CentraleSupélec

Pôle Projet Data Science - S7
Régression par Processus Gaussiens avec un grand nombre d'observations (scalable GPs)

Auteur :
Thierry ZHANG

Référent :
Julien BECT

4 février 2025

Résumé

Ce projet explore les méthodes scalables de régression par processus gaussiens (GP) afin de traiter de grands ensembles de données. La régression GP est un modèle bayésien non paramétrique largement utilisé en apprentissage automatique pour fournir des prédictions accompagnées de mesures d'incertitude. Cependant, son coût computationnel élevé en $\mathcal{O}(n^3)$ limite son application à des jeux de données massifs. Nous avons étudié plusieurs approches d'approximation telles que les processus gaussiens épars (Sparse GP), l'inférence variationnelle stochastique (SVI) et les processus gaussiens locaux approximés (laGP). Une évaluation expérimentale a été réalisée afin de comparer la précision, la scalabilité et la complexité computationnelle de ces méthodes. Les résultats obtenus permettent de proposer des recommandations pour le choix de la méthode en fonction des contraintes spécifiques d'un projet.

Table des matières

Chapitre 1

Introduction

1.1 Contexte

La régression par processus gaussien (GP) est un outil central en apprentissage automatique et en statistique. Ce modèle bayésien non paramétrique est particulièrement prisé pour sa flexibilité, permettant de modéliser une large gamme de fonctions, y compris des relations non linéaires. Contrairement à des approches de régression classiques, elle fournit des prédictions accompagnées de mesures d'incertitude, une caractéristique cruciale pour de nombreuses applications nécessitant des résultats robustes et interprétables.

Cependant, malgré ces avantages, la méthode traditionnelle de régression GP souffre de limitations computationnelles majeures. En effet, elle présente une complexité temporelle de $O(n^3)$ et une complexité mémoire de $O(n^2)$, où n représente la taille du jeu de données. Ces contraintes rendent son utilisation impraticable pour des ensembles de données volumineux. Par exemple, pour un dataset contenant 10^3 points, le temps de calcul dépasse largement la minute, et ce délai croît de manière exponentielle avec la taille des données. Ces limitations constituent un frein majeur à son adoption pour des applications modernes impliquant des données massives. La montée en puissance de la science des données et l'ère du Big Data imposent ainsi des défis nouveaux aux modèles de régression GP.

1.2 Etat de l'Art des Méthodes de Régression GP Scalables

Face à ces défis, des méthodes dites **scalables** ont été développées pour étendre les capacités des régressions GP traditionnelles. Ces techniques partagent un objectif commun : réduire la taille de la matrice de covariance à inverser, afin de limiter considérablement la complexité computationnelle.

L'état de l'art distingue principalement deux grandes approches :
- **Les approximations globales :** Ces méthodes condensent l'information présente dans l'ensemble des données. Elles incluent les approximations *sparse* basées sur des points pseudo-induits et les approximations structurées exploitant des propriétés spécifiques de la matrice de noyau.
- **Les approximations locales :** Ces techniques segmentent les données en sous-espaces et appliquent des modèles locaux.

De plus, la performance des régressions GP dépend fortement du choix de la fonction de noyau et des hyperparamètres associés. Les avancées récentes incluent des algorithmes d'optimisation automatique des noyaux, ainsi que des stratégies de sous-échantillonnage pour initialiser les hyperparamètres de manière efficace. Ces innovations réduisent les coûts computationnels tout en préservant la précision des prédictions.

1.3 Défis et Problématiques

Malgré ces avancées, plusieurs défis subsistent :
- **Réduction de la complexité computationnelle :** Trouver des méthodes permettant de réduire les complexités temporelle et spatiale des calculs sans compromettre les performances.
- **Précision des prédictions :** Maintenir des estimations fiables pour les prédictions et les mesures d'incertitude, malgré les approximations nécessaires.
- **Compromis entre efficacité et performance :** Identifier des compromis acceptables entre efficacité computationnelle et performance statistique.

— **Applications variées :** Étendre l'applicabilité des modèles aux contextes variés tels que les données spatiales ou les tâches multitâches.

1.4 Objectifs du Projet

Ce projet vise à explorer et évaluer des méthodes scalables de régression par processus gaussien afin de :

1. **Réduire les coûts computationnels :** Identifier des approches efficaces pour traiter de grands ensembles de données.
2. **Préserver la précision :** Garantir la fiabilité des mesures d'incertitude et des prédictions.
3. **Analyser les performances :** Réaliser une analyse comparative des techniques disponibles en fonction des caractéristiques des données.
4. **Proposer des recommandations :** Fournir des lignes directrices pour choisir les meilleures méthodes selon les cas d'utilisation spécifiques.

1.5 Importance du Projet

Avec l'explosion des volumes de données dans les applications modernes, il est crucial de développer des outils algorithmiques alliant scalabilité et performance. Les processus gaussiens, bien que puissants, nécessitent des adaptations pour être utilisés efficacement dans des contextes Big Data. Ce projet contribue à cet objectif, en réalisant une évaluation critique des méthodes existantes et en fournissant des recommandations pratiques pour les chercheurs et praticiens.

1.6 Organisation du Rapport

Ce rapport est structuré comme suit :
— Le **Chapitre 2** présente les fondements théoriques sur la régression et les processus gaussiens.
— Le **Chapitre 3** détaille la régression par processus gaussiens, de la phase d'entraînement à la prédiction.
— Le **Chapitre 4** explore les méthodes d'approximation adaptées aux grands ensembles de données.
— Enfin, la **Conclusion** résume les résultats et propose des recommandations pour les travaux futurs.

Chapitre 2

Fondamentaux mathématiques

2.1 Interpolation, Régression et Classification

Dans le cadre de ce projet, les termes *interpolation*, *régression* et *classification* seront fréquemment utilisés. Il est donc crucial de bien comprendre les différences entre ces concepts :

- Un modèle d'**interpolation** passe exactement par les points de la base de données d'entrée. Il cherche à reproduire les observations sans introduire d'erreur.
- Un modèle de **régression**, en revanche, ne passe pas nécessairement par les points d'entrée. Il vise à minimiser une erreur globale tout en modélisant la tendance générale des données.
- La **classification** associe à des entrées des valeurs discrètes, souvent appelées *étiquettes* (par exemple, *chien* ou *chat*). Elle est utilisée lorsque la sortie appartient à un ensemble de classes distinctes.

2.2 Cadre et Notations

Considérons un ensemble d'observations $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \ldots, n\}$ où :

- x_i représente un vecteur d'entrée (*input*) de dimension D ($x_i \in R^D$),
- y_i est un scalaire de sortie (*target* ou *output*), donc $y_i \in R$.

Notre objectif est de définir une fonction qui, pour toute nouvelle entrée $x \in R^D$, estime la sortie y correspondante en suivant le même modèle que celui appris à partir de l'ensemble d'entraînement $\mathcal{D}$.

2.3 Processus Stochastique et Processus Gaussien

Les processus gaussiens (GP) sont des processus stochastiques continus qui permettent de modéliser des distributions de probabilité sur des fonctions. Ils sont largement utilisés dans des contextes variés tels que l'apprentissage actif, l'optimisation ou encore les modèles financiers.

2.3.1 Définitions

2.3.1.0.1 Processus stochastique : Un processus stochastique $X = (X_t)_{t \in T}$ est une famille de variables aléatoires X_t, indexée par un ensemble T.

- Si $T \subseteq N$, on parle de **vecteur aléatoire**.
- Si $T \subseteq R^D$, on parle de **champ aléatoire**.

2.3.1.0.2 Processus gaussien : Un processus stochastique $(X_t)_{t \in T}$ est dit **gaussien** si, pour toute famille finie extraite $\tilde{T} \subseteq T$, $(X_t)_{t \in \tilde{T}}$ forme un vecteur gaussien.

En d'autres termes, $X = (X_t)_{t \in T}$ est un processus gaussien si toute combinaison linéaire des variables aléatoires $a_1 X_{t_1} + \ldots + a_n X_{t_n}$ suit une loi normale.

2.3.2 Caractéristiques des Processus Gaussiens

2.3.2.0.1 Vecteur gaussien : La loi d'un vecteur gaussien $(X_1, \ldots, X_n)$ est entièrement définie par :

- Son vecteur moyenne $E[X] = (E[X_1], \ldots, E[X_n])$,
- Sa matrice de covariance $\text{cov}(X_i, X_j)_{1 \leq i,j \leq n}$.

2.3.2.0.2 Processus gaussien : De manière similaire, un processus gaussien est entièrement caractérisé par :

— Sa **fonction moyenne** (*mean function*) : $m(t) = E[X_t]$,

— Sa **fonction de covariance** (*kernel ou noyau*) : $K(s,t) = \text{cov}(X_s, X_t)$.

La fonction de covariance joue un rôle clé en décrivant la structure de dépendance entre les différentes variables aléatoires du processus.

Ainsi, un processus gaussien est noté $GP(\mu(x), \Sigma(x,x'))$. Pour tout ensemble de points d'entrée $X = \{x_1, \ldots, x_n\}$, les sorties correspondantes $y = [y_1, \ldots, y_n]^T$ suivent une distribution normale multivariée donnée par :

$$y \sim \mathcal{N}(\mu(X), \Sigma(X)) \tag{2.1}$$

avec $\mu(X)$ le vecteur des moyennes et $\Sigma(X)$ la matrice de covariance évaluée sur X.

2.3.3 Exemple Intuitif

Imaginons un processus gaussien défini pour modéliser la température d'une ville au cours de la journée. Ici :

— La fonction moyenne $m(t)$ pourrait représenter la température moyenne attendue à une heure donnée t.

— La fonction de covariance $K(s,t)$ pourrait refléter la manière dont les températures à deux moments différents s et t sont corrélées.

2.3.4 Utilisation Pratique des Processus Gaussiens

Les processus gaussiens sont particulièrement utiles pour des tâches de régression où l'incertitude est importante. Par exemple, dans des systèmes de prévision météorologique ou des modèles financiers, ils permettent de fournir des prédictions accompagnées d'intervalles de confiance, ce qui améliore leur interprétabilité.

Chapitre 3

Déterminer le processus gaussien adapté à notre jeu de données

3.1 Kernels et hyperparamètres

Comme expliqué dans le chapitre 2 le Kernel correspond à la fonction de covariance. Il existe de nombreux types de fonctions de covariance qui ne seront pas détaillés dans ce rapport (pour plus d'information voir [1]). Chacune de ces fonctions de covariance a une forme spéciale. Par exemple, voici le cas le plus répandu, à savoir du kernel qui se construit à base de l'exponentiel carré :

$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_p^q$$

Ce kernel (comme tous les autres) s'accompagne de plusieurs paramètres libres que l'on appelle hyperparamètres, dans ce cas précis on en retrouve 3 :

— σ_f^2 correspond à la variance de l'entrée
— l^2 correspond à la longueur caractéristique
— σ_n^2 correspond à la variance du bruit

Ces hyperparamètres influencent directement la manière dont le processus gaussien modélise les relations entre les points. Leur rôle est essentiel pour :

— Ajuster la complexité du modèle. Par exemple, un l trop grand peut conduire à une sous-adaptation (*underfitting*), tandis qu'un l trop petit peut entraîner une sur-adaptation (*overfitting*).
— Contrôler la précision des prédictions ainsi que l'incertitude associée à ces dernières.

Dans le but d'avoir un processus gaussien qui représente le mieux possible notre jeu de données, il faut optimiser les valeurs des hyperparamètres en utilisant les données d'entraînement.

3.2 Calibration des hyperparamètres et ajustement du modèle

3.2.1 La théorie

Les hyperparamètres doivent être déduits du dataset d'entraînement pendant la phase de "training". Pour ce faire on utilise le principe de maximum de vraisemblance. La vraisemblance quantifie la probabilité d'observer les données d'entrées sous les hypothèses du modèle, on cherche donc à maximiser cette valeur.

De manière plus formelle, on introduit :
— θ le vecteur des hyperparamètres de notre kernel
— X notre jeu de données d'entrée
— y la sortie que l'on observe

D'après [1] :
Les observations $\mathbf{y}$ suivent une distribution gaussienne multivariée avec une moyenne nulle (choisie par souci de simplicité) :

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K(\theta) + \sigma_n^2 I) \tag{3.1}$$

avec :
— K : la matrice de covariance calculée à partir du kernel RBF, K est une fonction de θ,
— $\sigma_n^2 I$: le bruit ajouté à la covariance.

La densité de probabilité d'une gaussienne multivariée est donnée par :

$$p(\mathbf{y} \mid \mathbf{X}, \theta) = \frac{1}{\sqrt{(2\pi)^n |K + \sigma_n^2 I|}} \exp\left(-\frac{1}{2}\mathbf{y}^\top (K(\theta) + \sigma_n^2 I)^{-1}\mathbf{y}\right) \tag{3.2}$$

En prenant le logarithme, on obtient la log-vraisemblance :

$$\log p(\mathbf{y} \mid \mathbf{X}, \theta) = \underbrace{-\frac{1}{2}\mathbf{y}^\top (K(\theta) + \sigma_n^2 I)^{-1}\mathbf{y}}_{\text{distance au modèle}} \underbrace{-\frac{1}{2}\log |K(\theta) + \sigma_n^2 I|}_{\text{pénalité de complexité}} - \frac{n}{2}\log(2\pi)$$

On souhaite trouver la valeur de θ qui tend à maximiser cette valeur. Il peut y avoir des maxima locaux et dans les faits chaque maximum local correspond à une interprétation différente des données.

3.2.2 Complexité

Il existe plusieurs manières d'évaluer la valeur de θ maximisante. Les manières les plus efficaces ont recours à des algorithmes d'optimisation qui demandent le calcul du gradient de la log-vraissemblance. On peut montrer que [1] :

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\log p(\mathbf{y} \mid \mathbf{X}, \theta) &= \frac{1}{2} y^\top K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}\left(K^{-1}\frac{\partial K}{\partial \theta_j}\right) \\ &= \frac{1}{2}\operatorname{tr}\left(\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - K^{-1}\right)\frac{\partial K}{\partial \theta_j}\right) \text{ avec } \boldsymbol{\alpha} = K^{-1}\mathbf{y}\end{aligned}$$

Il reste donc à calculer les valeurs de ces dérivées pour obtenir le gradient. Le calcul de la trace demande de calculer les coefficients diagonaux de la matrice :

$$\left(\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - K^{-1}\right)\frac{\partial K}{\partial \theta_j}$$

Pour ce faire, il est nécessaire de calculer chacun des coefficients de K^{-1}. Or K est une matrice symétrique définie positive. La méthode la plus efficace pour calculer l'inverse de cette matrice consiste à d'abord calculer sa décomposition de Cholesky : $K = L * L^T$ avec L une matrice triangulaire inférieure (la complexité de calcul de cette décomposition est en $O(n^3)$). Ainsi il ne reste plus qu'à résoudre un système triangulaire, ce qui se fait avec une complexité en $O(n^2)$. La complexité de calcul de K^{-1} est donc finalement de $O(n^3)$. On en retrouve une preuve complète dans [**?**].

On peut donc trouver les valeurs des hyperparamètres qui annulent ces dérivées. Dans la grande majorité des cas, il n'existe pas de solution analytique, on procède donc de manière numérique.

Il faut ensuite calculer la valeur de θ_j qui permet d'annuler la trace on peut pour se faire utiliser une méthode type méthode de Newton-Raphson.

Finalement la recherche des hyperparamètres se fait en $O(n^3)$ en suivant cette méthode.

Pour résoudre le problème, on peut aussi utiliser le principe de descente du gradient (Méthode Nocedal and Wright (2006)) pour trouver le θ qui minimise :

$$\frac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} + \frac{1}{2}\log |K + \sigma_n^2 I|$$

3.3 Prédictions pour de nouveaux points

3.3.1 La théorie

On considère le dataset $\mathcal{D} = \{(x_i, y_i) | i = 1, ..., n\}$ tel que

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \overset{iid}{\sim} \mathrm{N}\left(0, \sigma^2\right)$$

On place une distribution à priori sur l'espace des fonctions en utilisant un processus gaussien.

$$f \sim \mathcal{G}P(\mu(\cdot), K(\cdot, \cdot))$$

On considère un autre dataset qui est le dataset de test $\mathcal{T} = \{(x_i^*, y_i^*) | i = 1, ..., m\}$. On a :

$$y_i^* = f(x_i^*) + \varepsilon_i^*, \quad \varepsilon_i^* \overset{iid}{\sim} \mathrm{N}\left(0, \sigma^2\right)$$

On introduit certaines notations :

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{X}^* = \begin{bmatrix} x_1^* \\ \vdots \\ x_m^* \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, \quad \mathbf{Y}^* = \begin{bmatrix} y_1^* \\ \vdots \\ y_m^* \end{bmatrix},$$

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix}, \quad \boldsymbol{\varepsilon}^* = \begin{bmatrix} \varepsilon_1^* \\ \vdots \\ \varepsilon_m^* \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}, \quad \mathbf{f}^* = \begin{bmatrix} f(x_1^*) \\ \vdots \\ f(x_m^*) \end{bmatrix}.$$

Notre objectif est de déterminer la distribution sur $\mathbf{Y}^*$ sachant qu'on sait qu'il s'agît d'un vecteur gaussien :

$$\mathbf{Y}^* \mid \mathbf{X}^*, \mathbf{X} \sim \mathrm{N}(\mu^*, \Sigma^*)$$

On observe que

$$\begin{bmatrix} \mathbf{Y} \\ \mathbf{Y}^* \end{bmatrix} | \, \mathbf{X}^*, \mathbf{X} = \begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} + \begin{bmatrix} \boldsymbol{\varepsilon} \\ \boldsymbol{\varepsilon}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I} & K(\mathbf{X}, \mathbf{X}^*) \\ K(\mathbf{X}^*, \mathbf{X}) & K(\mathbf{X}^*, \mathbf{X}^*) + \sigma^2 \mathbf{I} \end{bmatrix}\right)$$

Avec $K(\mathbf{X}, \mathbf{X})$ la matrice $n \times n$ telle que $\mathbf{K}_{i,j} = K(x_i, x_j)$, (K est grâce à la phase de training) et $K(\mathbf{X}^*, \mathbf{X}^*)$ la matrice $m \times m$ telle que $\mathbf{K}_{ij}^* = K\left(x_i^*, x_j^*\right)$.

On souhaite connaître la distribution sur $\mathbf{Y}^*$, on écrit donc la loi conditionnelle sur $\mathbf{Y}^* \mid \mathbf{X}^*, \mathbf{X}$. Sachant la distribution normale multivariable précédente, en conditionnant sur l'ensemble des autres variables, on peut obtenir le vecteur moyenne et la covariance pour la loi gausienne recherchée.

$$\mu^* = K(\mathbf{X}^*, \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{Y}$$
$$\Sigma^* = K(\mathbf{X}^*, \mathbf{X}^*) + \sigma^2 \mathbf{I} - K(\mathbf{X}^*, \mathbf{X}) \left(K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}\right)^{-1} K(\mathbf{X}, \mathbf{X}^*)$$

Ces deux quantités permettent de produire des prédictions fiables tout en indiquant leur niveau d'incertitude.

3.4 Complexité

A première vue le calcul de μ^* et de Σ^* demande de calculer l'inverse de la matrice $K(\mathbf{X}, \mathbf{X}) + \sigma^2 \mathbf{I}$ cette matrice étant symétrique définie positive, on peut utiliser la décomposition de Cholesky ce qui demande une complexité en $O(n^3)$. Le calcul de cet inverse est l'opération la plus complexe à effectuer.

En effet, pour une matrice K définie positive, la décomposition de Cholesky a pour but de trouver L triangulaire inférieure telle que : $K = LL^T$.

3.4.1 Preuve par récurrence sur n de la complexité en $\frac{n^3}{3}$ de la décomposition de Cholesky

Soit A une matrice carrée symétrique définie positive de taille $n \times n$. La décomposition de Cholesky de A consiste à trouver une matrice triangulaire inférieure L telle que $A = LL^T$.

Nous allons prouver par récurrence sur n que la complexité de cette décomposition est de $\frac{n^3}{3}$.

Initialisation

Pour $n = 1$, la matrice A est un scalaire positif a et sa décomposition de Cholesky est $L = \sqrt{a}$. La complexité de cette opération est de 1, ce qui est cohérent avec la formule $\frac{n^3}{3} = \frac{1}{3}$.

Hérédité

Supposons que la formule soit vraie pour une taille $n-1$, c'est-à-dire que la complexité de la décomposition de Cholesky d'une matrice de taille $(n-1) \times (n-1)$ est de $\frac{(n-1)^3}{3}$.

Nous allons montrer que la formule est vraie pour une taille n, c'est-à-dire que la complexité de la décomposition de Cholesky d'une matrice de taille $n \times n$ est de $\frac{n^3}{3}$.

Pour cela, nous allons décomposer la matrice A en blocs :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{pmatrix}$$

où A_{11} est une matrice carrée de taille $(n-1) \times (n-1)$, A_{12} est une matrice de taille $(n-1) \times 1$ et A_{22} est un scalaire positif.

La décomposition de Cholesky de A s'écrit alors :

$$A = \begin{pmatrix} L_{11} & 0 \\ L_{12}^T & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^T & L_{12} \\ 0 & L_{22}^T \end{pmatrix}$$

où L_{11} est la décomposition de Cholesky de A_{11}, $L_{12} = A_{12}/L_{11}$ et $L_{22} = \sqrt{A_{22} - L_{12}^T L_{12}}$.

La complexité de cette décomposition est la somme des complexités des opérations suivantes :

— La décomposition de Cholesky de A_{11}, qui est de $\frac{(n-1)^3}{3}$ par hypothèse de récurrence.

— Le calcul de L_{12}, qui est de $(n-1)$ opérations de division et de multiplication.

— Le calcul de L_{22}, qui est de 1 opération de soustraction, 1 opération de multiplication et 1 opération de racine carrée.

— La mise à jour de A_{22}, qui est de $(n-1)$ opérations de soustraction et de multiplication.

La complexité totale de la décomposition de Cholesky de A est donc :

$$\frac{(n-1)^3}{3} + (n-1) + (n-1) + 1 + 1 + 1 = \frac{n^3 - 3n^2 + 3n + 1}{3} = \frac{n^3}{3}$$

Ceci prouve que la formule est vraie pour une taille n, et donc par récurrence sur n que la complexité de la décomposition de Cholesky d'une matrice carrée symétrique définie positive de taille $n \times n$ est de $\frac{n^3}{3}$.

Chapitre 4

Méthode d'approximation dans le cadre de grands jeux de données

Comme vu précédemment pour trouver le processus gaussien correspondant à notre entrée, nous devons résoudre le problème associé à la matrice de covariance. Ce problème a une complexité en $O(n^3)$ ce qui rend la résolution impossible dans le cadre de grands jeux de données.

4.1 Sparse Gaussian Process - Méthode des points induits

4.1.1 Théorie mathématique

4.1.1.1 Introduction

Les points d'induction sont essentiellement motivés par la question suivante : plutôt que de modéliser l'ensemble des entrées X, que se passerait-il si nous nous concentrions plutôt sur la modélisation d'un sous-ensemble plus petit de régions de l'espace d'entrée ? Si nous étions en mesure de le faire, nous pourrions exploiter toute redondance et corrélation dans l'espace d'entrée pour réduire le temps de calcul.

4.1.1.2 Sélection des points d'induction

Pour ce faire, nous devons d'abord choisir un ensemble de points d'induction, qui sont une sorte de "points de données imaginaires". En pratique, plusieurs méthodes simples sont utilisées pour sélectionner ces points d'induction. Les plus courantes incluent la sélection aléatoire parmi les points d'entraînement, ainsi que l'algorithme des k plus proches voisins pour déterminer les centres de données.

4.1.1.3 Notation et vraisemblance

Appelons les M points d'induction $\bar{X} = (\bar{x}_1, \ldots, \bar{x}_M)$ et leurs sorties imaginaires correspondantes $\bar{y} = (\bar{y}_1, \ldots, \bar{y}_M)$.
On peut alors écrire la vraisemblance complète du modèle (voir [2] et [3]) :

$$p(y|X, \bar{X}, \Theta) = \mathcal{N}(y|0, K_{\mathrm{SPGP},N} + \sigma^2 I)$$

avec

$$K_{\mathrm{SPGP}}(x_n, x_{n'}) = K_{nM} K_M^{-1} K_{Mn'} + \lambda_n \delta_{nn'}$$
$$\lambda_n = K_{nn} - K_{nM} K_M^{-1} K_{Mn}$$

4.1.1.4 Training

La vraisemblance marginale est une fonction des hyperparamètres Θ et des pseudo-intrants $\bar{X}$, et elle est utilisée pour entraîner le SPGP. Les hyperparamètres et les pseudo-intrants sont appris conjointement en maximisant la vraisemblance à l'aide de la méthode de l'ascension du gradient, comme fait précédemment dans le chapitre 3. L'efficacité computationnelle provient du fait que la matrice de covariance $K_{\mathrm{SPGP},N}$ est constituée d'une somme d'une partie de faible rang et d'une partie diagonale, ce qui permet son inversion en un temps de $\mathcal{O}(M^2 N)$ au lieu de $\mathcal{O}(N^3)$.

4.1.1.5 Prédiction

Comme dans le processus gaussien standard, la distribution prédictive peut être calculée en considérant un nouveau point x^* et en conditionnant sur les données D :

$$p(y|x^*, D, \bar{X}, \Theta) = \mathcal{N}(y|\mu_*, \sigma_*^2)$$

$$\mu_* = K_{*M}Q^{-1}K_{MN}(\Lambda + \sigma^2 I)^{-1}y$$

$$\sigma_*^2 = K_{**} - K_{*M}(K_M^{-1} - Q^{-1})K_{M*} + \sigma^2$$

où $Q = K_M + K_{MN}(\Lambda + \sigma^2 I)^{-1}K_{NM}$ et $\Lambda = \text{diag}(\lambda)$. Après une phase de pré-calcul en $\mathcal{O}(M^2N)$, la moyenne prédictive et la variance prédictive peuvent être calculées respectivement en $\mathcal{O}(M)$ et $\mathcal{O}(M^2)$ par point de test.

4.1.1.6 Complexité

En résumé la méthode de l'inducing point permet de réaliser facilement l'inversion avec une matrice diagonale et de réduire le coût de calcul de N^3 à NM^2 avec $M << N$.

4.1.2 Résultats numériques pour le sparse GP

4.1.2.1 Expériences sur un jeu de donné synthétique

Pour évaluer la performance de la méthode des points induits, nous avons appliqué cette approche sur un jeu de donnée synthétique (une fonction sinus). L'objectif principal est de comparer la précision des prédictions obtenues avec cette méthode à celle des processus gaussiens standards tout en mesurant le gain en temps de calcul.

Configuration des expériences :
On travaille avec la librairie GPflow, qui implémente à la fois la méthode classique du gaussian process (GP) mais aussi la méthode du sparse gaussien process (SPG).
On a travaille avec un jeu de données synthétique généré à partir d'une fonction sinus avec bruit gaussien. Le jeu de données contient jusqu'à 10^4 points pour tester la scalabilité.

Les m points ont été extraits aléatoirement parmi les données d'entraînement (on prend m = 10% de la taille du dataset).

Analyse : La métriques d'évaluation comprennent :
— Erreur quadratique moyenne (RMSE) : Pour mesurer la précision des prédictions.
— Temps de training : Pour évaluer la vitesse d'entraînement
— Temps de prédiction : Pour évaluer la vitesse de prédiction.

Résultats : Dans ces conditions, on obtient les résultats suivants :

Performances temporelles :

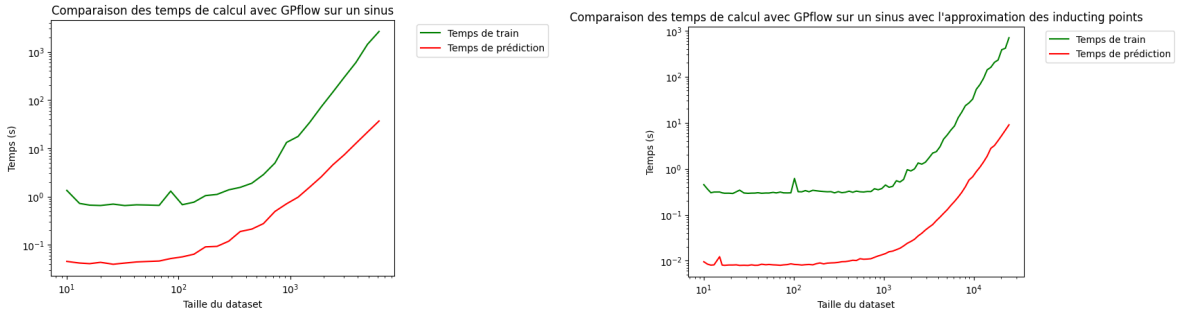


FIGURE 4.1 – Comparaison des performances temporelles : (gauche) GP, (droite) SGP avec $m = 10\%$ de la taille du dataset.

On observe une amélioration significative des performances temporelles, tant pour l'entraînement que pour la prédiction. En effet, on observe que la méthode des Sparse Gaussian Processes (SPG) permet une accélération d'un facteur d'environ 10 par rapport aux processus gaussiens classiques (GP).

Performances en terme de précision :

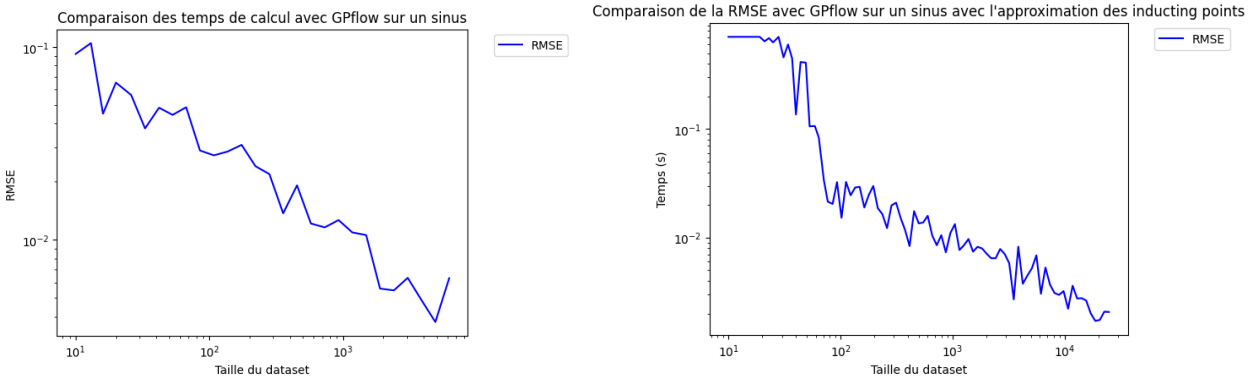


FIGURE 4.2 – Comparaison des précisions : (gauche) Précision du GP, (droite) Précision du SGP avec $m = 10\%$ de la taille du dataset.

Malgré une diminution de la précision, qui est particulièrement notable pour les jeux de données de petite taille, la méthode SPG parvient à maintenir une performance prédictive globalement très satisfaisante et proche de celle des processus gaussiens classiques (GP). Cependant, pour des tailles de datasets plus importantes, la précision du SPG reste remarquablement stable, ce qui en fait une solution équilibrée entre efficacité computationnelle et qualité des prédictions.

4.1.2.2 Influence du nombre de points induits sur les résultats

Jusqu'à maintenant on avait fixé la valeur de $\frac{m}{n}$ dans les expériences. Maintenant on réalise cette fois l'expérience de varier la valeur de m (le nombre de points induits) tout en maintenant fixe la taille du dataset (n), cela permet d'étudier l'influence de m sur les performances. Cette analyse met en évidence le rôle clé de m dans le compromis entre précision des prédictions et temps de calcul.

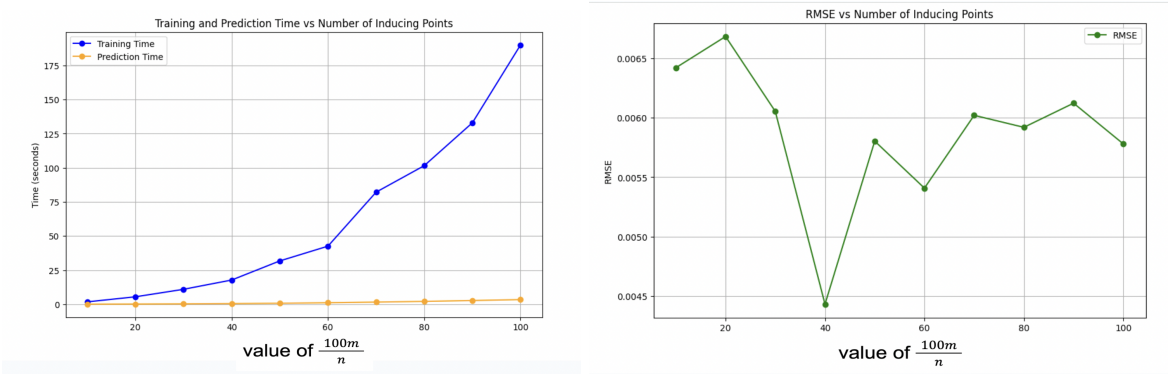


FIGURE 4.3 – Influence de $\frac{100m}{n}$: (gauche) Évolution des temps de calcul, (droite) Précision.

En étudiant l'influence de m, on observe que l'augmentation de m entraîne une hausse significative des temps de calcul, tandis que la diminution de m impacte légèrement la précision, comme en témoigne une légère augmentation de la RMSE. Le compromis entre le temps gagné et la perte minime de précision rend l'utilisation d'un m relativement petit (environ 10 à 20 % de n) bien plus intéressante. Cette stratégie permet de conserver une précision acceptable tout en bénéficiant de gains substantiels en efficacité computationnelle.

4.1.2.3 Expériences sur des jeux de données réels

Pour compléter l'évaluation, nous avons testé la méthode SPG sur deux jeux de données réels :

— **Évolution du CO2 (Mauna Loa) :** Ce dataset contient 2225 observations de la concentration mensuelle moyenne de CO2 (en ppm) mesurée à l'Observatoire Mauna Loa entre 1958 et 2001. L'objectif est de modéliser ces concentrations en fonction du temps et d'extrapoler les tendances futures.

— **Diamonds :** Ce dataset comprend les prix et caractéristiques (coupe, couleur, clarté, etc.) de 53 940 diamants (on s'arrête ici à 5000 données). L'objectif est de prédire le prix des diamants à partir de leurs attributs.

Résultats expérimentaux :

Dataset	Taille du dataset	Nombre de features	Méthode	RMSE	Temps d'entraînement (s)
Mauna Loa (CO2)	2225	1	GP	2.1649	71.26
			SPG (10% random)	2.1637 (↓ 0.06%)	4.09 (↓ 94.26%)
			SPG (10% k-means)	2.164 (↓ 0.04%)	3.93 (↓ 94.48%)
			SPG (50% random)	2.2503 (↑ 3.94%)	37.88 (↓ 46.85%)
			SPG (50% k-means)	2.1638 (↓ 0.05%)	7.92 (↓ 88.89%)
Diamonds	5000	10	GP	225.6611	3041.79
			SPG (10% random)	279.6288 (↑ 23.92%)	220.77 (↓ 92.74%)
			SPG (10% k-means)	264.6620 (↑ 17.27%)	199.74 (↓ 93.43%)

TABLE 4.1 – Comparaison des performances entre GP et SPG sur des jeux de données réels

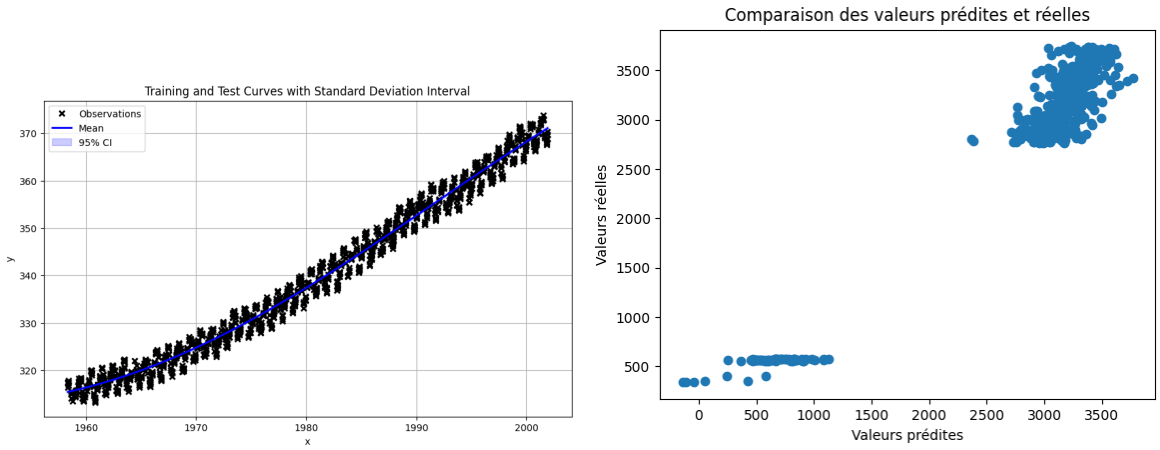


FIGURE 4.4 – Résultats des datasets utilisés : (gauche) Mauna Loa CO2, (droite) Diamonds

Analyse :

— Pour le dataset **CO2** :
  — La méthode SPG permet une réduction significative du temps d'entraînement, atteignant jusqu'à **94.48% de réduction** pour $m = 10\%$ (k-means), soit une accélération d'un facteur 18 par rapport à GP.
  — La précision reste quasiment inchangée, avec une diminution de la RMSE de **0.06%** pour $m = 10\%$ (random) et **0.05%** pour $m = 50\%$ (k-means), ce qui démontre l'efficacité de la méthode SPG.
  — Même avec une proportion de $m = 50\%$, SPG permet une réduction du temps de calcul de **46.85%**, tout en maintenant une performance prédictive quasi-identique avec une diminution de la RMSE de **0.05%** pour $m = 50\%$ (k-means).

— Pour le dataset **Diamonds** :
  — La méthode SPG réduit le temps de calcul de manière significative, avec une diminution allant jusqu'à **93.43%** pour $m = 10\%$ (k-means), permettant une accélération d'un facteur 15 à 16.
  — Une augmentation de la RMSE est observée, avec une hausse de **23.92%** pour $m = 10\%$ (random) et **17.27%** pour $m = 10\%$ (k-means). Cette perte de précision reste modérée compte tenu du gain en rapidité.
  — Un ajustement du paramètre m pourrait être envisagé afin de trouver un compromis optimal entre rapidité et précision selon les besoins de l'application.

— De manière générale, l'utilisation de SPG permet une **réduction du temps de calcul allant jusqu'à 94%**, ce qui en fait une approche particulièrement adaptée aux datasets de grande taille ou aux environnements nécessitant des calculs intensifs.

— La flexibilité dans le choix de m permet d'adapter la méthode aux exigences spécifiques du problème, en trouvant un équilibre entre gain de temps et performance prédictive.

4.2 Stochastic Variational Inference (SVI)

4.2.1 Théorie mathématique

4.2.1.1 Introduction

L'objectif de l'inférence variationnelle est d'approximer des distributions de probabilités complexes par des distributions plus simples, appelées distributions variationnelles, afin de rendre l'inférence plus efficace. Dans le cadre de SVI, on cherche à améliorer cette approximation en procédant de manière incrémentale à l'aide de sous-ensembles de données (mini-lots), permettant de traiter efficacement de grands jeux de données.

SVI repose sur l'approche variationnelle des variables inductrices proposée par Titsias (2009) [4], qui est une variante des processus gaussiens parcimonieux (Sparse Gaussian Processes, Sparse GP). Cette approche introduit des points dits "inducteurs" pour réduire la complexité computationnelle tout en préservant une précision acceptable.

4.2.1.2 Modélisation

Considérons un vecteur de données y, où chaque entrée y_i est une observation bruitée de la fonction $f(x_i)$ pour les points $X = \{x_i\}_{i=1}^n$. Nous supposons que le bruit est gaussien indépendant de précision β. Un processus gaussien est introduit comme a priori sur $f(\cdot)$, avec le vecteur f contenant les valeurs de la fonction aux points X.

Nous introduisons également un ensemble de variables inductrices, représentées par le vecteur u contenant les valeurs de la fonction aux points $Z = \{z_i\}_{i=1}^m$.

Les distributions conditionnelles peuvent être exprimées comme :

$$p(y|f) = \mathcal{N}(y|f, \beta^{-1}I),$$
$$p(f|u) = \mathcal{N}(f|K_{nm}K_{mm}^{-1}u, K_e),$$
$$p(u) = \mathcal{N}(u|0, K_{mm}),$$

avec $K_e = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}$.

4.2.1.3 Application de l'inégalité de Jensen

En appliquant l'inégalité de Jensen à $p(y|u)$:

$$\log p(y|u) \geq E_{p(f|u)}[\log p(y|f)] = L_1.$$

Pour un bruit gaussien, la borne L_1 peut être calculée avec une complexité $\mathcal{O}(m^3)$. La borne peut être exprimée sous la forme suivante :

$$\exp(L_1) = \prod_{i=1}^{n} \mathcal{N}(y_i|\mu_i, \beta^{-1}) \exp\left(-\frac{1}{2}\beta\tilde{k}_{i,i}\right),$$

où $\mu = K_{nm}K_{mm}^{-1}u$ et $\tilde{k}_{i,i}$ est l'élément diagonal de K_e.

4.2.1.4 Inférence Variationnelle Stochastique (SVI)

L'inférence variationnelle stochastique (SVI) introduit des variables globales (les variables induites u). La SVI exploite une factorisation variationnelle pour permettre l'optimisation incrémentale à l'aide de sous-ensembles (mini-lots) de données. Cela réduit la complexité globale tout en conservant une précision compétitive. Comme illustré dans la figure 4.5.

L'approximation variationnelle repose sur la borne inférieure suivante pour la vraisemblance marginale des données [5] :

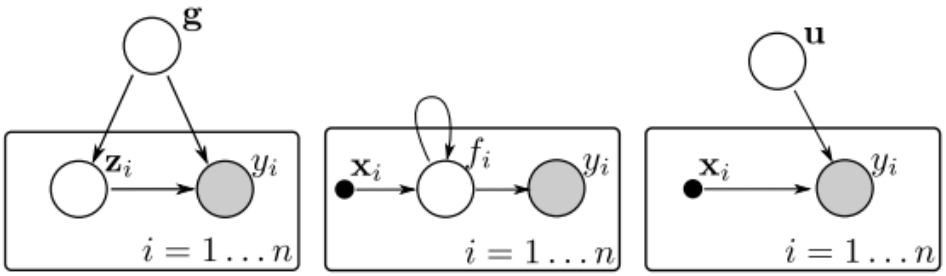
$$\log p(y|X) \geq E_{q(u)}\left[\log p(y|u) + \log p(u) - \log q(u)\right],$$

où $q(u) = \mathcal{N}(u|m, S)$ est une distribution variationnelle gaussienne avec moyenne m et covariance S. Ces paramètres sont optimisés pour maximiser la borne inférieure. Cette borne inférieure est :

$$L_3 = \sum_{i=1}^{n}\left[\log \mathcal{N}(y_i|k_i^T K_{mm}^{-1}m, \beta^{-1}) - \frac{1}{2}\beta\tilde{k}_{i,i} - \frac{1}{2}\mathrm{tr}(S\Lambda_i)\right] - \mathrm{KL}(q(u)||p(u)).$$

Elle fournit une estimation approchée de la vraisemblance $p(y|X)$.

En utilisant cette formulation, chaque lot de données contribue de manière indépendante à l'optimisation de la distribution variationnelle, ce qui rend le processus compatible avec des approches incrémentales basées sur des mini-lots.



(a) Requirements for SVI (b) Gaussian Process regression (c) Variational GP regression

FIGURE 4.5 – Modèles graphiques montrant (a) la structure requise pour un modèle probabiliste pour la SVI, avec des variables globales g et des variables latentes z, (b) le modèle graphique correspondant à une régression par processus gaussien, et (c) le modèle graphique pour les processus gaussiens parcimonieux (sparse GP), avec des variables induites u agissant comme variables globales.

4.2.1.5 Optimisation par gradients naturels

Pour stabiliser l'optimisation, les gradients naturels sont utilisés. Ils sont définis comme :

$$\tilde{g}(\theta) = G(\theta)^{-1} \frac{\partial L}{\partial \theta},$$

où $G(\theta)$ représente l'information de Fisher associée aux paramètres θ de la distribution variationnelle. Les paramètres θ de $q(u)$ sont exprimés en termes de la moyenne m et de la matrice de covariance S.

Les mises à jour incrémentales des paramètres se font selon la règle suivante :

$$\theta^{(t+1)} = \theta^{(t)} + \eta \tilde{g}(\theta),$$

où η est le taux d'apprentissage. Le gradient est calculé itérativement pendant le training, et nous choisissons le nombre d'itérations.

4.2.1.6 Prédiction avec SVI

Après l'entraînement, les paramètres m et S de $q(u)$ contiennent l'information nécessaire pour faire des prédictions. Les étapes sont les suivantes :

1. Distribution prédictive : Pour une nouvelle entrée x_*, la distribution prédictive est donnée par :

$$q(y_*|x_*) = \int p(y_*|x_*, u) q(u) du. \tag{4.1}$$

2. Calcul de la moyenne et de la variance prédictives :

$$E[y_*] = k_*^\top K_{mm}^{-1} m, \tag{4.2}$$
$$\mathrm{Var}[y_*] = k_{**} - k_*^\top K_{mm}^{-1} (K_{mm} - S) K_{mm}^{-1} k_*, \tag{4.3}$$

   où k_* est le vecteur de covariances entre x_* et les points induits, et k_{**} est la variance du noyau pour x_*.
3. Prédictions : Pour chaque nouvelle observation x_*, évaluer la moyenne et la variance prédictive pour obtenir une estimation et une incertitude associée.

4.2.1.7 Complexité computationnelle

La complexité temporelle de l'algorithme d'inférence variationnelle stochastique (SVI) dépend de plusieurs facteurs, notamment la taille des données n, le nombre de points inducteurs m, la taille du mini-lot b et le nombre d'itérations T. L'objectif est d'optimiser la borne variationnelle inférieure L_3 tout en minimisant le coût computationnel.

La complexité temporelle par itération est donnée par l'expression suivante :

$$\mathcal{O}(m^3 + bm^2)$$

Ainsi, la complexité totale pour T itérations est :

$$\mathcal{O}(T(m^3 + bm^2))$$

Les différentes contributions sont les suivantes :
— Formation de la matrice de covariance K_{mm} entre les points inducteurs : $\mathcal{O}(m^3)$.
— Calcul des covariances croisées K_{bm} entre le mini-lot et les points inducteurs : $\mathcal{O}(bm^2)$.
— Mise à jour des paramètres variationnels pour maximiser la borne L_3 : $\mathcal{O}(bm^2)$.

4.2.2 Résultats numériques pour l'inférence variationnelle stochastique (SVI)

Pour èvaluer les performances de l'inférence variationnelle stochastique (SVI), nous avons appliqué cette méthode à l'aide de la librairie GPflow. L'objectif ètait de mesurer la précision des prédictions (RMSE), le temps d'entraînement, et le temps de prédiction, tout en mettant en èvidence la scalabilité de la SVI. Les expériences ont ètè réalisées sur le jeu de données Diamonds, en faisant varier trois paramètres principaux : le nombre de points d'induction m, le nombre d'itérations, et la taille des mini-lots.

4.2.2.1 Configuration des expériences

- **Librairie utilisée :** GPflow, qui propose une implémentation efficace des modèles à processus gaussiens, incluant la SVI.
- **Jeu de données :** *Diamonds*, comprenant des caractéristiques de diamants pour prédire leur prix.
- **Paramètres variables :**
  - Nombre de points d'induction (m) : varie entre 50 et 200.
  - Nombre d'itérations d'entraînement : varie entre 500 et 10 000.
  - Taille des mini-lots : varie entre 50 et 500.

4.2.2.2 Résultats

Les résultats présentent les temps d'entraînement et les erreurs quadratiques moyennes (RMSE) obtenus pour différentes configurations. Les figures suivantes illustrent l'évolution des performances en fonction des paramètres variables.

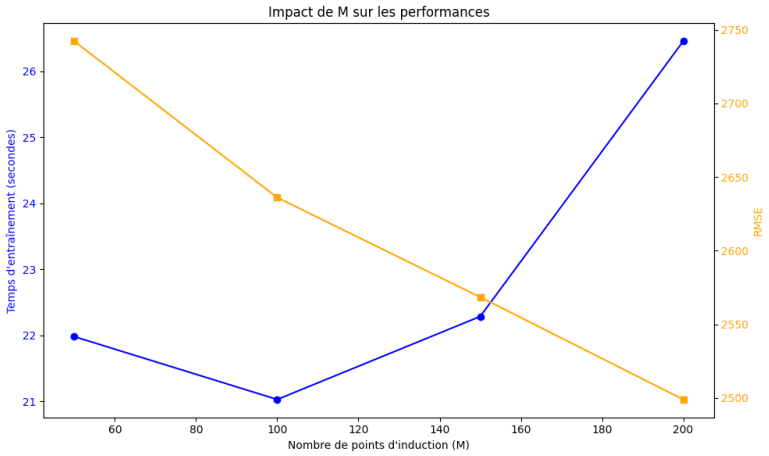


FIGURE 4.6 – Évolution du temps d'entraînement et de la RMSE en fonction du nombre de points d'induction m.

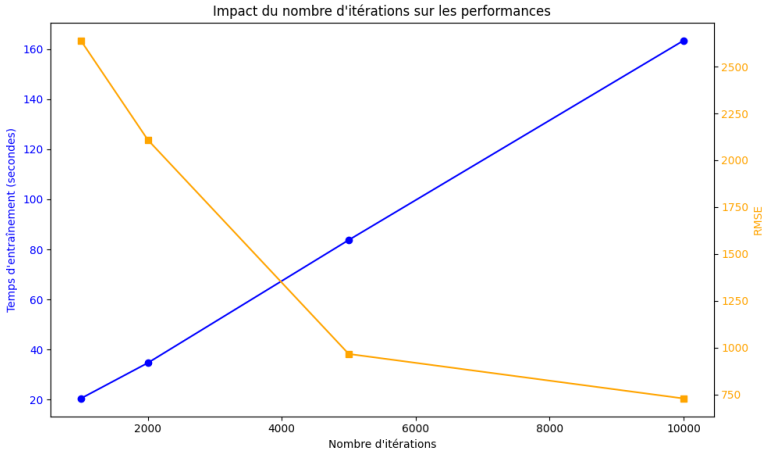


FIGURE 4.7 – Évolution du temps d'entraînement et de la RMSE en fonction du nombre d'itérations.

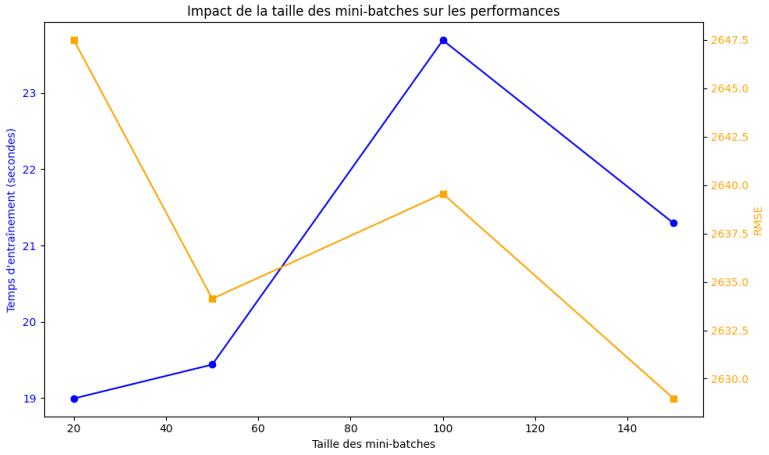


FIGURE 4.8 – Évolution du temps d'entraînement et de la RMSE en fonction de la taille des mini-lots.

Les figures montrent une tendance claire quant aux compromis entre le temps d'entraînement et la précision des modèles (RMSE) selon les différents paramètres.

- **Nombre de points d'induction (m) :** L'augmentation de m améliore la précision du modèle, comme en témoigne la diminution de la RMSE. Cependant, cela se fait au dépens d'un temps d'entraînement plus élevé, à cause de la complexité accrue des calculs.
- **Nombre d'itérations :** Une augmentation du nombre d'itérations réduit la RMSE jusqu'à un certain point, mais l'effet marginal diminue, suggérant une convergence. Le temps d'entraînement croît de manière proportionnelle au nombre d'itérations.
- **Taille des mini-lots :** Les mini-lots changent grandement le résultat.

Ces observations permettent de déterminer les paramètres optimaux en fonction des priorités, que ce soit le temps d'entraînement ou la précision des résultats.

D'après les résultats précédents, il semble que choisir $M = 100$ (nombre de points d'induction) et une taille de mini-batch de 50 soit un bon compromis entre précision et temps d'entraînement. En suivant cette configuration, le modèle SVGP a été entraîné avec 20 000 itérations. Les résultats obtenus sont les suivants :

- **Temps d'entraînement :** 276.73 secondes ($\downarrow$ 90.89% par rapport à GP).
- **RMSE :** 240.11 ($\uparrow$ 6.41% par rapport à GP).

Ces résultats confirment l'efficacité de la méthode **SVI** pour atteindre une précision satisfaisante tout en maintenant un temps d'entraînement raisonnable. Bien qu'une augmentation du nombre d'itérations aurait permis de réduire davantage la RMSE, cela aurait également entraîné une hausse du temps de calcul. Cette flexibilité permet d'adapter la configuration selon les contraintes de performance et de ressources disponibles.

4.3 Local Approximate Gaussian Processes (laGP)

4.3.1 Théorie mathématique

4.3.1.1 Introduction

Les processus gaussiens locaux approximés (laGP) permettent de modéliser efficacement de grands ensembles de données en exploitant des matrices creuses et des méthodes locales. Cette approche a été introduite et détaillée dans le package R 'laGP' par Gramacy (2016) [6].

Ce model s'appuit sur l'idée que pour prédire à un point donné x, seuls les points voisins influencent significativement le résultat. L'objectif est donc de construire un sous-ensemble local de données, $D_n(x)$, autour du point de prédiction, et de ne calculer les prédictions que sur ce sous-ensemble. Cela permet de réduire la complexité de $O(N^3)$ à $O(n^3)$, où $n \ll N$.

4.3.1.2 Sélection des sous-ensembles locaux

Deux stratégies principales sont utilisées pour construire $D_n(x)$. Ces méthodes reposent sur des paramètres, `k_start` et `k_end`, qui déterminent respectivement le nombre initial et final de points voisins pris en compte dans le sous-ensemble local $D_n(x)$. Ces valeurs peuvent être ajustées en fonction des besoins spécifiques de la modélisation, afin d'équilibrer précision et temps de calcul.

- **Active Learning Cohn (ALC)** : Cette méthode sélectionne les points à inclure dans $D_n(x)$ en maximisant la réduction de la variance prédictive. À chaque itération, un point x_{j+1} est ajouté au sous-ensemble actuel $D_j(x)$ de manière à minimiser l'incertitude prédictive globale.
- **Nearest Neighbors (NN)** : Une approche plus simple où $D_n(x)$ est constitué des n points les plus proches de x selon une métrique donnée (souvent la distance euclidienne).

Le critère ALC repose sur la minimisation de la variance prédictive à un point x. La réduction de la variance est donnée par :

$$v_j(x;\theta) - v_{j+1}(x;\theta),$$

où $v_j(x;\theta)$ est la variance prédictive calculée après l'ajout du point x_{j+1} au sous-ensemble local.

Dans le critère NN, $D_n(x)$ est simplement formé des n points d'entraînement les plus proches de x, sans optimisation supplémentaire. Bien que cette méthode soit rapide, elle peut manquer de précision pour des ensembles de données complexes.

4.3.1.3 Modélisation locale et prédictions

Pour un sous-ensemble $D_n(x)$, les prédictions locales pour un nouveau point x^* sont basées sur la distribution conditionnelle suivante :

$$p(y^*|x^*, D_n(x)) = \mathcal{N}(\mu^*(x^*), \sigma^2(x^*)),$$

où la moyenne prédictive $\mu^*(x^*)$ et la variance prédictive $\sigma^2(x^*)$ sont calculées en utilisant uniquement $D_n(x)$.

4.3.1.4 Complexité computationnelle

- **ALC** : La complexité de l'optimisation itérative pour sélectionner les points est $O(n^3)$ pour chaque point de prédiction.
- **NN** : La recherche des voisins proches est plus rapide, avec une complexité typique de $O(N \log n)$ en utilisant des structures de données comme les arbres k-d.

4.3.2 Résultats numériques pour la méthode `laGP`

Pour évaluer les performances de la méthode `laGP`, nous avons utilisé le package R `laGP` pour ajuster un modèle sur le jeu de données Diamonds. L'objectif était de comparer les deux approches principales d'approximation fournies par `laGP` : la méthode ALC (*Active Learning Cohn*) et l'approximation NN (*Nearest Neighbor*). Les résultats obtenus incluent la RMSE et le temps d'entraînement.

4.3.2.1 Configuration des expériences

- **Librairie utilisée :** `laGP`.
- **Jeu de données :** *Diamonds*, réduit à 5000 observations et pré-traité pour transformer les variables catégorielles en numériques.
- **Séparation des données :** 90 % des données pour l'entraînement, 10 % pour le test.
- **Paramètres principaux :**
  - **Méthode ALC :** $k_{\text{start}} = 6$, $k_{\text{end}} = 60$.
  - **Méthode NN :** $k_{\text{start}} = 6$, $k_{\text{end}} = 30$.

4.3.2.2 Résultats

4.3.2.2.1 Méthode ALC

- **Temps d'entraînement :** 49.5 secondes (↓ 82.01% par rapport à GP).
- **RMSE :** 295 (↑ 30.76% par rapport à GP).

4.3.2.2.2 Méthode NN

- **Temps d'entraînement :** 1.1 secondes (↓ 99.96% par rapport à GP).
- **RMSE :** 242 (↑ 7.23% par rapport à GP).

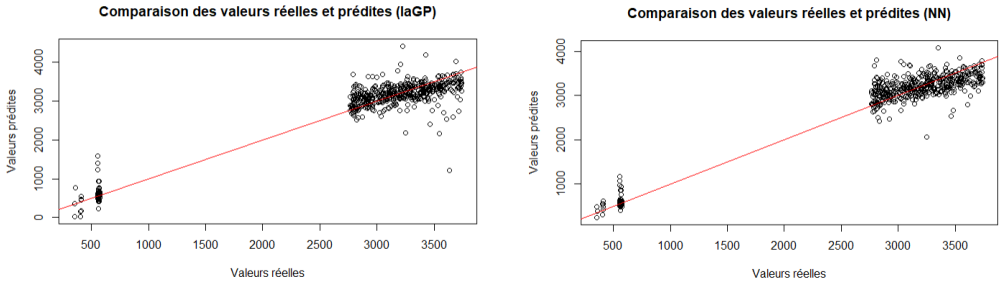


FIGURE 4.9 – Résultats de la librairie laGP

4.3.2.3 Analyse des performances

- **Méthode ALC :** Cette méthode offre une approche itérative et optimisée pour sélectionner les points dans $D_n(x)$, mais cela se traduit par un temps d'entraînement nettement plus long (49.5 secondes, soit une réduction de 82.01%). De plus, bien que la méthode soit conçue pour réduire l'incertitude prédictive, la RMSE obtenue reste relativement élevée avec une augmentation de 30.76% (RMSE : 295), ce qui peut limiter son intérêt pour ce jeu de données.
- **Méthode NN :** La méthode NN est beaucoup plus rapide (1.1 seconde, soit une réduction de 99.96%) tout en offrant une meilleure précision avec une augmentation de RMSE modérée de seulement 7.23% (RMSE : 242). Cela en fait une option plus adaptée pour ce jeu de données, où la rapidité et la précision sont des critères importants.

Ces résultats montrent que la méthode `laGP` offre une flexibilité entre précision et temps d'entraînement en fonction de la méthode choisie. Toutefois, dans le cadre du jeu de données **Diamonds**, la méthode NN semble mieux répondre aux besoins, en fournissant un bon compromis entre rapidité et précision. La méthode ALC, bien que plus coûteuse en calcul, pourrait s'avérer plus pertinente pour des applications nécessitant une incertitude prédictive plus faible ou une meilleure gestion des données complexes.

Chapitre 5

Comparaison finale des résultats

5.1 Synthèse et analyse comparative

Dans ce chapitre, nous procédons à une analyse approfondie des performances des différentes approches étudiées pour résoudre les limitations computationnelles des processus gaussiens sur des ensembles de données volumineux. L'objectif est de mettre en lumière les compromis entre précision, complexité et scalabilité.

5.1.1 Précision des méthodes

Les méthodes traditionnelles de régression par processus gaussiens offrent une précision optimale mais présentent des limitations majeures en termes de temps de calcul et de mémoire. Les approches scalables, telles que les processus gaussiens épars (Sparse GP), l'inférence variationnelle stochastique (SVI) et les processus gaussiens locaux approximés (laGP), permettent de réduire la complexité computationnelle tout en conservant une précision raisonnable.

- **SVI** : Offre un bon équilibre entre précision et temps de calcul, idéal pour des ensembles de données massifs grâce à son approche par mini-lots.
- **laGP (NN)** : Se distingue par son efficacité sur des sous-ensembles locaux, offrant un temps de calcul minimal tout en conservant une précision acceptable.
- **Sparse GP** : Offre des performances équilibrées, il s'agit d'une méthode simple et efficace.

5.1.2 Scalabilité et performances computationnelles

En termes de scalabilité, la méthode SVI est la plus adaptée aux grands ensembles de données, permettant une mise à l'échelle efficace grâce à son optimisation incrémentale. La méthode laGP, bien que rapide sur des sous-ensembles homogènes, montre ses limites face à des données très hétérogènes.

5.1.3 Résumé des résultats

Le tableau suivant résume les principales performances des différentes approches évaluées sur le dataset *Diamonds* :

Table 5.1 – Comparaison des méthodes de régression par processus gaussiens

Méthode	**RMSE**	**Temps d'entraînement (s)**	**Complexité**
GP standard	225.66	3041.79	$O(n^3)$
Sparse GP (10%)	264.66 (+17.3%)	199.74 (-93.4%)	$O(nm^2)$
SVI	240.11 (+6.4%)	276.73 (-90.9%)	$O(T(m^3 + bm^2))$
laGP (NN)	242.00 (+7.2%)	1.1 (-99.96%)	$O(n \log n)$

Ces résultats montrent que le choix de la méthode dépend fortement des priorités en termes de rapidité ou de précision.

Chapitre 6

Conclusion

Ce projet a permis d'explorer et d'évaluer diverses approches scalables de régression par processus gaussiens pour des ensembles de données volumineux. Les principaux enseignements sont :

- La méthode SGP est simple est efficace.
- La méthode SVI est idéale pour des applications nécessitant une haute scalabilité et une gestion efficace des ressources computationnelles.
- La méthode laGP (NN) constitue une alternative rapide et efficace pour des applications nécessitant une faible latence.

En conclusion, les résultats obtenus démontrent que l'utilisation de techniques scalables permet d'étendre significativement l'applicabilité des processus gaussiens à des contextes de *Big Data*, offrant ainsi des opportunités prometteuses pour des domaines tels que la santé, la finance et l'optimisation industrielle.

Bibliographie

[1] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. the MIT Press, 2006.

[2] Andy Jones. Inducing points for gaussian processes. Technical report.

[3] Edward Sneson ; Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. Technical report, 2005.

[4] Michalis K. Titsias. Variational learning of inducing variables in sparse gaussian processes. Technical report, 2009.

[5] James Hensman ; Nicolo Fusi ; Neil D. Lawrence. Gaussian processes for big data. Technical report, 2013.

[6] Robert B. Gramacy. lagp : Large-scale spatial modeling via local approximate gaussian processes in r. Technical report, Virginia Tech, 2016.