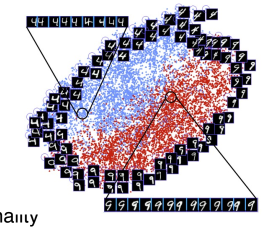
Dimensionality reduction

The slides are closely adapted from Subhransu Maji's slides

Motivation

- Data visualization
  - Hard to visualize data that lives in high dimensions — reduce it two two or three dimensions for visualization



[Hadsell et al, CVPR 06]

- Curse of dimensionality
  - Some learning methods don't scale well with the number of features (e.g., kNN, kernel density estimators)
  - Lower memory overhead and training/testing time
  - Fewer dimensions is a form of regularization

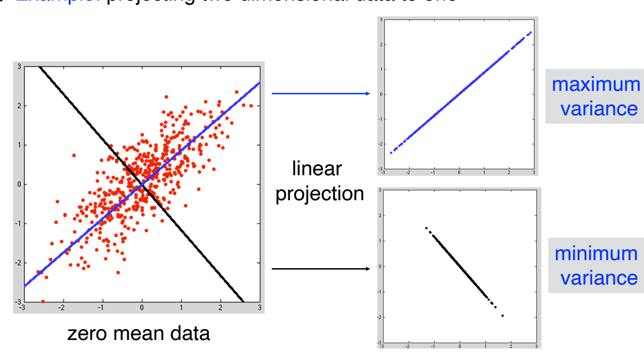
Dimensionality reduction

- The goal is to reduce the dimension of the data in high-dimensions (say 10000) to low dimensions (say 2) while retaining the "important" characteristics of the data
- Unsupervised setting, so the notion of important characteristics is hard to define
- Closely related to clustering
  - Clustering: reduce the number of data
  - Dimensionality reduction: reduce the number of features

$$\mathbf{x}_i \in R^D, i = 1, 2, \ldots, N$$
$$\text{data matrix} = R^{N \times D}$$

Linear dimensionality reduction

- All you can do is project the data onto a vector and use the projected distances as the embeddings
- Example: projecting two dimensional data to one



Optimal linear projection

- Find a linear projection that maximizes the variance of the projection
- Assume we have data $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \in R^D$ of zero mean
- Let $\mathbf{u}$ be the projection vector
- Let the projections of the data $p_1, p_2, \ldots, p_N$

$$p_i \leftarrow \mathbf{x}_i^T\mathbf{u}$$

- The mean of the projections is zero

$$\sum_i p_i = \sum_i \mathbf{x_i}^T\mathbf{u} = \left(\sum_i \mathbf{x_i}\right)^T \mathbf{u} = 0$$

- Maximize the variance of the projection:

$$\max_{\mathbf{u}} \sum_i \left(\mathbf{x}_i^T\mathbf{u}\right)^2 \text{ subject to: } ||u|| = 1$$

Optimal linear projection

- Lets rewrite this in matrix notation
- Let $\mathbf{X}$ be the NxD data matrix (each row is data point)
- The projection vector $\mathbf{u}$ is a Dx1 matrix
- The vector of projections is given by $\mathbf{Xu}$, a Nx1 matrix
- We can rewrite the optimization as:

$$\max_{\mathbf{u}} ||\mathbf{Xu}||^2 \text{ subject to: } \mathbf{u}^T\mathbf{u} = 1$$

- The corresponding Lagrangian is:

$$\mathcal{L}(\mathbf{u}, \lambda) = ||\mathbf{Xu}||^2 - \lambda(\mathbf{u}^T\mathbf{u} - 1)$$

- At maxima:

$$\Delta_u = 2\mathbf{X}^T\mathbf{X}\mathbf{u} - 2\lambda\mathbf{u}$$

$$\implies \left(\mathbf{X}^T\mathbf{X}\right)\mathbf{u} = \lambda\mathbf{u} \longleftarrow \text{eigenvalue problem}$$

Optimal linear projection

- Compute the data covariance matrix $\mathbf{X}^T\mathbf{X}$

$$\left[\mathbf{X}^T\mathbf{X}\right]_{ij} = \sum_n \mathbf{x}_{ni}\mathbf{x}_{nj}$$

  - The optimal (maximal variance) projection direction is the first eigenvector of the data covariance matrix

$$\left(\mathbf{X}^T\mathbf{X}\right)\mathbf{u} = \lambda\mathbf{u}$$

- What about learning a second projection direction?
  - For non-redundancy additionally require that $\mathbf{v}^T\mathbf{u} = 0$

$$\max_{\mathbf{v}} ||\mathbf{Xv}||^2 \text{ subject to: } \mathbf{v}^T\mathbf{v} = 1, \mathbf{v}^T\mathbf{u} = 0$$

  - This is the second eigenvector of the data covariance matrix

Optimal linear projection

1. First find the first component u
2. Reconstruct the data using it Xu
3. Find the residual $X' = X - Xu(\frac{u}{||u||})^T$
4. Find the next component using X'
   - Since $X'u = 0$ the new u will be orthogonal to the previous one.
   - Basically, we are removing all signals along the same component

Principal component analysis (PCA)

Find U and Λ such that

$$X^T X = U \Lambda U^T$$

where $U^T U = I$ and Λ is a diagonal matrix.

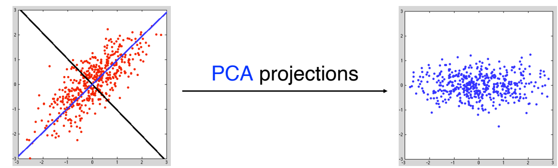
Columns of U are eigen vectors and Λ is a diagonal matrix of λ's.

Xu is the emdebbing of X in the direction of u.

Principal component analysis (PCA)

- Generalizing this argument leads to principal component analysis
- The eigenvectors give you the projection directions — to compute the embeddings you have to multiply the data by the projections
- For completeness here is the Matlab code:

```
function [E, U, lambda] = PCA(X, K)
mu = mean(X);                   % Data mean
N = size(X,1);                  % Number of data
X = X - ones(N,1)*mu;           % Center the data
covX = X'*X;                    % Data covaraince
[U,lambda] = eigs(covX, K);     % Compute top K eigenvalues
E = X*U;                        % Compute embeddings
```

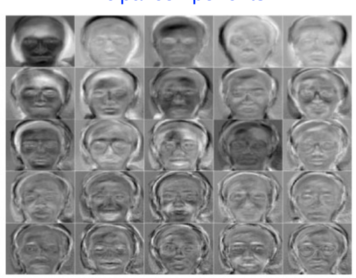


Application: Eigenfaces

- Eigenfaces — a linear basis for face images [Turk, Pentland '91]
- Each face is a weighted linear combination of eigenfaces
- Compare faces by comparing the weights

Input images

Principal components

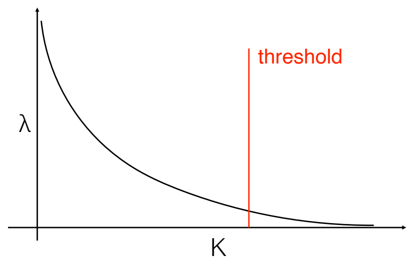


- Reconstructing face image using few components
- We need more components to see the details.

Image from: http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html

What should K be?

- For visualization K = 2 or 3
- For dimensionality reduction it depends on the problem
  - Option: ignore projections that correspond to small eigenvalues
  - Option: based on computational and memory constraints



PCA

Find U and Λ such that

$$X^T X = U \Lambda U^T$$

where $U^T U = I$ and Λ is a diagonal matrix.
Columns of U are eigen vectors and Λ is a diagonal matrix of λ's.

- What is the dimension of covariance matrix $X^T X$ if data is n-D?
  - What if n is large, say a 100x100 image?

- It is not easy to run PCA on such a large matrix

Singular Value Decomposition(SVD)

SVD problem: Find W, V, Σ such that

$$X = W \Sigma V^T$$

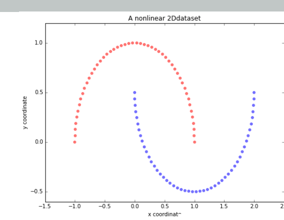
Where $W^T W = I$, $V^T V = I$ and Σ is a diagonal matrix.

We can write:

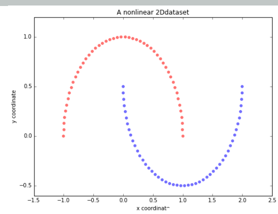
$$X^T X = (W \Sigma V^T)^T (W \Sigma V^T)$$
$$X^T X = (V \Sigma W^T)(W \Sigma V^T)$$
$$X^T X = V \Sigma^2 V^T$$

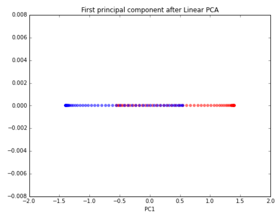
Compare it with PCA: $X^T X = U \Lambda U^T$
So eigen values are squares of elements in Σ
This is much faster and more stable compared to PCA for large dimensions.

Linear and kernel PCA

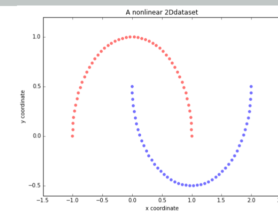


Linear and kernel PCA

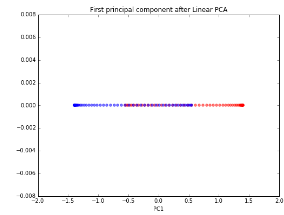


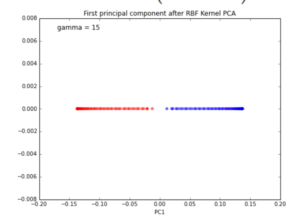


Linear and kernel PCA



$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp\left(-\gamma \, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \right)$$





Summary

- Dimensionality reduction for visualization or preprocessing
- Linear methods
  - PCA — linear projections of data

    solve $(\mathbf{X}^T\mathbf{X})x = \lambda x$ — eigenvectors of covariance matrix
- Non-linear methods
  - kernel PCA — linear projections in kernel space

    solve $\mathbf{K}x = \lambda x$ — eigenvectors of the kernel matrix

- There are several methods that we didn't discuss
  - Spectral clustering, ISOMAP, Locally linear embedding, tSNE, etc

Slides credit

- Most slides are adapted from Subhransu Maji's course and CIML book by Hal Daume III
- Linear and kernel PCA notes: http://pca.narod.ru/scholkopf_kernel.pdf
- The example for kernel PCA is from: http://sebastianraschka.com/Articles/2014_kernel_pca.html