Reconnaissance de Formes et ML

Réduction de dimensionnalité

Faten Chakchouk
Enseignant - Chercheur

1

Réduction de dimensionnalité

Qu'est ce que c'est ?

	V_1	V_2	V_3	...	V_{n-2}	V_{n-1}	V_n
	5	1	1	1	2	1	3
	3	1	1	1	2	2	3
	8	10	10	8	7	10	9
	5	3	3	3	2	3	4
	1	1	1	1	2	3	3
	8	7	5	10	7	9	5
	7	4	6	4	6	1	4
x_i	4	1	1	1	2	1	2
	4	1	1	1	2	1	3
	10	7	7	6	4	10	4
	6	1	1	1	2	1	3

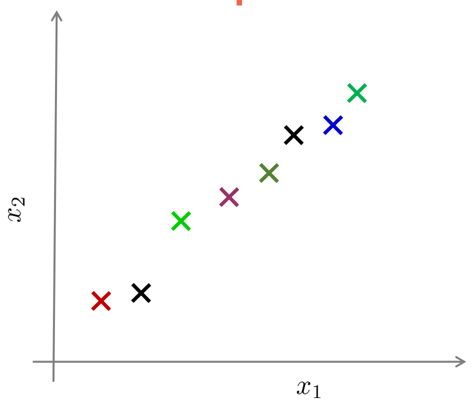
	C_1	C_2	C_3	...	C_k
z_i	2.3	0.3	0.3	0.3	1.7

$$x_i \in \mathbb{R}^n \xrightarrow{k \leq n} z_i \in \mathbb{R}^k$$

2

Réduction de dimensionnalité

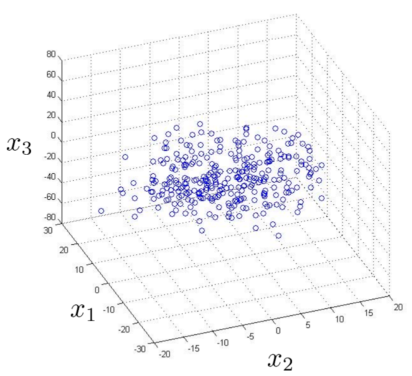
Motivation 1 : Compression de données

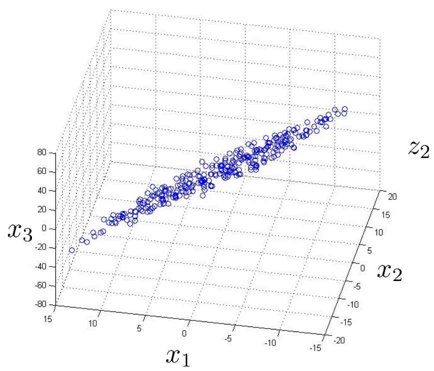


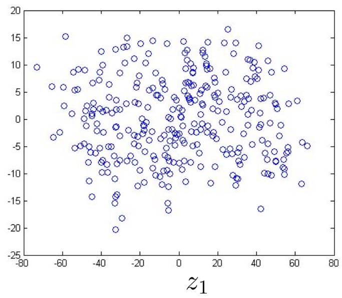
3

Réduction de dimensionnalité

Motivation 1 : Compression de données







4

Réduction de dimensionnalité

Motivation 2 : Visualisation des données en 2D ou 3D

12 températures mensuelles moyennes (sur 30 ans)

2 variables géographiques (latitude, longitude)

15 individus
Villes de France

	Janv	Févr	Mars	Avri	Mai	Juin	juil	Août	Sept	Octo	Nove	Déce	Lati	Long
Bordeaux	5.6	6.6	10.3	12.8	15.8	19.3	20.9	21	18.6	13.8	9.1	6.2	44.5	-0.34
Brest	6.1	5.8	7.8	9.2	11.6	14.4	15.6	16	14.7	12	9	7	48.24	-4.29
Clermont	2.6	3.7	7.5	10.3	13.8	17.3	19.4	19.1	16.2	11.2	6.6	3.6	45.47	3.05
Grenoble	1.5	3.2	7.7	10.6	14.5	17.8	20.1	19.5	16.7	11.4	6.5	2.3	45.1	5.43
Lille	2.4	2.9	6	8.9	12.4	15.3	17.1	17.1	14.7	10.4	6.1	3.5	50.38	3.04
Lyon	2.1	3.3	7.7	10.9	14.9	18.5	20.7	20.1	16.9	11.4	6.7	3.1	45.45	4.51
Marseille	5.5	6.6	10	13	16.8	20.8	23.3	22.8	19.9	15	10.2	6.9	43.18	5.24
Montpellier	5.6	6.7	9.9	12.8	16.2	20.1	22.7	22.3	19.3	14.6	10	6.5	43.36	3.53
Nantes	5	5.3	8.4	10.8	13.9	17.2	18.8	18.6	16.4	12.2	8.2	5.5	47.13	-1.33
Nice	7.5	8.5	10.8	13.3	16.7	20.1	22.7	22.5	20.3	16	11.5	8.2	43.42	7.15
Paris	3.4	4.1	7.6	10.7	14.3	17.5	19.1	18.7	16	11.4	7.1	4.3	48.52	2.2
Rennes	4.8	5.3	7.9	10.1	13.1	16.2	17.9	17.8	15.7	11.6	7.8	5.4	48.05	-1.41
Strasbourg	0.4	1.5	5.6	9.8	14	17.2	19	18.3	15.1	9.5	4.9	1.3	48.35	7.45
Toulouse	4.7	5.6	9.2	11.6	14.9	18.7	20.9	20.9	18.3	13.3	8.6	5.5	43.36	1.26
Vichy	2.4	3.4	7.1	9.9	13.6	17.1	19.3	18.8	16	11	6.6	3.4	46.08	3.26

5

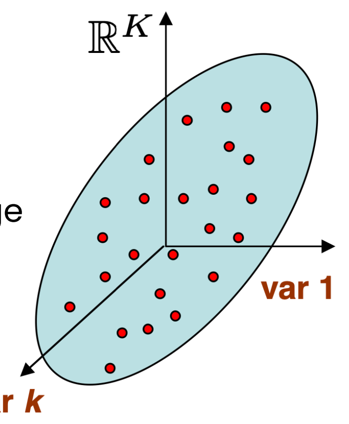
Réduction de dimensionnalité

Motivation 2 : Visualisation des données en 2D ou 3D

1 individu = 1 ligne du tableau ⇒ 1 point dans un espace à K dim

Représenter (visualiser) le **nuage des individus** :
Etudier les individus = Etudier la **forme** du nuage de points

- Si $K = 1$: Représentation axiale
- Si $K = 2$: Nuage de points
- Si $K = 3$: Représentation en 3D



6

Réduction de dimensionnalité
Motivations ...

Compression de données (avec perte)

Visualisation des données en 2D ou 3D

Extraction de caractéristiques :

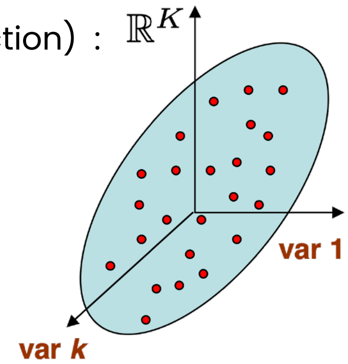
Fondamentales, explicatives, compactes, ...

Prétraitement => meilleure représentation de départ pour un autre algorithme (classification ou régression).

7

Analyse en Composantes Principales

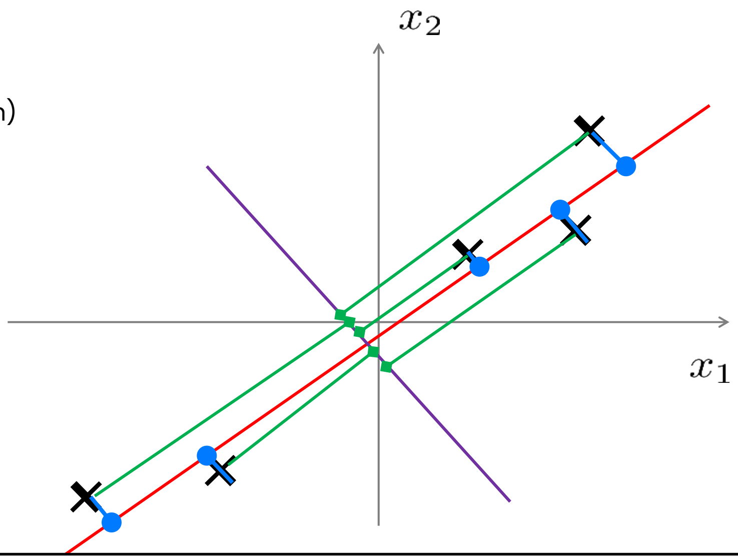
- Etudier la forme du nuage de points (individus)
- ACP : Technique de réduction de dimensionnalité tout en maximisant l'information utile retenue
- Trouver le meilleur hyperplan (plan de projection) : Minimiser l'erreur de projection



9

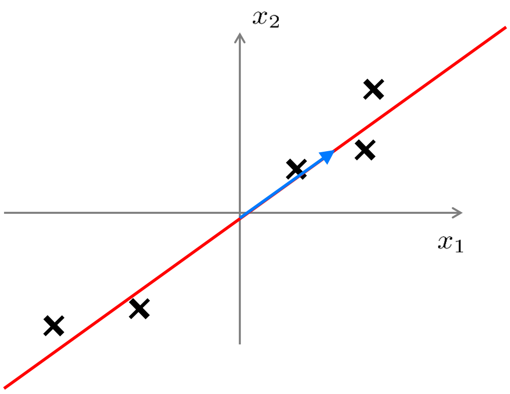
Analyse en Composantes Principales
Formulation - Minimiser l’erreur de projection

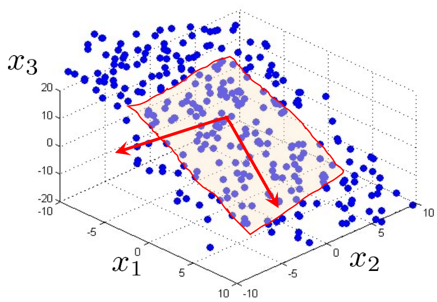
Trouver le meilleur axe (axe de projection)
qui minimiser l’erreur de projection



10

Analyse en Composantes Principales
Formulation – Minimiser l’erreur de projection





$3D \to 2D$
$K = 2$

- Réduction 2D → 1D : Trouver une direction (un vecteur $u^{(1)} \in \mathbb{R}^n$) qui minimise l’erreur de projection des données.
- Réduction nD → kD : trouver k vecteurs $u^{(1)}, u^{(2)}, \ldots, u^{(k)}$ (formant un hyperplane - base orthogonale) qui minimise l’erreur de projection des données.

11

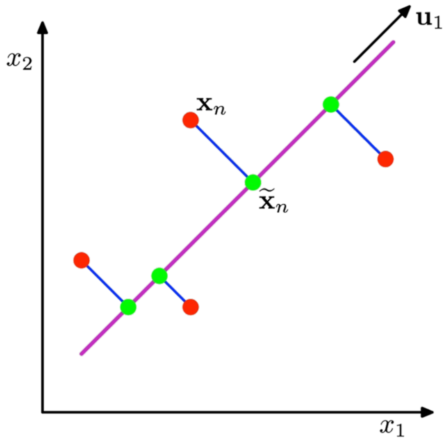
Analyse en Composantes Principales
Formulation - maximum de variance

Soit la moyenne empirique $\bar{\mathbf{x}} = \sum_{n=1}^{N} \mathbf{x}_n$
La variance des points projetés dans la direction $\mathbf{u}_1$ est:

$$\frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}})^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1$$

où $\mathbf{S}$ est la matrice de covariance empirique.
Maximiser sous la contrainte $\|\mathbf{u}_1\|^2 = \mathbf{u}_1^T\mathbf{u}_1 = 1$ (multiplicateur de Lagrange) donne:

$$\begin{aligned}\mathbf{S}\mathbf{u}_1 &= \lambda_1\mathbf{u}_1\\ \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 &= \lambda_1\end{aligned}$$



donc la variance est maximale quand $\mathbf{u}_1$ est le vecteur propre correspondant à la plus grande valeur propre λ_1.

12

Analyse en Composantes Principales
Formulation - maximum de variance

Si on a m échantillons $x^1, \ldots, x^m$ de même taille on définit leur matrice de variance-covariance Σ

$$\Sigma = \begin{pmatrix} \mathrm{Var}\,(x^1) & \cdots & \mathrm{cov}\,(x^1,x^k) & \cdots & \mathrm{cov}\,(x^1,x^m) \\ \vdots & \ddots & \vdots & & \vdots \\ \mathrm{cov}\,(x^k,x^1) & \cdots & \mathrm{Var}\,(x^k) & \cdots & \mathrm{cov}\,(x^k,x^m) \\ \vdots & & \vdots & \ddots & \vdots \\ \mathrm{cov}\,(x^m,x^1) & \cdots & \mathrm{cov}\,(x^m,x^k) & \cdots & \mathrm{Var}\,(x^m) \end{pmatrix}$$

$$\mathrm{cov}\,(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}).$$

$$\mathrm{Var}\,(x) = \mathrm{cov}\,(x,x) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2$$

13

Analyse en Composantes Principales
Prétraitement - Standardisation des données

Matrice **X** des données brutes

$$\begin{array}{c|ccc|}
 & 1\dots & j & \dots p \\ \hline
1 & & & \\
\vdots & & \vdots & \\
i & \dots & x_{ij} & \dots \\
\vdots & & \vdots & \\
n & & & \\ \hline\hline
\bar{x} & \dots & \bar{x}^j & \dots
\end{array}$$

Matrice **Y** des données centrées

$$\begin{array}{c|ccc|}
 & 1\dots & j & \dots p \\ \hline
1 & & & \\
\vdots & & \vdots & \\
i & \dots & y_{ij} & \dots \\
\vdots & & \vdots & \\
n & & & \\ \hline\hline
\bar{y} & \dots & 0 & \dots
\end{array}$$

avec dans le cas classique où $w_i = \frac{1}{n}$:

$$\bar{x}^j = \frac{1}{n}\sum_{i=1}^n x_{ij},$$
$$y_{ij} = x_{ij} - \bar{x}^j.$$

- Le nuage centré des points-individus a pour centre de gravité l'origine du repère.

14

Analyse en Composantes Principales
Prétraitement - Standardization des données

Matrice **X** des données brutes

$$\begin{array}{c|ccc|}
 & 1\dots & j & \dots p \\ \hline
1 & & & \\
\vdots & & \vdots & \\
i & \dots & x_{ij} & \dots \\
\vdots & & \vdots & \\
n & & & \\ \hline\hline
\bar{x} & \dots & \bar{x}^j & \dots \\
s & \dots & s_j & \dots
\end{array}$$

Matrice **Z** des données centrées-réduites

$$\begin{array}{c|ccc|}
 & 1\dots & j & \dots p \\ \hline
1 & & & \\
\vdots & & \vdots & \\
i & \dots & z_{ij} & \dots \\
\vdots & & \vdots & \\
n & & & \\ \hline\hline
\bar{z} & \dots & 0 & \dots \\
s & \dots & 1 & \dots
\end{array}$$

avec dans le cas classique où $w_i = \frac{1}{n}$:

$$s_j^2 = var(\mathbf{x}^j) = \frac{1}{n}\sum_{i=1}^n (x_{ij} - \bar{x}^j)^2,$$
$$z_{ij} = \frac{x_{ij} - \bar{x}^j}{s_j}.$$

15

Analyse en Composantes Principales
Algorithme

Etape 1 Pré-traitement des données : Normalisation (données centrées réduites)

Etape 2 Calculer la matrice de covariance Σ

$$\Sigma = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

```
# Calculating the covariance
matrixcovariance_matrix = np.cov(X.T)
```

Décomposition en valeurs et vecteurs propres

```
# Using np.linalg.eig function
from numpy import linalg as LA
U, S, V = LA.svd(a, full_matrices=True)
```

16

Analyse en Composantes Principales
Algorithme

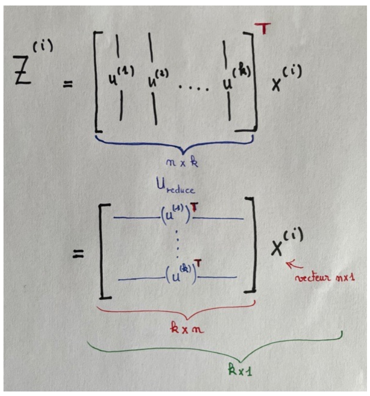
Etape 2 Décomposition en valeurs (V) et vecteurs singulières (U)

$$U = \begin{array}{c} \overbrace{\hphantom{u^{(1)} \quad u^{(2)} \quad \dots}}^{k} \\ \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \end{array} \in \mathbb{R}^{n\times n}$$

$$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^k$$

$$z^{(i)} = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x^{(i)}$$

U_{reduce}

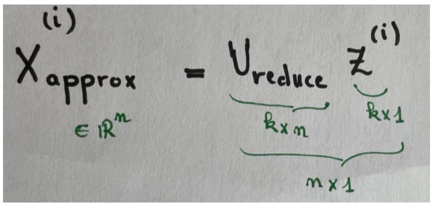


17

Analyse en Composantes Principales
Reconstruction à partir des données compressées

$$z = U_{reduce}^T x$$

x_{approx} = ?



18

Analyse en Composantes Principales
Choix du nombre de composantes principales k

Erreur quadratique moyenne de projection : $\boxed{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}$

Variation totale des données : $\boxed{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2}$

Souvent, on choisit la plus petite valeur de k telle que :

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \quad (1\%)$$

99% de variance est conservée

19

Analyse en Composantes Principales
Choix du nombre de composantes principales k

Algorithme :

Appliquer PCA avec différentes valeurs de k

Calculer $U_{reduce}, z^{(1)}, z^{(2)}, \ldots, z^{(m)}, x^{(1)}_{approx}, \ldots, x^{(m)}_{approx}$

Tester $$\frac{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)} - x^{(i)}_{approx}\|^2}{\frac{1}{m}\sum_{i=1}^{m}\|x^{(i)}\|^2} \leq 0.01?$$

$$\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{m} S_{ii}} \geq 0.99$$

20

Analyse en Composantes Principales (sur les individus)
Algorithme

Données

Tableau **m x n** de données **numériques**

n attributs

V_1	V_2	V_3	...	V_{n-2}	V_{n-1}	V_n
5	1	1	1	2	1	3
...	...	...	...	...	...	...
6	1	1	1	2	1	3

m observations

1. Pré-traitement des données : Normalisation (données centrées réduites)

2. Calculer la matrice de covariance **Cov**

$$\textbf{Cov} = \frac{1}{m}\sum_{i=1}^{n}(x^{(i)})(x^{(i)})^T$$

La matrice de covariance est de dimension **n x n**

3. Déterminer les composantes principales

Calcul de Valeurs et vecteurs propres

```
# Using np.linalg.eig function
from numpy import linalg as LA
eigen_values, eigen_vectors = LA.eig(Cov)
```

OU

Décomposition de Valeurs et vecteurs singulières (SVD)

```
# Using np.linalg.svd function
from numpy import linalg as LA
U, S, Vt = LA.svd(Cov, full_matrices=True)
```

21

Analyse en Composantes Principales (sur les individus)
Algorithme

3 Déterminer les composantes principales

Calcul de Valeurs et vecteurs propres

```
# Using np.linalg.eig function
from numpy import linalg as LA
eigen_values, eigen_vectors = LA.eig(Cov)
```

OU

Décomposition de Valeurs et vecteurs singulières (SVD) : Stabilité numérique

```
# Using np.linalg.svd function
from numpy import linalg as LA
U, S, Vt = LA.svd(Cov, full_matrices=True)
```

Trier dans l'ordre décroissant les valeurs propres

```
sorted_index = np.argsort(eigen_values)[::-1]
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvectors = eigen_vectors[:,sorted_index]
```

S : matrice des valeurs singulières (diagonals)

Vt : matrice des valeurs singulières

Pas besoin de trier : Avec la méthode svd les valeurs singulières sont triées dans un ordre décroissant

Extraction d'un sous ensemble de valeurs propres (k)

```
eigvector_subset = sorted_eigenvectors[:,0:k]
eigval_subset = sorted_eigenvalues[:k]
```

Réduction de la dimension : Projection

```
P = eigvector_subset.T.dot(X.T)
```

```
# Compute variance explained by principal components
rho = (S*S) / (S*S).sum()
```

22

Analyse en Composantes Principales (sur les individus)
Algorithme

3 Déterminer les composantes principales

Calcul de Valeurs et vecteurs propres

OU

Décomposition de Valeurs et vecteurs singulières (SVD) : Stabilité numérique

Trier dans l'ordre décroissant les valeurs propres

S : matrice des valeurs singulières (diagonals)

Vt : matrice des valeurs singulières

Pas besoin de trier : Avec la méthode svd les valeurs singulières sont triées dans un ordre décroissant

Extraction d'un sous ensemble de valeurs propres (k)

Comment ?

Calcul du pourcentatge de la variance expliquée : pour chaque composante

```
explained_var = []
for i in range(len(eigen_values)):
explained_var.append(eigen_values[i]/np.sum(eigen_values))
```

Calcul du pourcentatge de la variance expliquée

```
# Compute variance explained by principal components
explained_var = (S*S) / (S*S).sum()
```

Choisir k permettant de garder p% de variance expliquée des données (chercher k tel quee la somme des variances expliquées >=p%)

23

Analyse en Composantes Principales (sur les individus)
Algorithme

3 Déterminer les composantes principales

Calcul de Valeurs et vecteurs propres

```
# Using np.linalg.eig function
from numpy import linalg as LA
eigen_values, eigen_vectors = LA.eig(Cov)
```

OU

Décomposition de Valeurs et vecteurs singulières (SVD) : Stabilité numérique

```
# Using np.linalg.svd function
from numpy import linalg as LA
U, S, Vt = LA.svd(Cov, full_matrices=True)
```

Trier dans l'ordre décroissant les valeurs propres

```
sorted_index = np.argsort(eigen_values)[::-1]
sorted_eigenvalues = eigen_values[sorted_index]
sorted_eigenvectors = eigen_vectors[:,sorted_index]
```

S : matrice des valeurs singulières (diagonals)

Vt : matrice des valeurs singulières

Pas besoin de trier : Avec la méthode svd les valeurs singulières sont triées dans un ordre décroissant

Extraction d'un sous ensemble de valeurs propres (k)

```
eigvector_subset = sorted_eigenvectors[:,0:k]
eigval_subset = sorted_eigenvalues[:k]
```

Réduction de la dimension : Projection

```
P = eigvector_subset.T.dot(X.T)
```

```
# Compute variance explained by principal components
rho = (S*S) / (S*S).sum()
```

24