Lectures 3 and 4: Principal Component Analysis: A Technique for Data Decomposition and Approximation

Radu Balan

Department of Mathematics, NWC
University of Maryland, College Park, MD

Version: February 7, 2023

Approaches

The Principal Component Analysis (PCA) is data processing method that belongs to the class of dimension reduction and data embedding techniques.

Fundamentally it is a least-squares fitting algorithm with respect to a set of basis vectors that are determined based on data. As such it is naturally connected to the least-squares fitting problems we studied so far and hence can be presented now.

A list of popular dimension reduction and data embedding approaches includes:

1. Principal Component Analysis
2. Independent Component Analysis
3. Laplacian Eigenmaps
4. Local Linear Embeddings (LLE)
5. Isomaps

Approaches

In all these case, the input data is given by a collection of points (vectors) $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^N$ in the N-dimensional vector space $\mathbb{R}^N$.

If these points belong to a lower dimensional manifold, the problem is known under the name of *manifold learning*. If this manifold is linear (or, affine), then Principal Component Analysis (PCA) and Independent Component Analysis (ICA) are the typical methods.

However, if the manifold is not linear, then nonlinear methods are needed. In this respect, Laplacian Eigenmaps, LLE and ISOMAP can be thought of as *nonlinear PCA* methods. They are also known as *nonlinear embeddings*.

Principal Component Analysis

The Problem

Data: We are given a set $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^N$ of n points in $\mathbb{R}^N$.
Goals: We want to find a linear (or affine) subspace V of dimension d that best approximates this set. Specifically, if $P = P_V$ denotes the orthogonal projection onto V, then the goal is to minimize

$$J(V) = \sum_{k=1}^{n} \|x_k - P_V x_k\|_2^2.$$

Once we find this subspace (linear or otherwise), we want to compute either the approximation $P_V x$ or its embedding, i.e., a vector $y \in \mathbb{R}^d$ that encodes $P_V x$.
If V is a linear space (i.e. passes through the origin) then P is a $N \times N$ matrix (linear operator) orthogonal projection that satisfies $P = P^T$, $P^2 = P$, and $Ran(P) = V$.
If V is an affine space (i.e. a linear space shifted by a constant vector), then the "projection" onto the affine space is $T(x) = Px + b$ where b is a constant vector (the "shift").

Principal Component Analysis

Approaches

In the following we discuss four algorithms that perform PCA. The first two algorithms minimize the objective function

$$J(V) = \sum_{k=1}^{n} \|x_k - P_V x_k\|_2^2.$$

The other two algorithms address the problem of affine approximation (as opposed to linear approximation), i.e., approximations of the form $T(x) = Px + b$.
Algorithm 4 is a variation of the Algorithm 3, just as the Algorithm 2 is a variation of the Algorithm 1.

Principal Component Analysis

Algorithm 1

Algorithm (Principal Component Analysis - Alg1: Using Eigenpairs)

Input: Data vectors $\{x_1, \cdots, x_n\} \in \mathbb{R}^N$*; dimension* d.

1. *Compute the matrix*

$$R = \sum_{k=1}^{n} x_k x_k^T$$

2. *Solve the eigenproblems* $Re_k = \sigma_k^2 e_k$, $1 \leq k \leq N$, *order eigenvalues* $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_N^2$ *and normalize the eigenvectors* $\|e_k\|_2 = 1$.

Principal Component Analysis

Algorithm 1 - cont'ed

Algorithm (Principal Component Analysis - Alg1: Using Eigenpairs)

3. *Construct the co-isometry and isometry*

$$U = \begin{bmatrix} e_1^T \\ \vdots \\ e_d^T \end{bmatrix} \;,\quad U^T = \left[\; e_1 \;\middle|\; e_2 \;\middle|\cdots \;\middle|\; e_d \;\right].$$

4. *Project the input data points (or any additional/specific point)*

$$y_1 = Ux_1 \;,\; y_2 = Ux_2 \;,\; \cdots \;,\; y_n = Ux_n,$$

$$\hat{x}_1 = U^T y_1 = U^T U x_1 \;,\; \hat{x}_2 = U^T y_2 = U^T U x_2 \;,\; \cdots \;,\; \hat{x}_n = U^T y_n = U^T U x_n.$$

Output: Lower dimensional data vectors (embeddings) $\{y_1, \cdots, y_n\} \subset \mathbb{R}^d$ and approximation vectors $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\} \subset \mathbb{R}^N$. Furthermore, the optimal value of the objective function is $minJ(V) = \sum_{k=d+1}^{N} \sigma_k^2$.

Principal Component Analysis

Algorithm 2

Algorithm (Principal Component Analysis - Alg2: Using SVD)

Input: *Data vectors* $\{x_1, \cdots, x_n\} \in \mathbb{R}^N$; *dimension* d.

1. *Construct the data matrix* $X \in \mathbb{R}^{N \times n}$

$$X = \left[\begin{array}{c|c|c|c} x_1 & x_2 & \cdots & x_n \end{array} \right]$$

2. *Compute the Singular Value Decomposition (SVD) of* X, $[E, D, F] = svd(X)$ *so that*

$$X = EDF^T \ , \ EE^T = I_N \ , \ FF^T = I_n \ , \ D = diag(\sigma_1, \cdots, \sigma_p)$$

$E \in \mathbb{R}^{N \times N}$, $F \in \mathbb{R}^{n \times n}$ *are orthogonal matrices*, $p = min(n, N)$, $D \in \mathbb{R}^{N \times n}$ *contains singular values* $\sigma_1, \cdots, \sigma_p$ *on the main diagonal and zero elsewhere.*

Principal Component Analysis

Algorithm 2 - cont'ed

Algorithm (Principal Component Analysis - Alg2: Using SVD)

3. *If need be, permute the columns of E, F and diagonal elements of D so that the singular values are sorted monoton decreasing:* $\sigma_1 \geq \cdots \geq \sigma_p$.
4. *Denote by* $e_1, \cdots, e_N$ *the columns of* E. *Construct the co-isometry and isometry*

$$U = \begin{bmatrix} e_1^T \\ \vdots \\ e_d^T \end{bmatrix} \quad , \quad U^T = \left[\begin{array}{c|c|c|c} e_1 & e_2 & \cdots & e_d \end{array} \right].$$

Principal Component Analysis

Algorithm 2 - cont'ed

Algorithm (Principal Component Analysis - Alg2: Using SVD)

5. *Project the input data points (or any additional/specific point)*

$$y_1 = Ux_1 \ , \ y_2 = Ux_2 \ , \ \cdots \ , \ y_n = Ux_n,$$

$$\hat{x}_1 = U^T y_1 = U^T U x_1 \ , \ \hat{x}_2 = U^T y_2 = U^T U x_2 \ , \ \cdots \ , \ \hat{x}_n = U^T y_n = U^T U$$

Output: Lower dimensional data vectors (embeddings) $\{y_1, \cdots, y_n\} \subset \mathbb{R}^d$ *and approximation vectors* $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\} \subset \mathbb{R}^N$*. Furthermore, the optimal value of the objective function is* $\min_{V:dim(V)=d} J(V) = \sum_{k=d+1}^{N} \sigma_k^2$.

Principal Component Analysis

Remarks

1. Notice the two algorithms produce the same output. Indeed, this is due to the fact that $R = XX^T$ and therefore $R = EDD^TE^T$ where $\Lambda = DD^T$ is the diagonal matrix with all N eigenvalues of R, $\lambda_k = \sigma_k^2$, $k \in [N]$.
2. The algorithms 1 and 2 produce information about the orthogonal projection P, subspace V and an implicit hierarchy of approximations.
The orthogonal projection is given by $P = \sum_{k=1}^{d} e_k e_k^T$ and the optimal subspace is $V = Ran(P)$.
3. Residual: These theorems provide exact estimates of the residual. This estimate provides with the *ratio of explained variance* as:

$$\rho = \frac{\sum_{k=1}^{d} \sigma_k^2}{\sum_{k=1}^{N} \sigma_k^2} = 1 - \frac{\sum_{k=d+1}^{N} \sigma_k^2}{\sum_{k=1}^{N} \sigma_k^2}$$

One can utilize this ratio as a criterion for choosing d. For instance the smallest d so that $\rho \geq 0.9 = 90\%$.
4. The stochastic equivalent to this algorithm: The Karhunen-Loève Theorem (see wikipedia: https://en.wikipedia.org/wiki/Kosambi-Karhunen-Loeve_theorem).

Principal Component Analysis

Derivation

Here is the derivation in the case of approximation by linear spaces. As we have seen in the lectures on least-squares approximation by linear models, the approximation vectors are given by $y_k = Px_k$ for the orthogonal projection P onto subspace V, yet to-be-determined. Expand the criterion $J(V)$:

$$J(V) = \sum_{k=1}^{n} \|x_k\|^2 - \sum_{k=1}^{n} \langle Px_k, x_k \rangle = \sum_{k=1}^{n} \|x_k\|^2 - trace(PR)$$

where $R = \sum_{k=1}^{n} x_k x_k^T$. It follows the orthogonal projection that minimizes $J(V)$ maximizes also $trace(PR)$ subject to $P = P^T$, $P^2 = P$ and $trace(P) = d$. Recall that all symmetric matrices (such as R in the Algorithm 1) diagonalize by orthogonal matrices. That means step 2 of the algorithm is guaranteed to produce a complete set of eigenpairs. Notice also that R is positive semidefinite, hence all eigenvalues are non-negative. Therefore $R = \sum_{k=1}^{N} \sigma_k^2 e_k e_k^T$.

Principal Component Analysis

Derivation - cont'ed

Now:

$$trace(PR) = \sum_{k=1}^{N} \sigma_k^2 trace(Pe_k e_k^T) = \sum_{k=1}^{N} \sigma_k^2 \|Pe_k\|_2^2$$

Our problem is to maximize $\sum_{k=1}^{N} \sigma_k^2 \|Pe_k\|_2^2$ over P, subject to P an orthogonal projection of rank d. For orthogonal projections, rank equals the dimension of its range equals its trace. Hence $trace(P) = d$. Additionall, any eigenvalue of P is either 0 or 1. These two conditions imply:

$$\|Pe_k\|_2 \leq 1 \quad , \quad \sum_{k=1}^{N} \|Pe_k\|_2^2 = trace(P) = d.$$

Letting $w_k = \|Pe_k\|_2^2$, the optimization becomes:

$$\begin{array}{c} \text{maximize} \\ 0 \leq w_1, \cdots, w_N \leq 1 \\ w_1 + \cdots + w_N = d \end{array} \sum_{k=1}^{N} w_k \sigma_k^2.$$

Principal Component Analysis

Derivation - cont'ed

Using a "water-filling" principle[1], the optimal solution puts most of the weight on the largest eigenvalues: that is, $w_1 = \cdots = w_d = 1$ and $w_{d+1} = \cdots = w_N = 0$. It follows the optimal P is given by the orthogonal projection onto the top d eigenvectors, hence the algorithm 1.

Algorithm 2 follows once we observe that the columns of matrix E are exactly the eigenvectors $e_1, \cdots, e_N$ of matrix R from Algorithm 1:

$$R = XX^T = EDF^T(EDF^T)^T = EDD^TE^T.$$

[1] The "water filling algorithm": see https://en.wikipedia.org/wiki/Water_filling_algorithm

Principal Component Analysis

The Affine Case

Consider now the case when data $\{x_1, \cdots, x_n\} \subset \mathbb{R}^N$ is approximated by an *affine* space, that is $T(x) = Fy + b$, for some $F \in \mathbb{R}^{N\times d}$ and $b \in \mathbb{R}^N$. Here $0 < d \leq N$. The desired solution must solve the following optimization problem:

$$\underset{F \in \mathbb{R}^{N\times d},\, b \in \mathbb{R}^N}{\text{minimize}} \quad \sum_{k=1}^{n} \min_{y \in \mathbb{R}^d} \|x_k - Fy - b\|_2^2.$$

As we have seen earlier, given F and b, the inner optimization has solution: $\hat{y} = (F^T F)^{-1} F^T (x_k - b)$ and inner norm term becomes $\|(1-P)(x_k - b)\|_2^2$ where $P = F(F^T F)^{-1} F^T$ is the orthogonal projection onto $Ran(F)$. This formula holds under the assumption that F is full rank, i.e. $rank(F) = d$. In general, d is chosen to be at most the smaller of $\dim\{x_1, \cdots, x_n\}$ of N, in which case F achieves its full rank.

Principal Component Analysis

Algorithm 3: The Affine Case

Algorithm (Principal Component Analysis - Alg3: Affine case using eigenpairs)

Input: *Data vectors* $\{x_1, \cdots, x_n\} \in \mathbb{R}^N$; *dimension* d.

0. *Compute the average data vector* $\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$.
1. *Compute the matrix*

$$R = \sum_{k=1}^{n} (x_k - \bar{x})(x_k - \bar{x})^T$$

2. *Solve the eigenproblems* $Re_k = \sigma_k^2 e_k$, $1 \leq k \leq N$, *order eigenvalues* $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_N^2$ *and normalize the eigenvectors* $\|e_k\|_2 = 1$.

Principal Component Analysis

Algorithm 3 - cont'd

Algorithm (Principal Component Analysis - Alg3: Affine case using eigenpairs)

3. *Constructe the following:*

$$U = \begin{bmatrix} e_1^T \\ \vdots \\ e_d^T \end{bmatrix} \ , \ U^T = \left[\ e_1 \ \middle| \ e_2 \ \middle| \cdots \ \middle| \ e_d \ \right] \ P = U^T U \ , \ b = (I - P)\bar{x}.$$

4. *Project the input data points (or any additional/specific point)*

$$y_1 = Ux_1 \ , \ \cdots \ , \ y_n = Ux_n,$$

$$\hat{x}_1 = U^T y_1 + b = Px_1 + (I-P)\bar{x} \ , \ \cdots \ , \ \hat{x}_n = PU^T y_n + b = Px_n + (I-P)\bar{x}.$$

Output: Embedding $\{y_1, \cdots, y_n\} \subset \mathbb{R}^d$ *and approx.* $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\} \subset \mathbb{R}^N$*.*
Furthermore, the optimal value of the objective function is $\sum_{k=d+1}^{N} \sigma_k^2$*.*

Principal Component Analysis

Algorithm 4: The Affine Case

Algorithm (Principal Component Analysis - Alg4: Affine case using SVD)

Input: *Data vectors* $\{x_1, \cdots, x_n\} \in \mathbb{R}^N$; *dimension* d.

0. *Compute the average data vector* $\bar{x} = \frac{1}{n}\sum_{k=1}^{n} x_k$.
1. *Construct the centered data matrix* $X \in \mathbb{R}^{N \times n}$

$$X = \left[\begin{array}{c|c|c|c} x_1 - \bar{x} & x_2 - \bar{x} & \cdots & x_n - \bar{x} \end{array} \right]$$

2. *Compute the Singular Value Decomposition (SVD) of* X, $[E, D, F] = svd(X)$ *so that*

$$X = EDF^T \ , \ EE^T = I_N \ , \ FF^T = I_n \ , \ D = diag(\sigma_1, \cdots, \sigma_p)$$

$E \in \mathbb{R}^{N \times N}$, $F \in \mathbb{R}^{n \times n}$ *are orthogonal matrices,* $p = min(n, N)$,

Principal Component Analysis

Algorithm 4 - cont'ed

Algorithm (Principal Component Analysis - Alg4: Affine case using SVD)

3. *If need be, permute the columns of* E, F *and diagonal elements of* D *so that the singular values are sorted monoton decreasing:* $\sigma_1 \geq \cdots \geq \sigma_p$.
4. *Denote by* $e_1, \cdots, e_N$ *the columns of* E. *Construct the following*

$$U = \begin{bmatrix} e_1^T \\ \vdots \\ e_d^T \end{bmatrix} \quad , \quad U^T = \left[\begin{array}{c|c|c|c} e_1 & e_2 & \cdots & e_d \end{array} \right]$$

$$P = U^T U \quad , \quad b = (I - P)\bar{x}.$$

Principal Component Analysis

Algorithm 4 - cont'ed

Algorithm (Principal Component Analysis - Alg4: Affine case using SVD)

5. *Project the input data points (or any additional/specific point)*

$$y_1 = Ux_1 \ , \ \cdots \ , \ y_n = Ux_n,$$

$$\hat{x}_1 = U^T y_1 + b = Px_1 + (I - P)\bar{x} \ , \ \cdots \ , \ \hat{x}_n = U^T y_n + b = Px_n + (I - P)\bar{x}.$$

<u>Output</u>: Lower dimensional data vectors (embeddings) $\{y_1, \cdots, y_n\} \subset \mathbb{R}^d$ *and approximation vectors* $\{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n\} \subset \mathbb{R}^N$*. Furthermore, the optimal value of the objective function is* $\min_{V:dim(V)=d} J(V) = \sum_{k=d+1}^{N} \sigma_k^2$.

Principal Component Analysis

Derivation of Algorithms 3 and 4 in the Affine Case

Notice that, after optimization of the inner term, the objective function, parametrized by P and b, becomes:

$$J(P,b) = trace\left\{(I-P)\left(\sum_{k=1}^{N}(x_k-b)(x_k-b)^T\right)(I-P)\right\} = \dots =$$

$$= trace\{(I-P)R_0(I-P)\} - 2n\langle (I-P)b,(I-P)\bar{x}\rangle + n\|(I-P)b\|_2^2$$

where $R_0 = \sum_{k=1}^{n} x_k x_k^T$. Fixing P, the optimization over b seeks to minimize:

$$\min_b \|(I-P)b\|_2^2 - 2\langle (I-P)b,(I-P)\bar{x}\rangle$$

Cauchy-Schwarz inequality implies $\langle (I-P)b,(I-P)\bar{x}\rangle \leq \|(I-P)b\|\|(I-P)\bar{x}\|$, from where the minimum is achieved for b so that, firstly $(I-P)b||(I-P)\bar{x}$, and secondly (by optimization over norm of b) that $(I-P)b = (I-P)\bar{x}$. Choose $b = (I-P)\bar{x}$. Finally, note $(I-P)(x_k-b) = (I-P)(x_k-\bar{x})$ which reduces the minimization of $J(P,(I-P)\bar{x})$ to the linear case with R as in Algorithm 3.

Independent Component Analysis

Approach

Model (Setup): $x = As$, where A is an unknown invertible $N \times N$ matrix, and $s \in \mathbb{R}^N$ is a random vector of *independent* components.

Data: We are given a set of measurement $\{x_1, x_2, \cdots, x_n\} \subset \mathbb{R}^N$ of n points in $\mathbb{R}^N$ of the model $x_k = As_k$, where each $\{s_1, \cdots, s_n\}$ is drawn from the same distribution $p_s(s)$ of N-vectors with independent components.

Goal: We want to estimate the invertible matrix A and the (source) signals $\{s_1, \cdots, s_n\}$. Specifically, we want a square matrix W such that Wx has independent components.

Principle: Perform PCA first so the decorrelated signals have unit variance. Then find an orthogonal matrix (that is guaranteed to preserve decorrelation) that creates statistical independence as much as possible.

Caveat: Two inherent ambiguities: (1) Permutation: If W is a solution to the unmixing problem so is ΠW, where Π is a permutation matrix; (2) Scaling: If W is a solution to unmixing problem, so is DW where D is a diagonal matrix.

Independent Component Analysis

Algorithm

Algorithm (Independent Component Analysis)

Input: Data vectors $\{x_1, \cdots, x_n\} \in \mathbb{R}^N$.

1. *Compute the sample mean* $b = \frac{1}{n}\sum_{k=1}^{n} x_k$*, and sample covariance matrix* $R = \frac{1}{n}\sum_{k=1}^{n}(x_k - b)(x_k - b)^T$.
2. *Solve the eigenproblem* $RE = E\Lambda$*, where* E *is the* $N \times N$ *orthogonal matrix whose columns are eigenvectors, and* Λ *is the diagonal matrix of eigenvalues.*
3. *Compute* $F = R^{-1/2} := E\Lambda^{-1/2}E^T$ *and apply it on data,* $z_k = F(x_k - b)$, $1 \leq k \leq n$.
4. *Compute the orthogonal matrix* Q *using the JADE algorithm below.*
5. *Apply* Q *on whitened data,* $\hat{s}_k = Qz_k$, $1 \leq k \leq n$*. Compute* $W = QF$.

Output: Matrix W *and independent vectors* $\{\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_n\}$.

Independent Component Analysis – Cont.

Joint Approximate Diagonalization of Eigenmatrices (JADE)

Algorithm (Cardoso's 4th Order Cumulants Algorithm'92)

Input: Whitened data vectors $\{z_1, \cdots, z_n\} \in \mathbb{R}^N$.

1. *Compute the sample* 4^{th} *order symmetric cumulant tensor*

$$F_{ijkl} = \frac{1}{N}\sum_{t=1}^{N} z_t(i)z_t(j)z_t(k)z_t(l) - \delta_{i,j}\delta_{k,l} - \delta_{i,k}\delta_{j,l} - \delta_{i,l}\delta_{j,k}.$$

2. *Compute* N *eigenmatrices* $M_{i,j}$*, so that* $F(M_{i,j}) = \lambda_{i,j}M_{i,j}$.
3. *Maximize the criterion*

$$J_{JADE}(Q) = \sum_{i,j} |\lambda_{i,j}|^2 \|diag(QM_{i,j}Q^T)\|_2^2$$

over orthogonal matrices Q *by performing successive rotations marching through all pairs* (a, b) *of distinct indices in* $\{1, \cdots, N\}$.

Independent Component Analysis – Cont.

Wang-Amari Natural Stochastic Gradient Algorithm of Bell-Sejnowski MaxEntropy

Algorithm (Wang-Amari'97; Bell-Sejnowski'95)

Input: Sphered data vectors $\{z_1, \cdots, z_n\} \in \mathbb{R}^N$*; Cumulative distribution functions* g_k *of each component of* s*; Learning rate* η*.*

```
1. Initialize $W^{(0)} = F$.
2. Repeat until convergence, or until maximum number of steps reached:
    1. Draw a data vector $z$ randomly from data vectors, and compute
       $W^{(t+1)} = W^{(t)} + \eta(I + (1 - 2g(z))z^T)W^{(t)}$.
    2. increment $t \leftarrow t + 1$.
```

Output: Unmixing $N \times N$ *matrix* $W = W^{(T)}$*.*

Independent Component Analysis

Derivation