Anomaly Detection

Dimensionality reduction: PCA, robust PCA

Paul Irofti
Andrei Pătrașcu
Cristian Rusu

Computer Science Department

Faculty of Mathematics and Computer Science

University of Bucharest

Email: first.last@fmi.unibuc.ro

Outline

- eigenvalue and singular value decomposition
- Principal Component Analysis
- Robust PCA
- Matrix Factorization

The course references are Aggarwal 2017, Ch.3 with papers for Robust PCA by Candès et al. 2011 and Netrapalli et al. 2014.

For a thorough recap of eigen and singular values see Golub and Van Loan 2013.

Preliminaries

Eigenvalues and Eigenvalue Decomposition (EVD)

Given square matrix $A \in \mathbb{R}^{n \times n}$ then its eigenvalues λ and associated eigenvectors $\boldsymbol{v}$ follow:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{1}$$

which can be obtained through $(\boldsymbol{A} - \lambda \boldsymbol{I}_n)\boldsymbol{v} = 0$ or the factorization of the polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}_n) = 0$

Eigenvalues and Eigenvalue Decomposition (EVD)

Given square matrix $A \in \mathbb{R}^{n \times n}$ then its eigenvalues λ and associated eigenvectors $\boldsymbol{v}$ follow:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{1}$$

which can be obtained through $(\boldsymbol{A} - \lambda \boldsymbol{I}_n)\boldsymbol{v} = 0$ or the factorization of the polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}_n) = 0$

There are n eigenvalues and eigenvectors, thus we can write the eigenvalue decomposition (EVD):

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda} \iff \boldsymbol{\Lambda} = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} \tag{2}$$

Eigenvalues and Eigenvalue Decomposition (EVD)

Given square matrix $A \in \mathbb{R}^{n \times n}$ then its eigenvalues λ and associated eigenvectors $\boldsymbol{v}$ follow:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{1}$$

which can be obtained through $(\boldsymbol{A} - \lambda \boldsymbol{I}_n)\boldsymbol{v} = 0$ or the factorization of the polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}_n) = 0$

There are n eigenvalues and eigenvectors, thus we can write the eigenvalue decomposition (EVD):

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\Lambda \iff \Lambda = \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} \tag{2}$$

Remark: For symmetric matrices $\boldsymbol{A}^\top = \boldsymbol{A}$ we have $V^{-1} = V^\top$ such that $\boldsymbol{A} = \boldsymbol{V}\Lambda\boldsymbol{V}^\top$.

Singular Values and Singular Value Decomposition (SVD)

Given rectangular matrix $A \in \mathbb{R}^{n \times m}$ the singular values σ, the associated left-hand side singular vectors $\boldsymbol{u}$, and associated right-hand side singular vectors $\boldsymbol{v}$

$$\boldsymbol{A}\boldsymbol{v} = \sigma \boldsymbol{v} \quad ; \quad \boldsymbol{A}^\top \boldsymbol{u} = \sigma \boldsymbol{u} \tag{3}$$

Singular Values and Singular Value Decomposition (SVD)

Given rectangular matrix $A \in \mathbb{R}^{n \times m}$ the singular values σ, the associated left-hand side singular vectors $\boldsymbol{u}$, and associated right-hand side singular vectors $\boldsymbol{v}$

$$\boldsymbol{A}\boldsymbol{v} = \sigma \boldsymbol{v} \quad ; \quad \boldsymbol{A}^\top \boldsymbol{u} = \sigma \boldsymbol{u} \tag{3}$$

There are $\min(n, m)$ singular values, n left singular vectors, and m right singular vectors, thus we can write the singular value decomposition (SVD):

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{\min\{n,m\}} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{4}$$

Singular Values and Singular Value Decomposition (SVD)

Given rectangular matrix $A \in \mathbb{R}^{n \times m}$ the singular values σ, the associated left-hand side singular vectors $\boldsymbol{u}$, and associated right-hand side singular vectors $\boldsymbol{v}$

$$\boldsymbol{A}\boldsymbol{v} = \sigma \boldsymbol{v} \quad ; \quad \boldsymbol{A}^\top \boldsymbol{u} = \sigma \boldsymbol{u} \tag{3}$$

There are $\min(n, m)$ singular values, n left singular vectors, and m right singular vectors, thus we can write the singular value decomposition (SVD):

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \sum_{i=1}^{\min\{n,m\}} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top \tag{4}$$

Theorem: The optimal low-rank matrix $\boldsymbol{L} \in \mathbb{R}^{n \times m}$ with rank k that approximates $\boldsymbol{A}$ is $\sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^\top$.

Least Squares (LS)

Given data $\boldsymbol{X} \in \mathbb{R}^{N \times d}$, where d is the data dimension and N the number of samples, the least-squares problem solves the following:

$$\min_{\beta} \|\mathbf{X}\beta - \mathbf{y}\|_F^2 \tag{5}$$

- when $N = d$ we have $\beta^\star = \mathbf{X}^{-1}\mathbf{y}$
- when $N > d$ we have $\beta^\star = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$
- when $N < d$ we have $\beta^\star = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{y}$

See Lecture 2 for more details.

LS: line fit

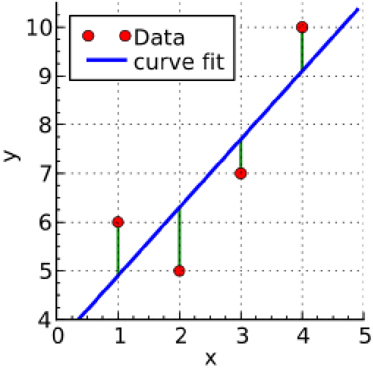


Figure: LS fits 2D points on a line

Source: https://en.wikipedia.org/wiki/Linear_least_squares

LS: projection

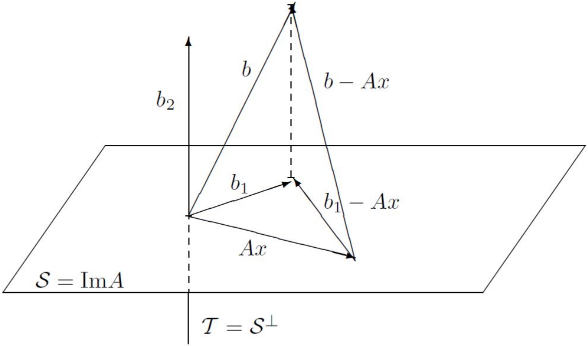


Figure: LS projects vectors on ImA

LS: remarks

Least-squares properties:

- finds $d - 1$ subspace or hyperplane
- the hyperplane is an optimum fit to data
- anomaly score: based on length on orthogonal direction

LS: remarks

Least-squares properties:

- finds $d-1$ subspace or hyperplane
- the hyperplane is an optimum fit to data
- anomaly score: based on length on orthogonal direction

Generalization:

- what is the $k < d$ subspace or hyperplane?
- what is the anomaly score then?
- what is an optimum fit on any k-dimensional subspace?

Principal Component Analysis

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Properties:

- the covariance matrix is symmetric and positive definite

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Properties:

- the covariance matrix is symmetric and positive definite
- the EVD of $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Properties:

- the covariance matrix is symmetric and positive definite
- the EVD of $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$
- $\boldsymbol{\Delta}$ is diagonal and contains the eigenvalues between λ_{max} and λ_{min}

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Properties:

- the covariance matrix is symmetric and positive definite
- the EVD of $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$
- $\boldsymbol{\Delta}$ is diagonal and contains the eigenvalues between λ_{max} and λ_{min}
- $\boldsymbol{P} \in \mathbb{R}^{d \times d}$ represents the orthonormal eigenvectors of the covariance corresponding to $\boldsymbol{\Delta}$

Principal Component Analysis (PCA)

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} \tag{6}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$ where element Σ_{ij} is the covariance between data dimension i and j.

Properties:

- the covariance matrix is symmetric and positive definite
- the EVD of $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$
- $\boldsymbol{\Delta}$ is diagonal and contains the eigenvalues between λ_{max} and λ_{min}
- $\boldsymbol{P} \in \mathbb{R}^{d \times d}$ represents the orthonormal eigenvectors of the covariance corresponding to $\boldsymbol{\Delta}$
- the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

This implies that:

- the eigenvectors subspace corresponding to the largest $d - 1$ eigenvalues provides a good data approximation

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

This implies that:

- the eigenvectors subspace corresponding to the largest $d - 1$ eigenvalues provides a good data approximation
- what about $k = d - 2$ or any $k \in [d - 1] = \{1, \ldots, d - 1\}$?

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

This implies that:

- the eigenvectors subspace corresponding to the largest $d - 1$ eigenvalues provides a good data approximation
- what about $k = d - 2$ or any $k \in [d - 1] = \{1, \ldots, d - 1\}$?
- the subspace will be generated by the largest k eigenvectors such that the residual is minimal

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

This implies that:

- the eigenvectors subspace corresponding to the largest $d - 1$ eigenvalues provides a good data approximation
- what about $k = d - 2$ or any $k \in [d - 1] = \{1, \ldots, d - 1\}$?
- the subspace will be generated by the largest k eigenvectors such that the residual is minimal
- anomalies are the data whose error is high in this new subspace

PCA as Generalized LS

PCA starts from the covariance matrix of the mean-centered data matrix $\boldsymbol{X} \in \mathbb{R}^{N \times d}$

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top \tag{7}$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Remark: the normal hyperplane to $\boldsymbol{p}_{min} \in \boldsymbol{P}$ is the LS-hyperplane of dimension $k = d - 1$.

This implies that:

- the eigenvectors subspace corresponding to the largest $d - 1$ eigenvalues provides a good data approximation
- what about $k = d - 2$ or any $k \in [d - 1] = \{1, \ldots, d - 1\}$?
- the subspace will be generated by the largest k eigenvectors such that the residual is minimal
- anomalies are the data whose error is high in this new subspace
- anomalies have a large normal component

Example: Principal Eigenvectors

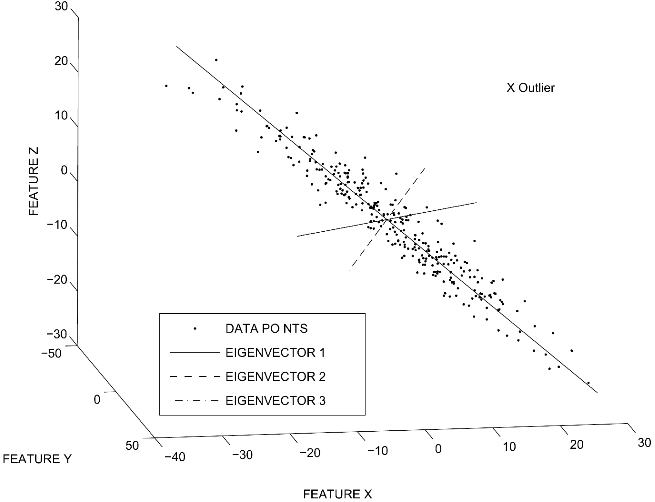


Figure: Distribution along first $k = 3$ eigenvectors (Aggarwal 2017)

PCA as Generalized LS

Remark: the subspace will be generated by the largest k eigenvectors such that the residual is minimal.

PCA as Generalized LS

Remark: the subspace will be generated by the largest k eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the k orthonormal eigenvectors

PCA as Generalized LS

Remark: the subspace will be generated by the largest k eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the k orthonormal eigenvectors
- there is no covariance in this new subspace as the eigenvectors are orthogonal

PCA as Generalized LS

Remark: the subspace will be generated by the largest k eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the k orthonormal eigenvectors
- there is no covariance in this new subspace as the eigenvectors are orthogonal
- the variance along each axis is the eigenvalue

PCA as Generalized LS

Remark: the subspace will be generated by the largest k eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the k orthonormal eigenvectors
- there is no covariance in this new subspace as the eigenvectors are orthogonal
- the variance along each axis is the eigenvalue
- a small eigenvalue implies a low variance

PCA as Generalized LS

Remark: the subspace will be generated by the largest *k* eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the *k* orthonormal eigenvectors
- there is no covariance in this new subspace as the eigenvectors are orthogonal
- the variance along each axis is the eigenvalue
- a small eigenvalue implies a low variance
- can we cancel axis with small eigenvalues? E.g. $\lambda < 10^{-3}$

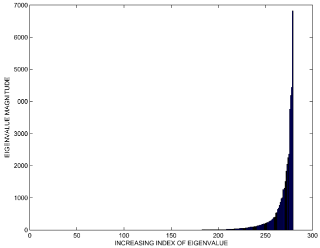
PCA as Generalized LS

Remark: the subspace will be generated by the largest *k* eigenvectors such that the residual is minimal.

Implications:

- the new axis of the data representation are the *k* orthonormal eigenvectors
- there is no covariance in this new subspace as the eigenvectors are orthogonal
- the variance along each axis is the eigenvalue
- a small eigenvalue implies a low variance
- can we cancel axis with small eigenvalues? E.g. $\lambda < 10^{-3}$
- not if we expect anomalies to have higher variance among low variance axis

Example: Eigen Histogram



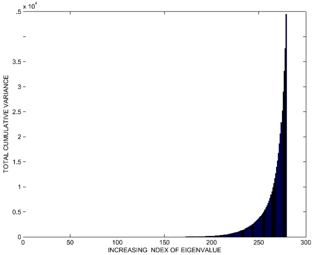


Figure: Eigenvalues magnitude and variance (Aggarwal 2017)

Example: Eigen Histogram Trimmed

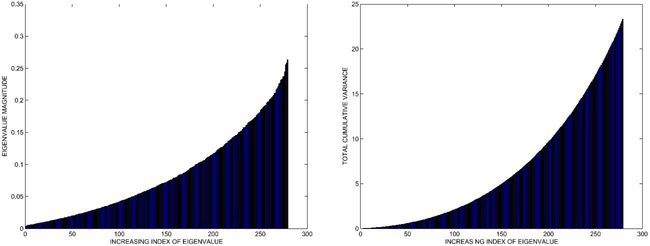


Figure: Eigenvalues magnitude and variance after trimming (Aggarwal 2017)

PCA space

PCA starts from the EVD of the covariance matrix

$$\mathbf{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$$

such that $\mathbf{\Sigma} \in R^{d \times d}$.

PCA space

PCA starts from the EVD of the covariance matrix

$$\mathbf{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$$

such that $\mathbf{\Sigma} \in R^{d \times d}$.

Then the transformed data space becomes:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P} \tag{8}$$

where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

PCA space

PCA starts from the EVD of the covariance matrix

$$\mathbf{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P \Delta P}^\top$$

such that $\mathbf{\Sigma} \in R^{d \times d}$.

Then the transformed data space becomes:

$$\boldsymbol{X}' = \boldsymbol{XP} \tag{8}$$

where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

Using the entire space $k = d$ has the following advantage:

PCA space

PCA starts from the EVD of the covariance matrix

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$$

such that $\boldsymbol{\Sigma} \in R^{d \times d}$.

Then the transformed data space becomes:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P} \tag{8}$$

where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

Using the entire space $k = d$ has the following advantage:

- take small eigenvector $\boldsymbol{p}_j$

PCA space

PCA starts from the EVD of the covariance matrix

$$\mathbf{\Sigma} = \frac{\boldsymbol{X}^T\boldsymbol{X}}{N} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$$

such that $\mathbf{\Sigma} \in R^{d \times d}$.

Then the transformed data space becomes:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P} \tag{8}$$

where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

Using the entire space $k = d$ has the following advantage:

- take small eigenvector $\boldsymbol{p}_j$
- we know that the entries $x'_{\ell j}$ do not vary much as λ_j is small

PCA space

PCA starts from the EVD of the covariance matrix

$$\mathbf{\Sigma} = \frac{\boldsymbol{X}^T\boldsymbol{X}}{N} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$$

such that $\mathbf{\Sigma} \in R^{d\times d}$.

Then the transformed data space becomes:

$$\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P} \tag{8}$$

where $\boldsymbol{X}' \in \mathbb{R}^{N\times d}$.

Using the entire space $k = d$ has the following advantage:

- take small eigenvector $\boldsymbol{p}_j$
- we know that the entries $x'_{\ell j}$ do not vary much as λ_j is small
- **outlier:** if x'_{ij} has a large deviation compared to other $x'_{\ell j}$ entries

PCA subspace

PCA starts from the EVD of $\mathbf{\Sigma} = \frac{\boldsymbol{X}^T \boldsymbol{X}}{N} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$, then the transformed data space becomes $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$ where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

PCA subspace

PCA starts from the EVD of $\mathbf{\Sigma} = \frac{\boldsymbol{X}^T\boldsymbol{X}}{\boldsymbol{N}} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$, then the transformed data space becomes $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$ where $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$.

The approximation through trimming the smallest $d - k$ eigenvectors $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}_k \in \mathbb{R}^{N \times k}$

$$\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{P}_{\boldsymbol{k}}\| \tag{9}$$

will contain in each x'_{ij} with $j \in [k]$

PCA subspace

PCA starts from the EVD of $\boldsymbol{\Sigma} = \frac{\boldsymbol{X}^T\boldsymbol{X}}{N} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$, then the transformed data space becomes $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$ where $\boldsymbol{X}' \in \mathbb{R}^{N\times d}$.

The approximation through trimming the smallest $d-k$ eigenvectors $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}_k \in \mathbb{R}^{N\times k}$

$$\|\boldsymbol{X} - \boldsymbol{X}\boldsymbol{P}_{\boldsymbol{k}}\| \tag{9}$$

will contain in each x'_{ij} with $j \in [k]$

Hard outlier score: the residuals representing the distance to the rank-k hyperplane described by XP_k.

Soft outlier score: normalize the residuals according to their corresponding variance along the $d-k$ distances.

PCA: Soft Score

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

PCA: Soft Score

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Decompose the sum of squares of the $d - k$ distances and normalize by their corresponding eigenvalue:

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=k+1}^{d} \frac{\left\| x_{\ell j} - \boldsymbol{x}_\ell^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{10}$$

PCA: Soft Score

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Decompose the sum of squares of the $d - k$ distances and normalize by their corresponding eigenvalue:

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=k+1}^{d} \frac{\left\| x_{\ell j} - \boldsymbol{x}_\ell^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{10}$$

Result: also reward large deviation along small variance.

PCA: Soft Score

Soft outlier score: normalize the residuals according to their corresponding variance along the $d-k$ distances.

Decompose the sum of squares of the $d-k$ distances and normalize by their corresponding eigenvalue:

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=k+1}^{d} \frac{\left\| x_{\ell j} - \boldsymbol{x}_\ell^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{10}$$

Result: also reward large deviation along small variance.

Remark: both scores focus on representing data in a low-dimensional space which induces parameter k: selecting the dimensionality

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

Algorithm:

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

Algorithm:

1. EVD: $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

Algorithm:

1. EVD: $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$
2. Transform: $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

Algorithm:

1. EVD: $\boldsymbol{\Sigma} = \boldsymbol{P\Delta P}^\top$
2. Transform: $\boldsymbol{X}' = \boldsymbol{XP}$
3. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Delta^{-1}$

PCA: Mahalanobis Connection

Soft outlier score: normalize the residuals according to their corresponding variance along the $d - k$ distances.

Mahalanobis performs the normalization across all d directions

$$\text{Score}(\boldsymbol{x}_\ell) = \sum_{j=1}^{d} \frac{\left\| (\boldsymbol{x}_\ell - \boldsymbol{\mu})^\top \boldsymbol{p}_j \right\|^2}{\lambda_j} \tag{11}$$

where $\boldsymbol{\mu} \in \mathbb{R}^d$ is the data centroid (the mean vector along the data dimension).

Algorithm:

1. EVD: $\boldsymbol{\Sigma} = \boldsymbol{P} \boldsymbol{\Delta} \boldsymbol{P}^\top$
2. Transform: $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{P}$
3. Normalize: $\boldsymbol{X}' = \boldsymbol{X}' \Delta^{-1}$
4. Anomaly score: $\text{Score}(\boldsymbol{x}'_\ell) \quad \forall \ell \in [N]$

Robust PCA

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

Robust PCA through iterative pruning:

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

Robust PCA through iterative pruning:

1. PCA: compute $\mathbf{\Sigma} = \boldsymbol{P}\mathbf{\Delta}\boldsymbol{P}^\top$
2. Anomaly score: compute associated scores $\text{Score}(\boldsymbol{x}_{\boldsymbol{\ell}}) \ \ \forall \ell \in [N]$

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

Robust PCA through iterative pruning:

1. PCA: compute $\boldsymbol{\Sigma} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$
2. Anomaly score: compute associated scores $\text{Score}(\boldsymbol{x}_{\boldsymbol{\ell}}) \ \ \forall \ell \in [N]$
3. Prune: remove obvious outliers from the dataset

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

Robust PCA through iterative pruning:

1. PCA: compute $\boldsymbol{\Sigma} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$
2. Anomaly score: compute associated scores $\text{Score}(\boldsymbol{x}_\ell) \ \ \forall \ell \in [N]$
3. Prune: remove obvious outliers from the dataset
4. Reconstruct: compute new covariance matrix

PCA Perturbations

The lack of data covariance in the eigenspace adds robustness to noise.

Robust PCA through iterative pruning:

1. PCA: compute $\boldsymbol{\Sigma} = \boldsymbol{P}\boldsymbol{\Delta}\boldsymbol{P}^\top$
2. Anomaly score: compute associated scores $\text{Score}(\boldsymbol{x}_\ell) \;\; \forall \ell \in [N]$
3. Prune: remove obvious outliers from the dataset
4. Reconstruct: compute new covariance matrix
5. Goto step 1

Example: Outlier Perturbation

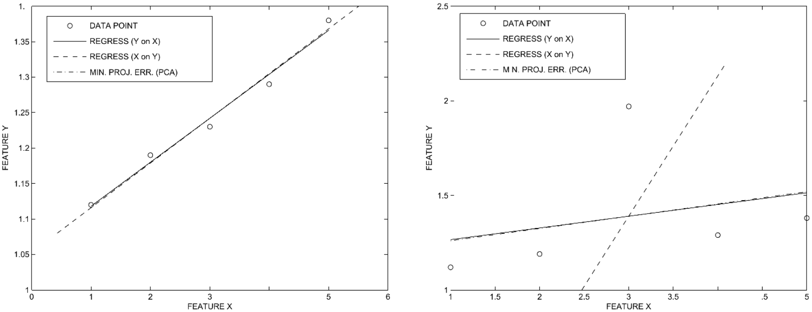


Figure: Sensitivity to outliers (Aggarwal 2017)

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

Score through cross-validation:

- split into m-folds

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

Score through cross-validation:

- split into m-folds
- perform PCA on $m - 1$ folds

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

Score through cross-validation:

- split into m-folds
- perform PCA on $m - 1$ folds
- score the data in the m^{th}-fold

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

Score through cross-validation:

- split into m-folds
- perform PCA on $m - 1$ folds
- score the data in the m^{th}-fold
- repeat for each fold

PCA Strengthening

Normalization: original dimensions scales can very widely – normalize to unit variance.

Regularization:

- zero variance among some dimensions implies $\lambda = 0$
- regularize the covariance matrix $\mathbf{\Sigma} + \alpha I_d$ with $\alpha > 0$
- shifts all eigenvalues by the constant value α
- equivalent to adding noise with variance α

Score through cross-validation:

- split into m-folds
- perform PCA on $m - 1$ folds
- score the data in the m^{th}-fold
- repeat for each fold
- alternative: use sub-sampling

Robust PCA

Treat measurement matrix as the super-position of a low-rank matrix with a sparse noise matrix $X = L_0 + S_0$, then recovering L_0 and S_0 involves solving the following optimization problem

$$\arg\min_{L,S} \rho(L) + \lambda \|S\|_0 \quad \text{s.t.} \quad \|X - L - S\|_F^2 = 0 \tag{12}$$

where $\rho(L)$ is the rank function and $\|\cdot\|_0$ is the ℓ_0-norm counting the number of non-zeros.

Robust PCA

Treat measurement matrix as the super-position of a low-rank matrix with a sparse noise matrix $X = L_0 + S_0$, then recovering L_0 and S_0 involves solving the following optimization problem

$$\arg\min_{L,S} \rho(L) + \lambda \|S\|_0 \quad \text{s.t.} \quad \|X - L - S\|_F^2 = 0 \tag{12}$$

where $\rho(L)$ is the rank function and $\|\cdot\|_0$ is the ℓ_0-norm counting the number of non-zeros.

Candès et al. 2011 show that the convex relaxation of the above can recover L_0 and S_0 under mild assumpitons

$$\arg\min_{L,S} \|L\|_\star + \lambda \|S\|_1 \quad \text{s.t.} \quad \|X - L - S\|_F^2 = 0 \tag{13}$$

where $\|\cdot\|_\star$ is the nuclear norm summing the singular values.

Escalator Example: PCA versus Robust PCA

Figure: Background separation: truth, PCA and two RPCA implementations (Netrapalli et al. 2014)

Restaurant Example: PCA versus Robust PCA

Figure: Background separation: truth, PCA and two RPCA implementations (Netrapalli et al. 2014)

Nonlinear PCA

PCA: sample space versus feature space

What if we used $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ instead of $\boldsymbol{\Sigma}$?

PCA: sample space versus feature space

What if we used $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ instead of $\boldsymbol{\Sigma}$?

The EVD of $\boldsymbol{S}$ becomes:

$$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \tag{14}$$

where $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ are the orthonormal eigenvectors such that only the first d correspond to non-zero eigenvalues.

PCA: sample space versus feature space

What if we used $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ instead of $\boldsymbol{\Sigma}$?

The EVD of $\boldsymbol{S}$ becomes:

$$\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top = \boldsymbol{Q}\Lambda^2\boldsymbol{Q}^\top \tag{14}$$

where $\boldsymbol{Q} \in \mathbb{R}^{N \times N}$ are the orthonormal eigenvectors such that only the first d correspond to non-zero eigenvalues.

The transform in the sample space is:

$$\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\Lambda)_d \tag{15}$$

where we can easily see that $[\boldsymbol{X}' \;\; \boldsymbol{O}] = \boldsymbol{X}\boldsymbol{Q}\Lambda = [\boldsymbol{X}(\boldsymbol{Q}\Lambda)_d \;\; \boldsymbol{O}]$.

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

1. Similarity: Build $N \times N$ similarity matrix $\boldsymbol{S}$ using kernel function κ

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

1. Similarity: Build $N \times N$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

1. Similarity: Build $N \times N$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

1. Similarity: Build $N \times N$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$

PCA: sample space uses a similarity matrix

Remark: $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$ has s_{ij} as the dot-product between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ thus acting as a similarity matrix.

Remark: the transform $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$ is actually the SVD transform.

Remark: We can use a different similarity matrix instead of the dot-product; we can use a kernel function and employ the kernel trick!

Algorithm:

1. Similarity: Build $N \times N$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$
5. Anomaly score: $\text{Score}(\boldsymbol{x}'_\ell) \ \ \forall \ell \in [N]$

Example: Kernel Space

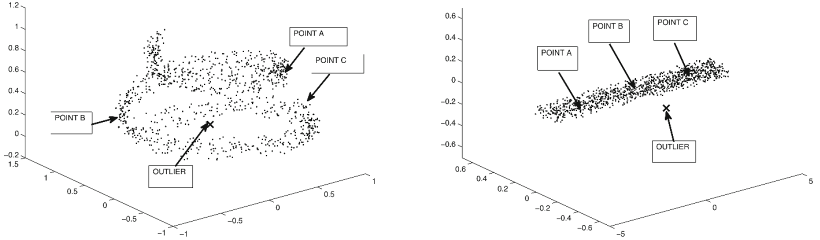


Figure: Sample space to kernel space (Aggarwal 2017)

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N\times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N\times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N\times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$
5. Anomaly score: Score($\boldsymbol{x}'_\ell$) $\forall \ell \in [N]$

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N\times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$
5. Anomaly score: $\text{Score}(\boldsymbol{x}'_\ell) \quad \forall \ell \in [N]$
6. Out-of-sample similarity: Build $(N - s) \times s$ similarity matrix $\boldsymbol{S_0}$ using the same kernel

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$
5. Anomaly score: $\text{Score}(\boldsymbol{x}'_\ell) \quad \forall \ell \in [N]$
6. Out-of-sample similarity: Build $(N - s) \times s$ similarity matrix $\boldsymbol{S_0}$ using the same kernel
7. Out-of-sample transform: $\boldsymbol{X}'_0 = \boldsymbol{S}_0(\boldsymbol{Q}\boldsymbol{\Lambda}^{-1})_k$

Nonlinear PCA: Dealing with large sample spaces

Remark: Computing $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top \in \mathbb{R}^{N \times N}$ for large N can be prohibitive (e.g. $N = 100,000$).

Solution: One can apply sub-sampling using s samples instead of N such that $k < s < N$ thus obtaining an $s \times s$ similarity matrix.

Algorithm:

1. Similarity: Build $s \times s$ similarity matrix $\boldsymbol{S}$ using kernel function κ
2. Decomposition: $\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}^2\boldsymbol{Q}^\top$
3. Transform: $\boldsymbol{X}' = \boldsymbol{X}(\boldsymbol{Q}\boldsymbol{\Lambda})_k$ ($N > k > d$ for some non-linear functions)
4. Normalize: $\boldsymbol{X}' = \boldsymbol{X}'\Lambda^{-1}$
5. Anomaly score: Score$(\boldsymbol{x}'_\ell)$ $\forall \ell \in [N]$
6. Out-of-sample similarity: Build $(N - s) \times s$ similarity matrix $\boldsymbol{S_0}$ using the same kernel
7. Out-of-sample transform: $\boldsymbol{X}'_0 = \boldsymbol{S}_0(\boldsymbol{Q}\boldsymbol{\Lambda}^{-1})_k$
8. Complete transform: $[(\boldsymbol{Q}\boldsymbol{\Lambda})_k \ ; \ \boldsymbol{S}_0(\boldsymbol{Q}\boldsymbol{\Lambda}^{-1})_k]^\top$

References

Aggarwal, Charu C (2017). *An introduction to outlier analysis*. Springer.

Candès, Emmanuel J et al. (2011). "Robust principal component analysis?" In: *Journal of the ACM (JACM)* 58.3, pp. 1–37.

Golub, G.H. and C.F. Van Loan (2013). *Matrix Computations*. John Hopkins University Press.

Netrapalli, Praneeth et al. (2014). "Non-convex robust PCA". In: *Advances in neural information processing systems* 27.