Prédiction de Données Fonctionnelles : Classification et Réduction de Dimension

Benjamin Auder

CEA - UPMC

16 juin 2009

Thèse depuis 02/2008

Directeur de thèse : Gérard Biau (UPMC)

Encadrant CEA : Bertrand Iooss (CEA)

Contexte industriel CEA

Laboratoire DER/SESI/LCFR : Conduite et Fiabilité des Réacteurs.

Code de calcul Cathare :

- Entrées $x_i \in \mathbb{R}^p$ = état initial du système physique ;
- Sorties $y_i \in \mathcal{C}^m([a, b])$ = évolution des paramètres physiques.

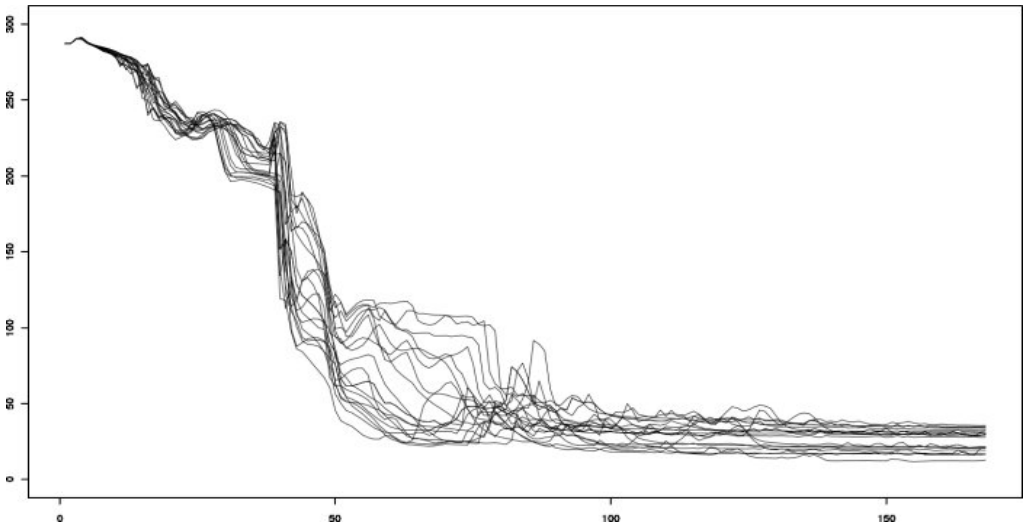


FIG.: Transitoires de température.

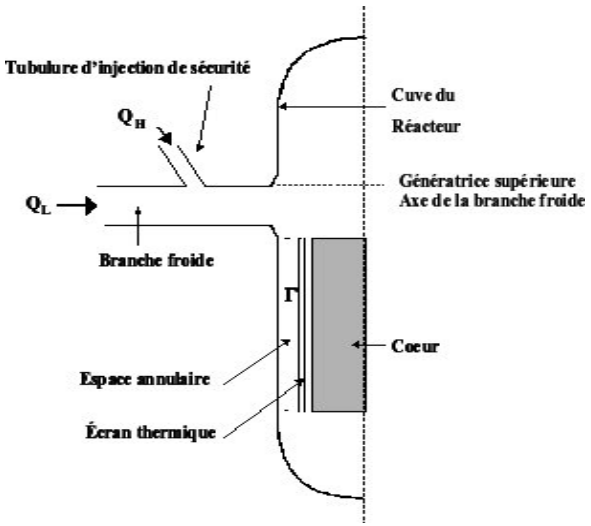


FIG.: Zone modélisée

But de la thèse

Code de calcul à *entrées vectorielles* et *sorties fonctionnelles*, coûteux.

$$\begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \longrightarrow \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix} = \begin{pmatrix} y_1(t_1) & \dots & y_1(t_D) \\ \vdots & \vdots & \vdots \\ y_n(t_1) & \dots & y_n(t_D) \end{pmatrix}$$

$i = 1..n$, $n \simeq 100$ à 1000 ; $x_{ij} \in \mathbb{R}$, $t \in [a, b]$.

But de la thèse

Code de calcul à *entrées vectorielles* et *sorties fonctionnelles*, coûteux.

$$\begin{pmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \dots & x_{np} \end{pmatrix} \longrightarrow \begin{pmatrix} y_1(t) \\ \vdots \\ y_n(t) \end{pmatrix} = \begin{pmatrix} y_1(t_1) & \dots & y_1(t_D) \\ \vdots & \vdots & \vdots \\ y_n(t_1) & \dots & y_n(t_D) \end{pmatrix}$$

$i = 1..n$, $n \simeq 100$ à 1000 ; $x_{ij} \in \mathbb{R}$, $t \in [a, b]$.

Objectif : prédiction de *données fonctionnelles* via un métamodèle.

\+ Intervalles de confiance sur les prédictions.

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

- *réaliser des analyses de sensibilité*
  estimation empirique des indices basés sur la variance p. ex.

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

- *réaliser des analyses de sensibilité*
  estimation empirique des indices basés sur la variance p. ex.
- *étudier la propagation d'incertitudes*
  incertitude Δy connaissant les lois des composantes d'entrée ?

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

- *réaliser des analyses de sensibilité*
  estimation empirique des indices basés sur la variance p. ex.
- *étudier la propagation d'incertitudes*
  incertitude Δy connaissant les lois des composantes d'entrée ?
- *déterminer des plans d'expériences*
  optimiser un critère sur $\{x_i\}$ via un algorithme évolutionaire p. ex.

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

- *réaliser des analyses de sensibilité*
  estimation empirique des indices basés sur la variance p. ex.
- *étudier la propagation d'incertitudes*
  incertitude Δy connaissant les lois des composantes d'entrée ?
- *déterminer des plans d'expériences*
  optimiser un critère sur $\{x_i\}$ via un algorithme évolutionaire p. ex.
- *calibrer les paramètres du code*
  $B(x;\theta)$: ajustement de θ (interne au code) a posteriori

Pourquoi modéliser un code de calcul ?

Un *métamodèle* est une approximation d'un code de calcul B calculée à partir de n entrées / sorties $(x_i, y_i = B(x_i))$.

Supposant avoir une bonne approximation de B, l'utilisateur peut :

- *réaliser des analyses de sensibilité*
  estimation empirique des indices basés sur la variance p. ex.
- *étudier la propagation d'incertitudes*
  incertitude Δy connaissant les lois des composantes d'entrée ?
- *déterminer des plans d'expériences*
  optimiser un critère sur $\{x_i\}$ via un algorithme évolutionaire p. ex.
- *calibrer les paramètres du code*
  $B(x;\theta)$: ajustement de θ (interne au code) a posteriori

..à moindre coût.

Étapes

Plan d'analyse du code de calcul :

1. partition des (entrées +) sorties en K clusters ;
2. apprentissage supervisé entrées − clusters ;
3. réduction de la dimension des sorties ;
4. régression adaptée sur les coordonnées réduites.

Plan

Motivation

Dans l'optique du métamodèle : plus facile de modéliser des comportements spécifiques du code, puis de combiner les sous-modèles.

Idée : les $\mathcal{M}_k(x)$ étant des métamodèles plus simples,

$$\mathcal{M}(x) = \sum_{k=1}^{K} \pi_k \mathcal{M}_k(x)\,.$$

Motivation

Dans l'optique du métamodèle : plus facile de modéliser des comportements spécifiques du code, puis de combiner les sous-modèles.

Idée : les $\mathcal{M}_k(x)$ étant des métamodèles plus simples,

$$\mathcal{M}(x) = \sum_{k=1}^{K} \pi_k \mathcal{M}_k(x)\,.$$

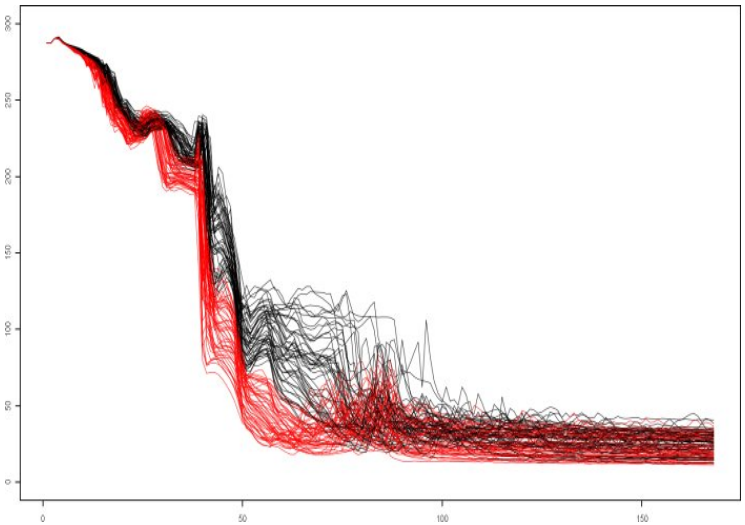


FIG.: 100 sorties du code Cathare.

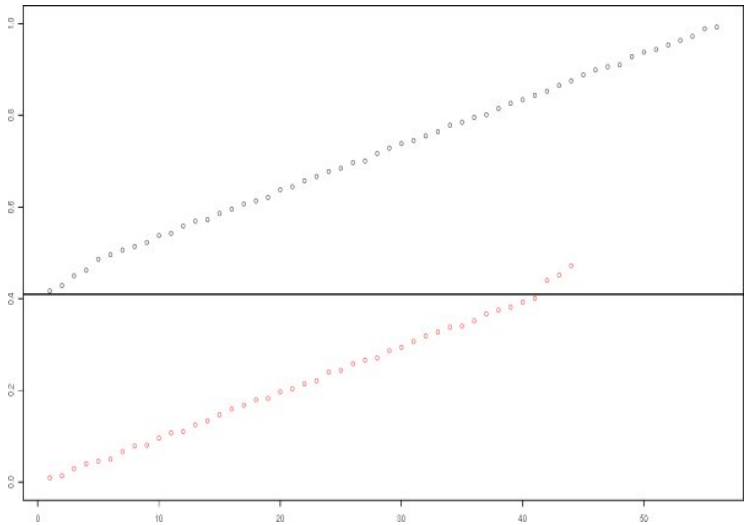


FIG.: Composante 4 des entrées.

Plan

"État de l'art"

Point de départ : $y_i(t) = \mu_k(t) + \eta_i^{(k)}(t) + \varepsilon_i(t)$.

"État de l'art"

Point de départ : $y_i(t) = \mu_k(t) + \eta_i^{(k)}(t) + \varepsilon_i(t)$.

Régression à effets aléatoires, *S. Gaffney & P. Smyth, 2003*

$$y_i(t) = T(t)m_k + T(t)a_i^{(k)} + \varepsilon_i(t)\,,$$

T matrice de régression linéaire généralisée en colonnes, $a_i^{(k)} \sim \mathcal{N}_q(0, C_k)$.

Estimation des moyennes et covariances : algorithme EM.

"État de l'art"

Point de départ : $y_i(t) = \mu_k(t) + \eta_i^{(k)}(t) + \varepsilon_i(t)$.

Régression à effets aléatoires, *S. Gaffney & P. Smyth, 2003*

$$y_i(t) = T(t)m_k + T(t)a_i^{(k)} + \varepsilon_i(t)\,,$$

T matrice de régression linéaire généralisée en colonnes, $a_i^{(k)} \sim \mathcal{N}_q(0, C_k)$.

Estimation des moyennes et covariances : algorithme EM.

Mélange de décompositions K-L, *J. M. Chiou & P.-L. Li, 2008*

$$y_i(t) = \theta_k m_k(t) + \sum_{j=1}^{q} \xi_j^{(k)} \phi_j^{(k)}(t) + \varepsilon_i(t)\,,$$

θ_k = effet aléatoire multiplicatif. Minimisation de la distorsion.

Fonctions m_k et ϕ_j estimées itérativement jusqu'à convergence.

Convergence vers le modèle "optimal"

Erreur d'approximation par les c_k = distorsion moyenne dans l'espace des y_i :

$$W(c) = \mathbb{E}_y \min_{j=1..K} \|y - c_j\|^2$$

Pratique : c^*_{dn}
Théorie : c^*

Convergence vers le modèle "optimal"

Erreur d'approximation par les c_k = distorsion moyenne dans l'espace des y_i :

$$W(c) = \mathbb{E}_y \min_{j=1..K} \|y - c_j\|^2$$

Pratique : c^*_{dn}
Théorie : c^*

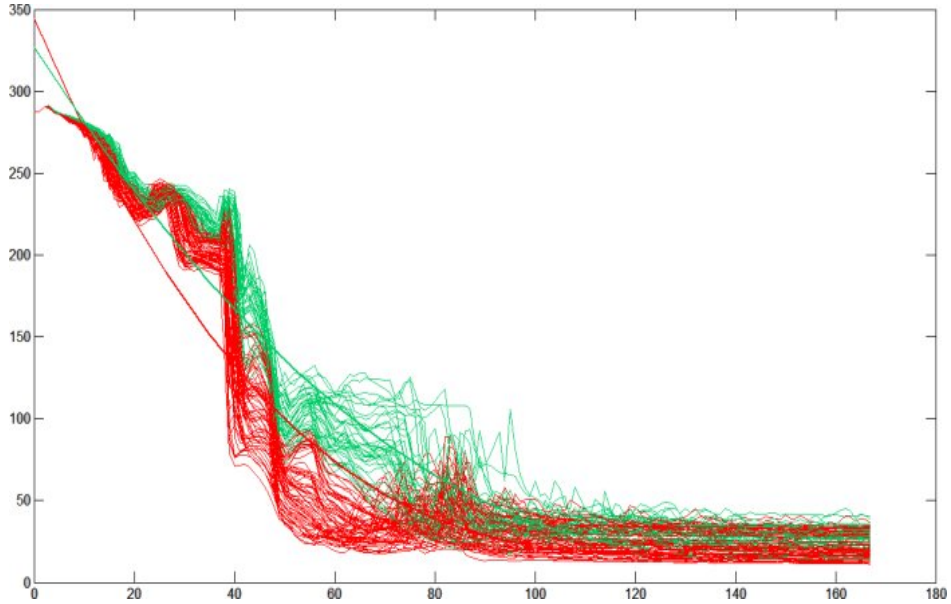
$$P(\alpha) = 1 - 4(2n)^{k(d+1)} e^{\frac{-n\alpha^2}{512B^2}} .$$

Théorème :
Soient $\alpha > 0$, $d \in \mathbb{N}^*$. Alors avec probabilité au moins $P(\alpha)$:

$$W(c^*_{nd}) - W(c^*) \leq \alpha + \frac{4R^2}{\varphi^d_{min}} .$$

Exemples (régression à effets aléatoires)

FIG.: 2 clusters avec CCT toolbox de S. Gaffney (2003), régressions linéaires :



Exemples (régression à effets aléatoires)

FIG.: 2 clusters avec CCT toolbox de S. Gaffney (2003), régressions linéaires :

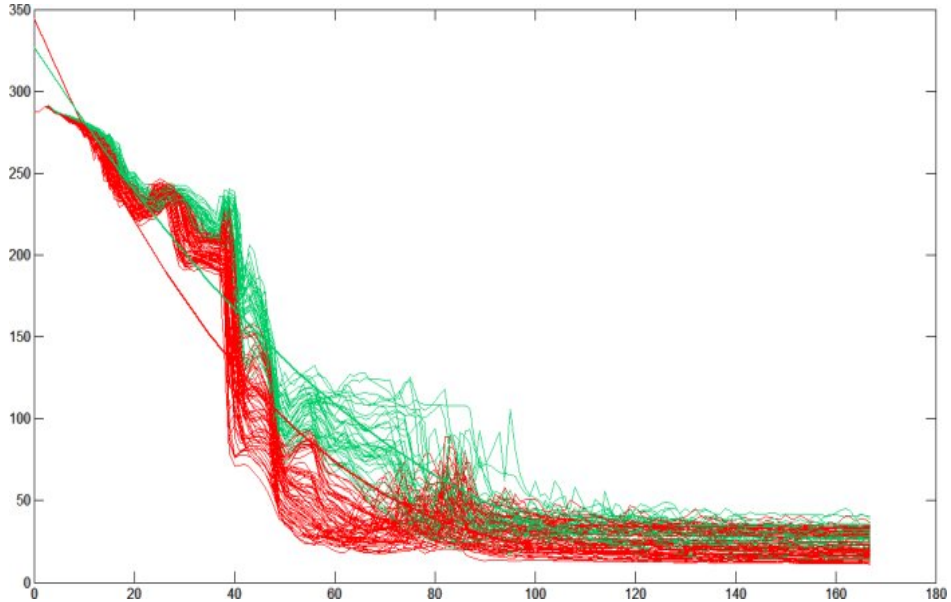
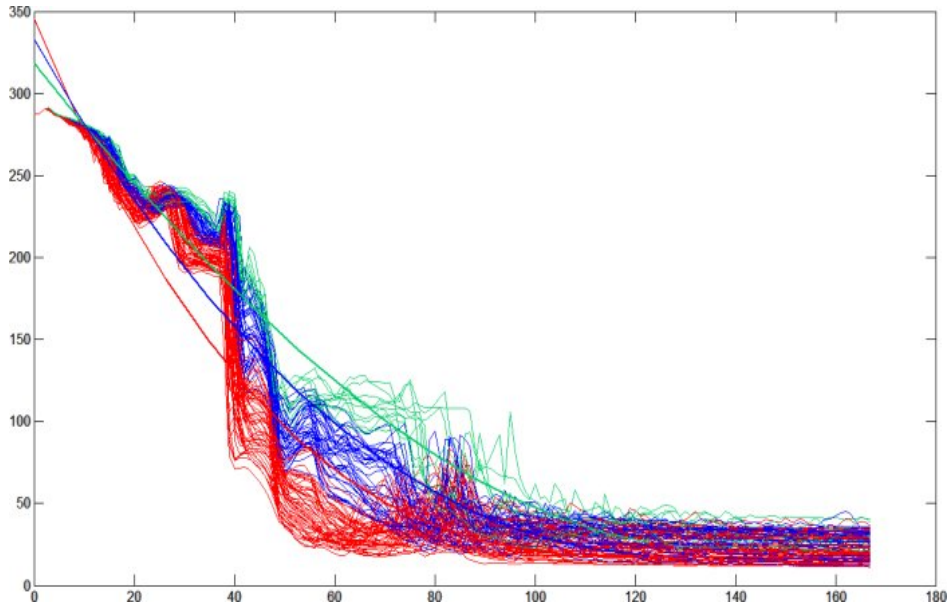


FIG.: ..3 clusters :



Plan

Point de départ

Graphe G_k des k plus proches voisins ; similarité $w(f,g) = e^{-\gamma\|f-g\|^2}$ p.ex.

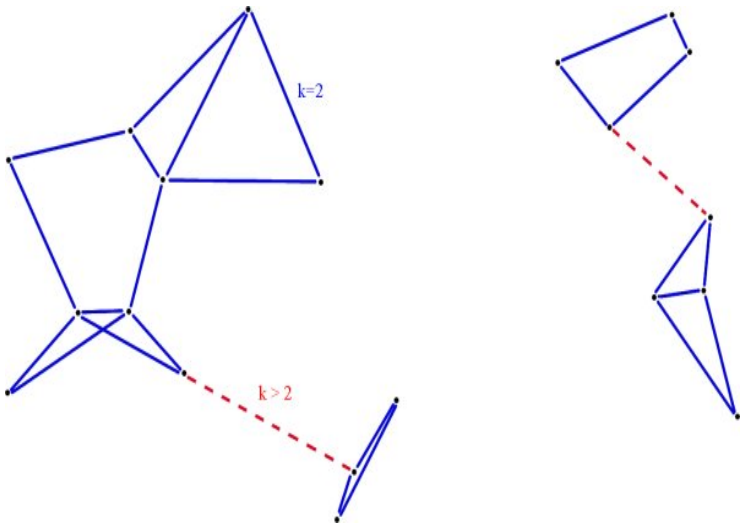


FIG.: Graphe 2-NN en bleu, quelques liens 3-NN en rouge.

$\Rightarrow$ Distances géodésiques estimées, ou..

Point de départ

Graphe G_k des k plus proches voisins ; similarité $w(f,g) = e^{-\gamma\|f-g\|^2}$ p.ex.

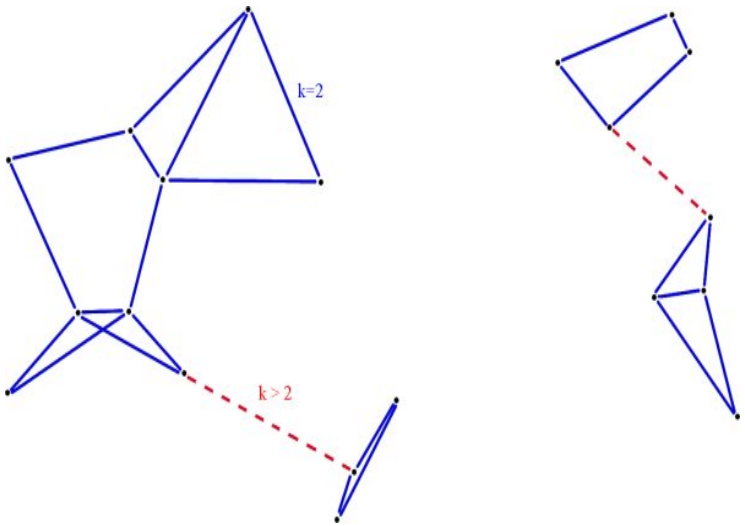


FIG.: Graphe 2-NN en bleu, quelques liens 3-NN en rouge.

$\Rightarrow$ Distances géodésiques estimées, ou..

Probabilités de transitions proportionnelles aux similarités.

Distance de marche aléatoire dans le graphe :
$d^2(f,g) = \mathbb{E}(t_{f\to g}) + \mathbb{E}(t_{g\to f})$, $t_{f\to g}$ temps de trajet moyen de f vers g.

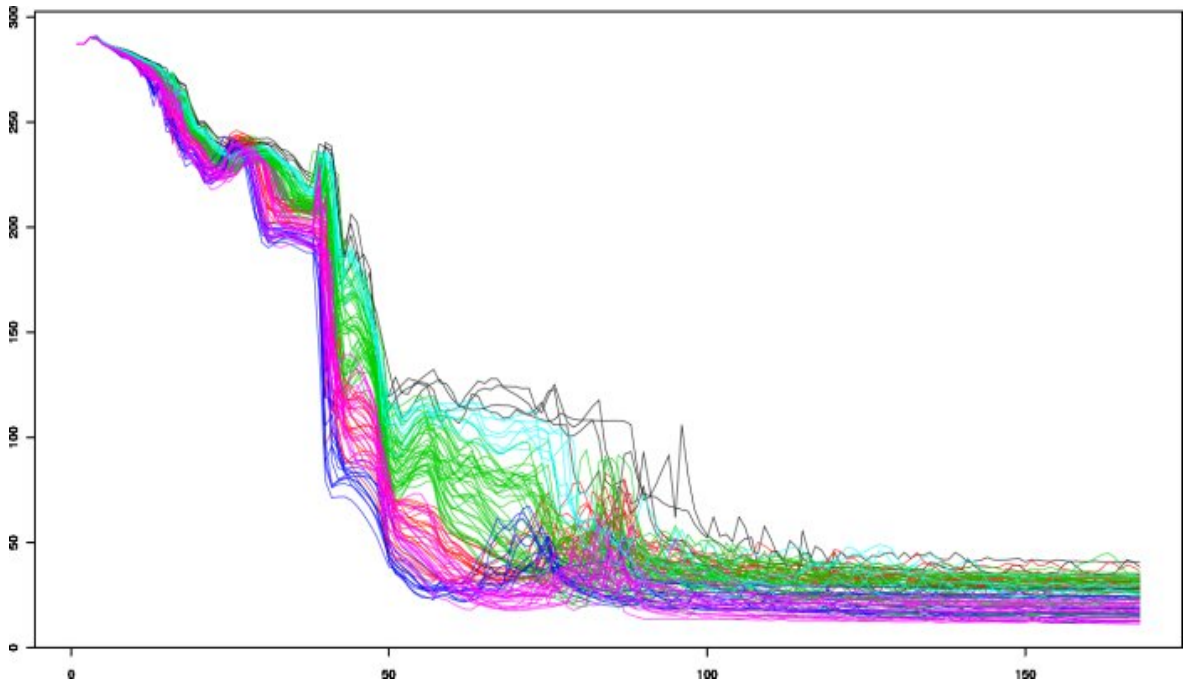
Algorithme utilisé : "CTH" (Commute-Time Hiérarchique)

1. choix d'une mesure de similarité sur les données ;
2. construction du graphe k-NN avec k plus petit entier tel que parties connexes(G_k) = parties connexes(G_{k-1}) ;
3. calcul de la matrice M des commute distances : temps moyens d'allers-retours entre deux sommets ;
4. clustering hiérarchique sur les sommets du graphe k-NN, utilisant M.

Algorithme utilisé : "CTH" (Commute-Time Hiérarchique)

1. choix d'une mesure de similarité sur les données ;
2. construction du graphe k-NN avec k plus petit entier tel que parties connexes(G_k) = parties connexes(G_{k-1}) ;
3. calcul de la matrice M des commute distances : temps moyens d'allers-retours entre deux sommets ;
4. clustering hiérarchique sur les sommets du graphe k-NN, utilisant M.

Six clusters sur 100 sorties de code Cathare :



Plan

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

- visualiser la structure des données ;

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

- visualiser la structure des données ;
- détecter des anomalies via un modèle ;

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

- visualiser la structure des données ;
- détecter des anomalies via un modèle ;
- réduire la complexité des algorithmes ;

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

- visualiser la structure des données ;
- détecter des anomalies via un modèle ;
- réduire la complexité des algorithmes ;
- sélectionnner les variables d'entrée ..etc.

Motivations

But : résumer l'information contenue dans des vecteurs de grande dimension, ou dans des fonctions.

..Pour :

- visualiser la structure des données ;
- détecter des anomalies via un modèle ;
- réduire la complexité des algorithmes ;
- sélectionnner les variables d'entrée ..etc.

Techniques classiques :

- analyse en composantes principales (ACP)
  $=$ *maximisation de la variance projetée ;*
- analyse en composantes indépendantes (ACI)
  $=$ *projections linéaires pour coefficients* $\simeq$ *indépendants ..etc.*

Plan

Décomposition de Karhunen-Loève

On considère la sortie du code comme un procesus stochastique : $y_i(t)$ réalisation de $\omega \mapsto Y(\omega)(t)$.

Décomposition de Karhunen-Loève

On considère la sortie du code comme un procesus stochastique : $y_i(t)$ réalisation de $\omega \mapsto Y(\omega)(t)$.

$$Y(\omega)(t) = \mu(t) + \sum_{k=1}^{+\infty} \xi_k(\omega)\phi_k(t)\,,$$

avec $\mu(t) = \mathbb{E}[Y(t)]$, et $\mathrm{Var}(\xi_k) = \lambda_k$, k^{eme} valeur propre de l'opérateur d'auto-covariance de Y.

$\phi_1, \ldots, \phi_k, \ldots$: base orthonormale.

Décomposition de Karhunen-Loève

On considère la sortie du code comme un procesus stochastique : $y_i(t)$ réalisation de $\omega \mapsto Y(\omega)(t)$.

$$Y(\omega)(t) = \mu(t) + \sum_{k=1}^{+\infty} \xi_k(\omega)\phi_k(t)\,,$$

avec $\mu(t) = \mathbb{E}[Y(t)]$, et $\mathrm{Var}(\xi_k) = \lambda_k$, k^{eme} valeur propre de l'opérateur d'auto-covariance de Y.

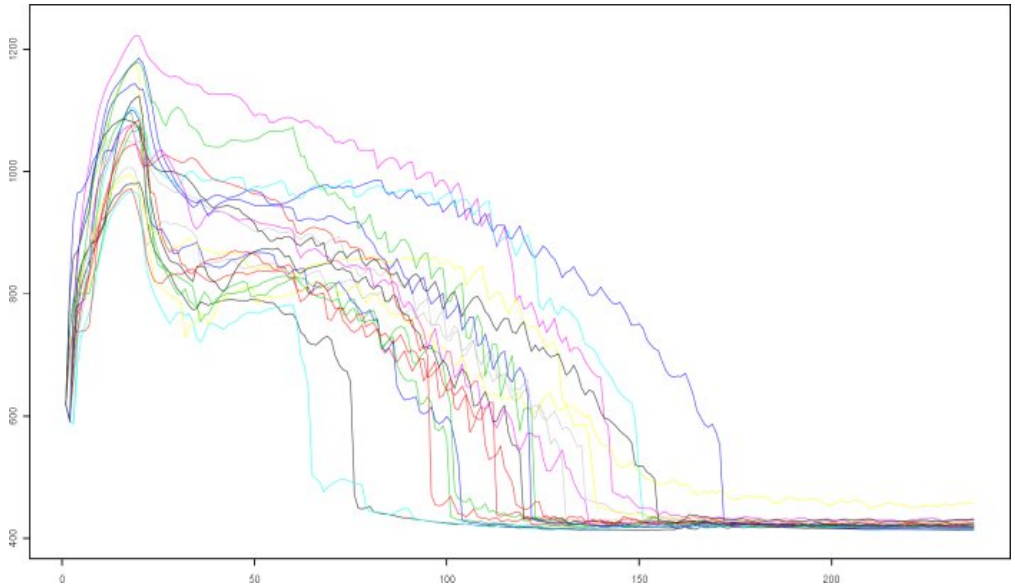
$\phi_1, \ldots, \phi_k, \ldots$: base orthonormale.

Autres options :

- base de polynômes orthogonaux ;
- base de Fourier ;
- base d'ondelettes ..etc.

Illustration

Courbes du jeu de données Bemuse (Cathare)



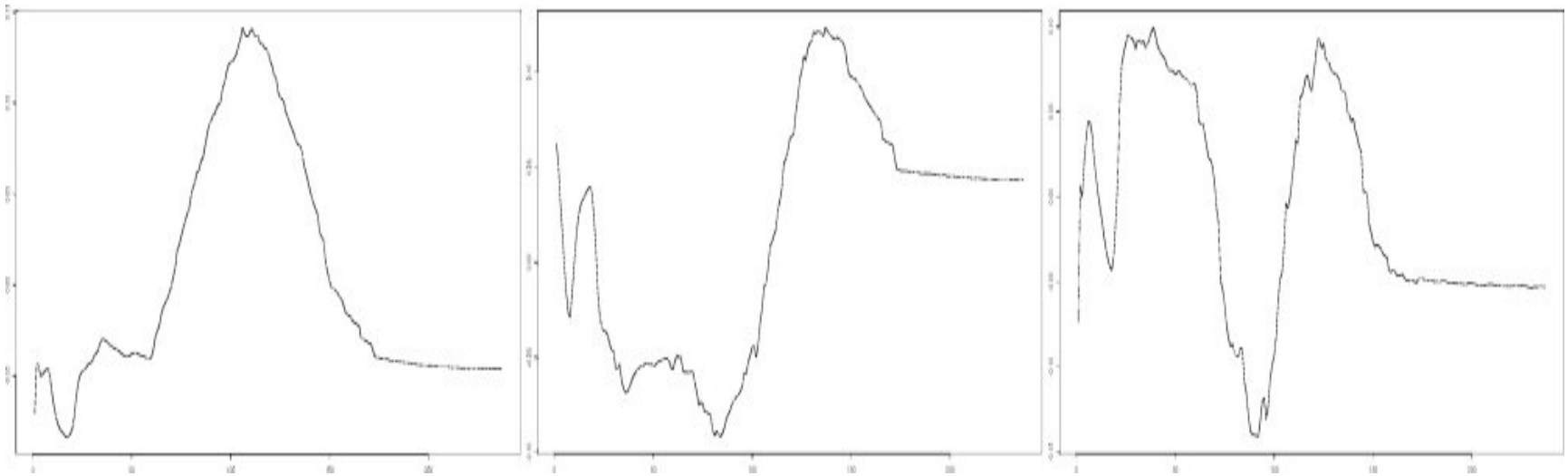


FIG.: 1^{eres} fonctions propres : 70%, 76% et 79% de variance expliquée (g. à d.).

Premiers résultats

Algorithme :

1. clustering spectral sur les courbes ;
2. décomposition K-L dans chaque cluster $\Rightarrow$ coefficients de projection ;
3. apprentissage des relations entrées / coefficients par SVR.

Premiers résultats

Algorithme :

1. clustering spectral sur les courbes ;
2. décomposition K-L dans chaque cluster $\Rightarrow$ coefficients de projection ;
3. apprentissage des relations entrées / coefficients par SVR.

Cas "facile" : entrées en dimension 4, 100 courbes.

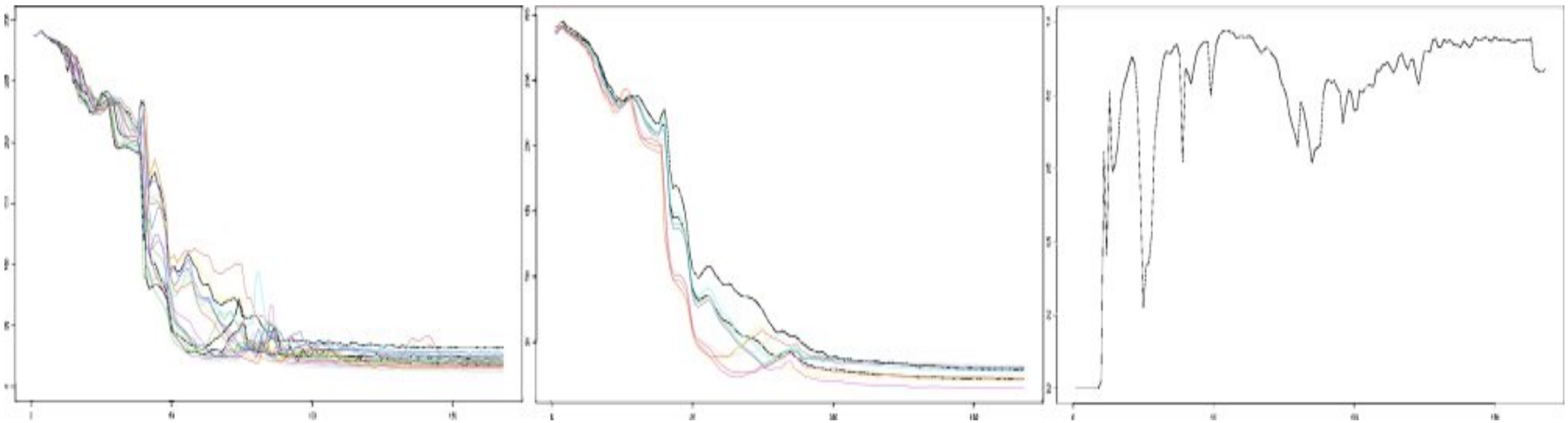


FIG.: Courbes d'origine à gauche, prédites au centre ; coefficient Q_2 pour 85% de variance expliquée à droite.

Résultats - suite

Cas où ça se passe mal : jeu de données Bemuse I ;
53 dimensions en entrée, 100 courbes.

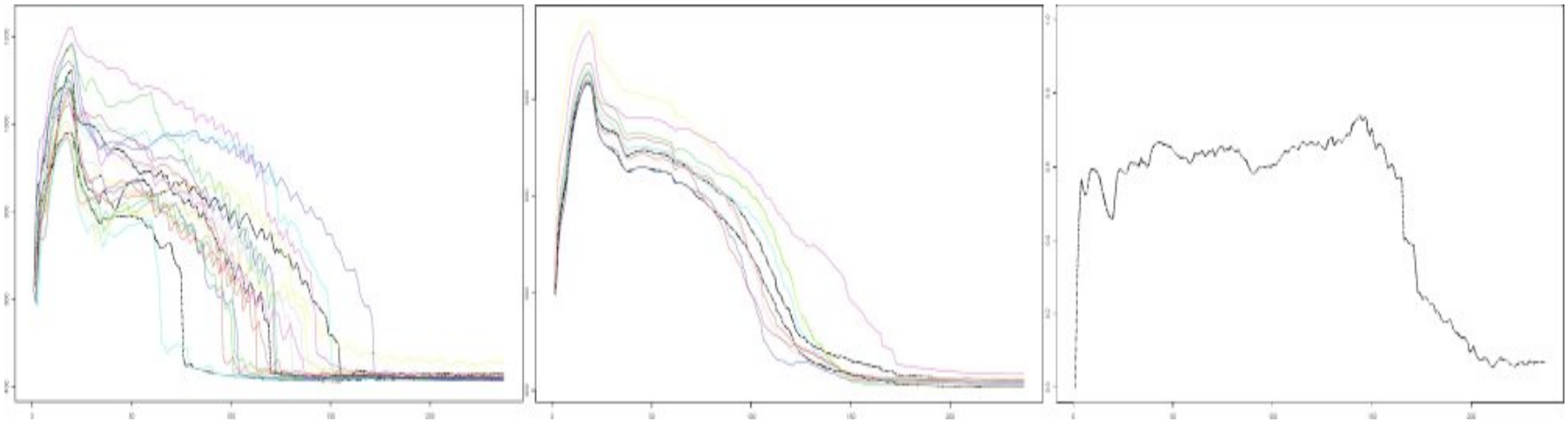


FIG.: Courbes d'origine à gauche, prédites au centre ; coefficient Q_2 pour 85% de variance expliquée à droite.

Résultats - suite

Cas où ça se passe mal : jeu de données Bemuse I ;
53 dimensions en entrée, 100 courbes.

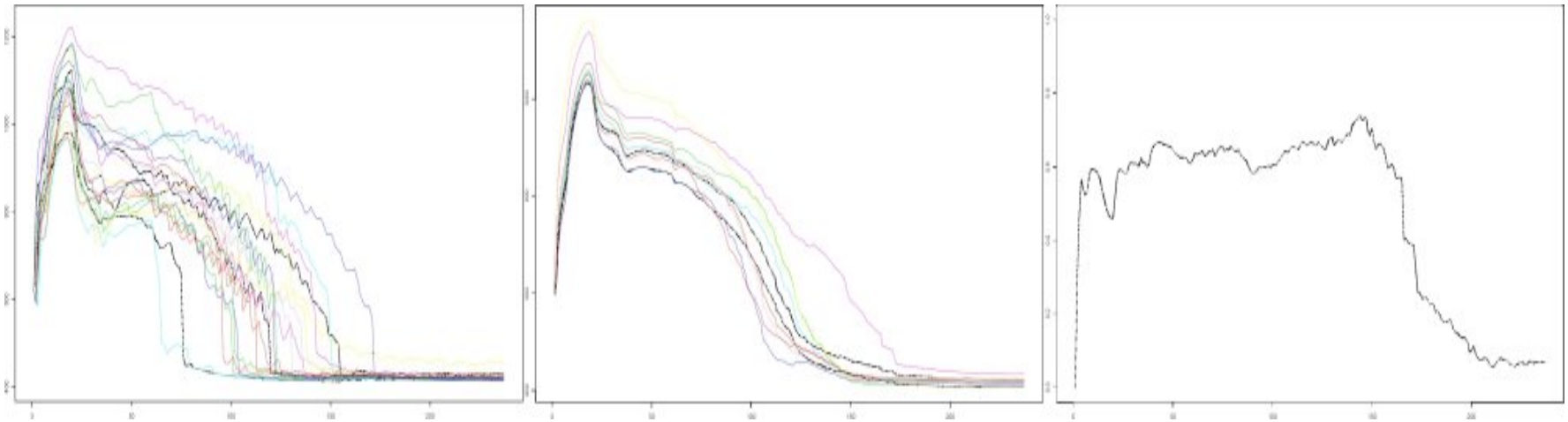


FIG.: Courbes d'origine à gauche, prédites au centre ; coefficient Q_2 pour 85% de variance expliquée à droite.

Explications possibles :

- mauvaise réduction de la dimension ;
- régression à améliorer ;
- rapport nombre de données / dimension en entrée trop faible ...

Plan

Illustration

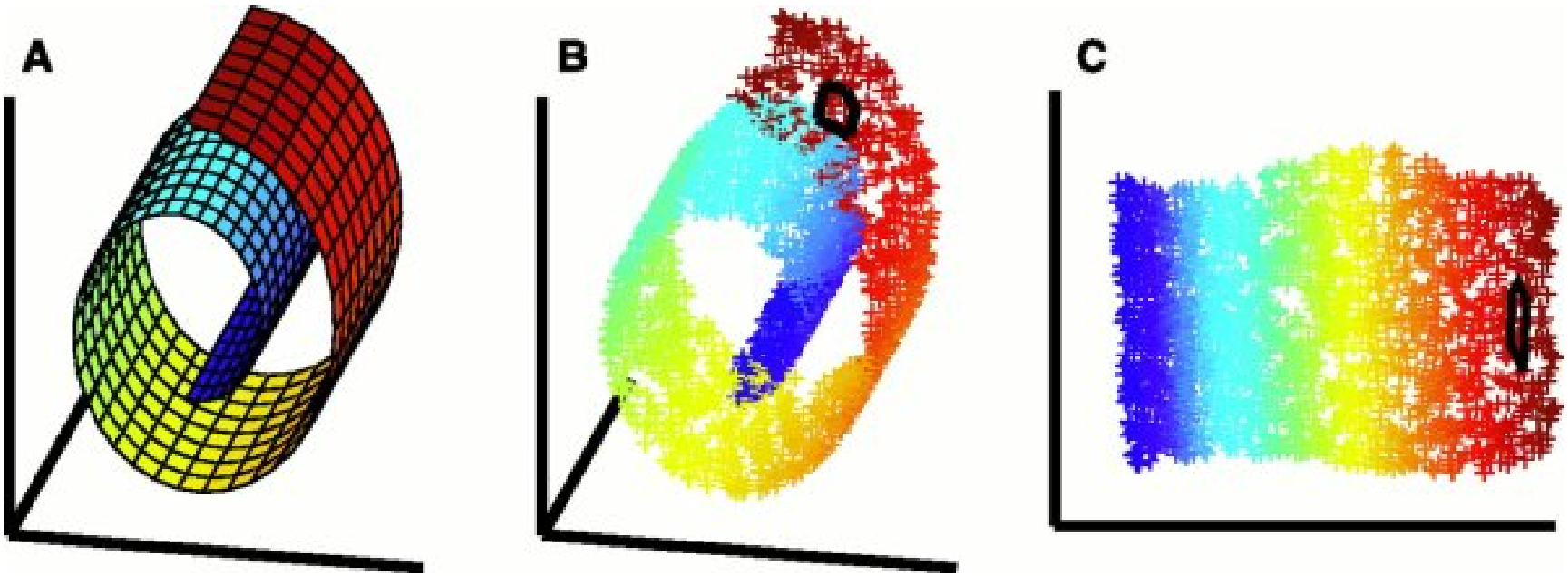


FIG.: jeu de données "swissroll".

Données "non linéaires", mais structure de variété.

But : trouver un système de coordonnées le plus réduit possible, qui préserve la topologie des données.

Aperçu des méthodes

ACP non linéaire (S. Girard, 1999)

Base orthonormée, mais restitution non linéaire
(ACP : $t_1, \ldots, t_d \mapsto \sum_{i=1}^{d} t_i \phi_i$).

Courbes principales (T. Hastie, 1984 ...)

La "base" de décomposition est constituée de courbes ;
plusieurs définitions / régularisations possibles.

Aperçu des méthodes

ACP non linéaire (S. Girard, 1999)

Base orthonormée, mais restitution non linéaire
(ACP : $t_1, \ldots, t_d \mapsto \sum_{i=1}^{d} t_i \phi_i$).

Courbes principales (T. Hastie, 1984 ...)

La "base" de décomposition est constituée de courbes ;
plusieurs définitions / régularisations possibles.

Approches plus locales :

- Isomap : estimation des distances géodésiques dans D, puis obtention d'une matrice de produits scalaires à partir de D ;
- LLE (Local Linear Embedding) $\simeq$ régressions linéaires locales ;
- Laplacian eigenmap ;
- Commute-time embedding ..etc.

Conclusion

Deux grands axes pour la classification :

- model-based clustering : regroupement autour de fonctions "moyennes" ;
- model-free clustering : optimisation d'un critère sur le tableau de similarité.

$\longrightarrow$ Model-free plus intéressant en général, car on ne dispose pas de modèle fiable sur les courbes en sortie.

Conclusion

Deux grands axes pour la classification :

- model-based clustering : regroupement autour de fonctions "moyennes" ;
- model-free clustering : optimisation d'un critère sur le tableau de similarité.

$\longrightarrow$ Model-free plus intéressant en général, car on ne dispose pas de modèle fiable sur les courbes en sortie.

Réduction de la dimension :

K-L pas toujours efficace, base d'ondelettes adaptative à explorer ;

méthodes plus sophistiquées peut-être inutiles en pratiques.

Conclusion

Deux grands axes pour la classification :

- model-based clustering : regroupement autour de fonctions "moyennes" ;
- model-free clustering : optimisation d'un critère sur le tableau de similarité.

$\longrightarrow$ Model-free plus intéressant en général, car on ne dispose pas de modèle fiable sur les courbes en sortie.

Réduction de la dimension :

K-L pas toujours efficace, base d'ondelettes adaptative à explorer ;

méthodes plus sophistiquées peut-être inutiles en pratiques.

Autres problèmes :

- coefficients corrélés ;
- plusieurs courbes corrélées en sortie ;
- validation du métamodèle construit ? ..etc