CSC 411 Lecture 12:Principle Components Analysis

Ethan Fetaya, James Lucas and Emad Andrews

University of Toronto

Today

- Unsupervised learning
- Dimensionality Reduction
- PCA

Unsupervised Learning

- Supervised learning algorithms have a clear goal: produce desired outputs for given inputs.
  - You are given $\{(x^{(i)}, t^{(i)})\}$ during training (inputs and targets)
- Goal of unsupervised learning algorithms less clear.
  - You are given the inputs $\{x^{(i)}\}$ during training, labels are unknown.
  - No explicit feedback whether outputs of system are correct.
- Tasks to consider:
  - Reduce dimensionality
  - Find clusters
  - Model data density
  - Find hidden causes
- Key utility
  - Compress data
  - Detect outliers
  - Facilitate other learning

Major Types

- Primary problems, approaches in unsupervised learning fall into three classes:

  1. Dimensionality reduction: represent each input case using a small number of variables (e.g., principal components analysis, factor analysis, independent components analysis)
  2. Clustering: represent each input case using a prototype example (e.g., k-means, mixture models)
  3. Density estimation: estimating the probability distribution over the data space

- Sometimes the main challenge is to define the right task.
- Today we will talk about a dimensionality reduction algorithm

Example

- 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 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
  - Can reduces overfitting
  - Can speed up computation and reduce memory usage.
- Unsupervised algorithm.
- Useful for:
  - Visualization
  - Preprocessing
  - Better generalization
  - Lossy compression

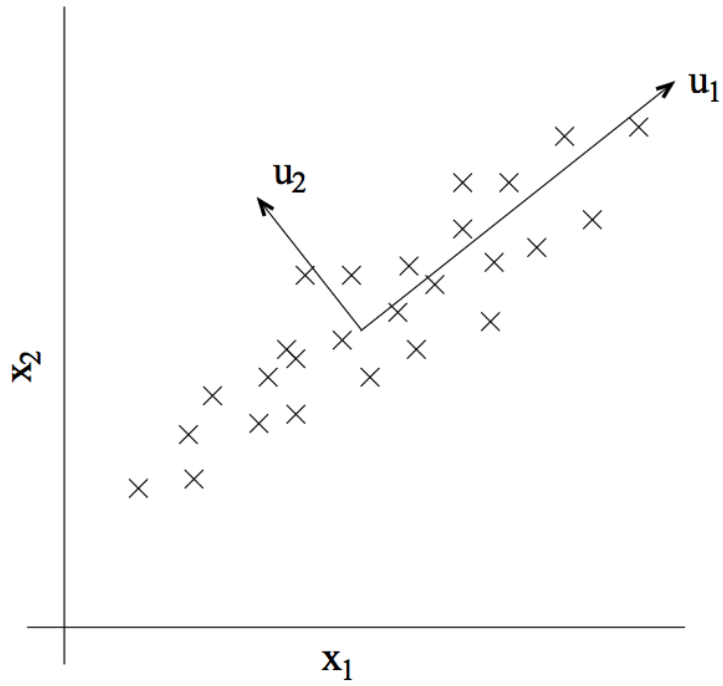
PCA: Intuition

- Aim to reduce dimensionality:
  - linearly project to a much lower dimensional space, $K << D$:

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

where U is a $D \times K$ matrix and $\mathbf{z}$ a K-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



Single dimension

- To find the principal component directions, we center the data (subtract the sample mean from each feature)
- Calculate the empirical covariance matrix: $\Sigma = \frac{1}{N} X^T X$ (some people divide by 1/(N-1))
- Look for a direction $\mathbf{w}$ that maximizes the projection variance $y^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)}$
  - Normalize $||\mathbf{w}|| = 1$ or you can just increase the variance to infinity.
- What is the variance of the projection?

$$Var(y) = \sum_j \frac{1}{N} (\mathbf{w}^T \mathbf{x}^{(i)})^2 = \frac{1}{N} \sum_i \mathbf{w}_i^T x^{(i)} \mathbf{x}^{(i)T} \mathbf{w} = \mathbf{w}^T \Sigma \mathbf{w}$$

- Our goal is to solve:

$$\mathbf{w}^* = \arg \max_{||\mathbf{w}||=1} \mathbf{w}^T \Sigma \mathbf{w}$$

Eigenvectors

Target: find $\mathbf{w}^* = \arg\max_{||\mathbf{w}||=1} \mathbf{w}^T \Sigma \mathbf{w}$

- Σ has an eigen-decomposition with orthonormal $\mathbf{v}_1, ..., \mathbf{v}_d$ and eigenvalues $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d \geq 0$
- Write $\mathbf{w}$ in that bases

$$\mathbf{w} = \sum_i a_i \mathbf{v}_i, \sum_i a_i^2 = 1$$

- The objective is now $\arg\max_{\sum_i a_i^2 = 1} a_i^2 \lambda_i$
- Simple solution! Put all weights in the larget eigenvalue! $\mathbf{w} = \mathbf{v}_1$
- What about reduction to dimension 2?
  - Second vector has another constrain - orthogonal to the first.
  - Optimal solution - second largest eigenvector.
- The best k dimensional subspace (max variance) is spanned by the top-k eigenvectors.

Eigenvectors

Another way to see it:

- Σ has an eigen-decomposition $\Sigma = U\Lambda U^T$
  - where U is orthogonal, columns are unit-length eigenvectors

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

and Λ is a diagonal matrix of eigenvalues in decreasing magnitude.

- What would happen if we take $z^{(i)} = U^T x^{(i)}$ as our features?
- $\Sigma_Z = U^T \Sigma_X U = \Lambda$
  - The dimension of z are uncorrelated!
- How can we maximize variance now? Just take the top k features, i.e. first k eigenvectors.

Algorithm

- Algorithm: to find K components underlying D-dimensional data
  1. Compute the mean for each feature $m_i = \frac{1}{N}\sum_j \mathbf{x}_i^{(j)}$.
  2. Select the top M eigenvectors of C (data covariance matrix):

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}^{(n)} - \mathbf{m})(\mathbf{x}^{(n)} - \mathbf{m})^T = U\Lambda U^T \approx U_{1:K}\,\Lambda_{1:K}\,U_{1:K}^T$$

  3. Project each input vector $\mathbf{x} - \mathbf{m}$ into this subspace, e.g.,

$$z_j = \mathbf{u}_j^T(\mathbf{x} - \mathbf{m}); \qquad \mathbf{z} = U_{1:K}^T(\mathbf{x} - \mathbf{m})$$

  4. How can we (approximately) reconstruct the original $\mathbf{x}$ if we want to?
     - $\tilde{\mathbf{x}} = U_{1:K}\mathbf{z} + \mathbf{m} = U_{1:K}\,U_{1:K}^T\mathbf{x} + \mathbf{m}$

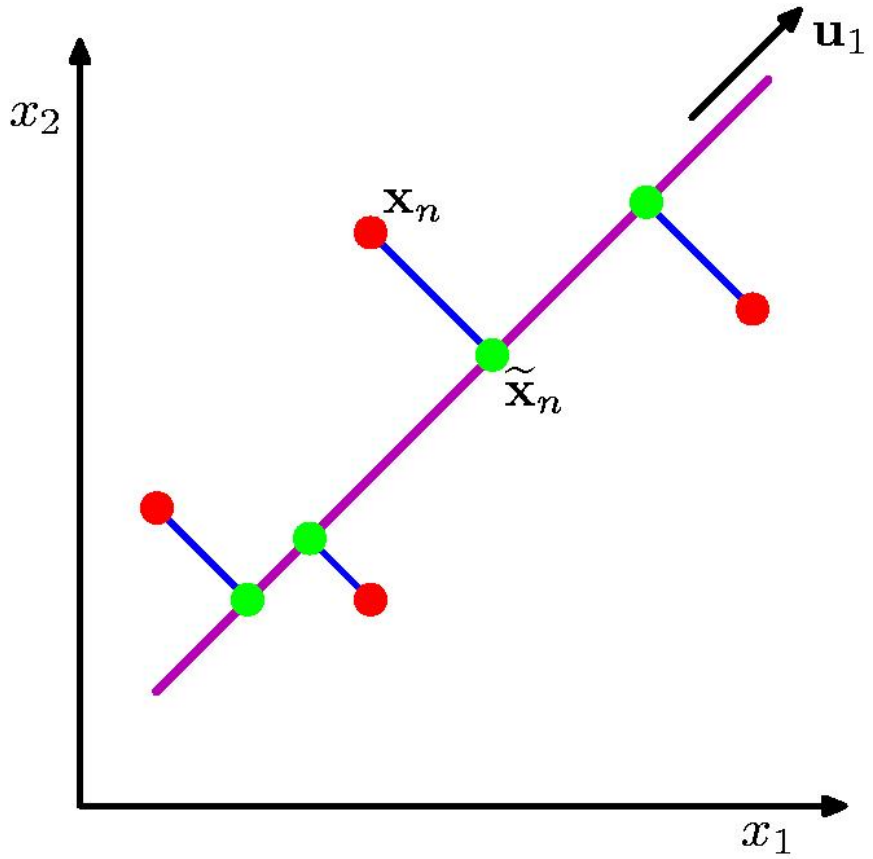
Choosing K

We have the hyper-parameter K, how do we set it?

- Visualization: k=2 (maybe 3)
- If it is part of classification/regression pipeline - validation/cross-validation.
- Common approach: Pick based on the percentage of variance explained by each of the selected components.
  - Total variance $\sum_{j=1}^{d} \lambda_j = \mathit{Trace}(\Sigma)$
  - Variance explained $\sum_{j=1}^{k} \lambda_j$
  - Pick smallest k such that $\sum_{j=1}^{k} \lambda_j > \alpha \mathit{Trace}(\Sigma)$ for some value α e.g. 0.9
- Based on memory/speed constraints.

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
- Another 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

$$\tilde{\mathbf{x}}^{(n)} = \sum_{j=1}^{K} z_j^{(n)} \mathbf{u}_j + \mathbf{m} \qquad z_j^{(n)} = \mathbf{u}_j^T (\mathbf{x}^{(n)} - \mathbf{m})$$

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

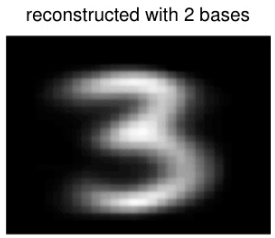
Applying PCA to faces

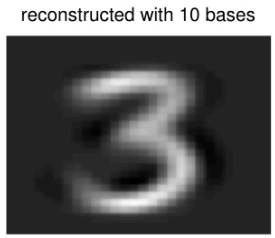
- Run PCA on 2429 19x19 grayscale images (CBCL data)
- Compresses the data: can get good reconstructions with only 3 components

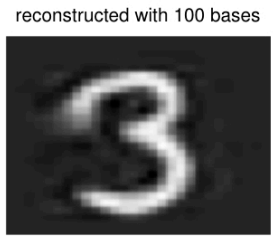
- PCA for pre-processing: can apply classifier to latent representation
  - PCA with 3 components obtains 79% accuracy on face/non-face discrimination on test data vs. 76.8% for GMM with 84 states
- Can also be good for visualization

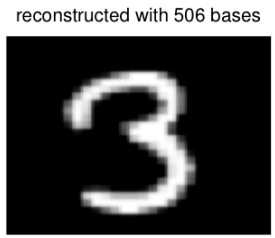
Applying PCA to faces: Learned basis

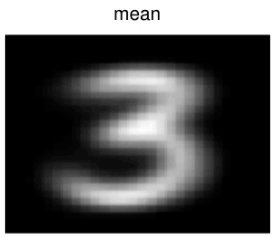
Applying PCA to digits

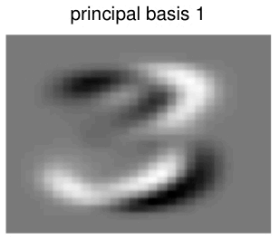


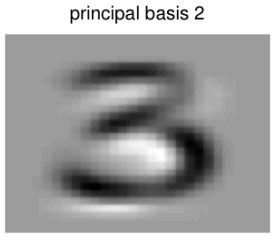


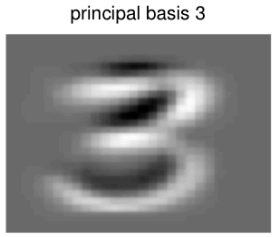






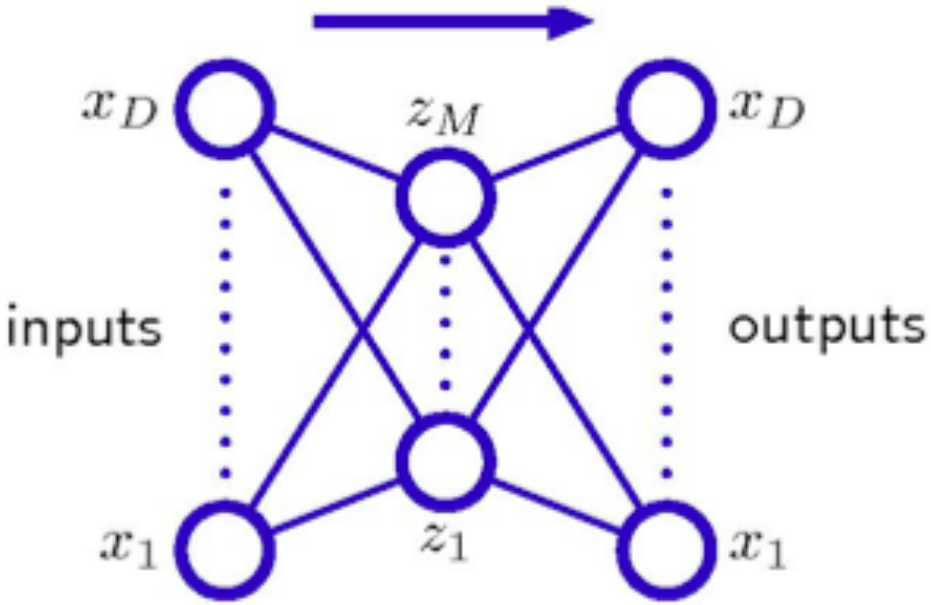






Relation to Neural Networks

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



- The goal is to minimize reconstruction error

Implementation details

What is the time complexity of PCA?

- Main computation - generating Σ matrix $\mathcal{O}(dn^2)$ and computing eigendecomposition $\mathcal{O}(d^3)$
- For $d \gg n$ can use a trick - compute eigenvalues of $\frac{1}{N}XX^T$ instead $\Sigma = \frac{1}{N}X^TX$ (how is that helpful?). Complexity is $\mathcal{O}(d^2n + n^3)$
- Don't need full eigendecomposition - only top-k! (much) faster solvers for that.
- Common approach nowadays - solve using SVD (runtime of $\mathcal{O}(mdk)$)
  - More numerically accurate

Singular value decomposition

What is singular value decomposition (SVD)?

- Decompose X, $X = V\Lambda U^T$ with orthogonal U, V and diagonal with positive elements Λ.
  - Holds for every matrix unlike eigen-decomposition.
- How do they connect to the eigenvectors of $X^T X$?

$$X^T X = (V\Lambda U^T)^T (V\Lambda U^T) = U\Lambda V^T V\Lambda U^T = U\Lambda^2 U^T$$

- The column of U are the eigenvectors of $X^T X$.
  - The corresponding eigenvalue is the square of the singular value.
- Finding the top k singular values of X is equivalent to finding the top k eigenvectors of $X^T X$.

Recap

- PCA is the standard approach for dimensionality reduction
- Main assumptions: Linear structure, high variance = important
- Helps reduce overfitting, curse of dimensionality and runtime.
- Simple closed form solution
  - Can be expensive on huge datasets
- Can be bad on non-linear structure
  - Can be handled by extensions like kernel-PCA
- Bad at fined-grained classification - we can easily throw away important information.