Dimensionality Reduction

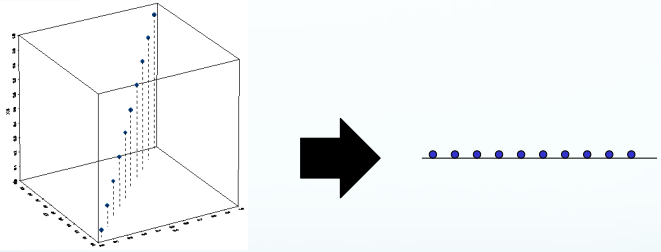
Mark Dredze

Machine Learning
CS 601.475

Data Representations

- In different learning settings **x** is a high dimensional vector
  - This can be problematic for some algorithms
  - Curse of dimensionality
    - K-NN
    - Clustering

Data Representations



Dimensionality Reduction

- Do we really need thousands or millions of dimensions to represent an example?
- We believe that examples can accurately be represented using fewer dimensions for learning
- This corresponds to the assumption that data points live close to a lower-dimensional manifold (subspace) of the data.
- **Dimensionality reduction**
  - Reduce high dimensional data to fewer dimensions

Benefits

- Easier learning
  - Fewer parameters to learn
  - Collapse multiple useful features into one useful feature
  - Remove unnecessary features
- Faster learning
- Learning without labels
  - Reduce dimensionality without labels
  - Uncovers latent structure in the data

Dimensionality Selection

- Belief: some of the features are not useful
- Approach: Reduce the number of dimensions by removing some of them
- **Feature Selection**
  - Select the features for learning that are helpful
    - Mutual information feature selection
    - Remove features to observe change in accuracy

Lower Dimensional Projections

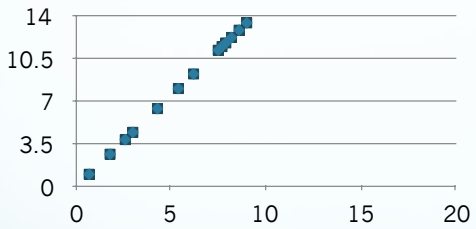
- Create new features that are combinations of existing features
- Belief: the given data can be accurately represented with fewer dimensions
- Approach: find the projection that gives the lowest recovery error

Linear Projections

- Examples given by $\{x_i\}_{i=1}^{N} \quad x_i \in \Re^D$
- Select a basis of M vectors: $\{u_m\}_{m=1}^{M} \quad u_m \in \Re^D$
  - Basis vectors are orthonormal (perpendicular and unit)
  - $u_i \cdot u_i = 1 \qquad u_i \cdot u_j = 0, \; i \neq j$
- Center of space $\bar{x}$ defines the offset
- Examples in low dimensional space

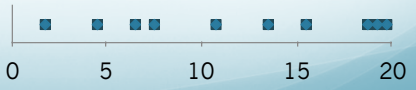
$$z_i = \left[z_i^1, z_i^2 \ldots z_i^M\right] \qquad z_i^m = (x_i - \bar{x}) \cdot u_m \qquad z_i = U(x_i - \bar{x})$$

Ideal Linear Projection

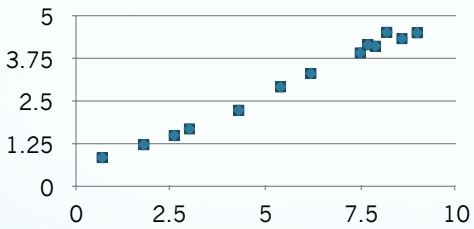


$y = 1.5x$

- Perfect reconstruction



Realistic Linear Projection



$y = 0.5x + \varepsilon$

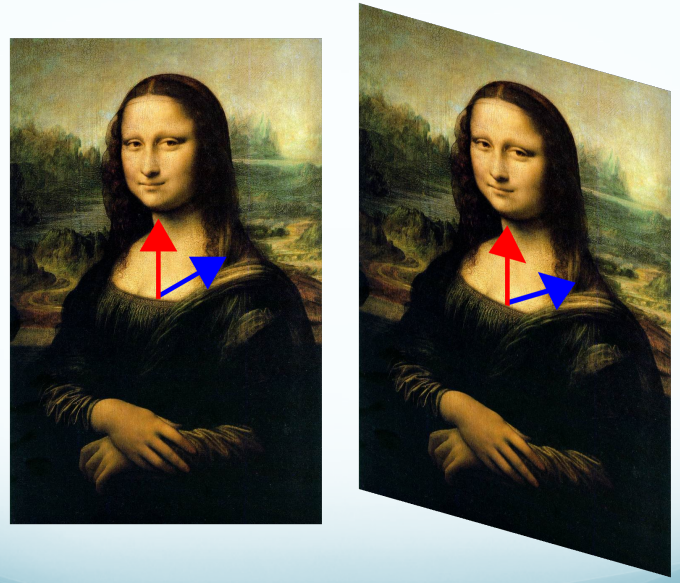
- ε error reconstruction

Learning Linear Projections

- Assume that data can be represented by linear projection
- Project D-dimensional data into M dimensions
  - Find the projection with the lowest reconstruction error
    - Preserve the dimensions with the most variance
- How to select M?
  - Tradeoff:
    - Larger M has more information
    - Smaller M is more efficient
  - Later: Bayesian methods for selecting M

Eigenvalues

- A matrix **W** transforms a vector **z** by changing magnitude and direction (**x**)
  - $\mathbf{Wz} = \mathbf{x}$
- The eigenvectors of a matrix **A** are changed only in magnitude
  - Eigenvalue- the factor by which the matrix is changed
  - Can also reverse direction (negative eigenvalues)
  - $\mathbf{A}u_i = \lambda_i u_i$
    - u are eigenvectors
    - λ are eigenvalues
    - D eigenvectors for D*D matrix



Example from Wikipedia

EigenFaces

- Compute the eigenvectors of a set of images of faces
- Each eigenvector can be plotted as an EigenFace

PCA

- Given an N*D matrix **X**
  - Each row of **X** is a D-dimensional point
- Re-center data by subtracting mean from each row
  - $\hat{X}_i = X_i - \bar{X}$
- Compute covariance matrix of data (D*D matrix)
  - $\Sigma = \frac{1}{N}\hat{\mathbf{X}}^T\hat{\mathbf{X}}$
- Find eigenvalues and eigenvectors of Σ
- Selected principal components
  - M eigenvectors with highest eigenvalues

Maximum Variance

- The eigenvector with the highest eigenvalue captures the dimension of highest data variance
  - This variance is preserved in the reduction
  - Removing small eigenvalues removes dimensions of smaller variances

PCA

- PCA selects the dimensions of the data that yield the maximum variance
  - The eigenvector with the largest eigenvalue gives the direction of maximum variance
  - We want the maximum variance dimensions so as to lose as little as possible in our reconstruction
- Example
  - http://www.igi.tugraz.at/lehre/CI/algorithms/pca_applet.html

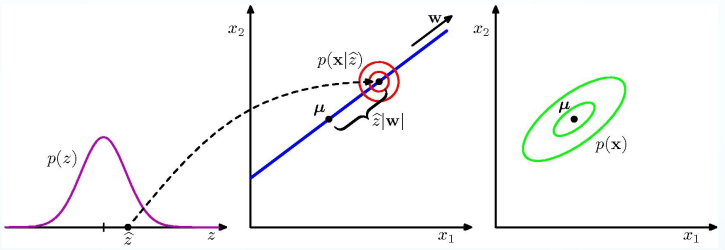
A Model for Dimensionality Reduction

- Let's define a model for dimensionality reduction using our intuitions
- Define a graphical model of how points are generated in high dimensional space
- Learn the model parameters from data
  - EM algorithm
  - Maximum Likelihood

Generative Story

- Generate a point z from distribution p(z) in low dimensional space (M)
- Generate a point x in high dimensional space (D) from a Gaussian with mean f(z) and covariance $\sigma^2\mathbf{I}$
  - p(z)- Gaussian (M dimensional)
  - f(z) = $\mathbf{W}z + \mu$
    - $\mathbf{W}$=D*M matrix, z=M*1 vector, μ=(D dimensional)
  - Assumes that high dimensional point has some Gaussian noise
- Find best hidden value z for observed x

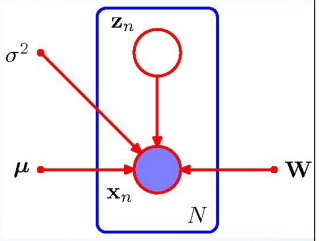
Generative Story



- Generate z from p(z)
- Generate x from Gaussian with mean $\mathbf{W}z + \mu$

Graphical Model

- For each of N high dimensional points
  - Generate z
  - Generate x given z, **W**, σ^2, μ



Graphical Model

- Let's define the model
  - Model likelihood
  - Model parameters
    - **W**, σ^2, μ
- Let's derive the model on the board

Notes on Board

Probabilistic PCA

- Same form as PCA (Principal Component Analysis)
  - Finds the M eigenvectors with the highest eigenvalues
  - We can easily find this using singular value decomposition
    - Python (numpy), Matlab and other packages

Data Reconstruction Error

- Distortion of the reconstructed points

$$J = \frac{1}{N}\sum_{i=1}^{N}\|x_i - \tilde{x}_i\|^2$$

- We can show this is equivalent to

$$J = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=M+1}^{D}\left\|x_i^T u_j - \bar{x}^T u_j\right\|^2$$

- Which is minimized when we select the D-M smallest eigenvectors to use in the distortion
  - This leave the M eigenvectors with largest eigenvalues for the principal subspace
  - This is the same as PCA

Advantages of PPCA

- Why probabilistic PCA?
  - Leads to a Bayesian PCA which finds the optimal value for M (number of dimensions)
  - Can be used to model class-conditional densities, leading to classification problems
  - Generative model that provides high dimensional samples

EM for PCA

- We can get all the benefits of EM as well
  - Handles missing data
  - Mixtures of PCA models
  - EM algorithm can be used to compute a few eigenvectors, much faster if the data is large
  - Same algorithm applied to **factor analysis** which doesn't have a closed form solution

Factor Analysis

- Each dimension has independent variance

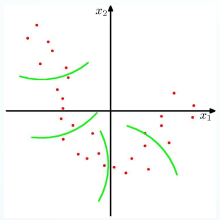
$$\Sigma = \sigma^2\mathbf{I} \longrightarrow \Sigma = \mathrm{diag}[\sigma_1^2, \sigma_2^2 \ldots \sigma_d^2]$$

- Learn parameters by maximum likelihood
  - No closed form solution for **W**
  - Iterative EM algorithm for solving **W**
- E step
  - Compute expectations for hidden low dimensional data points **z**
- M step
  - Update parameters **W** and **Σ**
    - μ still obtained with closed form solution

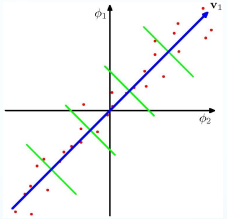
Kernel PCA

- We use the examples in the form of the covariance matrix **S**
- Replace **S** using only the examples, which allows us to express **C** using a kernel
- The kernel uses a non-linear transformation to project the data, we then use linear PCA on the resulting feature space

Kernel PCA



Non-linear data



Non-Linear projection with kernel

Summary: PCA

- Finds a low dimensional representation of the data using linear projections
- Many ways to understand PCA
  - Find maximum variance projection
  - Find projection with minimum reconstruction error
  - Graphical model (probabilistic PCA)
    - EM for high dimensional data
- Extensions
  - Factor analysis- removes assumptions about shared variance
  - Kernel PCA- non-linear version of PCA