Réseaux de neurones génératifs pour la résolution du problème de pré-image

Clement Gledel

Normandie Université

UNIVERSITÉ DE ROUEN NORMANDIE

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Rouen Normandie

Réseaux de neurones génératifs pour la résolution du problème de pré-image

Présentée et soutenue par
CLEMENT GLEDEL

Thèse soutenue le 13/10/2023 devant le jury composé de		
M. HACHEM KADRI	MAITRE DE CONFERENCES HDR, Aix-Marseille Université	Rapporteur du jury
M. ROMAIN RAVEAUX	MAITRE DE CONFERENCES HDR, UNIVERSITE DE TOURS	Rapporteur du jury
M. LUC BRUN	PROFESSEUR DES UNIVERSITES, ENSI DE CAEN	Membre du jury
M. CÉCILE CAPPONI	PROFESSEUR DES UNIVERSITES, Aix-Marseille Université	Membre du jury
M. BENOÎT GAUZERE	MAITRE DE CONFERENCES, Institut National des Sciences Appliquées Rouen Normandie	Membre du jury
M. PAUL HONEINE	PROFESSEUR DES UNIVERSITES, Université de Rouen Normandie	Directeur de thèse

Thèse dirigée par PAUL HONEINE (Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes)

MIIΣ

litis

Résumé

Que ce soit en reconnaissance des formes ou en apprentissage machine, il est courant de définir un espace de caractéristiques dans lequel les représentations des données sont obtenues en appliquant une transformation non-linéaire ou implicite. Cependant, cette transformation manque souvent d'interprétabilité en raison de l'accès limité à l'espace de caractéristiques. Par conséquent, il peut être fortement souhaitable de représenter, dans l'espace des observations, des éléments de l'espace de caractéristiques. Néanmoins, l'obtention de la transformation inverse est une tâche difficile qui implique la résolution du problème de pré-image. Cette tâche devient d'autant plus difficile lorsqu'il s'agit de données structurées telles que les graphes, qui sont complexes et discrets par nature.

Dans cette thèse, nous proposons plusieurs approches cherchant à résoudre le problème de pré-image grâce à l'application de modèle génératif profond. Pour ce faire, nous commençons par introduire les concepts essentiels qui constituent nos contributions. Ces concepts comprennent le problème de pré-image et les méthodes d'apprentissage machine, en particulier les noyaux, les modèles d'apprentissage profond, ainsi que les différentes approches génératives basées sur les réseaux de neurones. À partir de cette analyse, nous proposons l'utilisation de modèle génératif profond appelé Normalizing Flows (NF), qui se révèle particulièrement adapté à notre problématique. Ce choix est motivé par la propriété d'inversibilité exacte offerte par les NF. De cela, nous introduisons une première approche qui a pour objectif de résoudre le problème de pré-image rencontré lors de l'utilisation de noyau. Cette méthode repose sur l'alignement de l'espace de caractéristiques implicitement généré par le noyau avec l'espace latent généré par un NF. Ce modèle étant inversible par nature, le problème de pré-image associé au noyau peut être considéré résolu par sa transposition vers l'espace généré par le NF. Notre contribution suivante vise à proposer un formalisme général permettant la définition de méthodes d'apprentissage machine exemptes du problème de pré-image. Pour ce faire, nous proposons d'adapter l'apprentissage d'un NF, de manière non supervisée, qui, à partir d'une distribution de données complexe, permet la génération d'un espace de représentation dans lequel les données suivent une distribution prédéfinie. Combiné aux notions de projection par analyse en composantes principales, nous définissons deux méthodes de débruitage

travaillant dans l'espace généré. De plus, l'inversibilité de chaque transformation permet la génération de nouvelles données à partir de points d'intérêt de l'espace latent, ce qui permet de résoudre le problème de pré-image. Ainsi, basé sur ce formalisme insensible au problème de pré-image, nous présentons deux spécifications traitant deux tâches différentes. Pour chacune de ces tâches de prédiction, à savoir classification et régression, nous proposons une méthode d'apprentissage supervisée permettant la définition d'un modèle prédictif incluant la fonction de calcul de la pré-image de tout point de l'espace de caractéristiques généré. Un chapitre présente l'application de ses approches sur des données vectorielles et d'images. Enfin, nous nous intéressons à leurs applications sur des données structurées de type graphe. Ces données complexes et discrètes sont présentes dans de nombreux domaines permettant la modélisation de relation entre entités. Par exemple, les graphes sont utilisés en bio-informatique pour représenter des molécules composées de liaisons entre atomes. Ainsi, nous proposons d'étendre nos approches supervisées à ce type de données par l'utilisation de NF de graphe. Ceci permettant la génération d'un espace de représentation continu de graphe dans lequel des opérations et méthodes d'apprentissage machine peuvent être appliquées tout en permettant le calcul de graphe pré-image grâce à l'inversibilité de notre modèle.

Abstract

Whether in pattern recognition or machine learning, it is a common practice to define a feature space in which data representations are obtained by applying a nonlinear or implicit transformation. However, this transformation often lacks interpretability due to limited access to the feature space. Consequently, it may be highly desirable to represent elements of the feature space in the observation space. Nevertheless, obtaining the inverse transformation is a difficult task, which involves solving the pre-image problem. This task becomes even more difficult when dealing with structured data such as graphs, which are complex and discrete by nature.

In this thesis, we propose several approaches to solve the pre-image problem through the application of deep generative models. To this end, we begin by introducing the essential concepts that constitute our contributions. These concepts include the pre-image problem and machine learning methods, in particular kernels, deep learning models, as well as the various generative approaches based on neural networks. Based on this analysis, we propose the use of deep generative models called "Normalizing Flows" (NFs), which is particularly well-suited to our problem. This choice is motivated by the property of exact invertibility offered by NFs. Next, we introduce a first approach which aims at solving the pre-image problem encountered when using kernels for nonlinear embeddings. This method is based on aligning the feature space implicitly generated by the kernel with the latent space generated by a NF. As this model is invertible by nature, the pre-image problem associated with the kernel can be considered solved by transposing it to the space generated by the NF. Our next contribution aims at proposing a general framework that allows the definition of machine learning methods free of the pre-image problem. To this end, we propose to adapt the learning of an NF, in an unsupervised way, which, starting from a complex data distribution, allows the generation of a representation space in which the data follow a predefined distribution. Combined with the notions of principal component analysis, we define two denoising methods operating in the generated space. In addition, the invertibility of each transformation allows new data to be generated from points of interest in latent space, thus solving the pre-image problem. Thus, based on this framework insensitive to the pre-image problem, we introduce two specifications dealing with two different tasks. For each of these pre-

diction tasks, namely classification and regression, we propose a learning method allowing the definition of a predictive model including the inverse transformation used to compute the pre-image of any point in the generated feature space. A chapter introduces the application of these approaches to vector and image data. Finally, we look at their application to structured graph data. These complex and discrete data are present in many fields, allowing the representation of relationships between entities. For example, graphs are used in bioinformatics to represent molecules composed of bonds between atoms. We propose to extend our supervised approaches to this type of data by using graph NF. This allows the generation of a continuous graph representation space in which operations and machine learning methods can be applied, while also allowing the computation of graphs pre-images thanks to the invertibility of our model.

Table des matières

Table des figures

Liste des tableaux

Liste des algorithmes

Acronymes

ACP Analyse en composantes principales
AR Auto-régressif
CNF Normalizing flows continues
CNN Réseau de neurones convolutionnel
ConvGNN Réseau de neurones convolutionnel de graphes
ELBO Borne inférieur de l'évidence
GAN Réseau de neurones antagoniste génératif
GNN Réseau de neurones sur graphes
GraphNF Normalizing flows de graphes
GRU Gated Recurrent Unit
IVP Problème de valeur initiale
LSTM Long Short-Term Memory
ML Apprentissage machine
MLP Perceptron multicouches
NF Normalizing flows
ODE Équation différentielle ordinaire
PR Reconnaissance des formes
RecGNN réseau de neurones récurrents de graphes
RKHS Espaces de Hilbert à noyau reproduisant
RNN réseau de neurones récurrents
SVM Machine à vecteurs de support
VAE Autoencodeur variationnel
VI Inférence variationnelle

Introduction

L'apprentissage machine a connu de grandes avancées ces dernières années, notamment grâce à sa capacité à extraire et à représenter efficacement des informations à partir de données. Ceci est rendu possible par le rôle essentiel des espaces de représentation offrant une structure qui permet de capturer et de révéler les motifs et les relations sous-jacentes aux données [8]. En conséquence, ces espaces ont été largement exploités dans divers domaines d'application tels que la vision par ordinateur [67], le traitement du langage naturel [79] et bien d'autres. Un espace de représentation fait référence à un espace significatif et structuré dans lequel les caractéristiques et les relations des données sont mieux capturées. Cet espace est obtenu en appliquant une transformation aux données observées, permettant ainsi d'en extraire les caractéristiques. Par exemple, pour une tâche de classification, les espaces de représentation offrent un avantage majeur en permettant des représentations de données linéairement séparables. En projetant les données dans un espace où les différentes classes ou catégories sont distinctes, il devient plus facile de construire des modèles d'apprentissage machine performants. Les algorithmes prédictifs de classification et de régression peuvent alors tirer parti de ces représentations pour obtenir des prédictions plus précises.

Les graphes sont largement utilisés dans de nombreux domaines, car ils offrent une représentation puissante et flexible des relations entre entités. La richesse des informations contenues dans les graphes en fait des objets d'étude essentiels dans de nombreuses disciplines, notamment l'informatique, les mathématiques, la physique, la biologie et les sciences sociales. Ils sont couramment utilisés pour modéliser les réseaux sociaux [115] permettant des applications telles que les systèmes de recommandations [43], en bio-informatique où ils permettent la modélisation de molécules [126] ou en vision par ordinateur comme pour la représentation de nuage de points 3D [28]. Par exemple, dans le domaine de la bio-informatique, les graphes jouent un rôle crucial où ils sont largement utilisés pour de nombreuses applications. L'une de ces applications majeures est la conception de médicaments [80] où les graphes sont utilisés pour modéliser et analyser les interactions moléculaires permettant l'identification de nouveaux médicaments potentiels, ou la prédiction de leur efficacité. L'un des principaux défis consiste à extraire des informations si-

gnificatives des graphes afin de révéler des structures, des motifs et des propriétés intéressantes [18, 133]. Cependant, en raison de la complexité des graphes réels, cette tâche peut s'avérer difficile. Les méthodes d'apprentissage machine, en particulier l'apprentissage profond, offrent des perspectives prometteuses pour l'analyse des graphes [82]. Elles permettent d'extraire des représentations à partir de données de graphe qui capturent des informations structurelles et contextuelles. De plus, ces méthodes permettent l'utilisation de représentations pour résoudre une variété de tâches d'analyse et de prédiction.

Cependant, les méthodes d'apprentissage machine permettant la génération d'un espace de représentation, quel que soit le type de données, sont la plupart du temps confrontées à un problème important, à savoir le problème de pré-image. Le problème de pré-image peut être défini comme la recherche de la donnée antécédente, appelée pré-image, appartenant à l'espace des données observées et correspondant à un point d'intérêt issue de l'espace des caractéristiques généré. Ce point d'intérêt, peut être échantillonné ou calculé dans l'espace de caractéristiques, et pour lequel il est pertinent de connaître la donnée correspondante dans l'espace des observations. En effet, il existe de nombreuses tâches pour lesquelles connaître la pré-image de point d'intérêt peut être intéressant. Parmi celles-ci, nous pouvons tout d'abord noter que toutes tâches génératives nécessitent la connaissance d'une transformation inverse permettant la génération de données pré-images à partir de représentations d'un espace de caractéristiques. De plus, des tâches du type débruitage ou compression impliquent l'obtention de données pré-images de représentations obtenues par projection sur un sous-espace de l'espace de représentation, nécessitant ainsi une connaissance de la transformation inverse. Lorsque le problème de pré-image apparaît pour les tâches citées ci-dessus, il est aussi possible d'être confronté à ce problème lors de l'utilisation de données structurées de type graphe. En effet, les graphes décrivent des structures de données complexes et discrètes rendant leur encodage dans un espace de représentation nécessaire si l'on souhaite leur appliquer des opérations ou des méthodes d'apprentissage machine. Par exemple, lorsque le calcul de centroïde d'un ensemble de graphes est impossible à réaliser dans l'espace des observations, il est possible de moyenner des représentations de graphe dans l'espace de caractéristiques. Ceci résultant en un point d'intérêt auquel l'obtention de son graphe pré-image dans l'espace des observations est nécessaire.

Cette thèse a été réalisée avec l'objectif de proposer des solutions au problème de pré-image. Les solutions proposées reposent sur les nombreuses récentes avancées sur les modèles génératifs profonds et plus particulièrement sur les architectures appelées Normalizing Flows (NF). S'appuyant sur leurs propriétés d'inversibilité exacte, nous proposons dans ce document d'utiliser ces modèles pour définir plusieurs approches insensibles au problème de pré-image. Ces approches s'inscrivent dans plusieurs contextes d'apprentissages, allant tout d'abord d'un contexte non supervisé vers des spécifications pour des tâches supervisées, à savoir classification et régression. De plus, les graphes étant un type de données dont la génération de représentation est essentielle, nous nous intéressons aussi à étendre notre analyse à ce genre de données.

Contributions

Les principales contributions de la thèse sont :

1. Notre première contribution est présentée au Chapitre 2 et propose une méthode pour générer une solution au problème de pré-image rencontré lors de l'utilisation des méthodes à noyaux. Notre approche est basée sur la transposition du problème de pré-image, présent dans l'espace du noyau, vers un espace de caractéristiques généré par un NF. Le NF étant inversible, chaque représentation ou point d'intérêt appartenant à son espace peut être généré en appliquant sa transformation inverse. Pour effectuer cette transposition, nous considérons la définition d'une notion d'alignement entre les espaces. Pour cela, nous proposons l'utilisation d'un modèle de réseau de neurones permettant la génération d'un espace aligné sur celui du noyau et travaillant sur les représentations des données de l'espace du NF. Ainsi, notre méthode repose sur l'utilisation de ce modèle pour sélectionner une représentation dans l'espace du NF qui, une fois générée, correspond au point d'intérêt appartenant à l'espace noyau. La méthode est évaluée à l'aide d'un noyau classique, sur un ensemble de données d'images avec plusieurs architectures de NF différentes.
2. Une deuxième contribution, introduite dans le Chapitre 3, consiste en un formalisme permettant la définition d'approches d'apprentissage machine dans un contexte non supervisé. Ce formalisme repose sur l'adaptation de la méthode d'apprentissage d'un NF visant à produire un espace de représentation où les données sont organisées suivant une distribution de données prédéfinie. Combinée aux notions de projection par analyse en composantes principales, l'approche est utilisée pour des tâches de type débruitage/compression et de génération de données pré-images. Ainsi, afin de démontrer la pertinence de l'approche proposée, nous expérimentons sur des jeux de données vectorielles et d'images et obtenons des performances supérieures à celles résultant de l'application de différentes méthodes à noyaux, tout en permettant la génération de n'importe quelle pré-image.
3. De cela, nous proposons une troisième contribution, présentée dans le Chapitre 4, en revisitant le formalisme précédent pour une tâche supervisée de type classification. Cette fois, l'apprentissage du NF est adapté de façon à prendre en compte les classes des données d'entraînement, permettant la génération d'un espace de représentation où les classes sont linéairement séparables. Grâce à cela, l'approche permet à la fois la classification de données par l'application d'opérations linéaires dans l'espace de caractéristiques et la génération de données pré-images à partir de points d'intérêt. Ces derniers peuvent être échantillonnés d'une manière spécifique à un contexte de classification, c'est-à-dire sur la base d'une valeur d'étiquette souhaitée. Nous démontrons les bonnes performances de classification et de génération de pré-images sur des jeux de données vectorielles et d'images connues en comparant nos résultats avec ceux obtenus à l'aide de diverses méthodes à noyaux.
4. Toujours dans le Chapitre 4, nous présentons une quatrième contribution qui revisite notre formalisme introduit dans le Chapitre 3, pour une tâche

de régression supervisée. Dans ce cas, l'apprentissage du NF s'appuie sur les valeurs de régression quantitatives associées aux données. L'espace de représentation est ainsi modélisé de façon à ce que les données soient organisées de façon linéaire en fonction des valeurs quantitatives qui leur sont attribuées. De cette manière, notre approche permet des prédictions spécifiques à une tâche par l'application de méthodes linéaires dans l'espace de caractéristiques, tout en offrant la possibilité de générer des pré-images à partir de points d'intérêt. De plus, en raison de la structure de l'espace de caractéristiques spécifique au contexte de régression, des points d'intérêt peuvent être échantillonnés en fonction d'une valeur quantitative désirée. Ainsi, nous présentons les bonnes performances de notre approche en termes de génération de pré-images et de prédiction en la comparant à des méthodes à noyaux sur différents jeux de données vectorielles et d'images.

5. Une cinquième contribution, introduite dans le Chapitre 5, constitue une extension de notre approche de classification présentée dans la troisième contribution. Motivé par les bonnes performances de cette dernière, nous proposons de l'adapter à des données complexes de type graphe. Pour cela, un NF de graphe (GraphNF) est appliqué pour générer un espace de représentation continu. Bien qu'il existe de nombreuses représentations de graphes, nous proposons l'utilisation d'un modèle de GraphNF travaillant sur une représentation matricielle spécifique. Ainsi, cette approche de classification permet la génération d'un espace de représentation de graphe continu dans lequel ces représentations sont linéairement séparables. Cet espace offre la possibilité d'appliquer des méthodes d'apprentissage machine aux graphes, permettant la prédiction d'étiquettes associées par l'application de simples opérations linéaires. De plus, l'architecture du modèle permet la génération de graphes pré-images de tout point d'intérêt pouvant être échantillonné à partir de valeur d'étiquette souhaitée. Les bonnes performances de notre approche sont démontrées sur différents jeux de données de classification de graphes en mettant en évidence de meilleurs résultats que ceux obtenus avec des méthodes à noyaux classiques et des méthodes à noyaux de graphe.

6. Notre dernière contribution, également présentée dans le Chapitre 5, propose d'étendre notre quatrième contribution, permettant l'apprentissage supervisé dans un contexte de régression, sur des données de type graphe. De manière similaire à notre approche de classification de graphe, nous employons un GraphNF travaillant sur une représentation matricielle des graphes pour générer un espace de caractéristiques continu. Dans ce cas, les représentations de graphes sont organisées linéairement selon leurs valeurs quantitatives associées à la tâche de régression spécifique. Ainsi, notre approche permet la génération de graphes pré-images à partir de représentations de graphes échantillonnées en fonction d'une valeur de régression désirée. Nous démontrons les bonnes performances en régression de notre approche en la comparant aux résultats obtenus avec des méthodes à noyaux simples et des méthodes à noyaux de graphes sur divers jeux de données de classification de graphe.

Structure de la thèse

Pour présenter les différents travaux et contributions effectués, la thèse est structurée de la manière suivante :

Chapitre 1 Dans ce chapitre, nous présentons les notions importantes à partir desquelles les travaux ont été réalisés. Ces notions sont constituées de l'apprentissage machine (Section 1.1), du problème de pré-image (Section 1.2) ainsi que les modèles génératifs profonds (Section 1.3).

Chapitre 2 Ensuite, nous introduisons une première approche visant à utiliser une architecture de modèle génératif pour apporter une solution au problème de pré-image rencontré lors de l'utilisation de méthodes à noyaux. Pour cela, l'approche proposée est tout d'abord décrite dans la Section 2.2 puis les expérimentations menées sont présentées dans la Section 2.3. De ces expériences, nous concluons ce travail dans la Section 2.4.

Chapitre 3 Dans ce chapitre, nous proposons un nouveau formalisme, s'inscrivant dans un contexte d'apprentissage non supervisé, permettant la définition d'un modèle d'apprentissage machine insensible au problème de pré-image. Ce formalisme, détaillé dans la Section 3.2, est basé sur l'utilisation d'une architecture de réseau de neurones génératif dont l'apprentissage est adapté à l'aide de concepts d'analyse en composantes principales (ACP). Cette approche est constituée de deux méthodologies différentes que l'on évalue dans la Section 3.3 pour des tâches de débruitage, compression et de génération de pré-image. Enfin, la Section 3.4 conclut ce chapitre.

Chapitre 4 À partir du formalisme précédent, nous proposons dans ce chapitre deux spécifications différentes pour des tâches supervisées de classification et de régression. Ces deux méthodes permettent la définition de modèle d'apprentissage machine sans problème de pré-image dans un contexte de résolution d'une tâche prédictive. Pour cela, les deux approches sont présentées dans la Section 4.2, puis évaluées dans la Section 4.3. De cela, nous proposons une section de discussion (Section 4.4) où nous apportons une analyse supplémentaire sur les expériences et les approches proposées. Enfin, nous concluons ces travaux dans la Section 4.5.

Chapitre 5 Les méthodologies spécifiées dans le chapitre précédent étant pertinentes, nous nous intéressons dans ce chapitre à leurs applications sur des données de type graphe. Les approches, détaillées dans la Section 5.2, permettent la définition de méthodes d'apprentissage machine sans problème de pré-image sur des données de type graphe. Ces approches sont évaluées dans la Section 5.3 puis nous concluons dans la Section 5.4.

Chapitre 6 Dans ce dernier chapitre, nous apportons une conclusion à la thèse par un résumé de chacune des contributions présentées dans la Section 6.1 puis indiquons des potentielles perspectives de travaux dans la Section 6.2.

Liste des publications

Les travaux de recherche de cette thèse ont donné lieu aux articles suivants :

1. Clément Glédel, Benoit Gaüzère, and Paul Honeine. “Normalizing flow appliqué aux problèmes de pré-image de noyau”. In 24-ème Conférence d’Apprentissage automatique (CAp), pages 1–10, 2022.
2. Clément Glédel, Benoit Gaüzère, and Paul Honeine. “Machine learning without the pre-image problem thanks to normalizing flows”. Submitted for publication to Neurocomputing. 2023.
3. Clément Glédel, Benoit Gaüzère, and Paul Honeine. “Regression without the pre-image problem thanks to normalizing flows”. Submitted for publication to Pattern Recognition Letters. 2023.
4. Clément Glédel, Benoit Gaüzère, and Paul Honeine. “Normalizing flows pour éviter le problème de pré-image”. In : GRETSI 2023 : XXIXe Colloque. 2023.
5. Clément Glédel, Benoit Gaüzère, and Paul Honeine. “Graph normalizing flows to pre-image free machine learning for regression”. In : 13th IAPR-TC15 International Workshop on Graph-Based Representations in Pattern Recognition. 2023.

CHAPITRE 1

Le problème de pré-image et son contexte

Afin d'introduire ce document de thèse, il est nécessaire de présenter dans un premier temps le contexte dans lequel il s'inscrit. Pour cela, ce premier chapitre présente les différentes notions importantes à la compréhension du document séparées en trois grandes parties. Une première constituée d'une introduction générale à l'apprentissage machine ainsi qu'à trois de ses approches importantes, à savoir les méthodes à noyaux, les réseaux de neurones et les réseaux de neurones sur graphes. Une deuxième partie introduit le problème auquel la thèse s'intéresse appelé le problème de pré-image. La dernière partie présente les principales approches constituant les modèles génératifs profonds.

1.1 L'apprentissage machine

L'apprentissage machine (ML) a contribué à améliorer notre capacité à traiter et à analyser automatiquement de grandes quantités de données. Cette approche implique le développement de modèles et d'algorithmes capables d'apprendre par expérience et d'effectuer des tâches spécifiques sans être explicitement programmés.

Pour définir le principe de base des méthodes de ML, nous considérons un ensemble $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\}$ avec $\mathbf{x}_i$ une donnée appartenant à l'espace des observations $\mathcal{X}$ et y_i la propriété associée à cette donnée. Le principe de l'apprentissage machine est de déterminer une fonction paramétrique f_θ qui à partir d'une entrée $\mathbf{x}_i \in \mathcal{X}$ résulte sur la sortie associée y_i. Ainsi, ces approches cherchent à modéliser la fonction f_θ par l'apprentissage de paramètres θ de sorte que $f_\theta(\mathbf{x}_i) = \hat{y}_i$ avec $\hat{y}_i$ la sortie souhaitée égale à y_i.

Pour cela, l'apprentissage est effectué par la définition d'une fonction de perte, notée $\mathcal{L}$, permettant l'évaluation de l'erreur commise par le modèle f_θ entre la prédiction $\hat{y}_i$ et la vérité terrain y_i. Ainsi, la minimisation de cette perte par l'ajustement des paramètres θ, permet l'optimisation de ces derniers vers la solution optimale $\theta^\star$,

c'est-à-dire :

$$\theta^\star = \underset{\theta}{\operatorname{argmin}} \mathcal{L}(\mathbf{x}_i, y_i, \theta, f_\theta), \forall (\mathbf{x}_i, y_i) \in X.$$

Il existe différentes méthodes pour minimiser cette fonction de perte par rapport aux paramètres à optimiser. L'une des approches les plus connues est l'optimisation par descente de gradient sous réserve que la fonction de perte soit différentiable. Cette méthode implique le calcul du gradient de la perte par rapport aux paramètres à optimiser, noté $\nabla_\theta \mathcal{L}$, utilisé afin d'effectuer une mise à jour itérative des paramètres θ suivant

$$\theta \leftarrow \theta - \eta \sum_{(x_i, y_i) \in X} \nabla_\theta \mathcal{L}(\mathbf{x}_i, y_i, \theta, f_\theta),$$

avec η un hyperparamètre représentant le taux d'apprentissage contrôlant la taille des pas effectués lors de la mise à jour des paramètres. Cette approche d'optimisation permet une mise à jour des paramètres dans la direction opposée du gradient de la perte. Dans sa forme actuelle, cette approche présente quelques inconvénients. Parmi ceux-ci, nous pouvons mentionner la convergence vers des minima locaux, ainsi qu'un calcul potentiellement important à cause de l'utilisation de l'ensemble complet de données. Pour remédier à ces problèmes, une approche appelée calcul du gradient par mini-lot a été introduite [74]. Cette méthode consiste à diviser l'optimisation appliquée à l'ensemble complet des données en plusieurs itérations, chaque itération étant effectuée sur un lot de données de taille fixe. Basées sur cela, des méthodes d'optimisation couramment utilisées ont été proposées [123, 57].

Dans cette introduction à l'apprentissage machine, nous décrivons trois familles de méthodes importantes dans ce domaine : les méthodes à noyaux, les réseaux de neurones et les réseaux de neurones sur graphes.

1.1.1 Les méthodes à noyaux et les espaces de Hilbert à noyau reproduisant (RKHS)

Nous considérons $\mathcal{X}$, l'espace des observations avec $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \in \mathcal{X}$ un jeu de données observé. Les méthodes à noyaux constituent une approche de ML basée sur la notion de similarité entre données. Elles permettent la résolution de divers problèmes présents en ML tels que les problèmes de débruitage/compression, classification et régression.

L'idée des méthodes à noyaux reposent sur l'encodage des données dans un espace de caractéristiques de dimension potentiellement supérieure à celle de l'espace des observations. Pour cela, ces méthodes utilisent des fonctions noyaux [114, 116] permettant l'obtention de valeurs de similarité à partir de paires de données. L'espace de caractéristiques généré par ces fonctions permet une meilleure représentation des données où l'application d'opérations linéaires fonctionne efficacement. Ainsi, en choisissant une fonction noyau appropriée, il est possible de générer un espace de caractéristiques dans lequel les représentations des données présentent une meilleure séparabilité, facilitant ainsi une tâche de classification par exemple.

La transformation induite par les noyaux permet la résolution de problèmes complexes linéairement insolvables par l'application d'une transformation non-linéaire aux données. L'espace de caractéristiques obtenu par cette transformation permet

TABLE 1.1 – Exemples de 4 noyaux applicables à des données vectorielles.

Types	Expressions
Linéaire	$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2$
RBF	$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \lvert\rvert \mathbf{x}_1 - \mathbf{x}_2 \rvert\rvert^2\right)$
Polynomial	$\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)^n$
Sigmoïde	$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)$

ainsi une représentation des données permettant la résolution de ces problèmes de manière linéaire. De plus, ce qui rend ces méthodes attrayantes et pertinentes est la non-nécessité d'expliciter la transformation vers cet espace. En effet, les noyaux définissent les similarités entre données dans l'espace de caractéristiques généré sans y avoir explicitement accès.

L'application de méthodes à noyaux se fait par la sélection d'un noyau approprié aux données et au contexte. Un noyau peut être défini de la façon suivante :

Définition 1.1 (Noyau)**.** *Un noyau représente une fonction $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ symétrique telle que $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa(\mathbf{x}_2, \mathbf{x}_1)$.*

Cependant, il est important de noter que les méthodes à noyaux sont conçues pour travailler spécifiquement avec des noyaux dits "définis positifs" [116]. Cette spécification des noyaux offre des propriétés importantes telles que le théorème de représentation de Mercer [86] ainsi que l'application d'outils mathématiques pour la résolution de problèmes d'optimisation, d'estimation de paramètres, etc. De tels noyaux peuvent être spécifiés comme suit :

Définition 1.2 (Noyau défini positif)**.** *Un noyau κ est dit défini positif si la condition*

$$\sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \geq 0, \forall\, \mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathcal{X} \text{ et } c_1, \ldots, c_N \in \mathbb{R}$$

est remplie.

Il existe de nombreux noyaux dépendant du type de données observées. La Table 1.1 présente des noyaux définis positifs couramment utilisés sur des données vectorielles.

L'utilisation de noyau implique la définition d'une notion importante, à savoir l'espace de Hilbert pouvant être présenté comme suit :

Définition 1.3 (Espace de Hilbert)**.** *Un espace de Hilbert est un espace vectoriel $\mathcal{H}$ muni d'un produit scalaire $\langle u, v \rangle_{\mathcal{H}}$ et d'une norme induite telle que : $\|u\|_{\mathcal{H}}^2 = \langle u, u \rangle_{\mathcal{H}}$, pour tout $u, v \in \mathcal{H}$.*

De plus, une spécification d'espace de Hilbert, importante lors de l'utilisation de noyau définis positifs, est appelée espace de Hilbert à noyaux reproduisant (RKHS) :

Définition 1.4 (Espace de Hilbert à noyau reproduisant)**.** *Si $\mathcal{H}$ est l'espace de Hilbert constitué de fonctions de $\mathcal{X}$ dans $\mathbb{R}$, alors le noyau défini positif $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ est le noyau reproduisant de $\mathcal{H}$ si, et seulement si, avec $\kappa(\mathbf{x}, \cdot)$ la fonction noyau appliquée au point $\mathbf{x}$, nous avons $f(\mathbf{x}) = \langle \kappa(\mathbf{x}, \cdot), f \rangle_{\mathcal{H}}, \forall f \in \mathcal{H}$ et $\mathbf{x} \in \mathcal{X}$. Dans ce cas, $\mathcal{H}$ est appelé espace de Hilbert à noyau reproduisant.*

La propriété clé d'un RKHS est que les noyaux permettent de représenter efficacement les opérations de l'espace de Hilbert. Cela signifie que les calculs entre les fonctions peuvent être effectués à travers les produits scalaires induits par le noyau, évitant ainsi le besoin de manipuler explicitement les fonctions. De cela, le théorème de Moore-Aronszajn [3] est défini :

Théorème 1.1 (Théorème de Moore-Aronszajn [3])**.** *Pour chaque noyau défini positif κ, il existe un espace de Hilbert à noyau reproduisant dans lequel κ est un noyau reproduisant.*

Ainsi, nous pouvons définir le théorème de Mercer [86] de la manière suivante :

Théorème 1.2 (Théorème de Mercer [86])**.** *Pour tout noyau défini positif $\kappa(\mathbf{x}_1, \mathbf{x}_2)$, il existe une fonction implicite $\Phi(\mathbf{x})$ qui transforme des données observées $\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N \in \mathcal{X}$ vers un espace de Hilbert $\mathcal{H}$, tel que :*

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle_{\mathcal{H}}$$

L'astuce du noyau est une technique pratique qui découle de ce théorème permettant d'exploiter les avantages des noyaux définis positifs sans avoir à effectuer explicitement la transformation implicite Φ vers l'espace de Hilbert $\mathcal{H}$ de dimension potentiellement infinie. Les calculs sont effectués directement dans l'espace des observations, en utilisant le produit scalaire induit par le noyau. Ainsi, en utilisant l'astuce du noyau, il est possible de résoudre des problèmes d'apprentissage machine dans des espaces de dimensions élevées, voire infinies, sans rencontrer les limitations de calcul et de stockage associées à ces espaces de grande dimension. Dans la suite de ce document, nous utiliserons le terme de noyau pour désigner tout noyau défini positif.

L'une des pierres angulaires de l'apprentissage machine est le théorème de représentation [113] mis en évidence dans les méthodes à noyaux [116], les réseaux de neurone profonds [127], et plus généralement dans les problèmes inverses [128]. Ce dernier est formulé de la manière suivante :

Théorème 1.3 (*Théorème de représentation* [113])**.** *Soit un ensemble non vide $\mathcal{X}$, un noyau défini positif κ sur $\mathcal{X} \times \mathcal{X}$, un ensemble d'échantillon d'apprentissage $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots (\mathbf{x}_N, y_N)\} \in \mathcal{X} \times \mathbb{R}$, une fonction réelle strictement monotone g sur $[0, \infty[$ et une fonction de coût arbitraire c. Soit $\mathcal{H}$ l'unique RKHS engendré par une fonction implicite Φ associée au noyau κ. Toute fonction $\boldsymbol{\varphi}^* \in \mathcal{H}$ minimisant le risque fonctionnel régularisé*

$$c((\mathbf{x}_1, y_1, f(\mathbf{x}_1)), \dots, (\mathbf{x}_N, y_N, \varphi(\mathbf{x}_N)) + g(\|f\|),$$

admet une représentation de la forme

$$\boldsymbol{\varphi}^* = \sum_{i=1}^{N} \alpha_i \Phi(\mathbf{x}_i). \tag{1.1}$$

où chaque α_i représente le poids associé à l'échantillon $\mathbf{x}_i$.

Celui-ci stipule que les solutions optimales des méthodes d'apprentissage machinepeuvent être définies comme une combinaison linéaire des transformations des échantillons d'apprentissage.

1.1.2 Les réseaux de neurones

Les réseaux de neurones constituent une approche essentielle dans le domaine de l'apprentissage machine. Inspirés du fonctionnement du cerveau humain, ces modèles informatiques sont conçus pour résoudre des problèmes complexes en imitant la manière dont les neurones biologiques traitent l'information. Grâce à leur capacité d'apprentissage à partir de données, les réseaux de neurones sont utilisés dans un large éventail d'application pour des tâches prédictives telles que la prédiction d'une classe ou d'une propriété à partir d'une entrée, ou pour des tâches génératives permettant la génération de nouvelles données.

Dans cette introduction, nous explorerons les fondements de différentes architectures de réseaux de neurones ainsi que leurs fonctionnements de base, permettant ainsi les bases d'une compréhension plus approfondie. Ces architectures comprennent les perceptrons multicouches (MLP), les réseaux de neurones récurrents (RNN) [105] et les réseaux de neurones convolutifs (CNN) [71].

Perceptron multicouches (MLP)

Un perceptron peut être défini comme un modèle de neurone artificiel qui, à partir d'une entrée $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^D$, applique l'opération suivante :

$$y = \sigma(\mathbf{w}^\top \mathbf{x} + b),$$

où y définit le résultat de l'opération appliquée par le perceptron, $\mathbf{w} \in \mathbb{R}^D$ est un vecteur de paramètres aussi appelés poids, b un biais et σ une fonction d'activation. Cette dernière définit une fonction non-linéaire permettant l'apprentissage de transformation complexe sans se restreindre à des transformations linéaires. Elle est choisie, en général, parmi l'une des trois opérations suivantes :

— Unité Linéaire Rectifiée (ReLU) : $\sigma(x) = \max(0, x)$.
— Sigmoïde : $\sigma(x) = \frac{1}{1-\exp(-x)}$.
— Tangente hyperbolique : $\sigma(x) = \tanh(0, x)$.

L'utilisation de n différents perceptrons sur une même entrée constitue une couche et permet la génération d'une sortie à n dimensions pouvant être définie de la façon suivante :

$$\mathbf{y} = \sigma(\mathbf{W}\,\mathbf{x} + \mathbf{b}),$$

où $\mathbf{y} \in \mathbb{R}^n$ correspond au résultat de la couche de perceptron, $\mathbf{W} \in \mathbb{R}^{n \times D}$ est la matrice de poids, σ une fonction d'activation et $\mathbf{b} \in \mathbb{R}^n$ un vecteur de biais. Dans ce cas, toutes les dimensions de l'entrée $\mathbf{x}$ contribuent à évaluer chaque dimension de la sortie $\mathbf{y}$. Une telle couche est appelée une couche entièrement connectée (*fully-connected layer*).

Un MLP est un réseau de neurones définissant la combinaison de plusieurs de ces couches. La Figure 1.1 illustre un exemple de MLP constitué de 4 couches entièrement connectées qui, à partir d'une entrée $\mathbf{x} \in \mathbb{R}^2$, produit une sortie $\mathbf{y} \in \mathbb{R}^2$.

Réseau de neurones récurrents (RNN)

Les RNN sont des réseaux de neurones conçus pour traiter des séquences de données en maintenant une mémoire interne qui capture l'information contextuelle

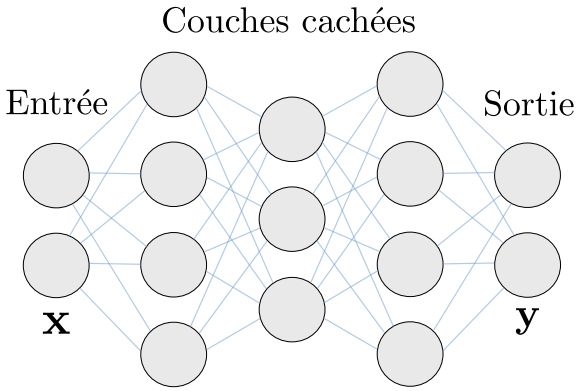


FIGURE 1.1 – Illustration d'un perceptron multicouches (MLP).

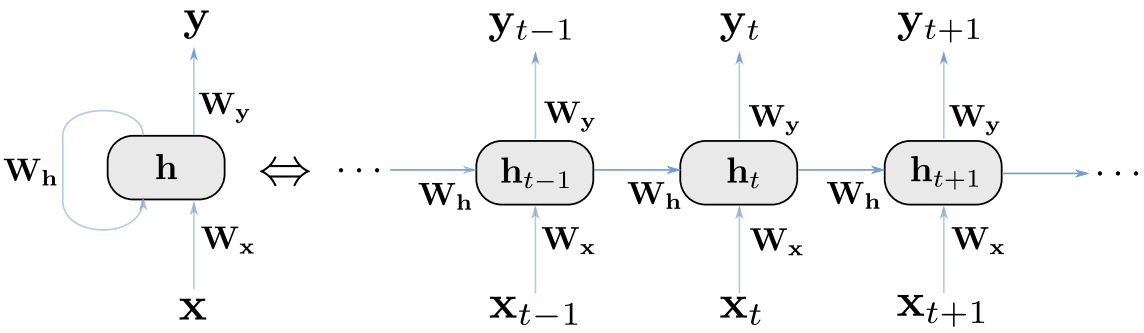


FIGURE 1.2 – Illustration du fonctionnement d'un réseau de neurones récurrents (RNN).

précédente. Cette mémoire récurrente, notée $\mathbf{h}$, leur permet de modéliser les dépendances à long terme dans les séquences.

Pour présenter le principe d'un RNN, nous considérons une donnée observée représentée par une séquence d'entrée $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_D\}$ où $\mathbf{x}_t$ représente la donnée à l'instant t. Un RNN maintient une mémoire cachée $\mathbf{h}_t$ qui capture l'information contextuelle précédente et est mis à jour à chaque pas de temps t avec :

$$\mathbf{h}_t = \sigma_{\mathbf{h}}(\mathbf{x}_t \mathbf{W}_{\mathbf{x}} + \mathbf{h}_{t-1}\mathbf{W}_{\mathbf{h}} + \mathbf{b}_{\mathbf{h}}),$$

où les poids appliqués sur l'entrée et la mémoire stockée sont représentés respectivement par les matrices $\mathbf{W}_{\mathbf{x}}$ et $\mathbf{W}_{\mathbf{h}}$. Le biais utilisé est représenté par le vecteur $\mathbf{b}_{\mathbf{h}}$ et une fonction d'activation $\sigma_{\mathbf{h}}$ est appliquée. Ainsi, de cette nouvelle valeur de mémoire, la sortie $\mathbf{y}_t$ peut être calculée de la manière suivante :

$$\mathbf{y}_t = \sigma_{\mathbf{y}}(\mathbf{h}_t \mathbf{W}_{\mathbf{y}} + \mathbf{b}_{\mathbf{y}}),$$

où $\mathbf{W}_{\mathbf{y}}$ représente la matrice de poids appliquée à la mémoire et $\mathbf{b}_{\mathbf{y}}$ le biais associé. De même, une fonction d'activation $\sigma_{\mathbf{y}}$ est utilisée afin d'obtenir la valeur de sortie à l'instant t. La Figure 1.2 illustre cette approche.

Cependant, les RNN peuvent être limités dans leur capacité à capturer des dépendances à long terme et à modéliser des relations complexes dans des séquences

de données très longues [9]. C'est pourquoi des variantes plus avancées des RNN, telles que les LSTM (Long Short-Term Memory) [47] et les GRU (Gated Recurrent Unit) [20], ont été proposées afin d'améliorer leur capacité de mémoire et leur capacité à modéliser des séquences à long terme.

Réseau de neurones convolutionnel (CNN)

Les réseaux de neurones convolutionnels (CNN) constituent une classe importante d'architectures de réseaux de neurones utilisées principalement sur des données de type image. Ces réseaux ont permis d'importants progrès dans le domaine de la vision par ordinateur, notamment sur des tâches telles que la classification d'images, la détection d'objets et la segmentation d'images.

Les CNN permettent une meilleure capacité à capturer des motifs locaux et les caractéristiques d'une image en utilisant des opérations appelées convolutions. Contrairement aux réseaux de neurones classiques, qui traitent chaque pixel de manière indépendante, les CNN sont composés de couches de convolution, chacune prenant en compte la structure spatiale des images par l'utilisation de filtres de convolution permettant ainsi la détection de motifs visuels. En considérant une image $I \in \mathbb{R}^{n \times n}$ et un filtre $k \in \mathbb{R}^{a \times b}$, une convolution au pixel de coordonnées (x, y) de l'image I par le filtre k est définie de la manière suivante :

$$(I \star k)(x, y) = \sum_{i=0}^{a} \sum_{j=0}^{b} I(x - \frac{1}{2}(a-1) + j, y - \frac{1}{2}(b-1) + i)k(i, j). \tag{1.2}$$

Ainsi, pour introduire le fonctionnement d'une couche de convolution, nous considérons un ensemble de c_l filtres constituant la couche de convolution l d'un CNN. Cet ensemble de filtre est noté $\{k_c^l\}_{c \in [1, c_l]}$. Sachant une entrée notée $I^{l-1} \in \mathbb{R}^{n \times n \times c_{l-1}}$ constituée de c_{l-1} canaux, l'application de la couche convolutionnelle permet l'obtention d'une sortie composée de c_l canaux, obtenu par l'application de chacun des c_l filtres. L'opération effectuée par un filtre k_c^l est définie par :

$$I_c^l = \sum_{i=1}^{c_{l-1}} I_i^{l-1} \star k_c^l + b_c^l,$$

avec $\star$ représentant l'opération de convolution 2D décrite dans l'équation (1.2) appliquée sur tout pixel de l'image I_i^{l-1} et b_c^l représente le biais associé au filtre k_c^l. La Figure 1.3 illustre l'application d'une couche convolutionnelle de c_l filtres sur une entrée constituée de c_{l-1} canaux. L'apprentissage d'une telle couche correspond à l'optimisation des paramètres constituant les filtres selon la tâche souhaitée.

De plus, grâce à la superposition de plusieurs couches convolutionnels, les CNN sont capables d'apprendre des représentations hiérarchiques à différent niveau de précision des images. C'est grâce à ces différents niveaux de caractéristiques que les CNN sont des réseaux de neurones efficaces et pertinents.

1.1.3 Les réseaux de neurones sur graphes

Les réseaux de neurones sur graphes (GNN) sont une classe de modèles d'apprentissage machine conçus pour traiter des données structurées sous forme de graphes.

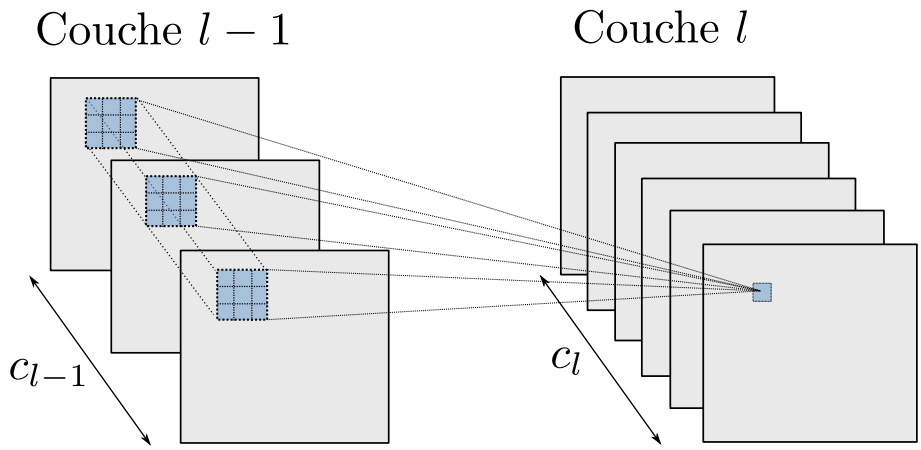


FIGURE 1.3 – Illustration du fonctionnement d'une couche convolutionnelle de CNN composée de c_l filtres sur une entrée de c_{l-1} canaux.

Alors que les modèles de réseaux de neurones précédents se concentrent sur des données vectorielles, séquentielles et d'images, les GNN sont spécialement conçus pour capturer les relations discrètes et complexes présentes dans les données de graphes. Les données de graphes sont omniprésentes dans de nombreux domaines tels que la chimie, la biologie, les réseaux sociaux ou les systèmes de transports et peuvent être représentées de différentes manières.

Représentations de graphe

Les données de graphes sont représentées sous la forme de nœuds (aussi appelées sommets ou vertex) reliés par des arêtes (également connues sous le nom de liens ou liaisons), qui capturent les relations entre les entités du graphe. Une représentation de graphe générale et applicable à tous les types de graphe peut être définie pour un graphe G tel que $G = (\mathcal{V}, \mathcal{E})$ où $\mathcal{V}$ représente un ensemble de nœuds de taille $|\mathcal{V}| = n$ et $\mathcal{E}$ représente un ensemble d'arêtes. Celui-ci est constitué par les arêtes existantes entre deux nœuds et peut être défini tel que $\mathcal{E} = \{(u_i, u_j)|u_i \in \mathcal{V}, u_j \in \mathcal{V}, \text{si une arête est présente entre } u_i \text{ et } u_j\}$.

Une seconde représentation, couramment utilisée lorsqu'il est nécessaire d'obtenir une représentation numérique du graphe, se compose d'une paire de matrice de caractéristiques et de tenseur d'adjacence, notée $G = (\mathbf{X}, \mathbf{A})$. Dans ce cas, en considérant des nœuds caractérisés sur d dimensions, la représentation de ces derniers est effectuée par la matrice de caractéristiques $\mathbf{X} \in \mathbb{R}^{n \times d}$ où chaque ligne correspond aux caractéristiques d'un nœud. Le tenseur d'adjacence $\mathbf{A}$ est défini avec $\mathbf{A} \in \mathbb{R}^{n \times n \times e}$ où e correspond au nombre de dimensions permettant la représentation des caractéristiques d'une arête. Celui-ci permet la représentation des arêtes d'un graphe en définissant $\mathbf{A}[i, j] = \mathbf{e_{i,j}}$ avec $\mathbf{e_{i,j}} \in \mathbb{R}^e$ représentant les caractéristiques de l'arête reliant le nœud u_i au nœud u_j. Lorsque l'on considère des caractéristiques d'arêtes d'appartenance à une classe, par exemple avec des graphes moléculaires où les liaisons peuvent être simples, doubles, triples ou inexistantes, il est courant d'utiliser l'encodage *one-hot*. Cet encodage défini $\mathbf{e_{i,j}} \in \{0, 1\}^e$ avec e égale au nombre de

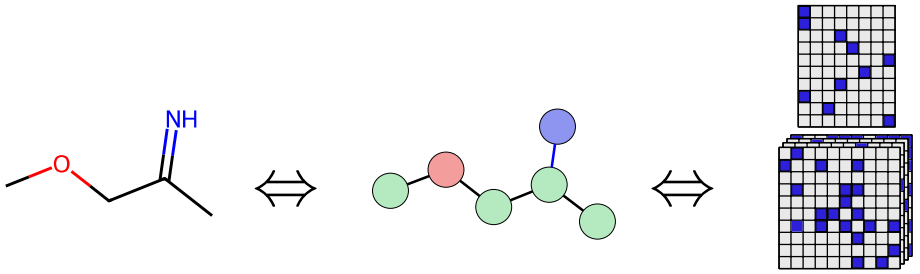


FIGURE 1.4 – Différentes représentations de graphe. (a) Représentation de graphe moléculaire. (b) Représentation de graphe $G = (\mathcal{V}, \mathcal{E})$ sous forme de nœuds $\{u_i \in \mathcal{V} | i \in [1, n]\}$ reliés par des arêtes $\{(u_i, u_j) \in \mathcal{E} | u_i \in \mathcal{V}, u_j \in \mathcal{V}\}$. (c) Représentation sous forme de paire de matrice de caractéristique et de tenseur d'adjacence $(\mathbf{X}, \mathbf{A})$.

classes où chaque position dans le vecteur correspond à une classe spécifique, et cette position est définie à 1 si l'arête appartient à celle-ci, sinon elle est définie à 0. Bien que cette méthode de représentation des graphes permette l'encodage des caractéristiques d'un graphe ainsi que ses dépendances structurelles, il convient de noter qu'elle présente certains inconvénients. En effet, le coût en mémoire d'un graphe peut être important, notamment lorsqu'il est de grande taille. De plus, comme cette représentation dépend de l'ordre des nœuds choisi pour son encodage dans ses matrices, elle ne permet pas de capturer une propriété importante des graphes, à savoir l'invariance aux permutations des nœuds.

En fonction du type de graphe, différentes représentations peuvent être utilisées. La Figure 1.4 présente trois exemples de représentations de graphe différentes. La première illustre une visualisation spécifique aux graphes moléculaires, tandis que les deux suivantes illustrent les deux types de représentations génériques de graphe mentionnées précédemment. Bien que notre attention se porte principalement sur celles-ci, il est important de noter qu'il existe une grande variété d'autres types de représentations de graphe. Par exemple, pour les graphes moléculaires, nous pouvons mentionner les empreintes digitales, les notations SMILES, les potentiels, les matrices de Coulomb, les sacs de liaisons, les géométries 3D et les densités électroniques [108].

Modèle de réseaux de neurones

Les GNN exploitent les relations de voisinage présentes dans les graphes pour apprendre des représentations à plusieurs niveaux. Ces représentations capturent les caractéristiques et informations structurelles des graphes et peuvent être définies au niveau d'un nœud ou d'un graphe entier. Ces modèles sont utilisés dans diverses tâches d'apprentissage machine sur les graphes tels que la classification de nœuds, la prédiction d'arête, la segmentation sémantique de graphe et d'autres.

Afin d'utiliser ces données, plusieurs méthodologies de GNN ont été proposées. Basés sur les deux précédentes structures de réseaux de neurones, celles-ci peuvent être classifiées en deux grandes catégories : les réseaux de neurones récurrents de

graphes (RecGNN) et les réseaux de neurones convolutionnels de graphes (ConvGNN).

Réseau de neurones récurrents de graphes (RecGNN). Les RecGNN sont les premiers réseaux de neurones sur graphes proposés [111]. Basés sur le principe des RNN (voir Section 1.1.2), ces modèles s'appuient sur la notion de propagation de l'information entre nœuds voisins dans un graphe. Ces modèles utilisent des mécanismes itératifs pour agréger l'information des nœuds voisins, en tenant compte de la structure du graphe. Ces mécanismes permettent aux GNN de propager l'information à travers le graphe et d'incorporer les caractéristiques des nœuds environnants dans les représentations des nœuds.

Soit $\mathcal{N}_u$ l'ensemble des nœuds voisins du nœud u tel que $\mathcal{N}_u = \{v|v \in \mathcal{V} \text{ et } (u,v) \in \mathcal{E}\}$ et $\mathbf{x}_u$, $\mathbf{x}^e_{(u,v)}$ représentant respectivement les caractéristiques associées au nœud u et à la liaison entre le nœud u et v. Ainsi, le calcul de l'état $\mathbf{h}^{(u)}_t$ d'un nœud u à l'instant t s'effectue en prenant en compte son état et celle de son voisinage à $t-1$ de la manière suivante :

$$\mathbf{h}^{(u)}_t = f(\mathbf{h}^{(u)}_{t-1}, \mathbf{Agg}(\{g(\mathbf{x}_u, \mathbf{x}^e_{(u,v)}, \mathbf{x}_v, \mathbf{h}^{(v)}_{t-1})|v \in \mathcal{N}_u\})),$$

où f représente une fonction pouvant être implémentée par un RNN et **Agg** correspond à une fonction d'agrégation. De plus, g représente la fonction permettant le calcul du message à propager d'un nœud v vers un nœud u. Basés sur ce principe d'information récurrente obtenue par le voisinage de chaque nœud, différentes architectures de RecGNN ont été proposées [111, 36, 77, 25] permettant la mise à jour itérative des représentations de nœud.

Réseau de neurones convolutionnel de graphes (ConvGNN). Une autre approche couramment utilisée pour définir un GNN consiste à adapter les CNN (voir Section 1.1.2) aux données de graphes. Cette approche est similaire aux RecGNN dans le sens où elle permet la mise à jour des états des nœuds du graphe en fonction de leurs voisinages. Cependant, au lieu de mises à jour itératives et récurrentes, elle utilise un nombre fixe de couches avec des paramètres différents pour effectuer cette mise à jour. Les ConvGNN peuvent être séparées en deux sous-catégories. La première définit l'opération de convolution de manière spectrale lorsque la deuxième la définit de manière spatiale.

Les ConvGNN spectrales s'appuient sur les propriétés spectrales des graphes pour effectuer des convolutions. Contrairement aux approches basées sur le voisinage des nœuds, ils exploitent les informations contenues dans les valeurs propres et les vecteurs propres de la matrice d'adjacence du graphe. Les convolutions spectrales sur graphe s'appuient sur les avancées en traitement du signal appliquées aux graphes [119, 109], en particulier sur l'utilisation de la matrice Laplacienne normalisée du graphe définie à partir de la matrice d'adjacence, à savoir

$$\mathbf{L} = \mathbb{I}_n - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}},$$

avec $\mathbb{I}_n$ la matrice identité de taille $n \times n$ et $\mathbf{D}$ la matrice diagonale de degré des nœuds $\mathbf{D}[i,i] = \sum_j \mathbf{A}[i,j]$. De cette matrice diagonale semi-définie positive, il est

possible de décomposer celle-ci de la manière suivante :

$$\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top,$$

avec $\mathbf{V} = (\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_n) \in \mathbb{R}^{n \times n}$ la matrice de vecteurs propres et $\mathbf{\Lambda}$ la matrice diagonale de valeurs propres associées telle que $\mathbf{\Lambda}[i,i] = \lambda_i$. En considérant le signal associé à un graphe $\mathbf{x} \in \mathbb{R}^n$ composé des signaux scalaires provenant des n nœuds du graphe, une convolution du signal de graphe $\mathbf{x}$ peut être définie dans le domaine spectral par l'application de la transformation de Fourier de graphe définie par $\mathbf{V}^\top \mathbf{x}$. Ensuite, le retour dans le domaine spatial est effectué par l'application de la transformation de Fourier de graphe inverse définie par $\mathbf{V}\hat{\mathbf{x}}$, où $\hat{\mathbf{x}}$ est un signal dans le domaine spectral obtenu à partir de la transformation de Fourier. Ainsi, la convolution spectrale d'un graphe avec un filtre $k_\theta \in \mathbb{R}^n$ est définie de la manière suivante :

$$k_\theta \star \mathbf{x} = \mathbf{V} k_\theta \mathbf{V}^\top \mathbf{x}\,.$$

Cette approche présente l'avantage de pouvoir capturer des informations globales sur la structure du graphe, ce qui la rend particulièrement adaptée aux tâches nécessitant une prise en compte des interactions à longue distance entre les nœuds. Cependant, elle peut être plus sensible aux perturbations locales dans le graphe et moins efficace pour modéliser des graphes de grande taille. Sur la base de cette définition, de nombreuses approches ont été proposées [14, 29, 73, 63, 75].

Similairement à l'opération de convolution dans un CNN classique sur une image, les convolutions spatiales de graphes sont définies en se basant sur les relations spatiales entre les nœuds. En effet, nous pouvons considérer les images comme une forme spécifique de graphe où chaque pixel représente un nœud et où chaque nœud est relié directement à ses pixels voisins. Lorsqu'un filtre de convolution est appliqué sur une image, il calcule une moyenne pondérée des valeurs du pixel central et de ses voisins dans chaque canal. De la même manière, les convolutions spatiales sur graphe fonctionnent en convoluant la représentation du nœud central avec les représentations de ses voisins, permettant ainsi de mettre à jour la représentation du nœud central.

La première approche de convolution spatiale sur graphe a été proposée par [87], définissant une couche de convolution spatiale en faisant la somme des informations des nœuds voisins de façon à obtenir la transformation suivante :

$$\mathbf{h}_l^{(u)} = \sigma_l(\mathbf{W}_r^{l^\top} \mathbf{x}_u + \sum_{v \in \mathcal{N}_u} \mathbf{W}^{l^\top} \mathbf{h}_v^{l-1}),$$

où σ_l désigne une fonction d'activation, $\mathbf{W}_r^l$ la matrice de poids appliquée aux caractéristiques initiales du nœud u et $\mathbf{W}^l$ la matrice de poids appliquée aux états précédents des nœuds voisins avec $\mathbf{h}_u^0 = \mathbf{0}$. Aussi, cette formulation peut être écrite sous forme matricielle avec :

$$\mathbf{H}_l = \sigma_l(\mathbf{X}\mathbf{W}_r^l + \mathbf{A}\mathbf{H}^{l-1}\mathbf{W}^l).$$

Afin d'apporter une capacité de mémorisation, l'approche [87] ajoute à cette définition l'utilisation de connexions résiduelles et directes.

De nombreuses approches proposent des méthodologies d'utilisation et d'amélioration de ConvGNN spatial, les articles [4, 38, 44, 93, 112] en sont des exemples. Lorsque l'opération de somme est fréquemment utilisée pour agréger les informations du voisinage, conférant aux GNN une invariance aux permutations des nœuds, certains autres modèles adoptent une approche différente en proposant des architectures équivariantes aux permutations. Cela signifie que lorsque les données d'entrée sont permutées, les sorties du modèle sont également permutées de la même manière. EGNN [110] en est un exemple proposant l'utilisation des coordonnées de nœuds permettant une équivariance aux transformations telles que les permutations.

1.2 Le problème de pré-image et sa résolution

Considérons un espace de caractéristiques généré par une transformation appliquée à des données observées. Le problème de pré-image consiste à trouver une donnée qui, une fois transformée dans l'espace des caractéristiques, correspond à un point d'intérêt. Ce point peut être obtenu par échantillonnage dans l'espace des caractéristiques ou par calcul mathématique, par exemple en utilisant une projection par analyse en composantes principales (ACP). En général, les méthodes de ML cherchent à obtenir une transformation complexe permettant de générer une représentation des données utile à la résolution d'une tâche spécifique, sans chercher à connaître sa transformation inverse. Cela implique une forte présence du problème de pré-image dans de nombreuses tâches de ML et de reconnaissance des formes (PR), ainsi qu'une complexité importante dans sa résolution. Ce problème se pose dans des tâches telles que le débruitage/la compression de signaux ou d'images, la factorisation et complétion de matrices [137, 34], l'apprentissage de dictionnaires parcimonieux [107] et l'analyse de séries temporelles interprétables [125]. Il se produit également dans les réseaux de neurones profonds, tels que l'annotation d'images médicales [65], la fusion de données multimodales [103], et la génération d'exemples contrefactuels [91, 124].

1.2.1 Le problème de pré-image

Soit $\mathcal{X}$ l'espace des observations, à savoir $\mathcal{X} \subset \mathbb{R}^D$ pour un espace à D dimensions. Pour définir un modèle non-linéaire, les méthodes de ML transforment souvent les données dans un espace de caractéristiques $\mathcal{H}$ avant d'appliquer les opérations conventionnelles de PR et d'exploration des données. Soit $\Phi\colon \mathcal{X} \to \mathcal{H}$ cette transformation non-linéaire, transposant les échantillons dans un certain espace de représentation $\mathcal{H}$, doté d'un produit scalaire $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. L'extracteur de caractéristiques peut être obtenu à l'aide d'un réseau de neurones profond ou induit implicitement dans des machines à noyau (Section 1.1). Dans ce dernier cas, un noyau défini positif $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ définit une généralisation du produit scalaire et permet d'effectuer implicitement une transformation non-linéaire. Cela signifie que pour tout $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$, nous avons $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ sans avoir besoin d'expliciter la fonction Φ. Le noyau κ permet ainsi de calculer efficacement les similarités entre les données dans l'espace de caractéristiques $\mathcal{H}$, sans nécessiter une connaissance directe de la transformation Φ.

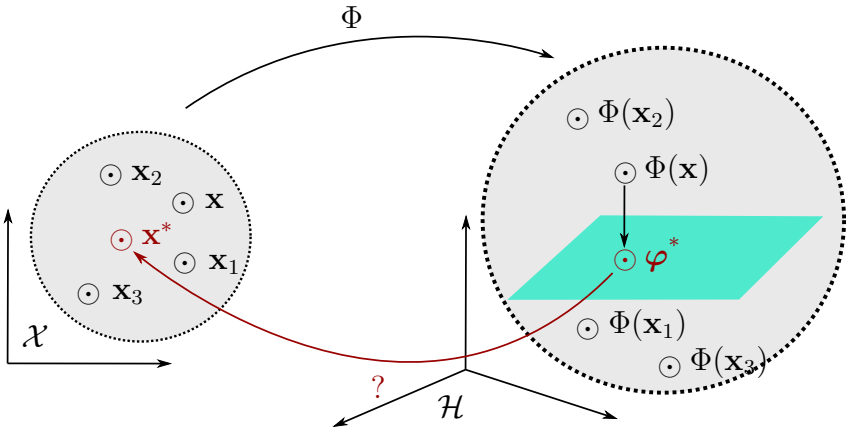


FIGURE 1.5 – Illustration du problème de pré-image, qui consiste à trouver $\mathbf{x}^* \in \mathcal{X}$ dont l'image dans $\mathcal{H}$ est aussi proche que possible de $\boldsymbol{\varphi}^*$. Le " ?" indique que l'inverse de Φ n'est pas connu en général.

Que ce soit en ML ou en PR, il peut être pertinent de revenir à l'espace des observations pour pouvoir extraire des résultats obtenus dans un espace de caractéristiques et les interpréter. Cependant, l'espace de caractéristiques $\mathcal{H}$ généré par un noyau est souvent de grande dimension, et la fonction de transformation Φ associée peut être implicite. Avec des réseaux de neurones profonds, cette dernière correspond à une cascade de transformations rectifiées non-linéaires, ce qui rend le problème de pré-image insoluble. Ainsi, ce problème peut être défini de la manière suivante :

Définition 1.5 (Problème de pré-image)**.** *Soit $\mathcal{X}$ l'espace des observations, et $\mathcal{H}$ l'espace de caractéristiques résultant de la transformation $\Phi : \mathcal{X} \to \mathcal{H}$. Le problème de pré-image consiste à retrouver à partir d'un point d'intérêt $\boldsymbol{\varphi}^* \in \mathcal{H}$ son antécédent $\mathbf{x}^* \in \mathcal{X}$ dans l'espace des observations, à savoir $\mathbf{x}^* = \Phi^{-1}(\boldsymbol{\varphi}^*)$, où Φ^{-1} représente la transformation inverse inconnue $\Phi^{-1} : \mathcal{H} \to \mathcal{X}$. Ainsi, la transformation par Φ de la pré-image $\mathbf{x}^*$ résulte sur le point d'intérêt $\boldsymbol{\varphi}^*$, c'est-à-dire $\Phi(\mathbf{x}^*) = \boldsymbol{\varphi}^*$.*

Une vue d'ensemble du problème de pré-image est illustrée dans la Figure 1.5, démontrant les trois parties principales du processus : (*i*) l'encodage des données avec une fonction de transformation Φ dans un espace de caractéristiques $\mathcal{H}$, (*ii*) puis des opérations linéaires sont effectuées dans cet espace, telles que les projections sur un plan, produisant un point d'intérêt $\boldsymbol{\varphi}^*$, (*iii*) et enfin la résolution du problème de pré-image afin d'extraire le motif $\mathbf{x}^*$ dans $\mathcal{X}$.

1.2.2 Méthodes de résolution

Dans les méthodes à noyaux aussi bien que dans les réseaux de neurones profonds, le problème de pré-image est un problème mal posé. Cela est dû à plusieurs raisons telles que la haute dimensionnalité de l'espace de caractéristiques nécessaire pour améliorer l'expressivité, l'encodage implicite dans les méthodes à noyaux, ou encore les transformations non-linéaires en cascade effectuées avec les couches ReLU dans les réseaux de neurones profonds. Une résolution exacte, comme décrite dans

la définition 1.5, est alors remplacée par une approximation avec $\mathbf{x} \approx \Phi^{-1}(\boldsymbol{\varphi}^*)$. De manière générale, l'utilisation de méthodes en ML ou PR a été conçue indépendamment de la résolution du problème de pré-image, qui est considérée comme une étape de post-traitement dans le processus. Bien que certaines tentatives aient été faites pour intégrer le problème de pré-image dans le processus d'apprentissage machine, comme dans [137] pour la factorisation non-linéaire en matrices négatives, toutes les méthodes susmentionnées souffrent du problème de pré-image.

Plusieurs algorithmes d'estimation de la pré-image ont été proposés dans la littérature. Bakır et al. [5] ont proposé de transformer le problème de pré-image en un problème de régression afin d'apprendre la transformation inverse de l'espace de caractéristiques à l'espace des observations. Cependant, cette technique repose sur la résolution d'un problème d'optimisation de la régression et donne une approximation de la pré-image qui en dépend. Honeine et Richard [50] ont introduit une résolution directe en fournissant une solution analytique à ce problème. D'autres techniques incluent les techniques de descente de gradient, les techniques basées sur le MDS et l'apprentissage par correspondance conforme. L'étude [51] décrit le problème de pré-image et ses différentes techniques de résolution. Outre les solutions proposées pour les données vectorielles, certaines solutions ont été proposées pour résoudre le problème de pré-image sur des données de graphes [6, 53]. Tous ces travaux démontrent la pertinence de la résolution du problème de pré-image et sa difficulté.

1.3 Les modèles génératifs profonds

Les modèles génératifs profonds [95, 45, 12] constituent des modèles de réseaux de neurones capables de générer de nouvelles données en imitant les caractéristiques statistiques d'un ensemble d'apprentissage. Ces modèles sont couramment utilisés dans des domaines tels que la génération de données vectorielles, séquentielles, d'images, de graphes, et bien d'autres. Ils reposent sur des réseaux de neurones profonds et peuvent être classés en plusieurs catégories, telles que les modèles auto-régressifs (AR) [42], les autoencodeurs variationnels (VAE) [58, 59], les réseaux de neurones antagonistes génératifs (GAN) [40, 97] ou les normalizing flows (NF) [64, 98].

Sachant $\mathcal{X}$ l'espace des observations, les modèles génératifs ont généralement pour objectif l'approximation de la densité de probabilité inconnue associée aux données observées $p_{\mathcal{X}}$. Pour cela, ces modèles définissent une distribution de probabilité p_θ paramétrée par θ cherchant à être la plus similaire possible à la densité inconnue à approximer, c'est-à-dire $p_{\mathcal{X}}$. L'apprentissage des modèles génératifs est généralement effectué par la maximisation de la (log-)vraisemblance des données, à savoir

$$\mathbb{E}_{\mathbf{x}\sim p_{\mathcal{X}}}\Big[\log p_\theta(\mathbf{x})\Big].$$

Cela est équivalent à optimiser les paramètres θ en minimisant la distance entre les deux distributions $p_{\mathcal{X}}$ et p_θ telle que :

$$\operatorname*{argmin}_\theta \mathbf{dist}(p_{\mathcal{X}}, p_\theta),$$

avec **dist** une fonction d'évaluation de la distance entre les deux distributions.

Pour définir une telle fonction, la divergence de Kullback-Leibler (KL) est couramment utilisée et permet l'évaluation de la divergence entre deux distributions. Elle peut être définie dans un contexte discret de la manière suivante :

$$D_{KL}(p_{\mathcal{X}} \| p_\theta) = \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{X}}}\Big[\log\left(\frac{p_{\mathcal{X}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right)\Big] = \sum_i p_{\mathcal{X}}(\mathbf{x}_i)\log\left(\frac{p_{\mathcal{X}}(\mathbf{x}_i)}{p_\theta(\mathbf{x}_i)}\right). \tag{1.3}$$

1.3.1 Modèle auto-régressif (AR)

Un modèle AR est une architecture de modèle générative qui permet la génération de données séquentielles en utilisant une approche pas à pas, où chaque élément de la séquence est conditionné par les éléments précédents. Les modèles AR peuvent être définis en utilisant des réseaux de neurones pour capturer la dépendance séquentielle complexe entre les éléments. Parmi ces réseaux de neurones, deux catégories peuvent être définies, à savoir les réseaux de neurones récurrents (RNN) (Section 1.1.2) et les réseaux de neurones convolutionnels (CNN) (Section 1.1.2).

Les modèles AR considèrent des données séquentiellement structurées telles que $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$. Ils sont basés sur la décomposition de la distribution des données $\mathbf{x}$ en considérant qu'une donnée à l'instant t dépend de son état précédent, à savoir

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{D} p_\theta(x_i|x_1, \ldots x_{i-1}).$$

Ainsi, l'apprentissage de ces modèles peut se faire directement par la maximisation de la vraisemblance des données en minimisant la log-vraisemblance négative avec :

$$-\log p_\theta(\mathbf{x}) = -\sum_{i=1}^{D} \log p_\theta(x_i|x_1, \ldots, x_{i-1}).$$

Cette formulation peut être évaluée par le calcul de chaque terme de la somme avec $p_\theta(x_i|x_1, \ldots, x_{i-1})$ une distribution simple dont les paramètres sont obtenus à partir des valeurs de $(x_1, \ldots, x_{i-1})$ en utilisant un réseau de neurones paramétré par θ. Grâce à leurs architectures, les RNN permettent la définition des distributions conditionnelles avec $p_\theta(x_i|x_1, \ldots, x_{i-1}) = g_\theta(x_i)$ où g_θ représente le RNN paramétré par θ (voir Section 1.1.2). Ainsi, plusieurs modèles génératifs AR utilisant des RNN ont été proposés [42, 85, 130].

D'une autre manière, les réseaux de neurones utilisés par les modèles AR peuvent être représentés par des CNN. Dans ce cas, les CNN sont composées de plusieurs couches convolutionnelles empilées (voir Section 1.1.2) modélisant les probabilités conditionnelles. Pour cela, ces couches de convolutions sont dites causales ou masquées, en fonction du type de données, pour prendre en compte le caractère séquentiel des données et que les probabilités conditionnelles ne soient générées qu'à partir du contexte précédent. Des modèles de génération AR basés sur l'utilisation de CNN ont été proposés tels que PixelCNN [129] et WaveNet [131].

Étant donné que ces derniers modèles AR permettent des résultats très satisfaisants sur les images, de nouvelles approches cherchant à les adapter à des données de type graphe ont été proposées. Parmi eux, il existe des modèles génératifs de

graphes basés sur l'utilisation de RecGNN (voir Section 1.1.3) tels que *Deep Generative Model of Graphs* (DeepGMG) [76] et GraphRNN [135]. DeepGMG génère un graphe en générant une séquence de décisions telles que l'ajout d'un nœud/d'une arête et la connexion d'un nœud à un autre. Les actions sont décidées en fonction de l'état des nœuds et de l'état du graphe croissant, mis à jour par un RecGNN. GraphRNN utilise deux différents RNN pour modéliser le processus de génération, l'un agissant au niveau du graphe, l'autre au niveau des arêtes. Le premier RNN ajoute des nœuds à une séquence de nœuds lorsque le second RNN produit une séquence binaire représentant les possibles connexions entre le nouveau nœud et les nœuds précédents.

Bien que les modèles AR offrent des capacités de génération de haute qualité, leur nature séquentielle ne permet pas de génération de données instantanées dès lors que l'on travaille sur des données à hautes dimensions ce qui peut être perçu comme un inconvénient dans certains contextes.

1.3.2 Auto-encodeur variationnel (VAE)

Les VAE, quant à eux, sont des modèles probabilistes qui, basés sur les auto-encodeurs (AE), cherchent à apprendre une représentation latente compressée des données en utilisant un encodeur, et à reconstruire les données originales à partir de cette représentation en utilisant un décodeur. Ils permettent ainsi de générer de nouvelles instances en échantillonnant dans l'espace latent, ce qui offre une grande flexibilité pour explorer et créer des variations de données.

Pour cela, les VAE utilisent les notions d'inférence variationnelle (VI) pour définir leur objectif. En VI, le but est d'obtenir des informations sur une représentation latente $\mathbf{z} \in \mathcal{Z}$ à partir de données $\mathbf{x}$ observées depuis un espace $\mathcal{X}$. Autrement dit, nous souhaitons obtenir la distribution de probabilité a posteriori $p(\mathbf{z} \,|\, \mathbf{x})$ pouvant être définie en appliquant le théorème de Bayes :

$$p(\mathbf{z} \,|\, \mathbf{x}) = \frac{p(\mathbf{x} \,|\, \mathbf{z})p(\mathbf{z})}{p(\mathbf{x})},$$

avec $p(\mathbf{x} \,|\, \mathbf{z})$ représentant la vraisemblance, $p(\mathbf{z})$ la distribution a priori dans $\mathcal{Z}$ et $p(\mathbf{x})$ la distribution dite "marginale" des données souvent complexe et inconnue. Nous pouvons définir $p(\mathbf{x})$ avec :

$$p(\mathbf{x}) = \int_{\mathcal{Z}} p(\mathbf{x}, \mathbf{z}) = \int_{\mathcal{Z}} p(\mathbf{x} \,|\, \mathbf{z})p(\mathbf{z}).$$

Cependant, pour de nombreuses raisons lors de son application à des scénarios réels, ce calcul est la plupart du temps impossible.

Pour palier à cela, l'idée en VI consiste à approximer $p(\mathbf{z} \,|\, \mathbf{x})$ par une fonction $q(\mathbf{z} \,|\, \mathbf{x})$. Ainsi, nous cherchons une distribution optimale $q^\star(\mathbf{z} \,|\, \mathbf{x})$ qui minimise sa distance avec $p(\mathbf{z} \,|\, \mathbf{x})$ en minimisant la divergence de KL entre les 2 distributions :

$$q^\star(\mathbf{z} \,|\, \mathbf{x}) = \underset{q \in Q}{\operatorname{argmin}}\ D_{KL}(q(\mathbf{z} \,|\, \mathbf{x}) \| p(\mathbf{z} \,|\, \mathbf{x})),$$

avec Q un ensemble de distributions paramétriques simples. Cette divergence peut être écrite de la manière suivante :

$$D_{KL}(q(\mathbf{z}\,|\,\mathbf{x})\|p(\mathbf{z}\,|\,\mathbf{x})) = \int_{\mathcal{Z}} q(\mathbf{z}\,|\,\mathbf{x}) \log\left(\frac{q(\mathbf{z}\,|\,\mathbf{x})}{p(\mathbf{z}\,|\,\mathbf{x})}\right).$$

Avec le théorème de Bayes, nous pouvons alors définir $p(\mathbf{z}\,|\,\mathbf{x})$ selon la distribution dite "jointe" $p(\mathbf{z},\mathbf{x})$, avec

$$p(\mathbf{z}\,|\,\mathbf{x}) = \frac{p(\mathbf{z},\mathbf{x})}{p(\mathbf{x})},$$

et développer la divergence de KL de la manière suivante :

$$\begin{aligned}
D_{KL}(q(\mathbf{z}\,|\,\mathbf{x})\|p(\mathbf{z}\,|\,\mathbf{x})) &= \int_{\mathcal{Z}} q(\mathbf{z}\,|\,\mathbf{x}) \log\left(\frac{q(\mathbf{z}\,|\,\mathbf{x})}{p(\mathbf{z}\,|\,\mathbf{x})}\right) \\
&= \int_{\mathcal{Z}} q(\mathbf{z}\,|\,\mathbf{x}) \log\left(\frac{q(\mathbf{z}\,|\,\mathbf{x})p(\mathbf{x})}{p(\mathbf{z},\mathbf{x})}\right) \\
&= \int_{\mathcal{Z}} q(\mathbf{z}\,|\,\mathbf{x}) \log\left(\frac{q(\mathbf{z}\,|\,\mathbf{x})}{p(\mathbf{z},\mathbf{x})}\right) + \int_{\mathcal{Z}} q(\mathbf{z}\,|\,\mathbf{x}) \log p(\mathbf{x}) \\
&= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\Big[\log\left(\frac{q(\mathbf{z}\,|\,\mathbf{x})}{p(\mathbf{z},\mathbf{x})}\right)\Big] + \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\Big[\log p(\mathbf{x})\Big] \\
&= -\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\Big[\log\left(\frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z}\,|\,\mathbf{x})}\right)\Big] + \log p(\mathbf{x}) \\
&= -\mathcal{L}(q) + \log p(\mathbf{x}).
\end{aligned}$$

De cette formulation, nous pouvons définir l'objectif $\mathcal{L}(q)$ appelé borne inférieure de l'évidence (ELBO), où l'évidence est représentée par $\log p(\mathbf{x})$, à maximiser pour que la divergence soit minimisée. Ainsi, l'expression de l'ELBO est définie de la manière suivante :

$$\mathcal{L}(q) = \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}\,|\,\mathbf{x})}\Big[\log\left(\frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z}\,|\,\mathbf{x})}\right)\Big]. \tag{1.4}$$

Les VAE utilisent l'ELBO comme objectif à maximiser pour l'apprentissage du modèle. Afin de décomposer l'objectif pour prendre en compte l'architecture incluant encodeur et décodeur, ces modèles paramètrent la distribution a posteriori avec un encodeur noté $q_\phi(\mathbf{z}\,|\,\mathbf{x})$ et la distribution de vraisemblance par un décodeur noté $p_\theta(\mathbf{x}\,|\,\mathbf{z})$. La fonction de l'ELBO peut alors être réarrangée en appliquant le théorème de Bayes de la manière suivante :

$$\begin{aligned}
\mathcal{L}(\phi,\theta) &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}\,|\,\mathbf{x})}\Big[\log\left(\frac{q_\phi(\mathbf{z}\,|\,\mathbf{x})}{p_\theta(\mathbf{x}\,|\,\mathbf{z})p_\theta(\mathbf{z})}\right)\Big] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}\,|\,\mathbf{x})}\Big[\log p_\theta(\mathbf{x}\,|\,\mathbf{z})\Big] - D_{KL}(q_\phi(\mathbf{z}\,|\,\mathbf{x})\|p_\theta(\mathbf{z})).
\end{aligned} \tag{1.5}$$

Cette fonction objective est composée de deux termes différents pouvant être intuitivement interprétés. Le premier terme représente la perte due à la reconstruction obtenue avec le décodeur p_θ sachant la représentation $\mathbf{z} \in \mathcal{Z}$. Le deuxième terme correspond à une perte de régularisation due à l'utilisation d'une distribution résultant

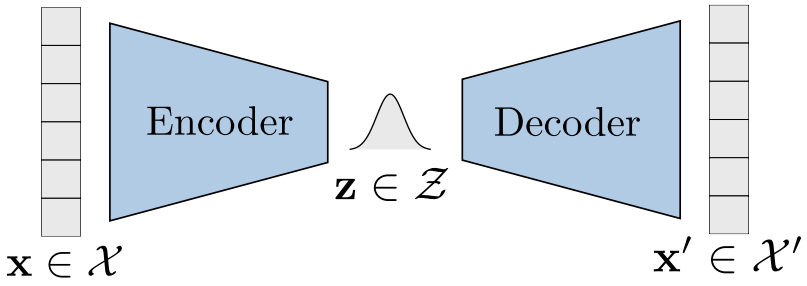


FIGURE 1.6 – Illustration de l'architecture d'un auto-encoder variationnel (VAE).

de l'encodeur q_ϕ sachant une donnée $\mathbf{x} \in \mathcal{X}$ par rapport à la véritable distribution a priori $p_\theta(\mathbf{z})$. Généralement, la distribution a priori est choisie simple telle une distribution gaussienne, permettant ainsi à l'encodeur de générer les paramètres qui la définit, à savoir une moyenne $\boldsymbol{\mu}(\mathbf{x})$ et un écart-type $\boldsymbol{\sigma}(\mathbf{x})$ permettant l'échantillonnage d'une représentation par $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{x}), \boldsymbol{\sigma}(\mathbf{x}))$.

La Figure 1.6 illustre l'architecture d'un VAE composé d'un encodeur générant une distribution dans laquelle $\mathbf{z}$ peut être échantillonné à partir d'une donnée observée $\mathbf{x} \in \mathcal{X}$ et d'un décodeur qui, à partir d'une représentation $\mathbf{z} \in \mathcal{Z}$, génère une donnée $\mathbf{x}'$ cherchant à être le plus proche possible de la donnée originale $\mathbf{x}$.

Un des avantages des VAE est que, en plus d'être un modèle génératif, ils permettent l'apprentissage de représentation. Cette génération d'un espace de représentation $\mathcal{Z}$ permet une meilleure interprétabilité des données et un possible contrôle dans la génération de données [59]. Cependant, certains désavantages lors de l'utilisation de VAE sont identifiables. En particulier, les VAE ont tendance à générer des données floues et donc non-réalistes. De plus, certains problèmes lors de l'optimisation de ces modèles peuvent apparaître, notamment un effondrement de la distribution a posteriori (*posterior collapse*) faisant référence aux distributions des représentations latentes concentrées, voire uniformes, entraînant une perte de la variabilité des données. Des analyses et résolutions de ces problèmes ont été proposées [1, 24].

VGAE [62] applique une architecture de VAE sur des données de type graphe dans lequel l'encodeur est implémenté par un ConvGNN (voir Section 1.1.3). Étant donné que VGAE est utilisé à des fins de prédictions de liens entre nœuds, ce modèle utilise une mesure de similarité pour définir le décodeur de VGAE permettant la génération d'une matrice d'adjacence. GraphVAE* [120], quant à lui, vise à générer un graphe composé de ses caractéristiques de nœuds et d'arêtes en plus sa matrice d'adjacence en utilisant un MLP comme décodeur. L'ajout d'une régularisation sur les générations d'un VAE sur graphe est proposé par RGVAE [81]. JT-VAE [55] est un modèle qui exploite l'architecture VAE pour encoder des graphes moléculaires en les décomposant en graphes et arbres de jonction. L'objectif du modèle est de générer une représentation continue des molécules à l'aide d'un encodeur, puis de générer des graphes moléculaires améliorés à l'aide d'un décodeur à partir de représentation continue optimisée. De plus, les auteurs proposent la génération de molécule par l'ajout progressif de composants/groupes valides d'un point de vue moléculaire. Cela est d'autant plus important que la génération de graphes, atome par atome, oblige

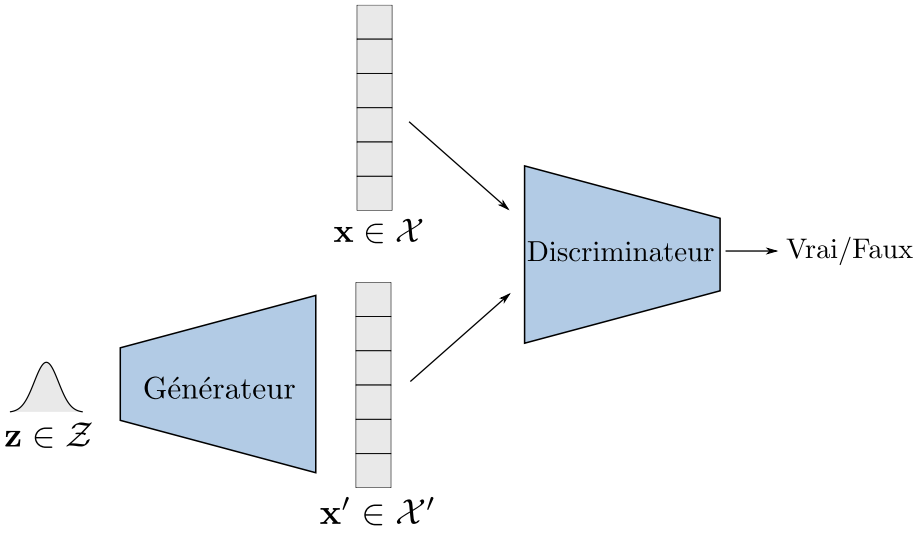


FIGURE 1.7 – Illustration de l'architecture d'un réseau antagoniste génératif (GAN).

le modèle à générer des molécules non valides à des étapes intermédiaires.

1.3.3 Réseaux antagonistes génératifs (GAN)

Les GAN, introduits par [40], sont des modèles génératifs puissants utilisant des architectures dites "antagonistes". Ils permettent l'entraînement de modèles génératifs sans minimiser une fonction de coût explicite, mais en optimisant un jeu min-max entre un générateur et un discriminateur. Dans ce jeu d'optimisation, le générateur, noté G, tente de produire des échantillons de plus en plus réalistes, tandis que le discriminateur, noté D, essaie de les distinguer des vrais échantillons.

Pour cela, ces modèles visent à optimiser l'objectif suivant :

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\mathcal{X}}}\Big[\log(D(\mathbf{x}))\Big] + \mathbb{E}_{\mathbf{z}\sim p_{\mathcal{Z}}}\Big[\log(1 - D(G(\mathbf{z})))\Big] \tag{1.6}$$

Cet objectif est décomposé en deux termes associés à la réponse du discriminateur selon la donnée, réelle ou générée. Ainsi, le discriminateur cherche à maximiser la probabilité $D(\mathbf{x})$ sachant une donnée réelle $\mathbf{x} \sim p_{\mathcal{X}}$ tout en maximisant $1 - D(G(\mathbf{z}))$ sachant une donnée générée $G(\mathbf{z})$ avec $\mathbf{z} \sim p_{\mathcal{Z}}$. Contrairement au discriminateur, le rôle du générateur est de tromper le discriminateur, ainsi le générateur cherche à minimiser ces deux termes. La Figure 1.7 présente l'architecture d'un GAN dans son ensemble.

Cette compétition favorise l'apprentissage de caractéristiques précises et aboutit à des générateurs capables de créer des données de haute qualité visuelle. Cependant, l'apprentissage d'un GAN est confronté à plusieurs problèmes connues d'instabilités lors de l'apprentissage [132]. Un premier correspond au problème de gradient qui disparaît (ou *vanishing gradient*) causé lorsque le discriminateur est trop performant par rapport au générateur impliquant un manque de gradient pour l'optimisation du générateur. Un deuxième problème auquel sont confrontés les GAN est l'effondrement des modes (ou *mode collapse*) correspondant à un générateur

bloqué et produisant des données très faiblement variées permettant de tromper le discriminateur sans couvrir la totalité de la distribution souhaitée. Pour réduire ces problèmes d'instabilités d'optimisation, des solutions ont été proposées [2, 88]. Parmi ces derniers, le WGAN [2] utilise la distance de Wasserstein permettant une meilleure convergence et stabilité de l'entraînement du GAN.

Plusieurs approches visant à l'application des GAN sur des données de type graphe ont été proposées. Inspiré par l'approche des GAN, ARVGA [96] propose d'entraîner l'encodeur d'un VAE sur graphe par compétition à l'aide d'un réseau discriminant qui cherche à déterminer si une représentation est échantillonnée à partir de la distribution encodée ou à partir d'une distribution préalable. MolGAN [27] est un modèle qui combine ConvGNN, GAN et des objectifs d'apprentissage par renforcement pour générer des graphes moléculaires présentant certaines propriétés. Lorsque le générateur cherche à produire un faux graphe avec sa matrice de caractéristiques, le discriminateur, quant à lui, vise à distinguer les faux échantillons des données réelles. En plus de cela, l'apprentissage par renforcement est intégré par l'utilisation d'un réseau de récompense en parallèle avec le discriminateur pour encourager les graphes générés à avoir certaines propriétés. NetGAN [11] est un autre modèle inspiré par les GAN. Il combine LSTM avec WGAN pour générer des graphes à partir de marche aléatoires. Le générateur de NetGAN est entraîné pour produire des marches aléatoires crédibles en utilisant un LSTM, tandis que le discriminateur est utilisé pour distinguer les marches aléatoires générées des vraies. Une fois le modèle entraîné, il est possible de dériver un nouveau graphe en utilisant les marches aléatoires produites par le générateur pour calculer une matrice de cooccurrence. GCPN [134] est une architecture de génération de graphes moléculaires itérative utilisant ConvGNN et combinant GAN et apprentissage par renforcement. Le modèle est entraîné pour optimiser une perte de GAN et des récompenses spécifiques au domaine. L'apprentissage par renforcement est utilisé ici afin d'explorer les molécules suivant des propriétés souhaitées n'appartenant pas aux jeux de données utilisés.

1.3.4 Normalizing Flow (NF)

Les NF sont des méthodes génératives basées sur la composition de fonctions bijectives inversibles, permettant des relations exactes entre deux distributions, la distribution correspondante aux données observées et une distribution cible générée. Ces modèles modélisent des transformations inversibles entre les deux espaces associés à chaque distribution par l'application des bijections composant le NF.

Soient $p_{\mathcal{X}}$ une distribution complexe et inconnue associées aux données de l'espace des observations $\mathcal{X}$, et $p_{\mathcal{Z}}$ une distribution appartenant à l'espace de caractéristiques $\mathcal{Z}$. Un NF [64] est un modèle génératif décrivant une transformation $\Phi : \mathcal{X} \to \mathcal{Z}$ inversible entre deux distributions de données. Ainsi, la définition d'une distribution gaussienne $p_{\mathcal{Z}}$ permet l'apprentissage d'une transformation inversible de l'espace des données $\mathcal{X}$ vers un espace latent $\mathcal{Z}$ où les représentations des données appartiennent à la distribution gaussienne $p_{\mathcal{Z}}$. Cette approche est illustrée dans la Figure 1.8. La génération de données est alors définie par l'application de la transformation inverse $\Phi^{-1} : \mathcal{Z} \to \mathcal{X}$ sur des éléments $\mathbf{z} \in \mathcal{Z}$, tel que $\mathbf{x} = \Phi^{-1}(\mathbf{z})$.

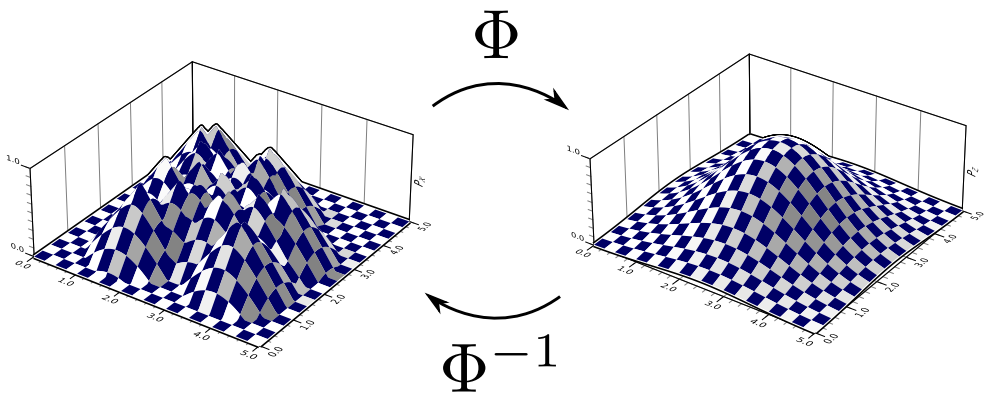


FIGURE 1.8 – Illustration de l'approche appliquée par un Normalizing Flow (NF).

Pour ce faire, l'idée constituant les NF s'appuie sur la formule de changement de variable définissant la relation exacte entre les deux densités de probabilité, à savoir

$$\begin{aligned} p_{\mathcal{X}}(\mathbf{x}) &= p_{\mathcal{Z}}(\mathbf{z}) \left| \det \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right) \right| \\ &= p_{\mathcal{Z}}(\Phi(\mathbf{x})) \left| \det \left(\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}} \right) \right|, \end{aligned} \tag{1.7}$$

où $\frac{\partial \Phi(\mathbf{x})}{\partial x}$ défini la jacobienne de Φ et dont le déterminant représente le coefficient de déformation appliquée entre les distributions.

Contrairement aux VAE, qui optimisent selon une borne inférieure, l'équation (1.7) permet une optimisation selon l'exacte vraisemblance. Ainsi, avec $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \in \mathcal{X}$, l'ensemble des données d'observations, l'entraînement d'un NF consiste à maximiser la log-vraisemblance, obtenue à partir de l'équation (1.7), définie de la manière suivante :

$$\log p_{\mathcal{X}}(\mathbf{x}) = \sum_{i=1}^{N} \log p_{\mathcal{Z}}(\Phi(\mathbf{x}_i)) + \log \left| \det \left(\frac{\partial \Phi(\mathbf{x}_i)}{\partial \mathbf{x}_i} \right) \right|. \tag{1.8}$$

Cependant, afin de pouvoir optimiser les paramètres d'un modèle Φ selon une telle formulation, certaines contraintes doivent être respectées. En effet, Φ doit définir une fonction bijective, facilement calculable et dont le calcul du déterminant associé doit être simple à déterminer. Ce n'est pas le cas lors de l'application de méthodes non-linéaires complexes sans restriction.

Afin d'améliorer l'expressivité du modèle, les NF définissent leurs transformations Φ par une composition de fonctions bijectives permettant ainsi la construction d'une fonction complexe Φ par l'association de multiples fonctions bijectives, à savoir

$$\Phi = \Phi_\ell \circ \Phi_{\ell-1} \circ \cdots \circ \Phi_2 \circ \Phi_1,$$

avec ℓ le nombre de fonctions bijectives constituant Φ. De plus, le déterminant de la jacobienne de Φ peut être calculé simplement par le produit des déterminants des

fonctions composantes,

$$\det\left(\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}}\right) = \prod_{i=1}^{\ell} \det\left(\frac{\partial \Phi_i(\mathbf{x})}{\partial \mathbf{x}_i}\right).$$

Ces dernières observations ont ainsi fait émerger plusieurs stratégies pour définir de telles transformations [64].

NF par couches de couplage

La méthodologie pour définir une architecture de NF la plus connue, proposée par NICE [32] et RealNVP [33], vise à simplifier le calcul du déterminant de la jacobienne de Φ par l'obtention d'une matrice jacobienne triangulaire. Pour cela, les bijections constituantes de Φ sont définies avec des couches de couplage. Celles-ci consistent en la division de l'entrée $\mathbf{x} \in \mathbb{R}^D$ en deux parties $(\mathbf{x}_A, \mathbf{x}_B) \in \mathbb{R}^d \times \mathbb{R}^{D-d}$, puis en appliquant des transformations différentes pour chacune d'entre elles. Ainsi, avec Φ_i, une couche de couplage, nous obtenons la sortie $\mathbf{z}$ telle que $\mathbf{z} = \Phi_i(\mathbf{x})$, de la manière suivante :

$$\begin{aligned}\mathbf{z}_A &= \mathbf{x}_A \\ \mathbf{z}_B &= h(\mathbf{x}_B; \Theta(\mathbf{x}_A))\end{aligned}$$

où $h(\cdot; \Theta(\mathbf{x}_A))$ correspond à une fonction bijective utilisant des paramètres définis par une fonction quelconque $\Theta(\cdot)$ prenant uniquement $\mathbf{x}_A$ comme entrée. De plus, la fonction $\Theta(\cdot)$ peut définir une transformation arbitrairement complexe, permettant ainsi une meilleure expressivité du modèle. Sachant h inversible, l'inverse d'une telle expression est simplement défini par :

$$\begin{aligned}\mathbf{x}_A &= \mathbf{z}_A \\ \mathbf{x}_B &= h^{-1}(\mathbf{z}_B; \Theta(\mathbf{x}_A))\end{aligned}$$

Ainsi, l'utilisation de la fonction identité, dans la transformation de la première partie de l'entrée $\mathbf{x}_A$, permet l'obtention d'une matrice jacobienne triangulaire par bloc composée de la matrice identité et de la jacobienne de h sur la diagonale :

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbb{I}_d & \mathbf{0} \\ \frac{\partial h(\mathbf{x}_B)}{\partial \mathbf{x}_A} & \frac{\partial h(\mathbf{x}_B)}{\partial \mathbf{x}_B} \end{bmatrix}$$

Ceci permettant l'obtention du déterminant d'une telle couche en calculant le déterminant de la jacobienne de h, c'est-à-dire : $\det(\frac{\partial h(\mathbf{x}_B)}{\partial \mathbf{x}_B})$. De plus, afin de faciliter le calcul du déterminant, h est souvent définie comme une transformation élément par élément permettant ainsi l'obtention d'une matrice jacobienne diagonale dont le déterminant peut être calculé simplement par le produit de la diagonale.

Couche de couplage affine. Plus spécifiquement, les couches appelées couches de couplage affine, utilisées dans de nombreuses approches de NF [33, 60], définissent la transformation h de la façon suivante :

$$h(\mathbf{x}_B) = \mathbf{x}_B \odot \exp(s(\mathbf{x}_A)) + t(\mathbf{x}_A),$$

avec $\odot$ le produit élément par élément et $s(\cdot)$ et $t(\cdot)$ représentant des fonctions de $\mathbb{R}^d \to \mathbb{R}^{D-d}$. L'utilisation d'une telle transformation h induit une matrice jacobienne de la forme

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \begin{bmatrix} \mathbb{I}_d & 0 \\ \frac{\partial h(\mathbf{x}_B)}{\partial \mathbf{x}_A} & \text{diag}(\exp(s(\mathbf{x}_A))) \end{bmatrix},$$

permettant un calcul efficient du déterminant correspondant à la transformation appliquée par la couche de couplage de la manière suivante :

$$\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right) = \exp(\sum_{i=1}^{D-d} s(\mathbf{x}_A)_i).$$

Ainsi, le calcul ne nécessitant pas la détermination de matrice jacobienne, s et t peuvent potentiellement définir des transformations complexes et non-linéaires permettant l'utilisation de réseau de neurones. De plus, l'inverse de la fonction h peut être simplement défini de la manière suivante :

$$h^{-1}(\mathbf{z}_B) = (\mathbf{z}_B - t(\mathbf{x}_A)) \oslash \exp(s(\mathbf{x}_A))$$

avec $\oslash$ la division élément par élément.

Parmi les modèles qui utilisent des couches de couplage affines, l'un des plus connus est Glow [60] travaillant avec des données de type image. Celui-ci étend l'approche utilisée par RealNVP [33] en intégrant une architecture multi-échelle, une couche de normalisation d'activation et des couches de convolution inversibles 1×1.

L'application de cette approche à des données de type graphe, appelé GraphNF, a été proposée par [78, 83, 48, 136], générant des graphes en une seule fois. La première tentative de conception d'un réseau de neurones sur graphes utilisant des structures NF a été proposée avec GNF [78], où les caractéristiques des nœuds sont mises à jour à l'aide d'une transformation réversible de passage de messages basée sur des couches de couplage. Introduite dans la Section 1.1.3, une des représentations de graphe couramment utilisées considère une paire de matrice de caractéristiques de nœuds avec un tenseur d'adjacence. GraphNVP [83] propose une génération directe (non séquentielle) de graphe moléculaire en utilisant ce type de représentation de graphe. De même, MoFlow [136] propose une stratégie pour améliorer la génération de molécules en introduisant une nouvelle architecture GraphNF. De la même manière que GraphNVP, MoFlow est basé sur l'utilisation de couches de couplage affine à la fois pour la matrice de caractéristiques des nœuds $\mathbf{X}$ et pour le tenseur d'adjacence $\mathbf{A}$. Ce modèle implique une version modifiée de Glow pour modéliser les liaisons et un nouveau flux conditionnel de graphe pour modéliser les atomes compte tenu des liaisons à l'aide de modèles R-GCN [112]. De plus, pour assurer la validité des molécules générées, une étape de correction post-hoc est appliquée.

MolGrow [70] est un NF hiérarchique, qui génère des graphes moléculaires de manière itérative, constitué de plusieurs niveaux de représentation latente. Chaque niveau est composé de transformations linéaires et de blocs qui contiennent d'autres transformations linéaires ainsi que des couches de RealNVP. Le processus de génération de ce modèle consiste à diviser récursivement un graphe initial constitué d'un seul nœud en deux parties distinctes.

NF auto-régressifs

Inspiré des modèles AR, les NF auto-régressifs proposés par [61] considèrent des données observées séquentielles telles que $\mathbf{x} = \{x_1, x_2, \ldots, x_D\}$. Sachant $\Phi : \mathbb{R}^D \to \mathbb{R}^D$ un modèle de NF auto-régressif, l'application de ce modèle à une donnée $\mathbf{x}$ produit une sortie $\mathbf{z}$ telle que $z = \Phi(\mathbf{x})$. Pour cela, la fonction Φ génère chaque valeur $\mathbf{z}$ en appliquant une fonction bijective h qui dépend des valeurs précédentes de la donnée observée dans le temps. Cela signifie qu'à l'instant t, la valeur $z_t \in \mathbf{z}$ est obtenue en utilisant l'expression suivante :

$$z_t = h(x_t; \theta_t(\mathbf{x}_{1:t-1})), \tag{1.9}$$

avec $\mathbf{x}_{1:t-1} = \{x_1, x_2, \ldots, x_{t-1}\}$ et θ_t les paramètres utilisés par la fonction h et définies avec les valeurs de $\mathbf{x}_{1:t-1}$. De cette manière, chaque sortie z_t ne dépendant que des valeurs $\mathbf{x}_{1:t-1}$, la jacobienne $\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}}$ devient triangulaire, permettant ainsi le calcul du déterminant par le produit des valeurs de la diagonale :

$$\det\left(\frac{\partial \Phi(\mathbf{x})}{\partial \mathbf{x}}\right) = \prod_{i=1}^{D} \frac{\partial h(x_t)}{\partial x_t}.$$

La fonction h étant bijective, son inverse peut alors être défini tel que :

$$x_t = h^{-1}(z_t; \theta_t(\mathbf{x}_{1:t-1})). \tag{1.10}$$

En utilisant l'équation (1.9) et sachant que les données observées $\mathbf{x}$ sont entièrement connues sous toutes leurs dimensions, il est possible de calculer leurs densités $p_{\mathcal{Z}}(\mathbf{x})$ en effectuant un seul passage dans le modèle à l'aide d'un masquage approprié. C'est précisément ce que propose MAF [99]. Cependant, l'inverse, permettant la génération de données en échantillonnant $\mathbf{z} \in \mathcal{Z}$, ne peut être effectuée que de manière séquentielle, car les paramètres de h sont définis en fonction des valeurs de $\mathbf{x}_{1:t-1}$ à générer.

D'une autre manière, IAF [61] propose la définition inverse de NF auto-régressif en calculant les paramètres de h avec les valeurs de sortie $\mathbf{z}_{1:t-1}$, à savoir

$$z_t = h(x_t; \theta_t(\mathbf{z}_{1:t-1})).$$

À l'inverse de MAF, une telle formulation permet la génération de données en un seul passage dans le modèle avec

$$x_t = h^{-1}(z_t; \theta_t(\mathbf{z}_{1:t-1})),$$

mais nécessite un calcul séquentiel de la densité $p_{\mathcal{Z}}(\mathbf{x})$ à partir d'une donnée observée $\mathbf{x}$. Ainsi, lorsque MAF est efficace lors de l'évaluation de la densité dans $\mathcal{Z}$ pour l'apprentissage du modèle, IAF est efficace pour la génération de nouvelles données. L'une de ces deux approches auto-régressives peut ainsi être sélectionnée en fonction du contexte d'application.

Inspiré par ces approches, GraphAF [118] est un NF auto-régressif conçu spécifiquement pour les données de type graphe moléculaire. Tout comme MAF, les auteurs de GraphAF proposent une approche de génération séquentielle des données.

Ils soulignent que pour la génération de molécules, il est avantageux de procéder séquentiellement afin de tirer parti des connaissances spécifiques au domaine chimique et de vérifier la validité de la génération à chaque étape. De plus, le calcul de la densité des données dans $\mathcal{Z}$ étant parallélisable, l'apprentissage du modèle est efficace.

NF continues

L'approche précédente est efficace à condition de se restreindre à une architecture de modèle inversible permettant le calcul de déterminant simple. Les NF continues (CNF) constituent une autre stratégie visant à obtenir Φ par la résolution d'équations différentielles (*ordinary differential equations*, ODE) [19] définies de la manière suivante :

$$\frac{\partial z(\mathbf{x}, t)}{\partial t} = F(z(\mathbf{x}, t), t, \Theta),$$

où $z : \mathbb{R}^d \times t \to \mathbb{R}^d$ définit une trajectoire en fonction du temps $t \in [t_0, t_1]$ avec $z(\mathbf{x}, t_0) = \mathbf{z} \in \mathcal{Z}$ et $z(\mathbf{x}, t_1) = \mathbf{x} \in \mathcal{X}$.

Cette approche s'appuie sur le théorème suivant :

Théorème 1.4 (Changement instantané de variables [19])**.** *Soit $z(\mathbf{x}, t)$ une variable aléatoire continue et finie tirée d'une probabilité $p(z(\mathbf{x}, t))$ dépendant du temps et $\frac{dz(\mathbf{x},t)}{dt} = F(z(\mathbf{x}, t), t)$ l'ODE décrivant une transformation de $z(\mathbf{x}, t)$ continue. Si F est une fonction lipschitzienne continue sur z et t, alors le changement en log-probabilité suit l'équation différentielle*

$$\frac{\partial \log p(z(\mathbf{x}, t))}{\partial t} = -\operatorname{Tr}\left(\frac{\partial F}{\partial z(\mathbf{x}, t)}\right),$$

où Tr *correspond à l'opération de trace.*

De plus, en considérant une équation différentielle Lipschitz continue sur z et t, alors, d'après le théorème de l'existence de Picard [21], la solution au problème de valeur initiale (IVP) est unique. Ainsi, par définition, si le modèle possède des poids finis et des fonctions d'activation non-linéaire Lipschitz telles que tanh et ReLU, alors la transformation est naturellement bijective. Basé sur le théorème du changement instantané de variables, en intégrant sur le temps, de t_0 à t_1, nous pouvons déduire l'expression du changement totale en densité suivante :

$$\log p(z(\mathbf{x}, t_1)) = \log p(z(\mathbf{x}, t_0)) - \int_{t_0}^{t_1} \operatorname{Tr}\left(\frac{\partial F}{\partial z(\mathbf{x}, t)}\right)$$

Considérant une donnée initiale $\mathbf{x} \in \mathcal{X}$ et sachant l'ODE définissant notre problème, nous pouvons définir l'IVP de la manière suivante :

$$\begin{aligned} \begin{bmatrix} \mathbf{z} \\ \log p_{\mathcal{X}}(\mathbf{x}) - \log p_{\mathcal{Z}}(\mathbf{z}) \end{bmatrix} &= \int_{t_1}^{t_0} \begin{bmatrix} F(z(\mathbf{x}, t), t) \\ -\operatorname{Tr}\left(\frac{\partial F}{\partial z(\mathbf{x}, t)}\right) \end{bmatrix} dt, \\ \begin{bmatrix} z(\mathbf{x}, t_1) \\ \log p_{\mathcal{X}}(\mathbf{x}) - \log p_{\mathcal{Z}}(z(\mathbf{x}, t_1)) \end{bmatrix} &= \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix}, \end{aligned} \tag{1.11}$$

où la première partie de l'expression décrit le changement de densité obtenu par application de la trajectoire z et la deuxième représente la condition initiale. Ainsi, la résolution d'un IVP devient réalisable en appliquant un solveur d'ODE [122], permettant ainsi la résolution d'un problème continue sans avoir à le discrétiser. L'intégration de solveurs d'ODE dans la conception de réseaux de neurones continus offre de multiples avantages [19] :

- Ces modèles ne nécessitent pas le stockage de valeurs intermédiaires obtenues lors de l'inférence permettant un coût mémoire constant.
- Grâce à l'utilisation de solveurs d'ODE modernes, il est possible de bénéficier d'une adaptabilité et d'un coût d'évaluation qui varient en fonction de la complexité du problème.
- Cela permet une réduction du nombre de couches constituant ces modèles et, par conséquent, du nombre de paramètres à utiliser.
- Dans le contexte des CNF, le calcul du changement de variable devient plus simple.
- À la différence des RNN (Section 1.1.2), l'utilisation de trajectoire offre la possibilité d'incorporer des données à tout moment t.

Cependant, l'apprentissage d'un tel modèle ne peut être réalisé par une rétropropagation classique. La méthode adjointe introduite par [19], aussi appelée *adjoint sensitivity method*, est une méthode permettant l'apprentissage du modèle et peut être considérée comme l'équivalent continue des méthodes de rétropropagation.

En désignant par θ, les paramètres de la fonction F, la dérivée d'une perte appliquée L par rapport aux paramètres θ prend la forme d'un autre problème de valeur initiale [101] :

$$\frac{dL}{d\theta} = \int_{t_1}^{t_0} \Big(\frac{\partial L}{\partial z(\mathbf{x}, t)}\Big) \frac{\partial F(z(\mathbf{x}, t), t)}{\partial \theta} dt. \tag{1.12}$$

La méthode adjointe consiste aux calculs des gradients par la résolution d'un second ODE en considérant la valeur de perte obtenue $\frac{\partial L}{\partial z(\mathbf{x}, t_1)}$ par rapport à l'état final $z(\mathbf{x}, t_1)$ comme la valeur initiale d'un IVP. Ainsi, l'apprentissage du modèle peut être réalisé en résolvant ce deuxième problème à l'aide d'un autre solveur d'ODE.

Des approches basées sur la résolution d'ODE ont été proposées dans [41, 7]. FFJORD [41] est un CNF introduisant l'utilisation de l'estimateur de l'opérateur de trace de Hutchinson [52], qui facilite le calcul de la trace et permet une extension plus directe aux données de plus grande dimension. De plus, l'application de notion de Transport Optimal (OT) dans la définition de CNF a été développée dans RNODE [35] et OT-Flow [94], permettant la génération de trajectoires minimales et non intersectantes. En outre, OT-Flow calcule la trace de la jacobienne de manière exacte, avec une complexité similaire à celle des estimateurs, ce qui permet une amélioration de la convergence. Cette approche permet une réduction du nombre d'étapes sans compromettre les performances.

E-NF [37] combine l'architecture équivariante aux permutations de graphe des EGNN avec l'approche des CNF pour créer un GraphNF équivariant aux permutations. Cette combinaison permet la préservation de la structure des graphes, en maintenant les symétries et les relations entre les nœuds.

CHAPITRE 2

Résolution du problème de pré-image de noyau par alignement

2.1 Introduction

En reconnaissance de formes et apprentissage machine, les méthodes à noyaux permettent la transformation de données observées vers un espace de dimension supérieure et mieux adapté pour la résolution du problème en question. Cet espace, créé de manière implicite grâce à l'utilisation d'une fonction noyau, est connu sous le nom d'Espace de Hilbert à Noyau Reproduisant (RKHS, pour Reproducing Kernel Hilbert Space, Section 1.1.1).

Le problème de pré-image se pose lorsqu'on souhaite effectuer l'opération inverse de la transformation, c'est-à-dire générer une donnée dans l'espace des observations à partir d'un point de l'espace associé au noyau. Toutefois, la fonction de transformation associée au noyau étant implicite et permettant la génération d'un espace de grande dimension, la transformation inverse est complexe, voire impossible à déterminer avec précision. Les Sections 1.2 et 1.2.2 abordent ce problème ainsi que les différentes solutions qui ont été proposées à cet égard.

Dans ce chapitre, nous présentons une stratégie pour décomposer le problème de pré-image de noyau en deux sous-problèmes distincts. L'idée consiste à transposer le problème présent dans le RKHS (voir Section 1.1.1) généré par un noyau dans un autre espace de caractéristiques pour lequel la transformation inverse est possible. Pour cela, nous proposons tout d'abord l'utilisation d'un modèle génératif (voir Section 1.3) permettant la génération de nouvelles données par l'échantillonnage depuis un espace latent vers l'espace des observations. Nous considérons cet espace latent défini par le modèle comme étant notre espace de caractéristiques. Ensuite, à partir de cet espace, propre à la méthode générative et potentiellement non-similaire au RKHS, nous proposons la génération d'un espace dit "aligné" au RKHS. Ainsi, par la transposition du problème de pré-image du noyau dans l'espace de caractéristiques, nous suggérons l'utilisation de l'espace aligné pour sélectionner une représentation

latente permettant la génération d'une pré-image pertinente pour le problème donné.

Pour définir notre modèle génératif, nous exploitons les avancées récentes dans le domaine des NF (voir Section 1.3.4), possédant une capacité d'inversibilité exacte, afin de proposer une nouvelle approche pour résoudre le problème de pré-image lié aux méthodes à noyaux. Notre méthode repose sur une mise en correspondance entre l'espace généré par le NF et l'espace engendré par le noyau. En combinant le NF avec une méthode à noyau, nous sommes en mesure de fournir une solution au problème de pré-image. Il est important de souligner que notre approche est générique, ce qui signifie que n'importe quelle méthode de NF peut être intégrée au sein de notre méthode. Nous démontrons l'efficacité de notre approche sur le jeu de données MNIST, où nous avons intégré trois modèles de NF de référence utilisant différents formalismes, à savoir Glow [60], FFJORD [41] et OT-Flow [94].

La structure de ce chapitre est la suivante : Notre méthode est décrite dans la Section 2.2, en détaillant le modèle génératif dans la Section 2.2.1, le modèle d'alignement dans la Section 2.2.2, et la procédure d'optimisation dans la Section 2.2.3. Enfin, nous concluons avec l'évaluation de notre approche dans la Section 2.3.

2.2 Approche proposée

Nous considérons l'espace des observations $\mathcal{X}$ avec $\{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N \in \mathcal{X}\}$ l'ensemble des données observables, ainsi que le noyau semi-défini positif $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Ce dernier engendre un RKHS $\mathcal{H}$ généré par une fonction implicite Φ avec $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ et dont le produit scalaire dans $\mathcal{H}$ est défini par $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

Notre objectif est de générer la pré-image $\mathbf{x}^* \in \mathcal{X}$ correspondant à un élément $\boldsymbol{\varphi}^* \in \mathcal{H}$. Une approche naïve consisterait simplement à appliquer l'inverse de la transformation Φ à $\boldsymbol{\varphi}^*$, c'est-à-dire $x^* = \Phi^{-1}(\boldsymbol{\varphi}^*)$. Cependant, lors de l'utilisation de noyau, la fonction de transformation Φ utilisée est généralement implicite. De plus, l'espace $\mathcal{H}$ engendré par le noyau peut avoir une dimension potentiellement très élevée. Par conséquent, résoudre exactement ce problème est généralement difficile, voire impossible [50]. En conséquence, nous proposons une formulation relaxée du problème de pré-image :

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} \| \Phi(\mathbf{x}) - \boldsymbol{\varphi}^* \|_{\mathcal{H}}^2, \tag{2.1}$$

où $\| \cdot \|_{\mathcal{H}}$ désigne la norme dans $\mathcal{H}$.

De cette définition générale, pour trouver une pré-image potentielle, une première approche simple consiste à choisir, parmi l'ensemble d'entraînement, la donnée $\mathbf{x}_d$ qui, après projection dans le RKHS $\mathcal{H}$ du noyau, minimise la distance quadratique selon

$$\mathbf{x}_d = \underset{\mathbf{x} \in \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}}{\operatorname{argmin}} \| \Phi(\mathbf{x}) - \boldsymbol{\varphi}^* \|_{\mathcal{H}}^2. \tag{2.2}$$

En considérant le théorème de représentation 1.3, cette distance à minimiser peut-

être réécrite de la manière suivante :

$$\begin{aligned} d^2_{\mathcal{H}} &= \| \Phi(\mathbf{x}) - \boldsymbol{\varphi}^* \|^2_{\mathcal{H}} \\ &= \langle \Phi(\mathbf{x}) - \boldsymbol{\varphi}^*, \Phi(\mathbf{x}) - \boldsymbol{\varphi}^* \rangle_{\mathcal{H}} \\ &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}) \rangle_{\mathcal{H}} - 2\langle \Phi(\mathbf{x}), \boldsymbol{\varphi}^* \rangle_{\mathcal{H}} + \langle \boldsymbol{\varphi}^*, \boldsymbol{\varphi}^* \rangle_{\mathcal{H}} \\ &= \kappa(\mathbf{x}, \mathbf{x}) - 2 \sum_{i=1}^{N} \alpha_i \kappa(\mathbf{x}, \mathbf{x}_i) + \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j). \end{aligned} \tag{2.3}$$

Cependant, en se limitant à examiner uniquement les données du jeu de données, représentant un sous-ensemble de $\mathcal{X}$, il est possible que nous n'obtenions qu'une pré-image non optimale parmi les données observées exclusivement.

Pour remédier à cela, nous suggérons d'adopter une méthode générative utilisant un espace latent $\mathcal{Z}$ à partir duquel il est possible de générer des nouvelles données non-présentes dans l'ensemble d'apprentissage dans l'espace des observations $\mathcal{X}$. Un tel espace permet d'établir une correspondance entre chaque $\mathbf{z}_i \in \mathcal{Z}$ et chaque $\mathbf{x}_i \in \mathcal{X}$, et donc chaque $\Phi(\mathbf{x}_i) \in \mathcal{H}$. Ainsi, la résolution du problème de pré-image implique de trouver un élément de $\mathbf{z}$ qui minimise l'équation (2.1) pour l'élément $\mathbf{x}$ correspondant.

En conséquence, nous adoptons une approche générative basée sur un NF, noté f^{-1}. Comme expliqué dans la Section 1.3.4, les NF nous permettent d'apprendre des fonctions inversibles entre l'espace des observations $\mathcal{X}$ et un espace latent $\mathcal{Z}$. Grâce à la propriété d'inversibilité de f, la génération d'une nouvelle donnée $\mathbf{x} \in \mathcal{X}$ est réalisée directement en calculant $f^{-1}(\mathbf{z})$, où $\mathbf{z} \in \mathcal{Z}$. Après avoir décrit la méthode générative, il est nécessaire de déterminer l'élément $\mathbf{z} \in \mathcal{Z}$ à partir duquel la pré-image sera générée. Une première approche consiste à sélectionner cet élément en calculant l'équivalent de l'équation (1.1) dans $\mathcal{Z}$ en utilisant $\mathbf{z} = \sum_{i=1}^{N} \alpha_i \, \mathbf{z}_i$.

Cependant, étant donné que l'espace $\mathcal{Z}$ généré par le NF détient une structure potentiellement différente de celle de l'espace $\mathcal{H}$ du noyau, nous définissons la notion d'alignement entre espaces, utilisé dans le reste de ce chapitre, comme étant deux espaces présentant des relations inter-données similaires. Ainsi, notre espace latent $\mathcal{Z}$ n'étant pas aligné à l'espace $\mathcal{H}$ engendré par le noyau, nous proposons une méthode visant à améliorer l'estimation d'un point $\mathbf{z} \in \mathcal{Z}$ pour la génération de la pré-image. À cet effet, nous proposons de générer un espace isométrique à $\mathcal{H}$ par l'utilisation d'une fonction d'alignement $g_\Theta : \mathcal{Z} \to \mathcal{H}'$. Ainsi, nous cherchons à modéliser cette fonction par l'apprentissage d'un réseau de neurones paramétré par Θ.

La Figure 2.1 présente un aperçu de l'approche, illustrant les relations entre les différents éléments de notre méthode suivant :

- La fonction de transformation implicite Φ, associée au noyau κ, établit la connexion entre l'espace des observations $\mathcal{X}$ et l'espace de Hilbert à noyau reproduisant (RKHS) $\mathcal{H}$, expliqué en détail dans la Section 1.2.
- Le bloc NF établit une correspondance entre l'espace $\mathcal{X}$ des données et l'espace $\mathcal{Z}$ du NF, ce qui permet la projection de $\mathbf{z} \in \mathcal{Z}$ vers $\mathbf{x} \in \mathcal{X}$ à l'aide de la fonction f^{-1}, et inversement. La Section 2.2.1 détaille cette opération.
- Enfin, le bloc d'optimisation permet l'alignement de l'espace latent $\mathcal{Z}$ associé au NF avec l'espace $\mathcal{H}$ du noyau, créant ainsi un nouvel espace $\mathcal{H}'$. Les Sections 2.2.2 et 2.2.3 détaillent ces processus.

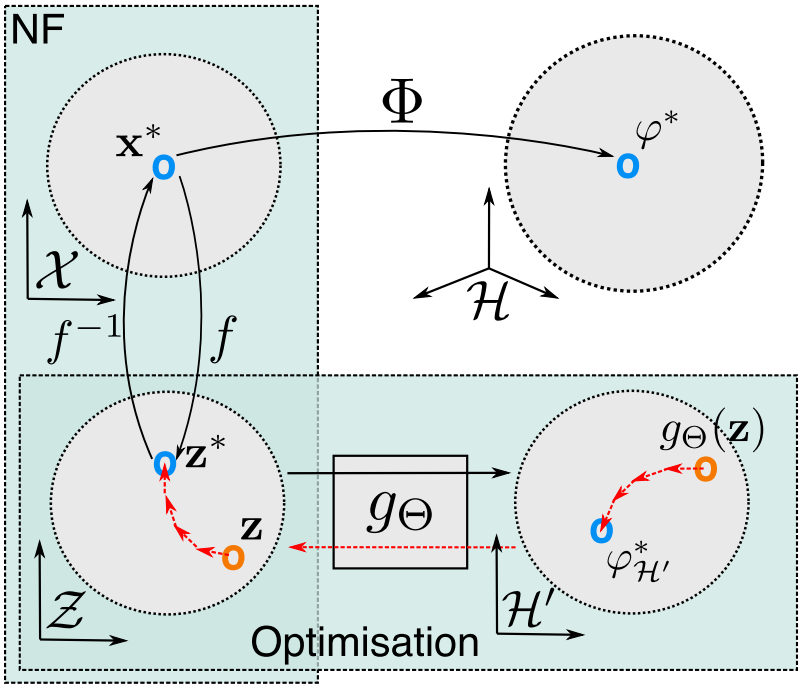


FIGURE 2.1 – Schéma de notre approche présentant la relation entre l'espace des observations $\mathcal{X}$, l'espace $\mathcal{H}$ engendré par le noyau, le bloc génératif utilisant le NF, ainsi que le bloc d'optimisation de $\mathbf{z} \in \mathcal{Z}$ par le modèle d'alignement g_Θ.

2.2.1 Modèle génératif

Nous suggérons l'utilisation d'un modèle génératif basé sur un NF $f : \mathcal{X} \to \mathcal{Z}$ pour notre approche, nous permettant la définition d'un espace latent $\mathcal{Z}$ correspondant à notre espace des observations $\mathcal{X}$. Notre objectif est de proposer une approche générique qui ne dépend pas spécifiquement de l'espace des observations $\mathcal{X}$ utilisé. Cependant, les jeux de données utilisés peuvent varier en fonction du domaine ou du problème. En effet, il est courant de travailler avec différents types de données représentées dans différents espaces, comme les données vectorielles représentées dans un espace euclidien. De plus, il est possible d'être confronté à des espaces discrets lors de l'utilisation de données de type graphe, par exemple. De manière similaire, il existe divers types de noyaux qui dépendent des caractéristiques des données de l'espace des observations, tels que les noyaux de graphes [54].

Un NF est un modèle qui permet la génération d'un espace continu $\mathcal{Z}$ dans lequel les données suivent une distribution simple, telle qu'une distribution gaussienne isotrope. Par conséquent, par l'utilisation d'un NF opérant sur l'espace des observations $\mathcal{X}$ donné, nous sommes en mesure d'apprendre un espace latent $\mathcal{Z}$ continu, simple et facilement interprétable, indépendamment de l'espace qu'il transforme. De cette façon, notre approche permet une flexibilité dans le choix des architectures de NF (Glow [60], FFJORD [41], OT-Flow [94], etc.), en fonction de la nature de l'espace des observations $\mathcal{X}$. Ainsi, aucune modification n'est apportée au modèle de NF utilisé, et son processus d'entraînement reste identique à ce qui est décrit dans l'état de l'art.

2.2.2 Modèle d'alignement

L'utilisation de NF permet la génération simple et efficace de données dans $\mathcal{X}$ par l'application de la transformation inverse associée au modèle sur des éléments échantillonnés dans l'espace latent $\mathcal{Z}$. Cependant, puisque l'espace $\mathcal{Z}$ et l'espace $\mathcal{H}$ du noyau sont potentiellement différents, transposer la définition d'un élément à pré-imager $\boldsymbol{\varphi}^* \in \mathcal{H}$ en utilisant une combinaison linéaire des représentations des données (défini par le théorème de représentation 1.3) dans l'espace $\mathcal{Z}$ sous la forme de $\mathbf{z} = \sum_{i=1}^{N} \alpha_i \mathbf{z}_i$ peut être incohérent et ne pas apporter une solution satisfaisante au problème initial.

Considérant cela, nous suggérons l'entraînement d'un modèle d'alignement g_Θ permettant la transformation de l'espace latent $\mathcal{Z}$ vers un espace aligné sur l'espace $\mathcal{H}$ du noyau. En dénotant cet espace transformé par $\mathcal{H}'$, nous obtenons ainsi :

$$g_\Theta \colon \mathcal{Z} \to \mathcal{H}'.$$

Nous visons à aligner l'espace $\mathcal{H}'$ à $\mathcal{H}$, ce qui permet une transposition plus cohérente de $\boldsymbol{\varphi}^*$ dans $\mathcal{H}'$. En d'autres termes, nous cherchons à obtenir une isométrie entre les deux espaces, où les produits scalaires sont similaires dans les deux. Sachant les données d'entraînement, $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N$, et leurs représentations dans $\mathcal{Z}$ par f notées $\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_N$, le processus d'alignement est effectué de manière que

$$\langle g_\Theta(\mathbf{z}_i), g_\Theta(\mathbf{z}_j) \rangle_{\mathcal{H}'} \approx \kappa(\mathbf{x}_i, \mathbf{x}_j), \forall i, j \in \{1, 2, \ldots, N\} \tag{2.4}$$

où le produit scalaire dans $\mathcal{H}'$ est noté $\langle \cdot, \cdot \rangle_{\mathcal{H}'}$.

Ainsi, pour tous les $i, j = 1, 2, \ldots, N$, l'apprentissage de g_Θ est obtenue par la minimisation de la perte suivante :

$$\mathcal{L}_e(\Theta, i, j) = (\langle g_\Theta(\mathbf{z}_i), g_\Theta(\mathbf{z}_j) \rangle_{\mathcal{H}'} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2. \tag{2.5}$$

Nous pouvons définir la fonction de coût à minimiser comme la moyenne des pertes pour chaque paire de données d'apprentissage, telles qu'introduites par l'équation (2.5) :

$$J_e(\Theta) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \mathcal{L}_e(\Theta, i, j). \tag{2.6}$$

Le modèle g_Θ défini une fonction différentiable paramétrée par Θ et travaillant sur les représentations des données observées dans $\mathcal{Z}$. Il convient de noter que l'architecture du réseau de neurones g_Θ peut varier en fonction du NF utilisé et de l'espace des observations $\mathcal{X}$. Par exemple, lorsque nous travaillons avec des données structurées telles que des images et que le NF sélectionné conserve cette structure dans l'espace latent $\mathcal{Z}$, comme c'est le cas avec Glow [60], il peut être préférable d'utiliser un réseau convolutionnel tel que ResNet [46]. En revanche, si les représentations des données dans $\mathcal{Z}$ sont vectorielles, alors un perceptron multicouches (MLP) peut être utilisé à la place. L'entraînement du modèle repose sur une stratégie de rétropropagation du gradient calculé sur la fonction de coût précédente (équation (2.6)). Ainsi, l'optimisation des poids $\theta^{(j)} \in \Theta$ est effectuée de la manière suivante :

$$\theta^{(j)}_{\text{new}} = \theta^{(j)}_{\text{old}} - \eta \frac{\partial J_e}{\partial \theta^{(j)}_{\text{old}}},$$

où η désigne le taux d'apprentissage prédéfini.

Une telle méthode d'entraînement ne restreint en aucune manière l'espace $\mathcal{H}'$ ni sa dimensionnalité. Nous optons pour l'utilisation d'un espace $\mathcal{H}'$ de dimension équivalente à celle de l'espace latent $\mathcal{Z}$ du NF, permettant une interprétation différente de g_Θ. En effet, la fonction g_Θ peut être perçue comme une fonction de déformation de $\mathcal{Z}$ afin de l'ajuster à $\mathcal{H}$.

2.2.3 Optimisation de la pré-image

Comme mentionné au début de la section, notre objectif est de trouver, à partir de $\mathbf{z}$, l'élément $\mathbf{z}^*$ qui, une fois généré dans l'espace $\mathcal{X}$, aura une projection dans $\mathcal{H}$ égale à $\boldsymbol{\varphi}^*$, c'est-à-dire $\Phi(\mathbf{x}^*) = \boldsymbol{\varphi}^*$. Ainsi, le modèle d'alignement g_Θ vise à optimiser un élément $\mathbf{z} \in \mathcal{Z}$ vers $\mathbf{z}^*$, afin d'obtenir une meilleure pré-image par application de la transformation inverse du NF, à savoir $\mathbf{x}^* = f^{-1}(\mathbf{z}^*)$. La Figure 2.1 illustre visuellement la méthode d'optimisation proposée. Une fois $\mathbf{z}$ initialisé, l'objectif est d'optimiser itérativement $\mathbf{z}$ vers $\mathbf{z}^*$ en utilisant le modèle g_Θ. Étant donné l'équation (2.3), il est possible de formuler le problème de minimisation pour obtenir la pré-image souhaitée de la manière suivante :

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}}\ \kappa(\mathbf{x},\mathbf{x}) - 2\sum_{i=1}^{N}\alpha_i\kappa(\mathbf{x},\mathbf{x}_i). \tag{2.7}$$

Plusieurs limitations entravent la résolution directe de ce problème d'optimisation [51]. Parmi celles-ci, le noyau doit être différentiable et l'espace d'optimisation continu. Cependant, dans de nombreux cas, cela n'est pas applicable, comme illustré par les graphes [53], par exemple.

Pour remédier à cette problématique, le modèle g_Θ offre une isométrie entre les espaces $\mathcal{H}$ et $\mathcal{H}'$ conformément à l'équation (2.4). En tenant compte de cela, nous pouvons substituer les termes dépendant de $\mathbf{x}$ en utilisant g_Θ. Par conséquent, nous sommes en mesure de transférer notre problème vers $\mathcal{H}'$ et de reformuler l'équation (2.7) en une fonction de coût à minimiser, comme suit :

$$J_o = \langle g_\Theta(\mathbf{z}), g_\Theta(\mathbf{z})\rangle_{\mathcal{H}'} - 2\sum_{i=1}^{N}\alpha_i\langle g_\Theta(\mathbf{z}), g_\Theta(\mathbf{z}_i)\rangle_{\mathcal{H}'}. \tag{2.8}$$

De cela, nous désirons l'optimisation de l'échantillon $\mathbf{z}$ tiré de l'espace latent $\mathcal{Z}$ généré par le NF. Pour ce faire, nous nous sommes appuyés sur la recherche effectuée dans plusieurs travaux proposant l'optimisation de molécule suivant une propriété donnée par optimisation bayésienne dans des architectures de VAE [39, 69, 55, 26] ou par utilisation d'apprentissage par renforcement [134, 102]. Puisque le modèle d'alignement est une fonction différentiable, il est possible d'optimiser les paramètres de $\mathbf{z}$ en utilisant la rétropropagation du gradient calculée sur le coût (équation (2.8)) selon

$$\mathbf{z}^{(j)}_{\text{new}} = \mathbf{z}^{(j)}_{\text{old}} - \eta\frac{\partial J_o}{\partial\, \mathbf{z}^{(j)}_{\text{old}}},$$

avec le taux d'apprentissage utilisé η et en notant $\mathbf{z}^{(j)} \in \mathbf{z}$ pour représenter la composante j de $\mathbf{z}$ à optimiser. Ainsi, il est possible d'optimiser $\mathbf{z}$ afin de le rapprocher de $\mathbf{z}^*$ dans le but d'obtenir une pré-image améliorée par l'application de la transformation inverse du NF, à savoir $f^{-1}(\mathbf{z}^*)$.

2.3 Expérimentations

Dans cette section, nous nous intéressons à l'expérimentation de l'approche que nous proposons pour les différentes tâches associées. Plus précisément, après avoir introduit la configuration utilisée pour ces expériences dans la Section 2.3.1, nous nous concentrons sur l'alignement de l'espace généré (Section 2.3.2) ainsi que sur la génération et l'optimisation de pré-image (Section 2.3.3).

2.3.1 Configuration

Pour évaluer notre méthode, nous employons le jeu de données MNIST [31]. Ce jeu de données est composé de 60 000 images de chiffres manuscrits ayant une résolution de 28×28 pixels. En utilisant ce jeu de données de référence, nous serons en mesure d'évaluer l'efficacité de notre approche de génération de pré-image. Dans le cadre de nos expérimentations, nous utilisons un noyau gaussien classique $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\| \mathbf{x}_i - \mathbf{x}_j \|^2 / 2\sigma^2)$, où $\sigma = 2500$. En utilisant un SVM par exemple, ce noyau parvient à classer correctement 96% d'un sous-ensemble du jeu de données MNIST, composé de 1000 images par classe, ce qui démontre sa pertinence. Pour évaluer ce noyau, nous utilisons un ensemble de validation qui représente 10% des données de chaque classe, soit 600 images. Il est important de noter que tout autre noyau semi-défini positif peut également être utilisé.

Étant donné que l'architecture de NF à utiliser est libre, comme indiqué dans la Section 2.2.1, notre choix s'est porté sur Glow, FFJORD et OT-Flow, des architectures introduites dans la section 1.3.4 et adaptées au traitement d'images. Conformément à la Section 2.2.2, l'architecture du modèle d'alignement g_Θ varie en fonction du NF utilisé. En effet, lorsque nous utilisons Glow, qui produit des sorties à plusieurs échelles tout en préservant la structure des images (à savoir, couche, largeur, hauteur), nous optons pour une architecture de réseau résiduel (ResNet) avec convolutions. En revanche, pour FFJORD et OT-Flow, nous utilisons des perceptrons multicouches (MLP) simples composés de deux couches cachées.

2.3.2 Alignement

Chaque modèle d'alignement est appris en suivant notre méthodologie indiquée dans la Section 2.2.2. Par conséquent, pour évaluer la pertinence de nos modèles, nous mesurons l'alignement de nos espaces générés avec celui engendré par le noyau $\mathcal{H}$. Pour cela, nous évaluons tout d'abord l'alignement de l'espace avant optimisation $\mathcal{Z}$ puis l'alignement de l'espace après optimisation $\mathcal{H}'$. L'évaluation de $\mathcal{Z}$ est réalisée en utilisant :

$$\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\langle \mathbf{z}_i, \mathbf{z}_j \rangle_{\mathcal{Z}} - \kappa(\mathbf{x}_i, \mathbf{x}_j))^2 \tag{2.9}$$

Après cela, nous calculons l'alignement de l'espace généré $\mathcal{H}'$ en utilisant le coût J_e, comme défini par l'équation (2.6). Les résultats sont présentés dans la Table 2.1. Ces derniers permettent la mise en évidence du bon alignement de l'espace généré $\mathcal{H}'$ par le modèle d'alignement g_Θ à partir de l'espace latent $\mathcal{Z}$ du NF qui, dans son cas, n'est pas aligné.

TABLE 2.1 – Évaluation de l'alignement des espaces latent $\mathcal{Z}$ (équation (2.9)) et généré $\mathcal{H}'$ avec g_Θ (équation (2.6))

Espace	Glow	FFJORD	OT-Flow
$\mathcal{Z}$	0,1828	0,0452	0,1306
$\mathcal{H}'$	8,62e-5	0,0001	6,20e-5

2.3.3 Génération

Pour évaluer notre méthode de génération de pré-images cohérentes pour un problème spécifique, nous considérons le scénario où $\boldsymbol{\varphi}^*$ représente le centroïde de k échantillons dans l'espace $\mathcal{H}$. Cette situation est courante dans des domaines tels que le filtrage ou l'approximation [53], par exemple en utilisant les k voisins les plus proches. Pour chaque expérimentation, nous effectuons une sélection aléatoire d'un ensemble de k plus proches voisins dans $\mathcal{H}$. Par conséquent, nous définissons la pondération de chaque échantillon pour définir notre point d'intérêt de la manière suivante :

$$\alpha_i = \begin{cases} 1/k, & \text{si } \Phi(\mathbf{x}_i) \in k \text{ plus proches voisins} \\ 0, & \text{sinon} \end{cases}$$

Ensuite, de la manière expliquée dans la Section 2.2, une première estimation triviale de la pré-image est réalisée en choisissant la meilleure donnée parmi toutes celles présentes dans l'ensemble d'apprentissage (équation (2.2)). Pour cela, nous procédons à une exploration complète de l'ensemble d'entraînement afin de trouver la donnée qui minimise l'équation (2.3). Cette pré-image appartenant à notre jeu de données, notée $\mathbf{x}_d$, représente à la fois la meilleure pré-image disponible dans notre jeu de données et notre référence pour la comparaison.

Génération de pré-image

Tout d'abord, notre objectif est de déterminer si la pré-image obtenue à partir de $\mathcal{Z}$ permet d'obtenir une distance $d^2_{\mathcal{H}}$ plus faible dans $\mathcal{H}$, ce qui se traduirait par une meilleure pré-image par rapport à $\mathbf{x}_d$. Pour ce faire, nous commençons par générer une pré-image appelée $\mathbf{x}_{\text{nf}}$, sans prendre en compte d'éventuelles différences de relation entre les données dans $\mathcal{Z}$ et $\mathcal{H}$, telles qu'indiquées dans la Section 2.2. L'obtention de celle-ci s'effectue par le calcul de la représentation dans $\mathcal{Z}$ de la même manière que dans $\mathcal{H}$, c'est-à-dire avec $\mathbf{z}_{\text{nf}} = \sum_{i=1}^{N} \alpha_i \, \mathbf{z}_i$, puis en appliquant la transformation inverse du NF pour générer la pré-image $\mathbf{x}_{\text{nf}} = f^{-1}(\mathbf{z}_{\text{nf}})$. La Figure 2.2 nous permet de faire une comparaison entre $\mathbf{x}_d$ et les $\mathbf{x}_{\text{nf}}$ générés par chaque modèle en termes de distance $d^2_{\mathcal{H}}$ (équation (2.3)). Afin de fournir davantage de précision, ces valeurs sont répertoriées dans la Table 2.2. Les résultats sont obtenus en moyennant 100 simulations pour chaque valeur de k, avec $2 \leq k \leq 15$. Ces résultats mettent en évidence les performances remarquables du modèle OT-Flow, où les valeurs de $d^2_{\mathcal{H}}$ calculées à partir de $\mathbf{x}_{\text{nf}}$ sont nettement inférieures à celles calculées à partir de $\mathbf{x}_d$. Dans le cas du NF Glow, nous constatons une amélioration de la pré-image en utilisant une valeur de $k > 4$.

TABLE 2.2 – Moyenne de $d^2_{\mathcal{H}}$ calculée sur 100 simulations de Monte Carlo, en variant le nombre de k plus proches voisins, appliquées aux images du chiffre 3 du jeu de données MNIST.

		Glow		FFJORD		OT-Flow	
k	$\mathbf{x}_d$	$\mathbf{x}_{\text{nf}}$	$\mathbf{x}_{\text{opti}}$	$\mathbf{x}_{\text{nf}}$	$\mathbf{x}_{\text{opti}}$	$\mathbf{x}_{\text{nf}}$	$\mathbf{x}_{\text{opti}}$
2	0,1451	0,1748	0,1727	0,8215	0,7222	0,0304	**0,0300**
3	0,1673	0,1895	0,1865	0,8310	0,7473	0,0372	**0,0367**
4	0,1869	0,1889	0,1858	0,8562	0,7675	0,0442	**0,0439**
5	0,1956	0,1854	0,1824	0,8270	0,7473	0,0480	**0,0475**
6	0,2017	0,1883	0,1845	0,7841	0,6977	0,0512	**0,0505**
7	0,2029	0,1848	0,1817	0,8080	0,7221	0,0480	**0,0476**
8	0,1950	0,1775	0,1742	0,7738	0,6941	0,0462	**0,0459**
9	0,2131	0,1871	0,1831	0,8053	0,7145	0,0534	**0,0525**
10	0,2073	0,1826	0,1794	0,7856	0,6980	0,0533	**0,0526**
11	0,2172	0,1888	0,1858	0,8165	0,7257	0,0542	**0,0537**
12	0,2116	0,1819	0,1784	0,7643	0,6931	0,0541	**0,0539**
13	0,2098	0,1818	0,1777	0,8023	0,7105	0,0525	**0,0519**
14	0,2111	0,1769	0,1737	0,7710	0,6894	0,0505	**0,0499**
15	0,2136	0,1783	0,1748	0,7731	0,6861	0,0564	**0,0553**

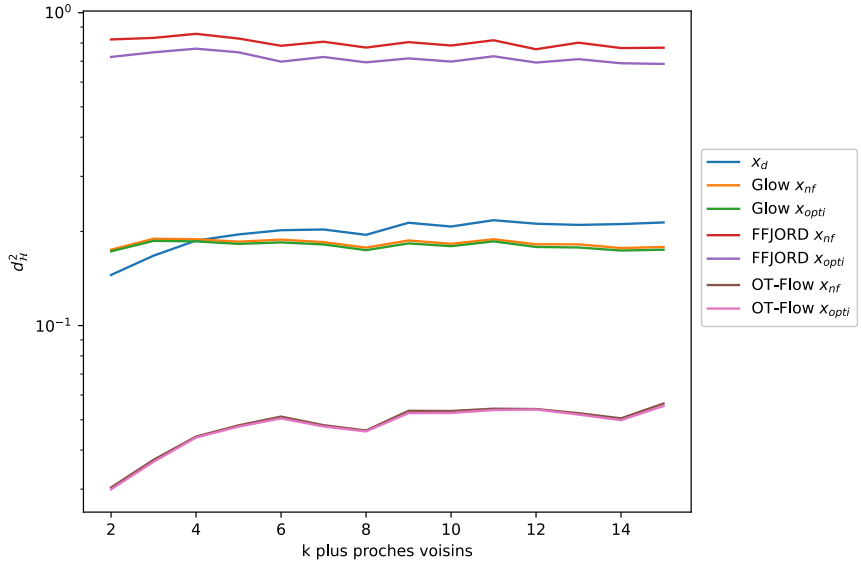


FIGURE 2.2 – Moyenne de $d^2_{\mathcal{H}}$, obtenue à partir de 100 simulations de Monte Carlo, tracée en fonction du nombre k de plus proches voisins sur les images du chiffre 3 de MNIST.

En revanche, les pré-images obtenues avec le modèle FFJORD ne correspondent pas au noyau utilisé, comme en témoignent les valeurs élevées des distances $d^2_{\mathcal{H}}$ obtenues. Ces résultats permettent d'évaluer la similarité entre $\mathcal{Z}$ et $\mathcal{H}$. En effet, l'espace $\mathcal{H}$ engendré par κ présente une grande similarité avec l'espace $\mathcal{Z}$ généré par OT-Flow, une similarité assez marquée avec celui de Glow, et une grande différence par rapport à celui de FFJORD. De plus, nous pouvons observer que le nombre de k plus proches voisins a peu d'impact sur la génération de la pré-image.

Optimisation de pré-image

Finalement, notre objectif est d'évaluer dans quelle mesure notre procédure d'optimisation, décrite dans les Sections 2.2.2 et 2.2.3, permet d'améliorer la qualité de la pré-image en termes de $d^2_{\mathcal{H}}$ (équation (2.3)). À cet effet, nous cherchons à optimiser l'entrée $\mathbf{z}_{\text{nf}}$ et évaluons ensuite une nouvelle valeur en utilisant l'équation (2.8). La Figure 2.3 présente des aperçus de l'optimisation en comparant l'évolution des pertes à minimiser J_o (équation (2.8)) avec le calcul de $d^2_{\mathcal{H}}$ (équation (2.3)) à chaque itération. La valeur la plus faible de $d^2_{\mathcal{H}}$ atteinte correspond à la pré-image $\mathbf{x}_{\text{opti}}$ retenue.

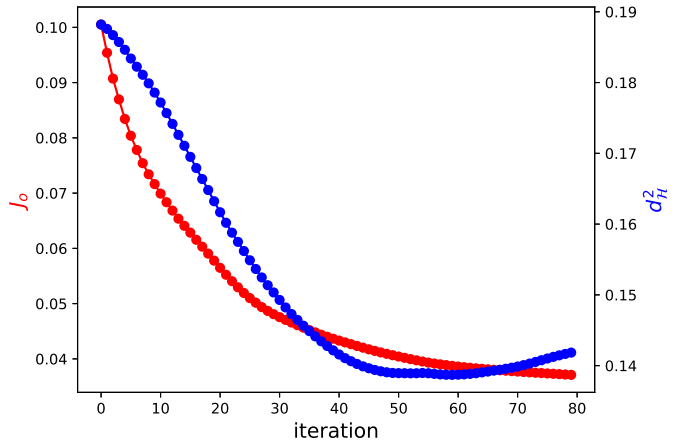
Une fois que nous avons obtenu $\mathbf{x}_{\text{nf}}$ et $\mathbf{x}_{\text{opti}}$ pour chaque modèle, les valeurs correspondantes de $d^2_{\mathcal{H}}$ associées peuvent être comparées, comme illustré dans la Figure 2.2. Ainsi, nous pouvons observer que pour les NF OT-Flow et Glow, les valeurs de $d^2_{\mathcal{H}}$ après optimisation ne diffèrent pas significativement, voire très peu, des valeurs sans optimisation, comme le montre la Table 2.2. En ce qui concerne FFJORD, l'optimisation fonctionne mieux, mais $\mathbf{x}_{\text{opti}}$ reste une pré-image moins satisfaisante que $\mathbf{x}_d$. De ces résultats, nous pouvons en déduire que lorsque la pré-image est bonne sans optimisation, le gain résultant de l'optimisation sera relativement faible. De plus, nous observons que le nombre des k plus proches voisins n'a aucune incidence sur les résultats d'optimisation.

2.4 Conclusion

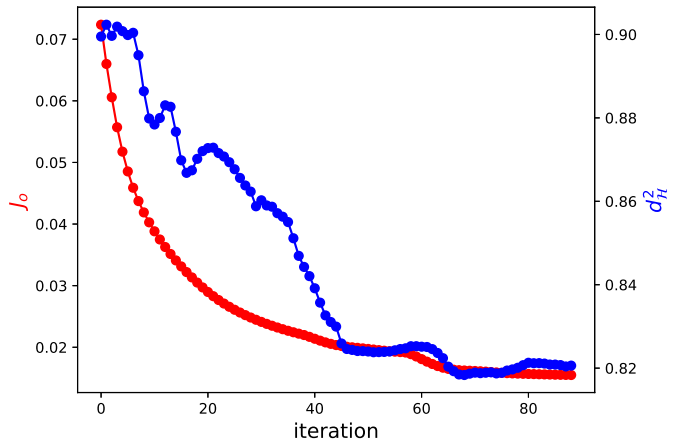
Dans ce chapitre, une approche basée sur l'utilisation de NF a été présentée pour résoudre le problème de pré-image associé aux méthodes à noyaux. Une méthode d'alignement a été proposée pour explorer l'espace latent des NF et déterminer une donnée qui, une fois générée, constitue une pré-image améliorée. Les résultats expérimentaux ont démontré que certains NF, tels que OT-Flow, permettent une génération de pré-image de bonne qualité. De plus, l'approche d'optimisation proposée permet une légère amélioration des pré-images. Cependant, les résultats obtenus ne permettent pas d'affirmer la pertinence de celle-ci.

Les espaces engendrés par les méthodes à noyaux peuvent différer de ceux générés pas les NF en général, comme illustré par le modèle FFJORD dans nos expériences. Au lieu de diviser l'approche en deux parties distinctes, à savoir l'apprentissage du modèle de NF suivi de l'alignement de l'espace engendré, il peut être bénéfique d'orienter la génération de l'espace en adaptant l'apprentissage du NF de manière à ce qu'il corresponde à l'espace engendré par un noyau.

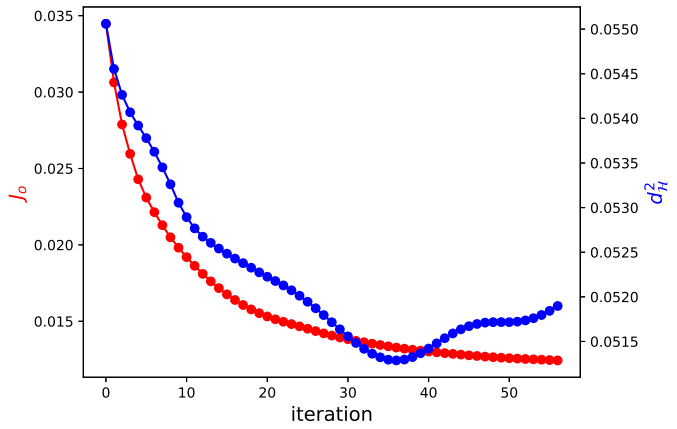
Même si les méthodes basées sur les noyaux peuvent offrir des représentations de données performantes, elles souffrent d'un manque d'expressivité et ne génèrent pas nécessairement les meilleures espaces de représentation possibles. Par conséquent, il n'est pas nécessaire de se limiter à l'approximation d'un espace généré par un noyau pour tirer parti des avancées majeures de l'apprentissage profond. Dans la suite de ce document, nous proposons la définition d'approche d'apprentissage machine sans problème de pré-image sans se restreindre à l'utilisation de noyau.



(a) Glow



(b) FFJORD



(c) OT-Flow

FIGURE 2.3 – Les graphiques illustrent l'évolution des pertes à minimiser, représentées en rouge pour la fonction objectif J_o (équation (2.8)), ainsi que le calcul de $d^2_{\mathcal{H}}$ (équation (2.3)) présenté en bleu, à chaque itération d'optimisation. Chaque courbe correspond à l'optimisation d'une simulation, avec un paramètre $k = 8$, pour les modèles d'alignement associés aux méthodes de NF Glow, FFJORD et OT-Flow.

CHAPITRE 3

Méthodes non supervisées insensibles au problème de pré-image

3.1 Introduction

Traditionnellement, l'apprentissage machine (voir Section 1.1) repose sur des approches supervisées, où les modèles sont entraînés à partir de données étiquetées afin de réaliser des prédictions. Cependant, l'utilisation de telles approches nécessite des ensembles de données étiquetés, ce qui peut être coûteux, voire impossible à obtenir dans de nombreux domaines. L'apprentissage non supervisé, quant à lui, permet la génération d'un espace de caractéristiques à partir de données non étiquetées, sans le besoin d'une supervision du modèle lors de l'apprentissage. L'objectif principal d'un tel apprentissage est de découvrir les structures et les motifs sous-jacents des données permettant une meilleure interprétation de ces dernières. À partir des informations structurelles des données apprises, il devient possible d'utiliser ces modèles sur des tâches spécifiques, telles que le débruitage de données ou la génération de nouvelles données. Ainsi, cette approche d'apprentissage est directement concernée par le problème de pré-image tel qu'introduit dans la Section 1.2. En effet, dès lors que l'on souhaite générer une nouvelle donnée à partir d'un point d'intérêt appartenant à l'espace de caractéristiques, il est nécessaire d'avoir connaissance d'une transformation inverse permettant le passage de l'espace des caractéristiques à l'espace des observations. De même, en considérant un point résultant d'une projection sur un sous-espace dans l'espace des caractéristiques, le débruitage ou la compression d'une donnée implique l'obtention de sa correspondance dans l'espace des observations.

Dans ce chapitre, nous proposons un formalisme de reconnaissance des formes qui atténue la malédiction de la pré-image. L'idée sous-jacente est de considérer un encodage non-linéaire inversible par construction. Ainsi, d'une part, les opérations dans l'espace de caractéristiques peuvent être facilement effectuées, comme la projection sur un sous-espace (par exemple, l'analyse des composantes principales (ACP)). D'autre part, le résultat obtenu peut être facilement représenté dans

l'espace des observations grâce à la réversibilité de la transformation

Pour démontrer la pertinence du formalisme proposé, nous proposons une approche en deux versions pour traiter des tâches de débruitage. Pour traiter ces tâches, nous démontrons qu'il est possible de dériver une méthode sans problème de pré-image reposant sur une ACP dans l'espace de caractéristiques, cette dernière générée grâce à un encodage non-linéaire réversible des données observées. Compte tenu de cette transformation, la pré-image d'un point débruité dans l'espace de caractéristiques peut être calculée grâce à la fonction de transformation inverse. La première version de notre approche proposée consiste à estimer la matrice de covariance nécessaire au calcul de l'ACP dans l'espace de caractéristiques. La seconde vise à paramétrer de manière appropriée l'apprentissage de la fonction de transformation de sorte que les données dans l'espace de caractéristiques suivent une distribution gaussienne particulière. Celle-ci est caractérisée par une matrice de covariance construite à partir d'une décomposition en éléments propres connue, résultant en une transformation avec projection intégrée. Ainsi, le temps nécessaire pour la décomposition en éléments propres lors de l'application d'une ACP conventionnelle est évité.

Nos approches visant à produire une fonction non-linéaire réversible s'inspirent des récentes avancées dans le domaine des modèles génératifs de l'apprentissage machine, où les VAE, GAN et NF ont tous été l'objet d'études approfondies. La Section 1.3 présente ces différentes approches en détail. De cette analyse, nous proposons l'utilisation des NF pour définir notre transformation non-linéaire inversible.

Dans les expériences menées, nous montrons l'efficacité de notre formalisme sur plusieurs ensembles de données, en explorant trois architectures de NF différentes, RealNVP [33], FFJORD [41] et Glow [60]. Ces modèles sont présentés dans la Section 1.3.4. Plus précisément, ces expériences montrent les bonnes performances de débruitage de notre approche par application de l'ACP dans l'espace de caractéristiques. Enfin, en plus de ces dernières utilisations, nous montrons que notre modèle permet des générations pertinentes et de bonne qualité.

Le reste de ce chapitre est organisé comme suit. Notre formalisme, présenté dans la Section 3.2, est divisé en deux versions introduites respectivement dans les Sections 3.2.1 et 3.2.2. Les expériences sont menées dans la Section 3.3. Enfin, la Section 3.4 conclut ce chapitre.

3.2 Approches proposées

Le problème de pré-image, tel qu'il a été introduit dans la Section 1.2, est une limitation majeure de la plupart des techniques d'apprentissage machine telles que les méthodes basées sur les noyaux ou les réseaux de neurones profonds. Cependant, le calcul de la pré-image d'un point particulier dans l'espace de caractéristiques $\mathcal{Z}$ peut être d'un grand intérêt lorsque l'on souhaite apporter de l'interprétabilité à un modèle et à ses prédictions.

Afin de résoudre ce problème, notre contribution considère l'espace latent $\mathcal{Z}$ généré par un NF comme l'espace de caractéristiques d'intérêt. L'idée principale, illustrée dans la Figure 3.1, consiste à

(i) apprendre une représentation latente des données à l'aide d'une fonction d'encodage Φ apprise par un modèle de NF,

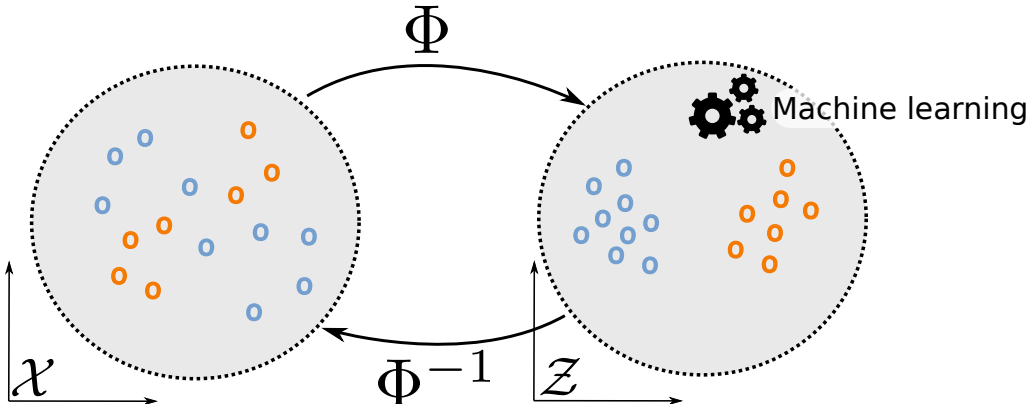


FIGURE 3.1 – Formalisme général de notre contribution. En considérant l'espace de caractéristiques $\mathcal{Z}$ d'un NF, nous opérons efficacement des modèles ML/PR linéaires dans cet espace. De plus, le problème de pré-image est facilement résolu en utilisant la fonction inverse du NF.

(*ii*) effectuer des opérations/modèles linéaires dans l'espace de caractéristiques $\mathcal{Z}$,

(*iii*) et enfin, le calcul de la pré-image de tout point de l'espace $\mathcal{Z}$ est trivial grâce à l'inversibilité de la fonction d'encodage Φ.

Dans ce qui suit, nous décrivons ces trois étapes.

1. Soit un jeu de données $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ pour $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^D$, nous proposons d'entraîner un NF pour apprendre une fonction $\Phi : \mathcal{X} \to \mathcal{Z}$ telle que la distribution générée par la transformation Φ suive une distribution gaussienne multivariée paramétrée par une moyenne arbitraire $\boldsymbol{\mu}$ et une matrice de covariance $\boldsymbol{\Sigma}$. En désignant Θ, l'ensemble des paramètres du modèle du NF, son apprentissage est détaillé dans l'Alg. 1.

   Dans la littérature, les NF s'appuient souvent sur une distribution gaussienne multivariée isotrope, en paramétrant $\boldsymbol{\mu}$ à zéro sur toutes les dimensions et la matrice de covariance $\boldsymbol{\Sigma}$ à la matrice d'identité. Nous proposons, dans ce chapitre, de personnaliser la définition de la distribution cible plutôt que d'utiliser une distribution cible isotrope, en fonction de l'objectif souhaité. Nous définissons donc la matrice de covariance $\boldsymbol{\Sigma}$ à l'aide d'un ensemble prédéfini de valeurs propres λ_i et de vecteurs propres $\mathbf{v}_i$, à savoir

$$\boldsymbol{\Sigma} = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i \,. \tag{3.1}$$

   De plus, nous voulons modeler notre distribution latente de manière à ce qu'elle contienne k composantes principales, tandis que les composantes restantes représentent les variations secondaires des données. Pour ce faire, nous définissons premièrement les vecteurs propres comme un ensemble de vecteurs orthogonaux correspondant à chaque dimension de l'espace latent, à savoir

$$\begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \ldots & \mathbf{v}_D \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}. \tag{3.2}$$

Algorithm 1 TrainNF(X, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$) : Procédure générale d'entraînement du NF.

Require: Ensemble d'apprentissage : X, Moyenne : $\boldsymbol{\mu}$, Matrice de covariance : $\boldsymbol{\Sigma}$, taux d'apprentissage : η

1: $(\Theta, \Phi, \Phi^{-1}) \leftarrow$ Initialisation du modèle de NF
2: **for** $\mathbf{x} \in X$ **do**
3: $\quad \mathbf{z} = \Phi(\mathbf{x})$
4: $\quad \log P_{\mathcal{Z}}(\mathbf{z}) = -\frac{1}{2}\left(D\log(2\pi) + (\mathbf{z}-\boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})\right) - \log(\det(\boldsymbol{\Sigma}))$
5: $\quad \mathcal{L}(\mathbf{x}) = -\log P_{\mathcal{Z}}(\mathbf{z}) - \log\left|\det\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)\right|$
6: $\quad \Theta \leftarrow \Theta - \eta\nabla_{\Theta}\mathcal{L}(\mathbf{x})$
7: $\quad (\Phi, \Phi^{-1}) \leftarrow$ Mise à jour des fonctions avec les paramètres Θ
8: **end for**
9: **return** (Φ, Φ^{-1})

Deuxièmement, nous définissons les valeurs propres avec k valeurs élevées pour encourager la distribution cible à s'organiser le long de ces k dimensions. À cette fin, nous fixons les valeurs propres élevées à une même valeur prédéfinie, c'est-à-dire

$$\lambda_1 = \lambda_2 = \cdots = \lambda_k. \tag{3.3}$$

Pour les valeurs propres inférieures restantes, nous les fixons de manière que le déterminant de la matrice de covariance $\boldsymbol{\Sigma}$ soit égal à 1, ce qui signifie que la dispersion des données dans l'espace $\mathcal{X}$ est préservée dans l'espace $\mathcal{Z}$. Comme le déterminant est le produit de ses valeurs propres, nous obtenons

$$\lambda_{k+1} = \lambda_{k+2} = \cdots = \lambda_D = \epsilon, \tag{3.4}$$

où $\epsilon = \sqrt[D-k]{1/\lambda_1^k}$.

2. Grâce à l'espace de caractéristiques appris $\mathcal{Z}$, nous pouvons directement effectuer des opérations simples (telles que des projections orthogonales) dans cet espace. Nous proposons dans les sections suivantes une approche divisée en deux différentes versions (Sections 3.2.1 et 3.2.2) qui tirent parti de la structure simple de l'espace de caractéristiques.
3. Enfin, comme nous utilisons un NF, qui est un modèle inversible par construction, la transformation apprise Φ et son inverse Φ^{-1} sont produits par notre algorithme d'apprentissage (Alg. 1), ce qui garantit la possibilité de calculer la pré-image de n'importe quel point de l'espace de caractéristiques $\mathcal{Z}$.

Dans ce chapitre, nous démontrons la pertinence du formalisme général proposé en développant deux implémentations pour traiter des tâches de débruitage.

3.2.1 Débruitage avec $\mathcal{Z}$-ACP

Le débruitage consiste à récupérer des données originales affectées par un certain bruit, ce bruit étant éventuellement caractérisé par une distribution de bruit (gaussien, de Poisson, poivre et sel, ... ou plus générale). Plus formellement, des données originales $\mathbf{x}$ sont modifiées par un bruit $\varepsilon \sim \mathcal{T}$, $\mathcal{T}$ correspondant à une

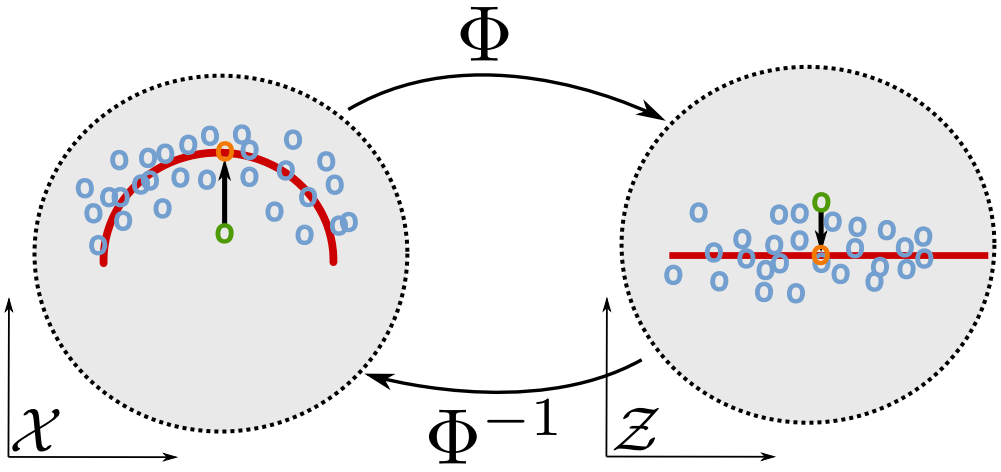


FIGURE 3.2 – Processus général de l'algorithme de débruitage présenté dans la Section 3.2. Les lignes rouges illustrent le sous-espace de projection linéaire dans $\mathcal{Z}$ et sa version non-linéaire dans $\mathcal{X}$.

distribution de probabilité particulière, par exemple $\mathcal{T} = \mathcal{N}(0, 1)$ désigne un bruit distribué selon une distribution normale. La version bruitée $\tilde{\mathbf{x}}$ de $\mathbf{x}$ est donnée par $\tilde{\mathbf{x}} = \mathbf{x} + \varepsilon$. Le débruitage consiste à retrouver $\mathbf{x}$ étant donné $\tilde{\mathbf{x}}$, généralement sans connaître directement la distribution $\mathcal{T}$ et ses paramètres. L'évaluation de la qualité du débruitage peut être effectuée en calculant les différences entre $\mathbf{x}$ et $\tilde{\mathbf{x}}$ lorsque $\mathbf{x}$ est connu, ou en utilisant une appréciation qualitative si $\mathbf{x}$ n'est pas connu.

Pour concevoir un algorithme de débruitage qui ne souffre pas du problème de pré-image, (*i*) nous définissons l'espace de caractéristiques $\mathcal{Z}$ par la transformation non-linéaire d'un NF, (*ii*) nous réalisons une ACP dans l'espace de caractéristiques $\mathcal{Z}$, (*iii*) et enfin, nous ramenons les résultats dans l'espace des observations par le calcul de pré-image. Nous appelons cette méthode $\mathcal{Z}$-ACP. Le processus général est présenté dans la Figure 3.2. Alors que les étapes (*i*) et (*iii*) sont présentées en détail au début de la Section 3.2, nous décrivons ci-après les opérations de $\mathcal{Z}$-ACP correspondant à une ACP linéaire sur les représentations des données dans l'espace de caractéristiques.

Soit $\mathbf{z} \in \mathbb{R}^{N \times D}$ la matrice des représentations des données, définie par

$$\mathbf{z} = \begin{pmatrix} \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2) & \cdots & \Phi(\mathbf{x}_N) \end{pmatrix}^\top,$$

et $\mathbf{z}_c$ la matrice des données centrées associée dans $\mathcal{Z}$, dont la j-ième ligne correspond à $\Phi(\mathbf{x}_j) - \bar{\mathbf{z}}$, avec $\bar{\mathbf{z}} = \frac{1}{N}\sum_{i=1}^{N} \Phi(\mathbf{x}_i)$ (voir [49] pour l'influence du centrage). L'exécution de l'ACP dans $\mathcal{Z}$ consiste à projeter les données dans le sous-espace couvert par les vecteurs propres associés aux plus grandes valeurs propres de la matrice de covariance. En estimant la matrice de covariance sur les données d'apprentissage dans $\mathcal{Z}$, nous avons $\hat{\mathbf{\Sigma}} = \frac{1}{N-1}\mathbf{z}_c^\top \mathbf{z}_c$, et sa décomposition en éléments propres prend la forme suivante

$$\hat{\mathbf{\Sigma}} = \widehat{\mathbf{V}}\widehat{\mathbf{\Lambda}}\widehat{\mathbf{V}}^\top. \tag{3.5}$$

où $\widehat{\mathbf{V}}$ contient les vecteurs propres $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_k$ associés aux k plus grandes valeurs propres de la matrice de covariance estimée et $\widehat{\mathbf{\Lambda}}$ la matrice diagonale des valeurs

Algorithm 2 $\mathcal{Z}$-ACP : Procédure de débruitage avec $\mathcal{Z}$-ACP

Require: Données d'apprentissage : X, Données bruitées : $\widetilde{X}$, nombre de composantes principales à garder : k

1: $(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_D) \leftarrow$ (3.2)
2: $(\lambda_1 \quad \lambda_2 \quad \dots \quad \lambda_k) \leftarrow$ (3.3)
3: $(\lambda_{k+1} \quad \lambda_{k+2} \quad \dots \quad \lambda_D) \leftarrow$ (3.4)
4: $\boldsymbol{\Sigma} = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
5: $\boldsymbol{\mu} = \mathbf{0}$
6: $(\Phi, \Phi^{-1}) = \text{TrainNF}(X, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ (Alg. 1)
7: $\mathbf{z} = \begin{pmatrix} \Phi(\mathbf{x}_1) & \Phi(\mathbf{x}_2) & \cdots & \Phi(\mathbf{x}_N) \end{pmatrix}^\top$
8: $\bar{\mathbf{z}} = \frac{1}{N} \sum_{i=1}^{N} \Phi(\mathbf{x}_i)$
9: $\mathbf{z}_c = \begin{pmatrix} \Phi(\mathbf{x}_1) - \bar{\mathbf{z}} & \Phi(\mathbf{x}_2) - \bar{\mathbf{z}} & \cdots & \Phi(\mathbf{x}_N) - \bar{\mathbf{z}} \end{pmatrix}^\top$
10: $\widehat{\boldsymbol{\Sigma}} = \frac{1}{N-1} \, \mathbf{z}_c \, \mathbf{z}_c^\top$
11: $\widehat{\mathbf{V}}, \widehat{\boldsymbol{\Lambda}} \leftarrow eig(\widehat{\boldsymbol{\Sigma}})$ (3.5)
12: $\mathbf{P} = \begin{pmatrix} \widehat{\mathbf{v}}_1 & \widehat{\mathbf{v}}_2 & \dots & \widehat{\mathbf{v}}_k \end{pmatrix}$
13: **for** $\widetilde{\mathbf{x}}_i \in \widetilde{X}$ **do**
14: $\quad \widetilde{\mathbf{z}}_i = \Phi(\widetilde{\mathbf{x}}_i)$
15: $\quad \widehat{\mathbf{z}}_i = (\widetilde{\mathbf{z}}_i - \bar{\mathbf{z}})\mathbf{P}\mathbf{P}^\top + \bar{\mathbf{z}}$
16: $\quad \widehat{\mathbf{x}}_i = \Phi^{-1}(\widehat{\mathbf{z}}_i)$
17: **end for**
18: **return** $(\widehat{\mathbf{x}}_1, \widehat{\mathbf{x}}_2, \dots)$

propres estimée. nous définissons le sous-espace engendré et la matrice de projection correspondante $\mathbf{P} : \mathbb{R}^D \to \mathbb{R}^k$, à savoir

$$\mathbf{P} = \begin{pmatrix} \widehat{\mathbf{v}}_1 & \widehat{\mathbf{v}}_2 & \cdots & \widehat{\mathbf{v}}_k \end{pmatrix}. \tag{3.6}$$

La matrice de projection $\mathbf{P}$ permet de débruiter (ou de compresser) les données dans l'espace latent, le niveau de débruitage étant paramétré par le nombre k de composantes principales retenues.

Considérons un ensemble d'échantillons $\widetilde{X} = \{\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, \dots\}$ à débruiter ou à compresser. Le débruitage d'un échantillon $\widetilde{\mathbf{x}}_i$ de $\widetilde{X}$ se résume à ces trois étapes : premièrement, intégrer l'échantillon dans l'espace latent $\mathcal{Z}$ avec $\widetilde{\mathbf{z}}_i = \Phi(\widetilde{\mathbf{x}}_i)$, puis le projeter avec la matrice de projection définie par l'équation (3.6), résultant sur $\widehat{\mathbf{z}}_i = (\widetilde{\mathbf{z}}_i - \bar{\mathbf{z}})\mathbf{P}\mathbf{P}^\top + \bar{\mathbf{z}}$ (après centrage et avant décentrage), et enfin pré-imager ce résultat pour récupérer sa projection dans l'espace des observations avec $\widehat{\mathbf{x}}_i = \Phi^{-1}(\widehat{\mathbf{z}}_i)$. En pratique, la projection peut être réalisée de manière matricielle en une seule opération pour tous les échantillons intégrés de $\widetilde{X}$.

L'algorithme correspondant à notre processus de débruitage $\mathcal{Z}$-ACP est résumé dans l'Alg. 2. Dans cet algorithme, le modèle est appris une fois en utilisant le jeu de données d'entraînement X et n'a pas besoin d'être entraîné à chaque fois que nous débruitons de nouveaux échantillons. Les lignes 7 à 12 correspondent au calcul des composantes principales, tandis que les lignes 13 à 17 correspondent au processus de débruitage d'un jeu de données $\widetilde{X}$.

Algorithm 3 Méthode de débruitage avec un NF basé ACP.

Require: Données d'apprentissage : X, Données bruitées : $\widetilde{X}$, nombre de composantes principales à garder : k

1: $(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_D) \leftarrow (3.2)$
2: $(\lambda_1 \quad \lambda_2 \quad \dots \quad \lambda_k) \leftarrow (3.3)$
3: $(\lambda_{k+1} \quad \lambda_{k+2} \quad \dots \quad \lambda_D) \leftarrow (3.4)$
4: $\boldsymbol{\Sigma} = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
5: $\boldsymbol{\mu} = \mathbf{0}$
6: $(\Phi, \, \Phi^{-1}) \leftarrow \text{TrainNF}(X, \, \boldsymbol{\mu}, \, \boldsymbol{\Sigma})$ (Alg. 1)
7: $\mathbf{P} = (\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_k)$
8: **for** $\widetilde{\mathbf{x}}_i \in \widetilde{X}$ **do**
9: $\quad \widetilde{\mathbf{z}}_i = \Phi(\widetilde{\mathbf{x}}_i)$
10: $\quad \widehat{\mathbf{z}}_i = \widetilde{\mathbf{z}}_i \mathbf{P} \mathbf{P}^\top$
11: $\quad \widehat{\mathbf{x}}_i = \Phi^{-1}(\widehat{\mathbf{z}}_i)$
12: **end for**
13: **return** $(\widehat{\mathbf{x}}_1, \, \widehat{\mathbf{x}}_2, \, \dots)$

3.2.2 Débruitage avec un NF basé ACP

La majeure partie du temps de calcul et de l'utilisation mémoire requis par notre processus de débruitage est consacrée à la décomposition en éléments propres de $\widehat{\boldsymbol{\Sigma}}$ (ligne 11 dans l'Alg. 2). En effet, notre précédente méthode (Alg. 2) opère séparément les deux différents processus correspondant à l'apprentissage du NF et la projection par ACP. Cependant, l'apprentissage du NF nécessite une matrice de covariance pour caractériser la distribution cible des données encodées. Dans ce qui suit, nous proposons d'intégrer la projection ACP dans le modèle de NF afin d'alléger l'étape de décomposition en éléments propres de la méthode de débruitage proposée dans l'Alg. 2. Pour ce faire, nous définissons a priori la matrice de covariance $\boldsymbol{\Sigma}$ qui, utilisée à la fois dans l'apprentissage du NF et dans la projection ACP, permet d'obtenir une transformation de NF avec projection intégrée.

Si l'on considère une telle configuration, la matrice de projection $\mathbf{P}$ appliquée dans $\mathcal{Z}$ peut être directement déduite de l'ensemble des vecteurs propres $\mathbf{v}_i$ utilisés pour construire la matrice de covariance. L'influence de chaque vecteur propre $\mathbf{v}_i$ est contrôlée par la valeur propre associée λ_i. En ne conservant que les k valeurs propres les plus importantes, la matrice de projection $\mathbf{P}$ utilisée pour le débruitage dans l'espace latent $\mathcal{Z}$ est alors définie comme suit :

$$\mathbf{P} = \begin{pmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_k \end{pmatrix}$$

L'Alg. 3 montre le processus de débruitage à l'aide de notre approche de NF basé ACP, mettant en évidence que la décomposition en éléments propres de $\boldsymbol{\Sigma}$ n'est plus nécessaire. Notez que le temps de calcul nécessaire à l'inférence par Φ est insignifiant par rapport au temps qui était nécessaire dans l'Alg. 2 pour la décomposition en éléments propres, en particulier pour les données de haute dimension.

Sous l'hypothèse que notre modèle est capable d'apprendre une bonne représentation de la variété sous-jacente des données, l'approche proposée dans cette sous-section peut reconstruire une version débruitée des données grâce à la résolution de

la pré-image fournie par Φ^{-1}. En suivant la même idée, nous pouvons étendre le formalisme proposé à d'autres techniques de débruitage basées sur d'autres définitions de variété.

3.3 Expérimentations

Dans cette section, nous proposons d'évaluer notre contribution à l'apprentissage machine sans problème de pré-image. Après une description du protocole utilisé (Section 3.3.3), les expériences menées visent à répondre aux questions suivantes :

Q1 L'ACP réalisée dans l'espace de caractéristiques est-elle une bonne procédure de débruitage ? (Section 3.3.4)

Q2 Notre modèle est-il toujours un bon modèle génératif ? (Section 3.3.5)

Les réponses à ces questions permettent d'évaluer les avantages et les limites du formalisme proposé.

3.3.1 Jeux de données

Les expériences de débruitage ont été réalisées sur divers types d'ensembles de données, à savoir un jeu de données de démonstration ainsi qu'un jeu de données plus complexe composé d'images.

Jeu de données *single moon*. *Single moon* est un jeu de données de démonstration généré à l'aide de la bibliothèque scikit-learn [100] composé de 1000 échantillons de 2 dimensions suivant une forme de lune non-linéaire.

Jeu de données *MNIST3*. Le jeu de données *MNIST3*, utilisé dans ces expérimentations, est basé sur MNIST [31] et se compose de 7141 images de taille 28×28 pixels représentant le chiffre 3.

3.3.2 Configuration

Comme notre approche repose sur l'utilisation d'un NF, nous suggérons de tester notre méthode avec trois NF différents, afin de mettre en évidence la polyvalence de notre proposition. Les deux premiers modèles visent à traiter des données vectorielles. Le premier modèle utilisé est **RealNVP** [33] constitué d'une séquence de 30 blocs composés chacun d'une couche de couplage affine et d'une couche de normalisation par lot. Le second est **FFJORD** [41] utilisant une approche de CNF. Enfin, pour traiter les données de type image, nous proposons d'utiliser un NF appelé **Glow** [60]. Afin de rendre les modèles plus robustes au bruit, la procédure d'entraînement est réalisée sur des données augmentées d'un bruit gaussien centré avec un écart-type de 0.5 pour les données *MNIST3* et 0.2 pour les données *single moon*.

Pour montrer l'efficacité de notre proposition du point de vue de l'apprentissage machine, nous comparons notre approche sans problème de pré-image à différentes méthodes classiques. Pour cela, nous nous comparons aux méthodes d'ACP à noyau, apportant ainsi de la non-linéarité par l'utilisation de différents noyaux. Nous avons

testé un noyau linéaire (**Linéaire**), un noyau gaussien (**RBF**), un noyau polynomial (**Polynomial**) et un noyau **Sigmoïde** (voir la Table 1.1 pour leurs définitions). Le réglage des hyperparamètres est décrit dans la section suivante. De plus, nous évaluons la capacité à générer des pré-images.

3.3.3 Protocole

Dans cette section, nous décrivons le protocole utilisé dans les expériences à des fins de reproductibilité. Pour évaluer la capacité de généralisation des méthodes proposées, nous avons divisé les jeux de données en ensembles d'entraînement et de test. L'ensemble d'entraînement représente 90% des données et est utilisé pour apprendre les paramètres du modèle. Le reste des données disponibles, soit 10% des données, constitue l'ensemble de test et est utilisé pour évaluer les performances du modèle.

Afin d'utiliser des noyaux efficaces, nous ajustons leurs paramètres à l'aide d'une grille de recherche et d'un protocole de validation croisée. Nous associons à chaque paramètre une plage de valeurs et ajustons les paramètres des différents noyaux à l'aide de l'ensemble de paramètres correspondant :

Linéaire : Paramètre de régularisation λ : 10 valeurs de 10^{-5} à 10^{3}.

RBF : Paramètre de régularisation λ : 10 valeurs de 10^{-5} à 10^{3}, γ : 10 valeurs de 10^{-5} à 10^{3}.

Polynomial : Paramètre de régularisation λ : 10 valeurs de 10^{-5} à 10^{3}, γ : 10 valeurs de 10^{-5} à 10^{3}, puissance n : 4 valeurs de 1 à 4.

Sigmoïde : Paramètre de régularisation λ : 10 valeurs de 10^{-5} à 10^{3}.

À l'exception des valeurs de lu coefficient de puissance n pour le noyau **Polynomial**, toutes les valeurs des paramètres sont échantillonnées selon une échelle logarithmique.

3.3.4 Expérimentations de débruitage

Dans cette section, nous proposons d'évaluer les performances de notre approche de débruitage décrite dans la Section 3.2. Ces expériences permettent de répondre à la question **Q1** grâce aux deux variantes proposées, à savoir $\mathcal{Z}$-ACP (Alg. 2) et NF basé ACP (Alg. 3).

Débruitage sur un jeu de données de démonstration. Tout d'abord, nous menons les expériences sur le jeu de données *single moon* et définissons $\mathbf{\Sigma}$ suivant l'équation 3.1 avec les vecteurs propres $(\mathbf{v}_1, \mathbf{v}_2) = ([1, 0], [0, 1])$ et les valeurs propres $(\lambda_1, \lambda_2) = (50, 0.002)$, favorisant ainsi une dimension pour répartir les données dans l'espace $\mathcal{Z}$. L'expérience est conçue comme suit : une donnée $\mathbf{x} \in \mathcal{X}$ est modifiée par un bruit gaussien additif $\varepsilon \sim \mathcal{N}(0, 0.2)$ pour construire une version bruitée $\widetilde{\mathbf{x}} = \mathbf{x} + \varepsilon$. Ensuite, $\widetilde{\mathbf{x}}$ suit le processus de débruitage pour reconstruire $\widehat{\mathbf{x}}$ comme décrit dans Alg. 2 et 3. Les performances de débruitage sont alors mesurées en calculant la distance entre les données originales et leur version débruitée, données par $\| \mathbf{x} - \widehat{\mathbf{x}} \|_2$. Les distances obtenues par différentes méthodes de pré-image sont présentées dans

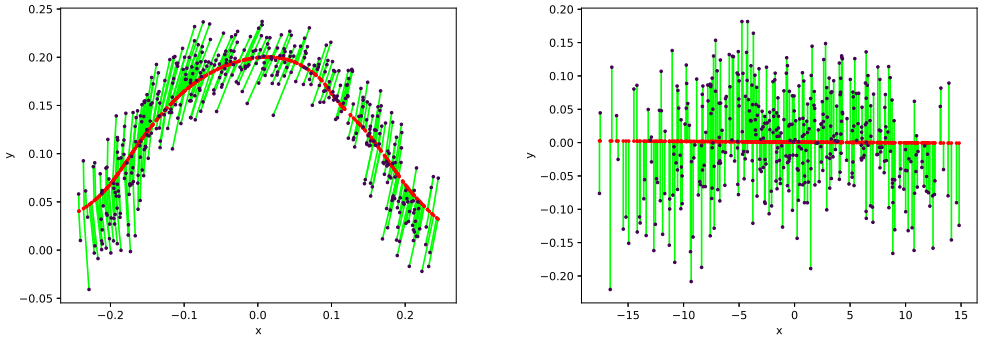
TABLE 3.1 – Évaluation dans une configuration de débruitage et comparaison avec les méthodes d'ACP à noyau en calculant la distance $\|\mathbf{x} - \hat{\mathbf{x}}\|_2$ entre les données débruitées $\hat{\mathbf{x}}$ et les données originales $\mathbf{x}$.

	Méthodes		Jeu de données
			Single moon
	Linéaire	K-ACP	0.1113
	RBF	K-ACP	0.0678
	Polynomial	K-ACP	0.0712
	Sigmoïde	K-ACP	0.2120
Nos approches	RealNVP	$\mathcal{Z}$-ACP	**0.0216**
	RealNVP	NF basé ACP	0.0669
	FFJORD	$\mathcal{Z}$-ACP	0.0336
	FFJORD	NF basé ACP	0.0339

la Table 3.1. Les pré-images obtenues avec les méthodes à noyaux, décrites dans la section précédente, sont calculées avec la méthode de Bakır et al. [5]. Cette première expérience montre comment l'utilisation de l'espace de caractéristiques $\mathcal{Z}$ aide à reconstruire une version débruitée précise des données, en utilisant l'une ou l'autre des deux architectures NF. Bien que le NF basé ACP permet de gagner du temps de calcul, il peut entraîner une légère perte de précision. Cependant, cette méthode demeure plus performante que les techniques de débruitage basées sur des noyaux linéaires ou non-linéaires. Cette légère différence de précision peut s'expliquer par le fait que l'apprentissage du NF étudié ne s'adapte pas parfaitement à la matrice de covariance cible, l'estimation empirique de cette matrice étant plus précise.

De plus, nous cherchons à déterminer s'il existe des différences significatives dans les performances de notre approche de débruitage par rapport aux autres. À cette fin, nous calculons les p-valeurs à l'aide du test H de Kruskal-Wallis [68], permettant la comparaison des distributions des performances des différentes méthodes de débruitage. Pour rejeter l'hypothèse nulle, indiquant qu'il n'y a pas de différence de performances entre les méthodes, nous exigeons que les p-valeurs soient inférieures à 0.05. En effectuant 20 évaluations différentes pour chaque méthode de débruitage en utilisant un bruit aléatoire, les p-valeurs obtenues pour l'approche NF basé ACP et l'approche $\mathcal{Z}$-ACP sur le jeu de données *single moon* sont proches de 10^{-9}, nous permettant de rejeter l'hypothèse nulle.

Enfin, comme nous travaillons en 2D, nous pouvons illustrer les données à la fois dans l'espace des observations et dans l'espace de caractéristiques. La Figure 3.3 montre les données dans les deux espaces, l'espace des observations $\mathcal{X}$ et l'espace de caractéristiques $\mathcal{Z}$ appris par le NF FFJORD. De plus, la figure montre les données d'apprentissage en noir et leurs projections à l'aide de $\mathcal{Z}$-ACP en rouge, c'est-à-dire des projections linéaires dans l'espace de caractéristiques qui se traduisent par des transformations non-linéaires dans l'espace des observations par application de la transformation inverse Φ^{-1}. Le lien entre chaque échantillon et sa projection est mis en évidence en vert.



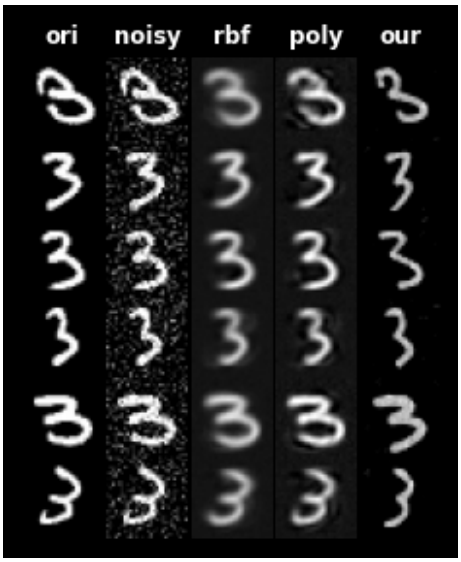
(a) Input space $\mathcal{X}$ (b) Feature space $\mathcal{Z}$

FIGURE 3.3 – (a) Pré-images (points rouges) générées par les projections $\mathcal{Z}$-ACP à partir du jeu de données *single moon* (points noirs). (b) Ces points du jeu de données et leurs projections linéaires (points rouges) dans l'espace de caractéristiques $\mathcal{Z}$ obtenu par la transformation FFJORD apprise Φ.

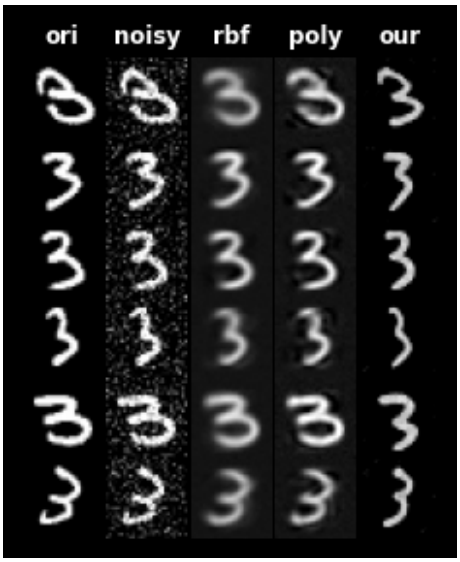
Débruitage d'images. La capacité de débruitage ayant été validée sur un jeu de données de démonstration, nous expérimentons maintenant ce processus sur des images réelles tirées du jeu de données *MNIST3*. Étant donné que ce jeu de données est constitué d'images, il n'est pas idéal de comparer les distances de la même manière que précédemment avec les données de *single moon*. En effet, les pré-images obtenues peuvent être de mauvaise qualité visuelle malgré une faible distance par rapport à l'original. Nous évaluons donc les modèles de débruitage d'images d'un point de vue qualitatif.

La Figure 3.4 montre différents résultats de débruitage en utilisant à la fois les méthodes basées sur les noyaux ainsi que notre approche proposée dans ses deux versions ($\mathcal{Z}$-ACP et NF basée ACP). Nous pouvons noter que les versions de débruitage basées sur les noyaux souffrent de certains artefacts dus à l'espace de caractéristiques utilisé par le noyau. Inversement, les versions obtenues par nos méthodes basées sur l'utilisation de NF rendent une version propre de chaque échantillon testé.

Compression d'images. Comme l'approche que nous proposons repose sur l'ACP, elle est naturellement capable d'effectuer à la fois des tâches de débruitage et de compression. Même si les représentations dans $\mathcal{Z}$ ont des dimensions égales, l'utilisation des composantes principales permet la compression des données. Par conséquent, nous évaluons notre approche en générant des représentations compressées dans $\mathcal{Z}$ par projection sur les composantes principales. La Figure 3.5 montre la génération des méthodes $\mathcal{Z}$-ACP et NF basé ACP utilisant des représentations compressées dans $\mathcal{Z}$ à différents niveaux de compression. Malgré des niveaux de compression élevés, les deux méthodes produisent systématiquement des images similaires à l'image originale non compressée. Plus précisément, la deuxième colonne de la figure représente la compression de la représentation de l'image originale dans $\mathcal{Z}$ à une seule dimension, tandis que la dernière colonne correspond à une compression à k dimensions. En outre, comme pour les expériences de débruitage *MNIST3*,

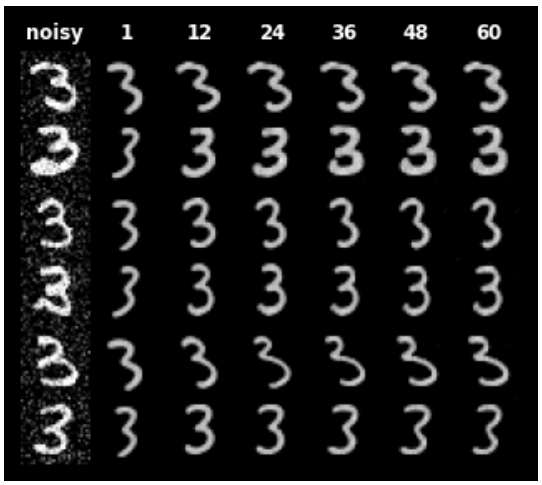


(a) Débruitage $\mathcal{Z}$-ACP

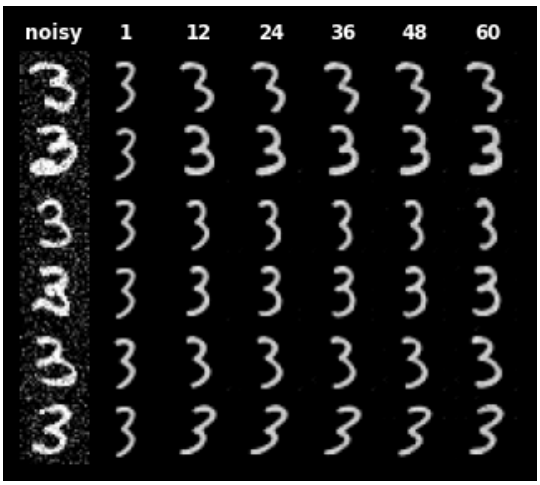


(b) Débruitage NF basé ACP

FIGURE 3.4 – La première colonne représente les images originales et la deuxième colonne les images avec un bruit gaussien additif. (a) Les colonnes suivantes représentent les résultats du débruitage à l'aide de RBF-ACP, de Polynomial-ACP et de $\mathcal{Z}$-ACP (b) Les colonnes suivantes représentent les résultats du débruitage à l'aide de RBF-ACP, de Polynomial-ACP et de NF basé ACP.



(a) Compression $\mathcal{Z}$-ACP



(b) Compression NF basé ACP

FIGURE 3.5 – Pré-images générées à différents niveaux de compression dans $\mathcal{Z}$. Les premières colonnes représentent les images originales et les 6 colonnes suivantes représentent les pré-images ordonnées de la compression la plus élevée à gauche à la compression la plus faible à droite. Les niveaux de compression sont échantillonnés linéairement de 1 à k.

l'utilisation d'un NF basé ACP permet de reconstruire des images similaires aux images moyennes de chaque classe. Cela contraste avec $\mathcal{Z}$-ACP, qui préserve la va-

FIGURE 3.6 – Pré-images obtenues en échantillonnant des vecteurs dans $\mathcal{Z}$ selon la distribution gaussienne associée utilisée lors de l'apprentissage du modèle NF.

riabilité de l'image originale dans l'image compressée.

3.3.5 Qualité de pré-image

Les deux expériences précédentes démontrent les bonnes performances du formalisme proposé sur des tâches de débruitage et de compression. Dans cette expérience, nous visons à répondre à la question **Q2** et démontrer la capacité générative de notre approche, montrant ainsi que le problème de pré-image est implicitement résolu. À cette fin, nous examinons notre approche de résolution de pré-image sur *MNIST3*, permettant une appréciation visuelle de la qualité de la pré-image calculée. Les pré-images sont obtenues en prenant un échantillon dans l'espace de caractéristiques $\mathcal{Z}$ et en le faisant passer par Φ^{-1} (voir Section 1.2).

Ainsi, la résolution de la pré-image permet l'exploration de l'espace de caractéristiques en pré-imageant n'importe quel point de $\mathcal{Z}$, grâce à l'inversibilité de Φ. La Figure 3.6 montre les pré-images de 49 échantillonnages aléatoires appliqués à la distribution gaussienne utilisée lors de l'entraînement du modèle. De plus, la Figure 3.7 illustre les pré-images de points échantillonnées par l'interpolation de représentation d'images existantes dans $\mathcal{Z}$. Comme nous pouvons le constater dans ces deux figures, les pré-images calculées sont, comme prévu, des représentations visuelles cohérentes du chiffre 3, démontrant la bonne capacité de génération de notre approche.

Nos expériences montrent que notre approche est capable de générer des pré-images cohérentes pour des points de l'espace $\mathcal{Z}$ qui ne font pas partie du jeu de données, permettant ainsi de répondre positivement à la question **Q2**.

FIGURE 3.7 – Illustration des pré-images d'interpolation dans $\mathcal{Z}$. Chaque ligne illustre deux images sélectionnées (l'image la plus à gauche et l'image la plus à droite) et les pré-images de 10 interpolations entre les représentations de ces deux images dans l'espace latent.

3.4 Conclusion

Dans ce chapitre, nous avons proposé une contribution à la conception d'un formalisme d'apprentissage machine sans problème de pré-image utilisant une architecture de NF. Cette dernière peut être utilisée pour générer un meilleur espace de représentation $\mathcal{Z}$ où des opérations et des méthodes linéaires peuvent être appliquées. Nos expériences montrent la capacité de notre contribution à obtenir de bonnes performances dans les tâches de débruitage/compression en utilisant des méthodes simples dans l'espace de caractéristiques, démontrant ainsi comment l'espace généré par le NF peut être utilisé pour exploiter des méthodes d'apprentissage machine. De plus, grâce à l'architecture du NF, notre modèle évite le problème de pré-image. En combinant ces deux faits, notre proposition constitue une première démonstration de la façon dont un NF peut être bénéfique pour construire des modèles d'apprentissage machine non-linéaires qui ne souffrent pas de la malédiction de la pré-image.

Notre formalisme offre une grande flexibilité dans le but de modéliser un espace de caractéristiques en fonction des besoins spécifiques. Par exemple, certaines tâches prédictives courantes telles que la classification et la régression peuvent nécessiter des spécificités qui ne sont pas prises en compte par l'entraînement non supervisé. Par conséquent, en utilisant ce formalisme comme base, il est possible de proposer des approches supervisées spécifiques à chaque besoin comme présentées au chapitre suivant.

CHAPITRE 4

Méthodes supervisées insensibles au problème de pré-image

4.1 Introduction

En apprentissage supervisé, le modèle est entraîné à partir d'un ensemble de données d'entraînement où chaque exemple est étiqueté. Ces étiquettes associées à chaque donnée observée peuvent représenter des valeurs quantitatives réelles pour une tâche de régression ou des classes pour une tâche de classification. L'objectif de l'apprentissage supervisé est de créer un espace de caractéristiques où les représentations des données sont organisées en fonction de leurs valeurs étiquetées permettant de bonnes performances de prédiction. Ainsi, le modèle est entraîné de manière à pouvoir généraliser à de nouvelles données non étiquetées et de permettre l'obtention de prédictions par la transformation de ces nouvelles données dans l'espace de caractéristiques. Contrairement à l'apprentissage non supervisé, sachant que chaque donnée est associée à une valeur d'étiquette cible, l'apprentissage supervisé consiste le plus souvent en la minimisation d'une fonction de perte directe qui mesure la différence entre la prédiction faite par le modèle et la vérité terrain que l'on souhaite générer.

Lorsque généralement, les tâches prédictives associées à un tel apprentissage ne cherchent pas à obtenir l'inverse de la transformation appliquée par le modèle, il peut être intéressant de résoudre le problème de pré-image (voir Section 1.2) pour améliorer aussi bien l'interprétabilité des résultats que du modèle. En effet, en utilisant la transformation inverse du modèle, il devient possible de générer de nouvelles données en fonction de leurs valeurs d'étiquettes. Par exemple, dans le cas d'une tâche de classification, cette transformation permettrait de générer des exemples spécifiques à chaque classe, tels que des prototypes représentatifs. De même, dans le cas d'une tâche de régression, la résolution du problème de pré-image permettrait une génération de nouvelles données à partir d'une valeur d'étiquette quantitative souhaitée.

Dans ce chapitre, nous proposons des modèles de prédiction interprétables, en revisitant notre formalisme proposé dans le Chapitre 3 précédent, en concevant

deux méthodologies spécifiques à une tâche particulière. Nous présentons ces deux approches distinctes non affectées par le problème de pré-image : l'une pour des tâches de classification et l'autre pour des tâches de régression. Contrairement aux méthodes utilisant des noyaux et aux réseaux de neurones, où la transformation non-linéaire n'est généralement pas inversible, nos méthodes permettent de surmonter le problème de pré-image en proposant une fonction de transformation inversible par construction. Le but de l'apprentissage du modèle est de générer un espace de caractéristiques favorisant la séparation linéaire (pour une tâche de classification) ou l'organisation linéaire (pour une tâche de régression) des échantillons afin d'améliorer les performances des méthodes de prédiction linéaire appliquées dans cet espace. De plus, la transformation inverse peut être utilisée pour générer des pré-images de tout échantillon intéressant de l'espace de caractéristiques.

Notre transformation non-linéaire réversible s'appuie sur les importants progrès des modèles génératifs dans le domaine de l'apprentissage machine. La Section 1.1 introduit les différentes méthodes étudiées, à savoir les VAE, GAN et NF. Basé sur le formalisme du chapitre précédent, nous suggérons l'application des NF pour la définition de notre transformation non-linéaire inversible.

Pour démontrer l'efficacité de nos approches dans nos expériences, nous employons les trois mêmes architectures de NF que celles utilisées dans le Chapitre 3 précédent. Les deux premières architectures, utilisées pour traiter des données vectorielles, appartiennent à deux familles de NF différentes : RealNVP [33], reposant sur des couches de couplage, et FFJORD [41], un CNF. La troisième, nommée Glow [60], est utilisée pour les expériences menées sur des données de type image. Ces modèles sont introduits dans la Section 1.3.4. Nous évaluons notre approche sur des benchmarks reconnus montrant sa pertinence pour les tâches de classification et de régression tout en générant un bon espace de représentation sans problème de pré-image.

Le chapitre est organisé comme suit. Dans un premier temps, notre contribution est présentée dans la Section 4.2, qui est séparée en deux parties distinctes. La Section 4.2.1 introduit notre approche pour la résolution de tâche de classification, tandis que la Section 4.2.2 décrit notre méthodologie de régression. Ces deux parties comprennent plusieurs sous-sections qui détaillent l'entraînement du modèle de NF, l'application d'opérations dans l'espace de caractéristiques à des fins de prédiction ainsi que les processus de génération de pré-images. Enfin, les deux sections suivantes sont composées d'une section décrivant les expériences menées (Section 4.3) et d'une section de discussion (Section 4.4) apportant des précisions sur nos approches et expériences. Enfin, la conclusion est présentée dans la Section 4.5.

4.2 Approches proposées

Dans cette section, nous proposons d'adapter notre formalisme non supervisé présenté dans le chapitre précédent (Section 3.2) pour des tâches de classification et de régression. Pour cela, nous proposons une spécification de l'approche pour les tâches mentionnées en supervisant les apprentissages associées. De la même manière que précédemment, nous travaillons avec une transformation inversible apprise par un modèle de NF (Section 1.3.4) et utilisons l'espace de caractéristiques résultant

comme espace d'intérêt pour appliquer des méthodes d'apprentissage machine.

L'idée consiste en trois étapes :

(*i*) générer une représentation latente des données en utilisant un NF en supervisant son apprentissage pour la tâche spécifique,

(*ii*) effectuer des opérations linéaires ou appliquer des modèles linéaires dans l'espace de caractéristiques $\mathcal{Z}$,

(*iii*) enfin, il est possible de calculer simplement la pré-image de n'importe quel point de l'espace $\mathcal{Z}$ en utilisant la propriété d'inversibilité de la fonction de transformation apprise par le modèle de NF.

La suite de cette section est divisée en deux parties, chacune présentant une approche supervisée pour la résolution d'une tâche de classification ou de régression tout en exploitant la structure inversible du modèle permettant la génération de pré-image facilement.

4.2.1 Modèle de classification génératif

De nombreuses méthodes d'apprentissage machine consistent à définir un modèle prédictif simple (linéaire) sur une représentation apprise des données dans un espace donné. La classification est la tâche la plus couramment utilisée parmi les différentes tâches de prédiction existantes. Dans cette section, nous proposons d'apprendre une représentation latente linéairement séparable des données grâce à un modèle de NF. Dans ces conditions, le problème de la définition d'un classifieur non-linéaire exempt du problème de pré-image se résume à l'apprentissage d'une représentation des données linéairement séparable.

Apprentissage du modèle

Dans les modèles de NF classiques, les données sont encodées de manière que la distribution dans l'espace latent soit conforme à une densité de probabilité souhaitée, à savoir une densité gaussienne. Cependant, avec une telle configuration, les données ne favorisent pas une répartition en fonction de leurs classes. Dans cette section, nous proposons une formulation de l'apprentissage du NF tenant compte des classes, en modifiant la fonction objective du NF afin d'encoder les données dans différentes régions de l'espace des caractéristiques, en fonction des classes associées aux données. Ainsi, cette formulation du NF tenant compte des classes construit une transformation Φ qui améliore la séparation linéaire des données dans l'espace de caractéristiques.

Considérons un jeu de données $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\}$ avec $\mathbf{x}_i \in \mathcal{X}$ les données observées et $y_i \in \{1 \dots C\}$ codant sa classe parmi les C classes. Ici, nous associons à chaque classe $c \in \{1 \dots C\}$ une distribution gaussienne dans $\mathcal{Z}$ définie par :

$$P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma}_c)}} e^{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{z}-\boldsymbol{\mu}_c))},$$

où la distribution associée à chaque classe c est définie par sa matrice de covariance $\boldsymbol{\Sigma}_c$ et sa moyenne $\boldsymbol{\mu}_c$. Ensuite, le modèle de NF doit être adapté de telle sorte que

chaque donnée $\mathbf{x}_i$ soit ajustée à la distribution gaussienne correspondante en fonction de la classe c codée par y_i. Par conséquent, le terme de log-probabilité utilisé dans la fonction de perte d'apprentissage (équation (1.8)) est alors défini comme suit :

$$\begin{aligned}\log P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = &- \frac{1}{2}\left(D\log(2\pi) + (\mathbf{z}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{z}-\boldsymbol{\mu}_c)\right) \\ &- \log(\det(\boldsymbol{\Sigma}_c)).\end{aligned} \tag{4.1}$$

Grâce à cette fonction objectif, toutes les données correspondant à une classe sont ajustées à une région particulière de l'espace de caractéristiques $\mathcal{Z}$, ce qui permet d'obtenir une représentation linéaire séparable entre les différentes classes.

Nous pouvons noter que, selon le jeu de données, certaines paires de classes peuvent être considérées comme plus similaires que d'autres. Pour encoder cette particularité, il peut être intéressant de pouvoir apprendre l'emplacement des différentes distributions gaussiennes, c'est-à-dire les moyennes $\boldsymbol{\mu}_c$ associées à chaque classe. Pour apprendre les moyennes $\boldsymbol{\mu}_c$, nous proposons d'ajouter un nouveau terme de perte à la fonction objective du NF afin d'encourager le modèle à séparer les distributions. Ainsi, notre problème d'optimisation sera composé de deux termes de perte. Le premier correspond à la fonction de log-vraisemblance d'un échantillon

$$\mathcal{L}_{\text{nf}}(\mathbf{x}_i, \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) = \log P_{\mathcal{Z}}(\Phi(\mathbf{x}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) + \log\left|\det\left(\frac{\partial\Phi(\mathbf{x}_i)}{\partial\,\mathbf{x}_i}\right)\right|,$$

où $\log P_{\mathcal{Z}}(\Phi(\mathbf{x}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$ se réfère à l'équation (4.1). Le second terme est le terme de séparation des distributions gaussiennes

$$\mathcal{L}_{\mu} = \log\left(1 + \frac{1}{C^2}\sum_{i=1}^{C}\sum_{j=1}^{C} \| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \|_2^2\right),$$

où $\boldsymbol{\mu}_i$ et $\boldsymbol{\mu}_j$ désignent les moyennes associées aux classes i et $j \in \{1 \dots C\}$. La maximisation de ce terme favorise la séparation des centres des distributions.

Compte tenu de ces deux termes, la fonction finale à minimiser pour une paire de données et d'étiquettes est définie comme suit :

$$\mathcal{L}(\mathbf{x}_i, y_i) = \mathcal{L}_{\text{nf}}(\mathbf{x}_i, \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) - \beta\mathcal{L}_{\mu}, \tag{4.2}$$

où β est un hyperparamètre de compromis qui équilibre les deux termes de perte. La minimisation d'une telle fonction encouragera le modèle à (i) séparer les différentes distributions associées à chaque étiquette grâce au terme de perte $\mathcal{L}_{\mu}$ et à (ii) minimiser le terme de log-vraisemblance $\mathcal{L}_{\text{nf}}$ de telle sorte que la distribution des données cibles soit proche de la distribution gaussienne spécifiée.

En plus de différencier les distributions par leurs moyennes, nous associons différentes dimensions de $\mathcal{Z}$ à différentes classes. Pour ce faire, nous calculons le nombre maximal de dimensions à attribuer à chaque classe avec $k = \lfloor \frac{D}{C} \rfloor$ où $\lfloor . \rfloor$ représente l'opération prenant un nombre réel en entrée et renvoie le plus grand entier inférieur ou égal à ce réel. Ensuite, pour chaque classe c, nous associons k valeurs propres à une valeur élevée, permettant l'extension de chaque distribution gaussienne sur différentes dimensions, à savoir

$$(\underbrace{\lambda_1, \lambda_2, \dots, \lambda_k}_{c=1}, \underbrace{\lambda_{k+1}, \lambda_{k+2}, \dots, \lambda_{2k}}_{c=2}, \dots, \lambda_D) \tag{4.3}$$

De plus, les $D - k$ valeurs propres restantes, pour chaque classe, sont fixées à ϵ, en suivant la même analogie que dans (3.4). Ainsi, chaque classe possède sa propre matrice de covariance $\mathbf{\Sigma}_c$ construite conformément à l'équation (3.1) en utilisant les valeurs propres attribuées et les vecteurs propres correspondants, à savoir pour $c \in \{1 \dots C\}$

$$\mathbf{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i \, .$$

où les vecteurs propres $\mathbf{v}_i$ sont définis comme étant un ensemble de vecteurs orthogonaux en utilisant l'équation (3.2) quelle que soit la classe.

En considérant Θ comme l'ensemble des paramètres composant Φ, nous définissons Ω comme l'ensemble des paramètres optimisables, c'est-à-dire

$$\Omega = \{\Theta, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_C\}.$$

Nous estimons donc l'ensemble des paramètres Ω à l'aide d'une stratégie stochastique de descente de gradient en minimisant la fonction de perte $\mathcal{L}(\mathbf{x}_i, y_i)$ (équation (4.2)) sur un lot de m échantillons sélectionnés aléatoirement dans le jeu de données d'apprentissage à chaque itération. Pour cela, nous appliquons

$$\Omega \leftarrow \Omega - \eta \sum_{k \in I} \nabla_\Omega \mathcal{L}(\mathbf{x}_k, y_k) \tag{4.4}$$

où η représente le taux d'apprentissage et I définit le lot de données sélectionné aléatoirement à l'itération correspondante. Le processus d'apprentissage de notre approche de classification est présenté dans l'Alg. 4.

Classifieur dans l'espace de caractéristiques

Ce processus d'apprentissage conduit à un espace de caractéristiques $\mathcal{Z}$ où les données sont linéairement séparables. Nous pouvons alors définir un modèle prédictif $f : \mathcal{X} \to \{1, \dots, C\}$ en apprenant un classifieur linéaire $g : \mathcal{Z} \to \{1, \dots, C\}$ dans cet espace de représentation $\mathcal{Z}$ en utilisant des machines à vecteurs de support (SVM) ou une régression logistique. L'inférence est ensuite réalisée par la combinaison de Φ pour transformer les données dans l'espace $\mathcal{Z}$ et de g pour dériver une classe prédite :

$$f(\mathbf{x}) = g(\Phi(\mathbf{x})). \tag{4.5}$$

Génération de pré-image

Grâce à l'existence de Φ^{-1}, la pré-image de n'importe quel point d'intérêt de l'espace de caractéristiques peut être calculée, fournissant ainsi un algorithme de classification sans problème de pré-image. Ainsi, nous pouvons générer la pré-image de points échantillonnés dans $\mathcal{Z}$ appartenant à l'une des distributions de classe afin de générer une donnée de la classe souhaitée. Ceux-ci peuvent être tirés aléatoirement en utilisant les paramètres de la distribution gaussienne correspondante ($\boldsymbol{\mu}_c$, $\mathbf{\Sigma}_c$) ou calculés pour représenter un prototype de classe. L'Alg. 5 montre la procédure de génération de pré-image à partir d'un échantillon tiré aléatoirement sachant une classe donnée.

La Figure 4.1 résume les différents blocs de cette approche.

Algorithm 4 Procédure d'entraînement d'un NF pour une tâche de classification.

Require: Ensemble d'apprentissage : $\mathcal{D}$, coefficient de compromis : β, taux d'apprentissage : η, taille de lot : m, nombre d'époques : E

$(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_D) \leftarrow$ équation (3.2)
for $c \in \{1 \dots C\}$ **do**
  $(\lambda_1 \quad \lambda_2 \quad \dots \quad \lambda_D) \leftarrow$ équation (4.3)
  $\mathbf{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
end for
$(\boldsymbol{\mu}_1 \quad \dots \quad \boldsymbol{\mu}_C) = (\mathbf{0} \quad \dots \quad \mathbf{0})$
$(\Theta, \Phi, \Phi^{-1}) \leftarrow$ Initialisation du modèle de NF
$\Omega = \{\Theta, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_C\}$
for $e \in \{1, \dots, E\}$ **do**
  Mélange aléatoire du jeu de données $\mathcal{D}$
  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\} \leftarrow \mathcal{D}$
  for $i \in \{1, \dots, N/m\}$ **do**
    $\mathcal{L}_\mu = -\log\left(1 + \frac{1}{C^2} \sum_{i=1}^{C} \sum_{j=1}^{C} \| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \|_2^2\right)$
    for $k \in \{m(i-1)+1, \dots, mi\}$ **do**
      $\mathbf{z}_k = \Phi(\mathbf{x}_k)$
      $\mathcal{L}_{\text{nf}} = -\log P_{\mathcal{Z}}(\mathbf{z}_k, \boldsymbol{\mu}_{y_k}, \mathbf{\Sigma}_{y_k}) - \log \left|\det\left(\frac{\partial \mathbf{z}_k}{\partial \mathbf{x}_k}\right)\right|$
      $\mathcal{L}(\mathbf{x}_k, y_k) = \mathcal{L}_{\text{nf}} + \beta \mathcal{L}_\mu$
    end for
    $\Omega \leftarrow \Omega - \eta \sum_{k=m(i-1)+1}^{mi} \nabla_\Omega \mathcal{L}(\mathbf{x}_k, y_k)$
    $(\Phi, \Phi^{-1}) \leftarrow$ Mise à jour des fonctions avec les paramètres Ω
  end for
end for
return $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_1 \quad \dots \quad \boldsymbol{\mu}_C))$

4.2.2 Modèle de régression génératif

La régression est une autre tâche de prédiction fréquemment utilisée. Dans cette section, nous proposons l'apprentissage d'un espace de caractéristiques où les représentations latentes des données sont organisées linéairement à l'aide d'un modèle de NF. De la même manière que dans un contexte de classification, le défi de la construction d'un modèle prédictif non-linéaire ne souffrant pas du problème de pré-image se traduit par l'apprentissage d'une bonne représentation linéaire des données.

Apprentissage du modèle

Les modèles de NF classiques cherchent à générer un espace latent où les données suivent dans une distribution gaussienne. Cependant, une telle configuration n'est pas pertinente pour une tâche de régression, car les données ne sont pas nécessairement organisées en fonction de leurs étiquettes quantitatives. Nous proposons donc une nouvelle formulation supervisée de l'apprentissage du NF prenant en compte les valeurs quantitatives associées aux données. À cette fin, nous modifions la fonction objective du NF afin d'encoder les données en fonction de leurs étiquettes quantitatives, concevant ainsi un NF permettant la représentation linéaire des données à

Algorithm 5 Procédure de génération de pré-image d'une classe c avec un NF entraîné pour une tâche classification.

Require: Modèle NF : $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_C))$, classe : c

1: $(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \ldots \quad \mathbf{v}_D) \leftarrow$ équation (3.2)
2: $(\lambda_1 \quad \lambda_2 \quad \ldots \quad \lambda_D) \leftarrow$ équation (4.3)
3: $\boldsymbol{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
4: $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
5: $\hat{\mathbf{x}} = \Phi^{-1}(\mathbf{z})$
6: **return** $\hat{\mathbf{x}}$

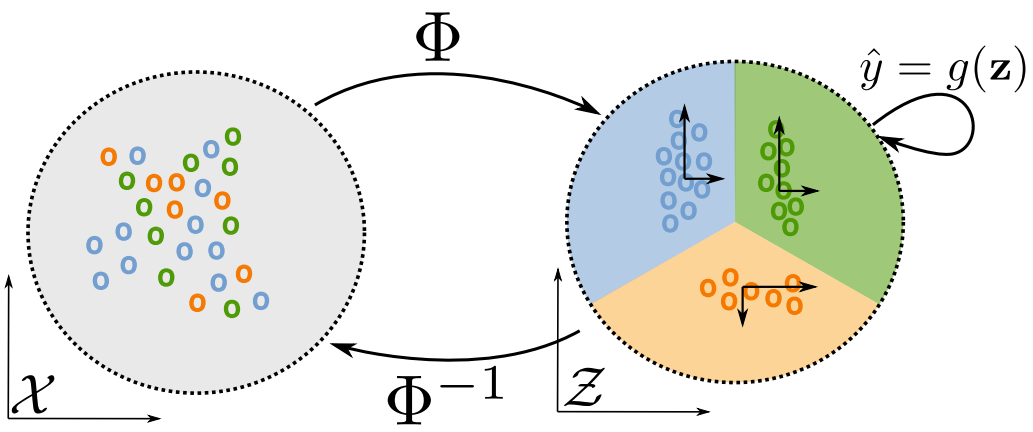


FIGURE 4.1 – Illustration de notre approche de classification pour un cas multi-classe. Le processus d'apprentissage vise à séparer linéairement les données dans l'espace des caractéristiques $\mathcal{Z}$. Un classifieur linéaire $g : \mathcal{Z} \to \mathcal{Y}$ détermine la classe de chaque donnée. La pré-image de tout point $\mathbf{z} \in \mathcal{Z}$ peut être retrouvée grâce à Φ^{-1}.

des fins de régression dans l'espace de caractéristiques.

Pour $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_N, y_N)\}$, un jeu de données d'apprentissage donné avec X l'ensemble des échantillons observés et Y l'ensemble des étiquettes quantitatives correspondantes. Pour une tâche de régression, nous proposons d'apprendre la fonction de transformation du NF $\Phi : \mathcal{X} \to \mathcal{Z}$ où l'espace de caractéristiques $\mathcal{Z}$ est composé de distributions gaussiennes multivariées isotropes, chacune paramétrée par une moyenne $\boldsymbol{\mu}$ et une matrice de covariance $\boldsymbol{\Sigma}$. Ainsi, chaque distribution gaussienne est spécifiée comme suit :

$$P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu}))}.$$

Par conséquent, le terme de log-probabilité utilisé dans la fonction de perte (équation (1.8)) est alors défini de la manière suivante :

$$\begin{aligned} \log P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = &- \frac{1}{2}\Big(D \log(2\pi) + (\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})\Big) \\ &- \log(\det(\boldsymbol{\Sigma})). \end{aligned} \tag{4.6}$$

Nous proposons de résoudre les problèmes de régression en modélisant l'espace de caractéristiques $\mathcal{Z}$ avec des distributions gaussiennes paramétrées par des moyennes interpolées. L'idée est d'apprendre une distribution qui s'étend le long d'un axe principal grâce à l'interpolation entre deux distributions gaussiennes, associées aux valeurs qualitatives extrêmes, tout en organisant les données en fonction de leurs étiquettes quantitatives. Nous définissons donc deux distributions gaussiennes paramétrées par $(\boldsymbol{\mu}_{\min}, \boldsymbol{\Sigma}_{\min})$ et $(\boldsymbol{\mu}_{\max}, \boldsymbol{\Sigma}_{\max})$, associées respectivement à $\min(Y)$ et $\max(Y)$. Nous utilisons des distributions gaussiennes isotropes simples définies par $\boldsymbol{\Sigma}_{\min} = \boldsymbol{\Sigma}_{\max} = \sigma^2 \mathbb{I}_D$, où $\mathbb{I}_D$ représente la matrice identité de taille $D \times D$ et $\sigma^2 \in \mathbb{R}$ correspond à la variance de la distribution sur chaque dimension.

Afin d'effectuer le processus d'interpolation, nous considérons pour chaque échantillon $(\mathbf{x}_i, y_i)$ un coefficient d'appartenance $\tau_{y_i} \in [0, 1] \subset \mathbb{R}$ en fonction de sa valeur quantitative comme suit :

$$\tau_{y_i} = \frac{y_i - \min(Y)}{\max(Y) - \min(Y)}. \tag{4.7}$$

En utilisant le coefficient résultant, nous calculons la moyenne $\boldsymbol{\mu}_{y_i}$ de la distribution gaussienne interpolée correspondant à $(\mathbf{x}_i, y_i)$ de la manière suivante :

$$\boldsymbol{\mu}_{y_i} = \tau_{y_i}\, \boldsymbol{\mu}_{\min} + (1 - \tau_{y_i})\, \boldsymbol{\mu}_{\max}\,. \tag{4.8}$$

Naturellement, pour qu'il soit intéressant d'utiliser notre méthode d'interpolation, les différentes positions des distributions extrêmes dans $\mathcal{Z}$ représentées par leurs moyennes $(\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max})$ doivent être différentes. Une possibilité est de définir à l'avance ces moyennes pour différents vecteurs dans $\mathcal{Z}$. Nous proposons d'apprendre leurs emplacements, c'est-à-dire les moyennes $(\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max})$ des distributions. Pour ce faire, nous ajoutons une nouvelle fonction objective à la fonction objective du NF afin de promouvoir la séparabilité des distributions.

Ainsi, deux termes sont utilisés dans la fonction de perte finale utilisée lors de l'apprentissage du modèle. Le premier, suivant (1.8), utilise la formule de changement de variable décrivant le changement de densité d'un échantillon, à savoir pour l'échantillon $\mathbf{x}_i$,

$$\mathcal{L}_{\text{nf}}(\mathbf{x}_i, \boldsymbol{\mu}_{y_i}) = -\log P_{\mathcal{Z}}(\Phi(\mathbf{x}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) - \log \left| \det \left(\frac{\partial \Phi(\mathbf{x}_i)}{\partial\, \mathbf{x}_i} \right) \right|,$$

où $\log P_{\mathcal{Z}}(\Phi(\mathbf{x}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$ fait référence à l'équation (4.6) en utilisant les paramètres de distribution interpolés ($\boldsymbol{\mu}_{y_i}$,$\boldsymbol{\Sigma}_{y_i}$). La moyenne $\boldsymbol{\mu}_{y_i}$ est calculée avec l'équation (4.8) et la matrice de covariance $\boldsymbol{\Sigma}_{y_i}$ est définie de manière identique aux autres distributions gaussiennes, c'est-à-dire, $\boldsymbol{\Sigma}_{y_i} = \sigma^2 \mathbb{I}_D$. Le second terme représente la proximité des deux gaussiennes extrêmes, avec

$$\mathcal{L}_{\mu} = -\log \left(1 + \| \boldsymbol{\mu}_{\min} - \boldsymbol{\mu}_{\max} \|_2^2\right)$$

à minimiser afin de favoriser leurs séparations. L'utilisation du logarithme permet à l'entraînement de mettre l'accent sur la séparation des distributions lors des premières itérations et de diminuer l'impact de ce terme au fur et à mesure que la distance augmente.

De manière similaire à notre approche de classification, nous pouvons définir la fonction finale à minimiser en utilisant ces expressions avec :

$$\mathcal{L}(\mathbf{x}_i, y_i) = \mathcal{L}_{\text{nf}}(\mathbf{x}_i, \boldsymbol{\mu}_{y_i}) + \beta \mathcal{L}_{\mu}, \tag{4.9}$$

où β correspond à un coefficient de compromis qui équilibre l'optimisation des deux termes. De la même manière que dans notre approche précédente (Section 4.2.1) nous définissons Θ comme l'ensemble des paramètres constituant Φ, avec Ω l'ensemble des paramètres à optimiser, défini dans ce cas comme $\Omega = \{\Theta, \boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}\}$. À chaque itération, nous utilisons une stratégie stochastique de descente de gradient pour estimer l'ensemble des paramètres Ω. Cette estimation est effectuée en minimisant la fonction de perte (équation (4.9)) sur un lot de m échantillons sélectionnés de manière aléatoire à partir du jeu de données d'apprentissage. Ainsi, nous effectuons

$$\Omega \leftarrow \Omega - \eta \sum_{k \in I} \nabla_{\Omega} \mathcal{L}(\mathbf{x}_k, y_k) \tag{4.10}$$

avec η utilisé pour représenter le taux d'apprentissage et I employé pour définir le lot de données choisi de manière aléatoire à chaque itération. L'optimisation du NF pour une tâche de régression est alors donnée dans l'Alg. 6.

Régression dans l'espace de caractéristiques

Le processus d'entraînement proposé aboutit à un espace de caractéristiques $\mathcal{Z}$ où les données sont organisées le long d'un axe linéaire entre $\boldsymbol{\mu}_{\min}$ et $\boldsymbol{\mu}_{\max}$. Dans ces conditions, la définition d'un modèle de régression est simple et peut être réalisée à l'aide de n'importe quelle méthode de régression linéaire. Par conséquent, nous proposons de définir un modèle de régression linéaire $g : \mathcal{Z} \to \mathbb{R}$ correspondant à un modèle de régression f dans $\mathcal{X}$ par l'application de la composition $f(x) = g(\Phi(\mathbf{x}))$. Étant donné que l'espace de caractéristiques $\mathcal{Z}$ est généré à l'aide de la fonction de transformation non-linéaire Φ du NF, la définition de g est équivalente à un modèle prédictif non-linéaire $f : \mathcal{X} \to \mathbb{R}$.

Pour le modèle de régression linéaire $g : \mathcal{Z} \to \mathbb{R}$, nous avons

$$g(\mathbf{z}) = \mathbf{z}^{\top} \boldsymbol{\varphi}^{*} \tag{4.11}$$

pour un vecteur de paramètres $\boldsymbol{\varphi}^{*} \in \mathcal{Z}$ à estimer.

Sans perte de généralité, nous considérons dans la suite la régression ridge dans l'espace de caractéristiques, qui consiste à estimer le modèle optimal en minimisant l'erreur quadratique moyenne régularisée suivante :

$$\min_{\boldsymbol{\varphi}} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \Phi(\mathbf{x}_i)^{\top} \boldsymbol{\varphi} \right)^2 + \lambda \left\| \boldsymbol{\varphi} \right\|_2^2, \tag{4.12}$$

où λ correspond au coefficient d'importance appliqué au terme de régularisation. L'application du théorème de représentation (1.3) conduit au problème d'optimisation suivant :

$$\min_{\alpha_1, \ldots, \alpha_N} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_i) \right)^2 + \lambda \sum_{i,j=1}^{N} \alpha_i \alpha_j \kappa(\mathbf{x}_j, \mathbf{x}_i), \tag{4.13}$$

Algorithm 6 Procédure d'apprentissage du NF pour une tâche de régression.

Require: Ensemble d'apprentissage : $\mathcal{D}$, coefficient de compromis : β, variance des gaussiennes : σ^2, taux d'apprentissage : η, taille de lot : m, nombre d'époques : E

$(\boldsymbol{\Sigma}_{\min}, \boldsymbol{\Sigma}_{\max}) = (\sigma^2 \mathbb{I}_D, \sigma^2 \mathbb{I}_D)$
$(\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}) = (\mathbf{0}, \mathbf{0})$
$(\Theta,\ \Phi,\ \Phi^{-1}) \leftarrow$ Initialisation du modèle de NF
$\Omega = \{\Theta, \boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}\}$
for $e \in \{1, \ldots, E\}$ **do**
  Mélange aléatoire de l'ensemble des données $\mathcal{D}$
  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \ldots (\mathbf{x}_N, y_N)\} \leftarrow \mathcal{D}$
  for $i \in \{1, \ldots, N/m\}$ **do**
    $\mathcal{L}_\mu = -\log\left(1 + \| \boldsymbol{\mu}_{\min} - \boldsymbol{\mu}_{\max} \|_2^2\right)$
    for $k \in \{m(i-1)+1, \ldots, mi\}$ **do**
      $\mathbf{z}_k = \Phi(\mathbf{x}_k)$
      $\tau_{y_k} = \frac{y_k - \min(Y)}{\max(Y) - \min(Y)}$
      $\boldsymbol{\mu}_{y_k} = \tau_{y_k}\, \boldsymbol{\mu}_{\min} + (1 - \tau_{y_k})\, \boldsymbol{\mu}_{\max}$
      $\boldsymbol{\Sigma}_{y_k} = \boldsymbol{\Sigma}_{\min} = \boldsymbol{\Sigma}_{\max}$
      $\mathcal{L}_{\text{nf}} = -\log P_{\mathcal{Z}}(\mathbf{z}_k, \boldsymbol{\mu}_{y_k}, \boldsymbol{\Sigma}_{y_k}) - \log \left|\det\left(\frac{\partial \mathbf{z}_k}{\partial \mathbf{x}_k}\right)\right|$
      $\mathcal{L}(\mathbf{x}_k, y_k) = \mathcal{L}_{\text{nf}} + \beta \mathcal{L}_\mu$
    end for
    $\Omega \leftarrow \Omega - \eta \sum_{k=m(i-1)+1}^{mi} \nabla_\Omega \mathcal{L}(\mathbf{x}_k, y_k)$
    $(\Phi,\ \Phi^{-1}) \leftarrow$ Mise à jour des fonctions avec les paramètres Ω
  end for
end for
return $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}))$

où $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_j), \Phi(\mathbf{x}_i) \rangle_{\mathcal{Z}}$. Il s'agit essentiellement de la régression ridge à noyau [90]. Contrairement aux machines à noyau où le noyau est fixé à l'avance, notre approche repose sur un noyau κ associé à la fonction d'encodage Φ, pour la tâche de régression. Le vecteur solution optimal $\boldsymbol{\alpha}^*$ à ce problème d'optimisation peut être trouvé en annulant son gradient, résultant sur

$$\boldsymbol{\alpha}^* = (\mathbf{Z}\,\mathbf{Z}^\top + \lambda \mathbb{I}_N)^{-1}\, \mathbf{y},$$

où $\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^\top$, $\mathbf{Z} = \begin{pmatrix} \Phi(\mathbf{x}_1) & \cdots & \Phi(\mathbf{x}_N) \end{pmatrix}^\top$ et $\mathbb{I}_N$ représente la matrice identité de taille $N \times N$. À partir de cette expression et du théorème de représentation (1.3), nous obtenons

$$\boldsymbol{\varphi}^* = \mathbf{Z}^\top \boldsymbol{\alpha}^* = \mathbf{Z}^\top (\mathbf{Z}\,\mathbf{Z}^\top + \lambda \mathbb{I}_N)^{-1}\, \mathbf{y}\,.$$

Dans notre cas, comme nous connaissons la transformation appliquée Φ, nous pouvons obtenir le modèle optimal $\boldsymbol{\varphi}^*$ en résolvant directement l'équation (4.12) résultant sur :

$$\boldsymbol{\varphi}^* = (\mathbf{Z}^\top\,\mathbf{Z} + \lambda \mathbb{I}_D)^{-1}\, \mathbf{Z}^\top\, \mathbf{y}\,. \tag{4.14}$$

avec $\mathbb{I}_D$ la matrice identité de taille $D \times D$.

Algorithm 7 Procédure de prédiction de valeurs d'étiquettes quantitatives avec un NF entraîné pour une tâche régression.

Require: Modèle de NF : $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}))$, Ensemble d'entraînement : $\mathcal{D}$, Donnée observée : X_{new}

1: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\} \leftarrow \mathcal{D}$
2: $\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^\top$
3: $\mathbf{Z} = \begin{pmatrix} \Phi(\mathbf{x}_1) & \cdots & \Phi(\mathbf{x}_N) \end{pmatrix}^\top$
4: $\boldsymbol{\varphi}^* = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbb{I}_D)^{-1} \mathbf{Z}^\top \mathbf{y}$
5: **for** $\mathbf{x}_j \in X_{\text{new}}$ **do**
6: $\quad \hat{y}_j = \Phi(\mathbf{x}_j)^\top \boldsymbol{\varphi}^*$
7: **end for**
8: **return** tous les $\hat{y}_j$

Une fois le modèle prédictif optimal g obtenu, à partir de (4.11) et (4.14), le modèle prédictif non-linéaire $f : \mathcal{X} \to \mathbb{R}$ est alors spécifié en combinant la fonction d'encodage Φ et le modèle de régression linéaire g de la manière suivante : $f(\mathbf{x}) = g(\Phi(\mathbf{x}))$. Ainsi, pour tout échantillon $\mathbf{x} \in \mathcal{X}$, nous pouvons prédire son étiquette quantitative par

$$f(\mathbf{x}) = \Phi(\mathbf{x})^\top \boldsymbol{\varphi}^*,$$

où $\boldsymbol{\varphi}^*$ est obtenu avec (4.14). La procédure globale de prédiction est décrite dans Alg. 7.

Génération de pré-image

De plus, notre approche peut être qualifiée d'algorithme de régression sans problème de pré-image grâce à l'existence et à l'exactitude de la transformation inverse Φ^{-1}. Ainsi, nous pouvons générer la pré-image d'un point dans $\mathcal{Z}$ associé à une étiquette quantitative y définie par sa position dans l'interpolation entre les deux gaussiennes. Cette pré-image correspond à un nouvel échantillon $\mathbf{x}^*$ dans l'espace des observations avec une valeur quantitative y^* proche de y.

Nous proposons donc la procédure suivante pour générer des pré-images. Puisque, par conception, la distribution gaussienne paramétrée par $(\boldsymbol{\mu}_{\min}, \boldsymbol{\Sigma}_{\min})$ correspond à la valeur quantitative la plus faible dans Y et celle paramétrée par $(\boldsymbol{\mu}_{\max}, \boldsymbol{\Sigma}_{\max})$ à la valeur quantitative la plus élevée dans Y, nous pouvons calculer l'emplacement, c'est-à-dire la moyenne $\boldsymbol{\mu}_y$, correspondant à une valeur quantitative y en suivant l'équation (4.8). Nous pouvons alors échantillonner $\mathbf{z} \in \mathcal{Z}$ à partir de la distribution gaussienne paramétrée par $(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, et calculer sa pré-image dans l'espace des observations $\mathcal{X}$, à savoir,

$$\hat{\mathbf{x}} = \Phi^{-1}(\mathbf{z}),$$

La procédure de génération de pré-images est présentée dans l'Alg. 8. Il convient de noter qu'une telle génération de pré-images n'est pas possible dans les méthodes de régression basées sur un noyau. En effet, l'échantillonnage dans $\mathcal{Z}$ en utilisant uniquement une valeur quantitative y n'est possible que si nous avons accès à l'espace

Algorithm 8 Procédure de génération de pré-image à partir d'une valeur d'étiquette quantitative y avec un NF entraîné pour une tâche régression.

Require: Modèle de NF : $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}))$, Ensemble d'apprentissage : $\mathcal{D}$, valeur quantitative : y

1: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2) \dots (\mathbf{x}_N, y_N)\} \leftarrow \mathcal{D}$
2: $Y = \{y_1, y_2 \dots, y_N\}$
3: $\tau_y = \frac{y - \min(Y)}{\max(Y) - \min(Y)}$
4: $\boldsymbol{\mu}_y = \tau_y \, \boldsymbol{\mu}_{\min} + (1 - \tau_y) \, \boldsymbol{\mu}_{\max}$
5: $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \, \mathbb{I}_D)$
6: $\hat{\mathbf{x}} = \Phi^{-1}(\mathbf{z})$
7: **return** $\hat{\mathbf{x}}$

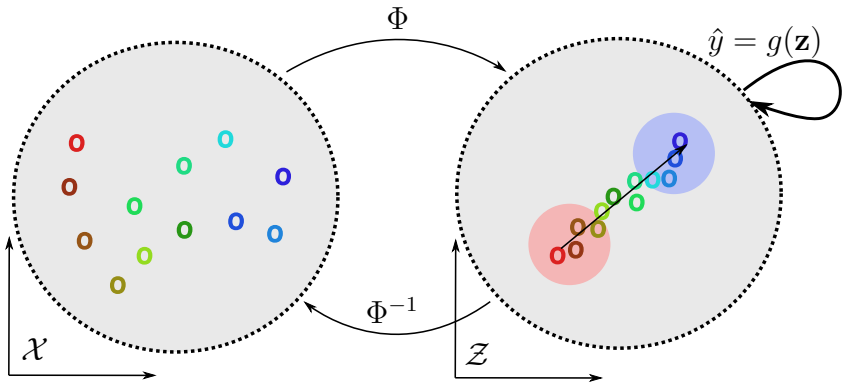


FIGURE 4.2 – Illustration de notre approche pour une tâche de régression, où la couleur de chaque échantillon correspond à sa valeur quantitative.

de caractéristiques généré $\mathcal{Z}$ et si ce dernier est structuré en fonction de cette valeur quantitative, comme le propose notre approche.

La Figure 4.2 illustre notre approche dans son ensemble.

4.3 Expérimentations

Dans cette partie, nous évaluons nos approches de classification et de régression ne souffrant pas du problème de pré-image. Nous proposons d'examiner nos contributions en répondant à quatre questions différentes :

Q1 Basé sur notre formalisme proposé dans le chapitre précédent (Section 3.2), notre approche de classification est-elle efficace pour les tâches de débruitage/compression ? (Section 4.3.4)

Q2 Notre modèle de classification est-il capable d'apprendre une bonne représentation des données à des fins de classification ? (Section 4.3.5)

Q3 Notre approche basée sur une tâche de régression génère-t-elle un bon espace de caractéristiques où les représentations des données peuvent être utilisées pour des objectifs de régression ? (Section 4.3.6)

Q4 Enfin, les générations de pré-images obtenues pour chacune de nos méthodes sont-elles intéressantes et pertinentes ? (Section 4.3.7)

Après avoir décrit les ensembles de données utilisés (Section 4.3.1) et les différents modèles employés (Section 4.3.2) à des fins de comparaison, nous commençons par la définition du protocole dans la Section 4.3.3, avant de décrire les expériences dans les sections suivantes.

4.3.1 Ensembles de données

Puisque nos approches proposées visent à résoudre différentes tâches, à savoir des tâches de classification et de régression, nous utilisons pour nos expérimentations des ensembles de données distincts pour chaque tâche, que nous détaillons ci-dessous.

Ensembles de données de classification. Ci-dessous, nous énumérons les descriptions des ensembles de données de classification.

Double moon : Le jeu de données *double moon* est généré à partir de la bibliothèque scikit-learn [100] et représente une extension du jeu de données *single moon* introduit dans la Section 3.3.1. Il se compose de 1000 données de 2 caractéristiques suivant deux formes de lune différentes, correspondant ainsi à deux classes différentes.

Iris : Iris est un jeu de données réel composé de 150 échantillons, chaque échantillon codant une fleur décrite par 4 caractéristiques différentes : largeur et longueur des sépales et pétales. Le jeu de données est divisé en 3 espèces, définissant ainsi 3 classes différentes.

Cancer du sein : Le jeu de données *Cancer du sein* [121] est un jeu de données réel composé de 569 échantillons correspondant à des masses mammaires, chacune étant décrite par 30 caractéristiques. La tâche correspondante est une tâche de classification binaire qui consiste à prédire si une masse est bénigne ou maligne.

MNIST : Le jeu de données MNIST [31] est un jeu de données de classification classique où chaque échantillon représente une image de 28×28 pixels codant une valeur numérique de 0 à 9. La tâche consiste à prédire la valeur numérique de chacune des 10 000 images de test en utilisant les 60 000 images d'entraînement.

Ensembles de données de régression. Ensuite, nous présentons ci-dessous les ensembles de données utilisés pour la régression.

Rouleau suisse : *Rouleau suisse* est un jeu de données généré à l'aide de la bibliothèque scikit-learn [100]. Il est composé de 1000 échantillons de 2 caractéristiques représentant les coordonnées des données suivant une forme de rouleau suisse bidimensionnelle.

Diabètes : Diabètes est un jeu de données réelles constitué de 442 données décrites avec 10 caractéristiques qui sont l'âge, le sexe, l'indice de masse corporelle, la pression artérielle moyenne et 6 mesures de sérum sanguin.

Chaque échantillon est associé à une valeur quantitative correspondant à la progression de la maladie.

QSAR Aquatic toxicity : Le jeu de données QSAR Aquatic toxicity [16] est composé de 546 molécules organiques décrites avec 8 caractéristiques quantitatives. Ces dernières correspondent aux descripteurs moléculaires suivants : TPSA(Tot) (propriétés moléculaires), SAacc (propriétés moléculaires), H-050 (fragments centrés sur l'atome), MLOGP (propriétés moléculaires), RDCHI (indices de connectivité), GATS1p (autocorrélations 2D), nN (indices constitutionnels) et C-040 (fragments centrés sur l'atome). Chaque échantillon est associé à une valeur quantitative de LC50 qui correspond à la concentration responsable de 50% des décès de Daphnia Magna sur une période de 48 heures.

QSAR Fish toxicity : Le jeu de données QSAR Fish toxicity [17] consiste en 908 molécules organiques représentées par 6 descripteurs moléculaires. Ces caractéristiques représentent les propriétés moléculaires suivantes : MLOGP (propriétés moléculaires), CIC0 (indices d'information), GATS1i (autocorrélations 2D), NdssC (nombre de types d'atomes), NdsCH (nombre de types d'atomes), SM1_Dz(Z) (descripteurs basés sur une matrice 2D). Comme dans le jeu de données précédent, les données sont associées aux valeurs LC50. Cette concentration est à nouveau choisie parce qu'elle est responsable de 50% de la mortalité des poissons sur une période de 96 heures.

4.3.2 Configuration

Étant donné que nos approches sont basées sur les NF, nous avons testé différents formalismes de NF afin de montrer la flexibilité des méthodes proposées. Lors de l'utilisation de données vectorielles, deux approches de NF ont été choisies, à savoir **RealNVP** [33], qui est composé de 32 blocs formés d'une couche de couplage affine et d'une couche de normalisation par lots, et **FFJORD** [41] utilisant une approche de CNF. De plus, certains ensembles de données étant composés d'images, nous avons choisi d'utiliser un modèle conçu pour travailler sur ce type de données appelé **Glow** [60]. Dans le but de renforcer la robustesse des modèles face au bruit, nous avons appliqué notre approche d'apprentissage sur des données augmentées avec un bruit gaussien centré d'un écart-type de $0,5$.

De plus, nous comparons nos approches proposées avec des méthodes à noyaux qui sont directement confrontées au problème de pré-image. À cet effet, nous utilisons les noyaux suivants :

Linéaire : $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2$

RBF : $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma || \mathbf{x}_1 - \mathbf{x}_2 ||^2\right)$

Polynomial : $\kappa(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)^n$

Sigmoïde : $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)$

Nos expériences comprenant des expérimentations de débruitage, classification et régression, nous utilisons ces noyaux avec différentes méthodes à noyaux applicables pour la tâche associée. Dans le cas des tâches de débruitage et de classification,

nous utilisons les méthodes d'ACP et SVM à noyau respectivement. Pour la résolution des tâches de régression et à l'aide des définitions des noyaux, nous entraînons chaque modèle prédictif associé en minimisant la fonction de coût exprimée dans l'équation (4.13). De plus, la capacité de génération de pré-images est comparée à la technique décrite dans [5] appliquée aux méthodes à noyaux précédentes.

4.3.3 Protocole

Pour permettre la reproductibilité des expériences, nous décrivons dans cette section le protocole utilisé. Lorsque les divisions du jeu de données ne sont pas disponibles, nous divisons celui-ci en 90% pour l'entraînement (ajustement des modèles et réglage fin des hyperparamètres) et 10% pour l'évaluation des performances. Pour une comparaison équitable, les noyaux ont été optimisés de manière efficace, leurs paramètres étant obtenus par l'application d'une recherche en grille avec validation croisée. Compte tenu des définitions des noyaux de la Section 4.3.2, 10 valeurs pour λ de 10^{-5} à 10^{2} et 5 valeurs pour γ de 10^{-5} à 10^{3} ont été échantillonnées sur une échelle logarithmique. Pour le noyau Polynomial, les valeurs candidates pour sa puissance n étaient $\{1, 2, 3, 4\}$.

4.3.4 Expérimentations de débruitage dans un contexte de classification

Dans cette section, nous proposons tout d'abord d'évaluer les performances de débruitage de notre approche de classification et ainsi répondre à la question **Q1**. En effet, étant donné que notre approche est une adaptation de notre formalisme général introduit dans le Chapitre 3 et qu'elle implique la même méthode de modélisation de distribution gaussienne dans $\mathcal{Z}$ auquel nous ajoutons la séparation des données par classe, nous pouvons étendre nos évaluations précédentes à chacune des classes présentes dans le jeu de données utilisé.

Débruitage sur *double moon*. Tout d'abord, nous menons les expériences sur le jeu de données *double moon*. Pour ces expériences, nous reprenons la même configuration utilisée lors des expérimentations faites sur *single moon* dans la Section 3.3.4 et définissons les matrices covariances de la même manière pour les deux formes de lune. Les deux formes se distinguent par les moyennes $\boldsymbol{\mu}_c$ associées à chaque gaussienne. De plus, comme nous utilisons le terme de distance supplémentaire de l'équation (4.2), nous définissons $\beta = 1$ et les valeurs moyennes initiales des gaussiennes à zéro. L'expérience suit le même protocole utilisé sur les données de *single moon* et expliqué dans la Section 3.3.4. Ainsi, les approches de débruitages $\mathcal{Z}$-ACP et NF basé ACP sont appliquées et comparées en considérant une distribution gaussienne associée à la classe correspondante. Les distances obtenues par les différentes méthodes de pré-image sont présentées dans la Table 4.1. Les résultats calculés mettent en évidence les bonnes performances en débruitage de nos modèles de classification pour chacune des classes suivant les deux différentes architectures de NF.

De plus, nous évaluons si notre approche de débruitage présente des performances significativement différentes de celles des autres méthodes. Pour ce faire, nous uti-

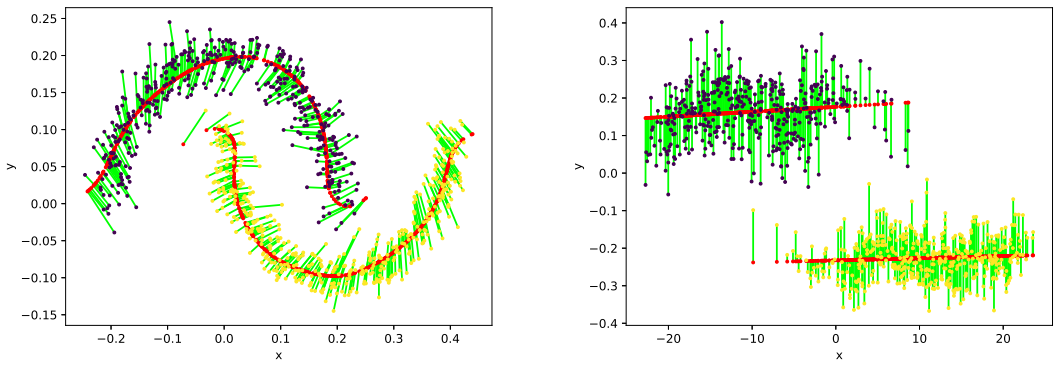
TABLE 4.1 – Évaluation de débruitage dans une configuration de classification et comparaison avec la kernel-ACP en calculant la distance $\|\mathbf{x} - \hat{\mathbf{x}}\|_2$ entre les données débruitées $\hat{\mathbf{x}}$ et les données d'origine $\mathbf{x}$.

Methodes			Ensembles de données
			Double moon
	Linéaire	K-ACP	0.1305
	RBF	K-ACP	0.0683
	Polynomial	K-ACP	0.0725
	Sigmoïde	K-ACP	0.2170
Nos approches	RealNVP	$\mathcal{Z}$-ACP	0.0368
	RealNVP	NF basé ACP	0.0460
	FFJORD	$\mathcal{Z}$-ACP	**0.0244**
	FFJORD	NF basé ACP	0.0370

lisons le test H de Kruskal-Wallis [68] pour comparer les distributions de performances des différentes méthodes de débruitage. Nous rejetons l'hypothèse nulle, qui indique qu'il n'y a pas de différence de performance entre les méthodes, lorsque les p-valeurs sont inférieures à $0,05$. Pour cela, nous avons réalisé 20 évaluations distinctes pour chaque méthode de débruitage en appliquant un bruit aléatoire. Les résultats montrent que les p-valeurs obtenues pour l'approche NF basée sur l'ACP et l'approche $\mathcal{Z}$-ACP sur le jeu de données *double moon* sont proches de 10^{-9}, ce qui nous permet de rejeter l'hypothèse nulle et de conclure que ces méthodes sont significativement plus performantes que les autres méthodes de débruitage.

De plus, étant donné que les données utilisées sont représentées en 2D, nous pouvons les représenter à la fois dans l'espace des observations et dans l'espace de caractéristiques. Pour visualiser cela, nous présentons la Figure 4.3, où les données apparaissent dans les deux espaces : l'espace des observations $\mathcal{X}$ et l'espace de caractéristiques $\mathcal{Z}$ appris par le modèle NF FFJORD. Les données d'apprentissage, qui correspondent aux deux lunes, sont représentées en noir et en jaune respectivement. De plus, pour chacune des lunes, nous avons calculé une projection à l'aide de la $\mathcal{Z}$-ACP, qui est représentée en rouge. Ces projections sont des transformations linéaires dans l'espace de caractéristiques et correspondent à des transformations non-linéaires dans l'espace des observations grâce à la pré-image. Enfin, pour souligner le lien entre chaque échantillon et sa projection, nous avons relié chaque point en vert.

Débruitage d'images. Après avoir validé la capacité de débruitage sur des ensembles de données jouets, nous avons procédé à des expériences sur des images réelles provenant du jeu de données *MNIST*. Toutefois, étant donné que les données sont des images, il n'est pas optimal de les comparer en utilisant des distances comme pour les ensembles de données de caractéristiques plus simples. En effet, même si les distances entre les pré-images et les images d'origine sont faibles, les pré-images obtenues peuvent avoir une faible qualité visuelle. Par conséquent, l'évaluation des



(a) Espace des observations $\mathcal{X}$ (b) Espace de caractéristiques $\mathcal{Z}$

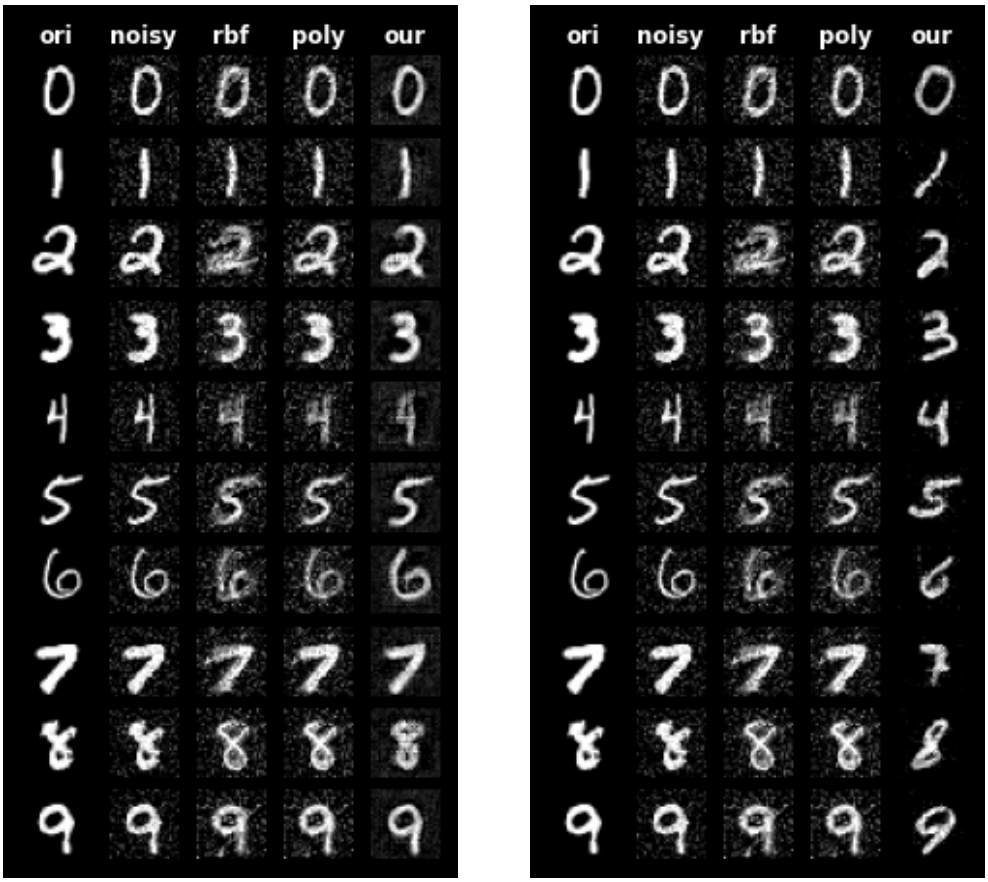
FIGURE 4.3 – (a) Pré-images (points rouges) générées par les projections $\mathcal{Z}$-ACP du jeu de données *double moon* (points noirs et jaunes). (b) Ces points du jeu de données et leurs projections linéaires (points rouges) dans l'espace des caractéristiques $\mathcal{Z}$ obtenu par la transformation FFJORD apprise Φ.

modèles de débruitage d'images a été réalisée qualitativement.

Pour évaluer la performance de débruitage de notre approche de classification, nous avons réalisé des tests sur des images appartenant à différentes classes, en comparant les résultats obtenus avec des procédures de débruitage basées sur le noyau et notre méthode proposée, sous ses deux versions ($\mathcal{Z}$-ACP et NF basé ACP). La Figure 4.4 présente les différents résultats obtenus. De la même manière que lors des expériences menées sur le jeu de données *MNIST3* dans la Section 3.3.4, il est à remarquer que les méthodes de débruitage basées sur les noyaux présentent des artefacts dus à l'espace de caractéristiques employé par le noyau. En revanche, les méthodes de pré-image basées sur les NF proposées produisent une version propre de chaque échantillon testé, sans artefacts perceptibles. Grâce aux résultats obtenus lors de cette expérience, nous pouvons constater que notre approche de classification permet d'obtenir des images débruitées de bonne qualité, permettant ainsi d'étendre la procédure de débruitage à toutes les classes d'images présentes dans le jeu de données.

Compression d'images. L'approche étant basée sur l'ACP, nous pouvons à la fois l'utiliser pour des tâches de débruitage et de compression. En utilisant les composantes principales, les données peuvent être compressées même si les représentations dans $\mathcal{Z}$ ont des dimensions égales à celles de l'espace observé $\mathcal{X}$. Ainsi, pour évaluer notre approche, nous avons généré des représentations compressées dans $\mathcal{Z}$ par projection sur les composantes principales. La Figure 4.5 montre les résultats de cette évaluation pour des niveaux de compression différents utilisant nos deux différentes approches.

De cette expérimentation, nous pouvons, de la même manière que dans la Section 3.3.4, noter une bonne génération des images compressées malgré le haut niveau de compression. En effet, les premières colonnes représentent une très haute compression de la représentation des images dans $\mathcal{Z}$ à une seule dimension. De plus, cette



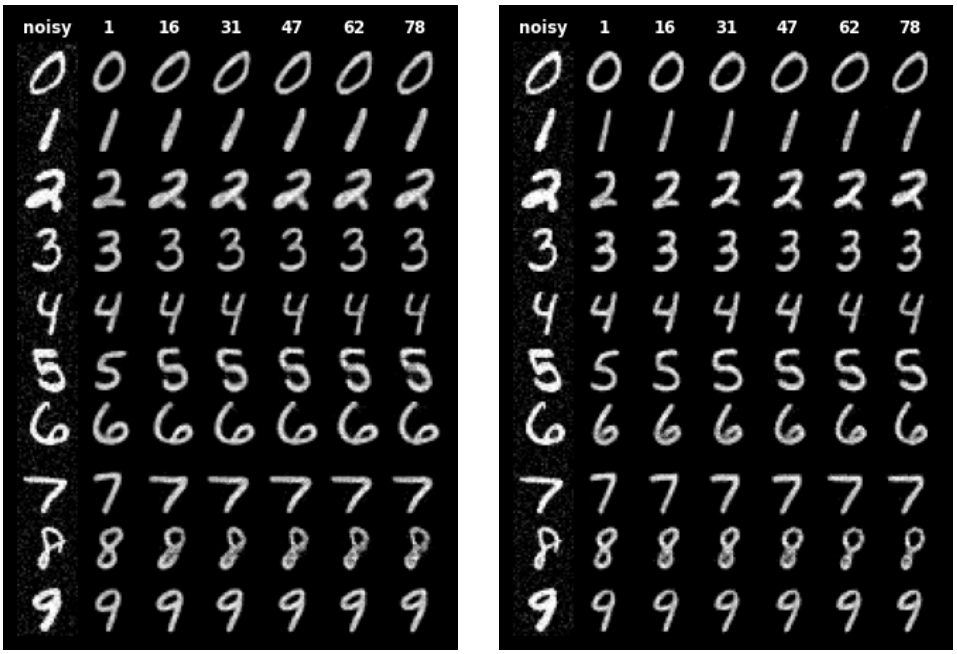
(a) Débruitage $\mathcal{Z}$-ACP (b) Débruitage NF basé ACP

FIGURE 4.4 – La première colonne représente les images originales et la deuxième colonne les images bruitées gaussiennes. (a) Les colonnes suivantes représentent les résultats du débruitage à l'aide de de RBF-ACP, de Polynomial-ACP et de $\mathcal{Z}$-ACP (b) Les colonnes suivantes représentent les résultats du débruitage à l'aide de RBF-ACP, de Polynomial-ACP et du NF basé ACP.

expérience permet de mettre en évidence que quelle que soit la classe de l'image utilisée, nos approches permettent la génération d'image compressée de haute qualité. Il est important de remarquer que la possibilité de compresser les représentations des images à une seule dimension est due en grande partie à la faible variance inter-classe dans ce jeu de données.

4.3.5 Expérimentations de classification

La deuxième question à laquelle nos expériences doivent répondre concerne l'évaluation de l'espace des caractéristiques à des fins de classification (**Q2**). L'hypothèse ici est que la séparation proposée des classes dans l'espace de caractéristiques permet de dériver un classifieur simple, c'est-à-dire linéaire dans l'espace des caractéristiques. Pour cela, nous évaluons l'approche décrite à la Section 4.2.1, d'abord sur des ensembles de données générées, puis sur des ensembles de données réels. Le



(a) Compression $\mathcal{Z}$-ACP (b) Compression NF basé ACP

FIGURE 4.5 – Pré-images générées à différents niveaux de compression dans $\mathcal{Z}$. Les premières colonnes représentent les images originales et les 6 colonnes suivantes représentent les pré-images classées de la compression la plus élevée à gauche à la compression la plus faible à droite. Les niveaux de compression sont échantillonnés linéairement de 1 à k.

critère d'évaluation utilisé pour ces expériences de classification correspond au taux de bonne classification.

En suivant le protocole décrit dans la Section 4.3.3, la Table 4.2 présente les performances obtenues par les différentes méthodes sur quatre ensembles de données. Nous pouvons noter que les approches proposées basées sur les NF, avec l'application d'un SVM linéaire dans l'espace de caractéristiques, obtiennent les meilleures précisions sur tous les ensembles de données, du plus simple à *MNIST*. Cette expérience montre ainsi que notre NF modifié est capable de calculer une transformation non-linéaire de l'espace des observations vers un espace de caractéristiques où les données sont linéairement séparables. Notez que les résultats sur *MNIST* utilisent le jeu de données officiel et sont directement comparables à d'autres méthodes. Ces résultats apportent une réponse positive à **Q2**.

Comme pour les expériences de débruitage, nous évaluons les différences de performances statistiques entre notre méthode de classification et les autres en calculant les p-valeurs. À cette fin, les performances de classification sont évaluées sur 20 différentes répartitions des données sur les différents ensembles de données. En utilisant le modèle RealNVP, nous obtenons $9,2 \times 10^{-5}$, $5,6 \times 10^{-5}$ et 10^{-14} comme p-valeurs calculées sur les ensembles de données de *Cancer du sein*, de *Iris* et de *double moon*, respectivement. Les p-valeurs obtenues à l'aide du modèle FFJORD sont respectivement de $4,5 \times 10^{-5}$, $4,3 \times 10^{-5}$ et 10^{-14}. En rejetant l'hypothèse nulle, ces résultats

TABLE 4.2 – Précisions de classification obtenues par nos approches basées sur les NF (RealNVP, FFJORD et Glow) en comparaison avec les méthodes SVM appliquées dans l'espace des caractéristiques $\mathcal{Z}$ en utilisant les noyaux linéaire, RBF, polynomial et sigmoïde. Les meilleurs résultats sont mis en évidence en gras.

Méthodes		Ensembles de données			
		Double moon	Iris	Cancer du sein	MNIST
SVM	Linéaire	$93,00\%$	$93,34\%$	$94,64\%$	94.63%
	RBF	$\mathbf{100,00}\%$	$93,34\%$	$94,64\%$	97.31%
	Polynomial	$93,00\%$	$93,34\%$	$94,64\%$	97.68%
	Sigmoïde	$90,00\%$	$80,00\%$	$96,43\%$	90.98%
Nos approches	RealNVP	$\mathbf{100,00}\%$	$\mathbf{100,00}\%$	$\mathbf{100,00}\%$	-
	FFJORD	$\mathbf{100,00}\%$	$\mathbf{100,00}\%$	$98,21\%$	-
	Glow	-	-	-	**99.47**%

soulignent la pertinence de notre approche.

4.3.6 Expérimentations de régression

Afin d'évaluer notre approche de régression, nous commençons par examiner la question (**Q3**). Dans ces expériences, nous estimons la capacité de notre espace de caractéristiques généré $\mathcal{Z}$ pour une tâche de régression. L'hypothèse est que celui-ci organise les données dans une direction suivant la régression linéaire. Le critère d'évaluation utilisé pour évaluer les performances de régression est le score R^2.

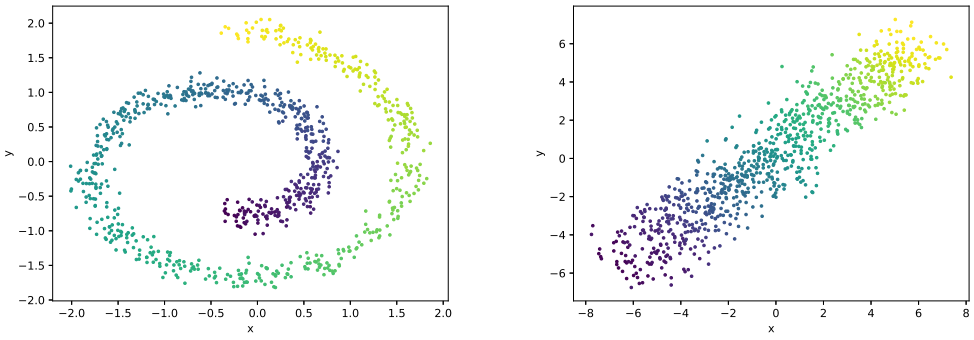
Comme les données du *rouleau suisse* sont en 2D, nous pouvons afficher à la fois l'espace des observations $\mathcal{X}$ et l'espace de caractéristiques $\mathcal{Z}$. La Figure 4.6 l'illustre à l'aide de la transformation FFJORD apprise Φ. Comme prévu, nous pouvons observer que les représentations des données dans l'espace $\mathcal{Z}$ se déploient le long de l'axe d'interpolation par rapport à la valeur quantitative y.

Les performances de régression obtenues sur les quatre différents ensembles de données sont reportées dans la Table 4.3. Cette évaluation nous permet de mettre en évidence la meilleure précision de nos approches appliquant la régression linéaire ridge dans l'espace de caractéristiques $\mathcal{Z}$ sur tous les ensembles de données.

Les expériences sur les deux derniers ensembles de données (*QSAR Aquatic toxicity* et *QSAR Fish toxicity*) utilisent la même division des données dans [16, 17], rendant la comparaison possible. Dans ces articles, les auteurs ont évalué l'approche en calculant le coefficient de détermination R^2 pour la prédiction des données d'apprentissage et le Q^2_{ext} pour la prédiction des données de test. Ce dernier est défini comme suit :

$$Q^2_{ext} = 1 - \frac{(\sum_{i=1}^{N_{test}}(y_i - \hat{y}_i)^2)/N_{test}}{(\sum_{i=1}^{N}(y_i - \bar{y})^2)/N},$$

où $\hat{y}_i$ est la valeur prédite à l'aide de $\mathbf{x}_i \in X_{test}$, $\bar{y}$ est la moyenne des valeurs quantitatives d'apprentissage et N_{test} est le nombre de données dans le jeu de données de test. La Table 4.4 présente une comparaison des résultats entre notre approche et leur méthode. Ces résultats montrent que notre modèle est plus performant sur le premier



(a) $\mathcal{X}$ (b) $\mathcal{Z}$

FIGURE 4.6 – (a) Jeu de données de *rouleau suisse* dans l'espace d'entrée $\mathcal{X}$. (b) Ces points du jeu de données dans l'espace des caractéristiques $\mathcal{Z}$ obtenu par la transformation de FFJORD apprise Φ.

TABLE 4.3 – Évaluation de notre approche dans une configuration de régression en appliquant la régression Ridge dans $\mathcal{Z}$ et en calculant le coefficient de détermination R^2 sur les résultats prédits en comparaison avec les méthodes Ridge avec noyaux utilisant les noyaux suivants : linéaire, RBF, polynomial et sigmoïde.

Méthodes		Ensembles de données			
		Rouleau suisse	Diabètes	*QSAR aquatic toxicity*	*QSAR fish toxicity*
	Linéaire	0.012	0.452	0.392	0.567
	RBF	**1.0**	0.458	0.411	0.600
	Polynomial	**1.0**	0.461	0.381	0.603
	Sigmoïde	**1.0**	0.454	0.403	0.567
Nos approches	RealNVP	**1.0**	**0.548**	**0.565**	0.635
	FFJORD	**1.0**	0.463	0.508	**0.661**

jeu de données, qu'il s'agisse de prédictions sur le jeu de données d'entraînement ou de test. De plus, notre méthode permet d'obtenir de meilleures performances sur les données d'entraînement du deuxième jeu de données, tandis que les prédictions sur le jeu de données de test aboutissent à des performances similaires.

Ainsi, nous pouvons répondre positivement à **Q3**, c'est-à-dire que la transformation non-linéaire Φ apprise par notre approche est capable de générer un bon espace de représentation où les données sont organisées linéairement.

4.3.7 Qualité de pré-image

Dans cette section, nous évaluons nos contributions sur la capacité de génération de pré-images afin de répondre à **Q4**. Répondre positivement à cette question nous permet de montrer que le problème de pré-image n'apparaît pas dans notre approche,

TABLE 4.4 – Comparaison de nos résultats sur les jeux de données QSAR avec les résultats des articles associés ([16, 17]). Calcul du coefficient de détermination R^2 pour la prédiction sur le jeu de données d'entraînement et Q^2_{ext} pour la prédiction sur le jeu de données de test.

Méthodes		Ensembles de données			
		QSAR aquatic toxicity		*QSAR fish toxicity*	
		R^2	Q^2_{ext}	R^2	Q^2_{ext}
[16, 17]		0.60	0.43	0.62	*0.61*
Nos approches	RealNVP	0.83	**0.56**	0.66	*0.59*
	FFJORD	**0.84**	0.51	**0.72**	*0.60*

puisque la transformation inverse est connue.

Pré-image de classification. Dans cette première partie, nous étudions notre méthode de classification à travers la génération de pré-images à partir du jeu de données *MNIST* permettant l'évaluation de la qualité visuelle de chaque pré-image obtenue. Pour ce faire, nous utilisons la fonction Φ^{-1} (voir Section 1.2) pour calculer la pré-image à partir d'un point dans l'espace des caractéristiques $\mathcal{Z}$.

Tout d'abord, nous évaluons la capacité de génération de notre NF modifiée en calculant la pré-image $\mathbf{x}_c$ de chaque prototype de classe $\mathbf{z}_c \in \mathcal{Z}$ défini comme suit :

$$\mathbf{z}_c = \frac{1}{|N_c|} \sum_{i|y_i=c}^{N_c} \mathbf{z}_i,$$

à savoir,

$$\mathbf{x}_c = \Phi^{-1}(\mathbf{z}_c),$$

où N_c désigne le nombre d'éléments de la classe c et $\mathbf{z}_i = \Phi(\mathbf{x}_i)$. Les pré-images de chaque classe de prototypes sont présentées dans la dernière colonne de la Figure 4.7. Comme nous pouvons le constater, la pré-image calculée est, comme prévu, une représentation visuelle de chaque nombre, ce qui démontre que notre approche NF conserve sa bonne capacité de génération.

De plus, la résolution de pré-images peut faciliter l'exploration de l'espace de caractéristiques en permettant de pré-imager n'importe quel point de $\mathcal{Z}$, grâce à l'inversibilité de Φ. Les 10 premières colonnes de la Figure 4.7 illustrent les pré-images générées de 10 échantillons aléatoires provenant de chaque distribution gaussienne paramétrée par $\mathbf{\Sigma}_c$ et $\boldsymbol{\mu}_c$, en fonction de la classe. Une fois de plus, les pré-images générées sont de bonne qualité et conformes à leurs classes.

Pour obtenir des résultats de génération de pré-images supplémentaires, nous évaluons la qualité de génération de pré-images en utilisant le jeu de données de visages *Olivetti*. À cet effet, nous entraînons un modèle de NF, dans ce cas Glow, qui opère sur les images du jeu de données, en suivant l'algorithme Alg. 4. Une fois le modèle entraîné, nous générons des visages sous forme de pré-images de points

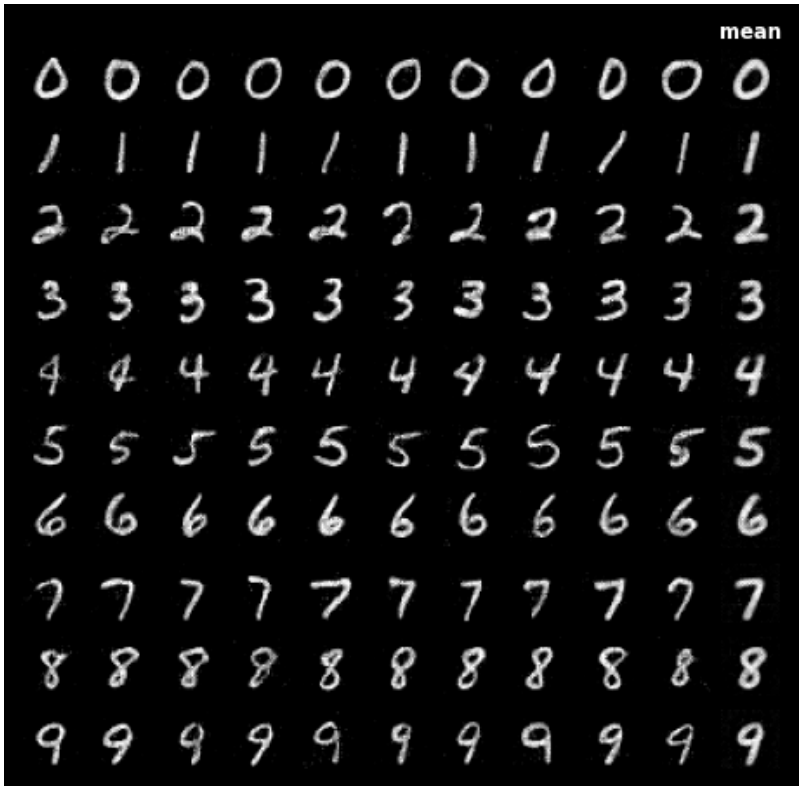


FIGURE 4.7 – Pré-images obtenues en échantillonnant des vecteurs dans $\mathcal{Z}$ selon chaque distribution gaussienne associée à chaque classe.

d'interpolation spécifiques entre deux représentations de visages dans l'espace latent $\mathcal{Z}$. La Figure 4.8 illustre les résultats de ce processus. Cette figure nous permet de mettre en évidence la cohérence de l'espace latent et des générations de pré-images.

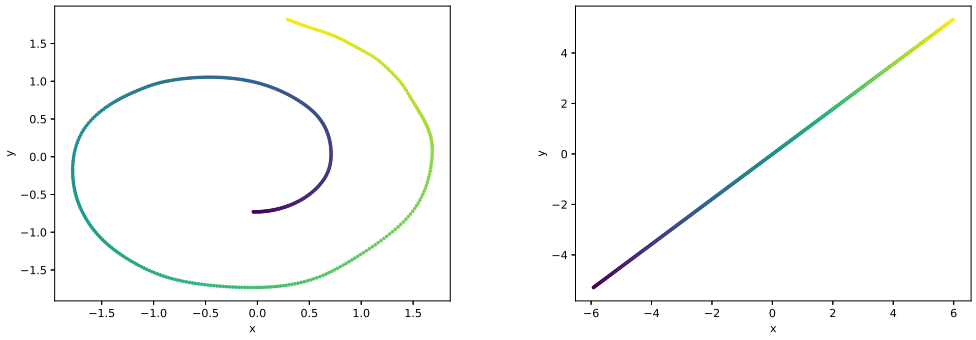
Pré-image de régression. Afin d'évaluer la capacité générative de notre approche de régression, nous générons des pré-images suivant l'Alg. 8 en échantillonnant des points dans $\mathcal{Z}$. Pour obtenir ces points, nous décidons d'échantillonner les valeurs de y uniformément entre les valeurs min et max de $y_i \in Y$. Ensuite, les moyennes associées des distributions gaussiennes $\boldsymbol{\mu}_y$ sont calculées avec l'équation (4.8) et utilisées comme nos points échantillonnés dans l'espace $\mathcal{Z}$. Ainsi, les pré-images correspondantes sont générées dans l'espace $\mathcal{X}$ à l'aide de la transformée inverse Φ^{-1}. Pour les ensembles de données en deux dimensions, tels que le jeu de données *rouleau suisse*, les pré-images générées peuvent être affichées comme illustré dans la Figure 4.9. Dans cette figure, les pré-images ont été générées en utilisant notre transformation inverse apprise avec FFJORD. Cette dernière montre la capacité de notre modèle à générer des pré-images pertinentes à l'aide de points échantillonnés dans $\mathcal{Z}$ ne faisant pas partie du jeu de données.

Les résultats de nos expériences démontrent que nos approches permettent la génération de pré-images cohérentes pour des points de l'espace $\mathcal{Z}$. Par conséquent, nos approches répondent positivement à la question **Q4**.

FIGURE 4.8 – Illustration des pré-images des échantillons d'interpolation dans $\mathcal{Z}$. Chaque ligne illustre deux visages sélectionnés (le visage le plus à gauche et le plus à droite) et 10 pré-images de 10 interpolations entre les représentations de ces deux visages dans l'espace latent.

4.4 Discussion

Dans ce chapitre, nous avons démontré l'efficacité de notre approche sur différentes tâches et ensembles de données. Bien que le jeu de données MNIST offre la possibilité d'expérimenter avec des données de haute dimension, comparé à des ensembles de données jouets comme le jeu de données de *double moon* avec seulement 2 d'échantillons de caractéristiques, les ensembles de données modernes, comme les images haute définition ou les nœuds de graphe, peuvent avoir des dimensions encore plus élevées. Par conséquent, l'utilisation de méthodes telles que l'ACP soulève des questions quant à l'extensibilité de notre approche. Dans le cadre d'apprentissage de modèle pour le débruitage de données, nous avons proposé l'approche NF basé ACP (Alg. 3) pour résoudre ce problème, fournissant une adaptation du formalisme pour les données de haute dimension en éliminant la nécessité de calculer l'ACP dans l'espace latent. En effet, les expériences de Table 3.1 et Table 4.1 démontrent que notre approche donne des résultats favorables tout en contournant les coûts de calcul excessifs. Toutefois, d'autres problèmes d'extensibilité peuvent se poser en fonction du modèle utilisé. Par exemple, des problèmes de scalabilité sont souvent



(a) $\mathcal{X}$ (b) $\mathcal{Z}$

FIGURE 4.9 – (a) Pré-images générées à partir de points de $\mathcal{Z}$ obtenus par la transformation inverse de FFJORD apprise Φ^{-1} sur le jeu de données de *rouleau suisse*. (b) 500 points dans $\mathcal{Z}$ échantillonnés uniformément entre les valeurs min et max de Y.

observés dans les modèles de NF courants basés sur des couches de couplage tels que RealNVP [33] et Glow [60]. Ces modèles nécessitent souvent des ressources mémoire importantes pour stocker toutes les activations intermédiaires. En revanche, les CNF telles que FFJORD [41] atténuent ce problème, n'ayant pas besoin de calculer ou de stocker ces valeurs de manière explicite. Cependant, le coût de calcul de la résolution des ODEs dans les CNF peut être élevé, en particulier pour les données à grandes dimensions et jeux de données massifs.

De plus, le plus grand jeu de données utilisé dans cette étude est *MNIST* composé de 60 000 échantillons. Toutefois, à l'heure actuelle, la taille de ce jeu de données peut être considérée comme petite, ce qui soulève des inquiétudes quant à sa pertinence pour les grands ensembles de données. Comme indiqué précédemment, notre approche NF basé ACP résout correctement le problème de la scalabilité aux données à grande dimension. De plus, elle est modulable en fonction du nombre d'échantillons et dépend uniquement de la complexité d'apprentissage du modèle NF utilisé. Généralement, l'utilisation de grands ensembles de données dans le processus d'apprentissage du NF peut prendre du temps, car l'augmentation de la taille du lot peut être gourmande en mémoire. D'autre part, nos autres approches dépendent également de la méthode appliquée dans l'espace latent, ce qui peut augmenter considérablement la consommation de mémoire et de temps. La réduction de ces coûts peut se faire soit en utilisant une méthode de projection ou de classification moins coûteuse dans l'espace latent, soit par un contournement trivial tel que la sélection d'un ensemble représentatif de données pour adapter ces méthodes dans l'espace latent.

Figure 4.10 montre une analyse de scalabilité de notre processus d'apprentissage par rapport au modèle utilisé. Pour évaluer l'impact de la taille et de la dimensionnalité du jeu de données sur les besoins en mémoire et en temps, nous avons effectué une analyse sur une période d'apprentissage avec une taille de lot fixe de 10, tout en gardant les hyperparamètres du modèle constants. Comme mentionné précédemment, la figure illustre le contraste entre RealNVP et FFJORD en termes d'utilisation de la mémoire et du temps. Plus précisément, alors que FFJORD né-

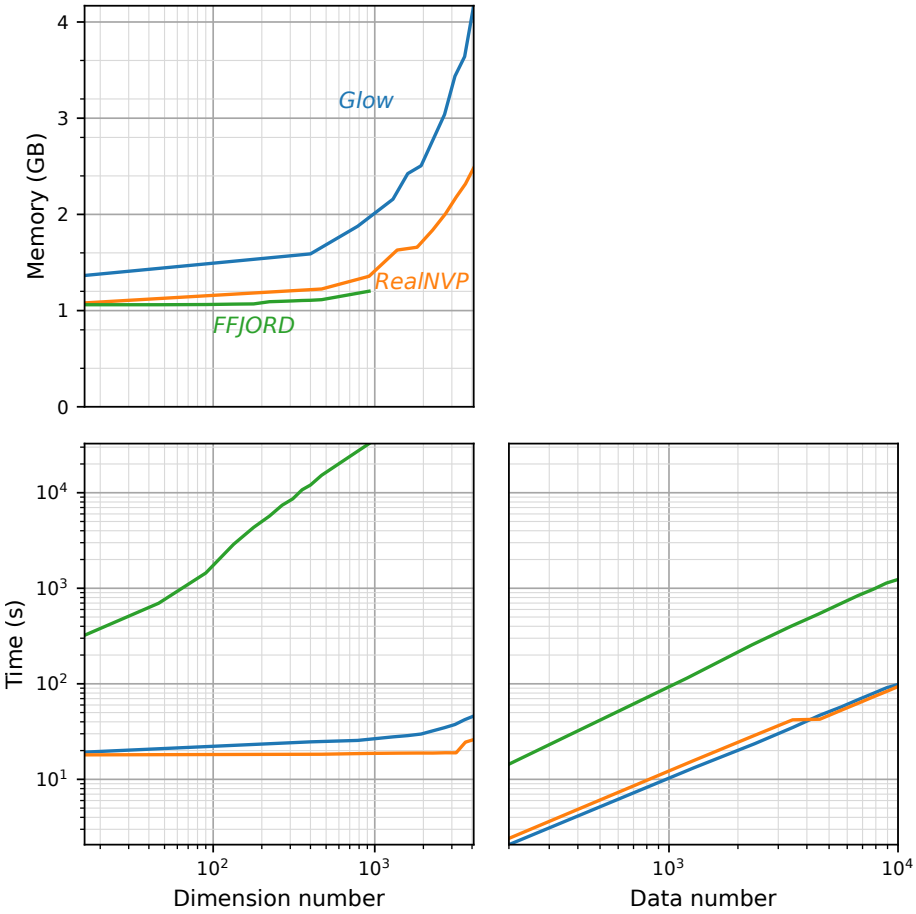


FIGURE 4.10 – Évaluation de la scalabilité de notre approche. L'utilisation mémoire et de temps proviennent de l'entraînement d'une époque avec une taille de lot de 10.

cessite une consommation de mémoire plus faible, il entraîne un coût de calcul plus élevé qui se traduit par une augmentation du temps d'apprentissage à mesure que la dimensionnalité augmente.

4.5 Conclusion

Dans ce chapitre, nous avons proposé deux spécifications de notre formalisme introduit Section 3.2 en supervisant l'apprentissage du modèle en fonction de la tâche. Ainsi, deux différentes approches ont été proposées pour la résolution des tâches de classification et de régression respectivement, sans problème de pré-image. Nos méthodes génèrent un espace supervisé dans lequel l'application d'opération linéaire permet de bons résultats. Les expériences ont montré de bonnes performances pour résoudre les problèmes de classification et de régression en appliquant des opérations linéaires simples, démontrant ainsi la pertinence de l'espace de caractéristiques généré pour une tâche donnée. De plus, nos approches proposées permettent une meilleure interprétabilité des données et de l'espace engendré en rendant possible la transformation de l'espace de caractéristiques à l'espace des observations,

ainsi que la génération de pré-images de points d'intérêt. Ces résultats soulignent l'intérêt des NF pour résoudre le problème de pré-image tout en permettant une génération non-linéaire et efficace d'un espace de caractéristiques pour une tâche donnée.

Les expérimentations ont été menées sur divers types de données, notamment des données vectorielles et des images. Toutefois, il existe d'autres types de données, comme les graphes. La représentation de ces données complexes et discrètes constitue un défi majeur de nos jours. De ce contexte, il est intéressant de noter que nos méthodes offrent la possibilité d'exploiter n'importe quelle architecture de NF, capable de traiter divers types de données, permettant ainsi d'adapter nos approches à chaque jeu de données, y compris potentiellement les graphes. Dans le chapitre suivant, nous nous intéressons à cela en cherchant à adapter nos approches à des données de type graphe.

CHAPITRE 5

Pré-image de données structurées de type graphe

5.1 Introduction

Les graphes permettent la représentation de données structurées utiles à la modélisation de divers types de scénarios du monde réel, tels que les réseaux sociaux, les molécules chimiques et les systèmes de transport. L'analyse et la modélisation de graphes constituent un domaine d'étude important en informatique, statistiques et apprentissage machine. Contrairement aux données vectorielles, les graphes représentent un type de données complexe composé d'une structure discrète, encodant la topologie entre les nœuds du graphe. Les algorithmes de ML n'étant pas adaptés à ce type de données, mais à des données vectorielles, il faut donc définir un encodage des graphes afin de pourvoir leur appliquer divers algorithmes de ML et de traitement de données.

Ces motivations ont conduit au développement de nombreuses approches de ML, telles que les noyaux sur graphes et les réseaux de neurones sur graphes introduits Section 1.1.3. Alors que les premiers permettent un encodage des graphes dans un espace de caractéristiques potentiellement implicite, les seconds cherchent à générer une transformation explicite des graphes dans un espace de représentation euclidien. Cependant, malgré les bons résultats obtenus par les noyaux sur graphes, leur limitation réside dans l'impossibilité d'accéder directement à l'espace de représentation des graphes, ce qui les confronte au problème de pré-image. De même, les réseaux de neurones sur graphes utilisés pour encoder les graphes en vue d'une tâche prédictive spécifique sont généralement directement confrontés à ce même problème. La résolution d'un tel problème permettrait l'obtention d'une transformation inverse qui, à partir d'un espace de représentation où diverses méthodes de ML peuvent être appliquées, permettrait de reconstruire des données discrètes et complexes de graphes appartenant à l'espace des observations. Grâce à cela, des opérations qui seraient normalement difficiles à réaliser dans l'espace des observations, telles que le calcul d'un graphe moyen, peuvent être effectuées plus facilement dans l'espace des caractéristiques. Ensuite, en appliquant la transformation inverse, il est possible

d'obtenir le graphe pré-image correspondant dans l'espace des observations.

De manière similaire au chapitre précédent, la résolution de ce problème dans le contexte de tâches prédictives telles que la classification et la régression présente de multiples avantages. Elle offre une meilleure interprétabilité des données et du modèle, ainsi que la possibilité de générer de nouvelles données en fonction d'une certaine valeur de la propriété associée à la tâche. Ainsi, il serait possible de créer de nouveaux graphes en échantillonnant des représentations dans l'espace des caractéristiques en fonction de valeurs d'étiquettes souhaitées. Ceci offrant la possibilité, par exemple dans un contexte de régression, de générer un graphe moléculaire basé sur une valeur de propriété recherchée.

Ce chapitre introduit des modèles de prédiction interprétables travaillant sur des données de type graphe et adaptés de nos approches de régression et classification décrites dans le chapitre précédent, sans problème de pré-image. Le concept clé consiste à développer une fonction de transformation non-linéaire appliquée aux graphes de données qui est intentionnellement inversible. L'espace d'encodage appris est conçu pour séparer ou organiser linéairement les échantillons permettant l'obtention de bonnes performances par l'application de méthodes prédictives linéaires dans cet espace. De plus, des pré-images peuvent être générées simplement en appliquant la transformation inverse sur n'importe quel échantillon d'intérêt de l'espace de caractéristiques.

Pour cela, nos approches, consistant à produire une fonction non-linéaire réversible, s'inspire des avancées récentes en matière de modèles génératifs dans le domaine de l'apprentissage machine. La Section 1.3 présente ces différentes approches ainsi que leur extension sur des données de type graphe. En considérant le même formalisme, nous considérons l'utilisation d'un GraphNF (voir Section 1.3.4) pour définir notre transformation inversible.

Nos résultats expérimentaux (Section 5.3) démontrent l'efficacité des méthodologies proposées grâce à l'utilisation de l'architecture MoFlow [136], un NF de graphes (GraphNF) utilisant des couches de couplage, introduit dans la Section 1.3.4, pour opérer sur des graphes représentés par une combinaison d'une matrice de caractéristiques et d'un tenseur d'adjacence. Les expériences ont été menées sur des jeux de données de graphes bien connus afin de démontrer la pertinence de notre approche pour traiter les tâches de classification et de régression sur des données de type graphe tout en produisant un espace de représentation de bonne qualité exempt du problème de pré-image.

La suite du chapitre est structurée comme suit : Nos contributions sont présentées dans la Section 5.2 divisée en deux grandes parties. La Section 5.2.1 décrit en détails notre méthodologie utilisée en classification, tandis que la Section 5.2.2 présente celle utilisée en régression. Celles-ci décrivent en plusieurs sous-parties l'apprentissage du GraphNF, l'application de modèle prédictif dans l'espace de caractéristiques ainsi que les opérations de génération de pré-images. Les expériences sont présentées dans la Section 5.3, suivies de la conclusion dans la Section 5.4.

5.2 Approches proposées

Nous présentons l'adaptation de nos approches aux données de type graphe sur deux tâches distinctes, à savoir la classification et la régression. Afin de traiter ces tâches, nous avons développé deux approches différentes basées sur les GraphNF (Section 1.3.4), où l'espace latent $\mathcal{Z}$ généré par le GraphNF est considéré comme l'espace de caractéristiques d'intérêt. Par conséquent, le problème de pré-image est éliminé par conception.

Étant donné l'espace des observations de graphes $\mathcal{G}$, les GraphNF sont des NF conçus pour travailler avec des données de type graphe. Cependant, il est important de noter que les graphes sont des structures de données complexes pouvant être représentées de différentes manières (voir Section 1.1.3). Ainsi, nous définissons de manière générale la fonction réversible d'un GraphNF, travaillant directement sur l'espace des graphes $\mathcal{G}$, par $\Phi_{\mathcal{G}} : \mathcal{G} \to \mathcal{Z}$. Certains GraphNF sont capables de traiter directement l'espace des graphes $\mathcal{G}$, comme GraphAF [118], un GraphNF auto-régressif qui modélise les graphes comme une séquence de décisions. D'autres GraphNF, comme MoFlow [136], opèrent sur des représentations matricielles de graphe. Cette représentation considère le cas où chaque graphe est divisé en une matrice de caractéristiques et un tenseur d'adjacence, c'est-à-dire $G = (\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e}$ où chaque graphe est représenté par maximum n nœuds de d dimensions et un ensemble d'arêtes caractérisées par e dimensions. Dans ce cas, nous définissons la fonction réversible générée par le GraphNF avec $\Phi : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e} \to \mathcal{Z}$. Cette représentation permet de travailler directement avec une représentation complète du graphe, mais présente certains inconvénients tels que des coûts de mémoire potentiellement élevés pour les graphes de taille importante et une sensibilité aux permutations des représentations matricielles des graphes. Cette dernière constitue un inconvénient majeur, car la non-unicité de la représentation d'un graphe de n nœuds entraîne la présence de $n!$ représentations différentes dans l'espace de caractéristiques généré par le modèle. Malgré cela, nous considérons que par l'application de nos approches d'apprentissages supervisées, même en présence de plusieurs représentations possibles pour un même graphe, une fois le modèle entraîné, ces représentations générées seront conformes à la structure de l'espace de caractéristiques souhaité.

Dans ce chapitre, nous considérons l'utilisation de GraphNF travaillant sur les matrices de caractéristiques et tenseurs d'adjacence pour représenter les graphes, où chaque donnée est représentée par une représentation latente en deux parties qui correspond à la matrice des caractéristiques de nœuds et au tenseur d'adjacence. De plus, nous proposons de concaténer les représentations aplaties de ces deux parties pour obtenir la représentation des données dans l'espace de caractéristiques $\mathcal{Z}$, correspondant à un vecteur à D dimensions dans $\mathbb{R}$ déterminé par $D = n^2 \times e + n \times d$.

Soit $\mathcal{D} = \{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\}$ un jeu de données avec $G_i \in \mathcal{G}$ et $y_i \in Y$ la valeur d'étiquette associée à G_i. L'idée, illustrée par la Figure 5.1, implique les trois étapes suivantes :

(*i*) À l'aide d'une fonction réversible $\Phi : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e} \to \mathbb{R}^D$ apprise par un GraphNF, la première étape vise à générer une distribution dans $\mathcal{Z}$ formée à partir de multiples distributions gaussiennes basées sur les étiquettes, chacune d'entre elles étant paramétrée par différentes moyennes et matrices de

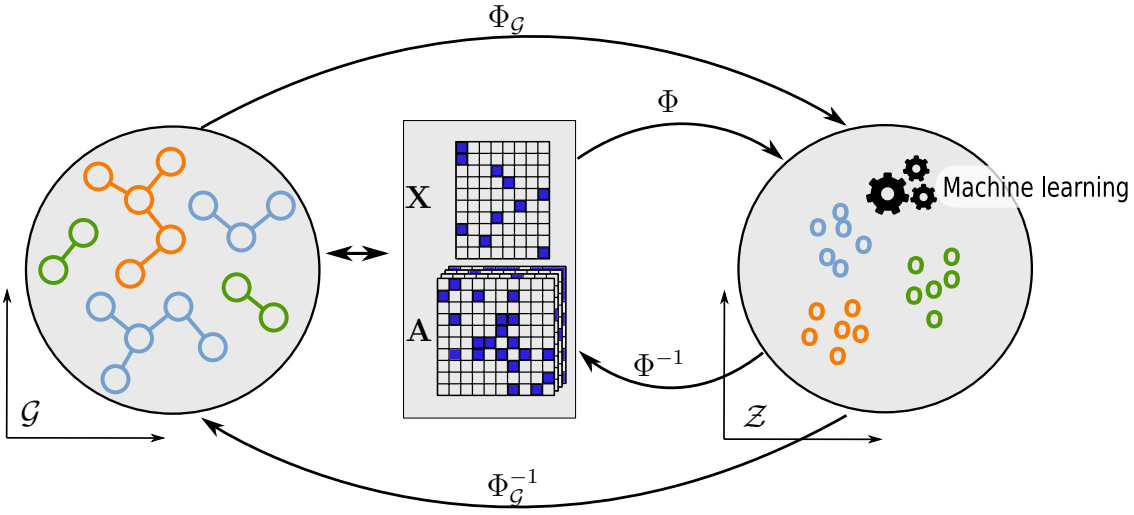


FIGURE 5.1 – Formalisme général de nos contributions. En considérant l'espace de caractéristiques $\mathcal{Z}$ généré par la transformation Φ d'un GraphNF, nous opérons efficacement des modèles ML/PR linéaires dans cet espace. Cette transformation prend en entrée une paire de matrice de caractéristiques de nœud $\mathbf{X}_i$ et de tenseur d'adjacence $\mathbf{A}_i$ décrivant un graphe $G_i \in \mathcal{G}$ et génère une représentation associée dans $\mathcal{Z}$. De plus, le problème de pré-image est résolu en utilisant la fonction inverse Φ^{-1} du modèle de GraphNF. À noter que l'utilisation d'une fonction de GraphNF $\Phi_{\mathcal{G}} : \mathcal{G} \to \mathcal{Z}$ travaillant directement sur $\mathcal{G}$ est possible.

covariance.

(*ii*) En utilisant l'espace de caractéristiques $\mathcal{Z}$ appris, nous pouvons appliquer des opérations simples ou des algorithmes plus sophistiqués (par exemple, les machines à vecteurs de support pour la classification ou la ridge régression pour la régression).

(*iii*) Enfin, puisque nous utilisons un GraphNF comme modèle, qui est un modèle inversible, la transformation Φ et son inverse Φ^{-1} sont tous deux produits par nos algorithmes d'apprentissage. Il est donc possible de calculer le graphe pré-image de n'importe quel point de l'espace de caractéristiques $\mathcal{Z}$.

Les sections suivantes présentent deux approches appliquées aux graphes spécifiques à une tâche, à savoir classification (Section 5.2.1) et régression (Section 5.2.2), permettant à la fois la génération de prédiction et de pré-image.

5.2.1 Modèle réversible de classification

Plusieurs méthodes d'apprentissage machine impliquent la définition d'un modèle prédictif sur une représentation de données apprises dans un espace particulier. Notre objectif dans cette section est d'utiliser un modèle de GraphNF, dans un contexte de classification, pour apprendre des représentations latentes des données de graphes linéairement séparables. La fonction inverse fournie par le GraphNF nous permettra de générer la pré-image sous forme de graphe de n'importe quel point de

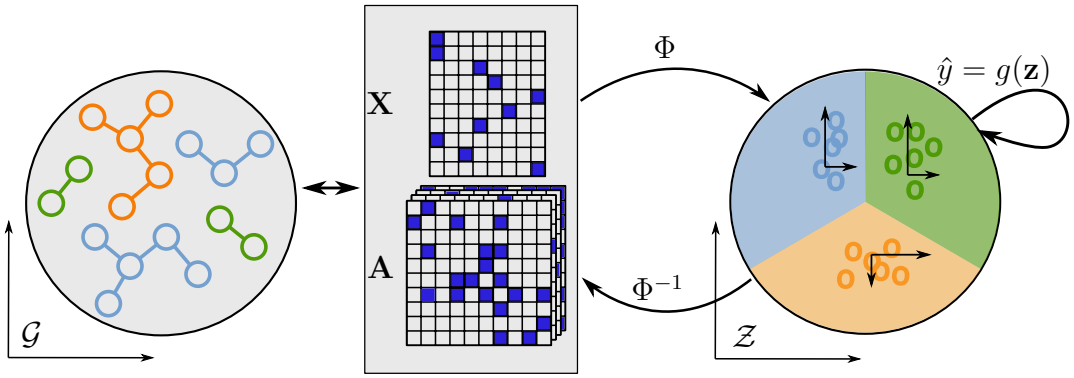


FIGURE 5.2 – Illustration de notre approche pour une tâche de classification, où la couleur de chaque échantillon correspond à son étiquette. La transformation Φ apprise par le GraphNF sur les représentations matricielles de graphe $(\mathbf{X}, \mathbf{A})$ permet la génération d'un espace de caractéristiques $\mathcal{Z}$ où les représentations de graphes sont linéairement séparables.

l'espace de caractéristiques $\mathcal{Z}$, qui est l'espace où le modèle prédictif opère. Ainsi, le défi que représente la création d'un classifieur non-linéaire qui ne souffre pas du problème de pré-image peut être réduit à l'apprentissage d'une transformation inversible permettant une représentation efficace et linéairement séparable des données. Notre approche complète de classification est illustrée dans la Figure 5.2.

Apprentissage du modèle

Les modèles de GraphNF traditionnels intègrent les données de manière que la distribution dans l'espace latent suive une densité de probabilité cible, généralement une distribution gaussienne. Toutefois, cette configuration ne permet pas de diviser les données en fonction de leurs classes. Dans cette section, nous présentons une nouvelle formulation supervisée de l'apprentissage d'un GraphNF tenant compte des classes des données basée sur notre approche de classification introduit dans le chapitre précédent. Pour cela, la fonction objectif du GraphNF a été modifiée pour permettre un encodage des graphes dans des régions spécifiques de l'espace de caractéristiques $\mathcal{Z} \subset \mathbb{R}^D$ en fonction de leur classe respective. Ainsi, la fonction de transformation apprise $\Phi : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e} \to \mathcal{Z}$ permet la génération d'un espace de caractéristiques dans lequel les représentations de données sont linéairement séparables.

Soit $\mathcal{D} = \{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\}$ un jeu de données où $G_i \in \mathcal{G}$ représente un graphe de données observé et $y_i \in Y = \{1 \dots C\}$ son encodage de classe. De manière similaire à la Section 4.2.1, une distribution gaussienne dans $\mathcal{Z}$ est attribuée à chaque classe $c \in \{1 \dots C\}$, à savoir

$$P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma}_c)}} e^{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{z}-\boldsymbol{\mu}_c))}$$

où la moyenne $\boldsymbol{\mu}_c$ et la matrice de covariance $\boldsymbol{\Sigma}_c$ correspondent aux paramètres de

la gaussienne associée à la classe c. De cela, nous devons adapter l'entraînement du GraphNF pour que chaque représentation de graphe dans $\mathcal{Z}$ soit ajustée à la distribution gaussienne appropriée en fonction de la classe c codée par y_i. Par conséquent, le terme de log-probabilité utilisé dans la fonction de perte d'apprentissage (équation (1.8)) est redéfini comme suit :

$$\begin{aligned}\log P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) = &-\frac{1}{2}\left(D\log(2\pi) + (\mathbf{z}-\boldsymbol{\mu}_c)^\top \boldsymbol{\Sigma}_c^{-1}(\mathbf{z}-\boldsymbol{\mu}_c)\right) \\ &- \log(\det(\boldsymbol{\Sigma}_c)).\end{aligned} \tag{5.1}$$

En utilisant cette fonction, les données appartenant à une même classe sont ajustées à une région spécifique de l'espace de caractéristiques $\mathcal{Z}$, créant ainsi des représentations linéairement séparables entre les différentes classes.

Comme présenté dans la Section 4.2.1, nous souhaitons apprendre la position des différentes distributions gaussiennes dans l'espace de caractéristique afin de tenir compte des relations potentiellement existantes entre les classes dans l'ensemble de données. Ainsi, nous incitons le modèle à distinguer les différentes classes en incorporant un terme de perte supplémentaire dans la fonction objectif du GraphNF pour l'apprentissage des moyennes des distributions de chaque classe $\boldsymbol{\mu}_c$. Par conséquent, notre optimisation consiste à minimiser deux pertes. La première décrit le changement de densité d'un échantillon à l'aide de la formule de changement de variable et peut être exprimée comme suit :

$$\mathcal{L}_{\text{nf}}(\mathbf{X}_i, \mathbf{A}_i, \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) = -\log P_{\mathcal{Z}}(\Phi(\mathbf{X}_i, \mathbf{A}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) - \log\left|\det\left(\frac{\partial\Phi(\mathbf{X}_i, \mathbf{A}_i)}{\partial\mathbf{X}_i\partial\mathbf{A}_i}\right)\right|,$$

où $\log P_{\mathcal{Z}}(\Phi(\mathbf{X}_i, \mathbf{A}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$ est défini suivant l'équation (5.1). La deuxième perte se concentre sur la proximité des distributions définie de la façon suivante :

$$\mathcal{L}_\mu = -\log\left(1 + \frac{1}{C^2}\sum_{i=1}^{C}\sum_{j=1}^{C} \| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \|_2^2\right),$$

où $\boldsymbol{\mu}_i$ et $\boldsymbol{\mu}_j$ correspondent aux moyennes des classes i et $j \in \{1 \dots C\}$. La minimisation de ce terme encourage le modèle à séparer les centres des distributions.

En combinant ces deux termes, nous pouvons définir la fonction de perte finale pour une donnée observée et son étiquette correspondante comme suit :

$$\mathcal{L}(\mathbf{X}_i, \mathbf{A}_i, y_i) = \mathcal{L}_{\text{nf}}(\mathbf{X}_i, \mathbf{A}_i, \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) + \beta\mathcal{L}_\mu, \tag{5.2}$$

où l'hyperparamètre β est utilisé pour équilibrer les deux termes.

De plus, pour distinguer les classes dans l'espace de caractéristiques, nous associons des dimensions spécifiques de $\mathcal{Z}$ à chaque classe, par la définition de C différentes matrices de covariance. Pour cela, nous définissons celles-ci avec un ensemble de valeurs propres λ_i spécifique à chaque classe. De plus, étant donné la structure en deux parties de la représentation latente et le fait que notre représentation latente dans $\mathcal{Z}$ soit une concaténation de ces représentations, nous proposons de définir ces valeurs propres séparément pour chaque partie. Ainsi, pour chacune de ces deux parties, nous appliquons la méthode d'assignation des vecteurs propres suivante :

(*i*) Pour commencer, nous calculons le nombre maximal de dimensions à attribuer à chaque classe en utilisant la formule $k = \lfloor \frac{l}{C} \rfloor$, où l désigne le nombre de dimensions et $\lfloor . \rfloor$ représente l'arrondi inférieur d'un nombre réel, c'est-à-dire le plus grand entier qui est inférieur ou égal à ce nombre.

(*ii*) Ensuite, pour chaque classe c, nous attribuons des valeurs élevées à k valeurs propres différentes selon

$$(\underbrace{\lambda_1, \lambda_2, \ldots, \lambda_k}_{c=1}, \underbrace{\lambda_{k+1}, \lambda_{k+2}, \ldots, \lambda_{2k}}_{c=2}, \ldots, \lambda_l). \tag{5.3}$$

De cette manière, chaque distribution s'étendra sur les dimensions qui lui ont été attribuées.

(*iii*) Enfin, les $l-k$ valeurs propres restantes sont calculées pour que le déterminant de la matrice de covariance soit égal à 1, préservant ainsi la dispersion des données de l'espace des observations $\mathcal{G}$ dans l'espace de caractéristiques $\mathcal{Z}$. Ainsi, étant donné des matrices de covariance diagonales, nous calculons le déterminant par le produit des valeurs propres et déterminons la valeur à attribuer aux valeurs propres restantes avec $\epsilon = \sqrt[l-k]{1/\lambda_1^k}$.

L'application de cette méthode aux deux parties de la représentation latente permet l'obtention des D valeurs propres pour chaque classe, permettant ainsi la construction des matrices de covariance en utilisant l'équation (3.1), à savoir

$$\mathbf{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i,$$

avec les vecteurs propres $\mathbf{v}_i$ correspondant à un ensemble de vecteur orthogonaux tel que défini par l'équation (3.2).

Soit Θ l'ensemble des paramètres de Φ, et, soit Ω l'ensemble des paramètres optimisables défini par $\Omega = \{\Theta, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_C\}$. Afin d'estimer les paramètres de Ω, nous utilisons un algorithme stochastique de descente de gradient qui minimise la fonction de perte (équation (5.2)) sur un lot de m échantillons sélectionnés de manière aléatoire à partir du jeu de données d'apprentissage à chaque itération.

$$\Omega \leftarrow \Omega - \eta \sum_{k \in I} \nabla_\Omega \mathcal{L}(\mathbf{X}_i, \mathbf{A}_i, y_i) \tag{5.4}$$

où le taux d'apprentissage est représenté par η et le lot de données sélectionné est noté I. L'Alg. 9 présente notre processus d'apprentissage du GraphNF dans un contexte de classification.

Classifieur dans l'espace de caractéristiques

Notre approche permet un apprentissage d'espace de caractéristiques $\mathcal{Z}$ où les graphes sont linéairement séparables pour une tâche de classification donnée. Par conséquent, travaillant sur des tâches de classification, la structure de l'espace de caractéristiques $\mathcal{Z}$ généré permet l'apprentissage d'un classifieur linéaire efficace par l'utilisation d'algorithmes du type machines à vecteurs de support (SVM) [23] ou régression logistique [84]. Nous notons $g : \mathcal{Z} \rightarrow \{1 \ldots C\}$ ce modèle. Ainsi, les

Algorithm 9 Procédure d'entraînement du GraphNF pour un contexte de classification de graphe.

Require: Ensemble d'apprentissage : $\mathcal{D}$, coefficient de compromis : β, taux d'apprentissage : η, taille de lot : m, nombre d'époques : E

$(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_D) \leftarrow$ équation (3.2)
for $c \in \{1 \dots C\}$ **do**
  $(\lambda_1 \quad \lambda_2 \quad \dots \quad \lambda_D) \leftarrow$ équation (5.3)
  $\mathbf{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
end for
$(\boldsymbol{\mu}_1 \quad \dots \quad \boldsymbol{\mu}_C) = (\mathbf{0} \quad \dots \quad \mathbf{0})$
$(\Theta, \Phi, \Phi^{-1}) \leftarrow$ Initialisation du modèle de GraphNF
$\Omega = \{\Theta, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \dots, \boldsymbol{\mu}_C\}$
for $e \in \{1, \dots, E\}$ **do**
  Mélange aléatoire de lu jeu de données $\mathcal{D}$
  $\{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\} \leftarrow \mathcal{D}$
  $\{(\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2) \dots (\mathbf{X}_N, \mathbf{A}_N)\} \leftarrow$ Conversion en représentations matricielles des graphes $\{G_1, G_2 \dots G_N\}$
  for $i \in \{1, \dots, N/m\}$ **do**
    $\mathcal{L}_\mu = -\log\left(1 + \frac{1}{C^2} \sum_{i=1}^{C} \sum_{j=1}^{C} \| \boldsymbol{\mu}_i - \boldsymbol{\mu}_j \|_2^2\right)$
    for $k \in \{m(i-1)+1, \dots, mi\}$ **do**
      $\mathbf{z}_k = \Phi(\mathbf{X}_k, \mathbf{A}_k)$
      $\mathcal{L}_{\text{nf}} = -\log P_{\mathcal{Z}}(\mathbf{z}_k, \boldsymbol{\mu}_{y_k}, \mathbf{\Sigma}_{y_k}) - \log \left|\det\left(\frac{\partial \mathbf{z}_k}{\partial \mathbf{X}_k \partial \mathbf{A}_k}\right)\right|$
      $\mathcal{L}(\mathbf{X}_k, \mathbf{A}_k, y_k) = \mathcal{L}_{\text{nf}} + \beta \mathcal{L}_\mu$
    end for
    $\Omega \leftarrow \Omega - \eta \sum_{k=m(i-1)+1}^{mi} \nabla_\Omega \mathcal{L}(\mathbf{X}_k, \mathbf{A}_k, y_k)$
    $(\Phi, \Phi^{-1}) \leftarrow$ Mise à jour des fonctions avec les paramètres Ω
  end for
end for
return $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_1 \quad \dots \quad \boldsymbol{\mu}_C))$

prédictions sont effectuées en transposant les graphes dans $\mathcal{Z}$ à l'aide de Φ et en obtenant une étiquette prédite avec notre modèle de prédiction linéaire g. De plus, étant donné que Φ correspond à une transformation non-linéaire, la définition d'un modèle prédictif linéaire g dans l'espace de caractéristiques $\mathcal{Z}$ est équivalente à la définition d'un modèle prédictif non-linéaire $f : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e} \to \{1 \dots C\}$. Ce dernier est décrit de la façon suivante :

$$f(\mathbf{X}, \mathbf{A}) = g(\Phi(\mathbf{X}, \mathbf{A})). \tag{5.5}$$

Génération de pré-image

De plus, la disponibilité de Φ^{-1} permet le calcul de graphe pré-image de n'importe quel point d'intérêt de l'espace de caractéristiques, éliminant ainsi le problème de pré-image. Dans un contexte de classification, nous pouvons générer des graphes pré-images de prototype de classe ou de point aléatoire de la gaussienne paramétrée par $(\boldsymbol{\mu}_c, \mathbf{\Sigma}_c)$ pour obtenir des graphes de la classe correspondante c. L'Alg. 10 montre

Algorithm 10 Procédure de génération de graphe pré-image avec un GraphNF basé sur la classification.

Require: Modèle GraphNF : $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_C))$, classe : c

1: $(\mathbf{v}_1 \quad \mathbf{v}_2 \quad \ldots \quad \mathbf{v}_D) \leftarrow$ équation (3.2)
2: $(\lambda_1 \quad \lambda_2 \quad \ldots \quad \lambda_D) \leftarrow$ équation (5.3)
3: $\boldsymbol{\Sigma}_c = \sum_{i=1}^{D} \lambda_i \, \mathbf{v}_i^\top \, \mathbf{v}_i$
4: $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
5: $(\hat{\mathbf{X}}, \hat{\mathbf{A}}) = \Phi^{-1}(\mathbf{z})$
6: $\hat{G} \leftarrow$ Conversion sous forme de graphe de $(\hat{\mathbf{X}}, \hat{\mathbf{A}})$
7: **return** $\hat{G}$

la procédure de génération de pré-image de graphe aléatoire sachant une classe c donnée.

Extension à l'approche

Enfin, nous proposons une extension de notre approche dans le but d'améliorer ses performances. Étant donné un espace des observations $\mathcal{G}$ à D dimensions, l'utilisation d'un GraphNF implique la génération d'un espace de caractéristiques $\mathcal{Z}$ limité à la même dimension que $\mathcal{G}$. En nous inspirant des méthodes à noyaux de graphe permettant l'encodage des graphes dans un espace de dimension potentiellement supérieure à celle des graphes, nous proposons d'accroître l'expressivité de l'espace des caractéristiques en augmentant manuellement le nombre de dimensions dans l'espace des observations. Cette augmentation de dimensions a pour effet d'accroître également le nombre de dimensions de l'espace des caractéristiques, ce qui peut être bénéfique lors de l'apprentissage pour une meilleure optimisation des pertes utilisées.

Compte tenu de la structure des données, nous proposons d'augmenter le nombre de dimensions des caractéristiques des nœuds par l'ajout de p dimensions initialisées à 0 aux données observées. Pour cela, nous représentons les données de graphes par $G = (\mathbf{X}, \mathbf{A})$ avec le tenseur d'adjacence inchangé caractérisé par $\mathbf{A} \in \mathbb{R}^{n \times n \times e}$ et $\mathbf{X}$ la matrice des caractéristiques définie par $\mathbf{X} \in \mathbb{R}^{n \times (d+p)}$. La Figure 5.3 présente une illustration de cette extension, pour une valeur de $p = 2$, appliquée lors de la conversion du graphe en représentation matricielle.

5.2.2 Modèle réversible de régression

Parmi les différentes tâches de prédiction, la régression est également l'une des plus courantes. Contrairement à la classification, notre objectif n'est pas de créer des représentations linéairement séparables des graphes, mais de générer des représentations linéairement organisées en fonction de leurs valeurs d'étiquette. Par conséquent, notre proposition implique une nouvelle formulation supervisée de l'entraînement d'un GraphNF qui prend en compte les étiquettes quantitatives associées aux graphes. Pour ce faire, nous adaptons la fonction objective du GraphNF pour intégrer les données sur la base de leurs valeurs quantitatives. Ainsi, le GraphNF est conçu pour représenter les données dans un espace de caractéristiques adapté

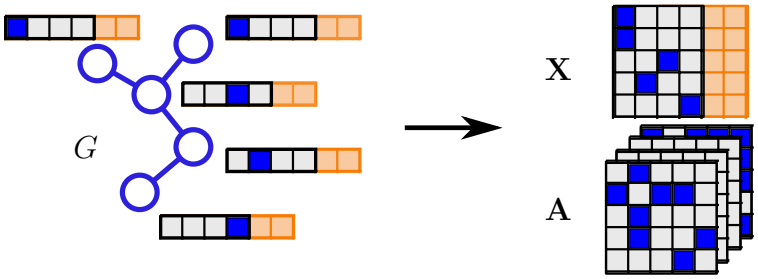


FIGURE 5.3 – Illustration de l'extension à l'approche de classification sur graphe par l'augmentation (couleur orange) du nombre de dimensions des caractéristiques de nœud. Avec $p = 2$, la conversion du graphe en représentation matricielle résulte en une augmentation de la taille de la matrice de caractéristiques de nœuds.

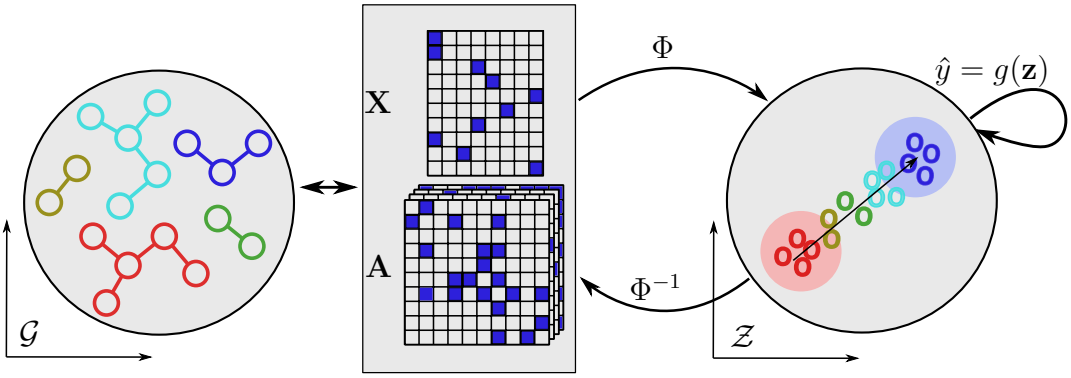


FIGURE 5.4 – Illustration de notre approche pour une tâche de régression, où la couleur de chaque échantillon correspond à son étiquette quantitative. La transformation Φ, apprise par le GraphNF à partir des représentations matricielles de graphe $(\mathbf{X}, \mathbf{A})$, permet de générer un espace de caractéristiques $\mathcal{Z}$ où les représentations de graphes sont organisées de manière linéaire.

à la régression linéaire. L'illustration de la Figure 5.4 montre notre approche de régression de graphe dans sa globalité.

Apprentissage du modèle

Considérons un jeu de données $\mathcal{D} = \{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\}$ composé de données de type graphe désignées par $G_i \in \mathcal{G}$ et de leurs étiquettes quantitatives correspondantes désignées par $y_i \in Y \subset \mathbb{R}$. [1] De la même manière que dans la section précédente, nous constituons notre espace de caractéristiques à l'aide de distributions gaussiennes, chacune paramétrée par une moyenne $\boldsymbol{\mu}$ et une matrice de covariance

1. Nous travaillons dans ce document sur une seule valeur d'étiquette quantitative, cependant son extension à des vecteurs d'étiquettes est facilement réalisable.

$\boldsymbol{\Sigma}$ définie par

$$P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^D \det(\boldsymbol{\Sigma})}} e^{-\frac{1}{2}(\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu}))}.$$

Ainsi, nous pouvons définir la log-probabilité d'appartenir à une gaussienne comme suit

$$\begin{aligned} \log P_{\mathcal{Z}}(\mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = &- \frac{1}{2}\Big(D\log(2\pi) + (\mathbf{z}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{z}-\boldsymbol{\mu})\Big) \\ &- \log(\det(\boldsymbol{\Sigma})). \end{aligned} \tag{5.6}$$

De manière similaire à la Section 4.2.2, nous souhaitons organiser nos représentations de données de manière linéaire dans l'espace de caractéristiques. Pour ce faire, nous associons chaque donnée à une distribution gaussienne alignée suivant un axe, en calculant une moyenne de distribution interpolée sur la base de la valeur quantitative de l'étiquette. Cette interpolation est réalisée en définissant deux distributions gaussiennes extrêmes correspondant aux valeurs maximales et minimales des étiquettes quantitatives. Pour cela, nous définissons deux distributions gaussiennes paramétrées par ($\boldsymbol{\mu}_{\min}$, $\boldsymbol{\Sigma}_{\min}$) et ($\boldsymbol{\mu}_{\max}$, $\boldsymbol{\Sigma}_{\max}$) respectivement associées à la valeur minimale ($\min(Y)$) et maximale ($\max(Y)$) de Y. Par souci de simplicité, nous utilisons des distributions gaussiennes isotropes, c'est-à-dire que leurs matrices de covariance sont définies par $\boldsymbol{\Sigma}_{\min} = \boldsymbol{\Sigma}_{\max} = \sigma^2 \mathbb{I}_D$, où la matrice identité $D \times D$ est notée $\mathbb{I}_D$, et $\sigma^2 \in \mathbb{R}$ représente la variance de la distribution.

Pour réaliser le processus d'interpolation, un coefficient d'appartenance τ_{y_i} est attribué à chaque échantillon (G_i, y_i) en fonction de sa valeur quantitative. Ce coefficient est calculé avec

$$\tau_{y_i} = \frac{y_i - \min(Y)}{\max(Y) - \min(Y)}. \tag{5.7}$$

La moyenne gaussienne interpolée $\boldsymbol{\mu}_{y_i}$ est ainsi calculée en utilisant le coefficient résultant avec

$$\boldsymbol{\mu}_{y_i} = \tau_{y_i}\, \boldsymbol{\mu}_{\min} + (1 - \tau_{y_i})\, \boldsymbol{\mu}_{\max}\,. \tag{5.8}$$

Pour que notre méthode d'interpolation soit utile, il est essentiel que les emplacements des distributions gaussiennes dans $\mathcal{Z}$ représentés par leurs moyennes ($\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}$) soient distincts et suffisamment séparés. Nous proposons d'apprendre ces moyennes ($\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}$) pendant l'apprentissage du GraphNF de la même manière que dans la section précédente. Pour ce faire, nous incorporons une fonction objectif supplémentaire dans la fonction objectif du GraphNF, visant à maximiser la séparabilité des gaussiennes.

La fonction de perte est donc composée de deux termes. Le premier terme, basé sur l'équation (1.8), applique la formule de changement de variable pour décrire le changement de densité d'un échantillon. Plus précisément, pour chaque graphe G_i, nous effectuons sa conversion vers une représentation matricielle $(\mathbf{X}_i, \mathbf{A}_i)$, sur laquelle l'équation suivante est utilisée :

$$\mathcal{L}_{\text{nf}}(\mathbf{X}_i, \mathbf{A}_i, \boldsymbol{\mu}_{y_i}) = -\log P_{\mathcal{Z}}(\Phi(\mathbf{X}_i, \mathbf{A}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i}) - \log \left| \det \left(\frac{\partial \Phi(\mathbf{X}_i, \mathbf{A}_i)}{\partial \mathbf{X}_i \partial \mathbf{A}_i} \right) \right|.$$

Dans la formule précédente, $\log P_{\mathcal{Z}}(\Phi(\mathbf{X}_i, \mathbf{A}_i), \boldsymbol{\mu}_{y_i}, \boldsymbol{\Sigma}_{y_i})$ fait référence à l'équation (5.6), qui utilise les paramètres gaussiens interpolés ($\boldsymbol{\mu}_{y_i}$, $\boldsymbol{\Sigma}_{y_i}$). La moyenne $\boldsymbol{\mu}_{y_i}$ est calculée à l'aide de l'équation (5.8), tandis que la matrice de covariance $\boldsymbol{\Sigma}_{y_i}$ est définie de la même manière que les autres distributions gaussiennes, à savoir $\boldsymbol{\Sigma}_{y_i} = \sigma^2 \mathbb{I}_D$. Le second terme favorise la séparation entre les deux gaussiennes extrêmes avec

$$\mathcal{L}_\mu = -\log\left(1 + \| \boldsymbol{\mu}_{\min} - \boldsymbol{\mu}_{\max} \|_2^2\right).$$

Ainsi, la fonction de perte finale est définie comme suit :

$$\mathcal{L}(\mathbf{X}_i, \mathbf{A}_i, y_i) = \mathcal{L}_{\text{nf}}(\mathbf{X}_i, \mathbf{A}_i, \boldsymbol{\mu}_{y_i}) + \beta \mathcal{L}_\mu, \tag{5.9}$$

avec β correspondant au coefficient de compromis entre les deux termes.

L'optimisation des paramètres est appliquée de manière similaire à notre approche de classification de graphe en suivant l'équation (5.4) avec un ensemble de paramètres optimisables définis comme $\Omega = \{\Theta, \boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}\}$. Ainsi, notre approche d'apprentissage de GraphNF pour la régression de graphe est donnée dans l'Alg. 11.

Régression dans l'espace de caractéristiques

Notre approche permet de créer un espace de caractéristiques $\mathcal{Z}$ sur mesure, où les représentations de graphe sont linéairement organisées. Par conséquent, un modèle prédictif simple et efficace peut être défini dans $\mathcal{Z}$. Nous notons $g : \mathcal{Z} \to \mathbb{R}$ un modèle prédictif linéaire. Comme Φ est une fonction non-linéaire, définir le modèle prédictif linéaire g dans $\mathcal{Z}$ équivaut à définir un modèle prédictif non-linéaire $f : \mathbb{R}^{n\times d} \times \mathbb{R}^{n\times n\times e} \to \mathbb{R}$, décrit comme suit

$$f(\mathbf{X}, \mathbf{A}) = g(\Phi(\mathbf{X}, \mathbf{A})).$$

Soit $g(\mathbf{z}) = \mathbf{z}^\top \boldsymbol{\varphi}^*$ le modèle linéaire prédictif $g : \mathcal{Z} \to \mathbb{R}$, où $\boldsymbol{\varphi}^* \in \mathcal{Z}$ sont les paramètres optimaux à estimer. Sans perdre en généralité, nous pouvons considérer qu'il s'agit d'une régression ridge dans l'espace de caractéristiques, permettant ainsi de trouver les meilleurs paramètres $\boldsymbol{\varphi}^*$ en minimisant l'erreur quadratique moyenne régularisée, exprimée par

$$\min_{\boldsymbol{\varphi}} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \Phi(\mathbf{X}_i, \mathbf{A}_i)^\top \boldsymbol{\varphi}\right)^2 + \lambda \|\boldsymbol{\varphi}\|_2^2, \tag{5.10}$$

où l'importance du terme de régularisation est pondérée par λ. L'utilisation du théorème de représentation (1.3) nous permet d'exprimer la solution optimale par $\boldsymbol{\varphi}^* = \sum_{i=1}^N \alpha_i \Phi(\mathbf{X}_i, \mathbf{A}_i)$, nous conduisant au problème d'optimisation suivant :

$$\begin{aligned} \min_{\alpha_1,\ldots,\alpha_N} \frac{1}{N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{N} \alpha_j \kappa((\mathbf{X}_j, \mathbf{A}_j), (\mathbf{X}_i, \mathbf{A}_i))\right)^2 \\ + \lambda \sum_{i,j=1}^{N} \alpha_i \alpha_j \kappa((\mathbf{X}_j, \mathbf{A}_j), (\mathbf{X}_i, \mathbf{A}_i)), \end{aligned} \tag{5.11}$$

où $\kappa((\mathbf{X}_j, \mathbf{A}_j), (\mathbf{X}_i, \mathbf{A}_i)) = \langle \Phi(\mathbf{X}_j, \mathbf{A}_j), \Phi(\mathbf{X}_i, \mathbf{A}_i) \rangle_{\mathcal{Z}}$. Cette formulation met en évidence l'applicabilité des méthodes à noyaux dans le calcul de prédiction pour la

Algorithm 11 Procédure d'entraînement du GraphNF pour un contexte de régression de graphe.

Require: Ensemble d'entraînement : $\mathcal{D}$, coefficient de compromis : β, variance des gaussiennes : σ^2, taux d'apprentissage : η, taille de lot : m, nombre d'époques : E

$(\boldsymbol{\Sigma}_{\min}, \boldsymbol{\Sigma}_{\max}) = (\sigma^2\,\mathbb{I}_D, \sigma^2\,\mathbb{I}_D)$
$(\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}) = (\mathbf{0}, \mathbf{0})$
$(\Theta,\ \Phi,\ \Phi^{-1}) \leftarrow$ Initialisation du modèle de GraphNF
$\Omega = \{\Theta, \boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}\}$
for $e \in \{1, \ldots, E\}$ **do**
  Mélange aléatoire du jeu de données $\mathcal{D}$
  $\{(G_1, y_1), (G_2, y_2) \ldots (G_N, y_N)\} \leftarrow \mathcal{D}$
  $\{(\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2) \ldots (\mathbf{X}_N, \mathbf{A}_N)\} \leftarrow$ Conversion en représentations matricielles des graphes $\{G_1, G_2 \ldots G_N\}$
  for $i \in \{1, \ldots, N/m\}$ **do**
    $\mathcal{L}_\mu = -\log\left(1 + \| \boldsymbol{\mu}_{\min} - \boldsymbol{\mu}_{\max} \|_2^2\right)$
    for $k \in \{m(i-1)+1, \ldots, mi\}$ **do**
      $\mathbf{z}_k = \Phi(\mathbf{X}_k, \mathbf{A}_k)$
      $\tau_{y_k} = \frac{y_k - \min(Y)}{\max(Y) - \min(Y)}$
      $\boldsymbol{\mu}_{y_k} = \tau_{y_k}\,\boldsymbol{\mu}_{\min} + (1 - \tau_{y_k})\,\boldsymbol{\mu}_{\max}$
      $\boldsymbol{\Sigma}_{y_k} = \boldsymbol{\Sigma}_{\min} = \boldsymbol{\Sigma}_{\max}$
      $\mathcal{L}_{\text{nf}} = -\log P_{\mathcal{Z}}(\mathbf{z}_k, \boldsymbol{\mu}_{y_k}, \boldsymbol{\Sigma}_{y_k}) - \log \left|\det\left(\frac{\partial\, \mathbf{z}_k}{\partial \mathbf{X}_k \partial \mathbf{A}_k}\right)\right|$
      $\mathcal{L}(\mathbf{X}_k, \mathbf{A}_k, y_k) = \mathcal{L}_{\text{nf}} + \beta \mathcal{L}_\mu$
    end for
    $\Omega \leftarrow \Omega - \eta \sum_{k=m(i-1)+1}^{mi} \nabla_\Omega \mathcal{L}(G_k, y_k)$
    $(\Phi,\ \Phi^{-1}) \leftarrow$ Mise à jour des fonctions avec les paramètres Ω
  end for
end for
return $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}))$

régression. Ainsi, pour obtenir le vecteur solution optimal $\boldsymbol{\alpha}^*$ de ce problème d'optimisation, nous pouvons annuler son gradient, nous conduisant à

$$\boldsymbol{\alpha}^* = (\mathbf{Z}\,\mathbf{Z}^\top + \lambda \mathbb{I}_N)^{-1}\,\mathbf{y},$$

où $\mathbf{Z} = \begin{pmatrix} \Phi(\mathbf{X}_1, \mathbf{A}_1) & \cdots & \Phi(\mathbf{X}_N, \mathbf{A}_N) \end{pmatrix}^\top$ et $\mathbf{y} = \begin{pmatrix} y_1 & \cdots & y_N \end{pmatrix}^\top$ et $\mathbb{I}_N$ correspond à la matrice identité de taille $N \times N$. De plus, en utilisant le théorème de représentation (1.3), nous obtenons :

$$\boldsymbol{\varphi}^* = \mathbf{Z}^\top\,\boldsymbol{\alpha}^* = \mathbf{Z}^\top (\mathbf{Z}\,\mathbf{Z}^\top + \lambda \mathbb{I}_N)^{-1}\,\mathbf{y}\,.$$

Dans notre cas, nous définissons la transformation Φ par la fonction associée au GraphNF permettant ainsi le calcul de $\boldsymbol{\varphi}^*$ directement par la résolution de (5.10) :

$$\boldsymbol{\varphi}^* = (\mathbf{Z}^\top\,\mathbf{Z} + \lambda \mathbb{I}_D)^{-1}\,\mathbf{Z}^\top\,\mathbf{y}, \tag{5.12}$$

où $\mathbb{I}_D$ représente la matrice identité de taille $D \times D$.

Algorithm 12 Procédure de génération de graphe pré-image avec un GraphNF basé sur la régression.

Require: Modèle GraphNF : $(\Phi, \Phi^{-1}, (\boldsymbol{\mu}_{\min}, \boldsymbol{\mu}_{\max}))$, Ensemble d'apprentissage : $\mathcal{D}$, valeur quantitative : y

1: $\{(G_1, y_1), (G_2, y_2) \dots (G_N, y_N)\} \leftarrow \mathcal{D}$
2: $Y = \{y_1, y_2 \dots, y_N\}$
3: $\tau_y = \frac{y - \min(Y)}{\max(Y) - \min(Y)}$
4: $\boldsymbol{\mu}_y = \tau_y \, \boldsymbol{\mu}_{\min} + (1 - \tau_y) \, \boldsymbol{\mu}_{\max}$
5: $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \, \mathbb{I}_D)$
6: $(\hat{\mathbf{X}}, \hat{\mathbf{A}}) = \Phi^{-1}(\mathbf{z})$
7: $\hat{G} \leftarrow$ Conversion sous forme de graphe de $(\hat{\mathbf{X}}, \hat{\mathbf{A}})$
8: **return** $\hat{G}$

Ainsi, une fois $\boldsymbol{\varphi}^*$ calculé, nous obtenons le modèle prédictif optimal g et nous pouvons spécifier le modèle prédictif non-linéaire $f : \mathbb{R}^{n \times d} \times \mathbb{R}^{n \times n \times e} \to \mathbb{R}$ par la combinaison de la fonction de transformation Φ et du modèle de régression linéaire g. L'utilisation de cette méthode permet la prédiction d'une valeur quantitative de n'importe quel graphe $G \in \mathcal{G}$ par sa conversion en représentation matricielle $(\mathbf{X}, \mathbf{A})$, puis en appliquant le modèle prédictif avec

$$f(\mathbf{X}, \mathbf{A}) = \Phi(\mathbf{X}, \mathbf{A})^\top \, \boldsymbol{\varphi}^*,$$

où les paramètres optimaux $\boldsymbol{\varphi}^*$ sont obtenus par l'équation (5.12).

Génération de pré-image

L'existence de Φ^{-1} permet de calculer le graphe pré-image de n'importe quel point d'intérêt de l'espace de caractéristiques, éliminant ainsi le problème de pré-image. Nous proposons une méthode de génération de graphes pré-images pour obtenir de nouvelles données à partir d'une étiquette quantitative y. En effet, notre approche de régression crée une relation linéaire entre les étiquettes quantitatives et leurs positions dans l'espace de caractéristiques. Comme la distribution gaussienne caractérisée par $(\boldsymbol{\mu}_{\min}, \boldsymbol{\Sigma}_{\min})$ est intentionnellement associée à l'étiquette quantitative minimum dans Y, et que celle définie par $(\boldsymbol{\mu}_{\max}, \boldsymbol{\Sigma}_{\max})$ est associée à l'étiquette quantitative maximum dans Y, nous pouvons déterminer la position de la moyenne $\boldsymbol{\mu}_y$ qui correspond à une étiquette quantitative y en employant l'équation (5.8). Ensuite, à partir de la distribution gaussienne paramétrée par $(\boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y)$, il est possible d'échantillonner un point $\mathbf{z} \in \mathcal{Z}$ avec $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \, \mathbb{I}_D)$ où σ^2 représente la variance des distributions gaussiennes choisies pendant l'apprentissage du modèle et d'obtenir son graphe pré-image avec

$$(\hat{\mathbf{X}}, \hat{\mathbf{A}}) = \Phi^{-1}(\mathbf{z}),$$

puis en convertissant le résultat en graphe $\hat{G} \in \mathcal{G}$. L'Alg. 12 montre une telle procédure de génération.

5.3 Expérimentations

Dans cette section, nous évaluons nos deux approches sur différents ensembles de données de classification et de régression de graphes. L'évaluation de ces approches est abordée en répondant à trois questions distinctes :

Q1 L'espace de caractéristiques généré par notre méthode de classification est-il adapté aux données de graphes structurées et les représentations qui en découlent peuvent-elles être utilisées efficacement à des fins de classification ? (Section 5.3.4)

Q2 L'espace de caractéristiques produit par notre méthode de régression pour les graphes de données peut-il être considéré comme efficace, et les représentations générées conviennent-elles aux objectifs basés sur la régression ? (Section 5.3.5)

Q3 La capacité de génération de nos modèles est-elle pertinente et cohérente ? (Section 5.3.6)

Après avoir présenté les ensembles de données (Section 5.3.1) et les modèles utilisés dans les évaluations (Section 5.3.2), nous commençons par établir le protocole dans la Section 5.3.3, puis nous détaillons les expériences dans les sections suivantes.

5.3.1 Ensembles de données

Nos deux approches travaillant sur différentes tâches, à savoir classification et régression, plusieurs ensembles de données de graphes dédiés à ces différentes tâches ont été utilisés dans les expériences menées. De plus, l'utilisation de plusieurs jeux de données de graphes implique l'évaluation avec plusieurs ensembles de graphes pouvant être définis différemment. En effet, un premier type de graphe, que nous appellerons graphe à nœuds attribués, comporte des nœuds caractérisés par des vecteurs de valeurs quantitatives. Le deuxième type de graphe utilisé dans ces expériences, appelé graphe de nœuds étiquetés, comporte des nœuds caractérisés par une classe de nœuds définie comme un vecteur par un encodage *one-hot*. Les différents jeux de données de graphe utilisés sont introduits par la suite.

Ensembles de données de classification. Les descriptions des ensembles de données de classification de graphes sont énumérées ci-dessous :

MUTAG : Ce jeu de données contient 188 composés chimiques, chacun représenté sous la forme d'un graphe non orienté de nœuds étiquetés. Chaque graphe de ce jeu de données comprend jusqu'à 28 nœuds représentant les atomes, et des arêtes représentant les liaisons chimiques entre les atomes (il y a 3 différents types de liaisons). Les étiquettes sur les nœuds indiquent l'élément atomique de l'atome parmi les 7 différents types présents. Le jeu de données est composé de 2 classes indiquant si le composé chimique correspondant est mutagène ou non mutagène.

Letter-med [104] : Jeu de données composé de 2300 graphes de nœuds attribués représentant l'une des 15 lettres majuscules de la liste suivante : A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z. Chaque graphe contient jusqu'à 9 nœuds où chaque nœud a 2 caractéristiques dimensionnelles représentant sa

position. Les nœuds sont reliés par des arêtes non étiquetées correspondant au tracé des lignes. De plus, les graphes sont déformés avec une force moyenne.

Ensembles de données de régression. Les ensembles de données de graphes utilisés pour la régression sont ensuite présentés ci-dessous :

QM7 [106, 10] : Le jeu de données de chimie quantique QM7 est constitué de molécules organiques de petite taille, comportant jusqu'à 7 atomes lourds (à l'exclusion de l'hydrogène). Il contient 7165 molécules, chacune représentée sous forme de graphe de nœuds étiquetés, où les atomes sont des nœuds et les liaisons chimiques sont des arêtes. Les types d'atomes présents dans le jeu de données sont limités à 4 valeurs : C, N, O et S. La tâche de prédiction utilisée pour nos expériences consiste à prédire l'énergie d'atomisation de chaque molécule.

ESOL [30] : Le jeu de données ESOL original est composé de 1128 molécules d'une taille maximale de 55 atomes, représentés sous forme de graphes de nœuds étiquetés. Les nœuds des graphes correspondent aux atomes, sélectionnés parmi un ensemble de 9 types d'atomes différents, dont C, O, Cl, S, N, I, Br, F et P. Les arêtes des graphes représentent les liaisons chimiques entre les atomes de la molécule. Nos approches utilisant une représentation de graphe sensible à la taille des graphes, nous avons décidé de filtrer les molécules comportant plus de 22 atomes, réduisant ainsi l'ensemble des données à 1015 graphes différents. La tâche de prédiction consiste à prédire la mesure de solubilité associée à chaque molécule.

FREESOLV [89] : Le jeu de données FreeSolv fournit des informations expérimentales et calculées sur les énergies libres d'hydratation de 643 petites molécules dans l'eau représentées par des graphes de nœuds étiquetés. Chaque graphe est composé d'un maximum de 24 nœuds représentant les atomes de la molécule. Il y a 9 différents types d'atomes présents dans les graphes, à savoir C, O, Cl, S, N, I, Br, F et P. Les arêtes des graphes représentent les liaisons chimiques de la molécule. En raison de la sensibilité à la taille des graphes de la représentation graphique utilisée, nous avons choisi d'exclure les molécules comportant plus de 22 atomes, résultant à une réduction du jeu de données à 632 graphes distincts.

5.3.2 Configuration

Dans le présent document, nous évaluons nos approches en utilisant le GraphNF nommé **MoFlow** [136]. Toutefois, comme nos expérimentations ne portent pas uniquement sur des données de graphes moléculaires, nous omettons l'étape de correction moléculaire dans le cas où les données ne représentent pas des molécules.

De plus, nous comparons nos approches aux méthodes de noyaux de graphes. En raison des différents types de graphe, introduit dans la Section 5.3.1, nous utilisons plusieurs types de noyaux de graphes. Ceux-ci sont présentés ci-dessous.

Noyaux à graphe de nœuds étiquetés. Les noyaux suivants ont été utilisés pour les ensembles de données constitués de graphes à nœuds étiquetés :

Weisfeiler-Lehman [117] (WL) : Ce noyau de graphe fonctionne en mettant à jour de manière itérative les étiquettes attribuées à chaque nœud d'un graphe en fonction de la structure de son voisinage local. Une fois les itérations terminées, le noyau calcule la similarité entre deux graphes en comparant la distribution des étiquettes finales attribuées à chaque nœud.

Plus court chemin [13] (SP) : Le noyau du plus court chemin est calculé entre deux graphes, en calculant tout d'abord la distance du plus court chemin entre chaque paire de nœuds dans chaque graphe. Il calcule ensuite une mesure de similarité entre les ensembles de distances de plus court chemin pour les deux graphes.

Code Hadamard [56] (Hadcode) : Ce noyau mesure la similarité entre des paires de graphes en codant les étiquettes des nœuds des graphes à l'aide de codes binaires et en comparant ces codes à l'aide du produit d'Hadamard.

Noyaux à graphe de nœuds attribués. Les noyaux ci-dessous ont été employés pour les ensembles de données contenant des graphes comportant des nœuds pourvus d'attributs quantitatifs :

Propagation [92] : Noyau de graphe reposant sur le concept de propagation des informations d'attributs entre les nœuds d'un graphe, en utilisant la structure sous-jacente du graphe comme base pour le processus de propagation.

Laplacien multi-échelle [66] (MSLap) : Ce noyau fonctionne en calculant la matrice laplacienne du graphe à plusieurs échelles, qui représente la structure de connectivité du graphe. Les matrices de Laplacien sont ensuite utilisées pour calculer les valeurs du noyau entre les paires de graphes.

De plus, comme notre approche consiste à appliquer une régression ridge sur la concaténation des représentations aplaties du graphe dans $\mathcal{Z}$, nous la comparons à des noyaux classiques travaillant sur la concaténation des représentations du graphe $(\mathbf{X}_i, \mathbf{A}_i)$ aplaties résultant sur des vecteurs $\mathbf{x}_i \in \mathbb{R}^D$. De cela, nous pouvons appliquer les méthodes de noyaux définies par :

Linéaire : $\kappa((\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2)) = \mathbf{x}_1^\top \mathbf{x}_2$

RBF : $\kappa((\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2)) = \exp\left(-\gamma || \mathbf{x}_1 - \mathbf{x}_2 |^2\right)$

Polynomial : $\kappa((\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2)) = (\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)^n$

Sigmoïde : $\kappa((\mathbf{X}_1, \mathbf{A}_1), (\mathbf{X}_2, \mathbf{A}_2)) = \tanh(\gamma\, \mathbf{x}_1^\top \mathbf{x}_2 + 1)$

Avec de telles définitions, les modèles prédictifs respectifs sont formés en minimisant la fonction de coût décrite dans l'équation (5.11).

De plus, la capacité de générer des pré-images est comparée à l'approche décrite dans [5], employée sur les méthodes de noyau mentionnées précédemment.

5.3.3 Protocole

Dans cette section, nous décrivons le protocole utilisé pour permettre la reproductibilité des expériences. Nous partitionnons les ensembles de données en allouant 90% des données pour l'entraînement du modèle et les 10% restants à des fins d'évaluation.

Pour garantir une comparaison équitable, les paramètres des noyaux, décrits dans la Section 5.3.2, ont été déterminés par validation croisée avec une recherche en grille où 10 valeurs de λ allant de 10^{-5} à 10^{2} ont été sélectionnées pour l'échantillonnage. De plus, pour les noyaux classiques, une échelle logarithmique a été utilisée pour échantillonner 5 valeurs de γ comprises entre 10^{-5} et 10^{3}. Pour le noyau polynomial, la puissance n a été sélectionnée à partir d'un ensemble de valeurs candidates : $1, 2, 3, 4$. Le score R^2 a été utilisé comme critère de notation pour l'évaluation des performances de régression.

Comme décrit dans les sections précédentes, nous convertissons nos données de graphes en une combinaison d'une matrice de caractéristiques de nœuds et d'un tenseur d'adjacence, à savoir $(\mathbf{X}, \mathbf{A})$ avec $\mathbf{X} \in \mathbb{R}^{n \times d}$ et $\mathbf{A} \in \mathbb{R}^{n \times n \times e}$. Les expériences qui suivent utilisent des arêtes étiquetées dans chaque jeu de données, ce qui conduit à définir e comme le nombre d'étiquettes d'arêtes plus une étiquette pour l'arête inexistante. La valeur de d peut varier en fonction du type du jeu de données de graphe utilisé. Par exemple, dans les graphes comportant des nœuds étiquetés, d est déterminé par le nombre d'étiquettes auquel nous ajoutons une étiquette correspondant au nœud inexistant. En revanche, dans les graphes composés de nœuds attribués, d est défini comme le nombre de caractéristiques. De plus, étant donné qu'un graphe composé de n nœuds peut être représenté de $n!$ façons distinctes avec cette méthode de représentation, nous entraînons nos modèles en mettant en œuvre une transformation de permutation aléatoire des données de graphes utilisées. En outre, pour évaluer la variabilité aux permutations de notre technique, chaque expérience est évaluée 10 fois en employant des permutations aléatoires, et par conséquent, les résultats sont rapportés sous la forme d'une paire de valeurs indiquant la performance moyenne et son écart-type.

5.3.4 Expériences de classification

La question initiale à laquelle nos expériences visent à répondre concerne l'évaluation de l'espace de caractéristiques généré à des fins de classification (**Q1**). Nous partons du principe que la séparation des classes dans l'espace de caractéristiques permet l'utilisation d'un classifieur linéaire simple.

Dans cette expérience, nous évaluons notre approche avec deux types distincts d'ensembles de données, à savoir MUTAG, un jeu de données de graphes à nœuds étiquetés, et Letter-med, un jeu de données de graphes composé de nœuds attribués. Par conséquent, les noyaux de graphes utilisés pour l'analyse comparative, tels que décrits dans la Section 5.3.2, sont différents, et les performances sont présentées dans la Table 5.1. Nos résultats dans les deux cas sont supérieurs à ceux obtenus par les autres méthodes, ce qui met en évidence la pertinence de notre approche.

De plus, nous évaluons les performances de notre modèle en augmentant les dimensions de l'espace de caractéristiques selon l'extension expliquée dans la Section 5.2.1. La dernière ligne de Table 5.1 montre les résultats obtenus sur le même jeu de données en faisant varier le nombre de dimensions des caractéristiques des nœuds par l'ajout de 10 dimensions, c'est-à-dire en posant $p = 10$. Ces résultats démontrent que l'extension de notre méthode a un impact favorable sur le jeu de données Letter-med où la performance de classification s'améliore au fur et à me-

TABLE 5.1 – Taux de bonne classification (± écart type) sur les jeux de données de graphe de nœuds étiquetés (MUTAG) ou à attributs numériques (Letter-med).

Méthodes		Ensembles de données	
		MUTAG	Letter-med
Noyaux classiques	Linéaire	0.761 ± 0.070	0.414 ± 0.022
	RBF	0.772 ± 0.039	0.419 ± 0.022
	Polynomial	0.717 ± 0.046	0.427 ± 0.028
	Sigmoïde	0.683 ± 0.056	0.281 ± 0.024
Noyaux de graphes	WL	0.778 ± 0.0	-
	SP	0.778 ± 0.0	-
	Hadcode	0.778 ± 0.0	-
	Propagation	-	0.298 ± 0.135
	MSLap	-	0.410 ± 0.039
Notre approche	MoFlow	$\mathbf{0.939} \pm 0.016$	0.752 ± 0.012
Extension ($p = 10$)	MoFlow	$\mathbf{0.944} \pm 0.0$	$\mathbf{0.867} \pm 0.012$

sure que les dimensions des données augmentent. Cependant, les résultats faiblement améliorés obtenus sur MUTAG suggèrent que l'augmentation du nombre de dimensions n'est pas utile dans toutes les situations. En effet, nous supposons que la taille des données d'observation dans MUTAG est déjà suffisamment importante pour différencier seulement 2 classes, de sorte qu'il n'est pas nécessaire d'ajouter des dimensions supplémentaires. En revanche, le jeu de données Letter-med contient de petits graphes avec un nombre limité de caractéristiques de nœuds qui doivent être classés en 15 catégories distinctes, ainsi l'augmentation de l'espace de caractéristiques permet l'amélioration des performances.

5.3.5 Expériences de régression

Afin de répondre à **Q2**, la Table 5.2 montre, pour chaque méthode, la performance moyenne et l'écart-type obtenus sur des ensembles de données de régression. Ces résultats montrent la bonne performance prédictive de notre méthode lors de l'utilisation de la régression linéaire dans l'espace de caractéristiques $\mathcal{Z}$ pour la plupart des ensembles de données. Cependant, les meilleurs résultats sur le jeu de données FREESOLV sont obtenus par les noyaux de Weisfeiler-Lehman et du code Hadamard. Pour comprendre ce point, il convient de noter que ce jeu de données comprend un plus petit nombre de graphes distincts, qui plus est sont relativement plus grands par rapport à d'autres ensembles de données tels que QM7. De plus, les valeurs quantitatives du jeu de données FREESOLV ne sont pas distribuées de manière uniforme, puisqu'elles s'échelonnent entre $-5,48$ et $1,79$, avec une proportion importante (plus de 97%) se situant dans la fourchette de $-2,57$ à $1,79$. Par conséquent, l'approche basée sur l'interpolation linéaire utilisée dans cette étude n'est peut-être pas la méthode la plus appropriée pour de tels ensembles de données. Cependant, notre approche présente l'avantage, par rapport aux noyaux de graphes,

TABLE 5.2 – Score R^2 (± écart type) sur des ensembles de données de graphes pour la régression

Méthodes		Ensembles de données		
		QM7	ESOL	FREESOLV
Noyaux classiques	Linéaire	0.681 ± 0.001	0.555 ± 0.032	0.254 ± 0.114
	RBF	0.680 ± 0.002	0.558 ± 0.032	0.262 ± 0.113
	Polynomial	0.681 ± 0.001	0.566 ± 0.034	0.264 ± 0.102
	Sigmoïde	0.673 ± 0.002	0.563 ± 0.037	0.255 ± 0.074
Noyaux de graphes	WL	0.490 ± 0.0	0.602 ± 0.0	**0.895** ± 0.0
	SP	0.721 ± 0.0	0.531 ± 0.0	0.543 ± 0.0
	Hadcode	0.491 ± 0.0	0.573 ± 0.0	**0.901** ± 0.0
Notre approche	MoFlow	**0.730** ± 0.008	**0.685** ± 0.039	0.754 ± 0.042

qu'une fois le modèle appris, les prédictions peuvent être faites directement et simplement, alors que les méthodes de noyaux de graphes peuvent être plus coûteuses en termes de calcul.

En outre, nous pouvons observer dans les résultats un écart-type non nul dans nos performances en raison de la nature des représentations de graphe utilisées, contrairement aux noyaux de graphes qui sont conçus pour être invariants aux permutations. Par conséquent, nous pouvons répondre positivement à la question **Q2** dans la plupart des cas, indiquant que la transformation non-linéaire Φ apprise par notre méthode est capable de produire un bon espace de représentation pour une tâche de régression.

5.3.6 Qualité de pré-image

Pour répondre à **Q3** et tester la capacité de notre modèle à générer des pré-images, nous avons généré des graphes moléculaires à partir de point d'échantillonnage dans $\mathcal{Z}$ en utilisant un modèle entraîné pour chacune des deux tâches.

Pré-image de classification. La génération de pré-image a été évaluée avec notre modèle entraîné sur MUTAG. Pour cela, nous avons échantillonné aléatoirement 12 points dans $\mathcal{Z}$ pour chaque classe, à savoir en appliquant $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$ avec $c \in \{1 \dots C\}$. Les pré-images ont été générées dans l'espace des observations en appliquant la transformation inverse Φ^{-1} aux $\mathbf{z}$ générés. Celles-ci sont présentées dans la Figure 5.5, ainsi que les valeurs de la classe utilisée y et de la classe prédite $f(\mathbf{X}, \mathbf{A})$ à l'aide de notre modèle de prédiction.

Pré-image de régression. L'évaluation des pré-images dans un contexte de régression a été effectuée en utilisant notre modèle entraîné sur le jeu de données QM7. Ces points ont été échantillonnés en interpolant 24 valeurs de y entre $\min(Y)$ et $\max(Y)$ comme suit : Pour chaque valeur, nous avons échantillonné un point dans $\mathcal{Z}$ avec $z \sim \mathcal{N}(\boldsymbol{\mu}_y, \sigma^2 \mathbb{I}_D)$. Les pré-images correspondantes ont été générées

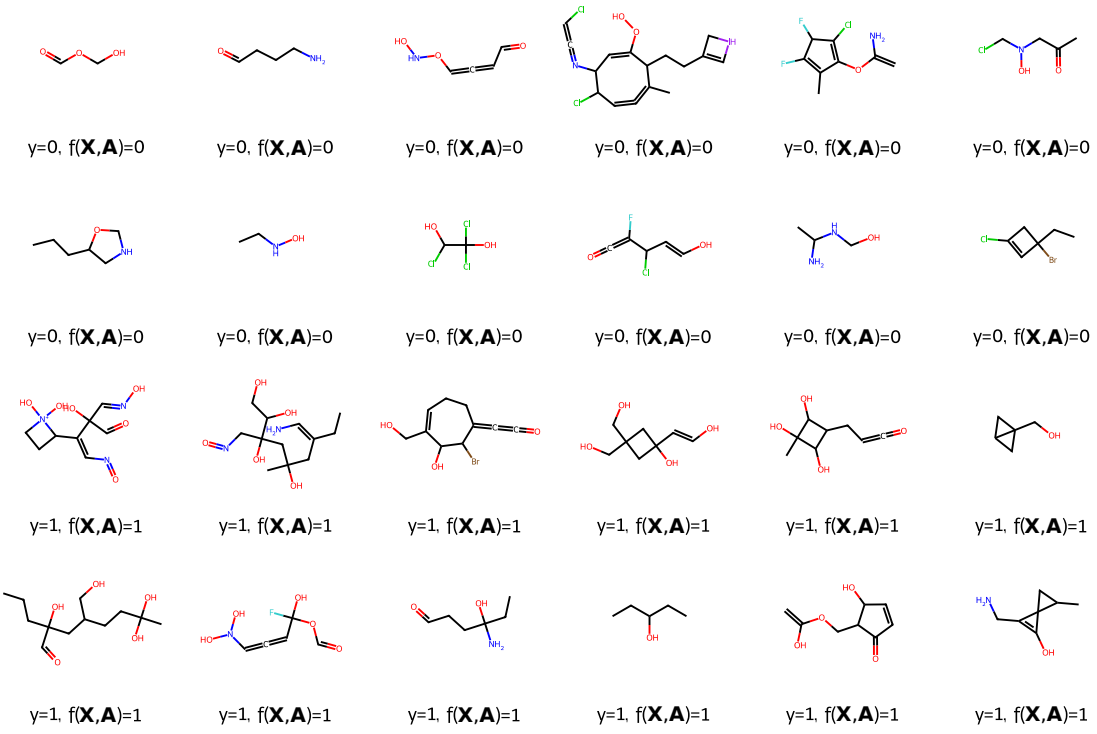


FIGURE 5.5 – Pré-images générées à partir de 24 points échantillonnés dans $\mathcal{Z}$ selon chaque distribution gaussienne associée à chaque classe et obtenues par la transformation inverse Φ^{-1} de notre modèle entraîné sur le jeu de données MUTAG.

dans l'espace des entrées en appliquant la transformation inverse Φ^{-1} aux $\mathbf{z}$ générés. Figure 5.6 montre les pré-images obtenues, ainsi que les valeurs quantitatives échantillonnées y et la valeur prédite $f(\mathbf{X}, \mathbf{A})$ en utilisant notre modèle de prédiction appris.

Ces expériences démontrent que nos modèles peuvent générer des pré-images cohérentes de points dans $\mathcal{Z}$ qui ne font pas partie du jeu de données, répondant ainsi positivement à **Q3**.

5.4 Conclusion

Les expérimentations menées dans ce chapitre montrent une adaptation de nos approches sur des données de graphes fonctionnelle résultant à de nouvelles approches sur graphe surmontant la malédiction de pré-image grâce aux GraphNF. Nos méthodes permettent la génération supervisée d'un espace dans lequel la classification linéaire ainsi que la régression linéaire peuvent être effectuées efficacement, comme le démontrent les expériences menées. De plus, nos méthodes permettent une meilleure interprétabilité grâce à l'obtention d'une transformation inverse rendant possible le passage de l'espace de caractéristiques vers l'espace des observations et donc la génération de pré-image de points d'intérêt.

Nos approches contribuent à l'application des GraphNF dans une tâche super-

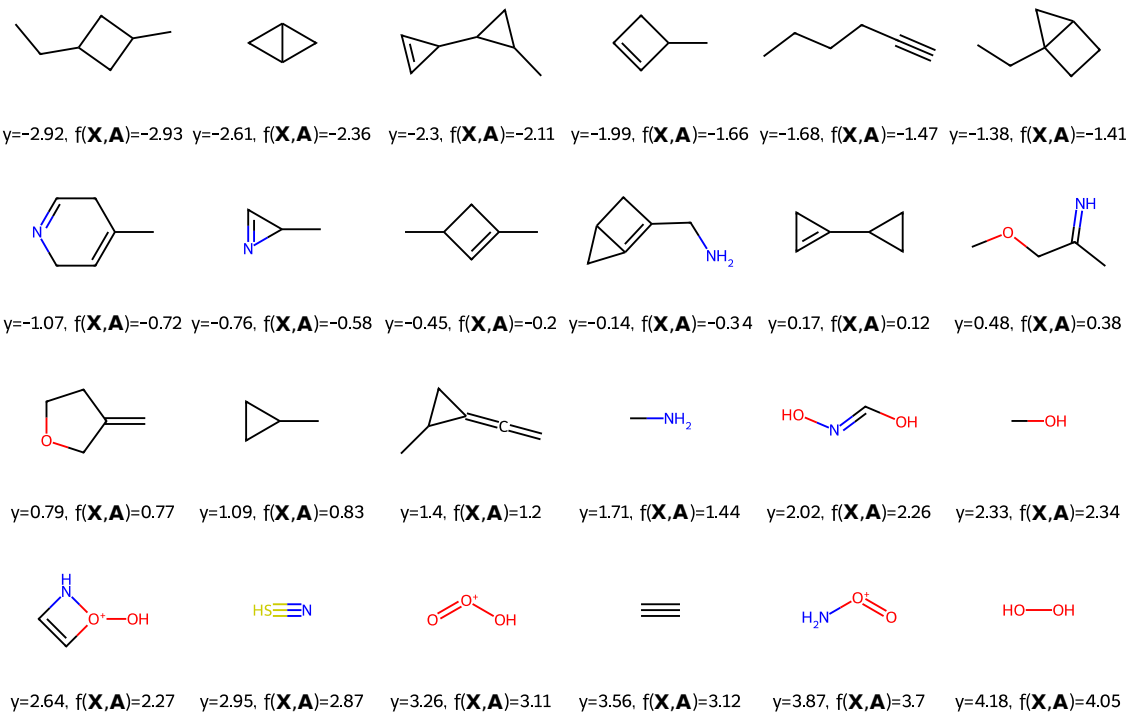


FIGURE 5.6 – Pré-images générées à partir de 24 points échantillonnés uniformément dans $\mathcal{Z}$ entre $\min(Y)$ et $\max(Y)$ et obtenues par la transformation inverse Φ^{-1} de notre modèle entraîné sur le jeu de données QM7.

visée spécifique et a le potentiel d'être adaptée à d'autres tâches. Bien que nos méthodes soient sensibles à la permutation du graphe en raison de la représentation utilisée, il pourrait être intéressant de l'étendre à d'autres types de représentations de graphe, permettant ainsi d'obtenir une possible invariance aux permutations.

CHAPITRE 6

Conclusion

Pour conclure cette thèse, nous commençons par un récapitulatif de chacune de nos contributions, tout en mettant en évidence leurs avantages et limitations dans la Section 6.1. Ensuite, nous abordons les perspectives potentielles de travaux dans la Section 6.2.

6.1 Résumé des contributions

Ce document de thèse a été rédigé afin de proposer des solutions au problème de pré-image couramment rencontré en ML. Pour cela, nous avons proposé plusieurs approches qui reposent sur l'utilisation d'un modèle génératif profond de type NF. Lorsque la première approche proposée cherche à résoudre le problème de pré-image qui apparaît spécifiquement lors de l'utilisation de méthode à noyaux, les suivantes proposent la définition de nouvelles approches de ML insensibles au problème de pré-image et adaptées à différentes tâches. Dans cette section, nous présentons les résumés de chaque contribution introduite dans les chapitres de 2 à 5.

Notre première proposition, présentée dans le Chapitre 2, a pour but d'introduire une solution au problème de pré-image rencontré par les méthodes à noyaux. Basée sur les avancées des méthodes génératives de réseaux de neurones, nous avons proposé une approche visant à transposer le problème de pré-image rencontré par les noyaux vers un espace généré par NF. Étant donné que la seule information disponible sur la structure de l'espace du noyau est représentée par les valeurs des produits scalaires entre les données, nous avons introduit la notion d'alignement entre les espaces par leur équivalence en termes de produits scalaires inter-données. Pour accomplir cet alignement, nous avons proposé la génération d'un espace aligné à celui implicitement généré par le noyau. Pour ce faire, nous utilisons un modèle de réseau de neurones appelé modèle d'alignement. Ce modèle opère sur l'espace du NF en transformant les représentations échantillonnées de cet espace vers un nouvel espace où les relations inter-données sont équivalentes à celles présentes dans l'espace du noyau. Ainsi, un point d'intérêt de l'espace du noyau est transposé dans l'espace du NF par le résultat de l'optimisation d'un échantillon de l'espace du NF

dans la direction indiquée par le modèle d'alignement. Grâce à l'inversibilité des NF, la génération d'une donnée pré-image peut être réalisée par l'application de la transformation inverse sur un échantillon dans l'espace du NF. Pour évaluer notre approche, nous avons utilisé un noyau gaussien et testé sur trois architectures de NF différentes. Dans nos diverses expériences, nous avons constaté que certains NF génèrent un espace plus ou moins structuré de la même manière que celui du noyau, comme dans le cas d'OT-Flow générant des pré-images proches de celles correspondant au noyau. Cependant, l'approche d'optimisation proposée ne permet qu'une légère amélioration des pré-images générées. Nos expériences démontrent que, dès lors que l'espace généré par le NF diffère de l'espace implicitement généré par le noyau, notre approche d'alignement proposée ne permet pas la correction de cette différence. Nous pouvons ajouter que l'utilisation d'un noyau gaussien simple permet une similarité entre l'espace du noyau et celui de OT-Flow. Cependant, l'utilisation de noyaux plus complexes ne permettront pas aux NF d'obtenir de bons résultats de pré-image. De ces observations, nous pouvons envisager la modification de l'approche par l'orientation dès l'apprentissage du NF vers l'espace du noyau.

La deuxième contribution, présentée dans le Chapitre 3, permet la définition d'un formalisme d'apprentissage machine non affecté par le problème de pré-image basé sur l'utilisation d'un modèle génératif de type NF. Celui-ci permet un apprentissage non supervisé d'une transformation non-linéaire vers un espace de représentation paramétrable où des opérations ou méthodes de ML linéaires peuvent être appliquées. De plus, l'utilisation d'un NF permet à la transformation vers cet espace d'être inversible, ce qui résout le problème de pré-image. Les expériences menées sur cette approche montrent que notre formalisme peut être utilisé pour des tâches de débruitage, compression et génération de nouvelles données par l'application de méthodes de ML simples dans l'espace généré. Ces résultats démontrent que ce formalisme permet une première approche dans l'application de NF pour l'apprentissage de transformation non-linéaire vers un espace de représentation sans problème de pré-image. Il convient de souligner que l'obtention de bonnes performances de débruitage et de compression par l'utilisation de notre formalisme, tel qu'introduit dans le Chapitre 3, dépend de la variance du jeu de données. En effet, bien que l'apprentissage soit non supervisé, il peut être intéressant d'avoir des connaissances a priori sur le jeu de données utilisé afin de paramétrer la distribution de l'espace de représentation de manière appropriée. En d'autres termes, l'utilisation d'un faible nombre de composantes principales pour le paramétrage de la distribution des données dans l'espace de représentation aboutit à une bonne qualité de débruitage et de compression que si la variance de l'ensemble de données est également faible. Sur la base de cette observation, et grâce à la grande flexibilité offerte par notre formalisme, nous avons proposé deux spécifications d'apprentissage supervisé qui permettent la prise en compte d'informations prédictives supplémentaires issues des données.

Le Chapitre 4 présente deux spécifications de notre formalisme pour les tâches prédictives de type classification et régression par la supervision de l'apprentissage. Ainsi, chacune de ces méthodes permet la génération d'un espace de représentation où les données sont, en fonction de la tâche, soit linéairement séparables, soit linéairement organisées. Des prédictions de bonne qualité ont été obtenues en appliquant des opérations linéaires simples. Ajouté à cela, étant basées sur notre formalisme du

Chapitre 3, nos approches sont insensibles au problème de pré-image grâce à l'utilisation d'un NF. Les expériences réalisées ont démontré de bonnes performances pour les tâches associées par l'application d'opérations linéaires simples, tout en permettant la génération de pré-images de points d'intérêt. De plus, notre approche de classification a permis une extension directe de notre formalisme non supervisé en introduisant les étiquettes de classe associées aux données. Ainsi, les tâches de débruitage et de compression ont été réalisées à partir de notre approche de classification, tout en offrant de très bonnes performances. Enfin, il est important de préciser que nos expériences ont été menées sur des données de faible dimension. Cependant, le passage à l'échelle de nos approches dépend du modèle de NF utilisé. En effet, l'utilisation de méthodes de ML simples et linéaires dans l'espace de représentation permet à nos approches de ne connaître de limitations qu'en lien avec le modèle de NF utilisé. Par conséquent, nos approches peuvent être adaptées à des problématiques spécifiques par l'utilisation d'un NF approprié.

Enfin, le dernier chapitre (5) a proposé l'extension de nos deux spécifications du Chapitre 4, travaillant sur des données vectorielles et d'images, à des données de graphes. Les expérimentations menées sur divers jeux de données ont montré de bons résultats sur les tâches de classification ou de régression associées. Ainsi, par l'utilisation d'un GraphNF, ces approches ont permis la définition de nouvelles méthodes travaillant sur des données de type graphe sans problème de pré-image. De la même manière que précédemment, un espace de représentation a été généré dans lequel des méthodes de ML linéaires simples ont été appliquées efficacement sur des données de type graphe. À cela s'ajoute la génération de pré-images de tout point d'intérêt dans l'espace de représentation, rendue possible par la structure inversible de NF. Il est important de noter que les expériences ont été réalisées en utilisant un GraphNF spécifique fonctionnant sur une représentation de graphe particulière. Ce dernier étant sensible aux permutations de graphes, notre approche présente également une sensibilité aux permutations de graphes, ce qui signifie que chaque permutation entraîne une représentation différente. De même, nos expériences ont été réalisées sur des graphes de petite taille, étant donné que le GraphNF utilisé est conçu pour traiter ce type de graphe. Néanmoins, nos approches offrent la possibilité d'utiliser n'importe quel modèle de NF visant à générer un espace de représentation. Par conséquent, l'évolution de nos approches peut être réalisée en utilisant différents modèles de NF adaptés à la problématique donnée.

6.2 Perspectives des travaux

Durant ces travaux de thèse, nous avons exploré de nombreuses pistes pour définir une solution au problème de pré-image par l'application de modèles génératifs profonds. Ce domaine étant en pleine expansion, nous avons pu examiner de nombreuses approches et leurs applications en vue d'une solution possible au problème de pré-image. Grâce à cela, nous sommes parvenus à proposer un formalisme général et adaptable permettant la définition d'approches de ML sans problème de pré-image. Du fait de la flexibilité de notre approche, nous avons pu proposer deux méthodologies différentes pour des tâches de classification et de régression. Cependant, il est tout à fait possible de proposer différentes spécifications du formalisme

pour différentes tâches. Ainsi, ces travaux offrent plusieurs angles d'amélioration possibles.

Un premier axe d'amélioration peut être dans la formulation même de l'apprentissage du NF. Le Chapitre 3 propose une approche générique d'apprentissage non supervisé dans laquelle l'objectif d'optimisation du modèle reste inchangé, excepté par la généralisation de la distribution gaussienne cible. Ce formalisme offre la possibilité de revisiter les NF en intégrant éventuellement des contraintes supplémentaires dans le processus d'apprentissage. Ces contraintes supplémentaires peuvent par exemple inclure des contraintes d'orthogonalité ou de voisinage. En incorporant ces contraintes, il est possible d'enrichir et d'adapter les modèles de NF pour mieux répondre aux exigences spécifiques de diverses tâches d'apprentissage non supervisé. De plus, dans cette approche, nous avons proposé l'application de l'ACP à des fins de débruitage de données. Cependant, il est tout à fait possible d'utiliser d'autres variétés telles que l'analyse en composantes indépendantes [22], la séparation aveugle de signaux [15], ou la factorisation par matrices non-négatives [72].

Les méthodologies proposées dans les chapitres 4 et 5 sont potentiellement améliorables en définissant autrement la distribution de représentation des données cibles par une modification de la perte utilisée lors de l'apprentissage du NF. Par exemple, nous pouvons constater que dans certaines expériences, notamment sur les données de graphes moléculaires de FREESOLV, la définition de distribution gaussienne suivant une interpolation linéaire n'est pas la plus adéquate. Une amélioration possible consisterait à adapter cette approche d'interpolation de manière à prendre en compte la distribution des valeurs d'étiquettes présentes dans le jeu de données. De plus, cette approche de régression permet une organisation des données suivant une valeur quantitative souhaitée permettant la génération de données à partir d'une valeur souhaitée. Cependant, il semble simple et pertinent d'étendre cette approche non pas à une valeur d'étiquette, mais également à un vecteur de valeurs pour des tâches de régression.

Un autre axe d'amélioration, rendu possible par la flexibilité de notre approche, concerne le choix du NF. En effet, nos expériences du Chapitre 5 ont été réalisées sur des données de type graphe avec un GraphNF performant, mais sensible aux permutations ainsi qu'à la taille des données. Toutefois, il serait pertinent d'accroître le nombre d'expériences en utilisant différents types de GraphNF fonctionnant de différentes manières. En effet, lorsque nous utilisons un GraphNF travaillant sur une représentation matricielle des données dans nos expériences, d'autres modèles de GraphNF peuvent travailler sur différentes représentations de graphes. Nous pouvons prendre l'exemple de GraphAF [118], un GraphNF auto-régressif modélisant les graphes par une séquence de décision, permettant ainsi une extension à des graphes de plus grande taille. De plus, nous pouvons imaginer que l'application de la fonction de transformation bijective d'un GraphNF travaillant sur une unique représentation de graphe observé permettrait également une unique représentation du graphe dans l'espace de caractéristiques. Ceci permettrait de contourner le problème de variance des permutations de graphes rencontré avec les représentations matricielles des graphes. De la même manière, il existe d'autres NF pouvant être utilisées dans la définition de nos approches des chapitres 3 et 4, pouvant permettre une meilleure extensibilité aux données vectorielles et d'images de grande dimension.

Bibliographie

[1] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *International conference on machine learning*, pages 159–168. PMLR, 2018.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3) :337–404, 1950.

[4] James Atwood and Don Towsley. Diffusion-convolutional neural networks. *Advances in neural information processing systems*, 29, 2016.

[5] Gökhan H Bakır, Jason Weston, and Bernhard Schölkopf. Learning to find pre-images. *Advances in neural information processing systems*, 16 :449–456, 2004.

[6] Gökhan H Bakır, Alexander Zien, and Koji Tsuda. Learning to find graph pre-images. In *Pattern Recognition : 26th DAGM Symposium, Tübingen, Germany, August 30-September 1, 2004. Proceedings 26*, pages 253–261. Springer, 2004.

[7] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[8] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828, 2013.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2) :157–166, 1994.

[10] Lorenz C Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25) :8732–8733, 2009.

[11] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan : Generating graphs via random walks. In *International conference on machine learning*, pages 610–619. PMLR, 2018.

[12] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G Willcocks. Deep generative modelling : A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[13] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Fifth IEEE international conference on data mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[14] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[15] J-F Cardoso. Blind signal separation : statistical principles. *Proceedings of the IEEE*, 86(10) :2009–2025, 1998.

[16] Matteo Cassotti, Davide Ballabio, Viviana Consonni, Andrea Mauri, Igor V Tetko, and Roberto Todeschini. Prediction of acute aquatic toxicity toward daphnia magna by using the ga-k nn method. *Alternatives to Laboratory Animals*, 42(1) :31–41, 2014.

[17] Matteo Cassotti, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. A similarity-based qsar model for predicting acute toxicity towards the fathead minnow (pimephales promelas). *SAR and QSAR in Environmental Research*, 26(3) :217–243, 2015.

[18] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning : a survey. *APSIPA Transactions on Signal and Information Processing*, 9 :e15, 2020.

[19] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[20] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation : Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103, 2014.

[21] Earl A Coddington, Norman Levinson, and T Teichmann. Theory of ordinary differential ėquations. *Physics Today*, 9(2) :18, 1956.

[22] Pierre Comon. Independent component analysis, a new concept ? *Signal processing*, 36(3) :287–314, 1994.

[23] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[24] Bin Dai and David Wipf. Diagnosing and enhancing vae models. In *International Conference on Learning Representations*, 2019.

[25] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR, 2018.

[26] Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*, 2018.

[27] Nicola De Cao and Thomas Kipf. Molgan : An implicit generative model for small molecular graphs. *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[28] Paulo de Oliveira Rente, Catarina Brites, Joao Ascenso, and Fernando Pereira. Graph-based static 3d point clouds geometry coding. *IEEE Transactions on Multimedia*, 21(2) :284–299, 2018.

[29] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[30] John S Delaney. Esol : estimating aqueous solubility directly from molecular structure. *Journal of chemical information and computer sciences*, 44(3) :1000–1005, 2004.

[31] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6) :141–142, 2012.

[32] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice : Non-linear independent components estimation. *arXiv preprint arXiv :1410.8516*, 2014.

[33] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[34] Jicong Fan and Tommy W.S. Chow. Non-linear matrix completion. *Pattern Recognition*, 77 :378–394, 2018.

[35] Chris Finlay, Jörn-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ode : the world of jacobian and kinetic regularization. In *International conference on machine learning*, pages 3154–3164. PMLR, 2020.

[36] Claudio Gallicchio and Alessio Micheli. Graph echo state networks. In *The 2010 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2010.

[37] Victor Garcia Satorras, Emiel Hoogeboom, Fabian Fuchs, Ingmar Posner, and Max Welling. E (n) equivariant normalizing flows. *Advances in Neural Information Processing Systems*, 34 :4181–4192, 2021.

[38] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[39] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2) :268–276, 2018.

[40] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[41] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

[42] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv :1308.0850*, 2013.

[43] Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 34(8) :3549–3568, 2020.

[44] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[45] GM Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. A comprehensive survey and analysis of generative models in machine learning. *Computer Science Review*, 38 :100285, 2020.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8) :1735–1780, 1997.

[48] Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. Graph residual flow for molecular graph generation. *arXiv preprint arXiv :1909.13521*, 2019.

[49] Paul Honeine. An eigenanalysis of data centering in machine learning. Technical report, ArXiv, March 2016.

[50] Paul Honeine and Cédric Richard. Solving the pre-image problem in kernel machines : A direct method. In *2009 IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–6. IEEE, 2009.

[51] Paul Honeine and Cédric Richard. Preimage problem in kernel-based machine learning. *IEEE Signal Processing Magazine*, 28(2) :77 – 88, March 2011.

[52] Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3) :1059–1076, 1989.

[53] Linlin Jia, Benoit Gaüzère, and Paul Honeine. A graph pre-image method based on graph edit distances. In *Proceedings of IAPR Joint International Workshops on Statistical techniques in Pattern Recognition (SPR 2020) and Structural and Syntactic Pattern Recognition (SSPR 2020).*, 2021.

[54] Linlin Jia, Benoit Gaüzère, and Paul Honeine. Graph kernels based on linear patterns : theoretical and experimental comparisons. *Expert Systems With Applications*, 189 :116095, March 2022.

[55] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[56] Tetsuya Kataoka and Akihiro Inokuchi. Hadamard code graph kernels for classifying graphs. In *ICPRAM*, pages 24–32, 2016.

[57] Diederik P. Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[58] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114*, 2013.

[59] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4) :307–392, 2019.

[60] Durk P Kingma and Prafulla Dhariwal. Glow : Generative flow with invertible 1x1 convolutions. In *Advances in neural information processing systems*, pages 10215–10224, 2018.

[61] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[62] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*, 2016.

[63] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[64] Ivan Kobyzev, Simon JD Prince, and Marcus A Brubaker. Normalizing flows : An introduction and review of current methods. *IEEE transactions on pattern analysis and machine intelligence*, 43(11) :3964–3979, 2020.

[65] Ryoichi Koga, Noriaki Hashimoto, Tatsuya Yokota, Masato Nakaguro, Kei Kohno, Shigeo Nakamura, Ichiro Takeuchi, and Hidekata Hontani. Stain transfer for automatic annotation of malignant lymphoma regions in h&e stained whole slide histopathology images. In *International Forum on Medical Imaging in Asia 2021*, volume 11792, page 117920R. International Society for Optics and Photonics, 2021.

[66] Risi Kondor and Horace Pan. The multiscale laplacian graph kernel. *Advances in neural information processing systems*, 29, 2016.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6) :84–90, 2017.

[68] William H Kruskal and W Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260) :583–621, 1952.

[69] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv :1703.01925*, 2017.

[70] Maksim Kuznetsov and Daniil Polykovskiy. Molgrow : A graph normalizing flow for hierarchical molecular generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8226–8234, 2021.

[71] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10) :1995, 1995.

[72] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755) :788–791, 1999.

[73] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. Cayleynets : Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1) :97–109, 2018.

[74] Mu Li, Tong Zhang, Yuqiang Chen, and Alexander J Smola. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–670, 2014.

[75] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[76] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv :1803.03324*, 2018.

[77] Yujia Li, Richard Zemel, Marc Brockschmidt, and Daniel Tarlow. Gated graph sequence neural networks. In *Proceedings of ICLR'16*, 2016.

[78] Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

[79] Zhiyuan Liu, Yankai Lin, and Maosong Sun. *Representation learning for natural language processing*. Springer Nature, 2020.

[80] Yu-Chen Lo, Stefano E Rensi, Wen Torng, and Russ B Altman. Machine learning in chemoinformatics and drug discovery. *Drug discovery today*, 23(8) :1538–1546, 2018.

[81] Tengfei Ma, Jie Chen, and Cao Xiao. Constrained generation of semantically valid graphs via regularizing variational autoencoders. *Advances in Neural Information Processing Systems*, 31, 2018.

[82] Yao Ma and Jiliang Tang. *Deep learning on graphs*. Cambridge University Press, 2021.

[83] Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp : An invertible flow model for generating molecular graphs. *arXiv preprint arXiv :1905.11600*, 2019.

[84] Peter McCullagh. *Generalized linear models*. Routledge, 2019.

[85] Soroush Mehri, Kundan Kumar, Ishaan Gulrajani, Rithesh Kumar, Shubham Jain, Jose Sotelo, Aaron Courville, and Yoshua Bengio. Samplernn : An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.

[86] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal*

society of London. Series A, containing papers of a mathematical or physical character, 209(441-458) :415–446, 1909.

[87] Alessio Micheli. Neural network for graphs : A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3) :498–511, 2009.

[88] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

[89] David L Mobley and J Peter Guthrie. Freesolv : a database of experimental and calculated hydration free energies, with input files. *Journal of computer-aided molecular design*, 28 :711–720, 2014.

[90] Kevin P Murphy. *Machine learning : a probabilistic perspective*. MIT press, 2012.

[91] Vivek Narayanaswamy, Jayaraman J Thiagarajan, and Andreas Spanias. Using deep image priors to generate counterfactual explanations. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2770–2774, 2021.

[92] Marion Neumann, Roman Garnett, Christian Bauckhage, and Kristian Kersting. Propagation kernels : efficient graph kernels from propagated information. *Machine Learning*, 102 :209–245, 2016.

[93] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023. PMLR, 2016.

[94] Derek Onken, S Wu Fung, Xingjian Li, and Lars Ruthotto. Ot-flow : Fast and accurate continuous normalizing flows via optimal transport. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

[95] Achraf Oussidi and Azeddine Elhassouny. Deep generative models : Survey. In *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, pages 1–8. IEEE, 2018.

[96] Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2609–2615, 2018.

[97] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans) : A survey. *IEEE Access*, 7 :36322–36333, 2019.

[98] George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57) :1–64, 2021.

[99] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[100] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss,

Vincent Dubourg, et al. Scikit-learn : Machine learning in python. *the Journal of machine Learning research*, 12 :2825–2830, 2011.

[101] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

[102] Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. Molecularrnn : Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv :1905.13372*, 2019.

[103] Arjun Naresh Punjabi. *Multimodal Data Fusion and Feature Visualization in Convolutional Neural Networks*. PhD thesis, Northwestern University, 2020.

[104] Kaspar Riesen, Horst Bunke, et al. Iam graph database repository for graph based pattern recognition and machine learning. In *SSPR/SPR*, volume 5342, pages 287–297, 2008.

[105] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088) :533–536, 1986.

[106] Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O Anatole Von Lilienfeld. Fast and accurate modeling of molecular atomization energies with machine learning. *Physical review letters*, 108(5) :058301, 2012.

[107] Diego Salazar, Juan Rios, Sara Aceros, Oscar Flórez-Vargas, and Carlos Valencia. Kernel joint non-negative matrix factorization for genomic data. *IEEE Access*, 9 :101863–101875, 2021.

[108] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning : Generative models for matter engineering. *Science*, 361(6400) :360–365, 2018.

[109] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7) :1644–1656, 2013.

[110] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[111] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1) :61–80, 2008.

[112] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web : 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, Proceedings 15*, pages 593–607. Springer, 2018.

[113] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Computational Learning Theory : 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001 Amsterdam, The Netherlands, July 16–19, 2001 Proceedings 14*, pages 416–426. Springer, 2001.

[114] Bernhard Schölkopf, Alexander J Smola, Francis Bach, et al. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[115] John Scott. Social network analysis : developments, advances, and prospects. *Social network analysis and mining*, 1 :21–26, 2011.

[116] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[117] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[118] Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. Graphaf : a flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

[119] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs : Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3) :83–98, 2013.

[120] Martin Simonovsky and Nikos Komodakis. Graphvae : Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018 : 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer, 2018.

[121] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[122] Endre Süli. Numerical solution of ordinary differential equations. *Mathematical Institute, University of Oxford*, 2010.

[123] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[124] Jayaraman J Thiagarajan, Vivek Narayanaswamy, Deepta Rajan, Jason Liang, Akshay Chaudhari, and Andreas Spanias. Designing counterfactual generators on-the-fly. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2021.

[125] Thao Tran Thi Phuong, Ahlame Douzal, Saeed Varasteh Yazdi, Paul Honeine, and Patrick Gallinari. Interpretable time series kernel analytics by pre-image estimation. *Artificial Intelligence*, 286 :103342, September 2020.

[126] Nenad Trinajstic. *Chemical graph theory*. Routledge, 2018.

[127] Michael Unser. A representer theorem for deep neural networks. *J. Mach. Learn. Res.*, 20(110) :1–30, 2019.

[128] Michael Unser. A unifying representer theorem for inverse problems and machine learning. *Foundations of Computational Mathematics*, 21(4) :941–960, 2021.

[129] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[130] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[131] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet : A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016.

[132] Zhengwei Wang, Qi She, and Tomas E Ward. Generative adversarial networks in computer vision : A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 54(2) :1–38, 2021.

[133] Hai-Cheng Yi, Zhu-Hong You, De-Shuang Huang, and Chee Keong Kwoh. Graph representation learning in bioinformatics : trends, methods and applications. *Briefings in Bioinformatics*, 23(1) :bbab340, 2022.

[134] Jiaxuan You, Bowen Liu, Zhitao Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. In *Advances in neural information processing systems*, pages 6410–6421, 2018.

[135] Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn : Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[136] Chengxi Zang and Fei Wang. Moflow : an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 617–626, 2020.

[137] Fei Zhu and Paul Honeine. Online kernel nonnegative matrix factorization. *Signal Processing*, 131 :143 – 153, February 2017.