Dimensionality reduction

EE219: Large Scale Data Mining

Professor Roychowdhury

Jan 17, 2018

Summary

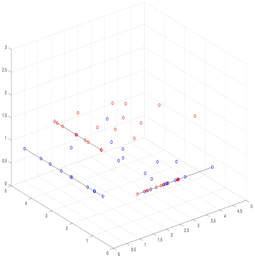
- Review
  - Binary classifier
  - Covariance matrix
- Unsupervised binary classifier
- PCA
  - Eigenvalue and eigenvector
  - Approximation
  - pick k
- SVD
  - SVD approximation
  - Term-Document matrix

Review: binary classifier

- In the linear binary classifier, the input is $x_1, ..x_n \in \mathrm{R}^d$ and the corresponding output is $y_1, ..y_n$. We want to find a function f: $\mathrm{R}^d \to \mathrm{R}$. It can be viewed as a projection operation.
- Without loss of generality, we replace x_i with $x_i - \overline{x}$, where $\overline{x} = \frac{1}{n}\sum\limits_{i=1}^{n} x_i$ is the center of data.
- We want to pick a w $\in \mathrm{R}^d$, then $y_i = w^T x_i$, and $\overline{y_i} = \frac{1}{n}\sum\limits_{i=1}^{n} w^T x_i = w^T \frac{1}{n}\sum\limits_{i=1}^{n} x_i = 0$
- $\hat{\sigma_y} = \frac{1}{n}\sum\limits_{i=1}^{n} y_i^2 = \frac{1}{n}\sum\limits_{i=1}^{n}(w^T x_i)(w^T x_i) = w^T(\frac{1}{n}\sum\limits_{i=1}^{n} x_i x_i^T)w$
- Define $R = \frac{1}{n}\sum\limits_{i=1}^{n} x_i x_i^T$, then $R = Cov(X) = E[XX^T]$, where $R_{k\ell} = \frac{1}{n}\sum\limits_{i=1}^{n} x_i(k)x_i(\ell)$
- For the original $x_1, ..x_n$ samples before shifting, $R_{k\ell} = Cov(X(k), X(\ell)) = E[(X(k) - E[X(k)])(X(\ell) - E[X(\ell)])]$

Unsupervised binary classifier

If the output or labels y_i are not given, our aim is to find a w that maximize $\hat{\sigma_y}(w) = w^T R w$.
This is also called Principal Component Analysis(PCA)



As shown in the picture, two projections bring different output distributions. Left one can distinguish data better than the right one.

PCA

- $\hat{\sigma_y}(cw) = c^2\hat{\sigma_y}(w)$, so if picking c $\to \infty$, you can get unbounded result. Without loss of generality, we add constraint $\|w\|_2 = 1$ to the optimization problem.
- $\max\limits_{\substack{w \\ s.t. \|w\|_2 = 1}} : w^T R w = \max\limits_{w} : \frac{w^T R w}{w^T w} = \lambda_{max}$
- λ_{max} is the largest eigenvalue of R.
- How to find the second largest eigenvalue and corresponding eigenvector?
- How to find k largest eigenvalues and corresponding eigenvectors? How to pick k?

Eigenvalue and eigenvector

- A vector $z \in \mathrm{C}^d$ is an eigenvector of an arbitrary matrix $R \in \mathrm{R}^{d \times d}$ if $Rz = \lambda z, \lambda \in \mathrm{C}$.
- If $R = R^T$ and real valued, then λ is real and $\lambda \geq 0$. In addition, if $Rz_1 = \lambda_1 z_1, Rz_2 = \lambda_2 z_2$, then z_1 and z_2 are orthogonal, or $z_1^T z_2 = 0$
-
  - $R[z_1 ... z_d] = [z_1 ... z_d] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_d \end{bmatrix}$ where $\lambda_1 \lambda_2 ... \geq \lambda_d$
  - RU = UΛ, then R = $U \Lambda U^T$ shows eigendecomposition of R, where $UU^T = I, U^{-1} = U^T$
  - $w^* = z_1$ is the principal eigenvector corresponding to the largest eigenvalue λ_1

PCA

Use the previous properties to find the second largest eigenvalue:

$$\max_{\substack{w \\ s.t.\|w\|_2=1, w^T z_1=0}} : \frac{w^T R w}{w^T w} = \lambda_2$$

For example:

- $\mathrm{f} : \mathrm{R^d} \rightarrow \mathrm{R^3}$

$$f(x_i) = \begin{bmatrix} \text{——} & z_1^T & \text{——} \\ \text{——} & z_2^T & \text{——} \\ \text{——} & z_3^T & \text{——} \end{bmatrix} \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(d) \end{bmatrix} = \begin{bmatrix} z_1^T x_i \\ z_2^T x_i \\ z_3^T x_i \end{bmatrix}$$

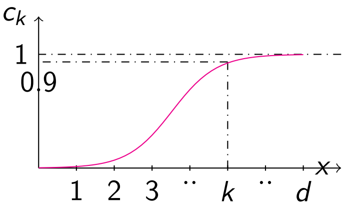
- f: $\mathrm{R^d} \rightarrow \mathrm{R^k}$

$$f(x_i) = \begin{bmatrix} \text{——} & z_1^T & \text{——} \\ \text{——} & z_2^T & \text{——} \\ & \vdots & \\ \text{——} & z_k^T & \text{——} \end{bmatrix} \begin{bmatrix} x_i(1) \\ x_i(2) \\ \vdots \\ x_i(d) \end{bmatrix} = \begin{bmatrix} z_1^T x_i \\ \vdots \\ z_k^T x_i \end{bmatrix}$$

PCA

How to pick k?

- Total variance post projection is $\sum_{i=1}^{d} \lambda_i$
- Variance after projecting along the first k eigenvectors is $\sum_{i=1}^{k} \lambda_i$
- The fraction $c_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$



Generalization of eigenvalue decomposition

Given $x_1, x_2, ..x_N \in \mathrm{R^d}$, $\mathrm{R} = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$. Let $\mathrm{Y} = \begin{bmatrix} | & \vdots & | \\ x_1 & \vdots & x_n \\ | & \vdots & | \end{bmatrix}$

Then $\mathrm{R} = \frac{1}{n} YY^T$. Instead of dealing with YY^T, we can analyze Y directly by singular value decomposition.

- $\mathrm{Y} = U\Sigma V^T$

$$= \underbrace{\begin{bmatrix}\mathbf{u}_1 & .. & \mathbf{u}_r\end{bmatrix}}_{\mathrm{Col}\, A} \underbrace{\begin{matrix} .. & \mathbf{u}_m \end{matrix}\Big]}_{\mathrm{Nul}\, A^T} \begin{bmatrix} \sigma_1 & 0 & .. & 0 & 0..0 \\ \dots & & & & \\ 0 & 0 & .. & \sigma_r & 0..0 \\ 0 & 0 & .. & 0 & 0..0 \\ \dots & & & & \\ 0 & 0 & .. & 0 & 0..0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \dots \\ \mathbf{v}_r^T \\ \mathbf{v}_{r+1}^T \\ \dots \\ \mathbf{v}_n^T \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Row } A \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} \text{Nul } A \end{matrix}$$

- $UU^T = I, VV^T = I$
- $YY^T = U(\Sigma\Sigma^T)U^T$, U are the eigen vectors of YY^T
- $Y^TY = V(\Sigma^T\Sigma)V$, V are the eigen vectors of Y^TY

SVD applications

When n,d have meanings, we can consider SVD.
For example,in the text analysis, we use $D_1, ..D_n$ to represent n documents and $T_1, ..T_d$ to represent all the words or terms shown in these documents and forgetting their orders.

$$Y = \begin{array}{c c} & \begin{array}{ccccc} D_1 & .. & D_j & .. & D_n \end{array} \\ \begin{array}{c} T_1 \\ \vdots \\ T_i \\ \vdots \\ T_d \end{array} & \left(\begin{array}{ccccc} & & & & \\ & & & & \\ & & f_{ij} & & \\ & & & & \\ & & & & \end{array} \right) \end{array}$$ is called Term/Document

Matrix , where Y_{ij} represents the number of times ith term appears on the j th document.
Y $= U_{d\times d}\Sigma_{d\times n}V^T_{n\times n}$

SVD application

For example, when d = 20k, n = 100k, using SVD can reduce the dimension. In this case,

$$\Sigma = \left[\begin{array}{cccc:c} \sigma_1 & 0 & .. & 0 & 0..0 \\ \dots & & & & \dots \\ 0 & 0 & .. & \sigma_d & 0..0 \end{array}\right]$$

- Approximate $\hat{Y} = U\hat{\Sigma}V^T$, where $\hat{\Sigma} = \left[\begin{array}{cccc:c:c} \sigma_1 & 0 & .. & 0 & 0 & 0..0 \\ \dots & & & & \dots & 0..0 \\ 0 & 0 & .. & \sigma_k & 0 & 0..0 \\ \hdashline 0 & 0 & .. & 0 & 0 & 0..0 \end{array}\right]$

Term-Document matrix

$$YY^T_{d\times d} = \begin{array}{cc} & \begin{array}{ccc} D_1 & .. & D_n \end{array} \\ \begin{array}{c} T_1 \\ \vdots \\ T_i \\ \vdots \\ T_d \end{array} & \begin{bmatrix} & & \\ & & \\ - & - & \\ & & \\ & & \end{bmatrix} \end{array} \begin{array}{c} \begin{array}{ccccc} T_1 & .. & T_j & .. & T_n \end{array} \\ \begin{bmatrix} & & | & & \\ & & | & & \\ & & | & & \\ & & | & & \end{bmatrix} \end{array}$$

- $(YY^T)_{i,j} = T_i^T T_j$ measures the cooccurrence of the jth and ith term. This value measures how similarity they are in the document space. It can be used to cluster terms.
- Given $D_j \in \mathrm{R}^d$, $T_j \in \mathrm{R}^n$, we can project the T_i and $D_i to \mathrm{R}^{\mathrm{k}}$ This is Latent Semantic Analysis/Indexing. It will be further discussed in the following lecture.