CSC 411: Lecture 14: Principal Components Analysis & Autoencoders

Richard Zemel, Raquel Urtasun and Sanja Fidler

University of Toronto

Today

- Dimensionality Reduction
- PCA
- Autoencoders

Mixture models and Distributed Representations

- One problem with mixture models: each observation assumed to come from one of K prototypes

Mixture models and Distributed Representations

- One problem with mixture models: each observation assumed to come from one of K prototypes
- Constraint that only one active (responsibilities sum to one) limits the representational power

Mixture models and Distributed Representations

- One problem with mixture models: each observation assumed to come from one of K prototypes
- Constraint that only one active (responsibilities sum to one) limits the representational power
- Alternative: Distributed representation, with several latent variables relevant to each observation

Mixture models and Distributed Representations

- One problem with mixture models: each observation assumed to come from one of K prototypes
- Constraint that only one active (responsibilities sum to one) limits the representational power
- Alternative: Distributed representation, with several latent variables relevant to each observation
- Can be several binary/discrete variables, or continuous

Example: Continuous Underlying Variables

- What are the intrinsic latent dimensions in these two datasets?

Example: Continuous Underlying Variables

- What are the intrinsic latent dimensions in these two datasets?

Example: Continuous Underlying Variables

- What are the intrinsic latent dimensions in these two datasets?

- How can we find these dimensions from the data?

Principal Components Analysis

- PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods

Principal Components Analysis

- PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods
- Aim: find a small number of "directions" in input space that explain variation in input data; re-represent data by projecting along those directions

Principal Components Analysis

- PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods
- Aim: find a small number of "directions" in input space that explain variation in input data; re-represent data by projecting along those directions
- Important assumption: variation contains information

Principal Components Analysis

- PCA: most popular instance of second main class of unsupervised learning methods, projection methods, aka dimensionality-reduction methods
- Aim: find a small number of "directions" in input space that explain variation in input data; re-represent data by projecting along those directions
- Important assumption: variation contains information
- Data is assumed to be continuous:
  - linear relationship between data and the learned representation

PCA: Common Tool

- Handles high-dimensional data

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing
  - Modeling – prior for new data

PCA: Common Tool

- Handles high-dimensional data
  - If data has thousands of dimensions, can be difficult for a classifier to deal with
- Often can be described by much lower dimensional representation
- Useful for:
  - Visualization
  - Preprocessing
  - Modeling – prior for new data
  - Compression

PCA: Intuition

- As in the previous lecture, training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$

PCA: Intuition

- As in the previous lecture, training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$
- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $M << D$:

$$\mathbf{x} \approx U\mathbf{z} + \mathbf{a}$$

    where U is a $D \times M$ matrix and $\mathbf{z}$ a M-dimensional vector

PCA: Intuition

- As in the previous lecture, training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$
- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $M << D$:

$$\mathbf{x} \approx U\mathbf{z} + \mathbf{a}$$

    where U is a $D \times M$ matrix and $\mathbf{z}$ a M-dimensional vector
- Search for orthogonal directions in space with the highest variance
  - project data onto this subspace

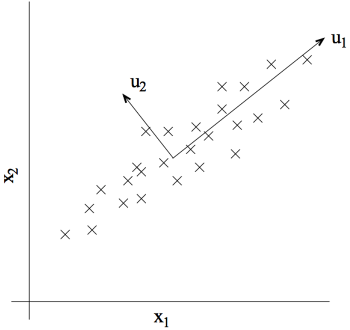
PCA: Intuition

- As in the previous lecture, training data has N vectors, $\{\mathbf{x}_n\}_{n=1}^N$, of dimensionality D, so $\mathbf{x}_i \in \mathbb{R}^D$
- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $M << D$:

$$\mathbf{x} \approx U\mathbf{z} + \mathbf{a}$$

    where U is a $D \times M$ matrix and $\mathbf{z}$ a M-dimensional vector

- Search for orthogonal directions in space with the highest variance
  - project data onto this subspace
- Structure of data vectors is encoded in sample covariance



Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)

Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)
- Calculate the empirical covariance matrix:

$$C = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

with $\bar{\mathbf{x}}$ the mean

Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)
- Calculate the empirical covariance matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

  with $\bar{\mathbf{x}}$ the mean

- What's the dimensionality of C?

Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)
- Calculate the empirical covariance matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

  with $\bar{\mathbf{x}}$ the mean

- What's the dimensionality of C?
- Find the M eigenvectors with largest eigenvalues of C: these are the principal components

Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)
- Calculate the empirical covariance matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

  with $\bar{\mathbf{x}}$ the mean
- What's the dimensionality of C?
- Find the M eigenvectors with largest eigenvalues of C: these are the principal components
- Assemble these eigenvectors into a $D \times M$ matrix U

Finding Principal Components

- To find the principal component directions, we center the data (subtract the sample mean from each variable)
- Calculate the empirical covariance matrix:

$$C = \frac{1}{N} \sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T$$

  with $\bar{\mathbf{x}}$ the mean
- What's the dimensionality of C?
- Find the M eigenvectors with largest eigenvalues of C: these are the principal components
- Assemble these eigenvectors into a $D \times M$ matrix U
- We can now express D-dimensional vectors $\mathbf{x}$ by projecting them to M-dimensional $\mathbf{z}$

$$\mathbf{z} = U^T \mathbf{x}$$

Standard PCA

- Algorithm: to find M components underlying D-dimensional data

Standard PCA

- Algorithm: to find M components underlying D-dimensional data
  1. Select the top M eigenvectors of C (data covariance matrix):

$$C = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T = U\Sigma U^T \approx U_{1:M}\Sigma_{1:M}U_{1:M}^T$$

     where U is orthogonal, columns are unit-length eigenvectors

$$U^T U = UU^T = 1$$

     and Σ is a matrix with eigenvalues on the diagonal, representing the variance in the direction of each eigenvector

Standard PCA

- Algorithm: to find M components underlying D-dimensional data
  1. Select the top M eigenvectors of C (data covariance matrix):

$$C = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^T = U\Sigma U^T \approx U_{1:M}\Sigma_{1:M}U_{1:M}^T$$

     where U is orthogonal, columns are unit-length eigenvectors

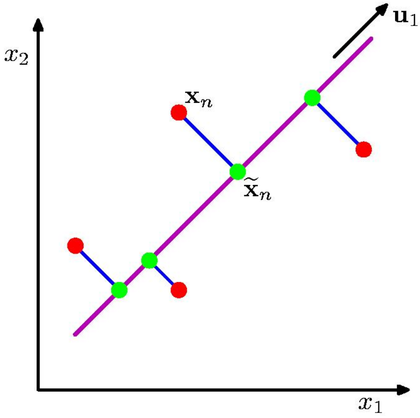
$$U^T U = UU^T = 1$$

     and Σ is a matrix with eigenvalues on the diagonal, representing the variance in the direction of each eigenvector
  2. Project each input vector $\mathbf{x}$ into this subspace, e.g.,

$$z_j = \mathbf{u}_j^T\mathbf{x}; \qquad \mathbf{z} = U_{1:M}^T\mathbf{x}$$

Two Derivations of PCA

- Two views/derivations:
  - Maximize variance (scatter of green points)
  - Minimize error (red-green distance per datapoint)



PCA: Minimizing Reconstruction Error

- We can think of PCA as projecting the data onto a lower-dimensional subspace

PCA: Minimizing Reconstruction Error

- We can think of PCA as projecting the data onto a lower-dimensional subspace
- One derivation is that we want to find the projection such that the best linear reconstruction of the data is as close as possible to the original data

$$J(\mathbf{u}, \mathbf{z}, \mathbf{b}) = \sum_n ||\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}||^2$$

where

$$\tilde{\mathbf{x}}^{(n)} = \sum_{j=1}^{M} z_j^{(n)} \mathbf{u}_j + \sum_{j=M+1}^{D} b_j \mathbf{u}_j$$

PCA: Minimizing Reconstruction Error

- We can think of PCA as projecting the data onto a lower-dimensional subspace
- One derivation is that we want to find the projection such that the best linear reconstruction of the data is as close as possible to the original data

$$J(\mathbf{u}, \mathbf{z}, \mathbf{b}) = \sum_n ||\mathbf{x}^{(n)} - \tilde{\mathbf{x}}^{(n)}||^2$$

where

$$\tilde{\mathbf{x}}^{(n)} = \sum_{j=1}^{M} z_j^{(n)} \mathbf{u}_j + \sum_{j=M+1}^{D} b_j \mathbf{u}_j$$

- Objective minimized when first M components are the eigenvectors with the maximal eigenvalues

$$z_j^{(n)} = \mathbf{u}_j^T \mathbf{x}^{(n)}; \quad b_j = \bar{\mathbf{x}}^T \mathbf{u}_j$$

Applying PCA to faces

- Run PCA on 2429 19x19 grayscale images (CBCL data)

Applying PCA to faces

- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components

Applying PCA to faces

- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components

- PCA for pre-processing: can apply classifier to latent representation
  - PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for GMM with 84 states

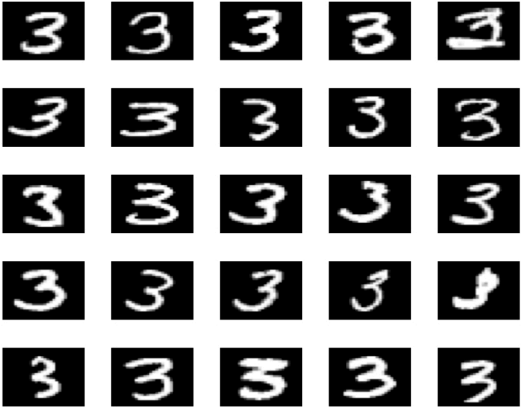
Applying PCA to faces

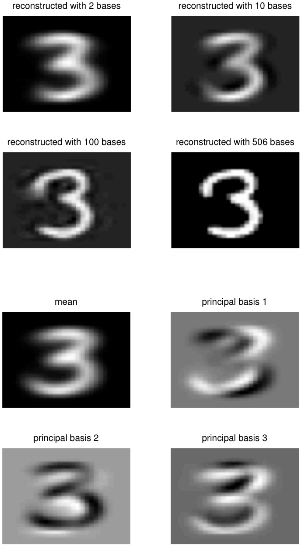
- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components

- PCA for pre-processing: can apply classifier to latent representation
  - PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for GMM with 84 states
- Can also be good for visualization

Applying PCA to faces: Learned basis

Applying PCA to digits



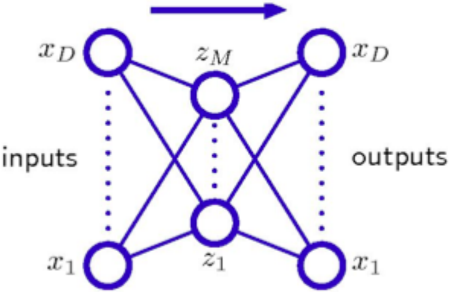


Relation to Neural Networks

- PCA is closely related to a particular form of neural network

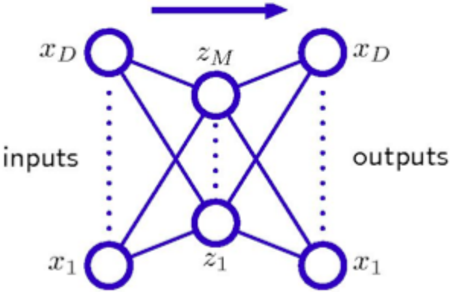
Relation to Neural Networks

- PCA is closely related to a particular form of neural network
- An autoencoder is a neural network whose outputs are its own inputs



Relation to Neural Networks

- PCA is closely related to a particular form of neural network
- An autoencoder is a neural network whose outputs are its own inputs



- The goal is to minimize reconstruction error

Autoencoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

Autoencoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2$$

Autoencoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2$$

- If g and f are linear

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - VW\mathbf{x}^{(n)}||^2$$

Autoencoders

- Define

$$\mathbf{z} = f(W\mathbf{x}); \quad \hat{\mathbf{x}} = g(V\mathbf{z})$$

- Goal:

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - \hat{\mathbf{x}}^{(n)}||^2$$

- If g and f are linear

$$\min_{\mathbf{W},\mathbf{V}} \frac{1}{2N} \sum_{n=1}^{N} ||\mathbf{x}^{(n)} - VW\mathbf{x}^{(n)}||^2$$

- In other words, the optimal solution is PCA.

Autoencoders: Nonlinear PCA

- What if $g()$ is not linear?

Autoencoders: Nonlinear PCA

- What if $g()$ is not linear?
- Then we are basically doing nonlinear PCA

Autoencoders: Nonlinear PCA

- What if $g()$ is not linear?
- Then we are basically doing nonlinear PCA
- Some subtleties but in general this is an accurate description

Comparing Reconstructions