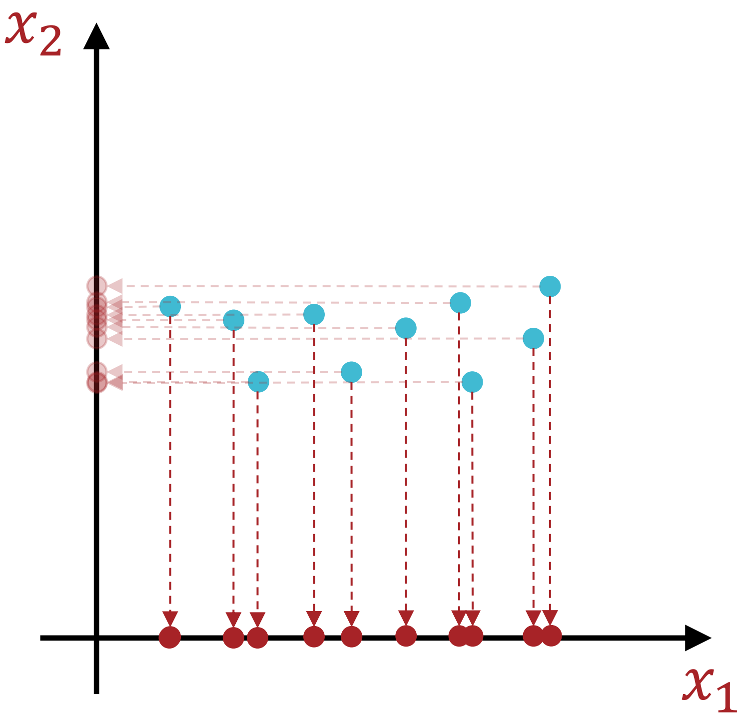
- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(n)}\right\}_{n=1}^{N}$
  - Clustering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

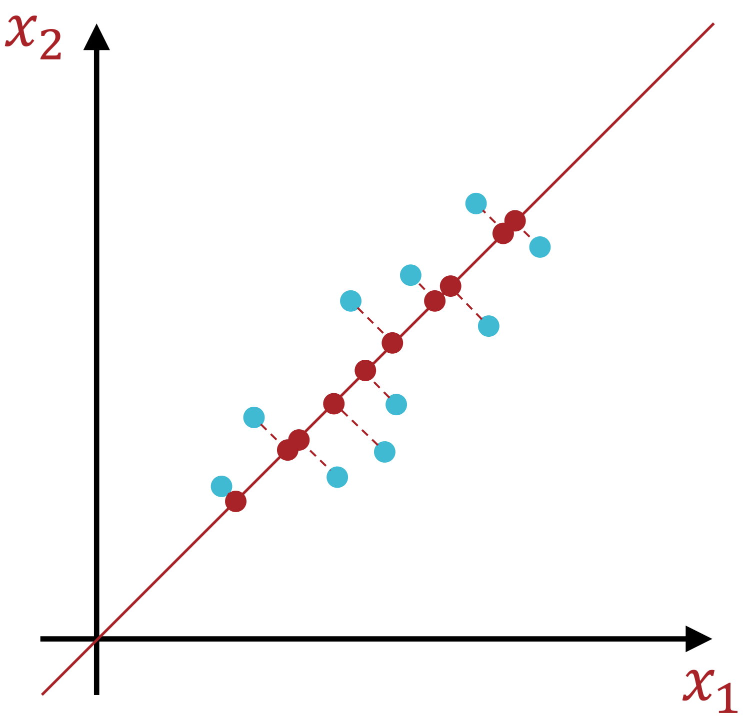
Dimensionality Reduction

- 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:
  - High-resolution images/videos
  - Text data
  - Financial or transaction data

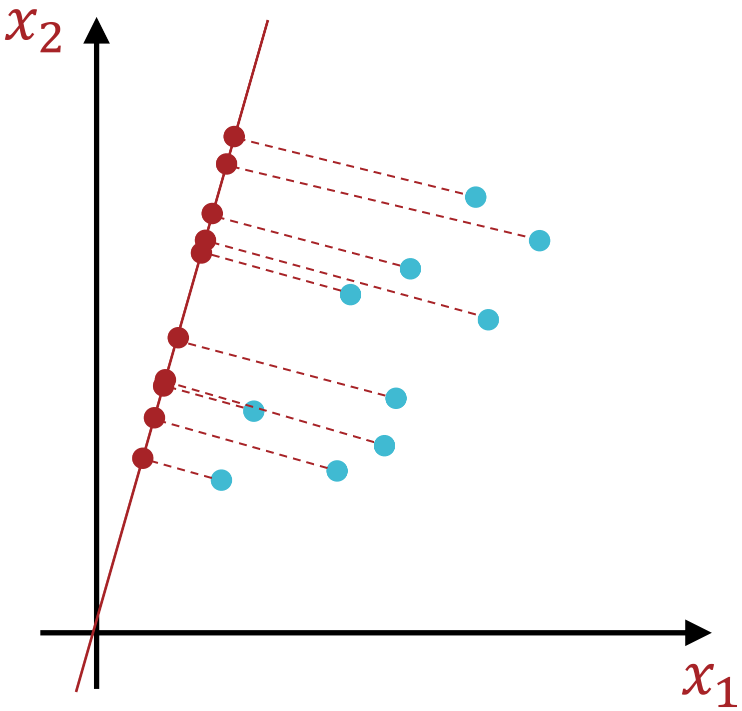
Recall: $L1$ (or $L0$) Regularization

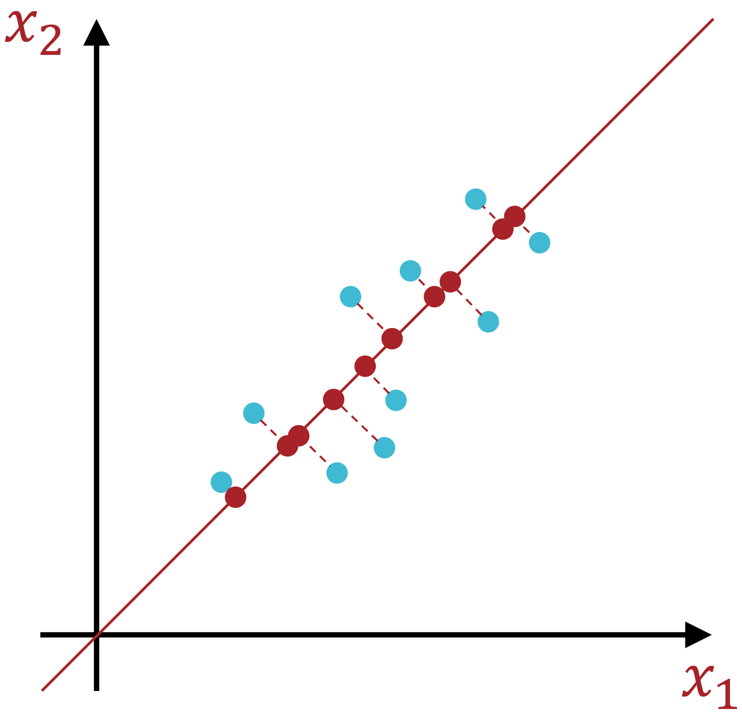




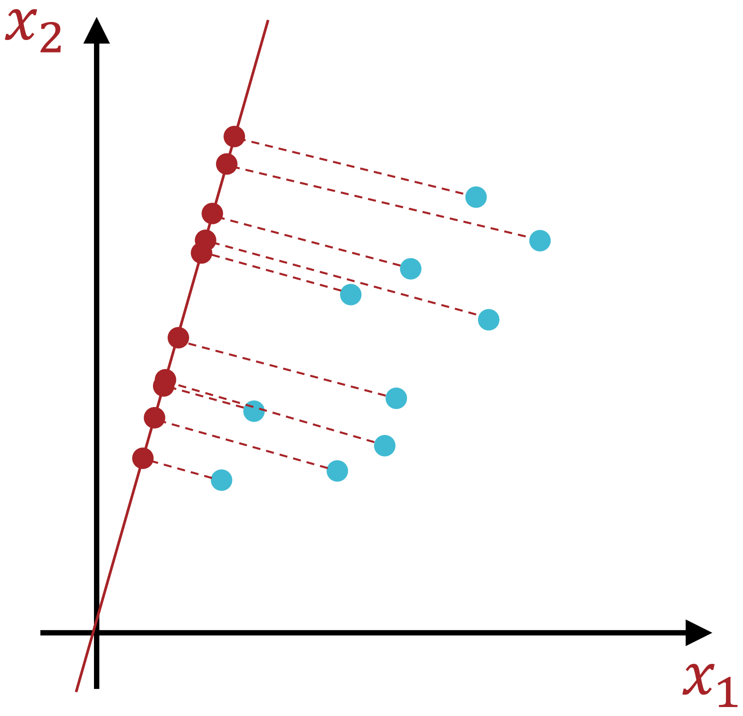


Feature Elimination

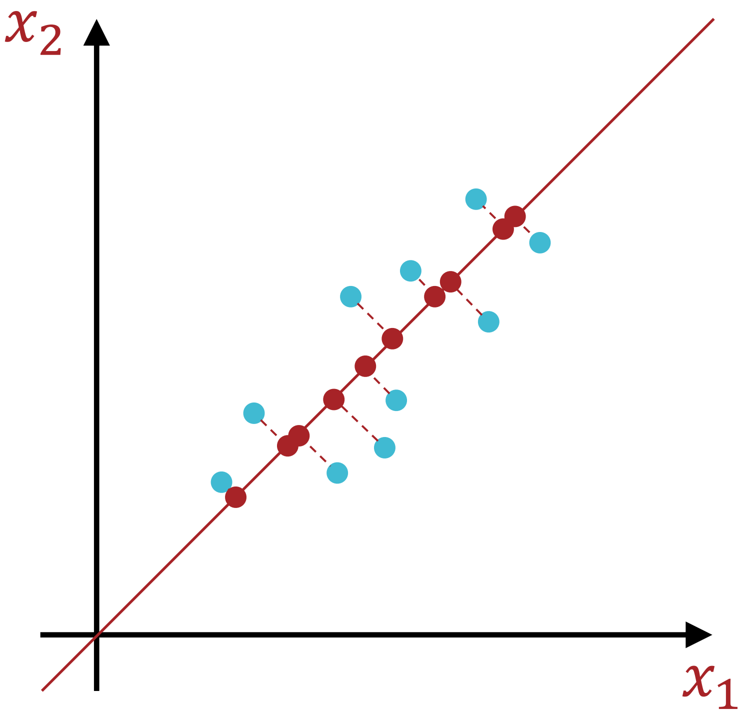




Feature Reduction



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

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)

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

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

Maximizing the Variance

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

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

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

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

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

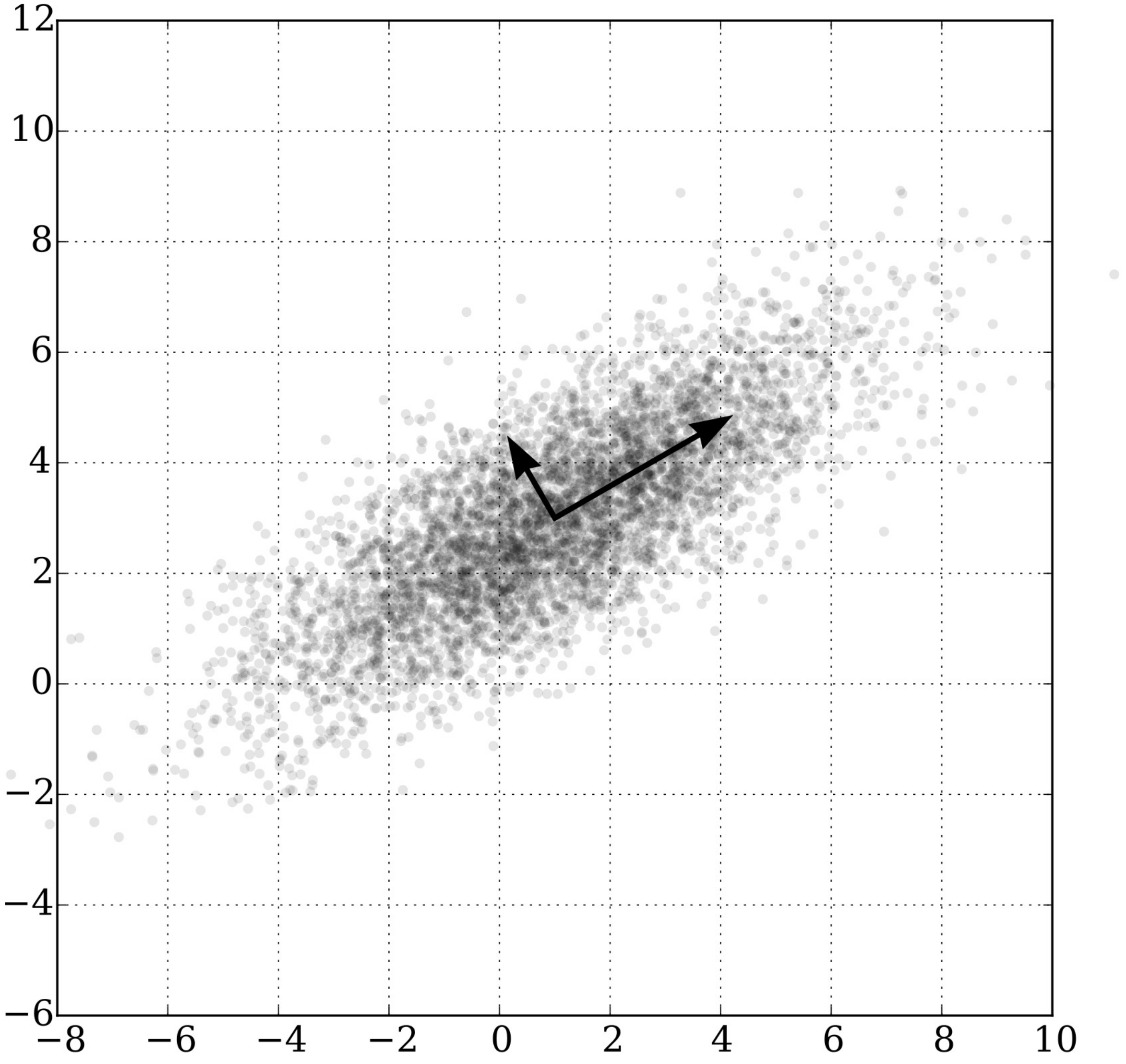
Maximizing the Variance

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

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

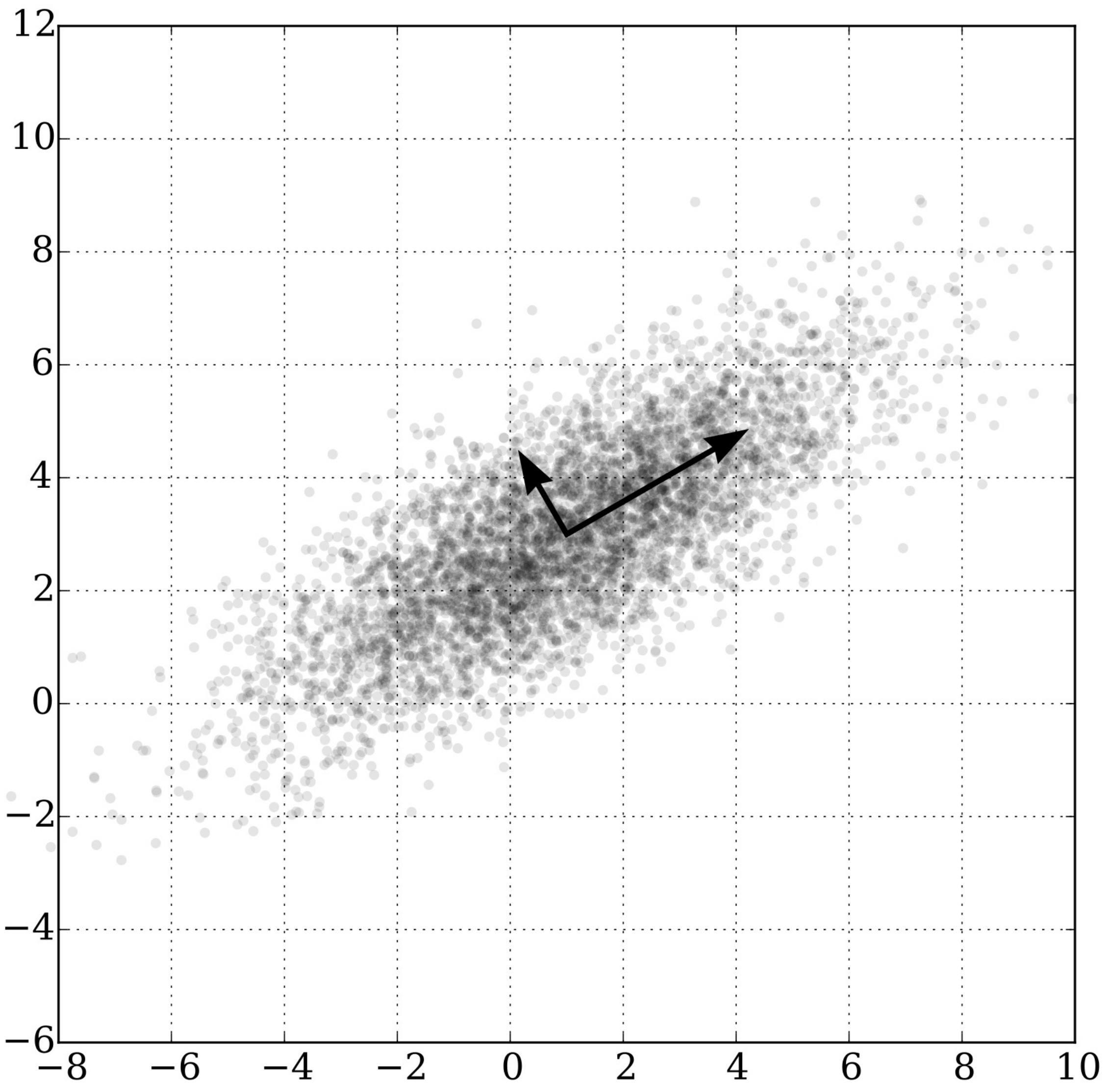
- The first principal component is the eigenvector $\hat{\boldsymbol{v}}_1$ that corresponds to the largest eigenvalue λ_1
- The second principal component is the eigenvector $\hat{\boldsymbol{v}}_2$ that corresponds to the second largest eigenvalue λ_2
  - $\hat{\boldsymbol{v}}_1$ and $\hat{\boldsymbol{v}}_2$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

PCA Algorithm

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

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

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

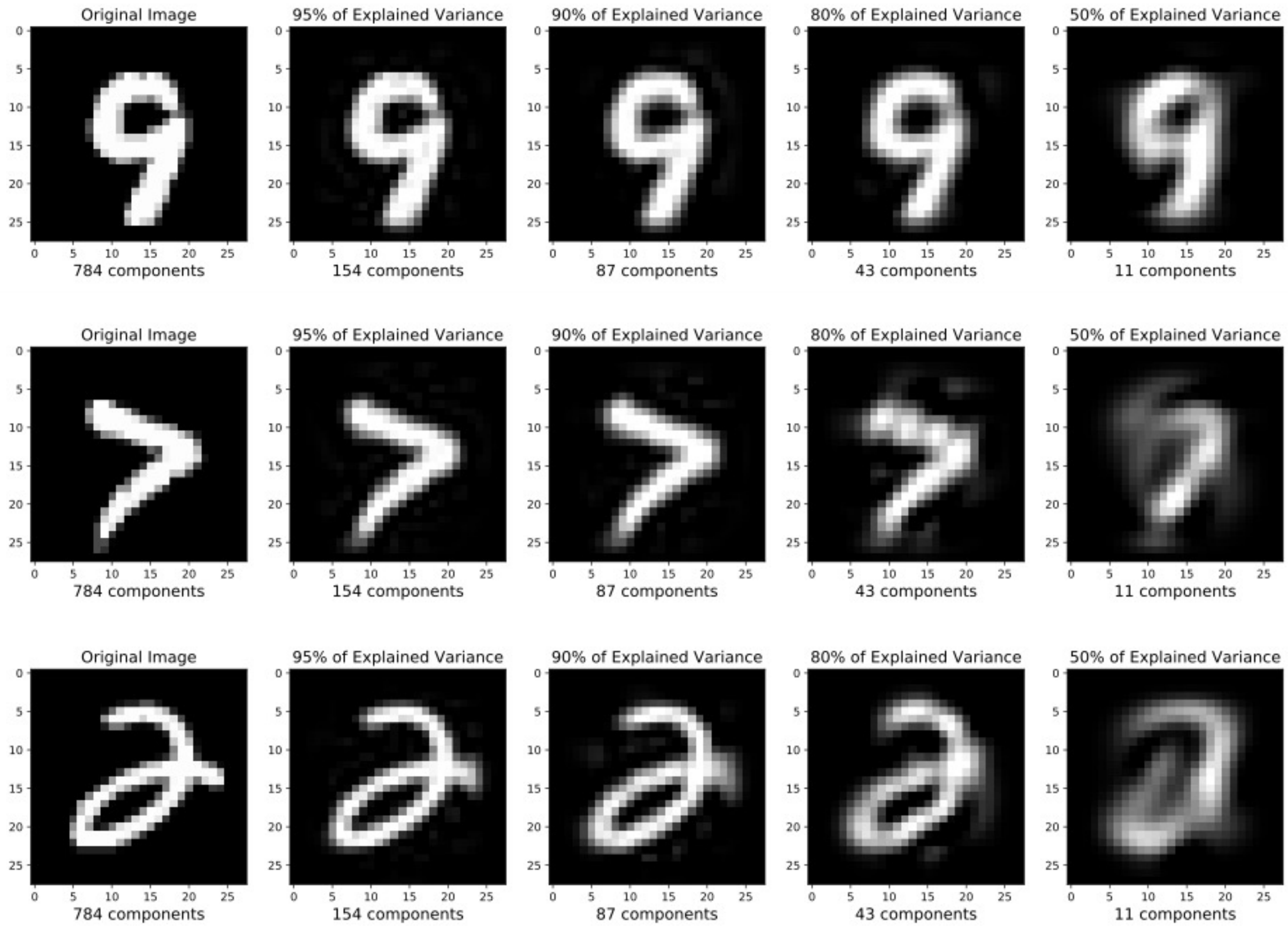
Choosing the number of PCs

- 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

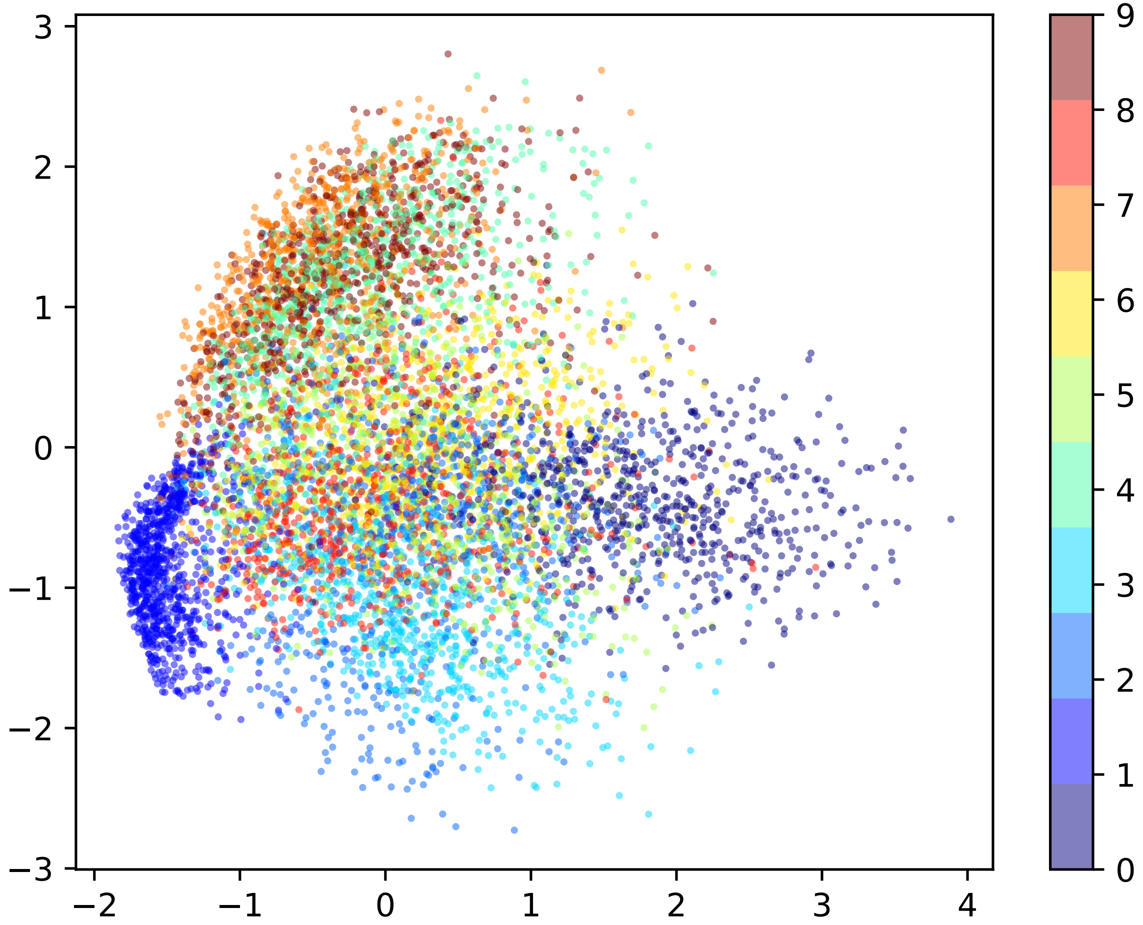


Figure courtesy of Matt Gormley

PCA Example: MNIST Digits

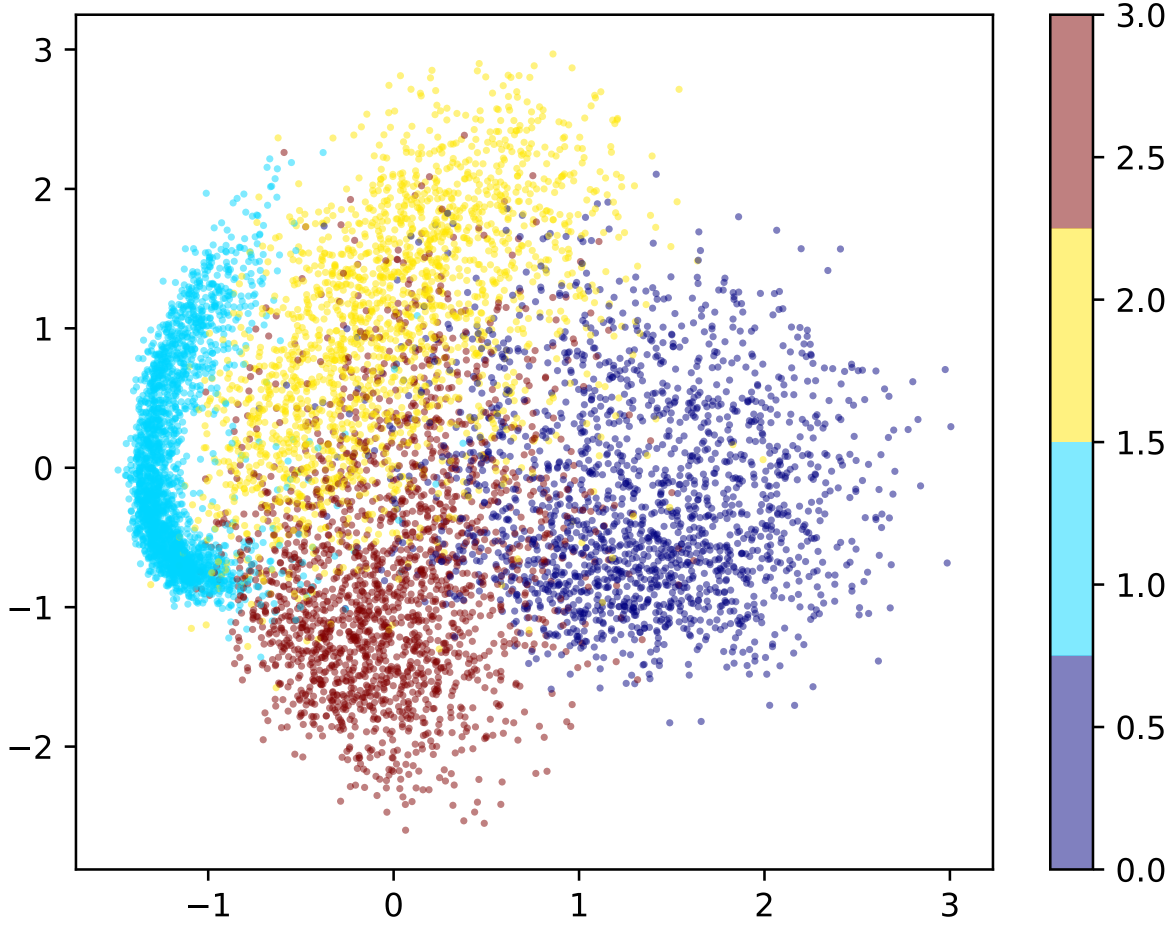


Figure courtesy of Matt Gormley

Key Takeaways

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