THÈSE

présentée

DEVANT L'UNIVERSITÉ DE STRASBOURG I

pour obtenir

le grade de : ***DOCTEUR DE L'UNIVERSITÉ DE STRASBOURG I***

Mention : Électronique, électrotechnique et automatique

par

Matthieu Brucher

Laboratoires d'accueil : LSIIT UMR 7005, STRASBOURG
LINC UMR 7191, STRASBOURG

École doctorale : Mathématiques, Sciences de l'Information et de l'Ingénieur

Titre de la thèse :

Représentations compactes et apprentissage non supervisé de variétés non linéaires. Application au traitement d'images.

Soutenue le 03/10/2008, devant la commission d'Examen

COMPOSITION DU JURY :

M.	Stéphane	GIRARD	Rapporteur externe
M.	Olivier	LEZORAY	Rapporteur externe
M.	Eric	SONNENDRÜCKER	Rapporteur interne
M.	Christian	HEINRICH	Examinateur
M.	Jean-Paul	ARMSPACH	Directeur de thèse
M.	Fabrice	HEITZ	Directeur de thèse

Remerciements

Je tiens tout d'abord à remercier Monsieur Stéphane Girard, chargé de recherche à l'Institut National de Recherche en Informatique et en Automatique (INRIA, Rhônes-Alpes), Monsieur Olivier Lezoray, maître de conférence au Groupe de Recherche en Informatique, Image, Automatique et Instrumentation (GREYC, Caen) et Monsieur Eric Sonnendrücker, professeur à l'Institut de Recherche Mathématique Avancée (IRMA, Strasbourg) d'avoir accepté de rapporter ce travail de thèse.

Je tiens ensuite à remercier mes deux directeurs de thèse, Monsieur Fabrice Heitz, professeur à l'Université Louis Pasteur (ULP, Strasbourg) et Monsieur Jean-Paul Armspach, ingénieur de recherche au Laboratoire d'Imagerie et de Neurosciences Cognitives (LINC, Strasbourg) pour leur encadrement, leur complémentarité, et le cadre de travail privilégié qu'ils ont su m'offrir.

Un grand merci à Christian Heinrich, maître de conférences à l'Université Louis Pasteur (ULP, Strasbourg), qui a encadré ce travail. Je le remercie pour ce qu'il m'a apporté sur le plan scientifique, pour la qualité de ses conseils, pour toutes les heures passées à la relecture attentive de mes différentes productions écrites et à l'écoute des différentes présentations, pour sa disponibilité et sa bonne humeur.

Ce travail de recherche a été mené en étroite collaboration entre le Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (LSIIT, Strasbourg) et Laboratoire d'Imagerie et de Neurosciences Cognitives (LINC, Strasbourg) dans le cadre d'un projet multilaboratoires sur le thème de l'Imagerie et Robotique Médicale et Chirurgicale. J'exprime toute ma reconnaissance à Monsieur Fabrice Heitz, directeur du LSIIT et Monsieur Christian Kelche, directeur du LINC, pour leur accueil dans leurs laboratoires respectifs.

Je remercie aussi toutes les personnes que j'ai pu côtoyer au sein du LSIIT et de l'IPB. Je tiens tout particulièrement à remercier les permanents Sylvain Faisan, Vincent Noblet, François Rousseau, Julien Lamy et Nicolas Passat, les différents doctorants avec qui j'ai eu de nombreux contacts dont Torbjorn Vik, Christelle Nithart, Jean-Christophe Beyler, Stéphanie Bricq, Bessem Bouraoui, Mathieu Tobie, Hervé Boisgontier, Félix Renard, ainsi que les différents stagiaires avec qui j'ai pu avoir de nombreuses discussions, scientifiques ou non, et qui ont contribué à la bonne ambiance générale de l'équipe.

Enfin, je tiens à remercier ma femme Emilie de m'avoir supporté pendant toute cette période et ma famille, sans qui je n'aurais pas pu achever ce travail, ainsi que mes amis qui m'ont soutenu pendant ces quelques années.

Table des matières

Introduction **xi**

1 État de l'art **1**
1.1 Représentation des données . . . 3
1.2 Réduction de dimension . . . 4
1.2.1 *MultiDimensional Scaling* (MDS) . . . 4
1.2.2 Modèles basés sur le graphe . . . 6
1.2.3 Autres modèles . . . 9
1.2.4 Détermination de la dimension de l'espace réduit . . . 10
1.3 Régression multidimensionnelle . . . 11
1.3.1 Construction d'un modèle pendant la compression . . . 11
1.3.2 Estimation d'une fonction entre les deux espaces . . . 12
1.4 Projection sur un modèle . . . 13
1.4.1 Projection à l'aide de réseaux de neurones . . . 14
1.4.2 Utilisation des données dans l'espace original . . . 14
1.4.3 Moindres carrés . . . 15
1.4.4 Projection probabiliste . . . 15
1.5 Classification . . . 16
1.5.1 Classification spectrale . . . 16
1.5.2 Classifieur probabiliste . . . 17
1.5.3 Classifieur SVM . . . 17
1.6 Objectifs . . . 17

2 Réduction de dimension **19**
2.1 Choix d'un critère de réduction . . . 20
2.2 Optimisation de la fonction . . . 20
2.2.1 Caractéristiques de la fonction à optimiser . . . 21
2.2.2 Optimisation classique . . . 22
2.2.3 Optimisation par incorporations successives . . . 22

2.3 Sélection d'une dimension adéquate . . . 22
2.4 Résultats et comparaisons . . . 23
2.4.1 Présentation des méthodes utilisées . . . 23
2.4.2 Résultats et évaluation . . . 26
2.5 Conclusion . . . 34

3 Régression et modélisation des données 35
3.1 Régression globale . . . 36
3.1.1 Régression simple . . . 36
3.1.2 Régression avec optimisation d'une fonction de coût . . . 37
3.1.3 Approximation d'un nouveau point . . . 40
3.2 Régression par blocs . . . 41
3.2.1 Position du problème . . . 41
3.2.2 Partitionnement par corrélation . . . 41
3.3 Résultats . . . 42
3.4 Conclusion . . . 46

4 Projection de nouvelles données sur la variété 47
4.1 Projection avec une régression globale . . . 47
4.1.1 Projection par maximum de vraisemblance . . . 48
4.1.2 Projection par maximum *a posteriori* . . . 48
4.2 Projection avec une régression par blocs . . . 49
4.2.1 Utilisation d'une grille régulière . . . 49
4.2.2 Utilisation d'une grille hiérarchique . . . 49
4.2.3 Utilisation d'une grille adaptative . . . 51
4.3 Résultats . . . 51
4.3.1 Qualité de projection . . . 51
4.3.2 Projection avec occlusion . . . 53
4.4 Conclusion . . . 60

5 Classification supervisée de données 61
5.1 Classification dans un seul espace réduit . . . 61
5.1.1 Découpage d'une variété continue . . . 62
5.1.2 Différence entre un point et une classe . . . 65
5.2 Classification dans plusieurs sous-espaces réduits . . . 66
5.3 Conclusion . . . 68

6 Applications **69**
6.1 Classification supervisée de textures . . . 69
6.1.1 Descripteur de texture . . . 70
6.1.2 Apprentissage et classification de textures . . . 70
6.2 Analyse de formes . . . 73
6.2.1 Génération de champs synthétiques . . . 73
6.2.2 Apprentissage et tests statistiques . . . 74

7 Conclusion et perspectives **81**
7.1 Résumé et discussion . . . 81
7.2 Perspectives . . . 82
7.2.1 Réduction de dimension . . . 82
7.2.2 Régression . . . 82
7.2.3 Classification . . . 82

A Optimiseurs **85**
A.1 Notions d'architecture logicielle . . . 86
A.1.1 Développement dirigé par les tests (*Test-Driven Development*) . . . 86
A.1.2 Orientation objet . . . 87
A.1.3 *Design patterns* . . . 88
A.1.4 Principe de séparation . . . 88
A.2 Application à la création d'un module d'optimisation . . . 89
A.2.1 Le coeur, la classe Optimizer . . . 89
A.2.2 Les différents modules disponibles . . . 90
A.2.3 Différentiation automatique . . . 90
A.2.4 Estimation . . . 91
A.2.5 Limites du modèle . . . 92
A.3 Conclusion . . . 92

B Divers **93**
B.1 Fonction de coût associée à Isomap . . . 93
B.2 Équivalence entre *Laplacian Eigenmaps* et *Diffusion Maps* . . . 94
B.3 Paramètres de Haralick . . . 95
B.4 Les Séparateurs à Vaste Marge (SVMs) . . . 95
B.4.1 Position du problème . . . 95
B.4.2 Extensions . . . 96
B.4.3 Estimation de probabilité d'appartenance . . . 97

C Publications **99**

Bibliographie **101**

Notations

Cette thèse a été rédigée à l'aide d'une notation cohérente. Par défaut, les éléments d'une matrice sont symbolisés par un caractère normal avec deux indices, un vecteur colonne d'une matrice par une minuscule en caractère gras avec un indice et une matrice en lettre capitale grasse.

Symbole	Définition
p	Dimension de l'espace de départ
d	Dimension de l'espace réduit
n	Nombre d'échantillons d'apprentissage
$\boldsymbol{Y} = (\boldsymbol{y}_1 \dots \boldsymbol{y}_n)$	Échantillons dans l'espace de départ (d'observation)
$\boldsymbol{X} = (\boldsymbol{x}_1 \dots \boldsymbol{x}_n)$	Échantillons dans l'espace réduit
$d_1(\boldsymbol{y}_i, \boldsymbol{y}_j)$	Distance entre les points $\boldsymbol{y}_i$ et $\boldsymbol{y}_j$ dans l'espace de départ
$\boldsymbol{D_Y}$	Matrice des distances $d_1(\boldsymbol{y}_i, \boldsymbol{y}_j)$
$d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$	Distance entre les points $\boldsymbol{x}_i$ et $\boldsymbol{x}_j$ dans l'espace réduit
$\boldsymbol{D_X}$	Matrice des distances $d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$
$\boldsymbol{W} = (\boldsymbol{w}_1 \dots \boldsymbol{w}_s)$	Matrice de passage de l'espace réduit à l'espace de départ
$\boldsymbol{S}$	Matrice de similarité entre les points considérés
$\boldsymbol{I}$	Matrice identité
$\boldsymbol{0}$	Matrice dont les éléments valent 0
$\boldsymbol{1}$	Matrice dont les éléments valent 1
$\boldsymbol{H}$	Matrice de centrage ($\boldsymbol{I} - \frac{1}{n}\boldsymbol{1}$)
$\mathcal{N}(\boldsymbol{x})$	Indices des points de l'ensemble $\boldsymbol{X}$ proches du point considéré $\boldsymbol{x}$
$\mathcal{N}(\boldsymbol{x}\|\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Densité de probabilité d'une loi aléatoire gaussienne centrée en $\boldsymbol{\mu}$ de covariance $\boldsymbol{\Sigma}$ au point $\boldsymbol{x}$
$\boldsymbol{X}_{i \in \mathcal{N}(\boldsymbol{x})}$	Ensemble des points de l'ensemble $\boldsymbol{X}$ proches du point considéré $\boldsymbol{x}$
$\mathbb{I}(\boldsymbol{x})$	fonction indicatrice $\mathbb{R}^d \mapsto \mathbb{N}$
$\|.\|_F$	Norme de Frobenius

TAB. 1: Symboles récurrents

Introduction

De nombreux problèmes de traitement de données mettent en jeu des vecteurs de dimensions très importantes (dans un espace $\mathbb{R}^p$ dit d'origine). Qu'il s'agisse de classer, détecter, tester ou estimer, on est confronté dans cet espace d'origine à la malédiction de la dimension (*curse of dimensionality*) : le nombre de vecteurs de données nécessaires à un peuplement raisonnable de l'espace augmente exponentiellement avec la dimension de l'espace. En pratique, puisque le nombre de données disponibles ne répond pas à cette condition, il faudrait considérer des voisinages de taille prohibitive avant de rencontrer, dans chaque direction, les points les plus proches d'un point donné. Posées de la sorte, les tâches de classification, détection et estimation ne peuvent être réalisées avec des performances acceptables. La démarche généralement adoptée pour faire face à cet obstacle est la *réduction de dimension* des données. Il s'agit de représenter les données originales dans un espace de dimension réduite : d'une part, on souhaite respecter la structure des données, c'est-à-dire reproduire dans l'espace réduit les distances originales entre couples de points, d'autre part on souhaite éliminer les redondances présentes dans les données initiales. Du point de vue de la géométrie différentielle, on peut considérer que les données initiales se répartissent sur une variété non linéaire dont il s'agit de capturer les degrés de liberté. Le problème est alors posé comme un problème d'apprentissage de variété non linéaire.

L'objectif général de ce travail est l'apprentissage de variétés non linéaires. Typiquement, les représentations compactes auront pour objet de modéliser l'ensemble des apparences prises par un objet sous divers angles et conditions d'éclairage afin de pouvoir détecter des occurrences ultérieures de cet objet. Plus généralement et à plus long terme, l'objectif des méthodes développées dans cette thèse est la construction et l'utilisation d'atlas de formes en imagerie cérébrale. Ces atlas peuvent être utilisés en recherche clinique comme outil d'aide au diagnostic et en neurosciences comme outil d'aide à la compréhension du développement et de la dégénérescence du cerveau.

Ce travail de recherche a été mené en étroite collaboration entre le Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection (UMR CNRS-ULP 7005) et le Laboratoire d'Imagerie et de Neurosciences Cognitives (UMR CNRS-ULP 7191) dans le cadre d'un projet multi-laboratoires sur le thème de l'Imagerie et de la Robotique Médicale et Chirurgicale. Les contributions de cette thèse portent sur plusieurs aspects de l'apprentissage de variétés non linéaires. La première contribution porte sur la détermination des coordonnées réduites. Ces coordonnées sont obtenues en tant que solution d'un problème d'optimisation. Dans la veine des approches de type *multidimensional scaling*, nous proposons une fonction de coût

originale permettant de reproduire, dans l'espace réduit, les distances géodésiques de l'espace original. Ce problème d'optimisation est fortement multimodal, et les approches de type descente de gradient se révèlent insatisfaisantes. Nous proposons un algorithme original permettant de résoudre ce problème de façon satisfaisante. La seconde contribution a pour objet l'estimation d'une application permettant de mettre en correspondance l'espace réduit avec l'espace original. Ce problème de régression non linéaire de dimension multiple n'est pas abordé dans la littérature à notre connaissance. Nous proposons un algorithme de régression linéaire par morceaux, présentant suffisamment de flexibilité pour pouvoir s'adapter aux différentes configurations rencontrées avec une précision maîtrisée par l'utilisateur. La troisième contribution de notre travail est constituée par une méthode de projection d'un point sur la variété. Cette projection est effectuée dans un cadre bayésien. Elle a pour objet d'établir l'appartenance d'un objet à la variété. L'ensemble des méthodes proposées est testé sur des données de référence (Swiss-Roll, S-Curve, base COIL-20) ainsi que sur des applications pratiques (analyse de formes et de texture).

Chapitre 1

État de l'art

Sommaire

1.1 Représentation des données 3
1.2 Réduction de dimension 4
1.2.1 *MultiDimensional Scaling* (MDS) 4
1.2.2 Modèles basés sur le graphe 6
1.2.3 Autres modèles 9
1.2.4 Détermination de la dimension de l'espace réduit 10
1.3 Régression multidimensionnelle 11
1.3.1 Construction d'un modèle pendant la compression 11
1.3.2 Estimation d'une fonction entre les deux espaces 12
1.4 Projection sur un modèle 13
1.4.1 Projection à l'aide de réseaux de neurones 14
1.4.2 Utilisation des données dans l'espace original 14
1.4.3 Moindres carrés 15
1.4.4 Projection probabiliste 15
1.5 Classification 16
1.5.1 Classification spectrale 16
1.5.2 Classifieur probabiliste 17
1.5.3 Classifieur SVM 17
1.6 Objectifs 17

L'analyse de données est un problème complexe : celles-ci vivent dans des espaces de grande dimension mais peuvent généralement être décrites par un faible nombre de paramètres. Modéliser ces données par une variété (un sous-espace de $\mathbb{R}^p$) permet plusieurs actions : la visualisation des éléments dans un espace réduit, la classification (sous plusieurs formes) ou la détection de différences. Par exemple, l'ensemble des images d'un objet sous le même angle possède 3 degrés de liberté (un vecteur 3D de translation) mais est de grande dimension (allant jusqu'à plusieurs dizaines de milliers de pixels). Malheureusement, cet espace de grande

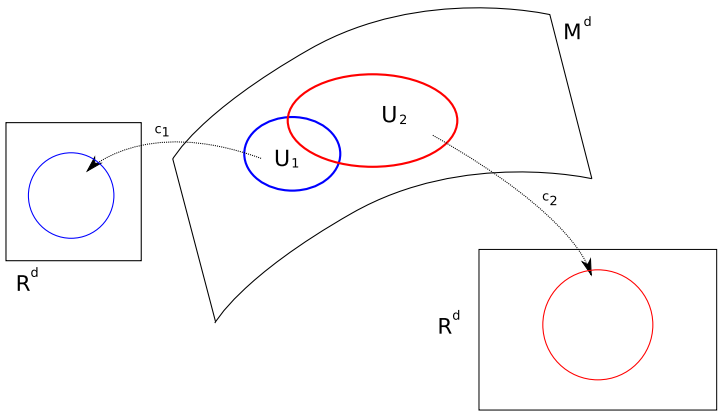


FIG. 1.1: Exemple de variété topologique M^d localement homéomorphe à $\mathbb{R}^d$.

dimension n'est pas rempli par les images de cet objet. Le but de l'apprentissage de la variété décrivant ces images d'objet peut être par la suite de tester si une nouvelle image d'objet est une image de ce même objet ou d'un autre objet. Si la non-linéarité de la variété n'est pas prise en compte lors de l'apprentissage, l'espace réduit ne sera pas non plus rempli. Ceci devra être pris en compte par la suite lors de la classification ou de la projection [VHC07]. Les outils nécessaires devant être plus complexes, il est donc plus intéressant d'apprendre de manière plus fine la variété pour travailler sur un espace le plus rempli possible.

Une telle variété ne possède pas les propriétés utilisées en mathématique euclidienne [Laf96]. Par exemple, l'image moyenne d'un objet n'est pas la moyenne euclidienne des images de ce même objet. Lorsque localement la variété est homéomorphe à $\mathbb{R}^d$ ($d = 3$ pour les images d'un même objet, correspondant aux trois degrés de translation), on parle de variété topologique. On possède alors la possibilité de définir des espaces locaux tangents à la variété, l'espace local étant de dimension d, comme la variété. Cette notion permet l'introduction de cartes locales (fig. 1.1, (U_i, c_i) étant une carte locale avec c_i l'application homéomorphe entre la variété et $\mathbb{R}^d$) et d'atlas, un atlas étant une famille de cartes. Si l'application de changement de cartes $c_i^{-1} \circ c_j$ est un difféomorphisme de classe C^k, alors l'atlas est dit C^k. La variété est différentielle lorsque la réunion de deux atlas de classe C^k est un nouvel atlas de classe C^k. Ces deux atlas sont dits compatibles, ce qui introduit une notion de classe d'équivalence des atlas. Enfin, les variétés riemanniennes définissent en outre une fonction de classe C^k entre un point sur la variété et l'espace tangent en ce point. Cette fonction permet la définition d'une métrique ainsi que le calcul de distance sur la variété, appelée aussi distance géodésique.

Si la variété analysée est connue analytiquement, il est possible de travailler directement sur la variété [Pen06] et d'effectuer des calculs en utilisant ses propriétés. En analysant l'espace sous-jacent, la répartition des données est nette, il est possible de classifier de manière

pertinente les données. De même, la variété peut être modélisée (décrite par un ensemble de fonctions) et un nouveau point peut être analysé et projeté sur la variété. Malheureusement, la variété considérée n'est généralement pas connue, seul un échantillonnage de celle-ci est connu. Des techniques dédiées de réduction de dimensionalité doivent être utilisées pour calculer un espace sous-jacent, des techniques de régression spécifiques développées pour mettre en correspondance ces deux espaces et enfin des outils de projection permettront de projeter un point sur la variété et de tester son appartenance à celle-ci.

Lorsque la variété est euclidienne, l'analyse en composantes principales (appelée aussi transformation de Karhunen-Loève ou ACP) [Jol02] est traditionnellement utilisée pour la réduction, la régression et la projection. L'ACP ne permet donc pas de remplir l'espace réduit pour une variété non linéaire. En revanche, grâce à ses hypothèses, son calcul est très rapide, d'où son intérêt qui reste très élevé, malgré des perfomances sous-optimales. Cette analyse part malgré tout sur une mauvaise hypothèse et de nouvelles techniques sont apparues pour permettre une modélisation pertinente de la variété.

L'apprentissage de variété peut donc se résumer à la résolution d'une équation $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$, avec $\boldsymbol{y}$ les échantillons des données connues et $(\boldsymbol{x}, \boldsymbol{f}())$ respectivement un jeu de paramètres (appartenant à $\mathbb{R}^d$) et la fonction de mise en correspondance entre l'espace réduit et l'espace d'origine. Dans le cas d'une variété linéaire, l'équation résolue par l'ACP est $\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$. Le problème peut être résolu en considérant les outils de réduction de dimension, les possibilités de mise en correspondance entre les deux espaces. Une fois le problème résolu, il est possible d'utiliser le modèle pour projeter des nouveaux échantillons sur la variété et les classer.

1.1 Représentation des données

Lors de l'apprentissage d'une variété riemannienne, il est nécessaire de définir la notion de distance sur celle-ci. Cela implique que les données brutes de l'ensemble ne sont pas forcément idéales pour cette tâche. Par exemple, dans le cadre de la reconnaissance faciale, les données sont traitées afin que toutes les images aient un référentiel commun (à l'aide d'une transformation affine).

Un exemple typique est l'analyse de formes. Une première approche consiste à utiliser des Procrustes [Boo96], insensibles aux transformations affines de la forme, mais elle nécessite l'intervention d'un expert pour placer des points remarquables sur la forme. Une autre méthode pourra utiliser la carte de distance [GGSK05] (après recalage pour supprimer les différences de rotation et de translation) pour détecter des différences plus subtiles. D'autres méthodes représentent directement la surface de la forme avec une représentation basée sur les fonctions harmoniques sphériques [BGK95] ou sur l'axe médian de la forme [BN78, FLPJ04]. La solution choisie pour l'analyse de formes est le champ de déformation entre une forme étudiée et une forme de référence. Ce choix permet d'utiliser une information plus complète que simplement la forme, mais aussi la variation de l'intensité au sein de la forme (les moments de l'image ont aussi cette propriété [MPR+03]). Une fois la représentation définie, une distance peut être calculée entre deux points (il est nécessaire d'avoir une métrique sur les représentations).

Dans le cadre de la classification non supervisée de données, il est important de fournir une information *a priori* sur la manière d'effectuer cette opération. En effet, si l'information pertinente est noyée dans le bruit (dimension des données trop importante, rapport signal sur bruit d'une composante pertinente trop faible, ...), la réduction de dimension ne permettra pas d'obtenir un espace dans lequel la classification sera efficace.

Si la représentation n'est pas inversible, il n'est pas possible de localiser les différences entre un point et son projeté sur la variété (voir section 5.1.2). Si deux classes ou plus divisent la variété, il est intéressant de pouvoir calculer la différence entre un point et un point ou la frontière d'une autre classe. Cela s'effectue en remontant le gradient de la fonction de classification (si cela est possible), mais il est nécessaire de pouvoir remonter à l'espace d'origine pour pouvoir visualiser cette différence [GGSK05]. Dans le cas d'une extraction d'information de texture, cette transformation inverse n'existe pas et il n'est donc pas possible de faire varier une texture continuement vers une autre classe.

1.2 Réduction de dimension

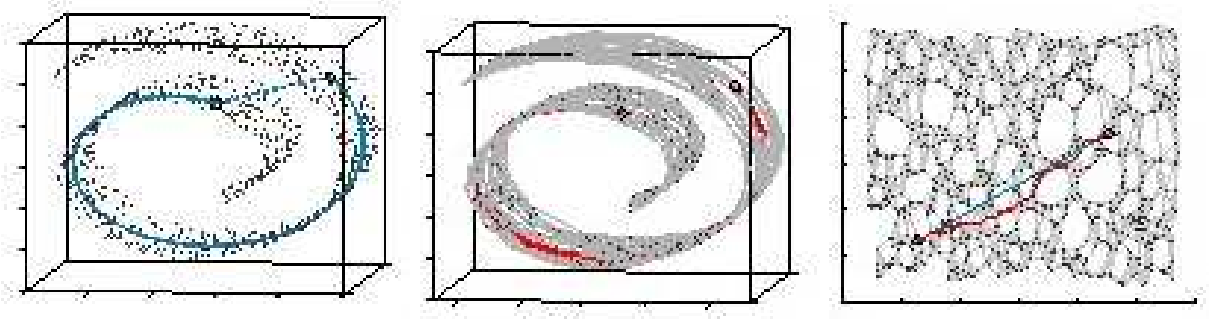
L'apprentissage d'une variété s'effectue principalement de deux manières. La première consiste à utiliser une approximation de la distance géodésique et à créer un nouvel espace de plus faible dimension dans lequel les distances géodésiques sont reportées par des distances euclidiennes. Si la réduction est trop importante, les distances géodésiques ne sont pas respectées dans l'espace réduit : il est nécessaire de trouver une méthode pour obtenir la dimension sous-jacente correcte. Par exemple, un cercle, paramétrable par une variable 1D, l'angle, ne peut être réduit dans un espace euclidien de dimension 1. L'espace des photographies d'un même objet sous le même angle est quant à lui de dimension 3. Une approche empirique consiste donc à choisir de manière arbitraire la dimension de l'espace sous-jacent grâce à de la connaissance *a priori*, mais dans le cas général, cette connaissance n'est pas fournie et une méthode déterministe doit être utilisée.

La seconde manière usuelle consiste à approcher l'opérateur de Laplace-Beltrami qui est l'extension du laplacien au cas des variétés riemanniennes. Dans ce cas, seules les distances avec les plus proches voisins peuvent être calculées. L'intérêt de cette méthode réside dans ce point : la matrice des distances entre points à calculer est creuse, contrairement à la première méthode. En revanche, la méthode n'est que locale et peut être sujette au repliement comme cela sera montré sur des exemples par la suite. Cette approche sera analysée par la suite.

Ces deux visions regroupent la majorité des approches actuelles et reposent sur le graphe des distances entre voisins [YXZ+07].

1.2.1 *MultiDimensional Scaling* (MDS)

Le MDS [Bor05] métrique permet d'approcher une mesure de dissimilarité, ici les distances géodésiques. A partir de celles-ci reportées dans une matrice carrée, le MDS métrique calcule



(a) Différence entre distance euclidienne (pointillés) et géodésique (trait plein)

(b) Construction du graphe des proches voisins et approximation de la distance géodésique

(c) Réduction de dimension (en rouge la distance géodésique approchée, en vert la distance euclidienne)

FIG. 1.2: Fonctionnement des algorithmes basés sur la distance géodésique (extrait de [TdSL00])

un jeu de coordonnées dans un espace réduit tel que ces distances géodésiques sont respectées dans l'espace réduit (fig. 1.2).

L'obtention des distances géodésiques s'effectue par une recherche des plus proches voisins (K-voisins ou fenêtre de Parzen généralement) puis la distance géodésique d'un point à un autre est calculée par le plus court chemin dans le graphe des distances entre ces voisins (un algorithme de Floyd-Warshall [Flo62] ou de Dijkstra [CLR01] pourra être utilisé). Cette technique a plusieurs inconvénients. Le premier est le temps de calcul en $O(n^2 log(n))$ (n étant le nombre de points) au minimum ainsi que la place mémoire utilisée. Certaines implémentations contournent le problème en proposant de ne travailler que sur des points remarquables [dST02], mais cela accentue le second problème, celui de la stabilité topologique [BST+]. En effet, la recherche des voisins [WZZ05, LZ08] peut entraîner un court-circuit dans la variété et par la suite une mauvaise approximation des distances géodésiques et donc de la variété (fig. 1.3). Diverses méthodes tentent de remédier à ces problèmes *a posteriori* en travaillant sur le graphe global [CC07, NA07] sans vraiment le résoudre complètement. Toutes les méthodes travaillant sur le graphe des plus proches voisins peuvent en réalité bénéficier d'un meilleur choix de ces voisins, y compris les méthodes ne travaillant pas sur les distances géodésiques. Choisir les voisins proches de l'espace tangent est l'objectif optimal, mais dans le cas de variétés à fort rayon de courbure, cet espace tangent s'écarte rapidement de la variété et la sélection des voisins est donc plus complexe.

Le MDS métrique se subdivise en plusieurs parties. Le MDS métrique dit classique utilise un critère quadratique pour respecter les distances d'entrée et la solution analytique au problème est calculée par une analyse en composantes principales. L'algorithme ainsi obtenu en utilisant des distances géodésiques est connu sous le nom Isomap [TdSL00]. La fonction optimisée est la suivante (voir B.1 pour l'explication de cette fonction), avec $\boldsymbol{D_Y}$ la matrice des distances dans

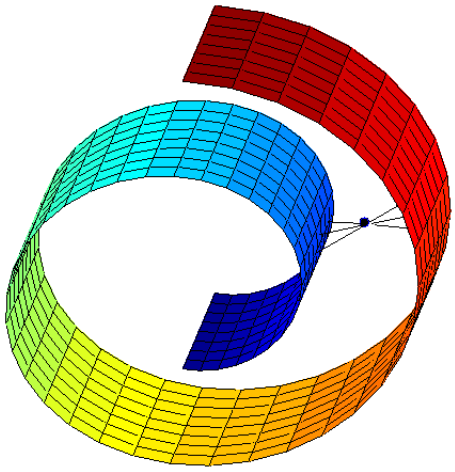


FIG. 1.3: Court-circuit dû à un point aberrant et entraînant une mauvaise estimation des distances géodésiques (extrait de [CC07])

l'espace original, $\boldsymbol{D}_{\boldsymbol{X}}$, celle dans l'espace à estimer et $\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}$ la matrice de centrage :

$$\mathcal{S}_{\text{ISO}}(\boldsymbol{Y}, \boldsymbol{X}) = \|\boldsymbol{H}(\boldsymbol{D}^2_{\boldsymbol{Y}} - \boldsymbol{D}^2_{\boldsymbol{X}})\boldsymbol{H}\|_F. \tag{1.1}$$

En optimisant cette fonction, les grandes différences entre les distances dans les deux espaces possèdent un poids quadratique (en effet, $\|M\|_F = \sqrt{\sum M^2_{i,j}}$). Or les distances pertinentes sont les distances moyennes : les faibles distances sont bruitées par le bruit sur la variété et les grandes distances sont mal approchées (elles sont calculées de proche en proche, donc sensibles au placement des points [MLX+08]). La solution plus générale consiste à faire appel au MDS métrique non classique qui optimise une fonction de coût pour résoudre le problème. Plusieurs fonctions de coût sont proposées dans la littérature [Sam69, DH97, HR02] avec diverses caractéristiques. Par exemple la fonction de Sammon [Sam69] peut s'exprimer sous la forme :

$$\mathcal{S}_{\text{SAM}}(\boldsymbol{Y}, \boldsymbol{X}) = \frac{1}{\sum_{i,j} d_1(\boldsymbol{y}_i, \boldsymbol{y}_j)} \sum_{i,j} \frac{[d_1(\boldsymbol{y}_i, \boldsymbol{y}_j) - d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)]^2}{d_1(\boldsymbol{y}_i, \boldsymbol{y}_j)}. \tag{1.2}$$

Le poids associé aux distances est hyperbolique, la fonction est donc robuste aux points aberrants, mais elle reste sensible aux erreurs sur les distances faibles.

1.2.2 Modèles basés sur le graphe

L'ensemble des inconvénients des méthodes précédentes (instabilité topologique, quantité de mémoire utilisée importante et coût en temps de calcul) peut plaider en faveur de l'utilisation d'une autre approximation de la variété riemannienne. Le principe consiste à construire une matrice (creuse) de similarité entre les échantillons puis de normaliser cette matrice sous

diverses formes et d'en extraire des vecteurs propres qui seront les coordonnées des points d'origine. La normalisation des similarités engendre une robustesse de la réduction de dimensionalité [NJW01].

Le premier algorithme est *Locally Linear Embedding* ou LLE [RS00, CY06]. L'objectif de cet algorithme est de transporter les coordonnées barycentriques de l'espace d'origine dans l'espace réduit. Chaque point est approché par ses plus proches voisins et la ligne associée dans la matrice creuse $\boldsymbol{S}$ contient les poids de chaque voisin dans cette approximation (fig. 1.4). Grâce à ces poids, une fonction de coût est introduite :

$$\Phi(\boldsymbol{X}) = \|\boldsymbol{X} - \boldsymbol{S}\boldsymbol{X}\|_F$$

Ainsi, l'objectif est clairement de retrouver un espace de faible dimension tel que les nouveaux éléments $\boldsymbol{X}_i$ soient le barycentre de leurs voisins. L'optimimum est calculé en diagonalisant la matrice $(\boldsymbol{I} - \boldsymbol{S})^T(\boldsymbol{I} - \boldsymbol{S})$ et les vecteurs propres associés aux valeurs propres les plus petites (sauf le premier car le vecteur ne contenant que des 1 a pour valeur propre 0) sont utilisés comme coordonnées.

La mesure de similarité est une mesure indirecte et collective, puisque les points les plus proches sont utilisés pour approcher un point. L'utilisation d'un poids relatif entraîne aussi des distorsions dans l'espace réduit reconstruit, comme les résultats de comparaison le montreront en partie 2.4.

Deux autres approches consistent à utiliser un noyau, par exemple gaussien, comme mesure de similarité. Les *Laplacian Eigenmaps* [BN03] ne calculent cette valeur que pour les plus proches voisins, tandis que les *Diffusion Maps* [CL06] le font pour chaque paire de points. Si le noyau est suffisamment resserré, un grand nombre de paires aura une valeur nulle et la matrice reste creuse. La littérature propose deux algorithmes différents qui sont en réalité identiques comme cela est démontré dans l'annexe B.2, les deux étant une variation du calcul sur le laplacien normalisé du graphe.

Soient la matrice (symétrique) des similarités $\boldsymbol{S}$ et la matrice diagonale $\boldsymbol{D}$ avec $d_{i,i} = \sum_j s_{i,j}$. La matrice $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S}$ est l'une des formulations du laplacien du graphe [Chu97]. Cette matrice est une matrice symétrique dont la décomposition en valeurs singulières est la suivante $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$. De plus, elle est semi-définie positive, car pour tout vecteur $\boldsymbol{v}$:

$$\boldsymbol{v}^T\boldsymbol{L}\boldsymbol{v} = \frac{1}{2}\sum_{i,j} s_{i,j}(\boldsymbol{v}(i) - \boldsymbol{v}(j))^2 \geq 0$$

Cette équation permet de mesurer la régularité du vecteur $\boldsymbol{v}$ sur le graphe (plus les valeurs sont faibles, plus le vecteur est régulier). On dit que le vecteur varie doucement sur le graphe ou qu'il suit la variété. Il est à noter que seules les valeurs fortes de $s_{i,j}$ ont un impact significatif sur cette régularité. En particulier, la régularité d'un vecteur propre est donnée par :

$$\boldsymbol{\phi}_i^T L\boldsymbol{\phi}_i = \lambda_i$$

Ainsi, les vecteurs propres $\boldsymbol{\phi}$ avec de faibles valeurs propres λ sont les plus doux. En sélectionnant les d plus faibles (sauf celui avec une valeur propre nulle), un espace réduit est

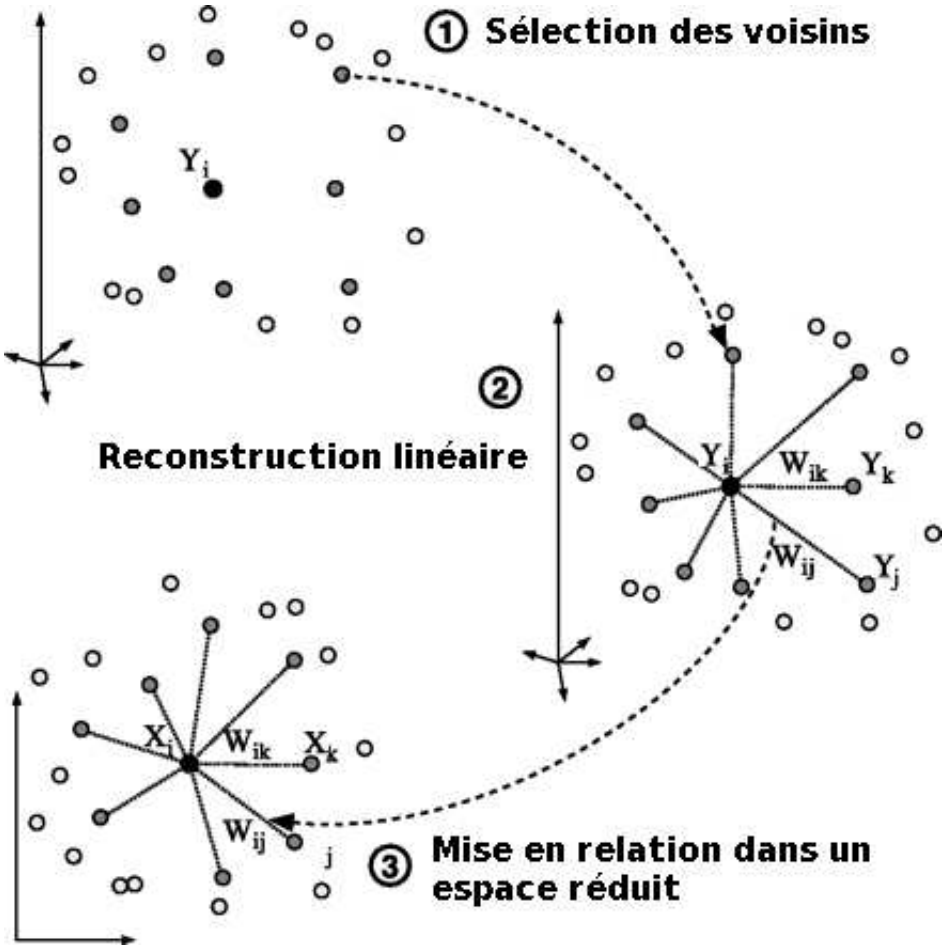


FIG. 1.4: Principe de l'algorithme LLE (extrait de [RS00])

créé. Le laplacien normalisé $\mathcal{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{S}\boldsymbol{D}^{-1/2}$ est calculé à partir du laplacien et possède des propriétés semblables [Wei99]. L'utilisation du laplacien normalisé permet en particulier d'obtenir des poids $s_{i,j}/(\sqrt{\sum_k s_{k,j}}\sqrt{\sum_k s_{i,k}})$ normalisés (des différences $(f(i) - f(j))^2$ qui possédaient un poids faible peuvent posséder un poids plus important).

Le principal inconvénient de ces deux algorithmes reste le choix des paramètres du noyau utilisé. En effet, un noyau large ou resserré entraîne des résultats pouvant être très différents. De plus, les *Laplacian Eigenmaps* ont une tendance à regrouper les points, les distances faibles et moyennes ne sont donc pas du tout respectées. Quant aux *Diffusion Maps*, la mesure des similarités entre tous les points d'apprentissage peut empêcher une bonne réduction de la variété. Enfin, le problème est à nouveau posé en terme d'une optimisation quadratique, donc sensible aux points aberrants.

Les *Diffusion Maps* peuvent s'interpréter dans un cadre probabiliste et markovien. La matrice de similarité brute (sans la normalisation selon les lignes et colonnes) est alors la matrice de transition d'un état à un autre. Celle-ci élevée à une puissance entière donne la probabilité de passer dans un état défini au bout d'un nombre fini d'itérations. Par la suite, cette matrice est normalisée pour obtenir une matrice symétrique de similarités. Un paramètre additionnel de cet algorithme est donc le choix du nombre d'itérations de la matrice de transition.

Enfin, les *Hessian Eigenmaps* [DG03] (aussi appelées *Hessian-based Locally Linear Embedding*) travaillent sur le hessien du graphe plutôt que sur le laplacien. A partir des plus proches voisins, les coordonnées sur l'espace tangent des voisins d'un point sont calculés puis un estimateur du hessien est calculé et enfin une décomposition en valeurs propres est réalisée de la forme quadratique associée. Cette méthode a été spécifiquement créée pour ne pas créer de trous dans l'espace réduit reconstruit, mais elle est beaucoup moins robuste au bruit que les autres méthodes.

1.2.3 Autres modèles

D'autres modèles ne considèrent que les distances locales [AX02] dans l'optimisation d'une fonction de coût quadratique. Ces optimisations sont difficiles et sources de nombreux minima locaux. En particulier, les distances grandes et moyennes ne servent pas pour donner de l'information, comme c'est le cas dans le MDS avec les distances géodésiques. De même les entrées nulles dans la matrice de similarité utilisée par les algorithmes précédents ajoutent une information de position de ces points. Sans ces informations, les résultats sont très aléatoires et dépendent de l'initialisation.

Une approche alternative est l'utilisation de noyaux dans l'ACP [SS02]. Ce processus, appelé ACP à noyau ou KPCA en anglais (pour *Kernel Principal Component Analysis*), effectue une décomposition en valeurs propres dans un espace de grande dimension, permettant de passer outre la linéarité de la méthode. Dans l'ACP, chaque entrée de la matrice de corrélation est le produit scalaire entre deux points considérés. Dans l'ACP à noyau, ce produit scalaire est remplacé par un noyau dit de Mercer [Mer09], comme par exemple un noyau gaussien. En considérant une application $\boldsymbol{\phi} : \boldsymbol{y} \mapsto \boldsymbol{\phi}(\boldsymbol{y})$ mettant en relation $\mathbb{R}^n$ avec un espace $\mathbb{R}^m$ de dimension potentiellement infinie, le noyau $k()$ est alors le produit scalaire entre deux mises

en correspondance, soit encore $k(\boldsymbol{y}_i, \boldsymbol{y}_j) =< \phi(\boldsymbol{y}_i), \phi(\boldsymbol{y}_j) >$. Il a été démontré que plusieurs méthodes présentées, telles que Isomap, LLE et les *Laplacian Eigenmaps*, sont en réalité des cas particuliers du KPCA [HLMS03].

Il est aussi à noter que les réseaux de neurones sont parfois utilisés pour la réduction de dimensionalité, en implémentant certaines fonctions de coût présentées plus tôt [JJ95]. En créant un réseau de neurones dont les couches d'entrée et de sortie ont un nombre de neurones égal à la dimension de l'espace d'origine et dont la couche intermédiaire possède un nombre restreint de neurones, un entraînement consistant à créer une fonction identité permettra de retrouver dans la couche intermédiaire des coordonnées dans un espace réduit. Pour une réduction non linéaire, des couches supplémentaires peuvent aussi être introduites.

1.2.4 Détermination de la dimension de l'espace réduit

Reste à définir la dimension de l'espace réduit. Cette question est essentielle pour la majorité des algorithmes précédents car sa réponse influe sur la taille du voisinage utilisé. Dans le cas de la sélection des k plus proches voisins, il est recommandé d'utiliser au moins $k = 2^d$ voisins avec d la dimension de l'espace réduit. En considérant un échantillonnage uniforme de la variété, au moins 2^d voisins sélectionnent des points sur la variété dans toutes les directions. Considérer trop d'éléments peut introduire des court-circuits dans la variété tandis que considérer trop peu d'éléments empêche une estimation correcte de la distance géodésique ou de l'opérateur de Laplace-Beltrami.

La sélection de la dimension adéquate repose généralement sur des tests utilisés à l'origine en psychologie. Le premier test est le Scree-test [Cat66]. Il consiste à analyser l'optimum de la fonction de coût optimisée (chacun des algorithmes exposés optimise analytiquement ou numériquement une fonction de coût explicite ou non) pour diverses dimensions et à choisir comme dimension de l'espace réduit l'entier à partir duquel la courbe se stabilise (voir figure 2.2a). Il est aussi possible d'utiliser un critère d'information [Aka74] pour régulariser le test et donner un résultat clair.

Un autre outil de détermination de la dimension de l'espace réduit est l'algorithme de Grassberger-Procaccia [GP83]. Il consiste à calculer une fonction $C_n(r)$ indiquant la fraction de points dont la distance est inférieure à r :

$$C_n(r) = \frac{1}{n(n-1)} \sum_{i \neq j} \mathbb{I}_{\|x_i - x_j\| < r}. \tag{1.3}$$

Par la suite, la dimension adéquate recherchée est donnée par la pente de $C_n(r)$ sur un intervalle le plus grand possible :

$$\hat{D}_{corr}(r_1, r_2) = \frac{\log C_n(r_1) - \log C_n(r_2)}{\log r_1 - \log r_2}, \tag{1.4}$$

où r_1 et r_2 définissent l'intervalle maximal où $C_n(r)$ est considéré comme étant linéaire. Cette méthode nécessite de calculer plusieurs valeurs de $C_n(r)$ mais fonctionne pour tout outil de réduction de dimension.

D'autres méthodes de sélection de cet espace existent et prennent en compte une analyse en composantes principales [FO71, BS98b], les plus proches voisins [Tru76, PBJD79, VD95, CH04] ou encore une approche fractale [CV02].

1.3 Régression multidimensionnelle

La régression résout le problème du passage de l'espace réduit à l'espace d'origine. Il s'agit donc d'une fonction de $\mathbb{R}^d$ dans $\mathbb{R}^p$. Certaines approches effectuent la régression et la réduction de dimension de manière simultanée (comme l'ACP), d'autres approches ne travaillent que sur la régression.

Des méthodes basées sur les réseaux de neurones non présentées ici calculent la régression directement [DH97] ou guident les algorithmes travaillant sur l'espace tangeant [BM05]. Un réseau de neurones créant la fonction identité propose aussi une régression en considérant la sortie de la couche intermédiaire jusqu'aux neurones de sortie.

1.3.1 Construction d'un modèle pendant la compression

L'incrément élémentaire sur l'ACP est l'ACP locale. Il s'agit d'effectuer localement une projection par ACP en considérant des groupes de points [FO71]. Cette méthode ne permet pas d'obtenir des coordonnées réduites sur toute la variété. Une extension permet de calculer une carte de l'espace réduit [Kra91] ; on parle alors de coordination de modèles locaux.

La méthode générale la plus ancienne [RSH01] crée un nombre fixe de modèles linéaires locaux, chaque modèle étant décrit par une gaussienne de variance unitaire pour la répartition des points et une gaussienne centrée pour le bruit associé. Pour coordonner les espaces entre eux, les axes de chaque modèle doivent être alignés. Le modèle général non coordonné est un mélange d'analyseurs de facteur [GH96] (*factor analyzer*) calculant la probabilité d'un point et d'un modèle :

$$\mathcal{P}(\boldsymbol{y}, s, \boldsymbol{z}_s) = \mathcal{P}(\boldsymbol{y}|s, \boldsymbol{z}_s)\mathcal{P}(\boldsymbol{z}_s|s)\mathcal{P}(s) \tag{1.5}$$

avec $\boldsymbol{y}$ le point en cours d'analyse, s l'indice du modèle utilisé et $\boldsymbol{z}_s$ les coordonnées du point dans le modèle considéré. En considérant les éléments suivants :

$$\mathcal{P}(s) = p_s \tag{1.6}$$

$$\mathcal{P}(\boldsymbol{z}_s|s) = (2\pi)^{-d/2} e^{-\frac{1}{2}\boldsymbol{z}_s^T \boldsymbol{z}_s} \tag{1.7}$$

$$\mathcal{P}(\boldsymbol{y}|s, \boldsymbol{z}_s) = |2\pi\boldsymbol{\Psi}_s|^{-1/2} e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu}_s-\boldsymbol{\Lambda}_s\boldsymbol{z}_s)^T \boldsymbol{\Psi}_s^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_s-\boldsymbol{\Lambda}_s\boldsymbol{z}_s)} \tag{1.8}$$

et en marginalisant selon les modèles existants et les coordonnées associées, on obtient :

$$\mathcal{P}(\boldsymbol{y}) = \sum_s p_s |2\pi(\boldsymbol{\Lambda}_s^T \boldsymbol{\Lambda}_s + \boldsymbol{\Psi}_s)|^{-1/2} e^{-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu}_s)^T (\boldsymbol{\Lambda}_s^T \boldsymbol{\Lambda}_s + \boldsymbol{\Psi}_s)^{-1}(\boldsymbol{y}-\boldsymbol{\mu}_s)} \tag{1.9}$$

A partir de cette équation, un algorithme EM (Expectation-Maximization) [DLR77, MB88] permet de calculer les paramètres du modèle global, à savoir p_s, $\mathbf{\Lambda}_s$ et $\mathbf{\Psi}_s$. La coordination entre les modèles est extérieure et s'effectue à l'aide de paramètres additionnels :

$$\boldsymbol{g}(s, \boldsymbol{z}_s) = \boldsymbol{A}_s \boldsymbol{z}_s + \boldsymbol{\kappa}_s \tag{1.10}$$

$\boldsymbol{g}()$ étant les coordonnées globales dans l'espace sous-jacent. Un terme additionnel dans l'optimisation de la vraisemblance du modèle, basé sur une divergence de Kullback-Leibler, est ajouté afin de déterminer les paramètres $\boldsymbol{A}_s$ et $\boldsymbol{\kappa}_s$ optimaux.

Des versions proches [Bra03, ZZ05, VRV03, TB99] peuvent permettre l'utilisation d'une gaussienne non unitaire pour décrire un ou plusieurs modèle(s) local(aux).

Les courbes principales [HS89] sont une extension des composantes principales. Au lieu d'avoir une fonction linéaire, une autre fonction continue non linéaire est utilisée (splines cubiques ou *locally weighted running-lines smoother* [Cle79]). Cette courbe est dite cohérente car chaque point de la courbe est la moyenne des points d'apprentissage qui se projettent dessus (fig. 1.5). D'un point de vue formel,

$$E(\boldsymbol{x}|\lambda_{\boldsymbol{f}}(\boldsymbol{x}) = \lambda) = \boldsymbol{f}(\lambda) \tag{1.11}$$

avec $\lambda_{\boldsymbol{f}}(\boldsymbol{x}) = sup_\lambda\{\lambda : \|\boldsymbol{x} - \boldsymbol{f}(\lambda)\| = inf_\mu\|\boldsymbol{x} - \boldsymbol{f}(\mu)\|\}$ l'index de projection et coordonnée réduite de $\boldsymbol{x}$. L'extension au cas multidimensionnel ($\mathbb{R}^d$ avec $d > 1$) est proposée, mais en réalité seul le cas $d = 2$ a été traité. Il est aussi à noter que la littérature ne permet pas d'affirmer l'existence d'une courbe principale pour un jeu de données quelconque.

Des extensions du modèle ont été proposées pour améliorer cet algorithme. Par exemple, [LT94] utilise des produits de fonctions $[s - t]_+ = (s - t)\mathbb{I}_{s>0}$ avec s une coordonnée réduite pour engendrer des régressions multidimensionnelles. [KK02] atteste l'existence d'une courbe de taille finie qu'il appelle courbe principale pour certains jeu de données, la courbe principale étant la courbe qui minimise les distances entre chaque point d'apprentissage et elle-même. Un algorithme permettant de trouver une telle courbe continue et linéaire par morceaux est proposé dans la littérature associée.

1.3.2 Estimation d'une fonction entre les deux espaces

Si l'estimation d'une fonction linéaire entre $\mathbb{R}^d$ et $\mathbb{R}^p$ est un problème simple, l'estimation d'une fonction plus complexe, même linéaire par morceaux, est plus compliquée. Diverses méthodes proposent d'estimer une fonction continue linéaire par morceaux entre $\mathbb{R}^d$ et $\mathbb{R}$ ($p = 1$ pour *Projection Pursuit Regression* [FS81], *Hinging Hyperplanes* [PS98, WS05]).

Ces deux modélisations s'appuient sur des fonctions de type $max\{l(\boldsymbol{x}, \boldsymbol{a}), l(\boldsymbol{x}, \boldsymbol{b})\}$. Pour générer deux hyperplans séparés par une charnière (*hinge*), il suffit que $l()$ soit une fonction linéaire, auquel cas la frontière entre les deux hyperplans est donnée par $\boldsymbol{b} - \boldsymbol{a}$. A chaque itération, une nouvelle fonction est ajoutée et seuls ses paramètres sont optimisés, le reste de la fonction étant considéré comme étant optimal. Les fonctions de type $[\boldsymbol{x}.\boldsymbol{a}]_+ = (\boldsymbol{x}.\boldsymbol{a})\mathbb{I}_{\boldsymbol{x}>0}$ sont aussi utilisées dans certains algorithmes équivalents. Les fonctions peuvent être étendues à $\mathbb{R}^p$

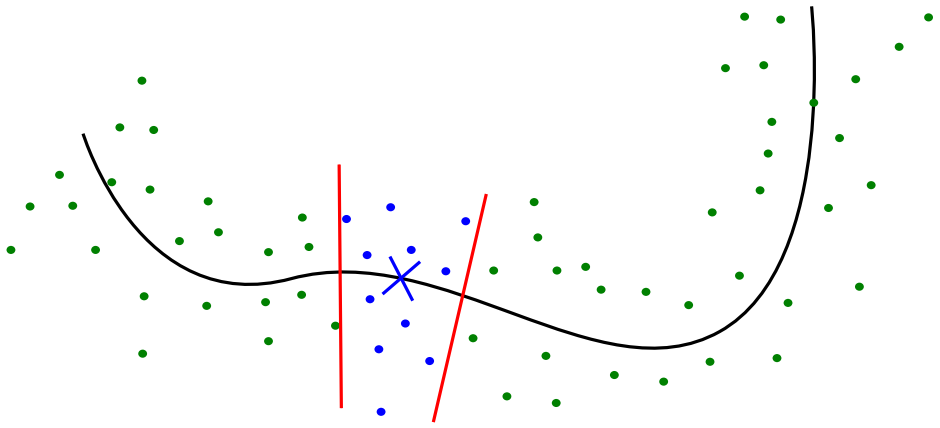


FIG. 1.5: Courbe principale (en gras) d'un nuage de points, la croix étant la moyenne des points bleus

au prix d'une complexité de calcul accrue. Le résultat final est une fonction continue linéaire par morceaux, les zones charnières étant uniquement des droites.

A partir d'interpolateurs basés sur les fonctions radiales généralisées [PG97], des régressions lisses peuvent être approchées [EL04]. Chaque coordonnée est représentée par une combinaison linéaire entre des fonctions radiales $\phi_k(\boldsymbol{x})$ (dont le centre est appris par une méthode quelconque) et une fonction linéaire des coordonnées réduites.

Une fois les paramètres appris, il est aisé de parcourir la variété, mais aussi de projeter de manière orthogonale un nouveau point sur la variété. En effet, un point est le produit matriciel entre les coefficients et le vecteur des fonctions radiales et des coordonnées réduites. En inversant cette matrice de coefficients, les valeurs des fonctions radiales $\tilde{\phi}_k(\tilde{\boldsymbol{x}})$, mais surtout les coordonnées réduites $\tilde{\boldsymbol{x}}$ peuvent être calculées. En revanche, le nombre de fonctions radiales est fixe, et la projection n'est pas forcément optimale car il n'y a pas forcément adéquation entre les coordonnées réduites calculées et les valeurs des fonctions radiales ($\tilde{\phi}_k(\tilde{\boldsymbol{x}}) \neq \phi_k(\tilde{\boldsymbol{x}})$).

1.4 Projection sur un modèle

Une fois la régression calculée, il est possible de parcourir l'espace réduit et d'obtenir les coordonnées correspondantes sur la variété. Il est aussi intéressant de tester l'appartenance d'un point à la variété. Cela est possible en testant si l'erreur obtenue en projetant le point sur la variété est faible (fig. 1.6) et/ou si les coordonnées réduites du point sont proches de celles des échantillons de la base d'apprentissage (à l'aide d'un mélange de gaussiennes par exemple). En connaissant le modèle, il est possible de tirer parti de ses caractéristiques.

Dans un premier temps, les outils basés sur les réseaux de neurones seront présentés. Puis les méthodes tirant parti des données originales et de la méthode de réduction de dimension utilisée seront exposées. Enfin, la projection sur des modèles linéaires à l'aide des moindres carrés ou d'une approche probabiliste sera décrite.

1.4.1 Projection à l'aide de réseaux de neurones

Des réseaux de neurones peuvent être entraînés pour résoudre ce problème. L'outil usuel pour cela est les *Self-Organizing Maps* [Koh01] ou SOM. La couche de sortie est un espace quantifié à partir de l'espace réduit original. L'entraînement consiste à proposer les points de départ et de forcer la sortie aux coordonnées associées dans l'espace réduit (on pourra utiliser n'importe quel outil de compression usuel). Après l'entraînement, un nouveau point proposé sera projeté à l'endroit où le neurone de sortie prend la plus grande valeur. Une autre approche probabiliste existe, les *Generative Topographic Mapping* [BS98a, TN02] ou GTM.

1.4.2 Utilisation des données dans l'espace original

Lorsque les données originales (utilisées pour déterminer un espace réduit) sont utilisées pour la projection, en considérant un nouveau point et ses voisins sur la variété, il est possible d'utiliser l'extension de Nyström [Bak77, WS01] pour calculer les coordonnées optimales (eq. 1.12) dans l'espace réduit [BPV03].

Le calcul de cette extension est le suivant. Soit $\boldsymbol{K}$ la matrice des similarités créée par un noyau $\tilde{k}(\boldsymbol{y}_i, \boldsymbol{y}_j)$ (les *Laplacian Eigenmaps* ou les *Diffusion Maps* travaillent directement sur cette matrice). La diagonalisation de cette matrice donne lieu à des vecteurs propres $\boldsymbol{U}_k$ et leurs valeurs propres associées λ_k. Par la suite, la $k$ème coordonnée réduite d'un nouvel échantillon $\boldsymbol{y}$ est donnée par

$$x_k = \frac{1}{\lambda_k} \sum_i u_{ki} \tilde{k}(\boldsymbol{y}_i, \boldsymbol{y}). \tag{1.12}$$

De là, il est aussi possible de calculer le projeté sur la variété dans l'espace original. Cette approximation est aussi celle fournie par la projection en utilisant l'ACP à noyau à un facteur près [SS02].

Sans utiliser de noyau, la solution intuitive consiste à utiliser les plus proches voisins pour interpoler un nouvel échantillon : le point original est approximé comme le barycentre de ses voisins et la projection optimale est alors le barycentre des points réduits, avec les poids calculés sur le point original. L'erreur de reconstruction est ainsi la différence entre le barycentre et le point original. Cette méthode, comme toutes celles nécessitant les données de l'espace original, possède un inconvénient de taille : il est nécessaire de conserver les points originaux sur la variété. Or, si le but est de compresser les données, il n'est pas envisageable de conserver ces données.

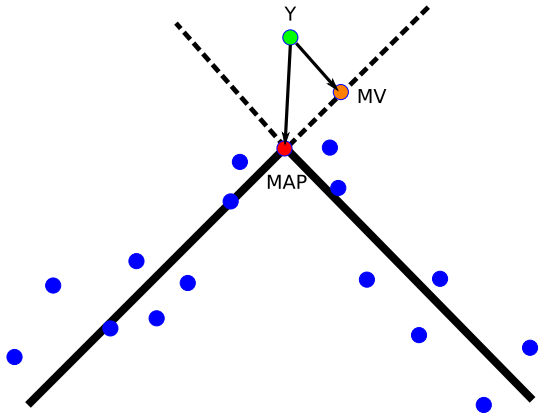


FIG. 1.6: Projection d'un point $\boldsymbol{y}$ sur la variété au croisement entre deux modèles, soit par maximum de vraisemblance (MV) soit par maximum *a posteriori* (MAP)

1.4.3 Moindres carrés

Si le modèle utilisé pour la régression est un modèle linéaire par morceaux, le nouveau point peut être projeté orthogonalement sur chacun des plans. Par la suite, la meilleure projection est la projection dont l'erreur est la plus faible. Cette extension simple de l'ACP permet d'obtenir très rapidement et de manière analytique la solution optimale au sens quadratique. En revanche, elle reste aussi sensible aux points aberrants. En utilisant une projection robuste, c'est-à-dire un coût non quadratique pour l'erreur de reconstruction, l'influence des éléments dont le rapport signal sur bruit est faible peut être diminuée.

1.4.4 Projection probabiliste

Lorsque le modèle possède une interprétation probabiliste, d'autres types de projections sont envisageables [TB99]. L'espace réduit peut apporter un terme *a priori* (fig. 1.6) forçant un point à se projeter sur un modèle local dans une zone pertinente grâce à l'utilisation de la règle de Bayes (on parle de maximum *a posteriori*).

Les points aberrants (par exemple une image présentant une occlusion, c'est-à-dire qu'une partie de l'image est masquée et remplacée par une valeur constante) peuvent alors être projetés de manière robuste en utilisant un modèle d'erreur de reconstruction plus adapté, comme une loi de Laplace ou une gaussienne généralisée. Par la suite, la recherche du projeté optimum peut s'effectuer de plusieurs manières. La plus simple est d'utiliser une grille (hiérarchique si

possible) et de choisir le candidat le plus vraisemblable. Cette solution fonctionne très bien car l'espace réduit est de faible dimension. L'algorithme du *Mean-Shift* modifié [VHC07] permet aussi de résoudre ce problème, même s'il est utilisé dans des espaces de taille bien plus importante dans la littérature.

1.5 Classification

La classification de données regroupe deux opérations distinctes : la première consiste à partionner un ensemble de données en classes (*clustering* en anglais), le second à assigner un point à une classe, pouvant avoir été définie par la première opération. Dans les deux cas, l'apprentissage de variétés est intéressant ; la réduction de dimension permet d'éviter la malédiction de la dimension et classer des données dans un espace restreint est plus facile à réaliser. De même, classer un nouveau point par projection permettra d'éviter divers problèmes de voisinage lorsque la variété ne remplit pas l'espace original.

La classification spectrale fait partie de la première classe d'opérations. Elle consiste à classer des données de manière non supervisées. Les classifieurs probabilistes et les SVMs réalisent la seconde opération, classer de nouveaux éléments à partir d'échantillons déjà classés.

1.5.1 Classification spectrale

La classification spectrale [NJW01] consiste à classer de manière non supervisée les données sur un graphe, généralement construit à partir des plus proches voisins, sans partition *a priori*. A partir d'un graphe comprenant les similarités (on parle parfois des corrélations dans la littérature) entre les points, plusieurs algorithmes permettent de créer des classes de données [Wei99] à partir des vecteurs propres de la matrice de corrélation. En revanche, le nombre de classes est connu à l'avance. En réalité, il a été démontré que la classification dans le domaine spectral revenait à travailler sur le laplacien du graphe des similarités [LL06]. Dans cet espace, la classification en utilisant les *k*-moyennes possède alors une justification théorique, selon les auteurs.

Lorsque le nombre de classes n'est pas connu à l'avance, le problème générique est connu sous le nom de partionnement par corrélation ou *Correlation Clustering* [BBC04]. Il s'agit de maximiser les similarités au sein de chaque classe et de minimiser les dissimilarités (toujours dans une même classe). Plusieurs alternatives à ce problème NP-complet sont proposées dans cet article : maximiser les similarités dans les classes ou maximiser les dissimilarités entre les classes. L'approche la plus simple consiste à sélectionner un point non classé qui est très peu similaire à un autre point non classé, et chaque point similaire à ce premier point est classé avec lui dans une nouvelle classe. Ces approximations peuvent se réaliser en temps polynomial et le nombre d'erreurs réalisées est une fonction bornée du nombre d'erreurs optimal (d'après les auteurs).

Une autre approche de ce problème est une extension des k-moyennes et recherche de manière stochastique un sous-espace de taille quelconque qui sera regroupé au sein d'une classe [HH07]. Contrairement aux k-moyennes, chaque classe est regroupée autour d'une variété

linéaire (point mais aussi plan ou hyperplan borné). Itérativement, une nouvelle classe est proposée de taille variable, ce qui permet à terme de séparer tous les éléments en différentes classes.

1.5.2 Classifieur probabiliste

Les classifieurs présentés par la suite classent un nouvel élément en fonction d'une segmentation en classes effectuée au préalable.

Le classifieur probabiliste utilise la probabilisation de l'espace réduit (par des mélanges de gaussiennes par exemple) et donc des classes pour assigner un échantillon à la classe la plus probable. A partir de la classification de l'espace réduit et de la projection $\boldsymbol{x}$ de l'échantillon $\boldsymbol{y}$, la vraisemblance de l'appartenance à chaque classe c_i est calculée par la règle de Bayes, et la classe dont la vraisemblance est maximale est choisie :

$$p(c_i|\boldsymbol{y}) \propto p(c_i)p(\boldsymbol{x}|c_i).$$

En effet, la classe d'un élément $\boldsymbol{y}$ ne dépend pas de l'erreur de projection, mais uniquement de sa projection $\boldsymbol{x}$. Ce classifieur peut aussi être utile dans le cadre de la projection sur plusieurs modèles distincts, ce qui est le cas lorsque les données appartiennent à plusieurs variétés. Dans ce cas, le point est projeté sur chaque variété, et le projeté le plus vraisemblable au sens du maximum *a posteriori* sera choisi.

Plusieurs informations additionnelles peuvent être incorporées en utilisant un classifieur bayesien dit naïf [DP97].

1.5.3 Classifieur SVM

Les SVM (pour *Support Vector Machines*) [Vap98] sont des classifieurs généralement utilisés entre deux classes (même s'ils peuvent être généralisés à plusieurs classes). A partir des vecteurs supports proches de la frontière de décision, une fonction de décision permet de décider à quelle classe un nouveau point appartient, selon son signe. Le nom francisé des SVMs est séparateur à vaste marge, en effet la frontière de décision est maximisée lors de l'apprentissage (fig. 1.7).

Les notions principales de ce classifieur sont résumées en partie B.4.

1.6 Objectifs

La majorité des outils d'apprentissage de variétés est posée en terme d'une optimisation quadratique, que cela soit les algorithmes de réduction de dimensionalité, tels que Isomap ou les *Laplacian Eigenmaps*, ou les outils de réduction/régression conjointes, tels que les méthodes de coordination de modèles locaux (basées sur des modèles gaussiens). Afin de proposer un apprentissage de variété robuste à tout type de bruit (sous condition), une méthode de réduction

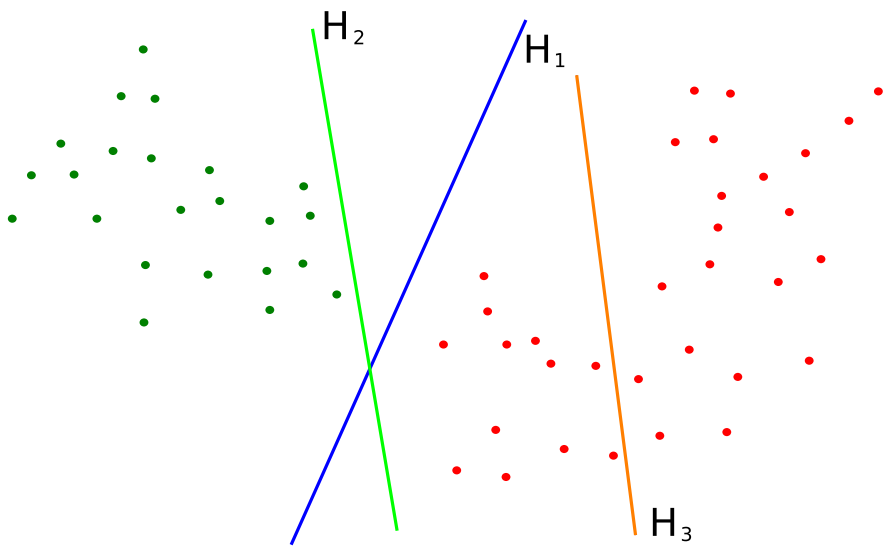


FIG. 1.7: Trois droites de séparation, H_1 la frontière avec la plus grande marge, H_2 une séparation non optimale et H_3 qui ne sépare pas les deux classes

de dimensionalité robuste au bruit a été développée. Dans l'objectif de permettre une classification efficace de nouveaux échantillons, une régression multidimensionnelle non supervisée est proposée, sans fixer à l'avance le nombre de paramètres nécessaires.

Ces outils seront testés et comparés aux outils usuels présentés dans la littérature. De plus, l'utilité de la réduction de dimensionalité sera établie dans plusieurs domaines tels que l'analyse de texture ou la classification de formes.

Chapitre 2

Réduction de dimension

Sommaire

2.1 Choix d'un critère de réduction 20
2.2 Optimisation de la fonction 20
2.2.1 Caractéristiques de la fonction à optimiser 21
2.2.2 Optimisation classique 22
2.2.3 Optimisation par incorporations successives 22
2.3 Sélection d'une dimension adéquate 22
2.4 Résultats et comparaisons 23
2.4.1 Présentation des méthodes utilisées 23
2.4.2 Résultats et évaluation 26
2.5 Conclusion 34

La première étape de l'apprentissage de variété consiste à représenter les données dans un espace réduit de dimension réduite. L'objectif de cette représentation peut être multiple : regrouper les points ou respecter les distances entre les points, pour citer les objectifs usuels. Selon l'objectif, l'algorithme de réduction de dimension utilisé pourra être différent : soit des *Laplacian Eigenmaps* ou un MDS métrique.

Les exemples proposés couvriront des variétés telles que le SwissRoll (fig. 2.3) ou le SCurve (fig. 2.4) qui sont des variétés de dimension 2 dans un espace de dimension 3. La base COIL-20 a été utilisée pour les exemples basés sur des images. Il s'agit d'images de 20 objets distincts. Pour chacun des objets, 72 images ont été prises, chacune sous un angle différent. A cet effet, les objets ont été placés sur une table tournante avec un appareil photographique fixe et une image a été acquise tous les 5 degrés (soit donc une rotation complète de l'objet). Chaque image a été segmentée pour enlever le fond de l'image et le rendre noir puis mise à l'échelle pour obtenir des vignettes de taille 128 x 128. Chaque jeu de données étant paramétré par un seul paramètre, à savoir l'angle de rotation de l'image, il représente une variété de dimension 1 dans un espace de dimension 16384.

2.1 Choix d'un critère de réduction

L'approximation choisie sera celle des distances géodésiques. En effet, comme il sera observé dans la partie sur les résultats (fig. 2.4.2), l'utilisation du laplacien ou du hessien du graphe conduit à une approximation locale dont les résultats ne sont pas adaptés aux besoins exprimés, à savoir respect de la structure globale et des distances mais aussi visualisation.

Tout comme pour l'algorithme Isomap, les distances géodésiques seront approximées par les distances aux plus proches voisins puis par la mise en oeuvre d'un algorithme du plus court chemin dans le graphe résultat. Une telle approximation a les effets suivants :

- les distances aux proches voisins sont sous-estimées par rapport à leurs distances géodésiques,
- les distances de proche en proche seront des surestimations des distances géodésiques réelles.

Il est donc nécessaire de :

- tenir compte de la sous-estimation des distances entre voisins, qui peuvent de plus être polluées par du bruit,
- considérer que le coût doit être robuste pour faire face aux points aberrants, de cette manière les grandes distances ne deviendront pas prépondérantes sur les autres distances plus faibles.

Voici la fonction de coût retenue :

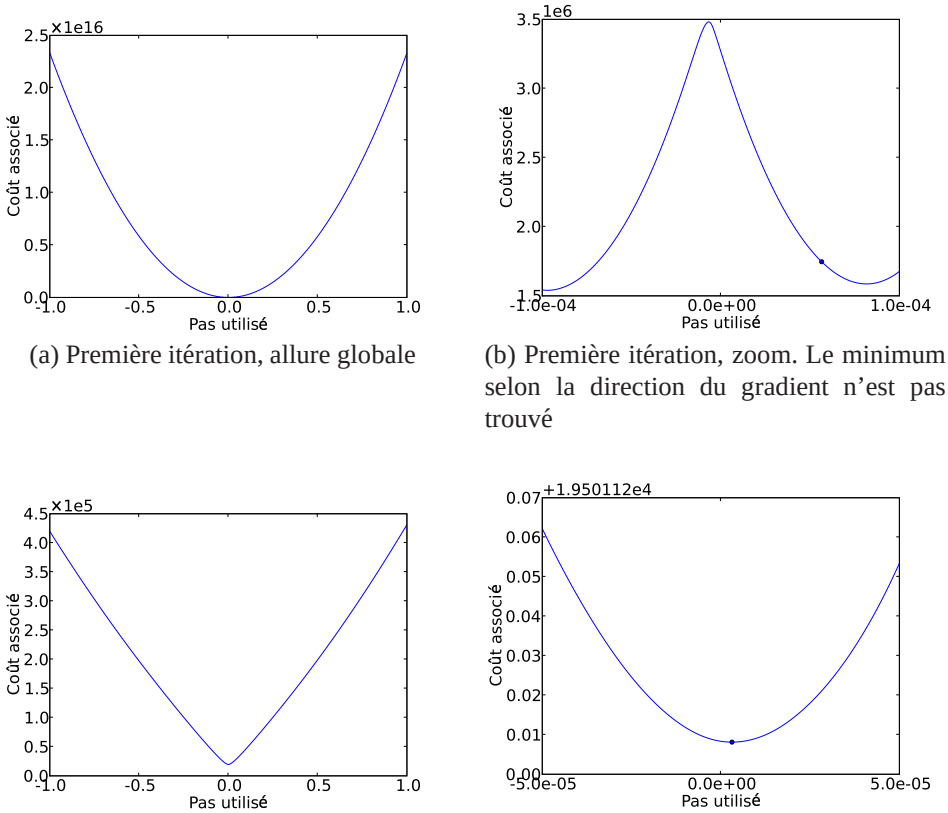
$$\mathcal{S}_{\mathrm{P}}(\boldsymbol{Y},\boldsymbol{X}) = \sum_{i,j} \sqrt{\gamma + [d_1(\boldsymbol{y}_i,\boldsymbol{y}_j) - d_2(\boldsymbol{x}_i,\boldsymbol{x}_j)]^2} \sqrt{\frac{\tau^2 + [d_1(\boldsymbol{y}_i,\boldsymbol{y}_j) - d_2(\boldsymbol{x}_i,\boldsymbol{x}_j)]^2}{\tau^2}} \frac{d_1(\boldsymbol{y}_i,\boldsymbol{y}_j)}{\sigma + d_1(\boldsymbol{y}_i,\boldsymbol{y}_j)} \tag{2.1}$$

avec d_1 la distance dans l'espace original, géodésique, et d_2 la distance dans l'espace d'arrivée, généralement euclidienne.

Le premier terme est le terme robuste, γ étant une petite valeur de telle sorte que la racine carrée reste toujours dérivable. Le second terme permet d'accélérer la convergence de l'algorithme lorsque la distance d_2 approche mal la distance d_1, le coût devenant alors quadratique lorsque la différence est supérieure à τ. Enfin, le dernier terme permet de pondérer de manière faible les faibles distances, sous-estimées ou bruitées, inférieures à σ, contrairement à la fonction de coût proposée par Sammon (cf 1.2) qui propose un poids hyperbolique pour les distances faibles.

2.2 Optimisation de la fonction

Cette fonction de coût possède un grand nombre de paramètres, le nombre de points multiplié par la dimension de l'espace réduit. Il existe plusieurs minima locaux. De plus, cette fonction de coût est insensible aux isométries, il existe donc une infinité de solutions. Afin d'éviter tout problème numérique (lorsque les coordonnées ont une moyenne très élevée devant



(a) Première itération, allure globale

(b) Première itération, zoom. Le minimum selon la direction du gradient n'est pas trouvé

(c) Dernière itération, allure globale

(d) Dernière itération, zoom

FIG. 2.1: Allure de la fonction de coût suivant la direction de descente, le point symbolisant le résultat local de la recherche du minimum.

l'amplitude de leurs variations, les calculs sont quantifiés et sujets à de grandes erreurs), les coordonnées réduites seront toujours centrées.

L'optimisation d'une telle fonction de coût est généralement effectuée à l'aide d'une descente de gradient conjugué. Ce genre d'optimisation pour une fonction de coût non convexe rencontre fréquemment un minimum local. Pour pallier ce problème, plusieurs optimisations peuvent être effectuées (*multi-start*) ou une stratégie adaptée à la fonction peut être envisagée. C'est cette dernière solution qui a été envisagée.

2.2.1 Caractéristiques de la fonction à optimiser

Lors d'une initialisation aléatoire, la fonction de coût le long du gradient est asymptotiquement quadratique en fonction de la taille du pas, comme ce qui est attendu (fig. 2.1a). Un zoom sur le graphe indique que la fonction de coût est multimodale et que la direction de descente ne donne pas lieu au meilleur minimum (fig. 2.1b). Cela indique que l'optimisation de cette

fonction peut facilement rester bloquée dans des minima locaux avec une descente de gradient classique. On note aussi que le pas nécessaire dans la direction de descente est très faible, la fonction de coût augmentant rapidement par la suite. En revanche, près du minimum global, la fonction est robuste (fig. 2.1c), indiquant que les distances sont globalement bien estimées. Lors de la dernière itération, le minimum global n'est pas atteint (fig. 2.1d). En effet, un pas faible pourrait permettrait d'obtenir un minimum plus faible, mais la variation de ce minimum est inférieure au critère d'arrêt.

2.2.2 Optimisation classique

Données : Coordonnées originales ($\boldsymbol{y}_i, i \in 1, \dots I$)
Résultat : Cordonnées réduites ($\widehat{\boldsymbol{x}}_i, i \in 1, \dots I$)
début
  Initialiser les points à zéro;
  répéter
    Ajouter un petit niveau de bruit à chaque $\boldsymbol{x}_i$;
    Optimiser tous les $\boldsymbol{x}_i$ en même temps par une descente de gradient;
    Centrer les $\boldsymbol{x}_i$;
  jusqu'à *convergence* ;
fin

Algorithme 1 : Optimisation classique

Afin d'échapper aux minima locaux, un optimiseur classique avec une descente de gradient est utilisé, avec l'addition avant chaque itération d'un bruit gaussien dont l'amplitude décroît en fonction de l'indice de l'itération (algorithme 1). La variance du bruit est aussi directement fonction de la valeur de la fonction de coût.

En revanche, cette stratégie ne permet pas d'éviter tous les optima locaux, et une seconde stratégie plus complexe est proposée.

2.2.3 Optimisation par incorporations successives

Les incorporations successives de nouveaux points permettent de mieux éviter les minima locaux (algorithme 2). En effet, le minimum global peut être facilement atteint pour un faible de nombre de points pris simultanément. Lorsqu'un nouveau point est ajouté, la contribution de ce point au coût est important et le déplacer permet d'atteindre un minimum global pour celui-ci. Une optimisation globale permet de se placer rapidement sur le minimum global de la fonction.

2.3 Sélection d'une dimension adéquate

La sélection de la dimension peut se faire naturellement à l'aide du Scree-test [Cat66]. L'optimum de la fonction de coût est calculé pour plusieurs dimensions de l'espace réduit. Lorsque

Données : Coordonnées originales ($\boldsymbol{y}_i, i \in 1, \dots I$)
Résultat : Coordonnées réduites ($\widehat{\boldsymbol{x}}_i, i \in 1, \dots I$)
début
  Choisir un sous-ensemble aléatoire de (typiquement 10) points et optimiser en utilisant l'algorithme 1;
  tant que *il reste des points à incorporer* **faire**
    Incorporer un nombre défini de points (par défaut 1, avec des coordonnées réduites aléatoires);
    Optimiser en ne déplaçant que les points nouvellement incorporés (descente de gradient);
    Optimiser tous les points simultanément (descente de gradient);
    Centrer les $\boldsymbol{x}_i$;
  fin
fin

Algorithme 2 : Algorithme d'optimisation par incorporations successives

l'augmentation du nombre de dimensions ne permet plus de diminuer de manière significative l'optimum, la dimension adéquate a été trouvée. Pour le SwissRoll (fig. 2.3), ce graphique est exposé dans la figure 2.2a pour plusieurs niveaux de bruit différents. Ce graphique montre qu'au-delà de deux dimensions, le minimum de la fonction de coût ne s'améliore plus, ou très peu par rapport à l'amélioration lors de l'addition des deux premières dimensions : le SwissRoll peut donc se modéliser par une variété de dimension 2.

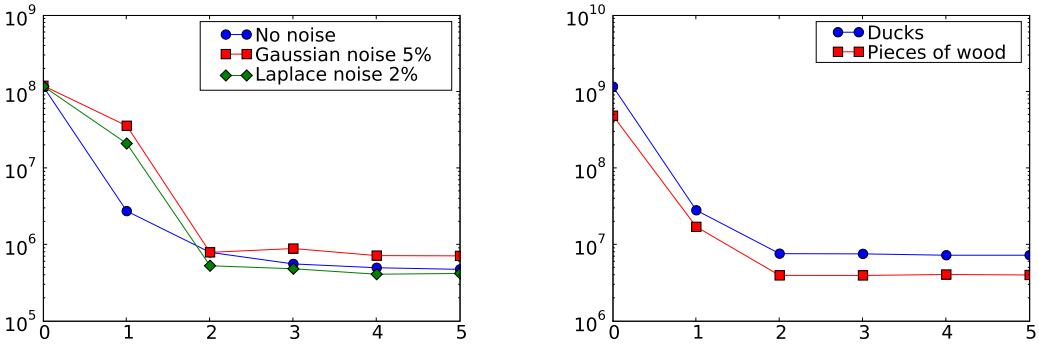
En ce qui concerne certains échantillons de la base COIL-20 [NNM96], la figure 2.2b montre que ces données sont représentables dans un espace de dimension 2, ceci par un cercle (voir fig. 2.11c).

2.4 Résultats et comparaisons

Les figures 2.3 et 2.4 exposent les deux variétés qui seront principalement utilisées comme tests. Dans chacun des deux cas, une figure expose la variété elle-même et la seconde figure propose les coordonnées d'origine utilisées pour générer la variété. Seul le SCurve est une variété dont la transformation entre l'espace de départ et celui d'arrivée est unitaire (le déterminant du jacobien de la transformation vaut un en tout point de la surface). Cette propriété permet de créer un test spécifique outre le test visuel de vérification de la structure. En effet, si la réduction de dimension réduit en dimension 2, il est alors possible de comparer les coordonnées réduites réelles et les coordonnées réduites à une transformation affine près.

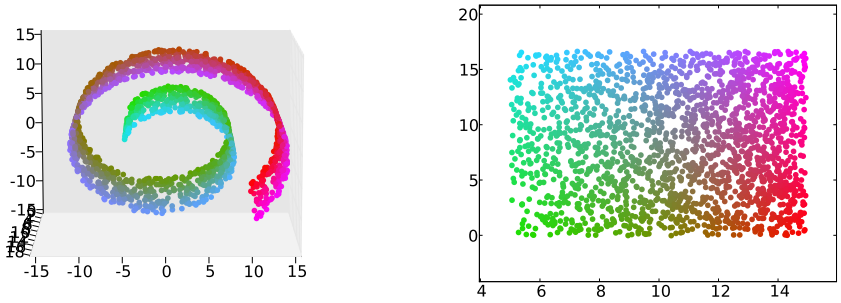
2.4.1 Présentation des méthodes utilisées

La méthode présentée précédemment sera comparée à 4 méthodes globales et à 4 méthodes locales :



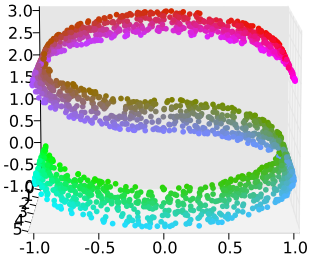
(a) SwissRoll (b) Échantillons de la base COIL-20

FIG. 2.2: Graphique du Scree-test (coût à l'optimum versus dimension)

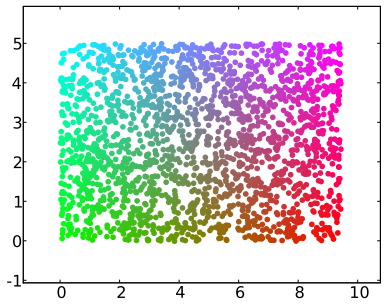


(a) SwissRoll (b) Coordonnées d'origine ayant servi à générer la variété

FIG. 2.3: SwissRoll



(a) SCurve



(b) Coordonnées d'origineavec une descente de gradient classique

FIG. 2.4: SCurve

– l'ACP [Jol02] étant l'outil usuel pour la réduction de données et la visualisation, une implémentation de référence a été utilisée afin d'avoir un étalon de comparaison ;
– Isomap [TdSL00] consiste à trouver de manière analytique à l'aide du MDS classique un jeu de coordonnées dans un espace euclidien tel que les distances géodésiques estimées sur la variété sont conservées. La fonction optimisée équivalente est :

$$\mathcal{S}_{\text{ISO}}(\boldsymbol{Y},\boldsymbol{X}) = \|\boldsymbol{H}(\boldsymbol{D}^2_{\boldsymbol{Y}} - \boldsymbol{D}^2_{\boldsymbol{X}})\boldsymbol{H}\|_F \ ;$$

– le Non Linear Mapping [Sam69] consiste à optimiser une fonction

$$\mathcal{S}_{\text{SAM}}(\boldsymbol{Y},\boldsymbol{X}) = \frac{1}{\sum_{i,j} d_1(\boldsymbol{y}_i,\boldsymbol{y}_j)} \sum_{i,j} \frac{[d_1(\boldsymbol{y}_i,\boldsymbol{y}_j) - d_2(\boldsymbol{x}_i,\boldsymbol{x}_j)]^2}{d_1(\boldsymbol{y}_i,\boldsymbol{y}_j)},$$

avec d_1 une distance sur la variété (dans le cas présent la distance géodésique approchée) et d_2 la distance euclidienne dans l'espace réduit ;
– CCA [DH97] propose une autre fonction de coût,

$$\mathcal{S}_{\text{CCA}}(\boldsymbol{Y},\boldsymbol{X}) = \sum_{i,j}[d_1(\boldsymbol{y}_i,\boldsymbol{y}_j) - d_2(\boldsymbol{x}_i,\boldsymbol{x}_j)]^2 F(d_2(\boldsymbol{x}_i,\boldsymbol{x}_j)),$$

avec $F(x) = 1$ pour $|x| < \lambda$ et 0 sinon, avec λ bien choisi.
Cette fonction est non convexe et dite récursive car le poids $F()$ qui module le coût associé à une distance permet de ne pas tenir compte lors de l'optimisation des distances estimées trop grandes (tandis que la fonction $\mathcal{S}_{\text{P}}$ utilise les distances géodésiques approchées). Malheureusement, cela entraîne une non-convexité de la fonction et une difficulté d'optimisation. L'article de référence utilise une optimisation stochastique, mais la méthode utilisée pour optimiser cette fonction est différente car la méthode initialement décrite ne permet pas d'obtenir un optimum global acceptable. Une modification de l'algorithme par incorporations successives sera utilisée, à savoir que lors de l'incorporation,

un nouveau point sera ajouté et déplacé avec un seuil λ maximum (de sorte à pouvoir placer le point proche de sa place optimale), puis tous les points sont optimisés avec un seuil λ décroissant linéairement avec le nombre d'incorporations de points entre 99% des distances et 5% des distances dans la matrice des distances géodésiques ;

– LLE [RS00] est la première des quatre méthodes locales envisagées. Son principe consiste à conserver les coordonnées barycentriques entre les points et leurs voisins respectifs (il s'agit d'une notion de similarité). La matrice des similarités $\boldsymbol{S}$ est obtenue en résolvant l'équation pour chaque point $\boldsymbol{x}_i$ par rapport à son voisinage $\mathcal{N}(\boldsymbol{x}_i)$:

$$(\boldsymbol{x}_i - \boldsymbol{X}_{j\in\mathcal{N}(\boldsymbol{x}_i)})(\boldsymbol{x}_i - \boldsymbol{X}_{j\in\mathcal{N}(\boldsymbol{x}_i)})^T \boldsymbol{s}_i = \mathbf{1}.$$

Une fois la matrice des poids normalisée (la somme des lignes est unitaire), une matrice symétrique $\boldsymbol{L}$ est calculée :

$$\boldsymbol{L} = (\boldsymbol{I} - \boldsymbol{S})^T(\boldsymbol{I} - \boldsymbol{S}).$$

Par la suite, les vecteurs propres de cette matrice sont calculés et ceux dont les valeurs propres associées sont les plus faibles (sauf le premier) permettent de définir un espace réduit ;

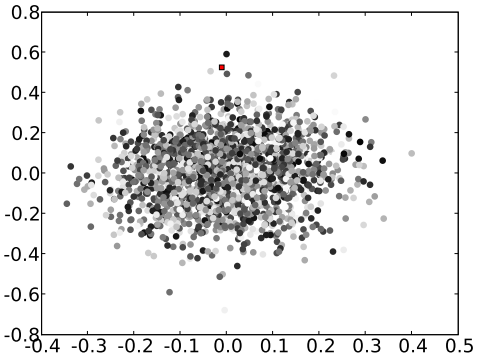
– les *Laplacian Eigenmaps* (LEM) [BN03] ont pour objectif de calculer le laplacien normalisé du graphe de similarité (avec une similarité mesurée à l'aide d'un noyau gaussien par exemple sur les plus proches voisins, mais cette fois-ci de manière symétrique). La matrice des poids $\boldsymbol{S}$ est normalisée à l'aide de la matrice diagonale $\boldsymbol{D}$ contenant la somme des poids selon les lignes ou les colonnes en la matrice du laplacien normalisé du graphe $\boldsymbol{L} = \boldsymbol{D}^{-1/2}\boldsymbol{S}\boldsymbol{D}^{-1/2}$. A partir de ce laplacien normalisé, des coordonnées peuvent être extraites tout comme pour LLE ;

– les *Diffusion Maps* (DM) [CL06] calculent elles aussi le laplacien normalisé du graphe mais en considérant les similarités entre tous les points ;

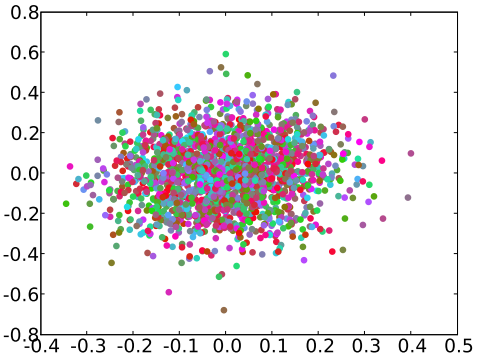
– les *Hessian Eigenmaps* (HEM) [DG03] estiment le hessien du graphe plutôt que le laplacien. Pour cela, le voisinage de chaque point est projeté sur l'espace tangent autour du point, lui-même obtenu à partir du même voisinage (à l'aide d'une analyse en composantes principales). A partir de ce voisinage, le hessien est estimé selon chaque couple de directions de l'espace tangent et ceci pour chaque coordonnée de l'espace d'origine. A l'aide de ces estimations, une matrice symétrique est créée puis décomposée en valeurs propres et les vecteurs propres dont les valeurs propres non nulles sont les plus faibles sont sélectionnés pour définir les coordonnées dans l'espace réduit. L'implémentation de référence a été utilisée lors des comparaisons.

2.4.2 Résultats et évaluation

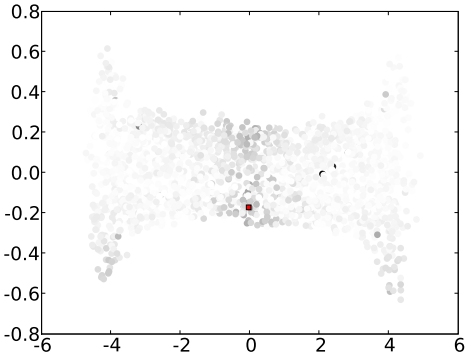
Avant toute chose, les paramètres de $\mathcal{S}_{\text{P}}$ ont été fixés et sont identiques pour tous les essais. γ a été fixé à 10^{-7}, τ à 60% des distances géodésiques estimées et σ à 1% de ces distances. D'autres algorithmes tels que les *Laplacian Eigenmaps* ou les *Diffusion Maps* nécessitent de changer de paramètres selon le jeu de données utilisé pour trouver une solution optimale.



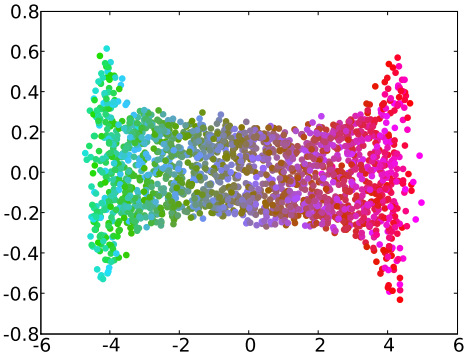
(a) Itération 1 : stress



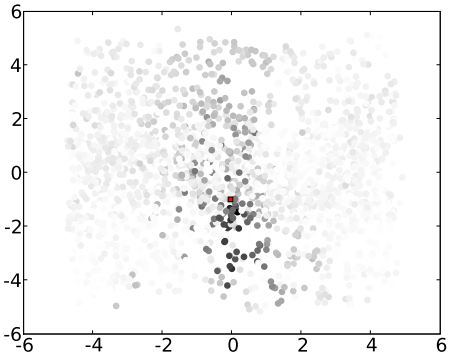
(b) Itération 1 : structure



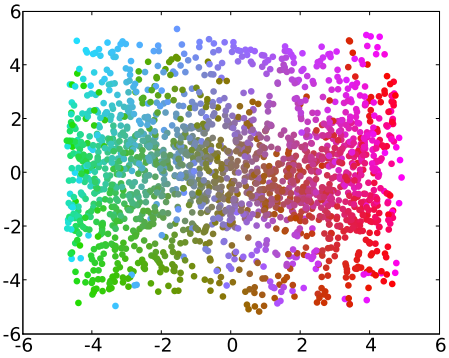
(c) Itération 10 : stress



(d) Itération 10 : structure

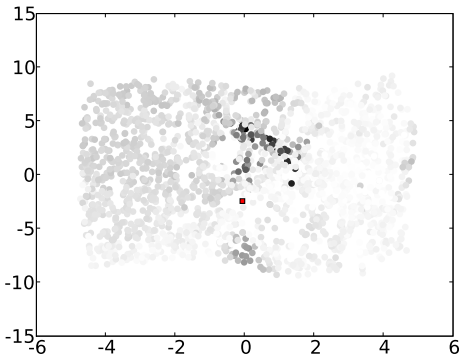


(e) Itération 30 : stress

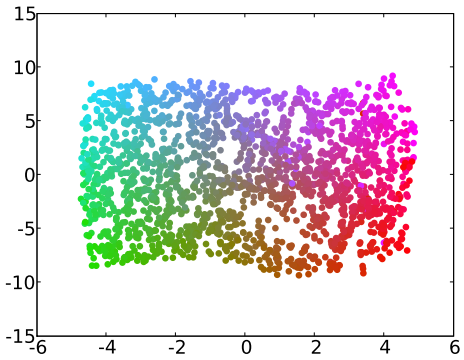


(f) Itération 30 : structure

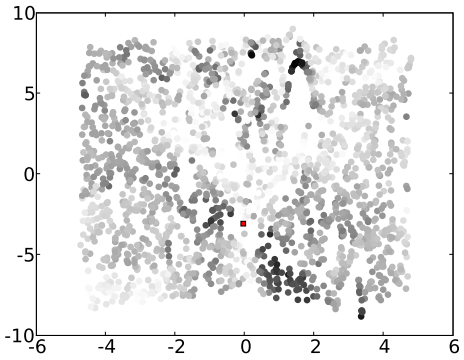
FIG. 2.5: Evolution du stress pour un point central et évolution de la structure



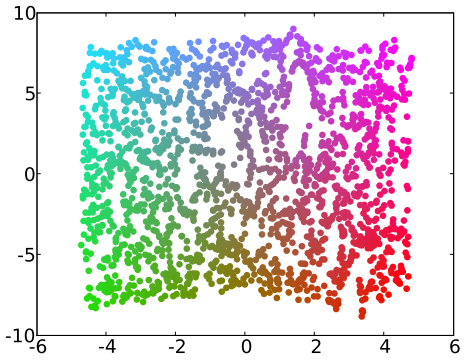
(a) Itération 50 : stress



(b) Itération 50 : structure



(c) Itération 70 : stress



(d) Itération 70 : structure

FIG. 2.6: Evolution du *stress* pour un point central $\boldsymbol{y}_0$ (une couleur foncée représente une forte différence $|d_1(\boldsymbol{y}_i, \boldsymbol{y}_0) - d_2(\boldsymbol{x}_i, \boldsymbol{x}_0)|$) et évolution de la structure (les couleurs reprennent celles de la figure 2.3, permettant de vérifier la réduction correcte de la variété)

Avant d'exposer les résultats, les figures 2.5 et 2.6 exposent l'évolution du coût (parfois aussi appelé *stress* dans la littérature anglaise) $|d_1(\boldsymbol{y}_i, \boldsymbol{y}_j) - d_2(\boldsymbol{x}_i, \boldsymbol{x}_j)|$ entre un point i du Swissroll et l'ensemble des autres points j ainsi que la structure des points (avec la carte des couleurs associée au SwissRoll). Une couleur pâle indique une grande adéquation entre les deux distances. Au départ, les points étant tirés aléatoirement, il y a peu d'adéquation. Très rapidement, l'espace réduit prend forme de manière adéquate, les points extrêmes par rapport au point considéré possèdent une distance bien estimée, ce qui n'est pas le cas des points centraux qui seront déplacés petit à petit lors des itérations suivantes. Lors des dernières itérations, les points les plus critiques sont proches des zones peu échantillonnées, lorsque la distance entre plusieurs voisins presque proches est mal estimée.

Deux tests permettent d'évaluer la réduction de dimension.

La première est la comparaison visuelle. Il s'agit de vérifier que la structure de l'espace réduit correspond à la structure d'origine (exposée fig. 2.3b pour le SwissRoll et fig.2.4b pour le SCurve). Le SCurve a donc été compressé par chacune des 9 méthodes présentées et le résultat est donné dans les figures 2.7 pour la version non bruitée et 2.8 pour la version avec 5% de bruit (par rapport à la variance totale des coordonnées des points).

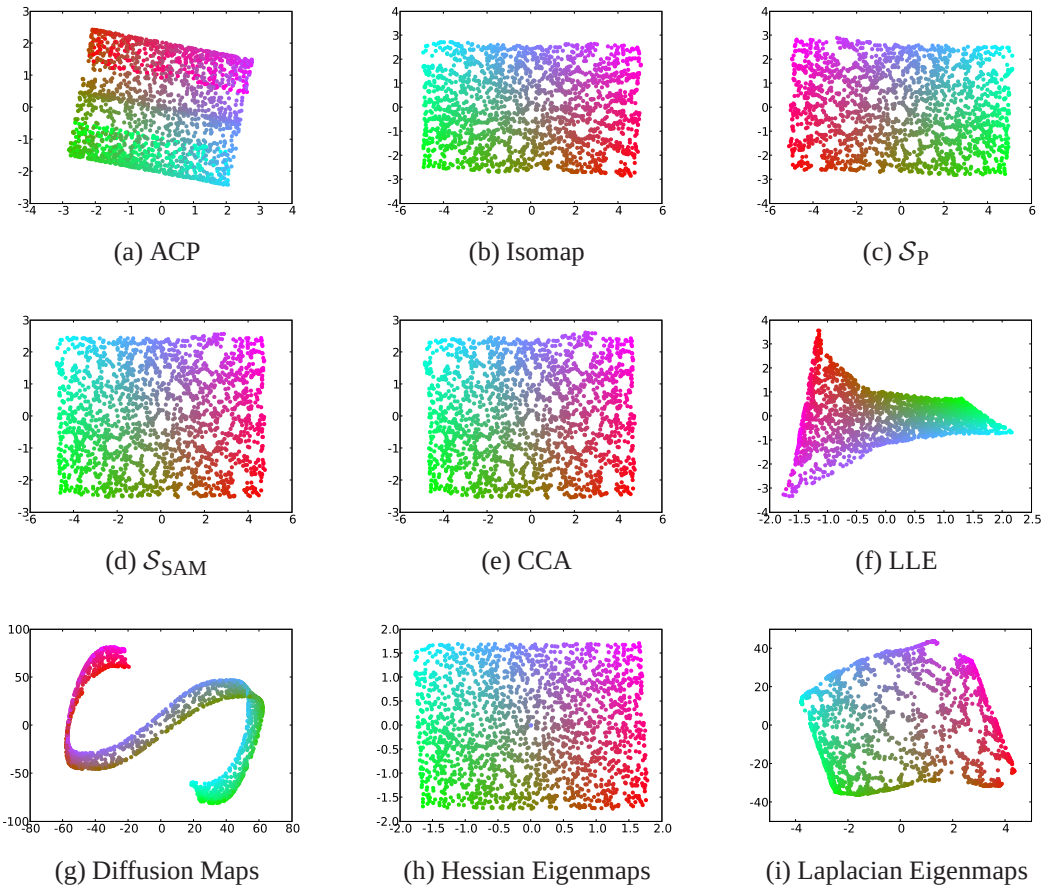
Visuellement, seuls l'ACP, LLE ou les *Diffusion maps* ne donnent pas un résultat clairement satisafaisant, tandis que les *Laplacian Eigenmaps* respectent la structure en agglomérant les points entre eux. Les *Hessian Eigenmaps* semblent donner le résultat le plus régulier, puis la méthode présentée (moins de trous agrandis dans l'espace réduit) puis CCA, $\mathcal{S}_{\text{SAM}}$ et Isomap.

En présence de bruit, seuls les *Hessian Eigenmaps* s'effondrent visuellement. Cela se comprend car l'approche est complètement quadratique et donc sensible au bruit. En revanche, des fissures plus nettes apparaissent sur $\mathcal{S}_{\text{SAM}}$, à cause du côté récursif de la fonction de coût (une fissure qui est apparue ne peut que s'agrandir).

Les représentations de l'espace réduit pour le SwissRoll (fig. 2.9) entraînent globalement une conclusion identique à celle pour le SCurve. En revanche, les *Laplacian Eigenmaps* ne permettent pas d'obtenir de résultat satisfaisant, ce qui se retrouve dans la littérature [LZ08].

Pour le deuxième test (tab. 2.1), la différence entre les coordonnées originales et les coordonnées réduites est calculée. Il s'agit de la norme de Frobenius de la matrice $\boldsymbol{X}^* - \boldsymbol{W}\hat{\boldsymbol{X}}$ avec $\boldsymbol{X}^*$ les coordonnées originales, $\hat{\boldsymbol{X}}$ les coordonnées réduites estimées et $\boldsymbol{W}$ la matrice de transformation affine minimisant cette erreur. Un bruit gaussien ou laplacien est ajouté à la variété (en pourcentage de la variance des coordonnées de la variété) et un dernier bruit de perturbation impulsionnel est ajouté à la matrice des distances géodésiques (le bruit impulsionnel est un bruit laplacien ajouté à 12.5% des distances dans la matrice, la variance du bruit étant déterminée par rapport à la différence des distances géodésiques et euclidiennes dans l'espace réduit $d_1 - d_2$ où d_2 a été calculé par $\mathcal{S}_{\text{P}}$).

Ce test indique clairement quelles sont les méthodes de compression qui tentent de conserver les distances géodésiques. Si tel est l'objectif désiré, seules Isomap, $\mathcal{S}_{\text{P}}$, $\mathcal{S}_{\text{SAM}}$ et $\mathcal{S}_{\text{CCA}}$ peuvent être envisagées. $\mathcal{S}_{\text{CCA}}$ est tout de même très instable du fait de la récursivité de la fonction et c'est ce qui explique les résultats surprenants avec un bruit gaussien à 5%. Même si les *Hessian Eigenmaps* offrent un résultat très bon en l'absence de bruit, les performances s'effondrent dès que celui-ci est légèrement présent. Cela est confirmé par la littérature [DG03], même si cette



(a) ACP (b) Isomap (c) $\mathcal{S}_{\mathrm{P}}$

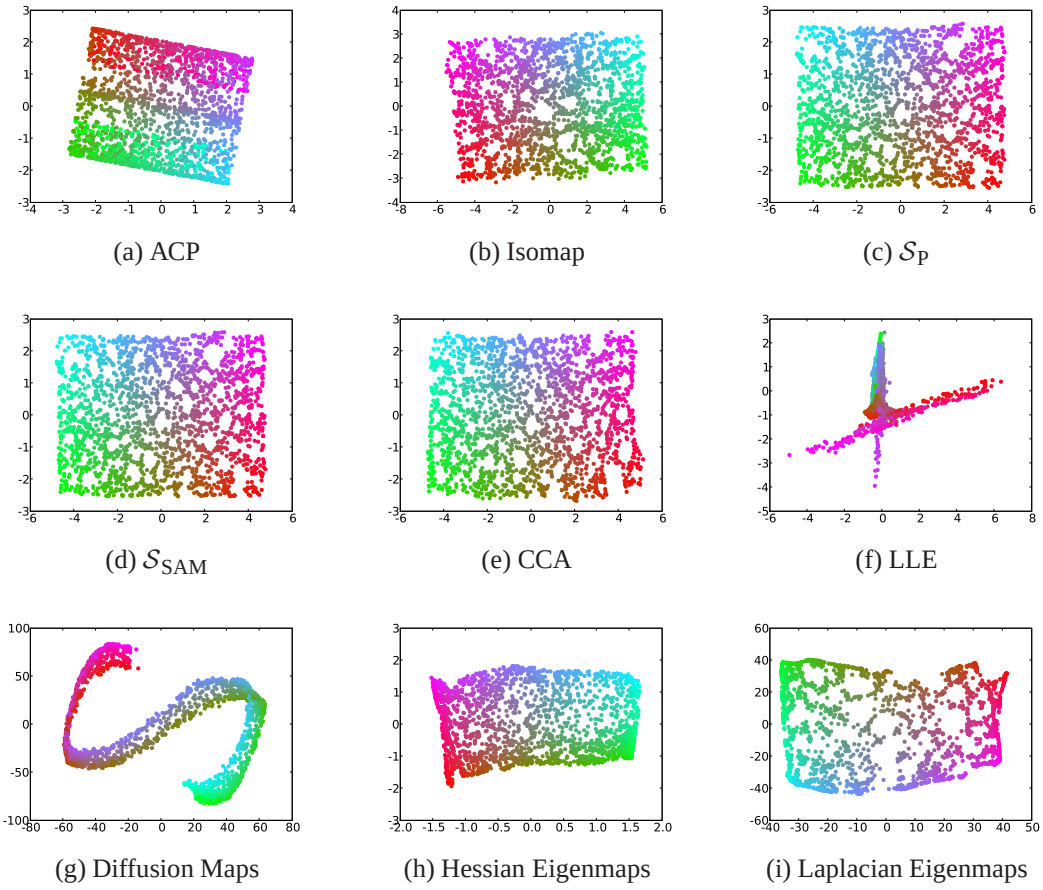
(d) $\mathcal{S}_{\mathrm{SAM}}$ (e) CCA (f) LLE

(g) Diffusion Maps (h) Hessian Eigenmaps (i) Laplacian Eigenmaps

FIG. 2.7: Espace réduit pour le SCurve non bruité

Bruit	ACP	Iso.	$\mathcal{S}_{\mathrm{P}}$	$\mathcal{S}_{\mathrm{SAM}}$	$\mathcal{S}_{\mathrm{CCA}}$	LLE	LEM	DM	HEM
Aucun	43.6	3.01	2.29	2.61	3.01	40.1	21.1	67.5	3.05
Gaussien 5%	43.6	8.55	2.94	2.60	6.22	90.2	23.5	67.7	18.6
Laplacien 2%	44.4	7.01	6.46	6.10	4.70	69.2	23.5	67.5	20.5
Impuls. 200%	n/a	3.80	2.93	3.22	3.09	n/a	n/a	n/a	n/a

TAB. 2.1: Norme de la matrice entre les coordonnées originales et les coordonnées réduites après transformation affine

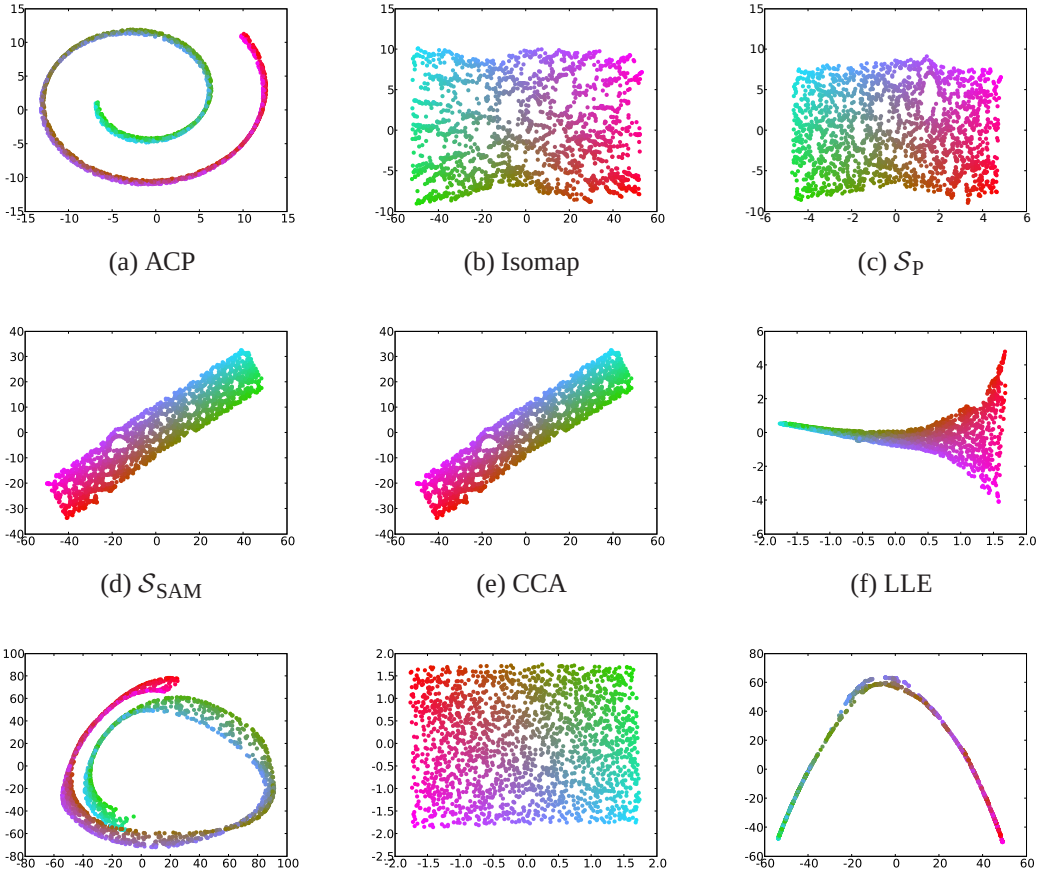


(a) ACP (b) Isomap (c) $\mathcal{S}_{\mathrm{P}}$

(d) $\mathcal{S}_{\mathrm{SAM}}$ (e) CCA (f) LLE

(g) Diffusion Maps (h) Hessian Eigenmaps (i) Laplacian Eigenmaps

FIG. 2.8: Espace réduit pour le SCurve bruité par 5% de bruit gaussien

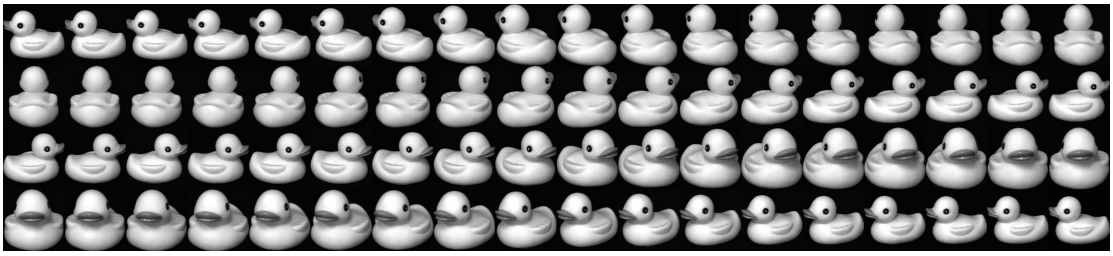


(a) ACP (b) Isomap (c) $\mathcal{S}_{\mathrm{P}}$

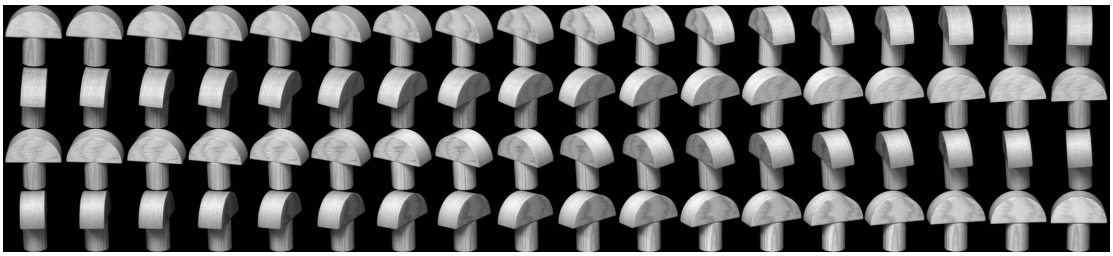
(d) $\mathcal{S}_{\mathrm{SAM}}$ (e) CCA (f) LLE

(g) Diffusion Maps (h) Hessian Eigenmaps (i) Laplacian Eigenmaps

FIG. 2.9: Espace réduit pour le SwissRoll non bruité



(a) Extrait 1 (Canards)



(b) Extrait 11 (Pièces de bois)

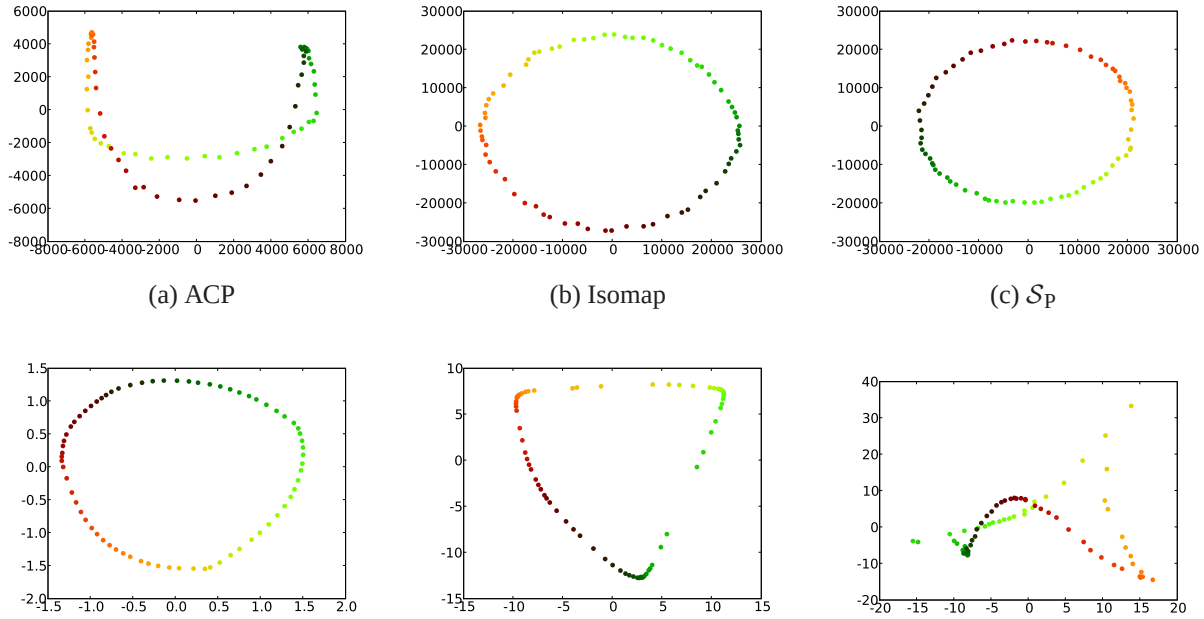
FIG. 2.10: Extraits de la base COIL-20

méthode possède l'avantage de ne pas trouer la variété comme les autres méthodes.

Enfin, en ce qui concerne les temps de calcul, les méthodes ne calculant pas les distances géodésiques sont les plus rapides, de l'ordre de la minute pour 2000 points sur le SCurve ou le SwissRoll. Les méthodes basées sur ces distances en optimisant une fonction de manière numérique sont plus longues que Isomap (12 minutes pour l'Isomap contre 70 minutes pour $\mathcal{S}_{\mathrm{P}}$ avec le premier algorithme ou $\mathcal{S}_{\mathrm{SAM}}$, et 36 heures avec l'approche avec incorporations successives, 48 heures pour $\mathcal{S}_{\mathrm{CCA}}$).

La compression d'un exemplaire de la base COIL-20 [NNM96] permet de détecter la manière de compresser une variété de dimension 1 représentant un angle dans un espace réduit. Cet angle peut être représenté en 2 dimensions par un cercle. Certains outils de réduction de dimension ont été testés sur le premier extrait (fig. 2.10a) de la base, un canard tournant sur lui-même. Les résultats (fig. 2.11) montrent que les outils respectant la distance géodésique permettent de retrouver ce cercle, ce qui n'est pas le cas des outils locaux tels que LLE, les *Laplacian Eigenmaps* ou les *Diffusion Maps*. Un autre extrait (fig. 2.10b) permet d'obtenir des résultats équivalents.

Dans tous les exemples considérés, la distance euclidienne a été utilisée pour approcher la distance géodésique sur la variété. Selon les jeux de données, cette distance pourra être remplacée par une autre distance plus pertinente.



(a) ACP (b) Isomap (c) $\mathcal{S}_{\mathrm{P}}$

(d) LLE (e) Laplacian Eigenmaps (f) Diffusion Maps

FIG. 2.11: Espace réduit pour les canards de la base COIL-20

2.5 Conclusion

Une méthode robuste de réduction de dimension a été présentée avec des comparaisons avec les méthodes standard actuelles. L'ACP, la méthode généralement utilisée pour résoudre ce problème, n'est pas capable de reproduire la structure des données dans un espace de dimension minimale, contrairement aux autres méthodes présentées. Les méthodes travaillant sur le laplacien du graphe ne respectent pas les distances géodésiques sur la variété mais souffrent aussi d'autres inconvénients critiques tels que la sélection des paramètres ou l'incapacité de trouver un espace réduit correct pour une variété. En revanche, elles ont comme avantage indéniable la capacité à travailler sur une matrice creuse et donc à permettre l'utilisation de plusieurs dizaines de milliers d'échantillons. Les *Hessian Eigenmaps*, si elles présentent un intérêt en l'absence de bruit, présentent une sensibilité qui empêche leur utilisation sur des variétés réelles.

Les méthodes cherchant à conserver les distances d'un espace à l'autre obtiennent logiquement les meilleurs résultats. Isomap permet d'obtenir rapidement un résultat, malgré une sensibilité aux points aberrants. $\mathcal{S}_{\mathrm{CCA}}$ est une évolution d'Isomap sans tenir compte des points éloignés, ce qui la rend très sensible aux problèmes d'optimisation. $\mathcal{S}_{\mathrm{SAM}}$ quant à elle est une solution robuste mais augmentant la sensibilité des petites distances. Finalement, $\mathcal{S}_{\mathrm{P}}$ est une solution robuste dans toutes les situations, en l'absence de bruit ou en présence de fortes perturbations des distances.

Chapitre 3

Régression et modélisation des données

Sommaire

3.1 Régression globale 36
3.1.1 Régression simple 36
3.1.2 Régression avec optimisation d'une fonction de coût 37
3.1.3 Approximation d'un nouveau point 40
3.2 Régression par blocs 41
3.2.1 Position du problème 41
3.2.2 Partitionnement par corrélation 41
3.3 Résultats 42
3.4 Conclusion 46

La régression consiste à définir une fonction d'approximation entre l'espace réduit $\mathbb{R}^d$ et l'espace de départ $\mathbb{R}^p$. La fonction qui sera définie ici est une fonction affine par morceaux.

Contrairement aux techniques mêlant réduction et régression, la régression considérée ici n'utilise pas les données dans l'espace d'origine une fois le modèle appris. Ceci est aussi la différence entre une fonction d'approximation et une fonction d'interpolation, cette dernière ne permettant pas de compresser la variété représentée par ses échantillons.

Un grand nombre d'algorithmes de régression peuvent se séparer en n problèmes de régression monodimensionnels. L'algorithme présenté ici tire parti de la multidimensionnalité grâce à un modèle d'erreur global permettant de gérer le nombre de modèles.

Dans un premier temps, une méthode de régression globale entre les deux espaces sera présentée. Par la suite, une méthode de régression par blocs basée sur la première méthode sera proposée. Enfin, ces différentes méthodes seront testées sur les jeux de données utilisés pour la réduction de données.

Afin de simplifier les notations, les coordonnées d'origine sont augmentées d'une constante valant 1 (*i.e.*, $\boldsymbol{X} \equiv (\boldsymbol{X}, 1)$).

3.1 Régression globale

Le modèle simple consiste à séparer l'espace de départ en plusieurs régions, chacune d'elles étant associée à un modèle affine. L'appartenance d'un point à l'un des modèles affines est décidée par une fonction d'appartenance $\mathbb{I}(\boldsymbol{x})$. Celle-ci indique quel est le modèle à utiliser à l'aide d'une estimation de densité par la méthode des noyaux ou *kernel-density estimation* [DHS00]. La probabilité d'appartenance est donné par un mélange de gaussiennes, chaque gaussienne ajoutant sa probabilité à celle du modèle auquel elle est associée. Par la suite, on parlera simplement de mélange de gaussiennes en se rapportant à cette méthode.

Le problème à résoudre peut donc se mettre sous la forme :

$$\hat{K}, \hat{\boldsymbol{W}}_{1,\dots,k}, \hat{\mathbb{I}}() = argmin_{K,\boldsymbol{W}_{1,\dots,k},\mathbb{I}()} \sum_j F(\boldsymbol{y}_j - \boldsymbol{W}_{\mathbb{I}(\boldsymbol{x}_j)}\boldsymbol{x}_j) \tag{3.1}$$

avec $\boldsymbol{W}_k$ les équations des modèles affines et K leur nombre total.

Sous l'hypothèse d'un bruit gaussien isotrope commun à tous les modèles linéaires, $F()$ est la norme 2 d'un vecteur. L'interêt d'utiliser cette formulation est sa simplicité de mise en oeuvre et d'optimisation. C'est aussi à cet endroit qu'est introduite la multidimensionnalité de l'algorithme, toutes les coordonnées étant modélisées en même temps et non séparément.

3.1.1 Régression simple

Le premier algorithme proposé (Algo. 3) est un algorithme stochastique visant à créer une régression rapidement.

Entrées : coordonnées originales et réduites ($\boldsymbol{y}_i, i \in 1,\dots I$ et $\widehat{\boldsymbol{x}}_i, i \in 1,\dots I$)
Sorties : étiquettes (l_i), matrices des modèles ($\boldsymbol{W}_k$) et variance du bruit (variance des $\hat{\epsilon}_i$)
début
  tant que *il existe un point dont les voisins ne sont pas étiquetés* **faire**
    Prendre un point au hasard dont les voisins ne sont pas étiquetés;
    Calculer la matrice $\boldsymbol{W}_k$ décrivant le voisinage ① ;
    Mettre à jour les étiquettes et toutes les matrices ②;
    Supprimer tout modèle (étiquette et matrice) ayant trop peu de points ③;
  fin
fin

Algorithme 3 : Mise en correspondance linéaire par morceaux 1 (*Piecewise Linear Mapping* PLM1)

① Soit $\boldsymbol{Y}_i$ la matrice composée des $\boldsymbol{y}_j$ voisins de $\boldsymbol{y}_i$ et $\boldsymbol{X}_i$ la matrice correspondante des $\boldsymbol{x}_j$ associés. La matrice $\boldsymbol{W}$ est estimée à partir de l'équation $\boldsymbol{Y}_i \simeq \boldsymbol{W}\boldsymbol{X}_i$ par moindres carrés.
② Un point est attribué à un modèle linéaire si le carré de la norme de l'erreur de reconstruction est inférieur à un facteur fois la variance associée au modèle linéaire. Les

étiquettes des points déjà étiquetés sont remises en cause. Après les mises à jour, les modèles sont recalculés.
③ Si un modèle linéaire a peu de points, la matrice qui le décrit ne peut pas être calculée. Dans ce cas, le modèle linéaire est supprimé.

Cet algorithme n'optimise pas la fonction de coût de manière explicite et ne permet pas de choisir un critère d'arrêt basé sur le nombre de modèles ou sur l'erreur résiduelle. De plus, des points peuvent ne pas être assignés à un modèle, ce qui n'est pas gênant lorsqu'un grand nombre de points d'apprentissage est donné. En revanche, un tel algorithme est très rapide et permet donc d'obtenir une première approximation rapidement.

La figure 3.1 montre l'évolution de l'algorithme. La première colonne expose dans l'espace réduit l'évolution des étiquettes associées aux points, une couleur par étiquette, et la couleur noire indique qu'aucun modèle n'est assigné au point. Lors de la première itération (fig. 3.1a), seule une fraction des points est assignée à un modèle, et donc seule cette fraction peut être modélisée (fig. 3.1b). Après la création d'un modèle local, celui-ci est propagé pour englober une partie plus importante des points (fig. 3.1d). A la fin de l'apprentissage, la variété n'est pas modélisée de manière fine (fig. 3.1f), un nombre relativement faible de modèles ayant été générés. Il est aussi à noter qu'il peut rester des points non étiquetés à la fin de l'apprentissage (fig. 3.1e), si aucun modèle ne peut les décrire correctement et si leurs voisinages ont déjà été étiquetés.

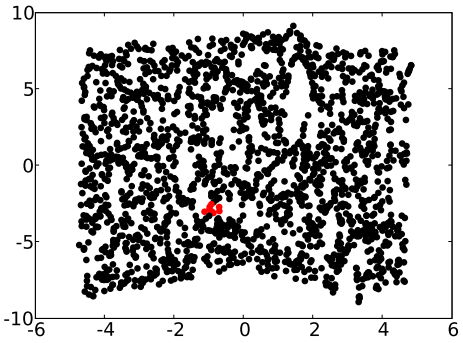
3.1.2 Régression avec optimisation d'une fonction de coût

Afin d'optimiser de manière explicite le problème 3.1, un algorithme basé sur la vraisemblance est proposé (algo. 4). Contrairement à l'algorithme précédent, tous les points sont toujours assignés à un modèle. La sélection d'un point où sera ajouté un modèle est effectuée par vraisemblance, *i.e.,* les points qui ont une erreur de modélisation importante (peu vraisemblables) sont utilisés pour générer un nouveau modèle. La boucle interne d'optimisation de la vraisemblance est une version modifiée d'un algorithme EM [GH96]. En effet, la première étape consiste à supprimer les modèles qui ne pourront pas être mis à jour, puis la phase *Expectation* met à jour les modèles et finalement la phase *Maximization* maximise la vraisemblance, permettant d'optimiser l'équation 3.1.

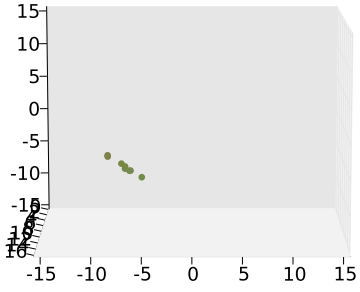
Cet algorithme permet de contrôler de manière plus fine la qualité de la régression. Un nombre de modèles maximum peut être spécifié en sélectionnant un critère d'information adéquat, mais l'optimisation est bien plus longue que celle utilisant l'algorithme 3. La connexité des modèles est assurée à la fin de la convergence. Si les points assignés à un modèle ne sont pas connexes (en considérant les n plus proches voisins), le modèle est découpé et la vraisemblance est à nouveau maximisée.

Le critère d'information utilisé par la suite sera le critère d'Akaike [Aka74] dont la formule est $AIC() = 2KIJ - 2ln(\mathcal{L})$ où $\mathcal{L}$ est la vraisemblance de la régression, K le nombre de modèles, I la taille de l'espace d'origine et J la taille de l'espace d'arrivée.

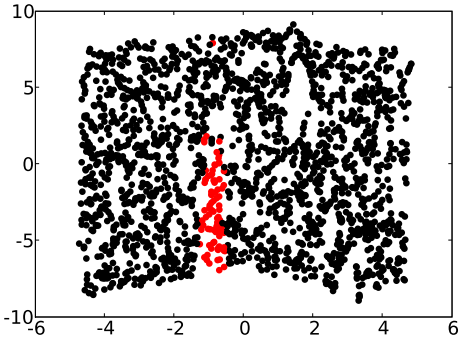
Enfin, cet algorithme ne trouve pas forcément de meilleur modèle que l'algorithme 3 lorsque peu d'échantillons sont disponibles.



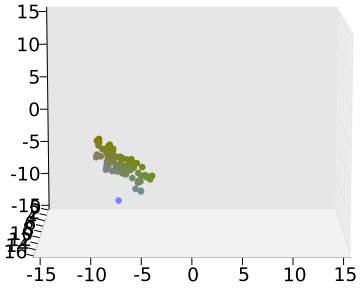
(a) Appartenance aux modèles (un seul modèle en rouge) à la première itération



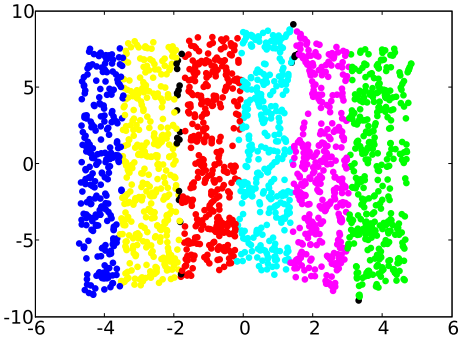
(b) Points étiquetés reconstruits au début de la première itération



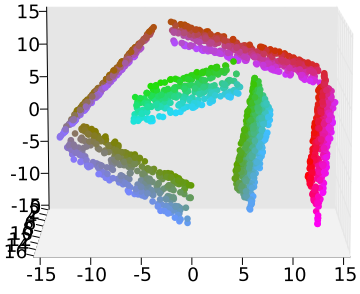
(c) Stabilisation de l'appartenance lors de la première itération



(d) Points étiquetés reconstruits à la fin de la première itération

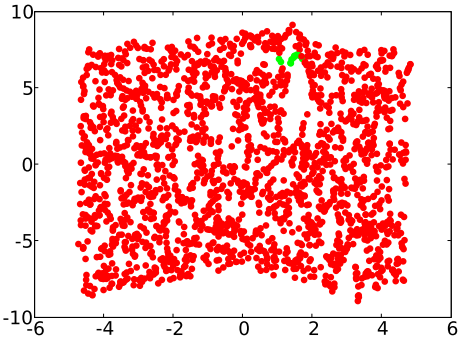


(e) Appartenance aux modèles lors de la dernière itération

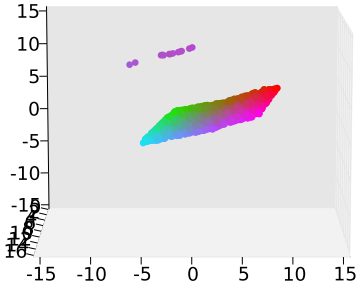


(f) Points reconstruits lors de la dernière itération

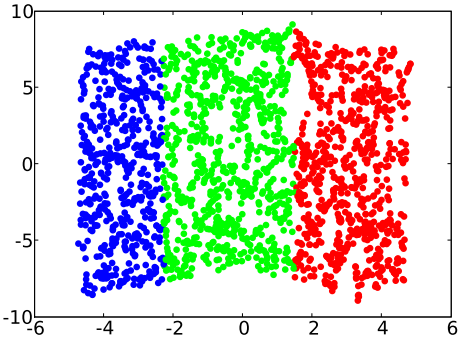
FIG. 3.1: Evolution de l'appartenance aux modèles et de la régression pour PLM1



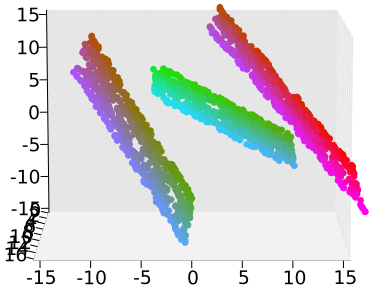
(a) Appartenance aux modèles (un seul modèle en rouge) à la première itération



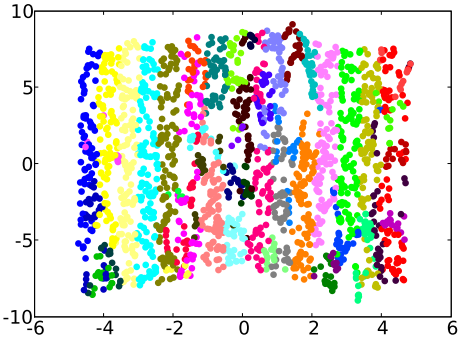
(b) Points étiquetés reconstruits au début de la première itération



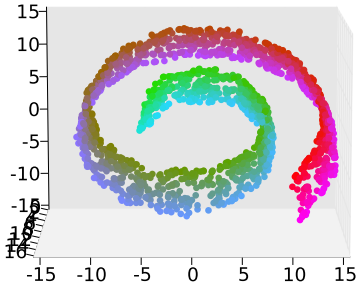
(c) Stabilisation de l'appartenance lors de la première itération



(d) Points étiquetés reconstruits à la fin de la première itération



(e) Appartenance aux modèles lors de la dernière itération



(f) Points reconstruits lors de la dernière itération

FIG. 3.2: Evolution de l'appartenance aux modèles et de la régression pour PLM2

Entrées : coordonnées originales et réduites ($\boldsymbol{y}_i, i \in 1, \dots I$ et $\widehat{\boldsymbol{x}}_i, i \in 1, \dots I$)
Sorties : étiquettes (l_i), matrices des modèles ($\boldsymbol{W}_k$) et variance du bruit (variance des $\hat{\epsilon}_i$)
début
  Créer un modèle linéaire pour décrire tous les points;
  répéter
    Déterminer un ensemble de points dont la vraisemblance d'appartenir aux modèles existants est faible;
    Choisir l'un de ces points;
    Calculer la matrice $\boldsymbol{W}_k$ décrivant le voisinage de ce point;
    Assigner chaque point au modèle le plus vraisemblable;
    répéter
      Supprimer tout modèle ayant trop peu de points;
      Mettre à jour les modèles;
      Assigner chaque point au modèle le plus vraisemblable;
    jusqu'à *convergence* ;
    Vérifier la connexité des modèles;
  jusqu'à *un critère d'information croît* ;
fin

Algorithme 4 : Mise en correspondance linéaire par morceaux 2 (PLM2)

La figure 3.2 montre l'évolution de l'algorithme. Au départ, un modèle (symbolisé par la couleur rouge) englobe la totalité des points. Un nouveau modèle (représenté par la couleur verte fig. 3.2a) est alors ajouté (fig. 3.2b) et englobe une fraction des points. Lors de la maximisation de la vraisemblance du modèle, la connexité du premier modèle (représenté par la couleur rouge) n'est plus assurée et un troisième modèle est alors créé (fig. 3.2c et fig. 3.2d). Après maximisation de la vraisemblance du modèle, d'autres modèles peuvent être ajoutés (représentés par la multitude de couleurs sur la figure 3.2e). Lorsque le critère d'information est satisfait, le modèle est finalisé (fig. 3.2f), et dans le cas présenté, un grand nombre de modèles a pu être ajouté aux endroits les moins vraisemblables.

3.1.3 Approximation d'un nouveau point

A partir des coordonnées dans l'espace réduit, il est nécessaire de sélectionner le modèle adéquat à utiliser pour reconstruire le point dans l'espace original. Le modèle choisi sera le modèle le plus probable en calculant la probabilité d'appartenance à partir d'un mélange de gaussiennes dans l'espace réduit. La largeur des gaussiennes sera spécifique à chaque modèle et sera calculée à l'aide du facteur de Scott [Sco92]. Ce facteur est fonction du nombre de points n et de leur dimension d : $h = n^{-1/d^4}$. On obtient alors le modèle à utiliser en calculant :

$$i = argmax_{k=1,\dots,K} \frac{1}{card\{i, \mathbb{I}(i) = k\}} \sum_{i,\mathbb{I}(\boldsymbol{x}_i)=k} \mathcal{N}_i(\boldsymbol{x}_k | \boldsymbol{x}_i, \boldsymbol{\Sigma}_k)$$

avec $\Sigma_k = h^2\ cov(\boldsymbol{x}_{i,\mathbb{I}(\boldsymbol{x}_i)=k})$

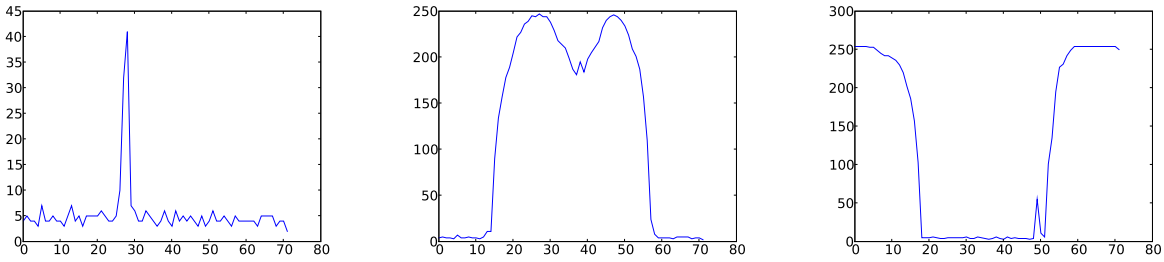


FIG. 3.3: Trois coordonnées dans l'espace d'origine tracées en fonction de l'angle pour le premier jeu de données de la base COIL-20, les canards.

3.2 Régression par blocs

Malheureusement, une régression globale ne peut convenir à toutes les variétés. En effet, les variations des coordonnées dans l'espace d'origine doivent être simultanées, ce qui n'est pas le cas de manière générale.

3.2.1 Position du problème

Les coordonnées d'une variété complexe correspondant à un jeu d'images ne varient pas forcément ensemble (fig. 3.3). La solution consiste à regrouper les coordonnées ayant un comportement semblable (ayant une forte similarité) puis à effectuer une modélisation sur chaque groupe, ou bloc de coordonnées.

D'après la figure 3.3, les coordonnées de cette variété ne peuvent pas être expliquées de manière conjointe. Il est donc nécessaire de trouver quelles sont les coordonnées variant ensemble afin de les regrouper pour créer une régression pour chaque groupe ou bloc. Par la suite, l'approximation d'un nouveau point s'effectue comme pour une régression globale, chaque régression contribuant à approcher une partie des coordonnées.

3.2.2 Partitionnement par corrélation

Le problème à résoudre est celui de la classification spectrale présentée en introduction (1.5.1). A partir d'une mesure de similarité, les coordonnées proches seront regroupées.

La mesure de similarité retenue ici est la corrélation linéaire. Il ne s'agit pas de la mesure optimale pour mesurer si deux coordonnées peuvent être décrites par une fonction affine par morceaux. En effet, si deux variables sont corrélées linéairement, elles peuvent être décrites par la même fonction linéaire par morceaux. En revanche, la réciproque est fausse. La solution utilisée ici (algo. 5) est une solution basée sur l'algorithme général [BBC04], permettant de ne pas fixer le nombre d'agrégats (*clusters*) *a priori*. Malheureusement, cet algorithme est coûteux en

temps et nécessite de calculer la corrélation d'un grand nombre de paires de coordonnées (il est possible de ne pas calculer toutes les corrélations à l'aide d'un algorithme d'évaluation dit paresseux ou *lazy* en anglais). De plus, la corrélation linéaire sous-estimant la mesure recherchée (la possibilité de reconstruire avec la même fonction par morceaux), un nombre plus important que nécessaire d'agrégats est créé.

Entrées : Corrélations entre les coordonnées originales $(corr(\boldsymbol{y}_i, \boldsymbol{y}_j), (i,j) \in [1, \dots I]^2)$
Sorties : Étiquettes (c_i)
début
  tant que *deux points non étiquetés* $\boldsymbol{y}_i$ *et* $\boldsymbol{y}_j$ *existent* **faire**
    Créer un nouvel agrégat à partir de $\boldsymbol{y}_i$;
    Assigner à cet agrégat tous les points non étiquetés avec lesquels $\boldsymbol{y}_i$ a une forte corrélation;
  fin
  tant que *un point non étiqueté* $\boldsymbol{y}_i$ *existe* **faire**
    Créer un nouvel agrégat à partir de $\boldsymbol{y}_i$;
    Assigner à cet agrégat tous les points non étiquetés avec lesquels $\boldsymbol{y}_i$ a une forte corrélation;
  fin
fin

Algorithme 5 : Partionnement dans le graphe

3.3 Résultats

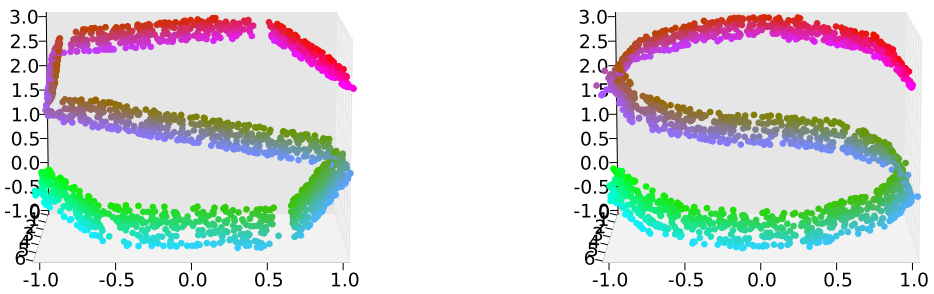
Afin de vérifier le fonctionnement de la régression, chaque point d'apprentissage est reconstruit à partir des coordonnées réduites par compression et la différence entre le point original et cette reconstruction est calculée. La variance de cette erreur de reconstruction permet par la suite de valider l'apprentissage.

La figure 3.1f expose la régression du SwissRoll avec l'algorithme PLM1 (algorithme 3) tandis que la figure 3.2f propose l'algorithme PLM2 (algorithme 4). Comme cela a été indiqué dans l'exposé de ces deux méthodes, l'algorithme PLM1 est moins précis que PLM2, en revanche le temps de calcul pour 2000 points du SwissRoll se compte en minutes contre des heures pour le second algorithme.

Lors des apprentissages exposés ici, l'erreur moyenne pour chaque coordonnée du SwissRoll est de 20% pour l'algorithme 3 et de 2.1% pour l'algorithme 4.

Les figures 3.4a et 3.4b permettent d'obtenir les mêmes conclusions avec le SCurve que pour le SwissRoll. L'erreur moyenne pour chaque coordonnée du SCurve est ici de 3.1% pour l'algorithme 3 et de 1.4% pour l'algorithme 4.

L'algorithme de réduction de dimensionnalité utilisé a un impact faible sur le résultat de l'algorithme 4. En revanche, l'utilisation des *Laplacian Eigenmaps* n'est pas conseillé car la variété ne peut pas être modélisée, vraisemblablement à cause de la propension de l'algorithme



(a) PLM1 (b) PLM2

FIG. 3.4: Régression du SCurve

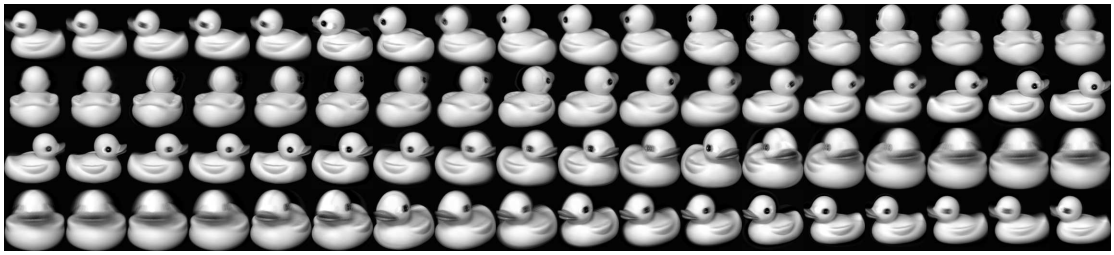
à regrouper les points de forte similarité. L'utilisation d'une méthode de réduction de dimension ne conservant pas les distances géodésiques augmente le nombre de modèles utilisés par l'algorithme 3 principalement. En effet, si les distances géodésiques ne sont pas respectées, la régression locale par un modèle linéaire n'est pas optimale, peu de points sont alors affectés au nouveau modèle.

La figure 3.5 représente les échantillons modélisés de deux jeux de données de la base COIL-20. L'erreur moyenne sur une base d'apprentissage varie entre 2% et 10% (sur toute la plage de valeurs d'une image) selon la complexité de celle-ci. Les échantillons de la figure 3.5 ont respectivement 3.4% et 4.6% d'erreur d'apprentissage. A titre de comparaison, une analyse en composantes principales avec 2 vecteurs propres engendre une erreur de 14.2% et de 10.5% respectivement et de 4.8% et de 4.9% pour une analyse avec 15 vecteurs propres.

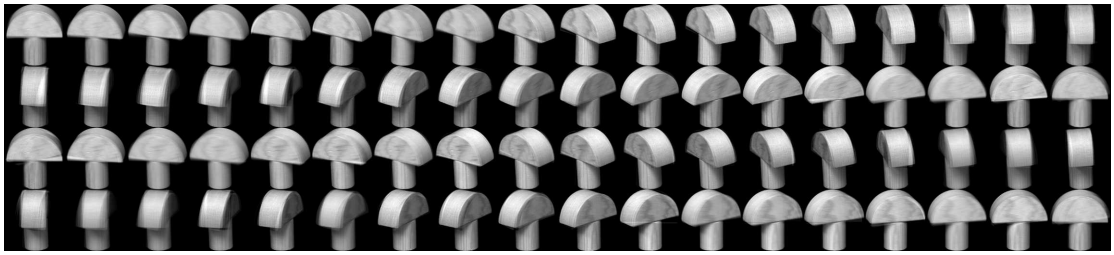
Lors d'une projection régularisée (par maximum *a posteriori* par exemple, présentée dans le prochain chapitre), le terme additionnel peut être la probabilisation de l'espace sous-jacent. La figure 3.6 expose la densité de probabilité sur tous les points (cette densité est utilisée pour une projection sur une régression par bloc par exemple). Certaines zones de la variété semblent sur-représentées (les zones rouge bordeaux), en revanche la zone d'intérêt (là où les points d'apprentissage ont été réduits) est toujours bien représentée, ce qui permettra une projection efficace. Les points d'apprentissage sont affichés par des points noirs sur cette figure.

En ce qui concerne la régression par bloc (fig. 3.7), l'erreur de régression pour chaque coordonnée est de 0.6%, soit une diminution de plus de 50% de l'erreur d'apprentissage de l'algorithme PLM2.

Tout comme pour la régression globale, la régression par blocs engendre une erreur d'apprentissage de 4.5% sur la base des canards (fig. 3.8a). Si la qualité au sens quadratique est comparable, il n'en est pas de même pour la qualité visuelle. La régression par blocs nécessite donc un découpage respectant des critères psychovisuels. Malheureusement, la réalisation de tels critères reste un problème ouvert [HPN02] Enfin, le facteur limitant pour garantir une meilleure

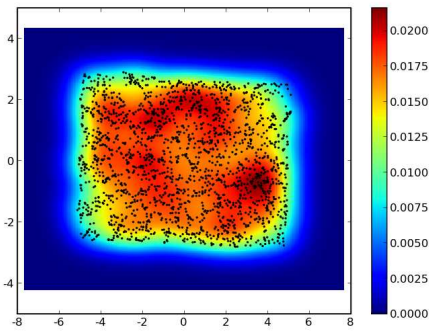


(a) Canards

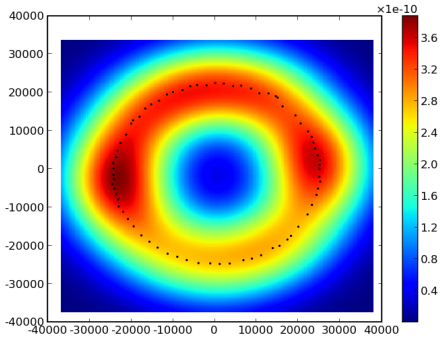


(b) Pièce de bois

FIG. 3.5: Modélisations (régressions) de données de la base COIL-20

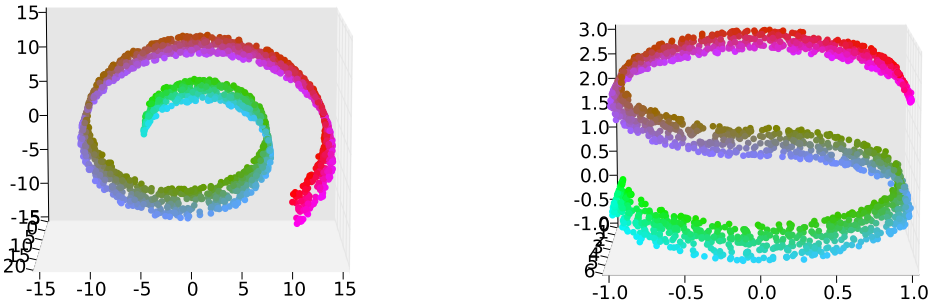


(a) *a priori* du SCurve



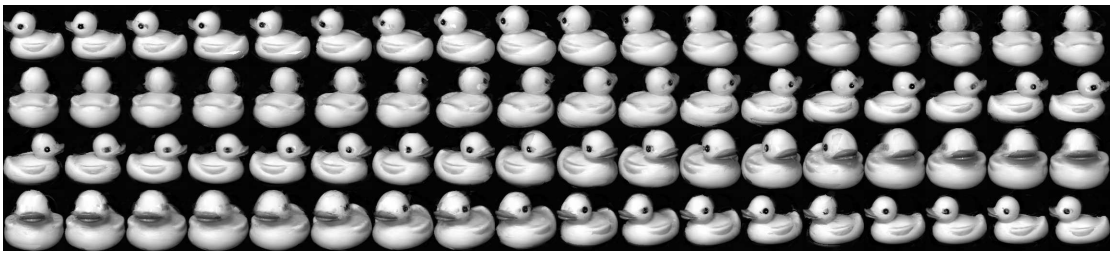
(b) *a priori* pour une série de la base COIL-20

FIG. 3.6: Carte d'*a priori* (densité de probabilité).

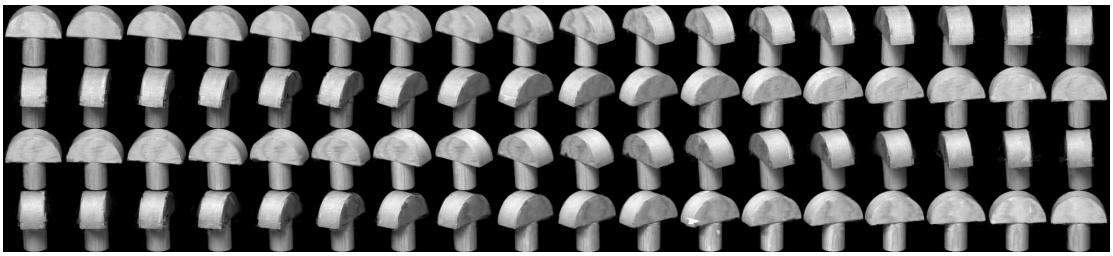


(a) SwissRoll (b) SCurve

FIG. 3.7: Régression par bloc



(a) Canards



(b) Pièce de bois

FIG. 3.8: Modélisations par blocs de données de la base COIL-20

qualité semble être le nombre d'échantillons, même sur un nombre plus restreint de coordonnées par rapport à l'image globale.

3.4 Conclusion

Une méthode d'estimation d'une fonction de mise en correspondance entre deux espaces a été présentée. En coordination avec la méthode de réduction de dimension $\mathcal{S}_{\mathrm{P}}$ présentée précédemment, la qualité de cet apprentissage en 2 dimensions est meilleure qu'un apprentissage à l'aide de l'ACP avec 15 vecteurs.

L'estimation de plusieurs fonctions de mise en correspondance pour plusieurs fractions des échantillons de la base d'apprentissage a été introduite. Cette mise en correspondance permet de diminuer l'erreur d'apprentissage par rapport à une mise en correspondance totale dans certains cas, mais le facteur limitant la qualité reste le nombre d'échantillons de la base d'apprentissage.

Chapitre 4

Projection de nouvelles données sur la variété

Sommaire

4.1 Projection avec une régression globale 47
4.1.1 Projection par maximum de vraisemblance 48
4.1.2 Projection par maximum *a posteriori* 48
4.2 Projection avec une régression par blocs 49
4.2.1 Utilisation d'une grille régulière 49
4.2.2 Utilisation d'une grille hiérarchique 49
4.2.3 Utilisation d'une grille adaptative 51
4.3 Résultats 51
4.3.1 Qualité de projection 51
4.3.2 Projection avec occlusion 53
4.4 Conclusion 60

La possibilité de projeter un nouvel élément sur la variété est primordiale dans le test d'un individu contre une population : elle permet de classer l'individu dans une sous-classe de la population et surtout d'indiquer les différences entre l'individu et la classe.

4.1 Projection avec une régression globale

Dans le cas présent, la variété a été modélisée et les échantillons d'origine servant à l'apprentissage ne sont plus disponibles. Dans le cadre d'une régression globale, chaque modèle linéaire représente une mise en correspondance entre une partie de l'espace réduit et une partie (inconnue) de l'espace original. En projetant le nouvel échantillon sur chaque modèle linéaire, il est possible de retrouver le modèle adéquat.

4.1.1 Projection par maximum de vraisemblance

Pour chaque point projeté (un par modèle linéaire), il est possible de calculer sa vraisemblance connaissant la loi de l'erreur associée à la régression globale. Dans le cas où celle-ci est une gaussienne isotrope, une projection orthogonale permet de trouver l'optimum directement, sinon une optimisation doit être effectuée.

$$
\begin{aligned}
p(\boldsymbol{y}, \boldsymbol{x}_k) &= p(\boldsymbol{y} - \boldsymbol{W}_k \boldsymbol{x}_k) && (4.1)\\
&= \mathcal{N}(\boldsymbol{y} - \boldsymbol{W}_k \boldsymbol{x}_k | \mu, \boldsymbol{\Sigma}) && (4.2)\\
& && (4.3)
\end{aligned}
$$

La probabilité $p(\boldsymbol{y}, \boldsymbol{x}_k)$ du projeté $\boldsymbol{x}_k$ d'un point $\boldsymbol{y}$ dépend ainsi de la probabilité de l'erreur de reconstruction $\boldsymbol{y} - \boldsymbol{W}_k \boldsymbol{x}_k$ et de la probabilité du point $\boldsymbol{x}_k$ dans l'espace réduit.

Lorsque la vraisemblance de chaque projeté a été calculée à l'aide de la formule 4.3 (dans le cas d'une loi d'erreur gaussienne), il suffit de choisir la vraisemblance la plus grande. En revanche, ce modèle a des limites car le meilleur projeté peut avoir des coordonnées à l'extérieur de la zone associée à un modèle linéaire. Il est donc pertinent de considérer un terme supplémentaire d'*a priori* (qui peut aussi être interprété comme une régularisation).

4.1.2 Projection par maximum *a posteriori*

A chaque modèle linéaire est associé un ensemble de points dans l'espace réduit provenant de l'apprentissage. Tout comme pour la fonction de régression, la projection utilisera plusieurs mélanges de gaussiennes identiques à ceux utilisés pour la régression. Lors de la projection sur un modèle linéaire, la vraisemblance associée au projeté dans un sous-espace réduit sera ajoutée à la vraisemblance du terme d'erreur.

$$
\begin{aligned}
p(\boldsymbol{y}, \boldsymbol{x}_k) &= p(\boldsymbol{y} - \boldsymbol{W}_k \boldsymbol{x}_k) p(\boldsymbol{x}_k) \\
&= \mathcal{N}(\boldsymbol{y} - \boldsymbol{W}_k \boldsymbol{x}_k | \mu, \boldsymbol{\Sigma}) \frac{1}{card\{i, \mathbb{I}(\boldsymbol{x}_i) = k\}} \sum_{i, \mathbb{I}(\boldsymbol{x}_i)=k} \mathcal{N}_i(\boldsymbol{x}_k | \boldsymbol{x}_i, \boldsymbol{\Sigma}_k) \qquad (4.4)
\end{aligned}
$$

Ici, la densité de probabilité $p(\boldsymbol{x}_k)$ a une expression spécifique pour le modèle k. Chaque point dans l'espace réduit $\boldsymbol{x}_i$ appartenant au modèle k ($\mathbb{I}(i) = k$) contribue à la probabilité $p(\boldsymbol{x}_k)$.

L'équation 4.4 expose cette solution dans le cas où l'erreur est gaussienne. Dans ce cas, il est possible d'obtenir le maximum de manière analytique, une optimisation avec méthode du gradient conjugué [NW06] sera utilisée dans le cas général.

4.2 Projection avec une régression par blocs

Lors d'une régression par blocs, il n'est plus possible de projeter le nouveau point sur chaque plan ou plutôt sur chaque combinaison de plans (un plan par bloc, ce qui engendre une explosion combinatoire), ceci par maximum de vraisemblance ou par maximum *a posteriori* (pour ce dernier, il n'est même plus possible de définir correctement le terme *a priori*). Il est donc nécessaire de parcourir l'espace réduit pour trouver le projeté le plus pertinent, soit au sens du maximum de vraisemblance ou du maximum *a posteriori*.

Il est à noter que l'espace à parcourir est de faible dimension ($\mathbb{R}^2$, $\mathbb{R}^3$ en général), ce qui fait qu'un parcours à l'aide d'algorithmes complexes tels que le mean-shift [CM02, VHC07] n'est pas nécessaire, même s'il peut être utile lorsque la dimension de l'espace augmente.

$$\begin{aligned} p(\boldsymbol{y}, \boldsymbol{x}_k) &= p(\boldsymbol{y} - \boldsymbol{W}_{\mathbb{I}(\boldsymbol{x}_k)}\boldsymbol{x}_k)p(\boldsymbol{x}_k) \\ &= \mathcal{N}(\boldsymbol{y} - \boldsymbol{W}_{\mathbb{I}(\boldsymbol{x}_k)}\boldsymbol{x}_k|\mu, \boldsymbol{\Sigma}_1)\frac{1}{n}\sum_i \mathcal{N}(\boldsymbol{x}_k|\boldsymbol{x}_i, \boldsymbol{\Sigma}_2) \end{aligned} \tag{4.5}$$

L'équation 4.5 propose la formule de calcul du maximum *a posteriori* dans la régression par blocs (le maximum de vraisemblance est obtenu en ne considérant pas le second terme dans le calcul). Pour chaque $\boldsymbol{x}_k$ testé, un modèle $\boldsymbol{W}_{\mathbb{I}(\boldsymbol{x}_k)}$ est constitué par blocs à partir des $\boldsymbol{W}_{i,\mathbb{I}_i(\boldsymbol{x}_k)}$ (i étant le bloc utilisé) indiquant la composition des modèles linéaires à utiliser pour estimer le point dans l'espace d'origine. La probabilisation de l'espace réduit s'effectue avec un seul mélange de gaussiennes dont la covariance est donnée par $\boldsymbol{\Sigma}_2$.

4.2.1 Utilisation d'une grille régulière

A partir des points de l'espace réduit, il est possible de créer une grille régulière (fig. 4.1) et de tester chaque élément sur cette grille. Cette grille doit englober de manière très large l'ensemble des points issus de l'apprentissage afin de pouvoir proposer un panel de points très large.

Malheureusement, cette solution nécessite une grille très fine si la précision souhaitée est élevée. Cette grille très fine engendre un coût dont la complexité est celle du nombre de dimensions de l'espace réduit ($O(N^2)$ pour $\mathbb{R}^2$, $O(N^3)$ pour $\mathbb{R}^3$, ...). Une solution plus adaptée doit donc être trouvée.

4.2.2 Utilisation d'une grille hiérarchique

L'utilisation d'une grille hiérarchique est une bonne solution initiale de remplacement. A partir d'un nombre fixé de points intéressants (dont le coût est faible), une sous-grille est créée et un nombre fixe de points d'intérêt est à nouveau récupéré (fig. 4.2). Ceci est effectué de manière itérative jusqu'à ce que la précision voulue (en terme de placement dans l'espace réduit) soit atteinte.

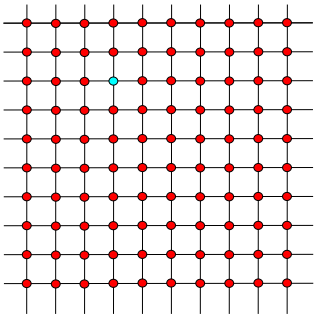


FIG. 4.1: Grille régulière où le point bleu est l'optimum global

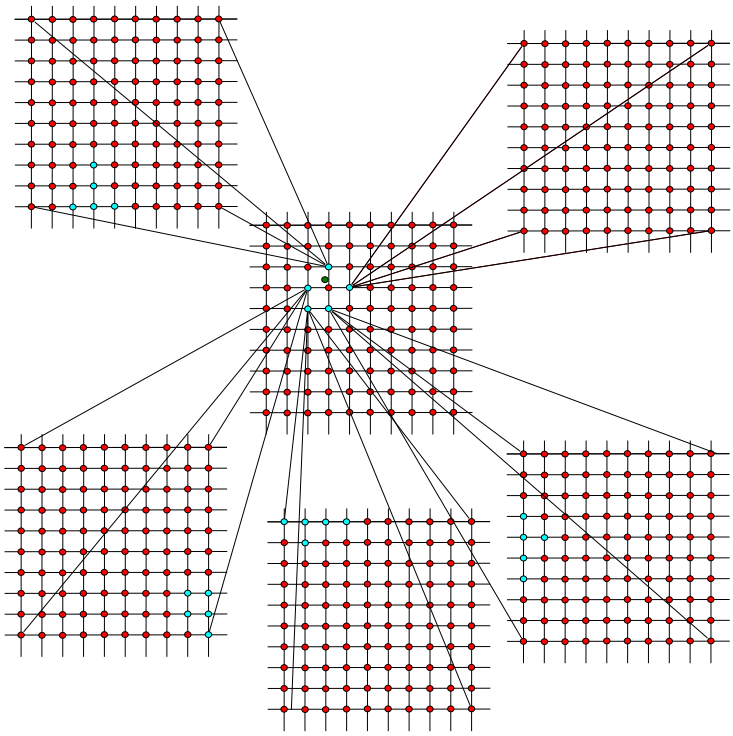


FIG. 4.2: Grille hiérarchique, chaque point bleu étant un point autour duquel une recherche plus fine sera effectuée. Le point vert est le point optimal, mais il ne se situe pas près d'un point bleu sélectionné.

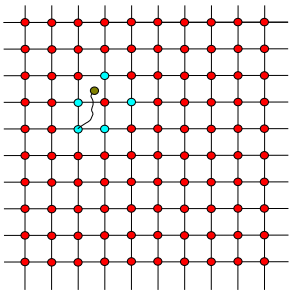


FIG. 4.3: Grille adaptative, où chaque point bleu, potentiellement optimal, est déplacé (seul l'un d'entre eux est montré ici).

Un problème inhérent à cette solution est que le point à un niveau l qui conduirait à l'optimum global peut ne pas être sélectionné car il existe un nombre important de points très vraisemblables. Il est donc nécessaire de sélectionner un nombre important de points à chaque niveau, ce qui ajoute à la complexité de cette recherche.

4.2.3 Utilisation d'une grille adaptative

En coordination avec une grille hiérarchique, il est possible d'optimiser localement le point sélectionné. Après la sélection au niveau l des meilleurs points, ceux-ci sont déplacés par optimisation (fig. 4.3). La fonction n'étant pas nécessairement continue à cause des $\mathbb{I}_i()$, une optimisation de Nelder-Mead [Rao96], aussi appelée simplexe, permettra d'obtenir une meilleure solution.

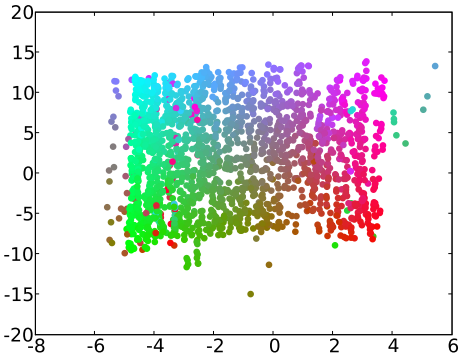
4.3 Résultats

La validation de la projection nécessite plusieurs phases :
- vérifier qu'un nouveau point de la variété se projette bien sur lui-même,
- valider la projection d'un point n'appartenant pas à la variété sur la variété, de sorte que le point projeté soit proche (en un certain sens) du point original.

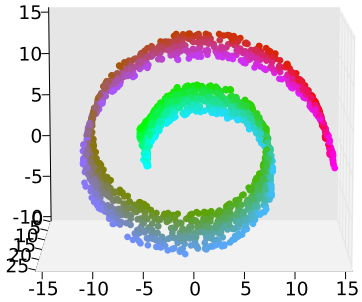
Le premier type de validation sera réalisée sur le SwissRoll et le SCurve tandis que le second sera réalisé sur l'ensemble de la base COIL-20.

4.3.1 Qualité de projection

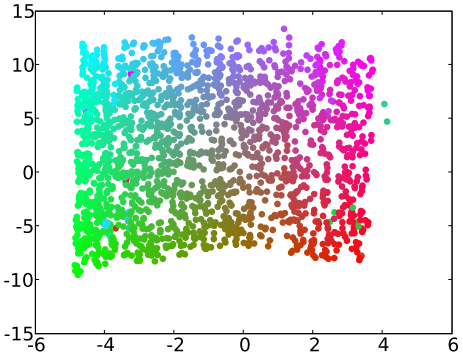
Deux mesures peuvent être effectuées sur la qualité de la projection :
- Est-ce que le point projeté sur la variété est éloigné du point original ?
- Est-ce que le point projeté dans l'espace réduit se place correctement par rapport à son emplacement théorique ?



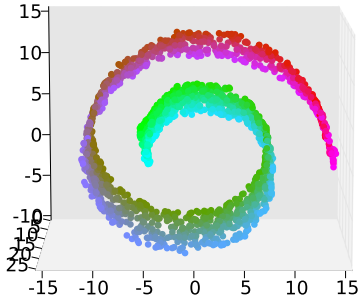
(a) Coordonnées réduites par projection par maximum de vraisemblance avec PLM2



(b) Projection par maximum de vraisemblance avec PLM2



(c) Coordonnées réduites par projection par maximum *a posteriori* avec PLM2



(d) Projection par maximum *a posteriori* avec PLM2

FIG. 4.4: Projection de nouveaux échantillons du SwissRoll

La deuxième question peut être résolue par l'utilisation de la même mesure que dans le tableau 2.1 avec le SCurve. En effet, il est possible de calculer les coordonnées réduites associées à de nouvelles coordonnées grâce à la transformation affine entre l'espace des coordonnées originales et l'espace des coordonnées réduites (ce qui n'est pas le cas pour le SwissRoll).

La figure 4.4 expose la projection de 2000 nouveaux échantillons du SwissRoll. Les projections par maximum de vraisemblance ou *a posteriori* sont visuellement très proches, l'erreur relative sur chaque coordonnée est de 1.1% et de 1.2% respectivement. La différence est logique, l'erreur relative étant une mesure quadratique minimisée par le maximum de vraisemblance, contrairement au maximum *a posteriori* qui propose un terme supplémentaire de régularisation. En revanche, l'espace réduit correspondant est très différent. De manière générale, la structure est bien respectée, sauf pour certains échantillons mal réduits. La projection par maximum *a posteriori* prend ici tout son sens puisque le nombre de ces échantillons mal réduits est bien

plus faible que pour la projection par maximum de vraisemblance.

Les conclusions avec le SCurve (fig. 4.5) sont identiques, le maximum *a posteriori* permet d'obtenir un espace réduit mieux structuré. L'erreur relative est de 0.7% pour le maximum de vraisemblance et de 0.9% pour le maximum *a posteriori*. En ce qui concerne l'erreur relative de réduction, elle est de 13.6% et de 2.1% respectivement. On observe bien l'amélioration considérable de la qualité des coordonnées réduites entre les deux types de projection. Dans les expériences menées ici, le maximum *a posteriori* a proposé un espace réduit adéquat pour tous les points, mais la variété est aussi bien moins courbée que le SwissRoll.

Une régression par blocs a aussi été utilisée pour projeter de nouveaux points sur la variété (fig. 4.6) à l'aide d'une grille adaptative et en estimant à chaque fois le maximum *a posteriori*. Les deux espaces montrent des points mal projetés et réduits, même si l'espace réduit conserve la structure de la variété (symbolisée par le dégradé de couleur de la figure 4.6a). De plus, le temps de calcul bien plus important qu'une projection sur un modèle global ne permet pas d'utiliser efficacement ce modèle.

4.3.2 Projection avec occlusion

Une occlusion dans une image est une partie masquée de celle-ci (fig. 4.8a). Plusieurs tailles d'occlusion ont été considérées sur l'ensemble de la base COIL-20 et ces images avec occlusions ont été projetées en utilisant plusieurs algorithmes :

- l'ACP avec deux vecteurs propres ;
- l'ACP avec 15 vecteurs propres ;
- l'ACP avec 15 vecteurs propres et une reconstruction robuste ;
- la compression par $\mathcal{S}_{\mathrm{P}}$, une régression par PLM1 ou PLM2 puis une projection par maximum *a posteriori*.

La reconstruction robuste a été effectuée à l'aide de la fonction de coût de Geman-McClure comme anti-log-vraisemblance de l'erreur de projection (en lieu et place du coût quadratique d'une vriable alétoire gaussienne). Pour chaque pixel de l'image à reconstruire, le coût est alors :

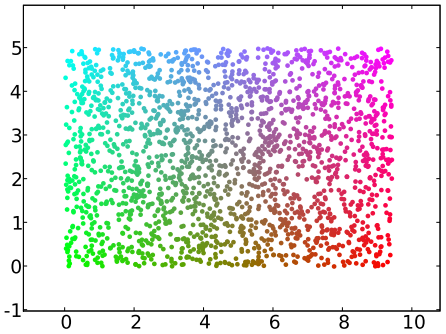
$$f(x) = \frac{x^2}{\sigma^2 + x^2},$$

avec σ l'écart-type du pixel calculé sur la population d'apprentissage.

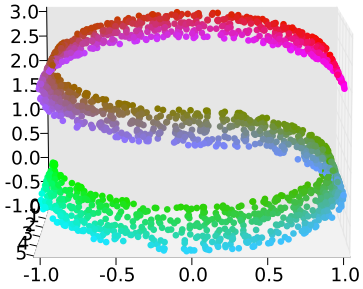
La projection par maximum *a posteriori* utilise quant à elle une hypothèse de variable aléatoire de Laplace en lieu et place de la variable aléatoire gaussienne utilisée lors de l'apprentissage de l'erreur du modèle. Cette variable aléatoire permet de mieux modéliser le bruit réel engendré par l'occlusion.

Les projections dites réussies sont les projections où l'image de la base d'apprentissage la plus proche du projeté est l'image testée sans occlusion. Les projections dites proches sont quant à elles les projections où l'image de la base d'apprentissage la plus proche du projeté est une image proche de l'image testée sans occlusion.

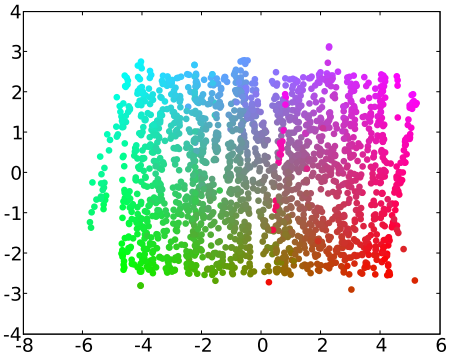
Malgré le nombre de projections réussies pour l'ACP, les résultats visuels sont tout de même en-dessous de la chaîne de traitement présentée. En effet, la figure 4.8 propose la reconstruction



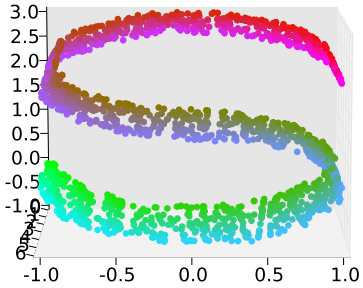
(a) Nouvelles coordonnées



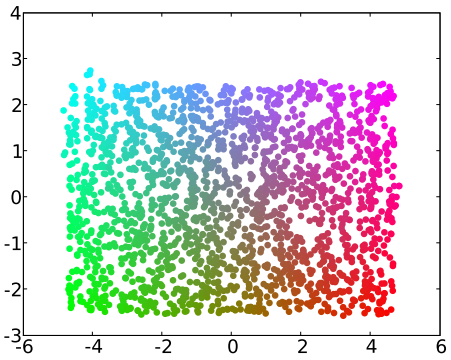
(b) Points correspondants



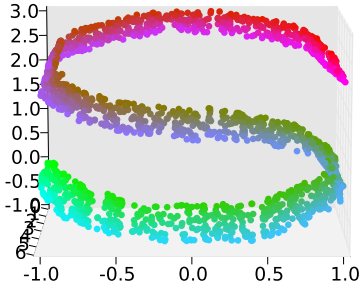
(c) Coordonnées réduites par projection par maximum de vraisemblance avec PLM2



(d) Projection par maximum de vraisemblance avec PLM2

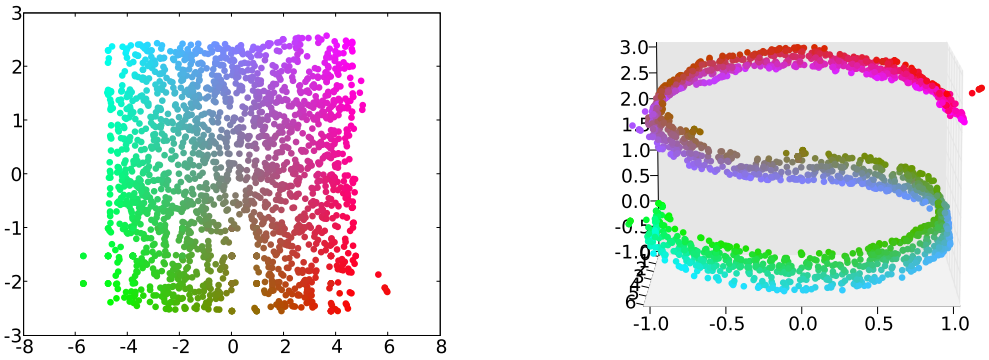


(e) Coordonnées réduites par projection par maximum *a posteriori* avec PLM2



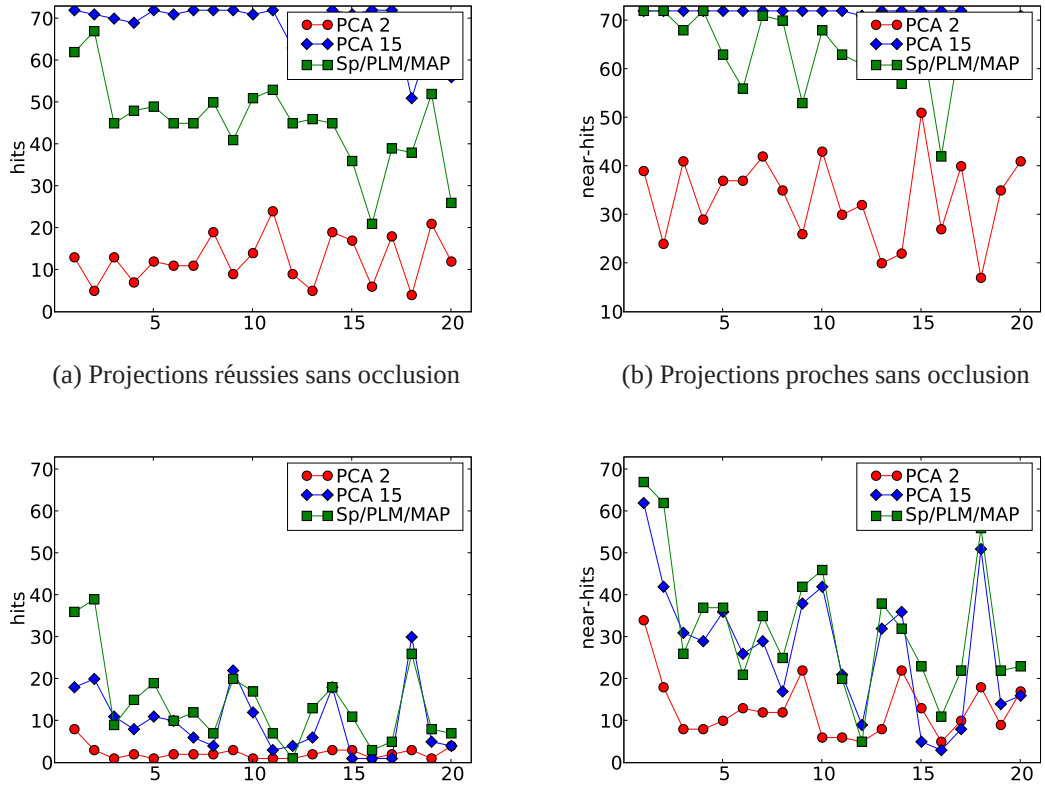
(f) Projection par maximum *a posteriori* avec PLM2

FIG. 4.5: Projection de nouveaux échantillons du SCurve.



(a) Nouvelles coordonnées (b) Points correspondants

FIG. 4.6: Projection de nouveaux échantillons du SCurve.



(a) Projections réussies sans occlusion (b) Projections proches sans occlusion

(c) Projections réussies avec occlusion (40%) (d) Projections proches avec occlusion (40%)

FIG. 4.7: Projection d'images avec ou sans occlusion pour chaque jeu de données de la base COIL-20.

de chaque élément avec 40% d'occlusion avec l'ACP puis avec l'apprentissage et la projection présentés. Avec l'ACP, les objets ne sont pas reconnaissables, ce qui n'est pas le cas des autres éléments reconstruits, même lorsque l'objet n'est pas projeté près de l'élément sans occlusion, celui-ci reste reconnaissable.

L'estimation de pose est aussi un critère intéressant. Elle consiste à estimer la pose de l'objet par rapport à sa pose théorique, puis un histogramme des différences peut être tracé (fig. 4.9). Dans les deux jeux de données considérés, les résultats sont semblables :

- l'ACP avec seulement deux vecteurs n'est pas capable d'estimer la pose sans occlusion et les performances se dégradent fortement avec une occlusion,
- l'ACP avec 15 vecteurs propres est stable, malgré les occlusions, estimant de manière correcte la pose, même en présence de bruit,
- la chaîne de traitement proposée obtient des résultats comparables à l'ACP à 15 dimensions, et ceci avec seulement deux dimensions.

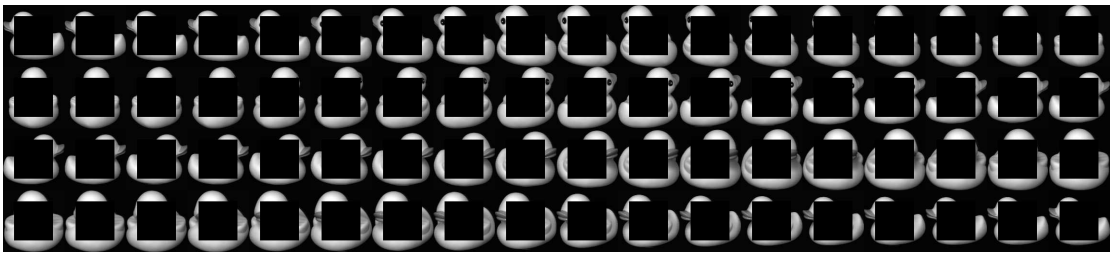
En utilisant une variable aléatoire robuste (variable de Laplace), les algorithmes présentés en deux dimensions sont à la hauteur de l'outil de référence, l'ACP, avec une fonction robuste (Geman-McClure) dans un espace de dimension 15. Cette différence de dimension permet une simplification des outils nécessaires à la projection (pas besoin d'un algorithme Mean-Shift modifié [VHC07] pour parcourir l'espace réduit) et ne nécessite pas de fonction de coût robuste et non convexe, telle qu'une fonction de Geman-McClure. En effet, une fonction robuste convexe telle que celle associée à une variable de Laplace (la fonction valeur absolue) est suffisante.

Un test *leave-one-out* [Bis06] a été effectué sur le premier jeu de la base COIL-20, les canards. 72 tests ont donc été réalisés, en ôtant à chaque fois un élément différent. Un apprentissage a été effectué, puis la totalité des échantillons de test (avec et sans occlusion) a été projetée et classée par rapport aux 72 images d'origine. Par la suite, la moyenne de ces 72 tests a été effectuée. Les tests sur le jeu de données ont donné des résultats identiques aux résultats précédents (exposés sur la figure 4.9).

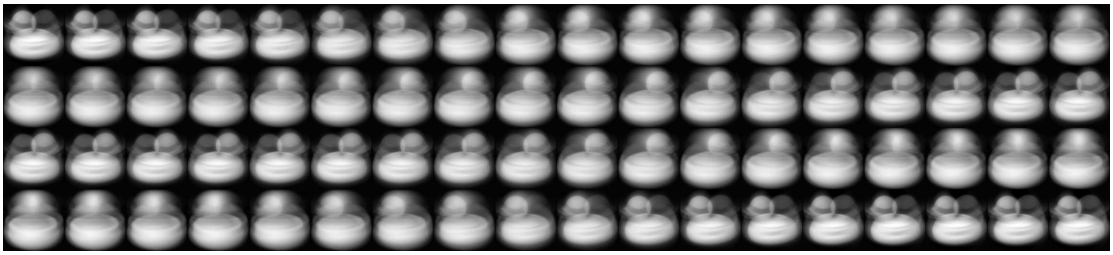
L'erreur de projection (entre le point projeté et le point non projeté) prouve aussi que la chaîne de traitement est meilleure que l'ACP à 2 ou 15 vecteurs propres. En effet, sur le troisième jeu de la base COIL-20, par exemple, l'ACP à 2 dimensions (avec une projection robuste) présente une erreur de 14.8% sans occlusion à 17.7% avec une occlusion de 40%. L'ACP avec 15 vecteurs propres quant à lui présente des taux de 8.3% et 20.1% respectivement. Enfin, la solution présentée obtient 6.9% et 19.0%.

Le tableau 4.10 reprend l'ensemble des données pour l'ensemble de la base COIL-20 (la projection n'a pas été effectuée pour tous les jeux de données avec la régression par blocs car le gain était trop faible par rapport au temps d'exécution). A cause du grand espace à parcourir en dimension 15, les performances de l'ACP se dégradent très vite, et en présence de fortes occlusions, l'ensemble $\mathcal{S}_{\mathrm{P}}$/PLM/MAP obtient les meilleurs résultats. Sans aucune occlusion, les deux outils proposent des résultats proches.

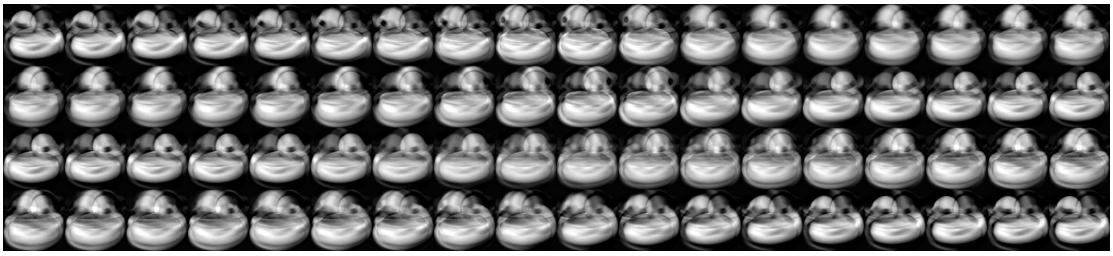
Le modèle par blocs a été testé sur le premier jeu de données de la base COIL-20. Les résultats sont corrects pour une projection sans occlusion (8.3%) par rapport à l'ACP avec 2 vecteurs propres, mais moins bons que pour les deux autres modélisations. En cas de projection avec occlusion, les résultats sont plus mauvais que les autres en raison des difficultés d'obtention



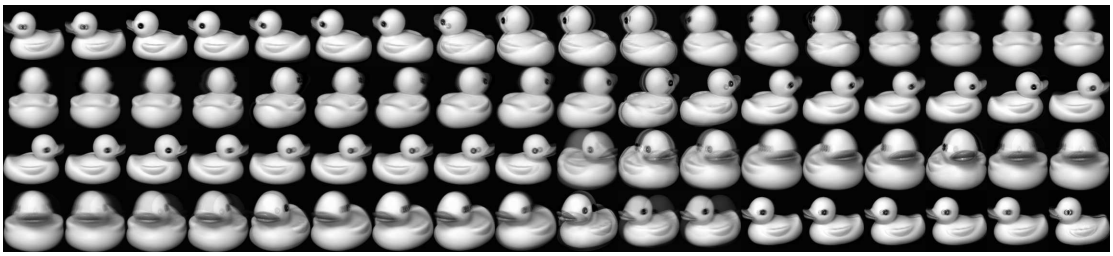
(a) Occlusion sur la base COIL-20



(b) Projection avec ACP et un apprentissage par ACP (2 vecteurs propres)

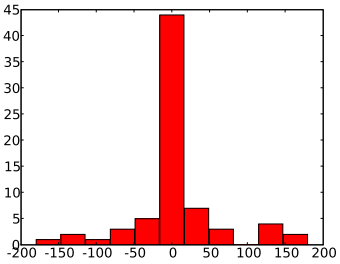


(c) Projection avec ACP et un apprentissage par ACP (15 vecteurs propres)

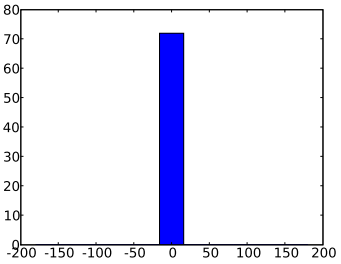


(d) Projection avec maximum *a posteriori* et apprentissage par $\mathcal{S}_{\mathrm{P}}$ et PLM2

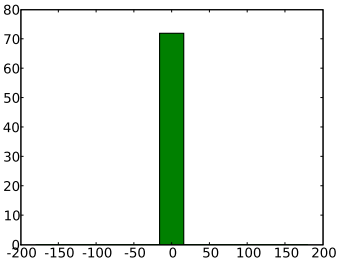
FIG. 4.8: Projection des canards (COIL-20) avec 40% d'occlusion.



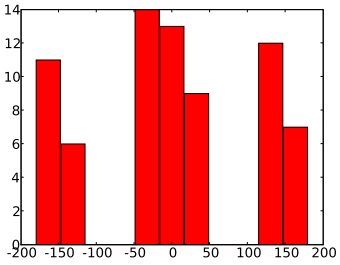
(a) Canards (0%) : ACP 2



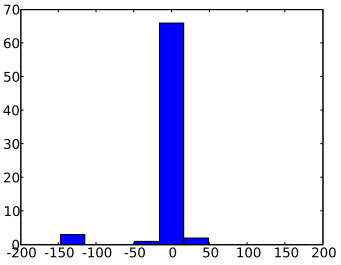
(b) Canards (0%) : ACP 15



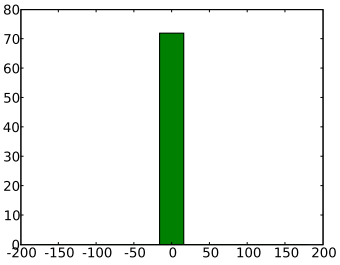
(c) Canards (0%) : $\mathcal{S}_{\mathrm{P}}$/PLM/MAP



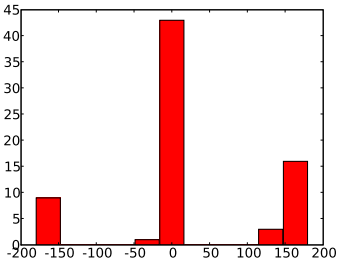
(d) Canards (40%) : ACP 2



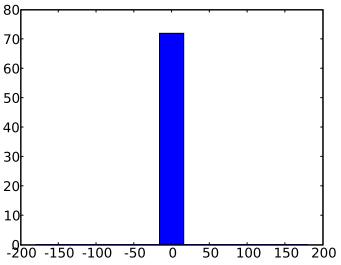
(e) Canards (40%) : ACP 15



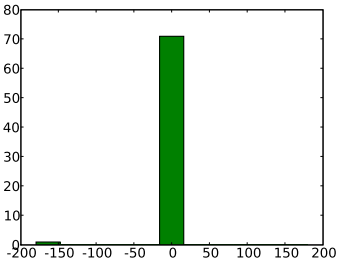
(f) Canards (40%) : $\mathcal{S}_{\mathrm{P}}$/PLM/MAP



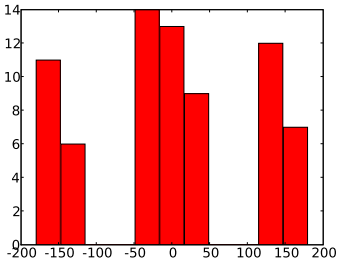
(g) Pièces de bois (0%) : ACP 2



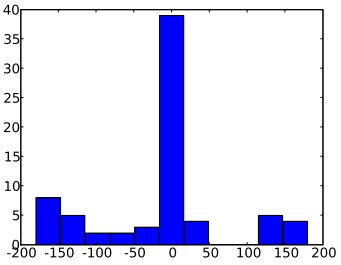
(h) Pièces de bois (0%) : ACP 15



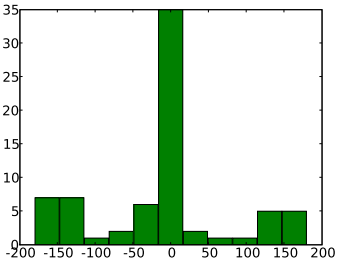
(i) Pièces de bois (0%) : $\mathcal{S}_{\mathrm{P}}$/PLM/MAP



(j) Pièces de bois (40%) : ACP 2



(k) Pièces de bois (40%) : ACP 15



(l) Pièces de bois (40%) : $\mathcal{S}_{\mathrm{P}}$/PLM/MAP

FIG. 4.9: Estimation de la position des images sans occlusion puis avec 40% d'occlusion

	ACP 2		ACP 15		$\mathcal{S}_P$/PLM/MAP	
Jeu de données	0%	40%	0%	40%	0%	40%
Jeu 01	14.2%	17.1%	4.8%	17.7%	3.7%	6.7%
Jeu 02	18.6%	21.0%	7.0%	23.8%	4.7%	10.5%
Jeu 03	14.8%	17.7%	8.3%	20.1%	6.9%	19.0%
Jeu 04	14.8%	19.6%	5.9%	22.2%	7.6%	15.9%
Jeu 05	15.7%	19.2%	9.5%	17.7%	8.6%	14.0%
Jeu 06	15.9%	19.1%	9.1%	17.9%	8.2%	19.9%
Jeu 07	11.5%	14.8%	4.5%	19.8%	6.8%	13.9%
Jeu 08	9.3%	12.5%	3.6%	17.9%	3.8%	11.2%
Jeu 09	17.7%	18.7%	11.2%	19.1%	9.6%	12.0%
Jeu 10	12.9%	16.1%	5.9%	17.5%	5.2%	13.8%
Jeu 11	10.6%	17.2%	4.9%	21.4%	4.0%	16.9%
Jeu 12	4.8%	11.7%	2.6%	15.1%	2.9%	15.4%
Jeu 13	14.9%	22.3%	7.1%	23.6%	8.0%	16.6%
Jeu 14	10.4%	11.4%	5.3%	16.9%	6.0%	10.1%
Jeu 15	3.0%	3.9%	1.4%	17.4%	1.8%	7.8%
Jeu 16	3.4%	7.7%	1.4%	19.4%	2.0%	13.7%
Jeu 17	5.2%	14.7%	2.4%	21.5%	3.1%	21.2%
Jeu 18	8.9%	10.6%	4.6%	10.2%	5.5%	7.3%
Jeu 19	12.3%	17.5%	6.6%	17.9%	5.1%	20.7%
Jeu 20	9.2%	10.6%	5.6%	10.8%	6.2%	10.4%

FIG. 4.10: Erreur de projection d'un objet sans et avec occlusion

d'un bon optimum en utilisant une grille.

4.4 Conclusion

La méthode de projection proposée combinée avec le modèle présenté dans le chapitre précédent a des résultats au moins aussi bons que l'ACP robuste utilisant 15 vecteurs propres. Avec seulement deux dimensions, la chaîne de traitement proposée permet de parcourir de manière simple l'espace réduit, ce qui se traduit par des performances en terme d'erreur de projection et d'estimation de pose supérieures à l'ACP, avec deux vecteurs propres ou avec quinze.

La projection sur un modèle par blocs souffre d'inconvénients qui l'empêchent d'être utilisable en l'état, notamment son temps d'exécution et son incapacité à trouver un optimum acceptable pour des données telles que celles de la base COIL-20. De plus, les résultats présentent des incohérences visuelles, il est donc nécessaire d'utiliser un critère de séparation par blocs psychovisuel [HPN02].

Chapitre 5

Classification supervisée de données

Sommaire

5.1 Classification dans un seul espace réduit 61

5.1.1 Découpage d'une variété continue 62

5.1.2 Différence entre un point et une classe 65

5.2 Classification dans plusieurs sous-espaces réduits 66

5.3 Conclusion 68

Une fois les données réduites à un espace de faible dimension, il est plus aisé de les classer, puisque la malédiction de la dimension est levée. De plus, la variété étant dépliée, un classifieur plus simple pourra être utilisé.

Deux méthodes de classification sont envisagées : la première projette tous les échantillons d'apprentissage avec leur étiquette dans un seul espace réduit, la seconde projette les échantillons dans différents espaces réduits selon leur étiquette.

5.1 Classification dans un seul espace réduit

La classification dans un seul espace permet de comparer directement une classe à une autre. En effet, un point projeté dans l'espace réduit peut y être déplacé vers l'une ou l'autre classe, permettant de fait de visualiser dans l'espace d'origine les modifications associées. Naturellement, l'observation de ces changements n'a pas de sens pour toutes les variétés. Par exemple, si on considère les 20 jeux de la base COIL-20 comme les échantillons d'une seule variété avec 20 classes différentes, il est difficile de transformer un canard (l'une des classes) en une pièce de bois (un autre jeu de données). En effet, il n'existe pas de série d'images (de la base) permettant de passer continuement d'une classe à l'autre.

Classifieur	K-voisins	Frontière linéaire	SVM à noyau gaussien
Original	99.8% (99.8%, 99.8%)	61.3% (50.5%, 69.7%)	99.9% (99.8%, 100%)
ACP	99.8% (99.8%, 99.8%)	69.6% (87.8%, 55.4%)	99.8% (99.8%, 99.8%)
Isomap	99.8% (99.9%, 99.7%)	99.6% (100%, 99.3%)	99.9% (99.9%, 99.9%)
$\mathcal{S}_{\mathrm{P}}$	99.8% (100%, 99.6%)	98.6% (99.0%, 98.2%)	99.7% (99.4%, 99.9%)
LLE	99.6% (99.7%, 99.6%)	98.8% (99.4%, 98.3%)	99.1% (98.6%, 99.5%)
LEM	99.9% (99.9%, 99.8%)	90.1% (77.5%, 100%)	99.2% (98.3%, 99.9%)
DM	99.3% (99.3%, 99.4%)	63.0% (54.5%, 69.6%)	99.4% (99.5%, 99.3%)
HEM	99.8% (100%, 99.6%)	99.3% (99.8%, 98.9%)	99.4% (99.8%, 99.0%)

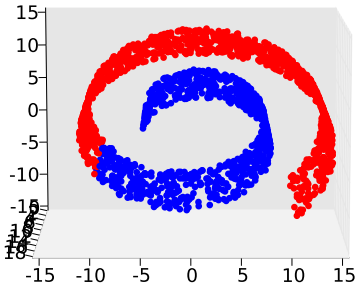
TAB. 5.1: Classification correcte, (vrais positifs, vrais négatifs) en l'absence de bruit (voir 2 pour un rappel des significations des acronymes).

5.1.1 Découpage d'une variété continue

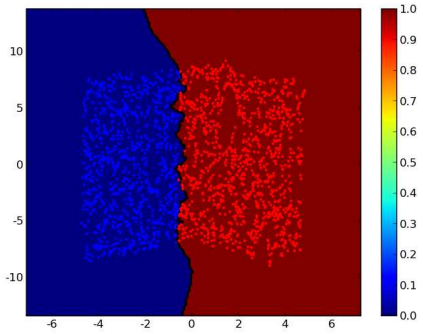
La figure 5.1 présente le SwissRoll découpé en deux classes arbitraires, puis les frontières de décision de plusieurs algorithmes de classification. Chaque classe est représentée par la couleur bleue ou rouge, la limite de décision étant représentée par la couleur verte lorsque cela est possible. Dans cet exemple, la frontière entre les deux classes est linéaire (fig. 5.1a) dans l'espace des coordonnées utilisées pour générer les données. Les *k*-plus proches voisins ($k = 10$ sur la fig. 5.1b) sélectionnent la classe de manière dure (classe 0 ou classe 1) tandis que les SVMs permettent d'obtenir une probabilité continue d'appartenance à l'une ou l'autre classe et donc permettent de se déplacer d'une classe à l'autre. Les classifieurs linéaires permettent de séparer linéairement l'espace réduit (fig. 5.1c). Il est donc primordial d'obtenir un espace réduit aussi fidèle que possible aux distances originales, si la séparation entre les classes à l'origine était linéaire. En revanche, cette séparation linéaire n'est pas généralement pas vérifiée ou vérifiable. Ainsi, le respect des distances originales n'est plus une nécessité. Les SVMs [Vap98] sont représentés ici à l'aide d'un noyau gaussien (fig. 5.1d). Une des classes englobe l'autre, ce qui est dû à la prépondérance d'une classe sur l'autre (880 points appartiennent à la classe 0 et 1120 points à la classe 1). En effet, la frontière de décision (symbolisée par la couleur verte) est concentrée autour de la classe bleue au lieu de s'étendre à l'infini comme sur la figure 5.2d.

Le tableau 5.1 recense les résultats de l'apprentissage sur les 2000 points du SwissRoll. La classification par séparation linéaire ne donne logiquement pas de bons résultats pour les techniques ne permettant pas de déplier la variété (telles que l'ACP ou les *Diffusion Maps*), mais curieusement, les *Laplacian Eigenmaps* ne permettent pas non plus d'obtenir un résultat optimal. Lorsque la variété n'est pas dépliée, il est à noter que cette classification linéaire peut prendre un temps largement plus important que les k-voisins. En ce qui concerne l'utilisation de SVMs avec noyau gaussien, les résultats sont comparables, avec une préférence pour les techniques préservant les distances (Isomap, $\mathcal{S}_{\mathrm{P}}$, ...) plutôt que les similarités (LLE, *Laplacian Eigenmaps*, *Diffusion Maps*, ...).

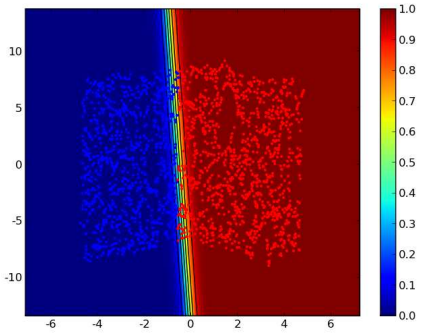
La classification de nouveaux échantillons de la variété est réalisée grâce à la régression



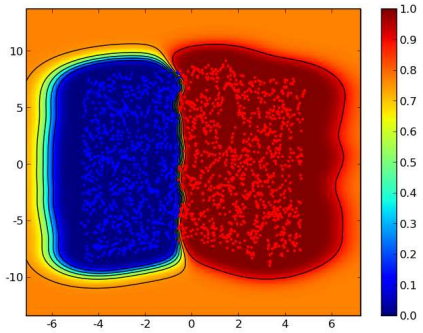
(a) Répartition des classes sur le SwissRoll



(b) Carte de probabilité d'appartenance à une classe (k-voisins)

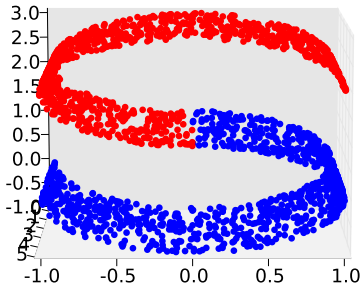


(c) Carte de probabilité d'appartenance à une classe (frontière de décision linéaire)

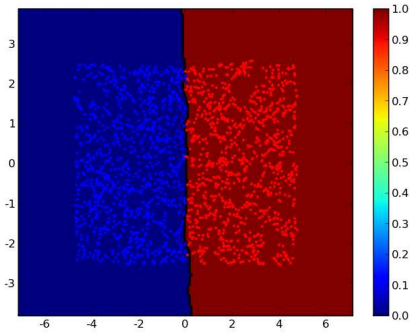


(d) Carte de probabilité d'appartenance à une classe (SVM avec noyau gaussien)

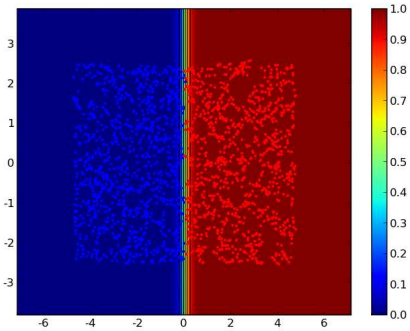
FIG. 5.1: Classification sur le SwissRoll en deux classes.



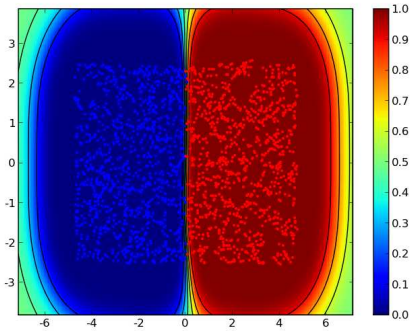
(a) Répartition des classes sur le SCurve



(b) Carte de probabilité d'appartenance à une classe (k-voisins)



(c) Carte de probabilité d'appartenance à une classe (frontière de décision linéaire)



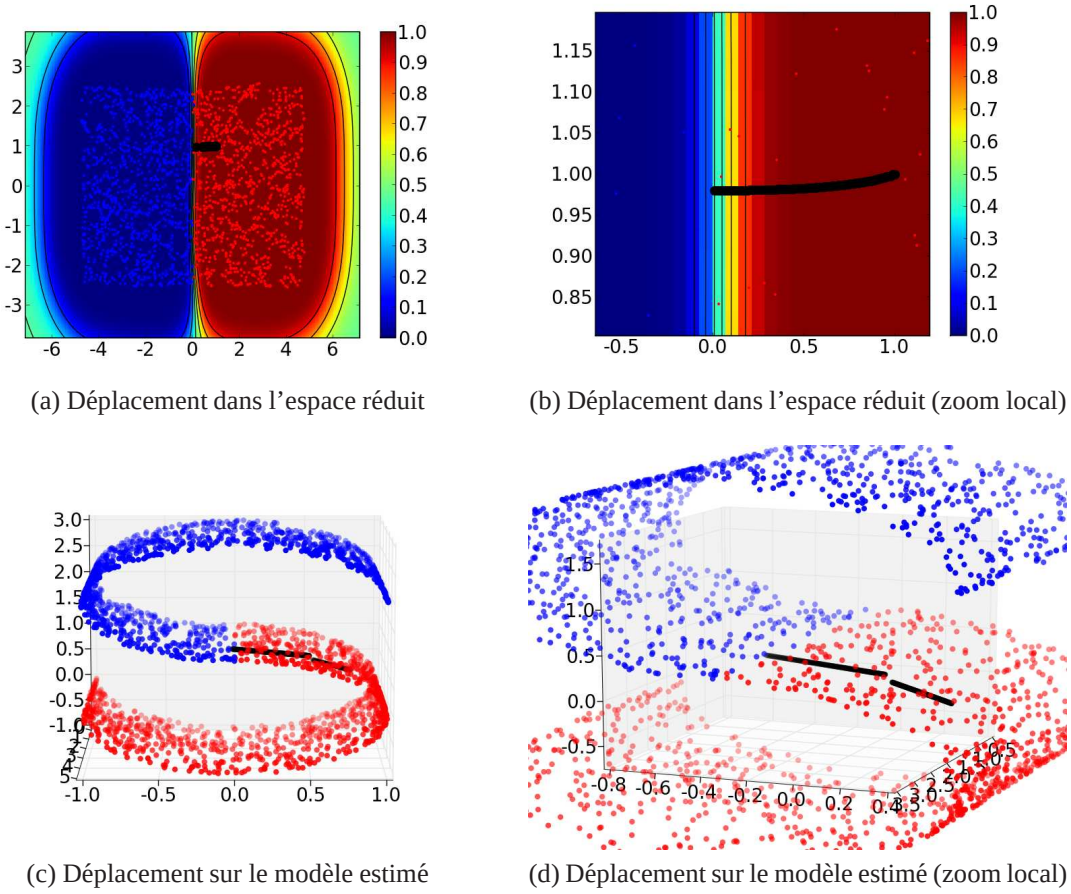
(d) Carte de probabilité d'appartenance à une classe (SVM avec noyau gaussien)

FIG. 5.2: Classification sur le SCurve en deux classes.

construite lors de l'apprentissage. Un nouvel échantillon est projeté dans l'espace réduit et classé dans celui-ci. L'utilisation de la projection avec maximum *a posteriori* permet une amélioration du taux de bonnes classifications de 6% par rapport à une projection par maximum de vraisemblance. La réduction de dimension avec Isomap permet ici d'obtenir des résultats meilleurs qu'avec $\mathcal{S}_{\mathrm{P}}$ (98.6% contre 97.4%), et l'utilisation des k-voisins ou d'une frontière linéaire est plus efficace que l'utilisation de SVMs à noyaux (une différence de 5% en moyenne).

La classification sur le SCurve (fig. 5.2a) donne les mêmes résultats que pour le SwissRoll, à savoir que les compressions avec distances géodésiques obtiennent des résultats élevés (99.8% de vrais positifs ou négatifs). Dans le cas du SCurve, les deux classes ont le même nombre d'échantillons, ce qui explique que la frontière de séparation pour les SVM à noyau (fig. 5.2d) n'entoure pas l'une des deux classes. Un fait intéressant est aussi que la frontière de décision est moins chahutée sur ce jeu de données (fig. 5.2b) que sur le SwissRoll (fig. 5.1b).

La classification sur la base de test du SCurve est aussi meilleure que celle du SwissRoll,



(a) Déplacement dans l'espace réduit

(b) Déplacement dans l'espace réduit (zoom local)

(c) Déplacement sur le modèle estimé

(d) Déplacement sur le modèle estimé (zoom local)

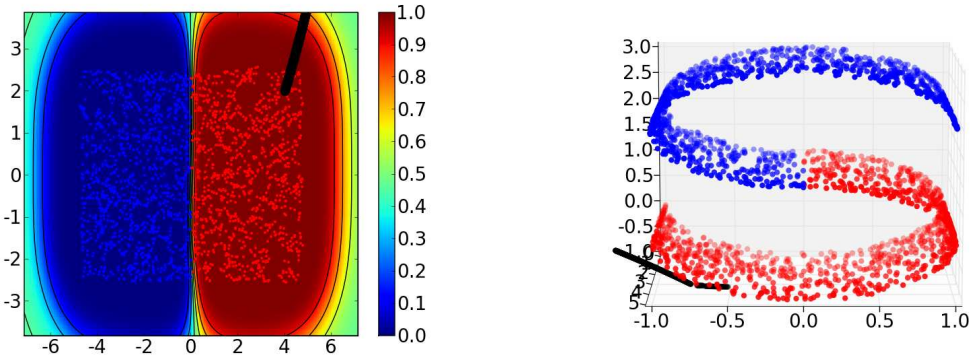
FIG. 5.3: Déplacement d'un point vers une autre classe.

avec des taux de bonnes classifications supérieures à 99.7% pour la réduction avec $\mathcal{S}_{\mathrm{P}}$ et à 98.9% avec Isomap. Avec cette variété, les différents classifieurs obtiennent des résultats comparables (au maximum, une différence de performance de 0.25% a été observée) selon la réduction et la compression utilisée.

Les résultats sont donc contrastés, sur une variété très pliée telle que le SwissRoll, Isomap serait une meilleure méthode de réduction de dimension que $\mathcal{S}_{\mathrm{P}}$, tandis que pour une variété plus simple telle que le SCurve, $\mathcal{S}_{\mathrm{P}}$ a un avantage. Il est aussi à noter que la réduction à l'aide de $\mathcal{S}_{\mathrm{SAM}}$ obtient des résultats semblables à Isomap.

5.1.2 Différence entre un point et une classe

Pour parcourir l'espace des points d'une classe à une autre, il est nécessaire d'avoir une fonction de distance, comme la probabilité d'appartenance à la classe. Sans cette notion, le parcours est aléatoire et il est impossible de sélectionner une direction vers la classe désirée.



(a) Déplacement dans l'espace réduit

(b) Déplacement dans l'espace réduit (zoom local)

FIG. 5.4: Mauvais déplacement d'un point vers une autre classe.

Un point dans l'espace réduit a été déplacé sur le SCurve d'une classe à une autre (fig. 5.3). Son déplacement s'oriente vers la frontière de décision dans l'espace réduit (marqué par une couleur verte) et s'arrête à celle-ci. Un zoom sur le déplacement montre bien que le déplacement n'est pas nécessairement en ligne droite. Le déplacement peut se traduire en 3D par un parcours sur plusieurs modèles linéaires. C'est ce qui se produit sur cet exemple.

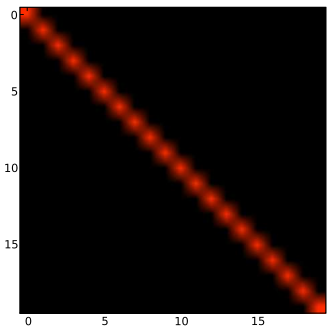
Lorsque les échantillons ne recouvrent pas tout l'espace, comme c'est le cas pour la base COIL-20, il est nécessaire d'ajouter une connaissance *a priori* pour guider la recherche, soit sous la forme d'une contrainte (il est nécessaire de rester dans des zones vraisemblables) ou à l'aide d'une régularisation (un terme additionnel dépendant de la vraisemblance de la zone considérée). Dans les deux cas, la contrainte ou la régularisation devra nécessiter une paramétrisation particulière.

Ces deux solutions doivent aussi être envisagées pour le parcours sur le SCurve. En effet, le point se déplace vers la frontière de décision la plus proche. Dans l'exemple proposé, la frontière s'étend à l'infini. Il est donc possible que le déplacement du point l'entraîne à l'extérieur de la zone vraisemblable, auquel cas le point final trouvé par cette recherche n'apporte aucune information pertinente sur la différence entre le point original et la classe considérée (fig. 5.4).

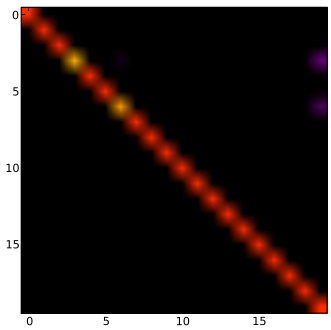
5.2 Classification dans plusieurs sous-espaces réduits

La classification à partir de plusieurs sous-espaces consiste à projeter un nouvel échantillon dans chacun de ces espaces puis de sélectionner le meilleur projeté au sens d'un certain critère parmi les différents candidats. Par la suite, si chaque espace correspond à une classe d'apprentissage, le meilleur projeté indique aussi la classe à laquelle le point appartient.

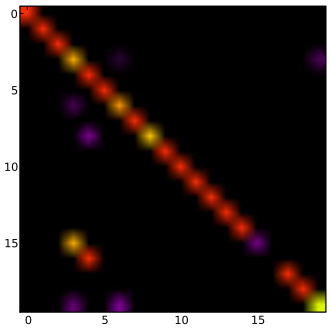
Un test a été effectué sur la base COIL-20, avec une variété par jeu de données. Les données d'apprentissage (1440 images) sont toutes correctement classées (fig. 5.5a). En revanche, les



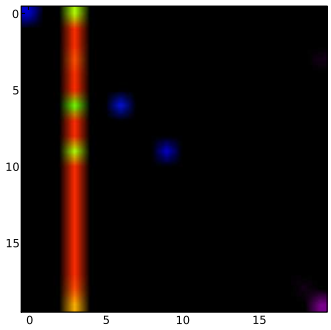
(a) Projection avec maximum *a posteriori*



(b) Projection robuste avec maximum *a posteriori*



(c) Projection robuste avec 10% d'occlusion



(d) Projection robuste avec 40% d'occlusion

FIG. 5.5: Classification sur la base COIL-20 sans et avec occlusion. Les ordonnées représentent les classes auxquelles appartiennent les échantillons et les abscisses les classes attribuées par les algorithmes. Plus la couleur est chaude, plus le nombre d'échantillons classés est important.

tests effectués sur les images avec occlusion montrent qu'une seule classe est de plus en plus probable à mesure que l'occlusion s'agrandit. La classe en question est une image sombre dont les éléments centraux sont noirs, ce qui explique que les autres classes sont moins probables.

La figure 5.5 expose ces résultats. Chaque ligne correspond à un jeu de données projeté. Plus la couleur est chaude (tirant vers le rouge), plus le nombre d'échantillons projetés dans cette classe a été important. Une diagonale rouge est donc l'optimum à obtenir. Lorsqu'une projection robuste est utilisée, les résultats ne sont pas parfaits, même sans occlusion. Pour 10% d'occlusion, deux classes sont très mal classées. A 40% d'occlusion, seuls des éléments marginaux sont bien classés, les autres sont regroupés dans une seule classe.

5.3 Conclusion

La réduction de dimension conjuguée à une projection dans un espace réduit permet de classer efficacement des données. Qu'il y ait un seul espace réduit dans lequel diverses classes existent ou autant d'espaces réduit que de classes, cette chaîne de traitement permet de classer un nouveau point.

L'intérêt d'un espace réduit unique est de pouvoir déplacer un point classé dans une classe vers une autre classe puis de visualiser la différence entre le point original et le point déplacé. Plusieurs espaces réduits distincts sont nécessaires lorsque les différentes classes ne sont pas connectées (le graphe des plus proches voisins n'est pas connexes). Dans ce cas, l'intérêt de déplacer un point est nul car il n'existe pas de manière de transformer un point en un autre d'une autre classe.

Chapitre 6

Applications

Sommaire

6.1 Classification supervisée de textures 69

6.1.1 Descripteur de texture 70

6.1.2 Apprentissage et classification de textures 70

6.2 Analyse de formes 73

6.2.1 Génération de champs synthétiques 73

6.2.2 Apprentissage et tests statistiques 74

Les différents jeux de données illustrant les chapitres précédents étaient des données simulées. Dans ce dernier chapitre, des textures seront analysées, puis des formes synthétiques. Dans chacun de ces exemples, les échantillons ne seront pas analysés directement. A la place, ils seront transformés dans un espace adapté à leur analyse (les matrices de cooccurrence pour les textures et les champs de déformation pour les formes synthétiques).

6.1 Classification supervisée de textures

L'analyse de texture est un problème de traitement d'image dont les premières études remontent à plus de 30 ans. Une texture n'est pas une image unique, mais un ensemble d'images qui sont ressenties par le cerveau humain comme étant identiques. Ainsi, des images d'un fragment de tissu constitueront des échantillons d'une même texture qu'il faut caractériser.

L'objectif de cette section est de présenter une méthode d'analyse de textures. Celles-ci seront caractérisées par un nombre défini de classes et d'échantillons associés à chaque classe qui serviront de base d'apprentissage. D'autres échantillons serviront d'échantillons de test qui seront classés.

6.1.1 Descripteur de texture

Un échantillon d'une texture est une image. Toute transformation rigide de cette image donnera un autre échantillon de la texture. Il est donc nécessaire de transformer l'image pour extraire l'information pertinente. La littérature [Har79] propose d'utiliser la matrice de cooccurrence $\boldsymbol{M}$ de l'échantillon analysé. Il s'agit d'estimer une densité conjointe d'ordre deux sur l'image, ce qui correspond à des statistiques spatiales utilisées par le système visuel humain [Jul62].

Une nouvelle image carrée $\boldsymbol{M}$ est créée et initialisée à $\mathbf{0}$. La taille de cette image est fonction de l'échantillonnage de l'image traitée, appelée ici $\boldsymbol{I}$. Par exemple si l'image comporte 32 niveaux de gris, la matrice de cooccurrence est de taille 32x32. Par la suite, un vecteur déplacement dans l'image est choisi, $\boldsymbol{v} = (1, 0)$ par exemple. Pour chaque point $\boldsymbol{x}$ de l'image, le pixel $\boldsymbol{M}(\boldsymbol{I}(\boldsymbol{x}), \boldsymbol{I}(\boldsymbol{x} + \boldsymbol{v}))$ est incrémenté.

Afin de rendre cette transformation à peu près invariante par rotation, une telle image est calculée pour d'autres vecteurs $\boldsymbol{v}$ (généralement $(0, 1)$, $(-1, 0)$ et $(0, -1)$, mais des déplacements diagonaux sont parfois aussi proposés). L'image finale utilisée est alors la moyenne de toutes ces images.

Dans le cas d'une texture de couleur, une matrice de cooccurrence est calculée pour chaque canal, puis pour chaque combinaison de deux canaux [ADBB04]. Ainsi, 6 matrices de cooccurrence sont formées (les matrices (rouge, bleu) et (bleu, rouge) étant identiques en raison de l'invariance par rotation de $180°$ utilisée lors de la moyenne des matrices de cooccurrence).

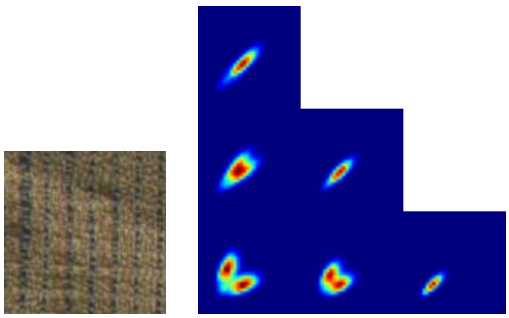
6.1.2 Apprentissage et classification de textures

Les textures analysées sont extraites de la base Outex [OPM+02]. Une sélection de 68 images de cette base a été choisie, chaque image ayant été prise sous des conditions semblables (cela n'a un impact que si les images considérées ont une dynamique très différente). Par la suite chacune de ces images de taille 746x538 a été découpée en 20 échantillons de taille 128x128. Cet ensemble de 20 échantillons constitue une texture unique. Un échantillon sur deux a été utilisé pour la base d'apprentissage, l'autre pour la base de test, selon un damier.

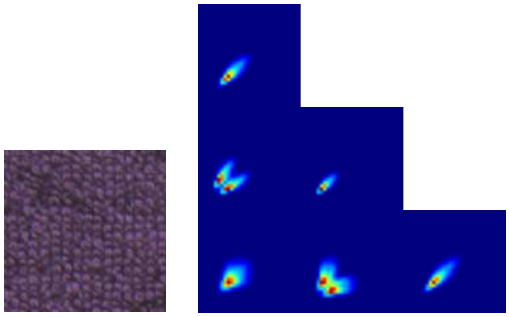
La figure 6.1 montre un échantillon pour les quatre premières textures. A côté de chaque texture se retrouvent les six matrices de cooccurrence qui seront utilisées comme vecteur de paramètres de chaque échantillon.

Lors de la phase d'apprentissage, chaque texture bénéficie d'un modèle appris avec notre chaîne de traitement ($\mathcal{S}_{\text{P}}$/PLM2/MAP). Le nombre de proches voisins utilisés est de 6, afin de garantir un graphe connexe. Avec seulement 10 échantillons, le modèle reste simple (un à deux plans lors de l'apprentissage).

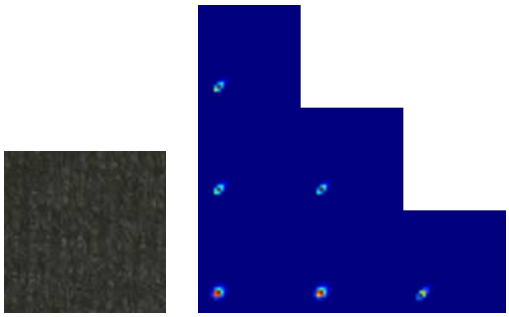
La figure 6.2 expose les résultats tirés de la base d'apprentissage et de la base de test. La base d'apprentissage est classée correctement à 99.9%, tandis que la base de test l'est à 96.5%. D'après la littérature [ADBB04], en utilisant une réduction de dimension linéaire à l'aide de l'analyse factorielle discriminante, suivie d'une classification à l'aide des 5 plus proches voisins,



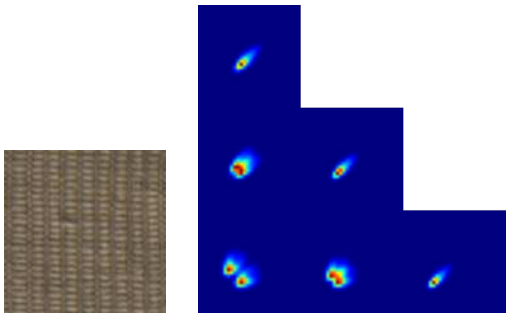
(a) Texture 1 (b) Matrice de cooccurrence de la texture 1



(c) Texture 2 (d) Matrice de cooccurrence de la texture 2

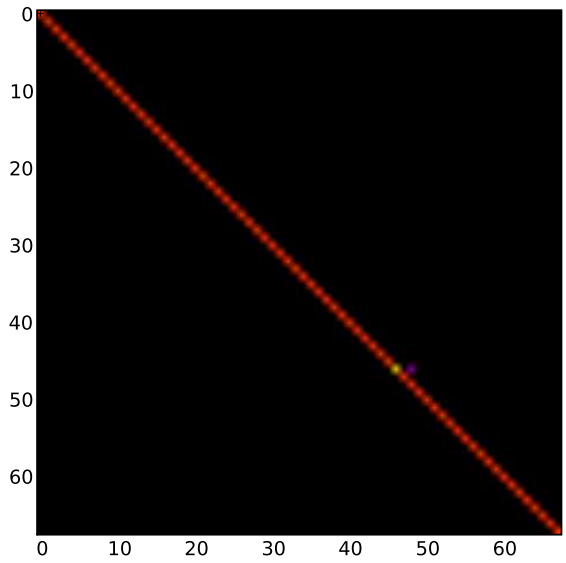


(e) Texture 3 (f) Matrice de cooccurrence de la texture 3

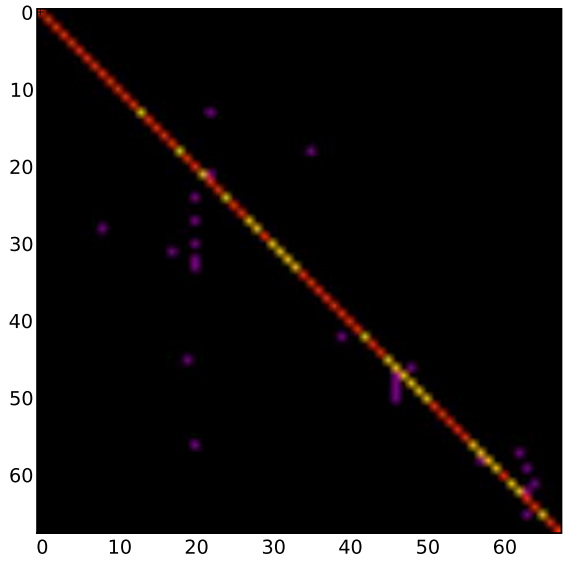


(g) Texture 4 (h) Matrice de cooccurrence de la texture 4

FIG. 6.1: Echantillons de la base Outex et transformées associées.



(a) Base d'apprentissage



(b) Base de test

FIG. 6.2: Classification sur la base Outex. Les ordonnées représentent les classes auxquelles appartiennent les échantillons et les abscisses les classes attribuées par les algorithmes. Plus la couleur est chaude, plus le nombre d'échantillons classés est important.

le taux de réussite dans la base de test est de 94.9%. La méthode proposée obtient donc de meilleurs résultats

Contrairement à la transformation qui sera utilisée sur les formes, la création de la matrice de cooccurrence ne permet pas de visualiser les différences entre un échantillon et une texture sur ce même échantillon. Dans le cas présent, cet inconvénient n'a pas d'impact sur les traitements, les différences entre un échantillon et une texture n'étant pas visualisables.

Traditionnellement, la matrice de cooccurrence n'est pas utilisée directement pour décrire un échantillon d'une texture. On utilise généralement 5 des 14 paramètres de Haralick [Har79] (voir annexe B.3 pour les formules de calcul de ces paramètres). En considérant l'ensemble des 6 matrices, 30 paramètres sont donc extraits par échantillon. Après apprentissage et projection, le taux de réussite sur la base d'apprentissage est de 97.4% et de 93.1% sur la base de test. Le score est donc inférieur à la littérature et à l'utilisation de la matrice de cooccurrence comme paramètre. Ce dernier point s'explique par la perte d'information lors de cette transformation, sachant que l'étape de réduction de dimension supprime par la suite l'information redondante dans la matrice.

Une transformation peut donc être utile pour mettre en valeur l'information recherchée. Dans le cas de la texture, il peut s'agir de la matrice de cooccurrence, qui est semblable à un critère visuel. En revanche, une transformation qui ne met pas en valeur cette information ou qui la supprime n'améliore pas les scores de la chaîne de traitement, comme c'est le cas pour les critères de Haralick pour de l'analyse de texture.

6.2 Analyse de formes

L'analyse de formes est un problème complexe. En effet, il est nécessaire de définir la distance entre deux formes. Cela n'est pas possible avec une simple norme euclidienne du vecteur image, car deux formes sont semblables si une translation et/ou une rotation permet de passer de l'une de l'autre et/ou translatée l'une par rapport à l'autre. L'objectif d'une transformation est de gommer ces différences et de mettre en valeur les vraies différences entre les formes. Dans le cas présent, le champ de déformation entre une image et une image de référence est choisi.

6.2.1 Génération de champs synthétiques

Les formes étudiées ici sont des sphères. Celles-ci sont déformées en plusieurs points indépendants sur la sphère (figure 6.3). Ces points peuvent être donc considérés comme les paramètres de la forme. A partir du déplacement de ces points et de points de contrôles (points où le déplacement est nul), les autres déplacements sont interpolés à l'aide de *thin-plates splines* [Boo89]. La première population est caractérisée par deux points (A et B), et un échantillon correspondant à ces déplacements est représenté sur la figure 6.4b.

Une deuxième population est générée en déplaçant deux autres points (C et D) à l'opposé des points déplacés dans la première population (figure 6.4c). Enfin, la troisième population contient les déplacements des quatre points considérés (A, B, C et D).

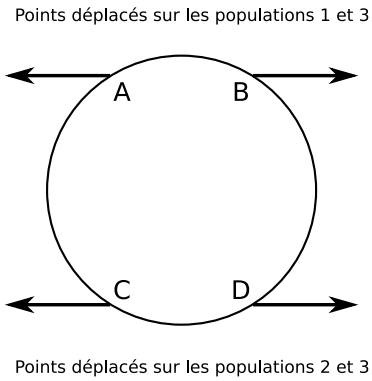


FIG. 6.3: Points déplacés sur la sphère pour les trois populations.

La sphère originale a 32 voxels de diamètre, l'image totale étant un cube de 64 voxels de côté. Le champ de déformation est constitué d'un cube de 64 voxels de côté contenant en chacun d'eux 3 composantes. Les points de contrôle sont différents selon la population considérée (certains points de contrôle pour une population étant positionnés à l'endroit où se trouve l'un des points déplacés).

Les déplacements considérés sur la première population le sont uniquement selon un axe, en fonction de deux paramètres aléatoires qui définissent les absisses sur les segments de la figure 6.3. Dans le cas de la deuxième population, lorsque l'un des paramètres dépasse un seuil, le point déplacé associé se déplace selon deux axes, et non plus un seul (il s'agit de l'axe perpendiculaire à la vue, donc non visible sur la figure 6.3).

Les amplitudes des déplacements ont été choisies majoritairement proches de zéro, avec des points plus éloignés. Les échantillons de la deuxième population projetés dans l'espace réduit de la première population doivent se retrouver proches des points de la première population dont l'amplitude des déplacements est faible.

Pour chacune des trois populations, 100 échantillons ont été générés. Lors de l'apprentissage de la première ou de la deuxième variété, 50 échantillons ont été utilisés. L'ensemble des 250 points restants a été utilisé pour tester la projection.

6.2.2 Apprentissage et tests statistiques

Afin de minimiser les temps de calcul, seule une partie des points (une enveloppe autour de la sphère, représentant environ 5000 sur les 262144 que compte un échantillon) des images a été considérée. Le champ de déformation généré transforme la sphère en l'image d'arrivée. Ainsi, en chaque point de la sphère, on définit le vecteur de déplacement pour tous les paramètres possibles. C'est aussi ce champ qui est estimé dans les outils de recalage utilisés (pour des raisons d'interpolation de l'image résultat). De plus, lors de l'analyse des résultats, ces données seront les seules à pouvoir être affichées sur la sphère. C'est donc cet ensemble de points (sur

(a) Sphère originale

(b) Échantillon de la première population

(c) Échantillon de la deuxième population

FIG. 6.4: Echantillons des différentes populations.

une certaine épaisseur) qui sera utilisé comme attributs de l'image.

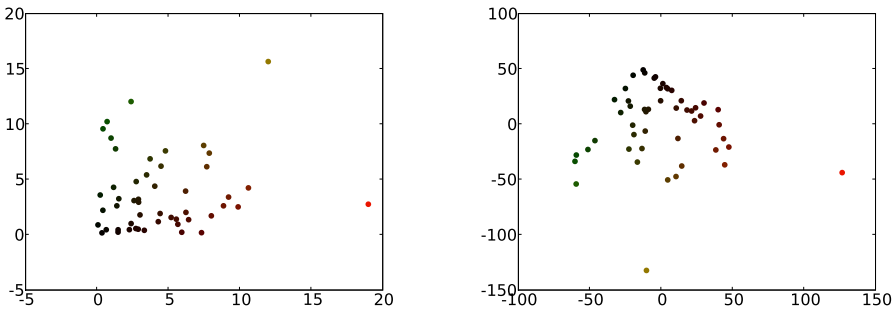
La réduction de dimension a été effectuée à l'aide de $\mathcal{S}_P$ en utilisant les 8 plus proches voisins. La figure 6.5 montre la répartition des paramètres aléatoires utilisés pour générer les données puis la répartition des points dans l'espace réduit. La structure est donc respectée par l'algorithme de réduction de dimension. Les points sombres sont ceux dont les déplacements sont les plus faibles.

Par la suite, la régression a été effectuée à l'aide de l'algorithme PLM2, en considérant les 6 plus proches voisins. Lors de cette phase, au moins 5 modèles linéaires ont été à chaque fois générés.

La répartition des points de test projetés dans l'espace réduit est intéressante (fig. 6.6). La première population (représentée par des disques) est projetée de manière satisfaisante, selon les déplacements rencontrés. Les points de la deuxième population (représentés par des losanges rouges) projetés sur la variété devraient présenter de faibles déplacements. C'est le cas puisque la figure montre les losanges localisés dans les zones sombres, là où les déplacements sont les plus faibles, sur une surface très réduite. Enfin, la troisième population qui présente les quatre déplacements se projette elle aussi de manière satisfaisante dans l'espace réduit des deux premiers déplacements.

Les figures 6.7 et 6.8 montrent des résultats similaires pour l'apprentissage de la deuxième population. Ainsi, l'introduction d'un terme non linéaire ne porte pas préjudice à la réduction de dimension.

L'analyse des différences nécessite le calcul d'un z-score pour chaque point de l'image. Il s'agit de mesurer la probabilité d'apparition de l'erreur associée au point. Dans le cas présent, un point possède 3 coordonnées, il n'est donc pas possible de calculer analytiquement ce score. La solution retenue est de comparer la distance entre l'erreur d'un point et la moyenne de la variable d'erreur à un certain nombre d'échantillons de la variable d'erreur (ici, 10000). On



(a) Répartition des paramètres de la première population

(b) Espace réduit après réduction de dimension

FIG. 6.5: Conservation de la structure lors de la réduction de dimension (population 1).

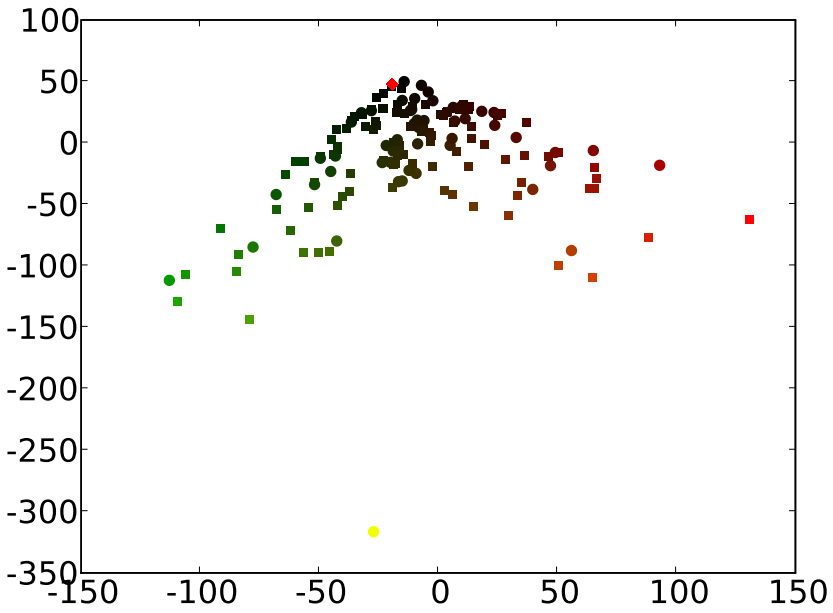
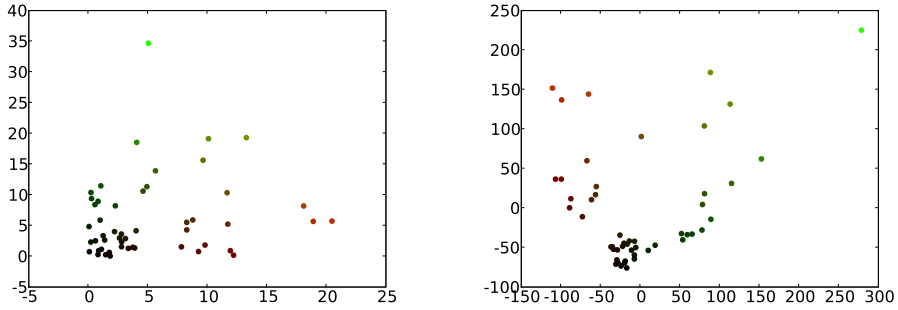


FIG. 6.6: Conservation de la structure lors des projections sur la première population. Les échantillons de test de la première population sont symbolisés par des disques, ceux de la deuxième population par des losanges rouges et ceux de la troisième population par des carrés.



(a) Répartition des paramètres de la deuxième population

(b) Espace réduit après réduction de dimension

FIG. 6.7: Conservation de la structure lors de la réduction de dimension (population 2).

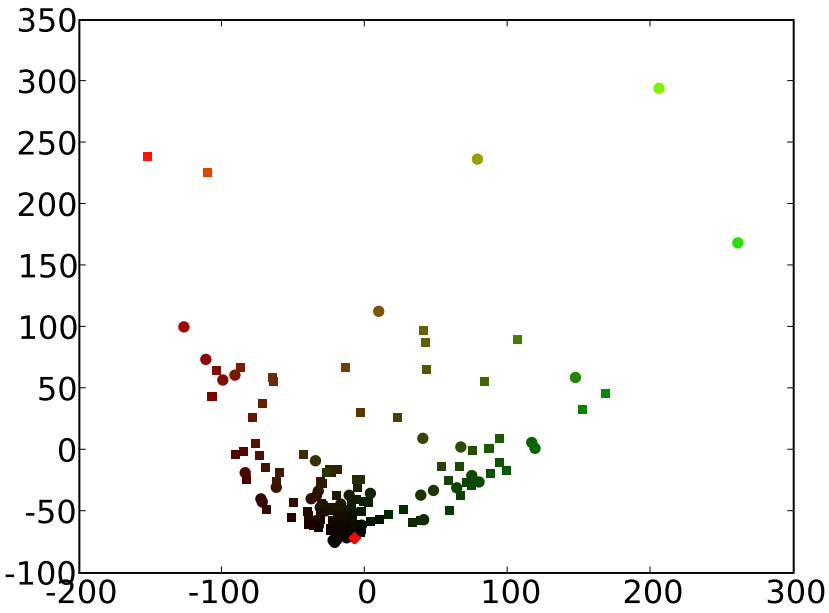
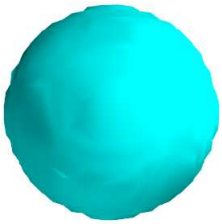


FIG. 6.8: Conservation de la structure lors des projections sur la deuxième population. Les échantillons de test de la deuxième population sont symbolisés par des disques, ceux de la première population par des losanges rouges et ceux de la troisième population par des carrés.

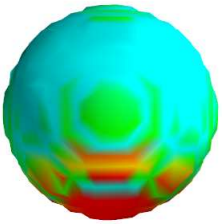
calcule le nombre d'échantillons dont la distance à la moyenne est inférieure à celle de l'erreur et on le divise par le nombre total d'échantillons générés. On obtient ainsi un score approché.

Un échantillon de chaque population a été projeté sur les deux variétés 6.9. L'échantillon de la première population (fig. 6.9a) ne présente pas de variation par rapport à la normalité, ce qui est attendu, tout comme la projection d'un échantillon de la deuxième population sur la deuxième variété (fig. 6.9d). La projection d'un élément de la deuxième (fig. 6.9b) ou de la troisième population (fig. 6.9c) sur la première variété montre des anomalies localisées aux points déplacés qui ne le sont pas dans la première population. Les projections sur la deuxième variété (première population sur la fig. 6.9e et troisième population sur la fig. 6.9f) permettent aussi une localisation des déformations différentes. Dans le cas d'un déplacement linéaire (la première population) comme dans le cas d'un déplacement non linéaire (la deuxième population), il est possible de repérer des déformations n'appartenant pas au modèle grâce à la chaîne de traitement proposée.

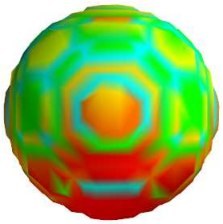
Cependant, la localisation des différences n'est pas parfaite. En effet, sur la figure 6.9c, tout un hémisphère indique une anomalie, ce qui n'est pas le cas en réalité. Il faut se rappeler que les points de contrôle sont peu nombreux (un peu plus de 300) et que plus de 5000 points ont été testés. Ces points sont proches des points de contrôle, mais la déformation générée les déplace de manière anormale par rapport à la population. Comme l'opération de visualisation moyenne des informations proches, on obtient les différentes images proposées.



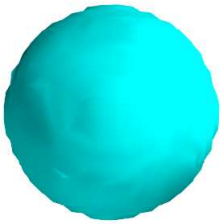
(a) Projection sur la première variété d'une forme appartenant à la première population



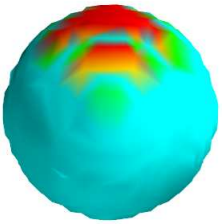
(b) Projection sur la première variété d'une forme appartenant à la deuxième population



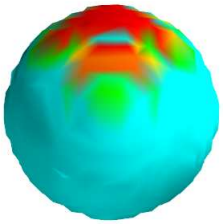
(c) Projection sur la première variété d'une forme appartenant à la troisième population



(d) Projection sur la seconde variété d'une forme appartenant à la première population



(e) Projection sur la seconde variété d'une forme appartenant à la deuxième population



(f) Projection sur la seconde variété d'une forme appartenant à la troisième population

(g) Carte de couleur des *z*-score

FIG. 6.9: Localisation des différences lors de projections sur les deux variétés.

Chapitre 7

Conclusion et perspectives

Dans ce chapitre final, un résumé de cette thèse et des contributions associées sera proposé. A titre de conclusion, les perspectives de futurs travaux sont présentées.

7.1 Résumé et discussion

Cette thèse et ses contributions ont proposé une modélisation innovante de données et son utilisation en vue de classifications.

La modélisation comprend une réduction de dimension robuste, permettant de faire face au bruit et respectant la structure des données dans un espace de dimension réduite. Cette méthode a été comparée aux méthodes usuelles de la littérature à l'aide de plusieurs jeux de données (SwissRoll, SCurve, base COIL-20). Par la suite, une méthode de régression linéaire par morceaux a été exposée, permettant de créer un lien entre l'espace de dimension réduite et l'espace d'origine. Contrairement aux méthodes présentées dans la littérature, l'approche choisie adapte le nombre de modèles linéaires nécessaires à la précision de modélisation recherchée.

Grâce à cette modélisation, une projection robuste de nouveaux échantillons sur la variété a été possible. Cette méthode a permis d'obtenir des résultats d'aussi bonne qualité ou de meilleure qualité que les approches robustes basées sur l'analyse en composantes principales qui est la méthode couramment utilisée pour l'apprentissage de variété. Une méthode originale de classification supervisée a été proposée, en modélisant chaque classe comme une variété puis en projetant chaque nouvel échantillon sur chacune des variétés et en sélectionnant le meilleur projeté (au sens d'un certain critère).

Des applications pratiques ont été entreprises sur des données de texture ou sur des formes synthétiques. Ces applications ont confirmé les performances de l'approche proposée. En effet, la classification sur les textures est meilleure que les approches proposées dans la littérature et les tests sur les formes ont permis de détecter des différences localisées sur des formes anormales.

Plusieurs bibliothèques de traitement ont été développées et améliorées dans le cadre de cette thèse. Elles ont été programmées en C++ ou en Python, en utilisant les avantages de

chaque langage (la vitesse ou l'utilisation des gabarits du C++ pour les calculs ; la souplesse ou l'encapsulation de Python pour la création des prototypes ou la création d'une bibliothèque d'optimisation). Enfin, ces bibliothèques ont été mises à disposition dans le logiciel interne *Medimax* et son successeur *MediPy*.

7.2 Perspectives

Les perspectives d'amélioration des approches exposées dans cette thèse sont diverses. Elles concernent principalement la première étape de réduction de dimension. La phase de régression quant à elle bénéficierait grandement d'une meilleure approche par blocs.

7.2.1 Réduction de dimension

La méthode de réduction de dimension robuste exposée dépend fortement de la qualité de la sélection des plus proches voisins. Si celle-ci permet la création de court-circuits dans les données d'apprentissage, l'espace réduit calculé ne sera pas optimal. Plusieurs algorithmes actuels sélectionnent de manière plus robuste le nombre de voisins et leur position qu'une simple recherche des k plus proches.

Les efforts se sont portés sur le respect des distances dans l'espace d'origine et dans l'espace réduit. Or les distances sur une variété riemannienne de dimension d se représentent dans un espace euclidien de dimension au plus $2d + 1$ [Gre69]. Cela signifie que la réduction de dimension peut ne pas être optimale et qu'il reste des possibilités pour obtenir une représentation dans un espace de dimension d. Une possibilité consisterait à rechercher un espace réduit pour tous les points, puis effectuer une phase de régression. Par la suite, les modèles linéaires obtenus définiraient des cartes locales dans lesquelles une nouvelle étape de réduction de dimension pourrait être effectuée, permettant d'obtenir une carte locale dans $\mathbb{R}^d$.

7.2.2 Régression

La régression par blocs présentée ne permet pas de modéliser de manière convenable des images. En effet, les artefacts dus au partitionnement des coordonnées reste visible. Une nouvelle métrique de séparation doit être envisagée, permettant de mieux mesurer si deux attributs peuvent être reconstruits par la même fonction linéaire par morceaux et permettant un meilleur rendu visuel.

Enfin, les modèles considérés ont toujours été linéaires, mais il est possible d'envisager des modèles non linéaires.

7.2.3 Classification

Lorsque l'objectif de la modélisation des données est la classification supervisée, il est nécessaire de tirer parti de cette information supplémentaire. En effet, lorsque la réduction de

dimension est effectuée sur la totalité des coordonnées des échantillons, les informations pertinentes peuvent être absorbées par la variabilité des données. Un algorithme de sélection d'attributs [DL97] peut être couplé à la modélisation. Dans ce cas, une modélisation de la variété est effectuée sur un jeu restreint de coordonnées puis une classification est effectuée. Selon les résultats de cette classification, le jeu restreint de coordonnées peut être modifié. Ainsi, la non-linéarité de la variété est introduite dans le processus de sélection des attributs.

Annexe A

Optimiseurs

Sommaire

A.1 Notions d'architecture logicielle 86
A.1.1 Développement dirigé par les tests (*Test-Driven Development*) 86
A.1.2 Orientation objet 87
A.1.3 *Design patterns* 88
A.1.4 Principe de séparation 88
A.2 Application à la création d'un module d'optimisation 89
A.2.1 Le coeur, la classe Optimizer 89
A.2.2 Les différents modules disponibles 90
A.2.3 Différentiation automatique 90
A.2.4 Estimation 91
A.2.5 Limites du modèle 92
A.3 Conclusion 92

Ecrire un algorithme d'optimisation convexe non contraint est un exercice qui peut être compliqué. L'algorithme de base 6 est simple et montre un découpage en 3 fonctions principales.

Entrées : Fonction à optimiser et point de départ
Sorties : Jeu de données optimal pour la fonction
début
  tant que *le critère n'est pas respecté* **faire**
    Recherche d'une direction de descente;
    Recherche d'un pas suivant cette direction
  fin
fin

Algorithme 6 : Algorithme d'optimisation simple

Malheureusement, la simplicité de cette présentation cache les difficultés de mise en oeuvre d'une boîte à outils dédiée à l'optimisation qui n'est pas limitée à une série de fonctions, comme

c'est le cas pour Matlab et Scipy[1]. En effet, il existe dans la littérature des dizaines de fonctionnalités pouvant être implémentées pour chacune des 3 fonctions principales, chacune d'elles nécessitant des connaissances différentes. Par exemple, l'utilisation d'une descente de gradient classique ne nécessite pas la taille du pas précédent, contrairement à une descente de gradient conjuguée. Ces différences soulignent l'importance d'une conception souple et modulaire pour une boîte à outils d'optimisation. La conception d'un cadre clair nécessite certaines notions de conception ainsi que de méthodologie.

A.1 Notions d'architecture logicielle

Dans un premier temps, la méthodologie de développement sera présentée. C'est elle qui gouverne le processus de développement et permet de valider le résultat final, c'est-à-dire certifier que l'outil correspond à ce qui est attendu. Par la suite, l'orientation objet est introduite. Ce concept permet d'introduire la modularité, à l'aide de l'héritage, du polymorphisme ou de l'encapsulation des données. Grâce à ce concept, les motifs de conception (*design patterns*) ont permis de résoudre des problématiques classiques rencontrées. Enfin, le principe de séparation permet de découpler des algorithmes des données traitées.

A.1.1 Développement dirigé par les tests (*Test-Driven Development*)

La notion de Développement dirigé par les tests (ou TDD) est tirée d'un ouvrage de Ken Beck [Bec02]. Ken Beck présente une méthodologie simple qui peut être associée à d'autres méthodes comme *eXtreme Programming* [BBMW04][2]. Contrairement au traditionnel cycle en V[3], l'objectif est de proposer à toute étape du développement un programme répondant aux besoins de l'utilisateur. Cela est réalisé à l'aide de tests qui sont écrits avant d'implémenter effectivement l'opération.

Afin d'obtenir à chaque étape un programme fonctionnel, il est nécessaire de ne pas tout tester et programmer en même temps. Au départ, un seul test est réalisé, avec le code correspondant. Puis une fois que le test est valide, on effectue une séance de *refactoring*[4], traditionnellement en utilisant des *design patterns* pour rendre cette architecture plus souple face aux autres éléments à ajouter au programme. Dans le cas de cette bibliothèque d'optimisation, la structure générale est déjà définie. En revanche, la refactorisation permet de réutiliser efficacement des routines existantes ; par exemple pour le calcul d'un gradient conjugué selon plusieurs

[1] http ://www.scipy.org/

[2] *eXtreme Programming* est une méthodologie de développement souple, basée sur plusieurs règles claires et définies. Le principal objectif de toutes ces méthodologies est de répondre au besoin du client au fur et à mesure, afin d'être en adéquation avec ses besoins réels.

[3] Le cycle en V est une ancienne méthodologie de développement où les besoins du clients sont définis en amont, les tests sont effectués après la phase de programmation, et après cette phase, le client vérifie le produit. Entre temps, la programmation et/ou l'interprétation des besoins a pu diverger des besoins réels du client.

[4] Cette phase de développement a pour but de réorganiser le code afin d'améliorer son architecture et donc sa souplesse et sa robustesse.

méthodes, la même structure sera utilisée.

Les tests unitaires utilisés ici existent à plusieurs niveaux :

– pour chaque sous-module, un ou plusieurs test(s) simple(s) permettra(ont) de vérifier si le module fonctionne correctement,
– une série de fonctions usuelles ainsi que leurs points de départ associés seront utilisés pour vérifier la convergence d'un optimiseur.

Dans ces tests, aucune part d'aléatoire n'est permis. En effet, il faut être sûr que l'optimisation réussira avec une certaine précision. Si un départ aléatoire est utilisé, le résultat final n'est pas garanti et le test n'est plus pertinent.

A.1.2 Orientation objet

La programmation orientée objet repose sur plusieurs techniques, dont l'héritage, le polymorphisme et l'encapsulation. Contrairement à la programmation usuelle procédurale, la programmation orientée objet promeut l'interaction entre des objets. Chaque objet est une instance d'une certaine classe qui elle-même peut être un objet dans certains langages (comme Python). Une classe est en réalité un patron décrivant le fonctionnement d'un objet.

L'héritage consiste à demander à réutiliser ce qu'on appelle l'interface d'une classe, comme les méthodes ou les attributs dits publics (visibles de l'extérieur). Le polymorphisme permet d'utiliser un type d'objet à la place d'un autre, soit parce que l'un hérite de l'autre, soit parce que leur interface est identique (on parle de polymorphisme dynamique ou statique). Enfin, l'encapsulation enferme les attributs d'un objet de sorte qu'ils ne soient plus accessibles de l'extérieur (on parle aussi de responsabilité, seule la classe est responsable des données qu'elle contient).

Par exemple, il existe plusieurs types de chaînes hi-fi, mais elles partagent toutes une interface commune : un bouton marche/arrêt, un bouton de lecture, d'arrêt, ... (polymorphisme). Les chaînes hi-fi les plus récentes ont hérité des fonctionnalités de leurs ancêtres (par exemple le support d'une télécommande a été ajouté par rapport à la génération précédente). L'encapsulation se manifeste par le fait qu'il n'est pas permis ni conseillé de modifier les circuits d'une chaîne hi-fi. On accède aux attributs de l'objet par des écrans LCD, par divers boutons, mais rien d'autre ne devrait être accessible. Pour mettre en marche la chaîne hi-fi, on appuie sur un bouton qui déclenchera le réveil de l'objet, ce n'est pas à l'utilisateur de réveiller chaque élément interne de la chaîne hi-fi.

De même en informatique, le principe de l'orientation objet permet de créer des objets, de les utiliser dans de nouveaux objets plus complexes, de changer un objet si celui-ci ne convient pas. Dans le cas d'une bibliothèque d'optimisation, il s'agit de pouvoir sélectionner avec précision le critère d'arrêt souhaité, le type de recherche de ligne ou de direction, de sélectionner la manière de calculer le gradient, de proposer des objets utilitaires, ...

A.1.3 *Design patterns*

Les *design patterns* ou patrons/motifs de conception sont des recettes permettant de répondre à des questions classiques. L'idée des motifs de conception provient de Christopher Alexander [AIS+78], architecte ayant proposé une liste de motifs architecturaux. L'idée sous-jacente était de ne pas réinventer la roue. Quelques années plus tard, ce principe a été appliqué à l'informatique par le *Gang of Four* [GHJV94].

Voici certains de ces motifs de conception[5] :

– Observeur permet à une série d'objets d'observer les changements sur un même objet,
– Stratégie définit des comportements interchangeables,
– Décorateur ajoute des possibilités au comportement d'un objet,
– Composition permet de composer et d'utiliser une hiérarchie complexe d'objets.

Tous les *design patterns* ne sont pas utiles dans le cadre d'une bibliothèque d'optimisation. Par exemple, le motif Observeur n'est pas utile (ce motif est souvent utilisé en programmation événementielle, comme la programmation d'interfaces graphiques). En revanche, la Stratégie est primordiale. En effet, c'est elle qui est à la base du choix d'un module de recherche de ligne ou de direction. Le Décorateur servira à modifier et à affiner une stratégie (par exemple en permettant d'utiliser la taille du pas précédent comme départ de la recherche de ligne lors de l'itération suivante). Enfin, la Composition permettant de fusionner des comportements, ce motif pourra être utilisé dans la sélection d'un critère.

A.1.4 Principe de séparation

Le principe de séparation [KL04] est généralement antagoniste au principe orienté objet et plus exactement au principe de responsabilité de ce dernier. L'idée sous-jacente est de séparer les données du traitement. Ce principe est clairement explicite dans la STL (*Standard Template Library*, un sous-ensemble de la bibliothèque standard C++) où les algorithmes sur les données (tels que la recherche dans un conteneur ou le tri) sont séparés dans leur grande majorité des conteneurs qui sont chargés des données.

Dans le cas d'une bibliothèque d'optimisation, ce principe de séparation est primordial car les données tournent entre les différents modules principaux. Les données utilisées pour chercher une direction sont utiles aussi pour la recherche du pas. Toutes ces données sont indispensables au critère d'arrêt. C'est pourquoi un dictionnaire contenant l'état courant de l'optimisation est passé en paramètre à chaque module. Ces modules pourront alors récupérer les données utiles et mettre à jour d'autres résultats.

En revanche, cela a des conséquences sur l'utilisation de certains motifs de conception. En effet, l'utilisation d'un décorateur peut être très dangereuse. Si une décoration utilise des données communes et met à jour le dictionnaire d'état et qu'un décorateur suivant utilise ces mêmes données dans un autre but, un problème peut en découler.

Par exemple, chaque module de recherche de direction peut initialiser sa recherche de pas à une valeur donnée dans le dictionnaire. Si une décoration de ce module indique une certaine

[5] voir http ://conception.developpez.com/cours/#dp pour des explications plus détaillées

valeur de départ et qu'un autre décorateur modifie à nouveau cette valeur, le premier décorateur ne sert à rien. Pour corriger ce problème, si le premier décorateur enregistre la donnée dans un nouveau champ du dictionnaire et que les autres décorateurs modifient cette valeur, comment le premier décorateur saura-t-il qu'il est le premier et qu'il doit créer à nouveau cette donnée (sachant qu'elle est dans le dictionnaire) ? Il est en effet impossible de vider le dictionnaire à chaque étape puisque les résultats de l'itération précédente peuvent être utiles (direction précédente de descente, valeurs et position lors de l'itération précédente pour le calcul du critère d'arrêt, ...). C'est aussi là la limite entre la théorie en informatique et l'application pratique des méthodes.

A.2 Application à la création d'un module d'optimisation

Afin d'être souple et réutilisable, une boîte à outils d'optimisation correcte doit présenter chacun des aspects exposés ci-dessus, tests unitaires, orientation objet et principe de séparation. Cette bibliothèque a principalement été crée pour l'optimisation de fonctions convexes, même si diverses extensions peuvent être envisagées pour intégrer l'optimisation convexe avec contraintes.

Outre le module principal d'optimisation avec ses sous-modules, plusieurs modules utilitaires sont fournis. Par exemple, une fonction de coût effectuant la somme du carré de la différence entre des données originelles et une approximation de ces données calculées par une fonction auxiliaire est proposée, y compris une avec un hessien approché.

A.2.1 Le coeur, la classe Optimizer

Une optimisation repose tout d'abord sur une fonction à optimiser et un point de départ. Par la suite, l'optimisation convexe repose globalement sur les trois modules présentés au début de cette annexe : la recherche d'une direction de descente, la recherche d'un pas et un critère d'arrêt. Dans cet optique de séparation des responsabilités (un autre principe de la programmation modulaire), une classe décrira un optimiseur global (il sera responsable de la fonction et du point de départ, mais aussi de l'initialisation du dictionnaire contenant l'état de l'optimiseur) et une classe dérivée représentera l'optimisation convexe elle-même.

La méthode du polytope [NM65], appelée aussi du simplexe, *downhill method* ou Nelder-Mead peut aussi être implémentée sous cette forme, la recherche de la direction étant alors la recherche d'un premier candidat puis la recherche du pas effectue la contraction ou la dilatation selon le coût au point sélectionné.

Outre l'optimisation simple décrite précédemment, une autre classe d'optimisation a été écrite. Celle-ci permet d'effectuer une opération avant chaque itération et une autre après cette itération.

Enfin, après chaque itération, il est possible de sauvegarder l'état de l'optimiseur, afin de découvrir les raisons d'éventuels échecs d'optimisation.

A.2.2 Les différents modules disponibles

Le développement a principalement été orienté vers l'optimisation de fonctions convexes sans contraines.

Les critères d'arrêt implémentés sont les suivants :
- atteinte de l'itération limite,
- variation absolue du coût inférieure à une limite,
- variation relative du coût inférieure à une limite,
- variation absolue du gradient du coût inférieure à une limite,
- variation relative du gradient du coût inférieure à une limite.

Ces différents éléments sont composables entre eux, soit avec un ET logique ou avec un OU logique. D'autres critères plus complexes, dépendant d'autres variables d'état de l'optimiseur peuvent être conçus et composés avec les critères présentés.

La sélection d'une direction de descente est un problème qui reste très ouvert dépendant de la fonction recherchée. Voici quelques-unes des directions implémentées :
- direction opposée au gradient,
- direction du gradient conjugué [HS52, FR64, PR69],
- direction de Newton (opposée au hessien multiplié par le gradient),
- direction de Newton modifiée [GP67]
- direction de quasi-Newton [Dav59, Bro70, Fle70, Gol70, Sha70],

Les directions par gradient conjugué sont chacune une composition entre la direction précédente et la direction donnée par le gradient courant. Le poids de chacune donne les différentes techniques existantes. C'est pourquoi toutes ces techniques sont basées sur le même modèle, seul le poids étant modulé. De plus, des décorateurs permettent de redémarrer le gradient conjugué, soit périodiquement, soit lorsque deux gradients consécutifs ne sont plus orthogonaux.

Enfin, plusieurs recherches de lignes exactes et inexactes ont été implémentées comme :
- section utilisant le nombre d'or ou la série de Fibonacci [SY06],
- interpolation quadratique ou cubique [SY06],
- recherche utilisant la règle d'Armijo (aussi appelée *Backtracking*) [Arm66],
- recherche utilisant les règles de Wolfe-Powell normales ou leur version forte [Wol69, Wol71],
- recherche utilisant les règles de Goldstein [Gol65],

Des recherches de lignes simples ont aussi été conçues, telles que les recherches amorties (recherche d'un pas par dichotomie) ou des pas dépendant du nombre d'itérations effectuées. Des décorateurs ont aussi été mis en place, permettant de commencer la recherche du pas à partir du pas précédent.

A.2.3 Différentiation automatique

Dans ce nombreux cas, un gradient analytique et à plus forte raison un hessien analytique ne sont pas disponibles. Il est alors nécessaire de faire appel à une approximation numérique de ces données. Dans les boîtes à outils usuelles, c'est l'optimiseur qui se charge de ces opérations,

lorsqu'une fonction gradient ou hessien n'est pas fournie.

En raison du principe de responsabilité et par souci de modularité, l'approche proposée par cette boîte à outils est différente. En effet, la décision d'utiliser un gradient analytique ou numérique revient à l'utilisateur de la fonction. De plus, la sélection du pas de différentiation est spécifique à une fonction et non à une méthode d'optimisation.

Les deux techniques de différentiation usuelles sont implémentées, *Forward Difference* et *Centered Difference* [NW06]. Il s'agit de deux classes qui seront utilisées comme classe de base de la fonction à optimiser. Si un gradient analytique existe, son implémentation sera alors directement utilisée lors d'un calcul de gradient ou lors du calcul du hessien numérique. De même, le hessien pourra être implémenté de manière analytique.

A.2.4 Estimation

Lorsqu'un modèle de fonction doit être adapté à des échantillons (par exemple, estimation des paramètres (a, b) d'une fonction linéaire $f(c) = ax + b$ à partir de couples (x, y)), une fonction d'optimisation générique peut être créée. En effet, ce genre d'optimisation est souvent effectué en minimisant la somme des carrés des différences $(y - f(x))$. En créant un objet encapsulant $f()$ et calculant ce coût, il est possible d'utiliser l'ensemble des outils d'optimisation présenté en amont.

Voici la forme générale du problème à résoudre selon les paramètres $\boldsymbol{\theta}$:

$$min_{\boldsymbol{\theta}\in\mathbb{R}} h(\boldsymbol{\theta}) = \sum c(y - f_{\boldsymbol{\theta}}(x)) \tag{A.1}$$

Deux coûts usuels $c()$ ont été implémentés. Le premier est le coût quadratique utilisé traditionnellement ($c(x) = \frac{1}{2}\|x\|^2$).

Un autre coût fréquemment utilisé est une approximation de celui-ci au deuxième ordre, appelée Gauss-Newton. Ce coût est une approximation au deuxième ordre du coût quadratique. En effet, l'approximation au deuxième ordre $q^k(\boldsymbol{\theta})$ de A.1 donne :

$$q^k(\boldsymbol{\theta}) = h(\boldsymbol{\theta}_k) + g(\boldsymbol{\theta})(\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T \boldsymbol{H}(\boldsymbol{\theta})(\boldsymbol{\theta} - \boldsymbol{\theta}_k) \tag{A.2}$$

Avec :

$$\boldsymbol{r}(\boldsymbol{\theta}) = [y_1 - f(x_1), y_2 - f(x_2) \ldots, y_n - f(x_n)] \tag{A.3}$$

$$\boldsymbol{J}(\boldsymbol{\theta}) = \begin{bmatrix} \frac{\delta r_1}{\delta \theta_1}(\boldsymbol{\theta}) & \frac{\delta r_1}{\delta \theta_2}(\boldsymbol{\theta}) & \ldots & \frac{\delta r_1}{\delta \theta_m}(\boldsymbol{\theta}) \\ \frac{\delta r_2}{\delta \theta_1}(\boldsymbol{\theta}) & \frac{\delta r_2}{\delta \theta_2}(\boldsymbol{\theta}) & \ldots & \frac{\delta r_2}{\delta \theta_m}(\boldsymbol{\theta}) \\ \ldots & \ldots & \ldots & \ldots \\ \frac{\delta r_n}{\delta \theta_1}(\boldsymbol{\theta}) & \frac{\delta r_n}{\delta \theta_2}(\boldsymbol{\theta}) & \ldots & \frac{\delta r_n}{\delta \theta_m}(\boldsymbol{\theta}) \end{bmatrix} \tag{A.4}$$

$$\boldsymbol{g}(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{r}(\boldsymbol{\theta}) \tag{A.5}$$

$$\boldsymbol{S}(\boldsymbol{\theta}) = \sum_{i=1} n r_i(\boldsymbol{\theta}) \nabla^2 r_i(\boldsymbol{\theta}) \tag{A.6}$$

$$\boldsymbol{H}(\boldsymbol{\theta}) = \boldsymbol{J}(\boldsymbol{\theta})^T \boldsymbol{J}(\boldsymbol{\theta}) + \boldsymbol{S}(\boldsymbol{\theta}) \tag{A.7}$$

L'optimum de cette fonction se trouve pour :

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - (\boldsymbol{J}(\boldsymbol{\theta}_k)^T \boldsymbol{J}(\boldsymbol{\theta}_k) + \boldsymbol{S}(\boldsymbol{\theta}_k))^{-1} \boldsymbol{J}(\boldsymbol{\theta}_k) \boldsymbol{r}\boldsymbol{\theta}_k. \tag{A.8}$$

Si on peut considérer $\boldsymbol{S}(\boldsymbol{\theta}_k) \simeq 0$, on obtient l'approximation de Gauss-Newton.

A partir de cette approximation, l'algorithme Levenberg-Marquardt [Lev44, Mar63] peut être construit avec la boîte à outils présentée.

A.2.5 Limites du modèle

Le principe de séparation a une influence lourde sur les possibilités d'évolution de la boîte à outils proposée. En effet, il n'est pas possible de chaîner plusieurs décorateurs sur la recherche de direction ou sur celle de ligne. Cet inconvénient peut être résolu à condition de complexifier l'état de l'optimiseur. Par exemple, dès qu'un décorateur travaille sur des données, il sauvegarderait celles-ci dans des listes associées. Le problème survient car il est nécessaire d'avoir une liste par élément modifié, et que tous les paramètres possibles ne sont pas connus *a priori*. Chaque décorateur doit pouvoir créer cette liste, la vider si besoin est, ce qui complexifie la création d'un nouveau décorateur.

Une autre limitation du modèle est que plusieurs modules incompatibles peuvent être utilisés ensemble, l'erreur ne se déclenchant que lors de l'optimisation, et non lors de la construction de l'optimiseur. Par exemple, il est possible d'utiliser d'utiliser un optimiseur de Newton avec une fonction qui ne peut pas calculer son hessien de manière analytique ou numérique. Un mécanisme de vérification statique peut être ajouté, à condition de spécifier pour chaque module ses besoins et ce qu'il propose.

Enfin, le nombre d'appels à la fonction (elle-même, son gradient ou son hessien) n'est pas compté et n'est pas non plus optimisé. Cette optimisation en fonction du nombre d'appels peut être ajoutée à l'état de l'optimiseur à peu de frais. En revanche, le fait que les appels eux-mêmes ne soient pas optimisés (donc qu'il peut exister plusieurs appels à la fonction, au gradient ou au hessien avec les mêmes paramètres) peut être critique dans certains cas. Une amélioration peut consister à stocker les derniers appels dans la variable d'état de l'optimiseur, mais le coût en temps gagén pour certaines optimisations difficiles peut être perdu pour d'autres optimisations rapides.

A.3 Conclusion

La boîte à outils proposée permet de faire face à de nombreux problèmes d'optimisation de fonctions convexes. Sa modularité permet une adaptation rapide à toute forme de fonctions et d'implémenter une grande partie des algorithmes usuels d'optimisation sans contrainte.

Malgré tout, elle présente des inconvénients et nécessite des améliorations, comme l'inclusion de contraintes lors des optimisations.

Annexe B

Divers

Sommaire

B.1 Fonction de coût associée à Isomap 93
B.2 Équivalence entre *Laplacian Eigenmaps* et *Diffusion Maps* 94
B.3 Paramètres de Haralick 95
B.4 Les Séparateurs à Vaste Marge (SVMs) 95
B.4.1 Position du problème 95
B.4.2 Extensions 96
B.4.3 Estimation de probabilité d'appartenance 97

B.1 Fonction de coût associée à Isomap

Isomap [TdSL00] est une technique analytique pour obtenir un espace réduit à partir d'une matrice de distances. Après avoir calculé la matrice des distances géodésiques, un MDS classique est effectué pour obtenir la solution optimale. La fonction de coût associée n'est jamais calculée explicitement. Il est donc nécessaire de préciser quelle est la fonction de coût minimisée afin de connaître ses caractéristiques.

Soient $\boldsymbol{D_Y}$ la matrice de distances dans l'espace original, $\boldsymbol{D_X}$ celle dans l'espace réduit, $\boldsymbol{H} = \boldsymbol{I} - \frac{1}{n}\mathbf{1}$ la matrice de centrage permettant de calculer une matrice de corrélation à partir d'une matrice de distances [Tor52]. L'objectif d'une analyse en composantes principales (l'outil pour obtenir la solution analytique du MDS) est de minimiser la différence entre les matrices de corrélation de $\boldsymbol{Y}$ et de $\boldsymbol{X}$:

$$f(\boldsymbol{X}) = \|\boldsymbol{Y}\boldsymbol{Y}^T - \boldsymbol{X}\boldsymbol{X}^T\|_F \quad \text{(B.1)}$$
$$= \|\boldsymbol{H}\boldsymbol{D_Y}\boldsymbol{H} - \boldsymbol{H}\boldsymbol{D_X}\boldsymbol{H}\|_F \quad \text{(B.2)}$$

Ici, la matrice $\boldsymbol{Y}$ n'est pas connue, mais sa connaissance explicite n'est pas utile, seule

la matrice des distances est nécessaire. A partir des propriétés de la norme de Frobenius, en effectuant une décomposition en valeurs singulières :

$$\boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}\boldsymbol{U}^T \tag{B.3}$$

et en sélectionnant les plus grandes valeurs propres $\boldsymbol{S}_1$ et les vecteurs propres associés $\boldsymbol{U}_1$ et posant $\boldsymbol{X} = \boldsymbol{U}_1\boldsymbol{S}_1$, la fonction f($\boldsymbol{X}$) est minimisée.

B.2 Équivalence entre *Laplacian Eigenmaps* et *Diffusion Maps*

Les *Laplacian Eigenmaps* [BN03] et les *Diffusion Maps* [CL06] sont en réalité le même algorithme. Les premiers consistent à calculer une corrélation à l'aide d'un noyau gaussien entre un point et ses proches voisins tandis que les derniers proposent d'utiliser une matrice de probabilités de transition (ou de Markov) basée sur cette même corrélation entre tous les points. Dans un second temps, une diagonalisation est effectuée et c'est cette diagonalisation qui est différente dans la littérature.

En considérant la matrice $\boldsymbol{S}$ de ces corrélations, la matrice diagonale $\boldsymbol{D}$ contenant la somme sur chaque colonne ou ligne de $\boldsymbol{S}$:

$$\boldsymbol{L}_1 = \boldsymbol{D} - \boldsymbol{S} \tag{B.4}$$
$$\boldsymbol{L}_2 = \boldsymbol{D}^{-1/2}\boldsymbol{S}\boldsymbol{D}^{-1/2} \tag{B.5}$$

Dans le cas des *Laplacian Eigenmaps*, ce sont les plus petites valeurs propres (sauf la plus petite qui est nulle) de la décomposition en valeurs singulières généralisée $\boldsymbol{L}_2\boldsymbol{f} = \lambda\boldsymbol{D}\boldsymbol{f}$ qui permettent de résoudre le problème. Dans le cas des *Diffusion maps*, il s'agit des plus grandes valeurs propres (sauf la première) de la décomposition en valeurs singulières $\boldsymbol{L}_1\boldsymbol{f} = \lambda\boldsymbol{f}$. En réalité, ces deux opérations sont identiques.

En effet, si λ est valeur propre de $\boldsymbol{L}_1$ pour la fonction propre $\boldsymbol{f}$:

$$\begin{aligned}
\boldsymbol{L}_1\boldsymbol{f} &= \lambda\boldsymbol{D}\boldsymbol{f} && \text{(B.6)}\\
(\boldsymbol{D}-\boldsymbol{S})\boldsymbol{f} &= \lambda\boldsymbol{D}\boldsymbol{f} &&\\
\boldsymbol{S}\boldsymbol{f} &= (1-\lambda)\boldsymbol{D}\boldsymbol{f} &&\\
\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{f} &= (1-\lambda)\boldsymbol{D}^{1/2}\boldsymbol{f} &&\\
\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}(\boldsymbol{D}^{1/2}\boldsymbol{f}) &= (1-\lambda)(\boldsymbol{D}^{1/2}\boldsymbol{f}) &&\\
\boldsymbol{L}_2(\boldsymbol{D}^{1/2}\boldsymbol{f}) &= (1-\lambda)(\boldsymbol{D}^{1/2}\boldsymbol{f}) && \text{(B.7)}
\end{aligned}$$

$(1-\lambda)$ est donc valeur propre de $\boldsymbol{L}_1$ pour la fonction propre $\boldsymbol{D}^{1/2}\boldsymbol{f}$. Un avantage du calcul de $\boldsymbol{L}_1$ est que les routines de décomposition en valeurs singulières sont plus fréquemment disponibles pour $\boldsymbol{L}_1$ que pour $\boldsymbol{L}_2$.

B.3 Paramètres de Haralick

A partir d'une matrice de cooccurrence $\boldsymbol{M}$ de taille (n, n), Haralick [HSD73] a proposé 14 paramètres. Par la suite, Conners et Harlow [CH80] ont démontré que ces 14 paramètres étaient corrélés et que seuls 5 coefficients spécifiques étaient suffisants. Ces paramètres sont calculés à partir de la matrice $\boldsymbol{M}$ ou de sa version normalisée $\tilde{\boldsymbol{M}} = \boldsymbol{M} / \sum_{i,j} m_{i,j}$.

– Homogénéité, aussi appelé moment angulaire d'ordre 2 :

$$E = \sum_{i,j} \tilde{m}_{i,j}^2$$

– Contraste :

$$C = \sum_{k} k^2 \sum_{|i-j|=k} \tilde{m}_{i,j}$$

– Corrélation :

$$Cor = \frac{n \sum_{i,j}(ijm_{i,j}) - (\sum_{i,j} im_{i,j})(\sum_{i,j} jm_{i,j})}{\sqrt{(n \sum_{i,j} i^2 m_{i,j}) - (\sum_{i,j} im_{i,j})^2}\sqrt{(n \sum_{i,j} j^2 m_{i,j}) - (\sum_{i} \sum_{j} jm_{i,j})^2}}$$

– Entropie :

$$H = \sum_{i,j} \tilde{m}_{i,j} log(\tilde{m}_{i,j})$$

– Homogénéité locale :

$$LH = \sum_{i,j} \frac{\tilde{m}_{i,j}}{1 + (i-j)^2}$$

B.4 Les Séparateurs à Vaste Marge (SVMs)

L'objectif des SVMs [Vap98] est de maximiser la marge entre la frontière de décision et les points d'apprentissage (fig. 1.7). Cela permet d'obtenir une robustesse lors de la généralisation (la droite H_2 ne permet pas cela car elle est très proche de l'une des classes, donc le taux de mauvaises classifications peut être plus important).

B.4.1 Position du problème

Soient $\{(\boldsymbol{x}_1, c_1), (\boldsymbol{x}_2, c_2), ...\}$, $c_i = \pm 1$ selon la classe à laquelle le point appartient. Le problème est alors de chercher $\boldsymbol{w}$ tel que $c_i(\boldsymbol{w}.\boldsymbol{x}_i - b) \geq 1$ pour $1 \leq i \leq I$. La contrainte supplémentaire du problème est de maximiser la marge entre les deux classes et donc de minimiser $\|\boldsymbol{w}\|$ (fig. B.1). Le problème s'exprime alors sous la forme d'un problème quadratique :

$$\begin{cases} \text{minimiser}_{\{\boldsymbol{w},b\}} \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{avec } c_i(\boldsymbol{w}.\boldsymbol{x}_i - \boldsymbol{b}) \geq 1, 1 \leq i \leq I \end{cases} \tag{B.8}$$

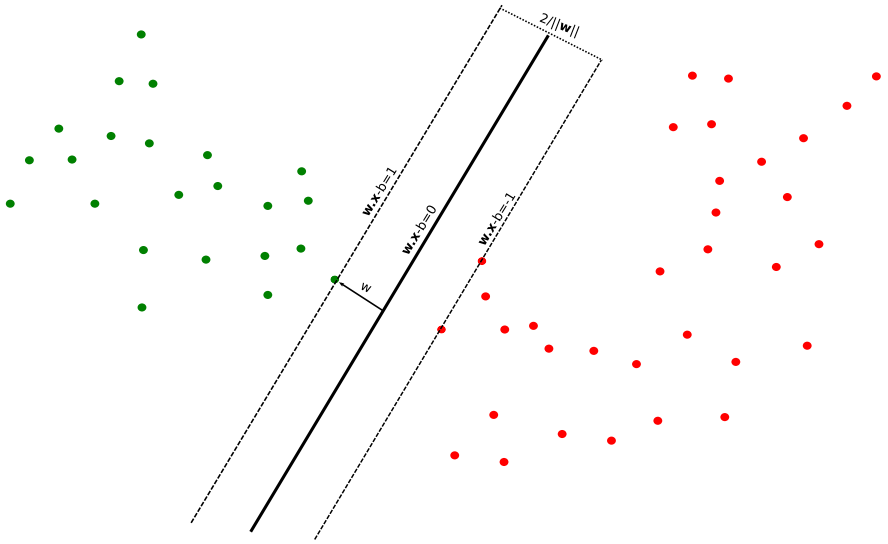


FIG. B.1: Optimisation de $\boldsymbol{w}$ pour maximiser la marge entre les deux classes

Ce problème peut être transformé en un autre, plus simple à résoudre :

$$\begin{cases} \text{maximiser } \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j \boldsymbol{x}_i^T \boldsymbol{x}_j \\ \text{avec } \sum_i \alpha_i c_i = 0 \text{ et } \alpha_i \geq 0. \end{cases} \tag{B.9}$$

avec $\boldsymbol{w} = \sum_i \alpha_i c_i \boldsymbol{x}_i$.

La résolution de ce problème est donc la recherche des α_i, un α_i non nul indiquant que $\boldsymbol{x}_i$ est un vecteur support. Ce problème quadratique peut s'optimiser efficacement à l'aide d'un algorithme *Sequential Minimal Optimization* [FCL05] (implémenté dans la bibliothèque utilisée [CL01]). Les points d'apprentissage pour lesquels α_i est nul ne seront pas utilisés lors de la classification. Ce comportement est la raison de la robustesse des SVMs à un entraînement, car un nombre important de points d'apprentissage ne modifie pas les vecteurs support sélectionnés.

B.4.2 Extensions

L'intérêt des SVM se situe principalement dans le domaine de la classification non linéaire. En effet, tout comme pour l'ACP à noyau, il est possible de remplacer le produit scalaire $\boldsymbol{x}_i.\boldsymbol{y}$ par un noyau $k(\boldsymbol{x}_i, \boldsymbol{y})$ lors du calcul de $\boldsymbol{w}.\boldsymbol{y} - b$ pour un nouveau point $\boldsymbol{y}$. Dans ce cas, la séparation linéaire s'effectue dans un espace de dimension potentiellement infini.

Dans un deuxième temps, il n'est pas toujours possible de séparer strictement deux classes. En effet, il se peut qu'un certain nombre d'éléments d'une classe se retrouvent entourés des éléments de l'autre classe et inversement. Afin d'éviter un surapprentissage, il est possible d'as-

souplir le problème B.8 en introduisant des variables dites ressort ξ_i :

$$\begin{cases} \text{minimiser}_{\{\boldsymbol{w},b\}} \ \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \\ \text{avec } c_i(\boldsymbol{w}.\boldsymbol{x}_i - \boldsymbol{b}) \geq 1 - \xi_i, 1 \leq i \leq I \end{cases} \tag{B.10}$$

avec C une variable arbitraire contrôlant le compromis entre largeur de marge et le nombre d'erreur de classement.

B.4.3 Estimation de probabilité d'appartenance

Un élément intéressant de ce classifieur est qu'il est possible de probabiliser l'appartenance à une classe et donc de se déplacer d'une classe vers une autre. Pour cela, on peut utiliser une fonction coût qui sera la différence entre la probabilité d'appartenance à une classe et à une autre (voir fig. 5.1d pour une figure utilisant une fonction similaire). A la fin du déplacement, on obtient alors une différence entre le point et le point le plus proche de l'autre classe [GGSK05] (voir aussi 5.1.2).

S'il existe K classes, l'objectif est d'estimer la probabilité d'appartenance à la classe c pour $\boldsymbol{x}$:

$$p_k = p(c = k|\boldsymbol{x}), k = 1, \ldots, K$$

Dans un premier temps, les probabilités de classe appairées entre la classe i et la classe j doivent être calculées :

$$r_{kl}(\boldsymbol{x}) = p(c = k|c = k \text{ ou } l, \boldsymbol{x})$$

Ceci est effectué par une amélioration [LLW07] de l'approximation de la probabilité *a posteriori* proposée par [Pla00] :

$$r_{ij} \approx \frac{A}{1 + e^{Af(\boldsymbol{x})+B}}$$

avec $f(\boldsymbol{x})$ la fonction de décision entre la classe k et la classe l pour le point $\boldsymbol{x}$. Les valeurs A et B sont les valeurs optimales du problème suivant :

$$\begin{cases} \text{minimiser}_{z=\{A,B\}} \ F(z) = -\sum_i (t_i \log(p_i) + (1 - t_i)\log(1 - p_i)) \\ \text{avec } p_k = r_{kl}(\boldsymbol{x}_i) \text{ et } t_i = \begin{cases} \frac{N_k + 1}{N_k + 2} \text{ si } c(\boldsymbol{x}_k) = k \\ \frac{1}{N_l + 2} \text{ si } c(\boldsymbol{x}_k) = l \end{cases} \end{cases} \tag{B.11}$$

où N_k est le nombre de points assignés à la classe l et N_k le nombre de points assignés à la classe l.

Une fois ces termes estimés, il est possible de calculer les probabilités p_k pour un échantillon $\boldsymbol{x}$. La seconde approche exposée dans [WLW04] a été sélectionnée. Le problème à résoudre s'écrit sous la forme :

$$\begin{cases} \text{minimiser}_{\{p_1,\ldots,p_K\}} \ \sum_k \sum_{l \neq k} (r_{kl}p_k - r_{kl}p_l)^2 \\ \text{avec } \sum_k p_k = 1, p_k \geq 0, \forall k \end{cases} \tag{B.12}$$

Annexe C

Publications

Revues internationales

[1] M. BRUCHER, Ch. HEINRICH, F. HEITZ et J.-P. ARMSPACH : A metric multidimensional scaling-based nonlinear manifold learning approach for unsupervised data reduction. *EURASIP Journal on Advances in Signal Processing*, 2008.

Conférences internationales avec comité de lecture et actes

[2] M. BRUCHER, C. HEINRICH, F. HEITZ et J.-P. ARMSPACH : Unsupervised nonlinear manifold learning. *In International Conference on Image Processing ICIP'07, San Antonio, USA, 16-19 September 2007.*, 2007.

Autres communications

[3] M. BRUCHER, C. HEINRICH, F. HEITZ et J.-P. ARMSPACH : Réduction de données et apprentissage non supervisé de variétés pour le traitement d'images. *In ORASIS'07, June 4-8, Obernai, France*, 2007.

Ouvrages

[4] M. BRUCHER : *Python : les fondamentaux du langage - la programmation pour les scientifiques*. Ressources Informatiques. Editions ENI, january 2008.

Bibliographie

[ADBB04] V. ARVIS, C. DEBAIN, M. BERDUCAT et A. BENASSI : Generalization of the co-occurrence matrix for colour images : Application to colour texture classification. *Image Analysis & Stereology*, 23:63–72, 2004.

[AIS+78] C. ALEXANDER, S. ISHIKAWA, M. SILVERSTEIN, M. JACOBSON, I. FIKSDAHL-KING et S. ANGEL : *A pattern language : Towns, buildings, construction*. Oxford University Press, août 1978.

[Aka74] H. AKAIKE : A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[Arm66] L. ARMIJO : Minimization of functions having Lipschitz continuous partial derivatives. *Pacific Journal of Mathematics*, 16:1–3, 1966.

[AX02] D.K. AGRAFIOTIS et H. XU : A self-organizing principle for learning non linear manifolds. *Proceedings of the national academy of science of the United States of America*, 99(25):15869–15872, décembre 2002.

[Bak77] C. BAKER : *The numerical treatment of integral equations*. Clarendon Press, Oxford, 1977.

[BBC04] N. BANSAL, A. BLUM et S. CHAWLA : Correlation clustering. *Machine learning*, 56:89–113, 2004.

[BBMW04] J.-L. BÉNARD, L. BOSSAVIT, R. MÉDINA et D. WILLIAMS : *Gestion de projet - eXtreme Programming*. Eyrolles, 3 édition, décembre 2004.

[Bec02] K. BECK : *Test-Driven Development*. Pearson Education, novembre 2002.

[BGK95] C. BRECHBÜHLER, G. GERIG et O. KÜBLER : Parametrization of closed surfaces for 3D shape description. *Computer vision and image understanding*, 61:154–170, 1995.

[Bis06] C.M. BISHOP : *Pattern recognition and machine learning*. Springer, 2006.

[BM05] Y. BENGIO et M. MONPERRUS : Non-local manifold tangent learning. *In* L. K. SAUL, Y. WEISS et L. BOTTOU, éditeurs : *Advances in neural information processing systems*, volume 17, pages 129–136. MIT Press, 2005.

[BN78] H. BLUM et R. NAGEL : Shape description using weighted symmetric axis features. *Pattern Recognition*, 10(3):167–180, 1978.

[BN03] M. BELKIN et P. NIYOGI : Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 13:1373–1397, 2003.

[Boo89] F.L. BOOKSTEIN : Principal warps : Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

[Boo96] F.L. BOOKSTEIN : Landmark methods for forms without landmarks : morphometrics of group differences in outline shape. *Medical Image Analysis*, 1:225–243, 1996.

[Bor05] I. BORG : *Modern Multidimensional Scaling*. Springer, 2ème édition, août 2005.

[BPV03] Y. BENGIO, J.-F. PAIEMENT et P. VINCENT : Out-of-sample extensions for LLE, Isomap, MDS, Laplacian eigenmaps and spectral clustering. Rapport technique 1238, Département d'informatique et recherche opérationnelle, université de Montréal, juillet 2003.

[Bra03] M. BRAND : Charting a manifold. Rapport technique TR-2003-13, MERL, mars 2003.

[Bro70] C.G. BROYDEN : The convergence of a class double-rank minimization algorithms. *Journal of the Institute of Mathematics and its Applications*, 6:76–90, 1970.

[BS98a] C.M. BISHOP et M. SVENSÉN : GTM : The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998.

[BS98b] J. BRUSKE et G. SOMMER : Intrinsic dimensionality estimation with optimally topology preserving maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):572–575, mai 1998.

[BST+] M. BALASUBRAMANIAN, E.L. SCHWARTZ, J.B. TENENBAUM, V. de SILVA et J.C. LANGFORD : The isomap algorithm and topological stability. *Science*, 295: 7a, janvier.

[Cat66] R. CATTELL : The scree test for the number of factors. *Multivariate Behaviorial Research*, 1:245–276, 1966.

[CC07] H. CHOI et S. CHOI : Robust kernel isomap. *Pattern Recognition*, 40:853–862, 2007.

[CH80] R.W. CONNERS et C.A. HARLOW : A theoretical comparison of texture algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(3):204–222, 1980.

[CH04] J.L. COSTA et A.O. HERO : Geodesic entropy graphs for dimension and entropy estimation in manifold learning. *IEEE Transactions on Signal Processing*, 52(8): 2210–2221, août 2004.

[Chu97] F. CHUNG : *Spectral graph theory*. Numéro 92 de Regional Conference Series in Mathematics. American Mathematical Society, 1997.

[CL01] C.C. CHANG et C.-J. LIN : *LIBSVM : a library for support vector machines*, 2001. Code disponible à l'adresse http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[CL06] R. COIFMAN et S. LAFON : Diffusion maps. *Applied and Computational Harmonic Analysis*, 21:5–30, 2006.

[Cle79] W.S. CLEVELAND : Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–835, 1979.

[CLR01] T. H. CORMEN, C. E. LEISERSON et R. L. RIVEST : *Introduction to algorithms*. MIT Press, 2001.

[CM02] D. COMANICIU et P. MEER : Mean shift : A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, mai 2002.

[CV02] F. CAMASTRA et A. VINCIARELLI : Estimating the intrinsic dimension of data with a fractal-based method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(10):1404–1407, octobre 2002.

[CY06] H. CHANG et D.-Y. YEUNG : Robust locally linear embedding. *Pattern recognition*, 39:1053–1065, 2006.

[Dav59] W.C. DAVIDON : Variable metric method for minimization. Rapport technique ANL-5990, Argonne National Laboratory, 1959.

[DG03] D. L. DONOHO et C. GRIMES : Hessian eigenmaps : locally linear embedding techniques for high-dimensional data. *Proceedings of the national academy of sciences of the United States of America*, 100(10):5591–5596, mai 2003.

[DH97] P. DEMARTINES et J. HÉRAULT : Curvilinear component analysis : a self-organizing neural network for non linear mapping of data sets. *IEEE Transactions on Neural Networks*, 8(1):148–154, janvier 1997.

[DHS00] R.O. DUDA, P.E. HART et D.G. STORK : *Pattern Classification*. Wiley, novembre 2000.

[DL97] M. DASH et H. LIU : Feature selection for classification. *Intelligent data analysis*, 1:131–156, 1997.

[DLR77] A.P. DEMPSTER, N.M. LAIRD et D.B. RUBIN : Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, B*, 39(8):1–38, 1977.

[DP97] P. DOMINGOS et M. PAZZANI : On the optimality of the simple Bayesian classifier under zero-one loss. *Machine learning*, 29:103–137, 1997.

[dST02] V. de SILVA et J. B. TENENBAUM : Global versus local methods in non linear dimensionality reduction. *In Advances in neural information processing systems*, volume 15, pages 721–728. MIT Press, 2002.

[EL04] A. ELGAMMAL et C.-S. LEE : Inferring 3D body pose from silhouettes using activity manifold learning. *In CVPR*, 2004.

[FCL05] R.-E. FAN, P.-H CHEN et C.-J. LIN : Working set selection using second order information for training SVM. *Journal of Machine Learning Research*, 6:1889–1918, 2005.

[Fle70] R. FLETCHER : A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

[Flo62] R.W. FLOYD : Algorithm 97 : Shortest Path. *Communications of the ACM*, 5:345, juin 1962.

[FLPJ04] P.T. FLETCHER, C. LU, S.M. PIZER et S. JOSHI : Principal geodesic analysis for the study of nonlinear statistics of shape. *IEEE Transactions on Medical Imaging*, 23(8):995–1005, août 2004.

[FO71] K. FUKUNAGA et D. OLSEN : An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 20(2):176–183, 1971.

[FR64] R. FLETCHER et C.M. REEVES : Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[FS81] J.H. FRIEDMAN et W. STUETZLE : Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–8238, décembre 1981.

[GGSK05] P. GOLLAND, W.E.L. GRIMSON, M.E. SHENTON et R. KIKINIS : Detection and analysis of statistical differences in anatomical shape. *Medical image analysis*, 9:69–86, février 2005.

[GH96] Z. GHAHRAMANI et G. HINTON : The EM algorithm for mixture of factor analyzers. Rapport technique CRG-TR-96-1, Université de Toronto, 1996.

[GHJV94] E. GAMMA, R. HELM, R. JOHNSON et J. VLISSIDES : *Design patterns : Elements of reusable object-oriented software*. Addison-Wesley, novembre 1994.

[Gol65] A.A. GOLDSTEIN : On steepest descent. *SIAM Journal of Control*, 3:147–151, 1965.

[Gol70] D. GOLDFARB : A family of variable metric methods derived by variation mean. *Mathematics of Computation*, 23:23–26, 1970.

[GP67] A.A. GOLDSTEIN et J.F. PRICE : An effective algorithm for minimization. *Numerical Mathematics*, 10:184–189, 1967.

[GP83] P. GRASSBERGER et I. PROCACCIA : Measuring the strangeness of strange attractors. *Physica D Nonlinear phenomena*, 9:189–208, 1983.

[Gre69] R.E. GREENE : Isometric embeddings. *Bulletin of the American Mathematical Society*, 75(6):1308–1310, 1969.

[Har79] R.R. HARALICK : Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67:786–804, 1979.

[HH07] R. HARALICK et R. HARPAZ : Linear manifold clustering in high dimensional spaces by stochastic search. *Pattern recognition*, 40:2672–2684, 2007.

[HLMS03] J. HAM, D.D. LEE, S. MIKA et B. SCHÖLKOPF : A kernel view of the dimensionality reduction of manifolds. Rapport technique 110, Max-Planck-Institut für biologische Kybernetik, juillet 2003.

[HPN02] B.G. HASKELL, A. PURI et A.N. NETRAVALI : *Digital video : An introduction to MPEG-2*, chapitre 6, pages 110–145. Springer, 2002.

[HR02] G. HINTON et S. ROWEIS : Stochastic Neighbor Embedding. *In Advances in neural information processing systems*, volume 15, pages 857–864. MIT Press, 2002.

[HS52] M.R. HESTENES et E. STIEFEL : Method of conjugate gradient for solving linear system. *Journal of Research of the National Bureau of Standards*, 49:409–436, 1952.

[HS89] T. HASTIE et W. STUETZLE : Principal curves. *Journal of the American Statistical Association*, 84(406):502–516, 1989.

[HSD73] R.M. HARALICK, K. SHANMUGAM et I. DINSTEIN : Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, 3(6):610–621, 1973.

[JJ95] M. JIANCHANG et A.K. JAIN : Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6(2):296–317, mars 1995.

[Jol02] I.T. JOLLIFFE : *Principal Components Analysis*. Springer, 2ème édition, 2002.

[Jul62] B. JULESZ : Visual pattern discrimination. *IEEE Transactions on Information Theory*, 8(2):84–92, février 1962.

[KK02] B. KÉGL et A. KRZYZAK : Piecewise linear skeletonization using principal curves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1): 59–74, janvier 2002.

[KL04] Y. KAMBAYASHI et H.F. LEDGARD : The separation principle : A programming paradigm. *IEEE Software*, 21(2):78–87, mars 2004.

[Koh01] T. KOHONEN : *Self-organizing maps*. Springer, 2001.

[Kra91] M.A. KRAMER : Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[Laf96] J. LAFONTAINE : *Introduction aux variétés différentielles*. EDP Sciences, 1996.

[Lev44] K. LEVENBERG : A method for the solution of certain nonlinear problems in least squares. *Quaterly of Applied Mathematics*, 2:164–166, 1944.

[LL06] S. LAFON et A.B. LEE : Diffusion maps and coarse-graining : A unified framework for dimensionality reduction, graph partitioning, and data set parametrization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1393–1403, septembre 2006.

[LLW07] H.-T. LIN, C.-J. LIN et R.C. WENG : A note on Platt's probabilistic outputs for support vector machines. *Machine Learning*, 68(3):267–276, 2007.

[LT94] M. LEBLANC et R. TIBSHIRANI : Adaptive principal surfaces. *Journal of the American Statistical Association*, 89(425):53–64, mars 1994.

[LZ08] T. LIN et H. ZHA : Riemannian manifold learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(5):796–809, mai 2008.

[Mar63] D.W. MARQUARDT : An algorithm for least-squares estimation of nonlinear inequalities. *SIAM Journal of Applied Mathematics*, 11:431–441, 1963.

[MB88] G.J. MACLACHLAN et K.E. BASHFORD : *Mixture Models : Inference and Applications to Clustering*. NY : Marcel Dekker, 1988.

[Mer09] J. MERCER : Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transaction of the Royal Society London*, 1909.

[MLX+08] D. MENG, Y. LEUNG, Z. XU, T. FUNG et Q. ZHANG : Improving geodesic distance estimation based on locally linear assumption. *Pattern recognition letters*, 29:862–870, 2008.

[MPR+03] J.-F. MANGIN, F. POUPON, D. RIVIÉRE, A. CACHIA, D.L. COLLINS, A.C. EVANS et J. RÉGIS : 3D moment invariant based morphometry. *In MICCAI*, 2003.

[NA07] J. NILSSON et F. ANDERSSON : A circuit framework fr robust manifold learning. *Neurocomputing*, 71:323–332, 2007.

[NJW01] A.Y. NG, M.I. JORDAN et Y. WEISS : On spectral clustering : Analysis and an algorithm. *In Advances in neural information processing systems*, volume 14. MIT Press, 2001.

[NM65] J.A. NELDER et R. MEAD : A simplex method for function minimization. *Computer Journal*, 7:308–313, mars 1965.

[NNM96] S.A. NENE, S.K. NAYAR et H. MURASE : Columbia object image library (coil-20). Rapport technique CUCS-005-96, Columbia University Computer Science, Feb 1996.

[NW06] J. NOCEDAL et S.J. WRIGHT : *Numerical optimization*. Springer, 2ème édition, 2006.

[OPM+02] T. OJALA, M. PIETIKÄINEN, T. MÄENPÄÄ, J. VIERTOLA, J. KILLÖNEN et S. HUOVINEN : Outex - new framework for empirical evaluation of texture analysis algorithms. *In 16th International Conference on Pattern Recognition*, volume 1, pages 701–706, 2002.

[PBJD79] K. PETTIS, T. BAILEY, T. JAIN et R. DUBES : An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):25–37, 1979.

[Pen06] X. PENNEC : *Statistical computation on manifolds for computational anatomy*. Mémoire d'H.D.R., Université de Nice Sophia-Antipolis, 18 décembre 2006.

[PG97] T. POGGIO et F. GIROSI : Network for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1997.

[Pla00] J. PLATT : Probabilistic outpus for support vector machines and comparison to regularized likelihood methods. *In* A. SMOLA, P. BARTLETT, B. SCHÖLKOPF et D. SCHUURMANS, éditeurs : *Advances in large margin classifiers*, pages 61–74. Cambridge, MA, 2000.

[PR69] E. POLAK et G. RIBIÈRE : Note sur la convergence de méthodes de directions conjugués. *Revue Française d'Informatique et de Recherche Opérationnelle*, 16: 35–43, 1969.

[PS98] P. PUCAR et J. SJOBERG : On the hinge-finding algorithm for hinging hyperplanes. *IEEE Transactions on Information Theory*, 44(3):1310–1319, mai 1998.

[Rao96] S.S. RAO : *Engineering optimization*. Wiley, 1996.

[RS00] S.T. ROWEIS et L.K. SAUL : Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, décembre 2000.

[RSH01] S.T. ROWEIS, L.K. SAUL et G. E. HINTON : Global coordination of local linear models. *In* T. G. DIETTERICH, S. BECKER et Z. GHAHRAMANI, éditeurs : *Advances in neural information processing systems*, volume 14, pages 889–896. MIT Press, 2001.

[Sam69] J.W. SAMMON : A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, 1969.

[Sco92] D. W. SCOTT : *Multivariate density estimation : Theory, practice, and visualization*. Wiley, 1992.

[Sha70] D.F. SHANNO : Conditioning of quasi-Newton methods for function minimization. *Mathematical Computation*, 24:647–656, 1970.

[SS02] B. SCHOLKÖPF et A.J. SMOLA : *Learning with kernels : Support vector machines, regularization, optimization and beyond*. MIT Press, janvier 2002.

[SY06] W. SUN et Y.-X. YUAN : *Optimization theory and methods : Nonlinear programming*. Springer édition, 2006.

[TB99] M. TIPPING et C. BISHOP : Mixtures of probabilistic principal component analysers. *Neural computation*, 11(2):443–482, 1999.

[TdSL00] J.B. TENENBAUM, V. de SILVA et J.C. LANGFORD : A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2.22, décembre 2000.

[TN02] P. TINO et I. NABNEY : Hierarchical GTM : Constructing localized non linear projection manifolds in a principled way. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):639–656, mai 2002.

[Tor52] W S. TORGERSON : Multidimensional scaling : 1. Theory and method. *Psychometrika*, 17:401–419, 1952.

[Tru76] G.V. TRUNK : Statistical estimation of the intrinsic dimensionality of a noisy signal collection. *IEEE Transactions on Computers*, 25(2):165–171, février 1976.

[Vap98] V.N. VAPNIK : *Statistical Learning Theory*. Wiley, 1998.

[VD95] P.J. VERVEER et R. DUIN : An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 79(1):81–86, janvier 1995.

[VHC07] T. VIK, F. HEITZ et P. CHARBONNIER : Robust pose estimation and recognition using non- Gaussian modeling of appearance subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:901–905, 2007.

[VRV03] J.J. VERBEEK, S.T. ROWEIS et N. VLASSIS : Non-linear CCA and PCA by alignment of local models. *In Advances in neural information processing systems*, volume 16, 2003.

[Wei99] Y. WEISS : Segmentation using eigenvectors : A unifying view. *In IEEE international conference on computer vision*, volume 14, pages 975–982, 1999.

[WLW04] T.-F. WU, C.-J. LIN et R.C. WENG : Probability estimates for multi-class classification by pair-wise coupling. *Journal of machine learning research*, 5:975–1005, 2004.

[Wol69] P. WOLFE : Convergence conditions for ascent methods. *SIAM Review*, 11:226–235, 1969.

[Wol71] P. WOLFE : Convergence conditions for ascent methods, (II) : some corrections. *SIAM Review*, 13:185–188, 1971.

[WS01] C.K.I. WILLIAMS et M. SEEGER : Using the Nyström method to speed up kernel machines. *In Advances in neural information processing systems*, volume 13, pages 682–688. MIT Press, 2001.

[WS05] S. WANG et X. SUN : Generalization of hinging hyperplanes. *IEEE Transactions on Information Theory*, 51(12):4425–4431, décembre 2005.

[WZZ05] J. WANG, Z. ZHANG et H. ZHA : Adaptive manifold learning. *In Advances in neural information processing systems*, volume 17, pages 1473–1480, 2005.

[YXZ+07] S. YAN, D. XU, B. ZHANG, H.-J. ZHANG, Q. YANG et S. LIN : Graph embedding and extensions : A general framework for dimensionality reduction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(1):40–51, janvier 2007.

[ZZ05] Z. ZHANG et H. ZHA : Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. *SIAM journal on scientific computing*, 26(1):313–338, 2005.