Introduction to Machine Learning

PCA and Spectral Clustering

Introduction to Machine Learning, 2013-14
Slides: Eran Halperin

Singular Value Decomposition (SVD)

The singular value decomposition (SVD) of any matrix X is:

$$X = U\Sigma V^t$$

$$X = \begin{pmatrix} | & & | \\ u_1 & \dots & u_m \\ | & & | \end{pmatrix} \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ 0 & & \sqrt{\lambda_n} \\ & \ddots & \\ & & 0 \end{pmatrix} \begin{pmatrix} - & v_1 & - \\ \vdots & \vdots & \vdots \\ - & v_n & - \end{pmatrix}$$

Existence of SVD

Spectral decomposition (assume for simplicity n = m):

$$XX^t = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^t$$

Existence of SVD

Spectral decomposition (assume for simplicity n = m):

$$XX^t = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} U^t$$

Define: $\Sigma = \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_n} \end{pmatrix}$

Define: $V = X^t U \Sigma^{-1}$

Existence of SVD

$$XX^t = U\Sigma\Sigma U^t$$

$$V = X^t U \Sigma^{-1}$$

We will assume all eigenvalues are > 0.

$$V^t V = \left(\Sigma^t\right)^{-1} U^t X X^t U \Sigma^{-1} = I$$

Existence of SVD

$$XX^t = U\Sigma\Sigma U^t$$

$$V = X^t U\Sigma^{-1}$$

We will assume all eigenvalues are > 0.

$$V^t V = \left(\Sigma^t\right)^{-1} U^t X X^t U \Sigma^{-1} = I$$

$$U\Sigma V^t = U\Sigma\Sigma^{-1} U^t X = X$$

Existence of SVD

$$XX^t = U\Sigma\Sigma U^t$$

$$V = X^t U\Sigma^{-1}$$

We will assume all eigenvalues are > 0.

$$V^t V = \left(\Sigma^t\right)^{-1} U^t X X^t U\Sigma^{-1} = I$$

$$U\Sigma V^t = U\Sigma\Sigma^{-1}U^t X = X$$

Note that the transposed matrix has the same eigenvalues:

$$X^t X v_i = V\Sigma^2 V^t v_i = \lambda_i v_i$$

SVD and linear regression

Recall, in linear regression:

$$\hat{a} = min_a \|y - Xa\|_2^2$$

$$\hat{a} = (X^t X)^{-1} X^t y$$

If $X^t X$ is singular there are many solution. How can we find the solution with the smallest norm?

$$\min \|a\|$$
$$s.t. \quad X^t X a = X^t y$$

SVD and linear regression

$$\min \|a\|$$

$$s.t. \quad X^t X a = X^t y$$

$$X = U \Sigma V^t$$

$$V \Sigma^2 V^t a = V \Sigma U^t y$$

$$\Sigma^2 V^t a = \Sigma U^t y$$

SVD and linear regression

Assume: $a = \sum_i \alpha_i v_i$ $\longrightarrow$ $\|a\|^2 = \sum_{i=1}^{n} \alpha_i^2$

$$\Sigma^2 V^t a = \Sigma U^t y$$

$$\Sigma^2 V^t a = \begin{pmatrix} \alpha_1 \lambda_1 \\ \alpha_2 \lambda_2 \\ \dots \\ \dots \\ \alpha_n \lambda_n \end{pmatrix} \longrightarrow \alpha_i = \frac{u_i^t y}{\sqrt{\lambda_i}}$$

The solution is unique since if $\lambda_i = 0$ then it is always better to set $\alpha_i = 0$

SVD and linear regression

$$a = \sum_i \alpha_i v_i \qquad \alpha_i = \frac{u_i^t y}{\sqrt{\lambda_i}}$$

$$a = \sum_i \frac{1}{\sqrt{\lambda_i}} u_i^t y v_i = V \Sigma^{-1} U^t y$$

Pseudoinverse of X^t

SVD and linear regression

$$a = \sum_i \alpha_i v_i \qquad \alpha_i = \frac{u_i^t y}{\sqrt{\lambda_i}}$$

$$a = \sum_i \frac{1}{\sqrt{\lambda_i}} u_i^t y v_i = V \Sigma^{-1} U^t y$$

Pseudoinverse of X^t

Exercise: Show that the Ridge solution converges to the pseudoinverse solution:

$$\lim_{\epsilon \to 0} (X^t X + \epsilon I)^{-1} X^t y = a$$

Unsupervised Learning

Supervised Learning:

$$(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, ?), \ldots, (x_m, ?)$$

Fill in the missing labels

Unsupervised Learning:

$$(x_1, ?), \ldots, (x_m, ?)$$

Fill in the missing labels

High Dimensional Data

- Documents
- Face images
- Genetic data

Dimensionality Reduction

- Avoid the risk of over-fitting.
- Data compression.
- Visualization.
- Outlier detection.

Linear Dimensionality Reduction

Input: $(x_1, \ldots, x_n)$

$$x_i \in R^m$$

$$X = \begin{pmatrix} | & & | \\ x_1 & \ldots\ldots & x_n \\ | & & | \end{pmatrix}$$

Output: $(z_1, \ldots, z_n)$

$$z_i = U^t x_i \in R^d$$

$$u_i \cdot u_j = 0$$

$$\|u_i\| = 1$$

$$U = \begin{pmatrix} | & & | \\ u_1 & \ldots\ldots & u_d \\ | & & | \end{pmatrix}$$

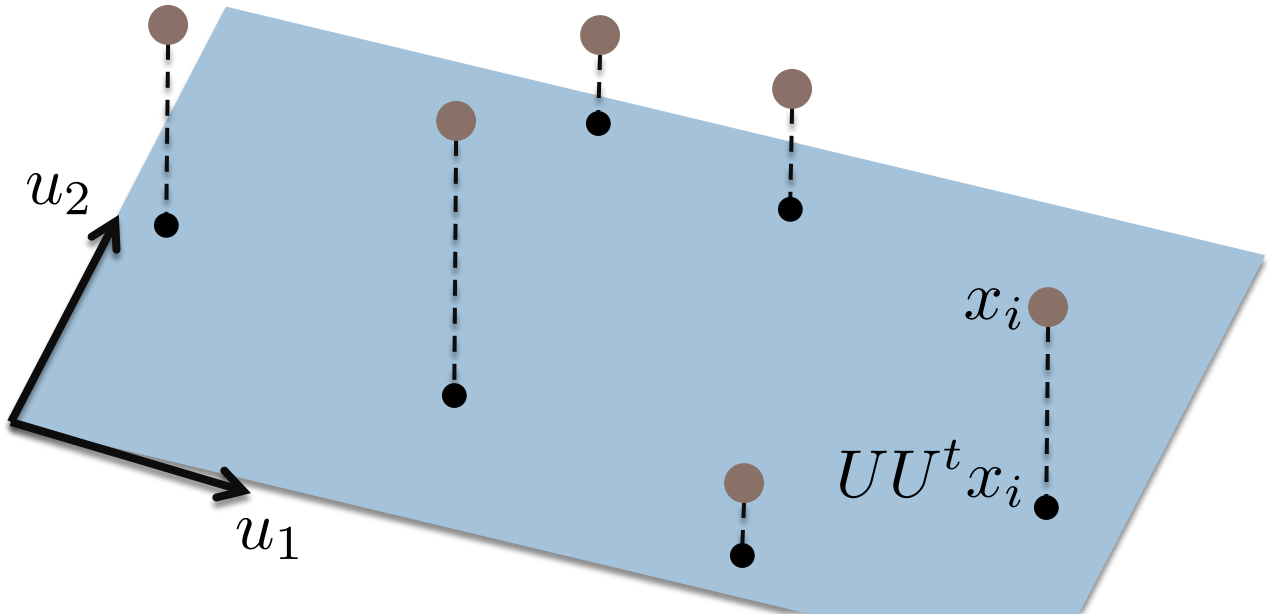
How should we choose U?

Reconstruction Error

We can think of dimensionality reduction as compression.

Encode step: $z_i = U^t x_i \in R^d$

Decode step: $x_i' = U z_i = U U^t x_i \in R^m$



Reconstruction Error

We can think of dimensionality reduction as compression.

Encode step: $z_i = U^t x_i \in R^d$

Decode step: $x_i' = U z_i = U U^t x_i \in R^m$

Minimize the loss of information:

$$U = \arg \min_{U \in R^{m \times d}} \sum_i \|x_i - U U^t x_i\|_2^2$$

Capturing Variance

Assume the data points are centered at zero. That is:

$$\sum_{i=1}^{n} x_i = 0$$

The estimate of the variance of the distance from the origin:

$$Var(X) = \frac{1}{n-1} \sum_{i=1}^{n} x_i^t x_i$$

The estimate of the variance of the projected distance:

$$Var(X') = \frac{1}{n-1} \sum_{i=1}^{n} x_i^t U U^t U U^t x_i = \frac{1}{n-1} \sum_{i=1}^{n} x_i^t U U^t x_i$$

Capturing Variance

Recall:

$$U = \arg \min_{U \in R^{m \times d}} \sum_i \|x_i - UU^t x_i\|_2^2$$

$$Var(X') = \frac{1}{n-1} \sum_{i=1}^{n} x_i^t UU^t UU^t x_i = \frac{1}{n-1} \sum_{i=1}^{n} x_i^t UU^t x_i$$

Note:

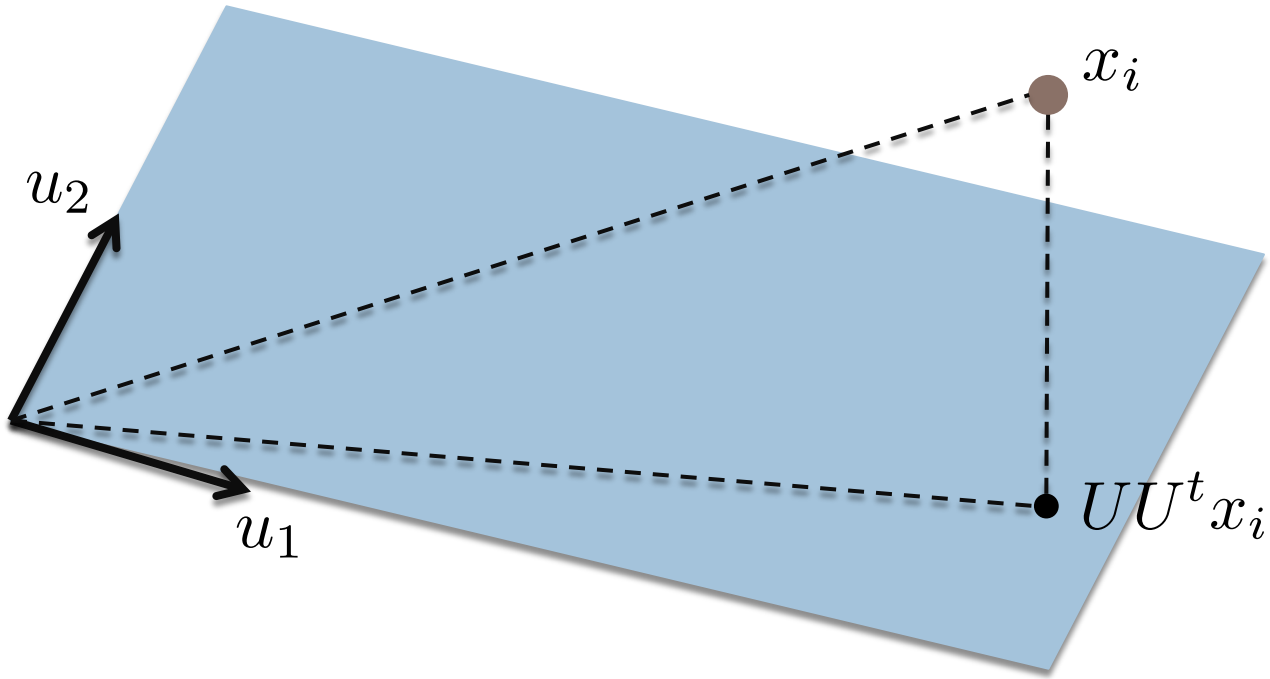
$$x_i = UU^t x_i + (x_i - UU^t x_i)$$

$$\|x_i\|_2^2 = \|U^t x_i\|_2^2 + \|x_i - UU^t x_i\|_2^2$$

Minimize reconstruction error = maximize projected variance

Reconstruction Error

$$\|x_i\|_2^2 = \|U^t x_i\|_2^2 + \|x_i - UU^t x_i\|_2^2$$



Principal Component Analysis

Input: $X = \begin{pmatrix} | & & | \\ x_1 & \ldots\ldots & x_n \\ | & & | \end{pmatrix}$

Output: $U = \begin{pmatrix} | & & | \\ u_1 & \ldots\ldots & u_d \\ | & & | \end{pmatrix}$

$$(z_1, \ldots, z_n)$$

$$z_i = U^t x_i \in R^d$$

$$u_i \cdot u_j = 0$$

$$\|u_i\| = 1$$

Objective:

$$U = \arg \min_{U \in R^{m \times d}} \sum_i \|x_i - UU^t x_i\|_2^2$$

$$U = \arg \max_{U \in R^{m \times d}} \sum_i x_i^t UU^t x_i$$

PCA – one dimension

$$u_1 = \arg \max_{u \in R^m, \|u\|=1} \sum_{i=1}^{n} x_i^t u u^t x_i$$

$$\begin{aligned} \sum_{i=1}^{n} x_i^t u u^t x_i &= \sum_{i=1}^{n} (u^t x_i)^2 \\ &= \|u^t X\|_2^2 = u^t X X^t u \end{aligned}$$

u_1 is the largest eigenvector of XX^t

PCA – d dimensions

$$\sum_{i=1}^{n} x_i^t U U^t x_i = \sum_{i=1}^{n} x_i^t (\sum_{j=1}^{d} u_j u_j^t) x_i = \sum_{j=1}^{d} u_j^t X X^t u_j$$

Let $v_1, \ldots, v_m$ be the eigenvectors of XX^t

with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_m$

PCA – d dimensions

$$\sum_{i=1}^{n} x_i^t U U^t x_i = \sum_{i=1}^{n} x_i^t (\sum_{j=1}^{d} u_j u_j^t) x_i = \sum_{j=1}^{d} u_j^t X X^t u_j$$

Let $v_1, \ldots, v_m$ be the eigenvectors of XX^t

with eigenvalues $\lambda_1 \geq \ldots \geq \lambda_m$

$$\sum_{i=1}^{n} x_i^t U U^t x_i = \sum_{j=1}^{d} u_j^t X X^t u_j = \sum_{j=1}^{d} \sum_{k=1}^{m} (u_j^t v_k)^2 \lambda_k = \sum_{k=1}^{m} \lambda_k \sum_{j=1}^{d} (u_j^t v_k)^2$$

When $u_k = v_k$ we get that the variance is $\sum_{k=1}^{d} \lambda_k$.

PCA – d dimensions

$$\sum_{i=1}^{n} x_i^t U U^t x_i = \sum_{k=1}^{m} \lambda_k \sum_{j=1}^{d} (u_j^t v_k)^2$$

Denote: $a_k = \sum_{j=1}^{d} (u_j^t v_k)^2 = \|U^t v_k\|^2$

$$\sum_{k=1}^{m} a_k = \sum_{k=1}^{m} \sum_{j=1}^{d} (u_j^t v_k)^2 = \sum_{j=1}^{d} \sum_{k=1}^{m} (u_j^t v_k)^2$$

$$= \sum_{j=1}^{d} \|u_j^t V\|^2 = d$$

PCA – d dimensions

$$\begin{aligned} \max \quad & \sum_{k=1}^{m} \lambda_k a_k \\ s.t. \quad & \sum_{k=1}^{m} a_k = d \\ & 0 \leq a_k \leq 1 \end{aligned}$$

Maximized when $a_1 = a_2 = \ldots = a_d = 1$

Maximal value of the solution is $\sum_{k=1}^{d} \lambda_k$

$u_k = v_k$ is the maximal solution.

Principal Component Analysis (PCA)

- The entries of u_i are called the loadings of the i-th principal component
- The contribution of the i-th principal component to the explained variance is: $u_i^t X X^t u_i = \lambda_i$
- The number of PCs can be determined by the eigenvalue distribution.
- The PC scores are obtained by

$$u_i^t X = u_i^t U \Sigma V^t = \sqrt{\lambda_i} v_i$$

$$v_i = \frac{u_i^t X}{\|u_i^t X\|} = \arg \max_{\|v\|=1, \forall j<i, v^t v_j = 0} v^t X^t X v$$

Computing PCA

Computing the eigenvectors requires the computation of XX^t

Time: $O(nm^2)$ + time for eigenvectors

More efficient – compute the singular value decomposition (SVD) with d singular values.

$$X = U\Sigma V^t$$

Example: Eigenfaces

Goal: Recognition and/or detection of a face.

Example: Eigenfaces

- Each picture is a matrix of pixels. We can think of it as a vector of real numbers.
- Compute PCA, represent each picture by a small number of directions.
- Given a new picture:
  - Project the picture on the space defined by the PCs.
  - Compute the distance to the face points.
  - If the distance is small enough this is a face.
  - Nearest neighbor classification (face detection).

Using more eigenfaces allows for a better reconstruction.

Number of eigenfaces: 10,30,50,70,…,310

Example: Latent Semantic Analysis

Goal: Cluster documents based on word count similarity.

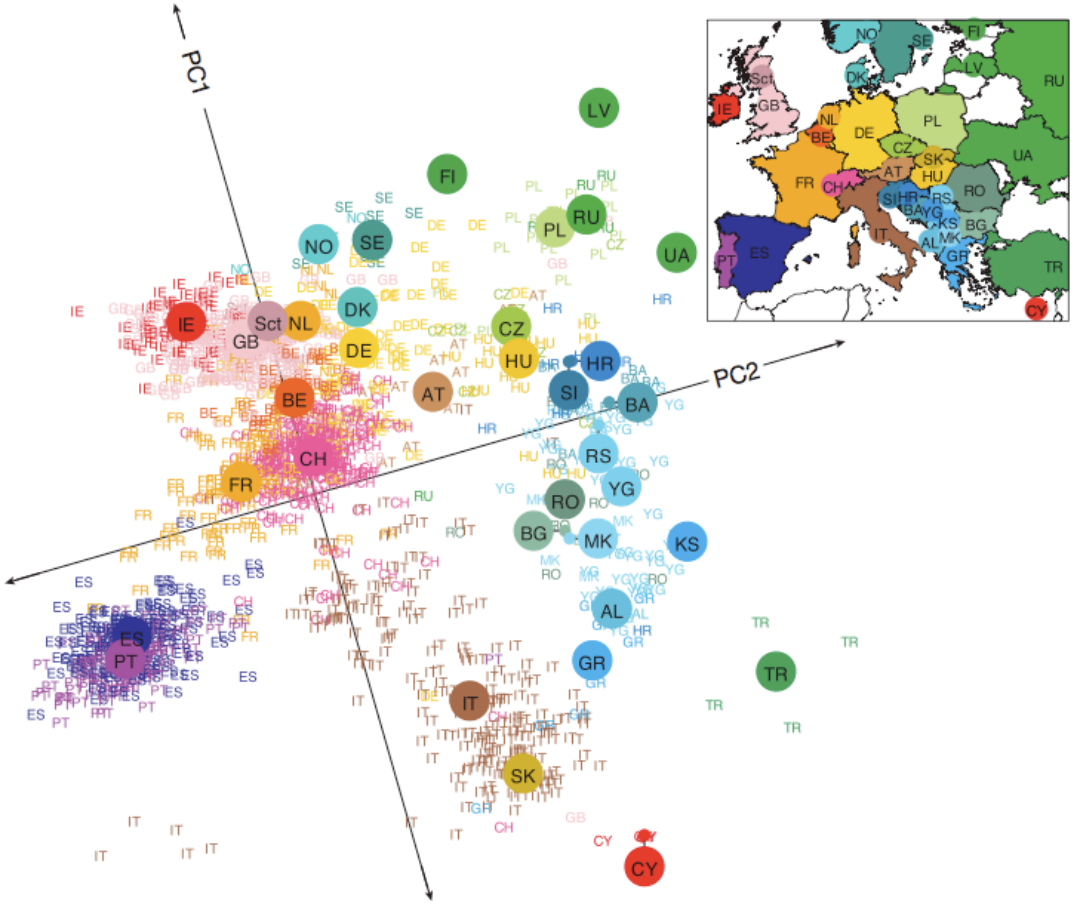
Generate a vector for each document: the histogram of words.

Compute PCA. Run clustering in the projected space.

Example: Population Structure

- Goal: Use genetic data to cluster individuals based on their ancestry.
- Input: the allele count (e.g., number of copies of 'A') in each position in the genome.
- Visualize and cluster populations based on the first PCs, detect outliers.

PCA on a European population



Practical Issues

- Features have to be standardized.
  - No standardization leads to a non-uniform contribution of the features.
- Outliers can affect the results. The variance explained in the dimension spanned by the outlier is typically large.
  - In population structure, the first PCs could be due to outliers (e.g., individuals with wrong genotype data, or a different platform).
  - Eigenfaces could be affected by the camera used, time of the day, etc.

Regression and Normalization

PCA is often used for normalization of data by regressing over the first PCs.

The first PCs often correspond to general phenomena that are not related to the data.

- Comparing two sets of faces – the main difference may be due to the camera used in each group or the amount of light used.
- Comparing genomes of patients and controls – population structure, genotype lab, are confounders.

Spectral Clustering - Intuition

Scenario: two clusters, A, B each of size n/2.

$$i, j \in A \Rightarrow x_i^t x_j \approx C \gg 0$$

$$i, j \in B \Rightarrow x_i^t x_j \approx C \gg 0$$

$$i \in A, j \in B \Rightarrow x_i^t x_j \approx -D \ll 0$$

Note:

$$v^t X^t X v = \sum_{i,j} v_i v_j x_i^t x_j$$

is maximized when

$$i \in B \Rightarrow v_i \approx -\frac{1}{\sqrt{n}} \qquad i \in A \Rightarrow v_i \approx \frac{1}{\sqrt{n}}$$

Spectral Clustering - Intuition

$$v^t X^t X v = \sum_{i,j} v_i v_j x_i^t x_j$$

$$\sum_{i,j} x_i^t x_j v_i v_j = -\frac{1}{2} \sum_{i,j} x_i^t x_j (v_i - v_j)^2 + \sum_i v_i^2 x_i^t \sum_j x_j$$

Spectral Clustering - Intuition

$$v^t X^t X v = \sum_{i,j} v_i v_j x_i^t x_j$$

$$\sum_{i,j} x_i^t x_j v_i v_j = -\frac{1}{2} \sum_{i,j} x_i^t x_j (v_i - v_j)^2 + \sum_i v_i^2 x_i^t \sum_j x_j$$

Spectral Clustering

Given a weighted graph with edge weights $W \in R_+^{n \times n}$

Define

$$d_i = \sum_{j=1}^{n} w_{ij} \qquad D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{pmatrix}$$

The Laplacian of G is defined as $L = D - W$

$$v^t(D-W)v = \sum_{i=1}^{n}\sum_{j=1}^{n} v_i v_j (D_{ij} - w_{ij}) = \sum_{i<j} (v_i - v_j)^2 w_{ij}$$

Find the smallest eigenvalues orthogonal to (1,1,…,1)

Spectral Clustering

Find the smallest eigenvalues orthogonal to (1,1,…,1):

$$\begin{aligned} \min \quad & \sum_{i<j} (v_i - v_j)^2 w_{ij} \\ s.t. \quad & \|v\| = 1 \\ & \sum_{i=1}^{n} v_i = 0 \end{aligned}$$

Cluster the values of v.

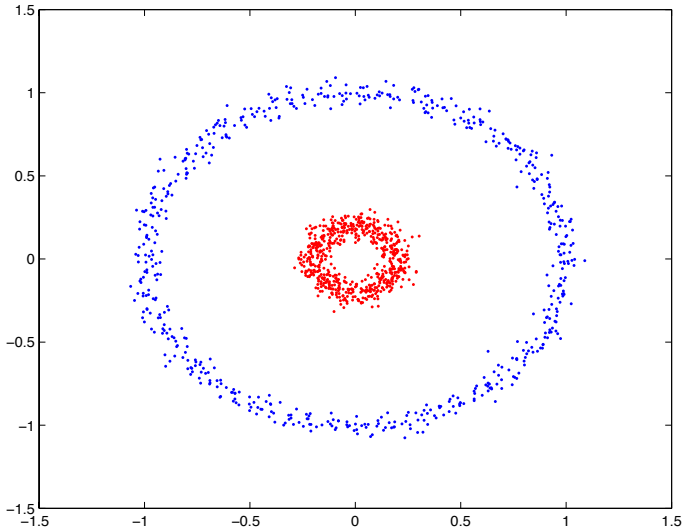
We can add more eigenvectors and run k-means in the resulting space.

Spectral Clustering

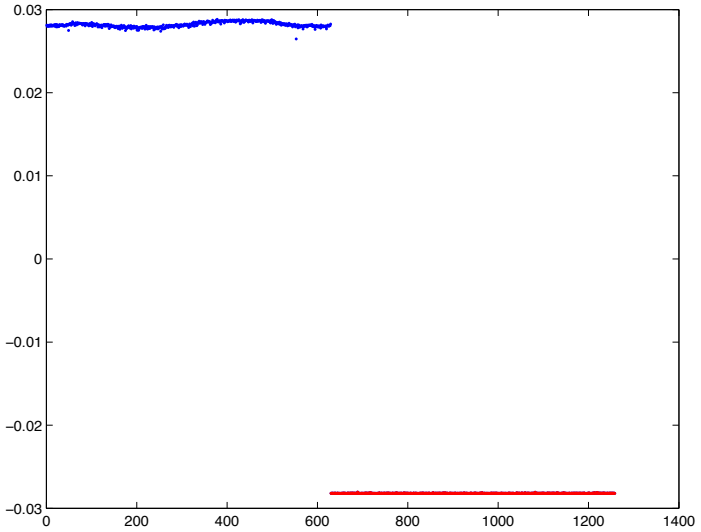
- A general framework that can work with every similarity measure (not only scalar product).
- It can be derived as a relaxation to the following min sparse cut problem (normalized cut problem):

$$score(A,B) = \frac{\sum_{i\in A, j\in B} w_{ij}}{\sum_{i\in A, j\in V} w_{ij}} + \frac{\sum_{i\in A, j\in B} w_{ij}}{\sum_{i\in B, j\in V} w_{ij}}$$

Spectral Clustering



Original data



2nd smallest eigenvector

$$w_{ij} = e^{-10\|x_i - x_j\|}$$

PCA – Probabilistic View

Model assumptions:

$$x = Wz + \epsilon$$

$$z \in R^d, W \in R^{m \times d}, \epsilon \sim N(0, \sigma^2 I_m)$$

PCA – Probabilistic View

Model assumptions:

$$x = Wz + \epsilon$$

$$z \in R^d, W \in R^{m \times d}, \epsilon \sim N(0, \sigma^2 I_m)$$

Likelihood:

$$\log L(X; Z, W) = -mn \log(\sqrt{2\pi}\sigma) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \|x_i - Wz_i\|^2$$

Maximizing the likelihood is equivalent to minimizing

$$\sum_{i=1}^{n} \|x_i - Wz_i\|^2$$

PCA – Probabilistic View

Minimize $\sum_{i=1}^{n} \|x_i - W z_i\|^2$

Fixing W, we get a standard linear regression problem:

$$\hat{z}_i = (W^t W)^{-1} W^t x_i$$

Plugging back in, we need to minimize:

Minimize $\sum_{i=1}^{n} \|x_i - W(W^t W)^{-1} W^t x_i\|^2$

PCA – Probabilistic View

Minimize $\sum_{i=1}^{n} \|x_i - W(W^tW)^{-1}W^t x_i\|^2$

Using SVD: $W = USV^t$

$$W(W^tW)^{-1}W^t = US(S^tS)^{-1}S^tU^t = U_dU_d^t$$

We get:

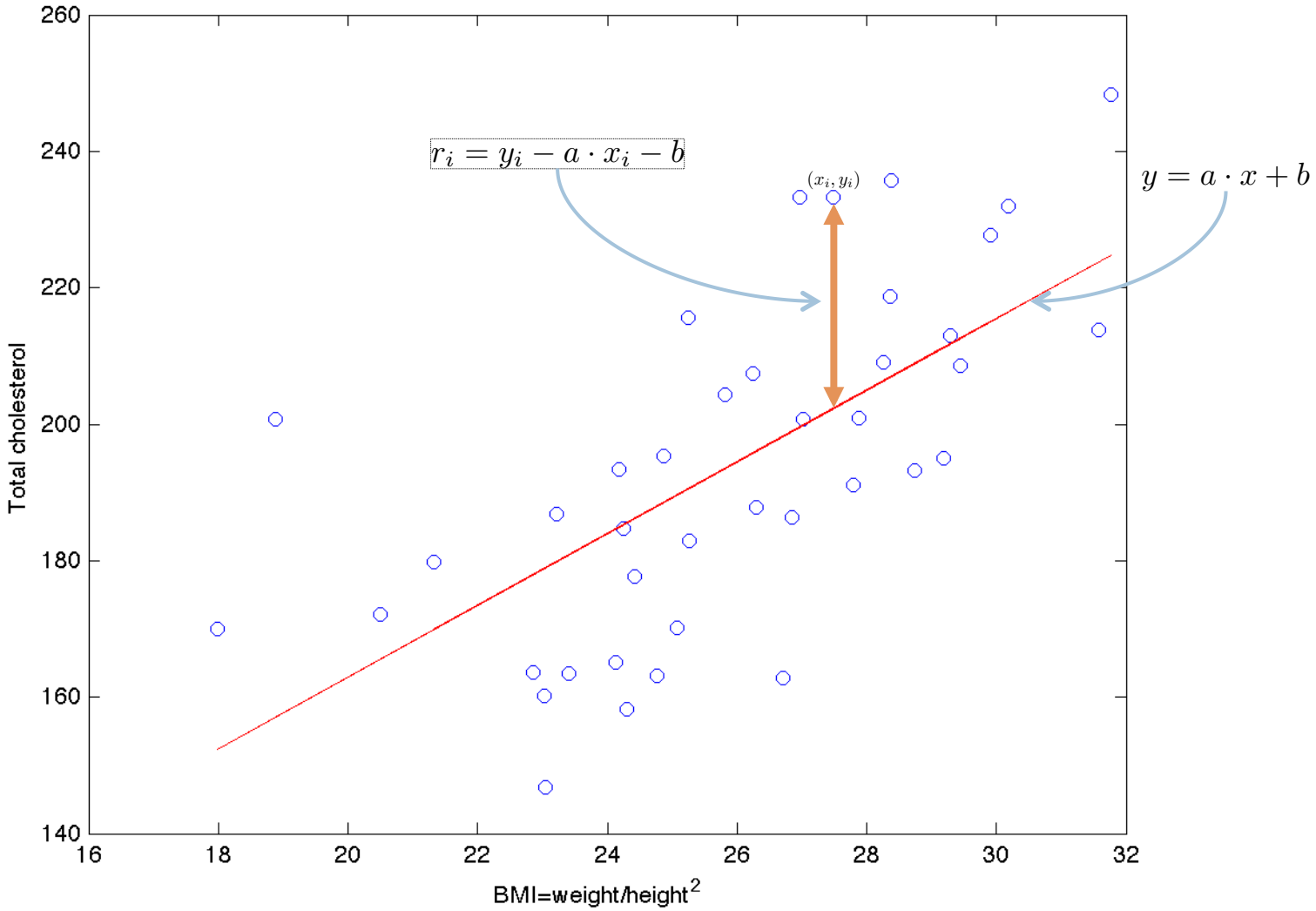
$$\hat{U}_d = \arg\min_{U_d} \sum_{i=1}^{n} \|x_i - U_dU_d^t x_i\|^2$$

PCA – Probabilistic View

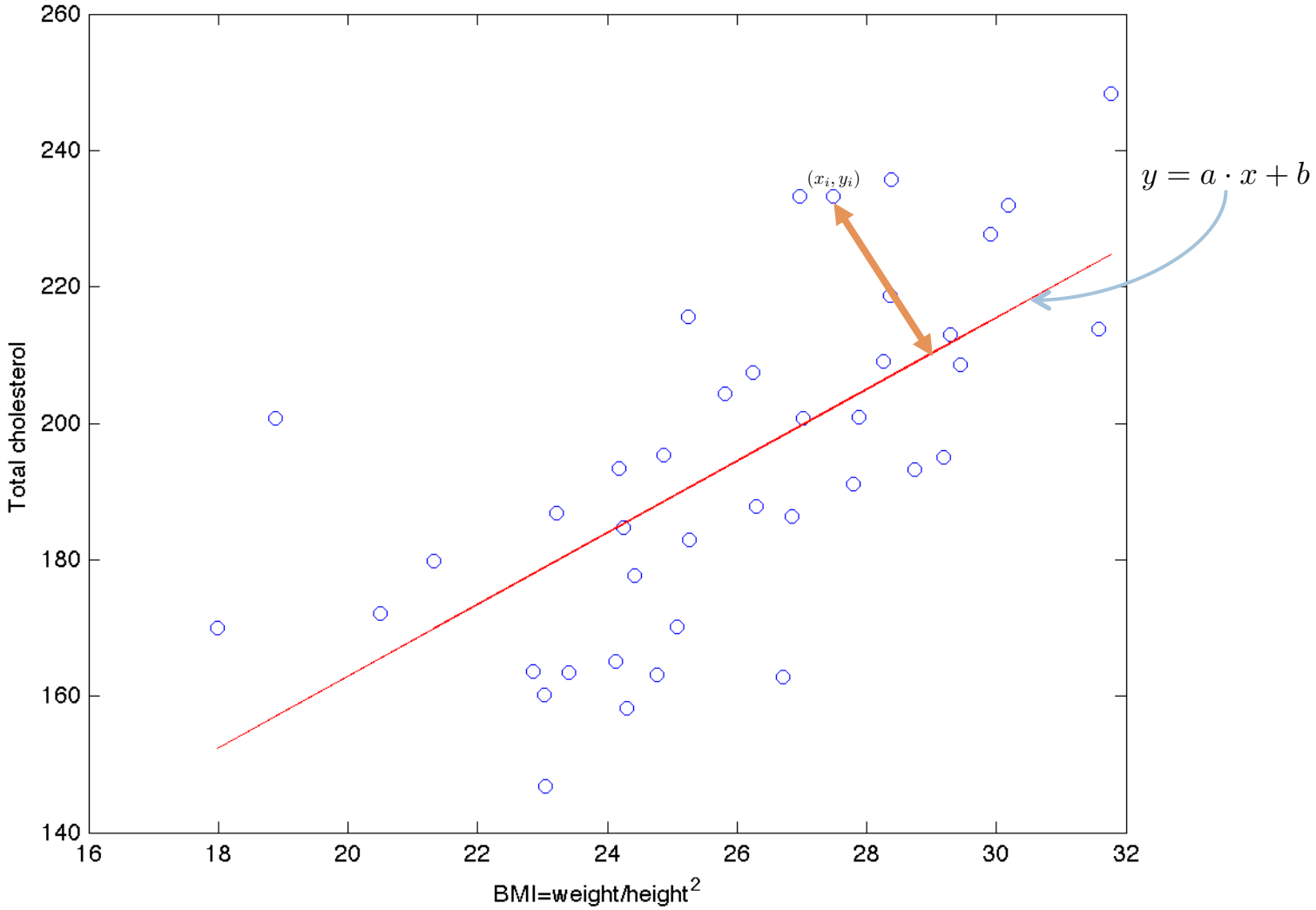
$$x = Wz + \epsilon$$

- We can deal with missing data using Expectation Maximization.
- We can model a mixture of PCAs.
- Often the variables z_i are treated as random variables and the standard deviation as unknown.
  - The derivation is then different, as well as the result.

Regression: Minimize the Residuals



PCA: Minimize the reconstruction error



Interpretation of the Loadings

- Loading interpretation:
  - Which pixels provide the most important contribution to the eigenfaces?
  - Which genes contribute the most to the difference in ancestry?

Sparse PCA

- Goal: PCs with a small number of non-zero loadings.
  - Interpretation is easier.
  - Avoid capturing noise when m = O(n) and the 'true structure' is defined by a small dimension.

$$u_1 = \arg \max_{u \in R^m, \|u\|_2 = 1, \|u\|_0 \leq k} u^t X X^t u$$

Sparse PCA

SCoTLASS: Simplified Component Technique-Lasso

$$
\begin{aligned}
u_1 = \quad & \arg\max u^t X X^t u \\
s.t. \quad & \|u\|_2 = 1 \\
& \|u\|_1 \leq t
\end{aligned}
$$

Sparse PCA

SCoTLASS: Simplified Component Technique-Lasso

$$\begin{aligned} u_1 = \quad & \arg\max u^t X X^t u \\ s.t. \quad & \|u\|_2 = 1 \\ & \|u\|_1 \leq t \end{aligned}$$

$$\begin{aligned} u_k = \quad & \arg\max u^t X X^t u \\ s.t. \quad & \|u\|_2 = 1 \\ & \forall i < k, u^t u_i = 0 \\ & \|u\|_1 \leq t \end{aligned}$$

Disadvantage: Slow (non-convex optimization).
In practice not sparse when a high fraction of the variance is explained.

Sparse PCA

A regression view - Run PCA and approximate the PCs using sparse PCs.

$$\hat{\beta}_i = \arg\min_{\beta} \|u_i^t X - \beta^t X\|^2 + \lambda_1 \|\beta\|_1$$

Problem: If X is not full rank, the above formulation might not return the PC when $\lambda_1 = 0$

Sparse PCA

A regression view - Run PCA and approximate the PCs using sparse PCs.

$$\hat{\beta}_i = \arg\min_{\beta} \|u_i^t X - \beta^t X\|^2 + \lambda_1 \|\beta\|_1$$

Problem: If X is not full rank, the above formulation might not return the PC when $\lambda_1 = 0$

Solution: Add L2 regularization:

$$\hat{\beta}_i = \arg\min_{\beta} \|u_i^t X - \beta^t X\|^2 + \lambda_2 \|\beta\|^2 + \lambda_1 \|\beta\|_1$$

Sparse PCA

Theorem: If $\lambda_1 = 0$ and $\lambda_2 > 0$ then $\frac{\hat{\beta}_i}{\|\hat{\beta}_i\|_2} = u_i$.

Proof:
$$\begin{aligned} \hat{\beta}_i &= (XX^t + \lambda_2 I)^{-1} XX^t u_i \\ &= U(\Sigma^2 + \lambda_2 I)^{-1} U^t U \Sigma^2 U^t u_i \\ &= u_i \frac{\Sigma_{ii}^2}{\Sigma_{ii}^2 + \lambda_2} \end{aligned}$$

Note this formulation is a convex quadratic program.