

**CENTRALE
LYON**

ÉCOLE CENTRALE LYON

RAPPORT

**Processus Gaussiens Augmentés par Réseaux
de Neurones : De l'Inférence Variationnelle
aux Deep Kernel Learning**

Élèves :

Pierre JOLY

Adam EL MANOUZI

Enseignant :

Christophette BLANCHET

April 8, 2025

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 5 |
| 1.1 | Contexte et motivation | 5 |
| 1.2 | Problématique | 5 |
| 1.3 | Organisation du rapport | 6 |
| 2 | Krigeage | 7 |
| 2.1 | Principe | 7 |
| 2.2 | Régression par Processus Gaussiens | 11 |
| 2.3 | Optimisation d'hyperparamètres | 13 |
| 2.4 | Limites et extensions | 14 |
| 3 | Modélisation bayésienne paramétrique | 15 |
| 3.1 | Principe | 15 |
| 3.2 | Régression linéaire bayésienne | 15 |
| 3.3 | Problème de l'inférence exacte | 19 |
| 3.4 | Méthodes de Monte-Carlo par chaînes de Markov | 20 |
| 4 | Inférence Variationnelle | 22 |
| 4.1 | Principe | 22 |
| 4.2 | Famille Variationnelle | 23 |
| 4.3 | Optimisation | 23 |
| 4.4 | Comparaison entre MCMC et Inférence Variationnelle | 31 |
| 5 | Parcimonie et Processus Gaussiens | 33 |
| 5.1 | Inférence variationnelle pour les GPs | 33 |
| 5.2 | Distribution prédictive et approximation variationnelle | 34 |
| 5.3 | Borne inférieure variationnelle (ELBO) | 36 |
| 5.4 | Sélection des points induits | 42 |
| 5.5 | Inférence Variationnelle Stochastique (SVI) | 42 |
| 6 | Deep Kernel Learning | 45 |
| 6.1 | Problème des GP en haute dimension | 45 |
| 6.2 | Principe du Deep Kernel Learning | 45 |
| 6.3 | Formulation mathématique | 46 |
| 6.4 | Optimisation du modèle | 47 |
| 6.5 | Formalisation algorithmique | 47 |
| 6.6 | Vers une extension scalable des DKL : les SVDKL | 51 |
| 7 | Expériences et Résultats | 52 |
| 7.1 | Jeu de données : Olivetti Faces | 52 |
| 7.2 | Protocole expérimental | 52 |
| 7.3 | Réseaux de neurones | 53 |
| 7.4 | Optimisation | 55 |
| 7.5 | Différentiation automatique | 56 |
| 7.6 | Résultats | 58 |
| 8 | Conclusion | 60 |

Notations

Ensembles et espaces

- \mathbb{N} : ensemble des entiers naturels.
- \mathbb{Z} : ensemble des entiers relatifs.
- \mathbb{R} : ensemble des nombres réels.
- \mathbb{R}^n : espace euclidien de dimension n .
- \mathcal{X} : espace des variables d'entrée.
- \mathcal{Y} : espace des variables de sortie.
- X : matrice (généralement $n \times d$) regroupant les n observations d'entrée (chaque ligne de X est x_i^\top).
- Y : vecteur (dimension n) regroupant les observations de sortie $(y_1, \dots, y_n)^\top$.

Opérations et fonctions usuelles

- $\|x\|$: norme euclidienne d'un vecteur $x \in \mathbb{R}^n$.
- $\langle x, y \rangle$: produit scalaire de $x, y \in \mathbb{R}^n$.
- $\det(A)$: déterminant de la matrice A .
- $\text{tr}(A)$: trace de la matrice A .
- $f : \mathcal{X} \rightarrow \mathcal{Y}$: fonction définie de \mathcal{X} vers \mathcal{Y} .
- ∇f : gradient de la fonction f .
- Δf : laplacien de f .
- $O(g(n))$: notation de Landau pour un comportement asymptotique.

Probabilités et statistiques

- $\mathbb{E}[X]$: espérance de la variable aléatoire X .
- $\mathbb{V}[X]$: variance de la variable aléatoire X .
- $\text{Cov}[X]$: covariance de la variable aléatoire X .
- $\mathbb{P}(A)$: probabilité de l'événement A .
- $X \sim \mathcal{N}(\mu, \sigma^2)$: la variable aléatoire X suit une loi normale de moyenne μ et de variance σ^2 .
- $p(\theta)$: loi a priori (prior) sur le paramètre θ .

- $p(\theta \mid y)$: loi a posteriori (posterior) sur le paramètre θ , en présence des données observées y .
- $p(y \mid \theta)$: vraisemblance (likelihood) des observations y sachant θ .
- $p(y)$: vraisemblance marginale (évidence) des observations y .

Inférence bayésienne et variationnelle

- $\text{KL}(q \parallel p)$: divergence de Kullback–Leibler entre les distributions q et p .
- $\mathcal{L}(q)$ ou ELBO : (*Evidence Lower BOund*) borne inférieure sur $\log p(y)$ utilisée en inférence variationnelle.

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(y, \theta)] - \mathbb{E}_q[\log q(\theta)] = \mathbb{E}_q[\log p(y \mid \theta)] - \text{KL}(q \parallel p).$$

- $q(\theta)$: distribution variationnelle choisie pour approximer la loi a posteriori $p(\theta \mid y)$.
- λ : paramètres variationnel décrivant $q_\lambda(\theta)$
- $\widehat{\nabla_\lambda \mathcal{L}}$: estimateur Monte-Carlo du gradient de l'ELBO par rapport aux λ .
- S : nombre d'échantillons Monte-Carlo utilisés pour l'estimation

Processus gaussiens et approximations à points induits

- $\{x_i, y_i\}_{i=1}^n$: n observations (entrées $x_i \in \mathbb{R}^d$, sorties $y_i \in \mathbb{R}$).
- GP (Processus Gaussien) : famille de fonctions où tout sous-ensemble fini de valeurs suit une loi normale multidimensionnelle.
- $k(\mathbf{x}, \mathbf{x}')$: fonction noyau (kernel) du processus gaussien.
- K_{nn} : matrice de covariance $n \times n$ entre les n points d'entraînement.
- m : nombre de “points induits” (ou pseudo-points) utilisés dans l'approximation parcimonieuse.
- X_m : ensemble (ou matrice) des m points induits; f_m : valeurs du processus gaussien aux points induits X_m .
- K_{mm} : matrice de covariance $m \times m$ entre les points induits X_m .
- K_{nm}, K_{mn} : matrices de covariance croisées entre les n points d'entraînement et les m points induits ($K_{mn} = K_{nm}^\top$).
- $Q_{nn} = K_{nm} K_{mm}^{-1} K_{mn}$: approximation de rang m de la matrice K_{nn} par méthode de Nyström.
- σ^2 : variance (ou hyperparamètre de bruit) supposée connue ou estimée (par ex. en régression gaussienne).
- $\phi(f_m)$: distribution variationnelle pour les valeurs du processus aux points induits (souvent gaussienne $\mathcal{N}(\mu, \Lambda)$).

Divers

- I : matrice identité (taille précisée par le contexte).
- A^{-1} : matrice inverse de A (lorsque A est inversible).
- (y_{n+1}, x_{n+1}) : nouvelles données (hors échantillon) pour lesquelles on prédit ou infère une valeur.

1 Introduction

1.1 Contexte et motivation

Les processus gaussiens (GP) sont une classe d'outils probabilistes puissants et flexibles, utilisés en apprentissage automatique pour des tâches telles que la régression, la classification et l'optimisation. Leur capacité à modéliser des fonctions de manière non paramétrique, tout en offrant une quantification rigoureuse de l'incertitude, en fait des candidats idéaux pour de nombreuses applications. Historiquement, les GP tirent leurs origines du *krigeage*, une méthode d'interpolation spatiale stochastique développée en géostatistique pour estimer des valeurs en des points non échantillonnés à partir d'observations voisines. Dans le cadre de l'apprentissage automatique, les GP généralisent cette approche en définissant un *prior* sur l'espace des fonctions, mis à jour via le théorème de Bayes pour obtenir une distribution postérieure conditionnée aux données observées.

Cependant, malgré leurs avantages, les GP présentent des limitations significatives dans des contextes modernes impliquant des volumes de données importants ou des espaces d'entrée de haute dimension. Leur principal défaut réside dans leur complexité computationnelle : l'entraînement d'un GP standard nécessite l'inversion d'une matrice de covariance de taille $n \times n$, où n est le nombre de points de données, entraînant une complexité en $\mathcal{O}(n^3)$. Cette contrainte rend les GP difficilement applicables à des ensembles dépassant quelques milliers d'observations. De plus, dans des espaces de grande dimension, la définition d'un noyau approprié capable de capturer les structures non linéaires complexes des données devient problématique, limitant ainsi leur efficacité.

En parallèle, les réseaux de neurones profonds (*Deep Neural Network*, DNN) ont démontré une capacité exceptionnelle à apprendre des représentations complexes à partir de données brutes, notamment dans des domaines comme la vision par ordinateur et le traitement du langage naturel. Cependant, les DNN standards produisent des prédictions déterministes et ne fournissent pas de quantification systématique de l'incertitude, ce qui constitue une lacune critique pour des applications nécessitant une fiabilité élevée, telles que la prise de décision automatisée ou l'analyse scientifique.

Pour surmonter ces limitations, des approches hybrides combinant les forces des GP et des DNN ont émergé. Parmi elles, le *Deep Kernel Learning* (DKL) propose d'utiliser un réseau de neurones pour transformer les données d'entrée dans un espace latent où un GP peut être appliqué plus efficacement. Cette transformation permet d'exploiter la capacité des DNN à capturer des relations non linéaires complexes, tout en préservant la quantification de l'incertitude offerte par les GP. Cependant, même avec le DKL, le coût computationnel reste élevé, nécessitant des techniques d'approximation telles que l'utilisation de points induits ou l'inférence variationnelle pour rendre ces modèles *scalables*.

1.2 Problématique

Ce rapport explore l'intégration des réseaux de neurones avec les processus gaussiens à travers le Deep Kernel Learning, en mettant l'accent sur des techniques d'inférence variationnelle et des approximations parcimonieuses pour améliorer leur scalabilité et leur efficacité. L'objectif principal est de développer des méthodes permettant d'exploiter les

avantages des GP dans des scénarios où les données sont abondantes et complexes, tout en tirant parti des capacités de représentation dans un espace latent des DNN.

1.3 Organisation du rapport

Ce rapport est structuré de manière progressive pour couvrir les concepts fondamentaux, les avancées techniques et les résultats expérimentaux :

- **Section 2** : Krigeage – Présentation des principes du krigeage comme fondement historique des GP, avec ses variantes et ses liens avec la régression par processus gaussiens.
- **Section 3** : Modélisation bayésienne paramétrique – Introduction aux bases de la modélisation bayésienne, illustrée par la régression linéaire bayésienne, en tant que précurseur des GP.
- **Section 4** : Inférence variationnelle – Exploration des méthodes d’approximation de la distribution postérieure pour les modèles complexes, essentielles à la scalabilité des GP.
- **Section 5** : Parcimonie et processus gaussiens – Discussion des techniques d’approximation *sparse*, notamment via les points induits, pour réduire la complexité computationnelle.
- **Section 6** : Deep Kernel Learning – Présentation du DKL, combinant DNN et GP pour améliorer la modélisation en haute dimension.
- **Section 7** : Expériences – Application des concepts abordés sur un jeu de données d’images, illustrant l’efficacité des modèles hybrides.
- **Section 7** : Conclusion.

2 Krigeage

2.1 Principe

Le krigeage est une méthode d'interpolation spatiale stochastique (Matheron, 1969). Il s'agit d'un estimateur linéaire qui utilise la covariance spatiale pour prédire la valeur d'une variable régionale en un point non échantillonné, de l'espace D , à partir des valeurs observées aux points voisins. L'estimateur est modélisé par un champ aléatoire Z . La prédiction par krigeage correspond au meilleur, au sens du minimum de variance, prédicteur linéaire sans biais (BLUP) de Z aux emplacements non échantillonnés. Le processus aléatoire se décompose comme suit :

$$Z(x) = \mu(x) + \delta(x), \quad \forall x \in D,$$

où μ est déterministe et δ une variable aléatoire d'espérance nulle et de structure de dépendance connue. Le modèle du krigeage diffère du modèle de régression classique, car les erreurs sont supposées dépendantes spatialement. Ce processus aléatoire admet une certaine forme de stationnarité.

Stationnarité de Z

Il est nécessaire d'ajouter une forme de régularité pour traiter le problème. Il y a trois notions de régularité qui sont utilisées dans le cadre du krigeage.

Stationnarité stricte : il est supposé que toutes les caractéristiques de la loi de probabilité sont invariantes par translation. Formellement, pour tout h et pour tout $x_i \in D$, le vecteur aléatoire $(Z(x_1), \dots, Z(x_n))$ a la même distribution que $(Z(x_1 + h), \dots, Z(x_n + h))$. C'est une hypothèse très contraignante et peu utilisée en pratique.

La stationnarité au second ordre ou stationnarité faible : il est supposé que les deux premiers moments sont invariants par translations. Formellement :

- $\mu(x) = \mathbb{E}[Z(x)] = \mu$ est constant,
- $\text{Cov}(Z(x), Z(x'))$, la covariance ne dépend que du décalage h entre deux points.

En particulier cela implique que la variance est constante et finie. La stationnarité faible permet de définir un modèle de covariance k global. Cependant, dans de nombreuses situations réelles, la moyenne peut varier spatialement ou la variance peut être non bornée sur de larges domaines.

Stationnarité intrinsèque : il est supposé que les incréments ont une espérance nulle et une variance finie qui ne dépend que de h . Formellement,

- $\mathbb{E}[Z(x + h) - Z(x)] = 0$,
- $\mathbb{V}[Z(x + h) - Z(x)]$ existe, est finie et ne dépend que de h

Ces conditions permettent de construire la fonction de variogramme. La stationnarité intrinsèque est plus générale que la stationnarité au second ordre : tout processus faiblement stationnaire est intrinsèquement stationnaire, mais l'inverse n'est pas forcément vrai.

Type de krigeage

Il existe plusieurs variantes du krigeage qui dépendent de l'hypothèse formulée sur μ .

Krigeage simple : il est supposé que μ est connue et constante sur tout le domaine D . Cette variante est associée à la stationnarité faible.

Krigeage ordinaire : il est supposé que μ inconnue, mais est constante sur tout le domaine D . Cette variante est associée à la stationnarité faible.

Krigeage universel : il est supposé que μ est une combinaison linéaire de fonctions linéairement indépendantes, $\mu = \sum_{l=1}^p \beta_l f_l$. Cette variante est associée à la stationnarité intrinsèque.

Prédiction par krigeage

Soit $x_* \in D$ un point où l'on souhaite estimer $Z(x_*)$ à partir de n observations disponibles $Y = (Z(x_1), \dots, Z(x_n))^T$ $X = (x_1, \dots, x_n)^T$. Le krigeage correspond à une optimisation, au sens du minimum de variance, sur la classe des estimateurs linéaires. Les estimateurs linéaires ont la forme,

$$\hat{Z}(x_*) = \sum_{j=1}^n \lambda_j Z(x_j) \cdot Y,$$

où les λ_j sont les poids à déterminer. L'optimisation est sous contrainte de non-biais soit, $\mathbb{E}[\hat{Z}(x_*)] = \mathbb{E}[Z(x_*)]$.

Le problème d'optimisation s'écrit alors :

$$(\lambda_1, \dots, \lambda_n) = \arg \min_{(\lambda_1, \dots, \lambda_n) \in \mathbb{R}^n} \mathbb{V}[Z(x_*) - \hat{Z}(x_*)] \quad \text{sous la contrainte} \quad \mathbb{E}[\hat{Z}(x_*)] = \mathbb{E}[Z(x_*)].$$

En krigeage simple, la moyenne étant connue, il suffit d'appliquer l'optimisation à l'estimateur centré $Z_0(x) = Z(x) - \mu$ pour que la contrainte soit respectée. En krigeage ordinaire, la condition conduit à,

$$\begin{aligned} \mathbb{E}[Z(x_*)] &= \mathbb{E}[\hat{Z}(x_*)] \\ \iff \mu &= \sum_{j=1}^n \lambda_j \mu \end{aligned}$$

Si la moyenne est nulle alors la contrainte est respectée sinon la contrainte est équivalente à $\sum_{i=1}^n \lambda_i = 1$. Plus généralement en krigeage universel,

$$\begin{aligned} \mathbb{E}[Z(x_*)] &= \mathbb{E}[\hat{Z}(x_*)] \\ \iff \sum_{l=1}^p \beta_l f_l(x_*) &= \sum_{j=1}^n \lambda_j \sum_{l=1}^p \beta_l f_l(x_j) \\ \iff f(x_*)^T \beta &= \left(\sum_{j=1}^n \lambda_j f(x_j)^T \right) \beta, \end{aligned}$$

avec $f = (f_1 \cdots f_p)^\top$ et $\beta = (\beta_1 \cdots \beta_p)^\top$. Cette expression étant valable pour tout β ,

$$\begin{aligned} \mathbb{E}[Z(x_*)] &= \mathbb{E}[\hat{Z}(x_*)] \\ \iff f(x_*) &= \sum_{j=1}^n \lambda_j f(x_j) \\ \iff F(X)^\top \lambda &= f(x_*), \end{aligned}$$

avec $F(X) = (f(x_1) \cdots f(x_n))_{n \times p}^\top$.

La méthode des multiplicateurs de Lagrange est appliquée. Le Lagrangien est défini comme suit,

$$\mathcal{L}(\lambda, \gamma) = \mathbb{V}[Z(x_*) - \hat{Z}(x_*)] + \gamma^\top (F(X)^\top \lambda - f(x_*))$$

Le système d'équation est donné par,

$$\frac{\partial}{\partial \lambda} \mathcal{L}(\lambda, \gamma) = 0 \quad \wedge \quad \frac{\partial}{\partial \gamma} \mathcal{L}(\lambda, \gamma) = 0$$

En développant la variance $\mathbb{V}[Z(x_*) - \hat{Z}(x_*)]$, il vient,

$$\begin{aligned} \mathbb{V}[Z(x_*) - \hat{Z}(x_*)] &= \mathbb{V}[Z(x_*)] + \mathbb{V}[\hat{Z}(x_*)] - 2 \text{Cov}[\hat{Z}(x_*), Z(x_*)] \\ &= \mathbb{V}[Z(x_*)] + \mathbb{V}\left[\sum_{j=1}^n \lambda_j Z(x_j)\right] - 2 \text{Cov}\left[\sum_{j=1}^n \lambda_j Z(x_j), Z(x_*)\right] \\ &= \mathbb{V}[Z(x_*)] + \sum_{j=1}^n \sum_{k=1}^n \lambda_j \lambda_k \text{Cov}[Z(x_j), Z(x_k)] - 2 \sum_{j=1}^n \lambda_j \text{Cov}[Z(x_j), Z(x_*)] \end{aligned}$$

En injectant cette expression dans le système, il vient

$$\begin{cases} \forall i \in [1, n], & -2 \text{Cov}[Z(x_*), Z(x_i)] + \sum_{k=1}^n \lambda_k \text{Cov}[Z(x_i), Z(x_k)] + \sum_{j=1}^n \lambda_j \text{Cov}[Z(x_j), Z(x_i)] \\ & + \gamma^\top \sum_{j=0}^p f(x_j) \lambda_i = 0 \\ F(X)^\top \lambda - f(x_*) = 0. \end{cases}$$

Par symétrie de la covariance il est obtenu,

$$\forall i \in [1, n], \quad 2 \sum_{j=1}^n \lambda_j \text{Cov}[Z(x_j), Z(x_i)] + \gamma^\top \sum_{j=0}^p f(x_j) \lambda_i = 2 \text{Cov}[Z(x_*), Z(x_i)].$$

Il est noté $K_{nn} = (\text{Cov}[Z(x_i), Z(x_j)])_{1 \leq i, j \leq n}$, $\lambda = (\lambda_i)_{1 \leq i \leq n}$ et $c = (\text{Cov}[Z(x_i), Z(x_*)])_{1 \leq i \leq n}$. Alors le système est équivalent à,

$$\begin{pmatrix} K_{nn} & F(X) \\ F(X)^T & 0 \end{pmatrix} \cdot \begin{pmatrix} \lambda \\ \gamma \end{pmatrix} = \begin{pmatrix} c \\ f(x_*) \end{pmatrix}$$

Ainsi,

$$\begin{cases} K_{nn} \lambda + F(X) \gamma = c \\ F(X)^T \lambda = f(x_*) \end{cases}$$

En injectant la première équation dans la deuxième,

$$\begin{cases} \lambda = K_{nn}^{-1} (c - F(X) \gamma) \\ F(X)^T K_{nn}^{-1} (c - F(X) \gamma) = f(x_*) \end{cases}$$

Puis en reformulant,

$$\begin{cases} \lambda = K_{nn}^{-1} (c - F(X) \gamma) \\ F(X)^T K_{nn}^{-1} c - f(x_*) = F(X)^T K_{nn}^{-1} F(X) \gamma \end{cases}$$

Enfin,

$$\begin{cases} \lambda = K_{nn}^{-1} (c - F(X) \gamma) \\ \gamma = (F(X)^T K_{nn}^{-1} F(X))^{-1} [F(X)^T K_{nn}^{-1} c - f(x_*)] \end{cases}$$

En résumé,

$$\lambda = K_{nn}^{-1} (c - F(X) (F(X)^T K_{nn}^{-1} F(X))^{-1} [F(X)^T K_{nn}^{-1} c - f(x_*)])$$

La prédiction est alors donnée par,

$$\hat{Z}(x_*) = f(x_*)^T \hat{\beta} + c^T K_{nn}^{-1} [Y - F(X) \hat{\beta}], \quad \text{avec} \quad \hat{\beta} = (F(X)^T K_{nn}^{-1} F(X))^{-1} F(X)^T K_{nn}^{-1} Y.$$

En répétant le processus, il est possible d'étendre cette formulation à l'estimation de m points cibles $X_* = \{x_{*,1} \dots x_{*,m}\}$. Il est noté,

$$K_{mn} = \begin{pmatrix} c^{(1)} \\ \vdots \\ c^{(m)} \end{pmatrix} = \begin{pmatrix} \text{Cov}[Z(x_{*,1}), Z(x_1)] & \dots & \text{Cov}[Z(x_{*,1}), Z(x_n)] \\ \vdots & \ddots & \vdots \\ \text{Cov}[Z(x_{*,m}), Z(x_1)] & \dots & \text{Cov}[Z(x_{*,m}), Z(x_n)] \end{pmatrix}$$

Il vient ainsi,

$$\hat{Z}(X_*) = F(X_*)^T \hat{\beta} + K_{mn}^T K_{nn}^{-1} [Y - F(X) \hat{\beta}], \quad \text{avec} \quad \hat{\beta} = (F(X)^T K_{nn}^{-1} F(X))^{-1} F(X)^T K_{nn}^{-1} Y.$$

2.2 Régression par Processus Gaussiens

La régression par processus gaussien (GP) se place dans le cadre de krigeage où le champ aléatoire est modélisé par un processus gaussien. Un processus gaussien est défini tel que toutes collections finies $(Z(x_0), \dots, Z(x_n))$ d'éléments du processus suit une loi normale conjointe,

$$(Z(x_0), \dots, Z(x_n)) \sim \mathcal{N}(\mu, K_{nn}),$$

avec $\mu = \mathbb{E}[(Z(x_0), \dots, Z(x_n))]$ et $K_{nn} = (\text{Cov}(Z(x_i), Z(x_j)))_{1 \leq i, j \leq n}$.

La fonction de moyenne $\mu(x)$ dépend du type de krigeage considéré. L'hypothèse de stationnarité intrinsèque impose que la covariance ne dépend que du décalage entre deux points. Il existe plusieurs types de fonction de covariance k communément utilisés. Il est présenté plusieurs noyaux.

Noyau exponentiel

Le noyau exponentiel modélise une décroissance rapide de la dépendance spatiale en fonction de la distance entre deux points :

$$k_{\text{exp}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|}{\ell}\right),$$

où σ^2 est la variance marginale du processus, et ℓ la longueur caractéristique, appelée longueur d'échelle, contrôlant la portée spatiale des corrélations. Ce noyau définit des réalisations continues mais non différentiables en zéro.

Noyau de Matérn

Le noyau de Matérn généralise le noyau exponentiel en introduisant un paramètre ν , appelé paramètre de régularité, qui contrôle la régularité des réalisations du processus :

$$k_{\text{Matérn}}(x, x') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|x - x'\|}{\ell}\right)^\nu K_\nu\left(\sqrt{2\nu} \frac{\|x - x'\|}{\ell}\right),$$

où K_ν est la fonction de Bessel modifiée de seconde espèce et d'ordre ν , Γ est la fonction Gamma, et $\nu > 0$. Lorsque $\nu \rightarrow \infty$, on retrouve le noyau gaussien, et lorsque $\nu = 1/2$, ce noyau coïncide avec le noyau exponentiel. Le noyau de Matérn est très utilisé en pratique grâce à sa flexibilité pour modéliser des régularités variées.

Noyau gaussien (RBF)

Le noyau gaussien, aussi appelé Radial Basis Function (RBF), est l'un des noyaux les plus utilisés en pratique en raison de sa régularité infinie :

$$k_{\text{RBF}}(x, x') = \sigma^2 \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right).$$

Ce noyau génère des réalisations infiniment différentiables, particulièrement adapté lorsque le phénomène étudié est très régulier.

En reprenant la formule du krigeage,

$$\hat{Z}(X_*) = F(X_*)^\top \hat{\beta} + K_{mn}^\top K_{nn}^{-1} [Y - F(X) \hat{\beta}], \quad \text{avec} \quad \hat{\beta} = (F(X)^\top K_{nn}^{-1} F(X))^{-1} F(X)^\top K_{nn}^{-1} Y,$$

Il peut être spécifié, $K_{mn} = (k(x_{*,i}, (x_j)))_{1 \leq i \leq m, 1 \leq j \leq n}$ et $K_{nn} = (k(x_i, x_j))_{1 \leq i, j \leq n}$. Généralement, les noyaux ont pour hyperparamètres σ la variance marginale du processus et ℓ la longueur caractéristique.

Les noyaux s'annulent pour deux points identiques. Ainsi, la matrice K_{nn} a une diagonale nulle, ce qui garantit l'interpolation. En effet, il en résulte que la variance est nulle au point de mesure. Toutefois, les mesures peuvent être bruitées. Alors, pour modifier cette propriété, il faut ajouter un terme sur la diagonale. Les mesures étant i.i.d., le même terme doit être ajouté pour chaque mesure. Un troisième hyperparamètre est alors introduit, σ_n le bruit de mesure et l'on re-note σ_f la variance marginale du processus. Ce dernier paramètre est ajouté comme suit,

$$K_{nn} \leftarrow K_{nn} + \sigma_n I_{nn}$$

La prédiction effectuée par le krigeage fournit une distribution gaussienne conditionnelle en chaque point test X_* , notée :

$$Z(X_*) \mid X, Y \sim \mathcal{N}(\mu(X_*), \Sigma(X_*)),$$

où,

- l'espérance conditionnelle est donnée par :

$$\mu(X_*) = F(X_*)^\top \hat{\beta} + K_{mn}^\top K_{nn}^{-1} (Y - F(X) \hat{\beta}),$$

- la matrice de covariance conditionnelle entre les points test est :

$$\Sigma(X_*) = K_{mm} - K_{mn}^\top K_{nn}^{-1} K_{mn} + (F(X_*) - K_{mn}^\top K_{nn}^{-1} F(X))^\top (F(X)^\top K_{nn}^{-1} F(X))^{-1} (F(X)^\top K_{nn}^{-1} F(X) - (F(X_*) - K_{mn}^\top K_{nn}^{-1} F(X))^\top)$$

En particulier, la variance de prédiction en un point x_* est donnée par :

$$\mathbb{V}[Z(x_*) \mid X, Y] = k(x_*, x_*) - k(x_*, X)^\top K_{nn}^{-1} k(x_*, X) + r(x_*)^\top (F^\top K_{nn}^{-1} F)^{-1} r(x_*),$$

$$\text{avec } r(x_*) = F(x_*) - F^\top K_{nn}^{-1} k(x_*, X).$$

La régression par GP est résolue pour un jeu d'hyperparamètres $\theta = \{\sigma_f, \ell, \sigma_n\}$ donné. Seulement, ces paramètres influent grandement sur la qualité de la prédiction et leur valeur optimale dépend des données. Il est donc nécessaire d'employer une méthode d'optimisation sur ces hyperparamètres.

2.3 Optimisation d'hyperparamètres

La qualité de la prédiction par processus gaussiens dépend fortement du choix des hyperparamètres du noyau $\theta = \{\sigma_f, \ell, \sigma_n\}$. Ces hyperparamètres déterminent la forme et la régularité des fonctions réalisables, ainsi que l'importance relative du bruit de mesure. Leur optimisation s'effectue généralement par maximisation de la log-vraisemblance marginale. Les observations suivent une loi normale,

$$Y = Z(X) \sim \mathcal{N}(\mu(X), K_{nn}(\theta)).$$

Ainsi la vraisemblance est donnée par,

$$p(Y | X, \theta) = \frac{1}{(2\pi)^{n/2} |K_{nn}(\theta)|^{1/2}} \exp \left(-\frac{1}{2} (Y - \mu(X))^\top K_{nn}^{-1}(\theta) (Y - \mu(X)) \right).$$

Finalement, il est obtenu,

$$\log p(Y | X, \theta) = -\frac{1}{2} Y^\top K_{nn}^{-1}(\theta) Y - \frac{1}{2} \log |K_{nn}(\theta)| - \frac{n}{2} \log(2\pi).$$

En pratique, la maximisation se fait souvent via des méthodes de descente de gradient, notamment L-BFGS. L-BFGS est une méthode de descente de gradient utilisant une approximation de la Hessienne pour accélérer la convergence.

Les méthodes de descente de gradient nécessitent le calcul des dérivées partielles. Le calcul des dérivées partielles de la log-vraisemblance marginale, par rapport à un hyperparamètre θ_j , s'effectue comme suit,

$$\frac{\partial}{\partial \theta_j} \log p(Y | X, \theta) = \frac{\partial}{\partial \theta_j} \left(-\frac{1}{2} Y^\top K_{nn}^{-1} Y - \frac{1}{2} \log |K_{nn}| - \frac{n}{2} \log(2\pi) \right).$$

Le terme $\frac{n}{2} \log(2\pi)$ est constant et disparaît à la dérivation,

$$\frac{\partial}{\partial \theta_j} \log p(Y | X, \theta) = -\frac{1}{2} \frac{\partial}{\partial \theta_j} (Y^\top K_{nn}^{-1} Y) - \frac{1}{2} \frac{\partial}{\partial \theta_j} (\log |K_{nn}|).$$

Chacun des termes est calculé séparément. Sachant que $\frac{\partial K_{nn}^{-1}}{\partial \theta_j} = -K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} K_{nn}^{-1}$, il vient :

$$-\frac{1}{2} \frac{\partial}{\partial \theta_j} (Y^\top K_{nn}^{-1} Y) = -\frac{1}{2} Y^\top \left(-K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} K_{nn}^{-1} \right) Y = \frac{1}{2} Y^\top K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} K_{nn}^{-1} Y.$$

En posant $\alpha = K_{nn}^{-1} Y$, il vient,

$$= \frac{1}{2} \alpha^\top \frac{\partial K_{nn}}{\partial \theta_j} \alpha.$$

La dérivée du logarithme du déterminant est :

$$-\frac{1}{2} \frac{\partial}{\partial \theta_j} \log |K_{nn}| = -\frac{1}{2} \text{tr} \left(K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} \right).$$

Il est donc obtenu,

$$\frac{\partial}{\partial \theta_j} \log p(Y|X, \theta) = \frac{1}{2} \alpha^\top \frac{\partial K_{nn}}{\partial \theta_j} \alpha - \frac{1}{2} \text{tr} \left(K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} \right).$$

En remarquant que $\alpha^\top \frac{\partial K_{nn}}{\partial \theta_j} \alpha$ est un scalaire, égal à sa propre trace,

$$\frac{\partial}{\partial \theta_j} \log p(Y|X, \theta) = \frac{1}{2} \left(\text{tr} \left(\alpha \alpha^\top \frac{\partial K_{nn}}{\partial \theta_j} \right) - \text{tr} \left(K_{nn}^{-1} \frac{\partial K_{nn}}{\partial \theta_j} \right) \right).$$

Finalement, en factorisant sous une seule trace :

$$\frac{\partial}{\partial \theta_j} \log p(Y|X, \theta) = \frac{1}{2} \text{tr} \left[(\alpha \alpha^\top - K_{nn}^{-1}) \frac{\partial K_{nn}}{\partial \theta_j} \right],$$

en rappelant que $\alpha = K_{nn}^{-1} Y$, il est finalement obtenu,

$$\frac{\partial}{\partial \theta_j} \log p(Y|X, \theta) = \frac{1}{2} \text{tr} \left[(K_{nn}^{-1} Y Y^\top K_{nn}^{-1} - K_{nn}^{-1}) \frac{\partial K_{nn}}{\partial \theta_j} \right].$$

2.4 Limites et extensions

Bien que très flexible et interprétable, la régression par processus gaussiens présente une complexité algorithmique élevée, de l'ordre de $\mathcal{O}(n^3)$. Cette complexité provient principalement du calcul de l'inverse de la matrice de covariance K_{nn} de taille $n \times n$. Cette limitation rend l'approche classique impraticable dès que la taille du jeu de données dépasse quelques milliers de points. Des méthodes ont été développées pour outrepasser ces limitations.

3 Modélisation bayésienne paramétrique

3.1 Principe

En modélisation bayésienne paramétrique, les paramètres du modèle sont considérés comme des variables aléatoires (Laplace, 1820; Box and Tiao, 2011). Une distribution *a priori* $p(\theta)$ est alors assignée aux paramètres θ . En présence d'observations $\{(x_i, y_i)\}_{i=1}^n$, l'objectif est d'estimer la distribution *a posteriori*

$$p(\theta | y),$$

où $y = \{y_i, x_i\}_{i=1}^n$ désigne l'ensemble des données observées, supposées être des réalisations indépendantes issues de la distribution jointe sous-jacente $(\mathcal{Y}, \mathcal{X})$. Cette distribution permet d'effectuer des inférences sur de nouvelles données en intégrant l'incertitude liée aux paramètres, selon la relation suivante :

$$p(y_{n+1} | x_{n+1}, y) = \int p(y | x_{n+1}, \theta) p(\theta | y) d\theta.$$

L'application du théorème de Bayes conduit à l'expression suivante :

$$p(\theta | y) = \frac{p(y | \theta) p(\theta)}{p(y)},$$

où $p(y | \theta)$ représente la vraisemblance du modèle et l'évidence $p(y)$, aussi appelée vraisemblance marginale, est obtenue par marginalisation,

$$p(y) = \int p(y | \theta) p(\theta) d\theta.$$

3.2 Régression linéaire bayésienne

Un exemple classique est celui de la régression linéaire bayésienne sous l'hypothèse de normalité des résidus.

Un ensemble de n observations $y = \{(x_i, y_i)\}_{i=1}^n$ est considéré, avec $x_i \in \mathbb{R}^d$ et $y_i \in \mathbb{R}$, généré selon le modèle de régression linéaire

$$y_i = x_i^T \theta + \varepsilon_i,$$

avec $\theta \in \mathbb{R}^d$ le vecteur de paramètres et les $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$ sont des bruits gaussiens indépendants et identiquement distribués. Afin de simplifier les calculs, la variance σ^2 est supposée connue dans cet exemple. Dans le cadre classique, un prior lui est attribué selon une loi inverse-gamma.

Les hypothèses bayésiennes sont les suivantes :

- Une loi a priori gaussienne sur θ :

$$p(\theta) = \mathcal{N}(\theta | \mu_\theta, \Sigma_\theta),$$

où μ_θ et Σ_θ sont les paramètres de la distribution a priori.

- Une vraisemblance gaussienne conditionnellement à θ :

$$p(y \mid \theta) = \prod_{i=1}^n \mathcal{N}(y_i \mid x_i^\top \theta, \sigma^2).$$

La relation suivante, décrite précédemment, doit être utilisée pour pouvoir inférer y_{n+1} à partir d'une nouvelle entrée x_{n+1} :

$$p(y_{n+1} \mid x_{n+1}, y) = \int p(y_{n+1} \mid x_{n+1}, \theta) p(\theta \mid y) d\theta.$$

La loi a posteriori est calculée par le théorème de Bayes :

$$p(\theta \mid y) = \frac{p(y \mid \theta) p(\theta)}{p(y)},$$

La loi a posteriori est alors proportionnelle à la quantité suivante,

$$p(\theta \mid y) \propto p(y \mid \theta) p(\theta) \propto \exp \left[-\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - x_i^\top \theta)^2 - \frac{1}{2} (\theta - \mu_\theta)^\top \Sigma_\theta^{-1} (\theta - \mu_\theta) \right].$$

En introduisant la matrice X de dimension $n \times d$ dont la i -ème ligne est x_i^\top et le vecteur $Y = (y_1, \dots, y_n)^\top$, il vient :

$$\sum_{i=1}^n (y_i - x_i^\top \theta)^2 = \|Y - X\theta\|^2.$$

La densité devient alors :

$$p(\theta \mid y) \propto \exp \left[-\frac{1}{2} \left\{ \frac{1}{\sigma^2} (Y - X\theta)^\top (Y - X\theta) + (\theta - \mu_\theta)^\top \Sigma_\theta^{-1} (\theta - \mu_\theta) \right\} \right].$$

L'objectif est de réécrire l'exposant sous la forme d'un carré complété en θ , c'est-à-dire sous la forme :

$$-\frac{1}{2} (\theta - \mu)^\top \Sigma (\theta - \mu) + \text{constante},$$

Pour ce faire, les formes quadratiques sont développées séparément.

La partie liée au prior s'écrit,

$$(\theta - \mu_\theta)^\top \Sigma_\theta^{-1} (\theta - \mu_\theta) = \theta^\top \Sigma_\theta^{-1} \theta - 2\mu_\theta^\top \Sigma_\theta^{-1} \theta + \mu_\theta^\top \Sigma_\theta^{-1} \mu_\theta.$$

La partie liée à la vraisemblance s'écrit,

$$\frac{1}{\sigma^2} (Y - X\theta)^\top (Y - X\theta) = \frac{1}{\sigma^2} (Y^\top Y - 2Y^\top X\theta + \theta^\top X^\top X\theta).$$

En combinant les deux développements, l'exposant devient :

$$-\frac{1}{2} \left\{ \theta^\top \left(\frac{1}{\sigma^2} X^\top X + \Sigma_\theta^{-1} \right) \theta - 2\theta^\top \left(\frac{1}{\sigma^2} X^\top Y + \Sigma_\theta^{-1} \mu_\theta \right) + \frac{1}{\sigma^2} Y^\top Y + \mu_\theta^\top \Sigma_\theta^{-1} \mu_\theta \right\}.$$

Pour simplifier la notation, il est défini :

$$A = \frac{1}{\sigma^2} X^\top X + \Sigma_\theta^{-1}, \quad b = \frac{1}{\sigma^2} X^\top Y + \Sigma_\theta^{-1} \mu_\theta, \quad \text{et} \quad c = \frac{1}{\sigma^2} Y^\top Y + \mu_\theta^\top \Sigma_\theta^{-1} \mu_\theta.$$

L'exposant s'écrit alors :

$$-\frac{1}{2} \{ \theta^\top A \theta - 2\theta^\top b + c \}.$$

A est symétrique et définie positive par construction alors,

$$\theta^\top A \theta - 2\theta^\top b = (\theta - \mu)^\top A (\theta - \mu) - \mu^\top A \mu,$$

avec un vecteur moyen μ donné par

$$\mu = A^{-1}b = \left(\frac{1}{\sigma^2} X^\top X + \Sigma_\theta^{-1} \right)^{-1} \left(\frac{1}{\sigma^2} X^\top Y + \Sigma_\theta^{-1} \mu_\theta \right).$$

En remplaçant dans l'exposant, il est obtenu :

$$-\frac{1}{2} \{ (\theta - \mu)^\top A (\theta - \mu) - \mu^\top A \mu + c \}.$$

Les termes qui ne dépendent pas de θ , à savoir $-\mu^\top A \mu + c$, peuvent être absorbés dans la constante de normalisation. Il en résulte que la densité a posteriori est donnée par,

$$p(\theta | y) \propto \exp \left[-\frac{1}{2} (\theta - \mu)^\top A (\theta - \mu) \right].$$

L'expression obtenue correspond à la densité d'une loi normale multivariée de vecteur de moyenne μ et de matrice de covariance $\Sigma = A^{-1}$, c'est-à-dire :

$$\theta | y \sim \mathcal{N}(\mu, \Sigma) \quad \text{où} \quad \Sigma = \left(\frac{1}{\sigma^2} X^\top X + \Sigma_\theta^{-1} \right)^{-1}.$$

Pour prédire une nouvelle observation y_{n+1} associée à l'entrée x_{n+1} , on utilise la distribution prédictive :

$$p(y_{n+1} | x_{n+1}, y) = \int p(y_{n+1} | x_{n+1}, \theta) p(\theta | y) d\theta,$$

Le terme $p(y_{n+1} | x_{n+1}, \theta)$ s'écrit comme suit,

$$p(y_{n+1} | x_{n+1}, \theta) = \mathcal{N}(x_{n+1}^\top \theta, \sigma^2)$$

L'objectif est de montrer que le résultat de cette intégrale revient à la convolution de deux lois gaussiennes. La première étape consiste à effectuer le changement de variable $z = x_{n+1}^\top \theta$. Une base orthonormée $\{v_i\}_{i=1}^d$ est construite tel que son premier vecteur v_1

est parallèle à x_{n+1} , $v_1 = \frac{x_{n+1}}{\|x_{n+1}\|}$, et les suivants sont orthogonaux à ce même vecteur. Le vecteur des paramètres θ admet une décomposition unique dans cette base.

$$\theta = \sum_{i=1}^d u_i v_i, \quad u_i = v_i^\top \theta, \quad u_1 = \frac{z}{\|x_{n+1}\|}$$

La matrice de rotation associée au changement de base entre la base canonique et la base $\{v_i\}_{i=1}^d$ est notée V . Il est défini

$$\tilde{\mu} = V^\top \mu, \quad \tilde{\Sigma} = V^\top \Sigma V$$

Ce changement de base conduit à,

$$\begin{aligned} \mathcal{N}(\theta \mid \mu, \Sigma) &= \mathcal{N}\left(\sum_{i=1}^d u_i v_i \mid \mu, \Sigma\right) \\ &= \mathcal{N}(u \mid \tilde{\mu}, \tilde{\Sigma}) \end{aligned}$$

Le changement de variable entre θ et u est linéaire et orthonormé donc le déterminant de la jacobienne est de 1. Ainsi, avec ce changement de variable et en utilisant le théorème de Fubini il vient,

$$\begin{aligned} p(y_{n+1} \mid x_{n+1}, y) &= \int \dots \int \mathcal{N}(y_{n+1} \mid u_1 \|x_{n+1}\|, \sigma^2) \cdot \mathcal{N}(u \mid \tilde{\mu}, \tilde{\Sigma}) du_1 \dots du_d \\ &= \int \mathcal{N}(y_{n+1} \mid u_1 \|x_{n+1}\|, \sigma^2) \left(\int \dots \int \mathcal{N}(u \mid \tilde{\mu}, \tilde{\Sigma}) du_2 \dots du_d \right) du_1 \end{aligned}$$

Par marginalisation de la loi normale, il vient,

$$\begin{aligned} p(y_{n+1} \mid x_{n+1}, y) &= \int \mathcal{N}(y_{n+1} \mid u_1 \|x_{n+1}\|, \sigma^2) \mathcal{N}(u_1 \mid \tilde{\mu}_1, \tilde{\Sigma}_{11}) du_1 \\ &= \int \mathcal{N}(y_{n+1} \mid z, \sigma^2) \mathcal{N}(z \mid \|x_{n+1}\| \tilde{\mu}_1, \|x_{n+1}\|^2 \tilde{\Sigma}_{11}) dz \\ &= \frac{1}{\sqrt{4\pi^2 \sigma^2 \|x_{n+1}\|^2 \tilde{\Sigma}_{11}}} \int \exp \left[-\frac{(y_{n+1} - z)^2}{2\sigma^2} \right] \exp \left[-\frac{(z - \|x_{n+1}\| \tilde{\mu}_1)^2}{2\|x_{n+1}\|^2 \tilde{\Sigma}_{11}} \right] dz \\ &= \frac{1}{\sqrt{4\pi^2 \sigma^2 \|x_{n+1}\|^2 \tilde{\Sigma}_{11}}} \int \exp \left[-\frac{(y_{n+1} - z)^2}{2\sigma^2} - \frac{(z - \|x_{n+1}\| \tilde{\mu}_1)^2}{2\|x_{n+1}\|^2 \tilde{\Sigma}_{11}} \right] dz \end{aligned}$$

Il reste à développer les termes quadratiques et compléter le carré comme précédemment. Ainsi, il vient finalement,

$$p(y_{n+1} \mid x_{n+1}, y) = \mathcal{N}\left(y_{n+1} \mid \|x_{n+1}\| \tilde{\mu}_1, \sigma^2 + \|x_{n+1}\|^2 \tilde{\Sigma}_{11}\right).$$

Cet exemple illustre un cas particulier de familles conjuguées, où le prior gaussien et la vraisemblance gaussienne assurent un postérieur également gaussien. Cela montre qu'une inférence bayésienne exacte est réalisable dans ce cadre restreint. Toutefois, même dans ce cas simple, le calcul explicite de $p(y)$ a été évité. Dès lors, on comprend que l'inférence bayésienne exacte, hors de telles familles conjuguées, devient généralement impossible ou très complexe à mener sans recourir à des approches approximatives.

3.3 Problème de l'inférence exacte

L'inférence bayésienne repose sur l'application du théorème de Bayes, qui permet d'exprimer la distribution a posteriori des paramètres du modèle sous la forme :

$$p(\theta | y) = \frac{p(y | \theta)p(\theta)}{p(y)}.$$

Dans cette expression, le terme l'évidence joue un rôle crucial car, il normalise la distribution a posteriori. Toutefois, dans la plupart des modèles bayésiens complexes, son calcul analytique est impossible. Ainsi, la normalisation explicite de la loi a posteriori est généralement impossible.

Exemple de modèle où l'inférence exacte est impossible : mélange de gaussien

L'impossibilité de calculer l'évidence peut être illustrée par un modèle de mélange gaussien bayésien (Bayesian Gaussian Mixture Model, BGMM) (Blei et al., 2017). Un mélange bayésien de Gaussiennes univariées de variance unitaire est considéré, avec K composantes de mélange correspondant à K distributions gaussiennes de moyennes $\mu = \{\mu_1, \dots, \mu_K\}$. Les paramètres de moyenne sont supposés être tirés indépendamment d'une loi a priori commune $p(\mu_k)$, choisie comme une distribution gaussienne $\mathcal{N}(0, \sigma^2)$, où σ^2 est un hyperparamètre. La génération d'une observation x_i dans ce modèle suit plusieurs étapes. Tout d'abord, une attribution de cluster c_i est déterminée, indiquant à quel cluster latent appartient x_i . Cette attribution est représentée par un vecteur indicateur K -dimensionnel, composé uniquement de zéros sauf à la position correspondant au cluster associé. Ensuite, l'observation x_i est tirée dans la gaussienne correspondante $\mathcal{N}(c_i^\top \mu, 1)$.

Le modèle hiérarchique complet est défini par :

$$\mu_k \sim \mathcal{N}(0, \sigma^2), \quad k = 1, \dots, K$$

$$c_i \sim \text{Categorical}(1/K, \dots, 1/K), \quad i = 1, \dots, n$$

$$x_i | c_i, \mu \sim \mathcal{N}(c_i^\top \mu, 1), \quad i = 1, \dots, n$$

Pour un échantillon de taille n , la densité conjointe des variables latentes et observées s'écrit sous la forme :

$$p(\mu, c, x) = p(\mu) \prod_{i=1}^n p(c_i) p(x_i | c_i, \mu).$$

Les variables latentes du modèle sont $\theta = \{\mu, c\}$, comprenant les K moyennes de classe et les n attributions de classe. L'évidence est définie par l'intégrale suivante :

$$p(x) = \int p(\mu) \prod_{i=1}^n \sum_{c_i} p(c_i) p(x_i | c_i, \mu) d\mu.$$

L'analyse de cette équation montre que l'intégrande ne se factorise pas en termes indépendants pour chaque μ_k , car chaque μ_k apparaît dans les n termes du produit. Ainsi, cette intégrale ne se réduit pas à un produit d'intégrales unidimensionnelles sur les μ_k ,

ce qui complique son calcul. L'évaluation numérique de cette intégrale K -dimensionnelle présente une complexité temporelle de l'ordre de $O(K^n)$.

En développant le produit sur la somme et en réorganisant les termes, il est possible d'exprimer l'évidence comme une somme sur toutes les configurations possibles d'assignations de clusters :

$$p(x) = \sum_c p(c) \int p(\mu) \prod_{i=1}^n p(x_i | c_i, \mu) d\mu.$$

Chaque intégrale individuelle est calculable grâce à la conjugaison entre la loi a priori gaussienne sur les composantes et la vraisemblance gaussienne. Toutefois, le nombre de configurations possibles d'assignations de clusters est de K^n , entraînant une explosion combinatoire rendant le calcul de l'évidence exponentiel en K et donc incalculable en pratique.

3.4 Méthodes de Monte-Carlo par chaînes de Markov

L'impossibilité de calculer exactement $p(y)$ rend l'inférence exacte inapplicable dans la plupart des cas pratiques. Pour pallier cette difficulté, les méthodes de Monte Carlo par chaînes de Markov (MCMC) (Robert et al., 1999) ont été développées. Le principe général consiste à générer une suite de variables aléatoires $\{\theta^{(n)}\}_{n=1}^N$ formant une chaîne de Markov dont la distribution stationnaire est la loi $\pi(\theta) = p(\theta | y)$. Ainsi, pour tout estimateur fonction de θ , on peut approximer l'espérance a posteriori par une moyenne empirique :

$$\mathbb{E}_{p(\theta|y)}[f(\theta)] = \int f(\theta) p(\theta | y) d\theta \approx \frac{1}{N} \sum_{n=1}^N f(\theta^{(n)}),$$

où $\theta^{(n)}$ est le n -ième échantillon de la chaîne. Quelques méthodes sont présentées par la suite à titre d'exemples.

La méthode de Metropolis–Hastings (Metropolis et al., 1953; Hastings, 1970) repose sur la définition d'une distribution de proposition $q(\theta' | \theta)$. À chaque itération, on propose un nouvel échantillon θ' à partir de $q(\theta' | \theta^{(n)})$, puis on l'accepte avec une probabilité d'acceptation $\alpha(\theta^{(n)}, \theta')$:

$$\alpha(\theta^{(n)}, \theta') = \min \left\{ 1, \frac{\pi(\theta') q(\theta^{(n)} | \theta')}{\pi(\theta^{(n)}) q(\theta' | \theta^{(n)})} \right\},$$

où $\pi(\theta) = p(\theta | y)$ est la densité cible (le posterior). Si l'échantillon θ' est accepté, on pose $\theta^{(n+1)} = \theta'$; sinon, on conserve l'ancien $\theta^{(n+1)} = \theta^{(n)}$.

La méthode de Gibbs (Gelfand, 2000) est un cas particulier de Metropolis–Hastings où la distribution de proposition est choisie comme la loi conditionnelle complète de chaque paramètre. Dans sa forme la plus simple, on balaye séquentiellement chaque composante du vecteur θ pour en générer un nouvel échantillon à partir de la loi conditionnelle correspondante. Ceci élimine le besoin de calculer un ratio d'acceptation explicite, car la proposition est toujours « parfaitement » alignée sur la densité cible pour la composante

mise à jour. Cette simplicité de mise en œuvre fait de la méthode de Gibbs un outil très populaire, notamment en modélisation hiérarchique ou dans certains cas où les conditionnelles sont accessibles analytiquement.

Parmi les améliorations majeures visant à accélérer ou stabiliser la convergence, on peut citer Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Girolami and Calderhead, 2011) qui exploite la dynamique hamiltonienne pour parcourir l'espace des paramètres plus efficacement et les méthodes dites stochastiques (e.g. SGLD SGHMC) (Chen et al., 2014; Welling and Teh, 2011), adaptées aux grandes masses de données par des techniques d'échantillonnage par mini-lots.

Malgré leur efficacité, ces méthodes sont coûteuses et le nombre d'échantillons requis pour explorer de façon satisfaisante la distribution croît souvent fortement avec la dimension du problème, phénomène dit de « malédiction de la dimension ». Dans le cadre de réseaux de neurones profonds ou de problèmes à très haute dimension, il peut alors devenir prohibitif d'obtenir un échantillonnage fiable. De plus, de nombreuses passes (burn-in, thinning, etc.) sont parfois nécessaires pour garantir la convergence à la distribution stationnaire, ce qui accroît encore le coût de calcul. Face à ces limitations, l'inférence variationnelle a émergé comme une alternative plus rapide à mettre en œuvre, particulièrement lorsque la dimension du paramètre à estimer est élevée ou quand l'on recherche une estimation en ligne. L'idée est d'approximer la distribution a posteriori par une famille de distributions paramétriques $q(\theta)$, choisie pour être à la fois proche de la distribution réelle et manipulable analytiquement. Cette approche contourne la difficulté du calcul explicite de $p(y)$ et fournit une approximation raisonnable de la loi a posteriori dans de nombreux contextes, notamment en apprentissage profond. Nous reviendrons plus en détail sur ces techniques dans la section suivante.

4 Inférence Variationnelle

4.1 Principe

L'inférence variationnelle reformule le problème de l'inférence comme un problème d'optimisation. Elle repose sur l'approximation de la distribution a posteriori $p(\theta | y)$ par une distribution q appartenant à une famille de distributions calculables \mathcal{Q} (Jordan et al., 1999; Blei et al., 2017; Tran et al., 2021). Une notion de distance entre les distributions est nécessaire pour établir une relation d'ordre relative à l'écart entre la distribution cible p et q . La divergence de Kullback-Leibler est la mesure de similarité entre les distributions p et q . Bien qu'elle soit parfois interprétée comme une distance, elle n'en vérifie pas la définition : elle n'est pas symétrique et ne respecte pas l'inégalité triangulaire. Cependant, elle vérifie la positivité, la séparation et l'additivité. La positivité s'obtient via l'inégalité de Jensen.

Ainsi, on peut définir que notre problématique est de trouver le minimum de la divergence de Kullback-Leibler (KL) entre q et $p(\theta | y)$, définie par :

$$\text{KL}(q||p(\cdot | y)) = \int q(\theta) \log \frac{q(\theta)}{p(\theta | y)} d\theta.$$

La distribution optimale q^* est obtenue en résolvant le problème d'optimisation suivant :

$$q^* = \arg \min_q \text{KL}(q||p(\cdot | y)).$$

Cependant, KL ne peut généralement pas être calculée analytiquement. Il faut donc établir une borne inférieure sur laquelle l'optimisation est possible. L'expression de la divergence KL peut être réécrite en exploitant la définition de la loi :

$$\text{KL}(q||p(\cdot | y)) = \int q(\theta) \log \frac{q(\theta)}{p(y, \theta)} d\theta + \log p(y).$$

En réarrangeant les termes, l'identité fondamentale suivante est obtenue :

$$\log p(y) = \text{KL}(q||p(\cdot | y)) + \underbrace{\mathbb{E}_q[\log p(y, \theta) - \log q(\theta)]}_{\mathcal{L}(q)}. \quad (1)$$

Le terme $\mathcal{L}(q)$ est appelé Evidence Lower Bound (ELBO). La divergence KL étant positive, la minimisation de la divergence KL est équivalente à la maximisation de cette borne inférieure :

$$q^* = \arg \max_q \mathcal{L}(q).$$

L'intuition derrière l'optimisation de l'ELBO est explorée en réécrivant cette quantité comme la somme de l'espérance de la vraisemblance des données et de la divergence de Kullback-Leibler entre le prior p et la distribution variationnelle q :

$$\mathcal{L}(q) = \mathbb{E}_q[\log p(y, \theta)] - \text{KL}(q||p)$$

L'optimisation de cet objectif incite la distribution $q(\theta)$ à concentrer sa masse sur certaines valeurs de θ . Le premier terme, correspondant à une espérance de vraisemblance,

favorise les densités qui attribuent une probabilité élevée aux configurations des variables latentes expliquant les observations. Le second terme, correspondant à la divergence négative entre la densité variationnelle et le prior, encourage une proximité avec le prior. Ainsi, l'objectif variationnel reflète l'équilibre classique entre la vraisemblance et le prior.

4.2 Famille Variationnelle

Une stratégie courante en inférence variationnelle consiste à restreindre la famille des distributions admissibles \mathcal{Q} à une classe paramétrique spécifique $\{q_\lambda \mid \lambda \in E\}$, où $E \subset \mathbb{R}^d$ pour un certain $d \in \mathbb{N}^*$. Cette formulation permet de représenter chaque distribution q_λ au sein de cette famille par un vecteur de paramètres λ , appelés paramètres variationnels. Cette approche est connue sous le nom d'approximation variationnelle à forme fixe.

Cette approche permet une grande expressivité pour la famille variationnelle, mais elle est coûteuse. Une autre approche répandue est l'approximation par champs moyens. Elle consiste à poser une famille variationnelle qui est factorisable selon ses variables latentes,

$$q_\lambda(\theta) = \prod_{i=1}^d q_i(\theta_i, \lambda_i).$$

Ce choix réduit grandement l'expressivité, en effet cela introduit une hypothèse d'indépendance conditionnelle entre les variables latentes. Cependant, cela permet de bénéficier de la structure imposée pour réduire le coût de calcul. Cette approche ne sera pas développée ici (Blei et al., 2017).

Il est possible de supposer que l'application $\lambda \mapsto q_\lambda$ est injective sans perte de généralité. En effet, si certains paramètres distincts $\lambda_1 \neq \lambda_2$ définissaient la même distribution, on pourrait restreindre l'espace E pour éliminer cette redondance et obtenir ainsi l'injectivité souhaitée. Il suffit à présent de restreindre cette application à son image pour obtenir que cette application est bijective.

Dans ce cadre, il est possible d'assimiler la fonction ELBO, initialement définie sur l'espace des distributions :

$$\mathcal{L} : q \mapsto \mathbb{E}_q [\log p(y, \theta) - \log q(\theta)],$$

à une fonction définie sur l'espace des paramètres :

$$\mathcal{L} : \lambda \mapsto \mathbb{E}_{q_\lambda} [\log p(y, \theta) - \log q_\lambda(\theta)].$$

Par la suite, sauf mention contraire, il sera noté $\mathcal{L}(\lambda)$ pour désigner l'ELBO en fonction des paramètres variationnels λ , ce qui permet d'exprimer directement l'optimisation en termes de paramètres plutôt que de distributions.

4.3 Optimisation

Dans les cas complexes, l'optimisation de l'ELBO repose souvent sur la méthode de descente de gradient. Le pseudo-code de l'algorithme est présenté ci-dessous.

Algorithm 1 Optimisation variationnelle par descente de gradient

Require: Fonction objectif $\mathcal{L}(\lambda)$, taux d'apprentissage η_t , nombre d'itérations T

- 1: Initialisation des paramètres $\lambda_0 \in E$
 - 2: **for** $t = 0, 1, \dots, T - 1$ **do**
 - 3: Calcul du gradient : $g_t = \nabla_{\lambda} \mathcal{L}(\lambda_t)$
 - 4: Mise à jour des paramètres : $\lambda_{t+1} = \lambda_t + \eta_t g_t$
 - 5: **end for**
 - 6: **return** λ_T
-

La mise à jour classique des paramètres s'écrit :

$$\lambda_{t+1} = \lambda_t + \eta_t \nabla_{\lambda} \mathcal{L}(\lambda_t),$$

où η_t représente le taux d'apprentissage. Cet algorithme (Boyd and Vandenberghe, 2004) est conçu pour les problèmes où la fonction objectif est convexe et différentiable. Dans ce contexte, en choisissant une séquence de η_t suivant la recherche linéaire exacte ou par backtracking, la convergence vers l'optimal global est garantie. Ces séquences correspondent à des stratégies adaptatives de sélection du pas d'apprentissage qui assurent une descente efficace et stable de la fonction objectif, en équilibrant vitesse de convergence et précision. L'ordre de convergence dépend du conditionnement de la Hessienne de la fonction objectif autour du minimum.

Pour une majorité de problèmes, la fonction objectif, l'ELBO, n'est pas convexe par rapport aux paramètres. Ainsi, il n'y a pas de garantie de convergence vers le minimum global de la fonction. Cependant, en pratique, cet algorithme donne des résultats satisfaisants en un temps raisonnable.

Critère d'arrêt

Jusqu'à présent, la condition d'arrêt n'a pas été discutée. En effet, l'algorithme itère jusqu'à l'étape T , mais aucune méthode pour calculer T n'a été vue. L'idée est que, à la convergence, $\mathcal{L}(\lambda_t)$ est approximativement le même. L'idée est donc de stopper l'algorithme si cette valeur ne s'améliore pas, c'est-à-dire n'augmente pas, après P itérations (Tran et al., 2021). P est appelé le paramètre de patience. Seulement, l'approximation de cette quantité,

$$\widehat{\mathcal{L}(\lambda_t)} = \frac{1}{S} \sum_{i=1}^S \log \frac{p(y, \theta)}{q_{\lambda}(\theta)}, \quad \theta_i \sim q_{\lambda},$$

est trop bruitée pour cela. Ainsi, la quantité utilisée pour le critère est la moyenne mobile de l'estimateur précédent, c'est-à-dire,

$$\overline{\mathcal{L}(\lambda_t)} = \frac{1}{t_W} \sum_{k=1}^{t_W} \widehat{\mathcal{L}(\lambda_{t-k+1})},$$

avec t_W le nombre d'échantillons utilisés pour la moyenne mobile. Ainsi, l'algorithme s'arrête après P itération où la moyenne mobile ne diminue pas.

Taux d'apprentissage

Il est noté que les méthodes dites de recherche linéaire exacte ou par backtracking, évoquées précédemment, sont coûteuses et inefficaces dans un contexte non convexe. En pratique, une diminution progressive de ce taux est nécessaire afin que les mises à jour des paramètres deviennent de plus en plus petites. Cette approche permet d'abord une phase exploratoire facilitant l'échappement aux minima locaux peu profonds, suivie d'une phase d'exploitation visant à stabiliser la solution. Diverses stratégies de mise à jour existent et la stratégie optimale dépend du problème (Smith, 2017; You et al., 2019; d'Ascoli et al., 2022).

De plus, toutes les coordonnées du vecteur λ n'ont pas besoin du même taux d'apprentissage. Il faut donc employer des stratégies de taux d'apprentissages adaptatifs (Zaheer et al., 2018). Les méthodes suivantes peuvent être citées, ADAM (Kingma and Ba, 2014) et AdaGrad (Duchi et al., 2011). Ces méthodes consistent à normaliser le gradient par la variance de la coordonnée en question, qui est estimée par moyenne mobile.

Gradient naturel

Il a été supposé qu'une faible distance euclidienne entre λ et λ' implique une faible différence entre q_λ et $q_{\lambda'}$. En effet, la descente de gradient consiste à effectuer de "petits" pas suivis de réévaluations successives, afin d'explorer l'espace de façon progressive. Or, plutôt que de réaliser cette descente directement dans l'espace des distributions q_λ , on l'a remplacée par une descente dans l'espace paramétrique λ , en s'appuyant sur l'idée qu'il existe une bijection entre λ et q_λ . Ainsi, on a implicitement supposé que la notion de "petit" déplacement est préservée lors de ce changement de variable. Plus précisément, il a été supposé que l'espace des paramètres et l'espace des distributions paramétrés sont suffisamment isomorphes localement. Cependant, cela n'est pas garanti : une petite distance euclidienne dans l'espace des paramètres peut correspondre au parcours d'une grande distance dans l'espace des distributions. En effet, l'ensemble des distributions $\{q_\lambda\}$ n'est pas un espace euclidien : c'est une variété différentielle riemannienne (Amari and Nagaoka, 2000), et cette géométrie doit être prise en compte pour décrire plus fidèlement la notion de proximité entre distributions.

Une variété riemannienne (\mathcal{M}, g) est une variété différentielle \mathcal{M} munie d'un tenseur métrique g qui définit une notion de distance locale entre les points. Dans notre cas, l'espace des distributions paramétriques $\{q_\lambda\}$ peut être muni d'une métrique de Fisher-Rao, qui est donnée par la matrice d'information de Fisher :

$$g_\lambda = \mathbb{E}_{q_\lambda} [\nabla_\lambda \log q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta)^\top].$$

Cette métrique encode la courbure locale de l'espace des distributions et permet d'ajuster la direction du gradient pour mieux respecter la structure géométrique sous-jacente. Ainsi, pour prendre en compte cette structure géométrique, la métrique de l'espace, que l'on peut aussi noter $I_F = \text{Cov}_{q_\lambda}(\nabla_\lambda \log q_\lambda(\theta))$, est utilisée pour adapter la direction du gradient :

$$\lambda_{t+1} = \lambda_t - \eta I_F^{-1}(\lambda) \widehat{\nabla_\lambda \mathcal{L}(\lambda_t)}.$$

Cette approche, appelée gradient naturel, améliore la convergence en suivant mieux la géométrie de la distribution variationnelle. Cette méthode peut être vue comme une méthode d'optimisation de second ordre avec la matrice d'information de Fisher qui agit comme un substitut à la Hessienne (Martens, 2020). En grande dimension, le coût de l'inversion de la matrice d'information de Fisher pose problème. En effet, l'inversion via une décomposition LU est en $O(N^3)$, avec $N = |\lambda| = d$ la dimension de la matrice (Datta, 2010). La matrice étant symétrique définie positive, sa structure peut être exploitée avec la décomposition de Cholesky. Bien que cette méthode soit plus rapide et plus stable, le coût asymptotique de son inversion est en $O(N^3)$. Cependant, le but n'est pas de calculer $I_F^{-1}(\lambda)$, mais $I_F^{-1}(\lambda) \nabla_{\lambda} \widehat{\mathcal{L}}(q_{\lambda_t})$. Ce calcul peut être reformulé en le problème suivant :

$$\text{Trouver } x \in \mathbb{R}^d \text{ tel que, } I_F(\lambda) \cdot x = \nabla_{\lambda} \widehat{\mathcal{L}}(\lambda_t).$$

Soit la résolution d'un système linéaire $Ax = b$. Seulement, inverser ce système pour résoudre ce problème pour une seule valeur de b est inefficace. Une méthode efficace pour résoudre ce type de système linéaire est la méthode du gradient conjugué (Nazareth, 2009). Il s'agit d'une méthode itérative permettant de converger en au plus $O(N)$ étapes et en général le nombre d'itérations est proportionnel à $\sqrt{\kappa}$ avec $\kappa = \frac{|\lambda_{max}|}{|\lambda_{min}|}$ le conditionnement de la matrice. Cette méthode amène à une complexité en $O(N^2)$. Il peut être noté que cet algorithme ne nécessite que des produits matrice-vecteur $A \cdot v$ ce qui peut permettre de ne pas calculer entièrement la matrice d'information de Fisher et de ne la stocker qu'implicitement.

Une méthode moderne adaptée aux familles variationnelles construites à l'aide de réseaux de neurones est K-FAC (Kronecker-Factored Approximate Curvature) (Martens and Grosse, 2015; Eschenhagen et al., 2023). Le principe de cette méthode est d'approximer la matrice d'information de Fisher en exploitant la structure spécifique des réseaux de neurones. Plus précisément, K-FAC considère la matrice de Fisher comme une matrice bloc-diagonale, chaque bloc correspondant à une couche du réseau. Chaque bloc est ensuite approximé par le produit de Kronecker de deux matrices de plus petite taille, représentant les covariances des activations et des gradients de la couche concernée. Cette factorisation permet une inversion plus efficace de la matrice de Fisher approximée, réduisant ainsi le coût computationnel associé aux méthodes de gradient naturel.

De façon générale, il existe une littérature riche sur le calcul efficace de cette descente de gradient naturel (Tran et al., 2021; 2020; Martens and Grosse, 2015; Ong et al., 2018).

Gradient de l'espérance

L'ELBO est une espérance. Ainsi, pour la majeure partie des familles paramétriques complexes, la forme analytique de l'espérance n'est pas connue et donc il n'est pas possible de calculer directement son gradient. L'objectif est donc d'inverser le gradient et l'intégrale potentiellement généralisée.

Les hypothèses permettant cette inversion reposent sur le théorème de différentiation sous le signe intégral de Lebesgue. Plus précisément, il est noté $(\theta, \lambda) \mapsto f(\theta, \lambda)$, alors sous les conditions suivantes :

- Continuité : $\lambda \mapsto f(\theta, \lambda)$ est continue pour $\theta \in \Theta$.
- Différentiabilité : $\lambda \mapsto f(\theta, \lambda)$ est différentiable presque partout pour $\theta \in \Theta$.
- Dominance intégrable : Il existe une fonction intégrable $\theta \mapsto g(\theta)$ indépendante de λ telle que :

$$|\nabla_\lambda f(\theta, \lambda)| \leq g(\theta), \quad \forall \lambda \in \Lambda.$$

alors l'opérateur de gradient peut être échangé avec l'intégrale :

$$\nabla_\lambda \int_{\Theta} f(\theta, \lambda) d\theta = \int_{\Theta} \nabla_\lambda f(\theta, \lambda) d\theta.$$

Pour l'ELBO la continuité et la différentiabilité sont vérifiées. Pour la dominance intégrale la vérification de l'hypothèse dépend à la fois de la distribution cible et variationnelle. L'hypothèse de domination doit donc être vérifié au cas par cas. Ainsi, il est obtenu par application du théorème,

$$\begin{aligned} \nabla_\lambda \mathcal{L}(\lambda) &= \nabla_\lambda \int q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)] d\theta \\ &= \int \nabla_\lambda \{q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)]\} d\theta \\ &= \int \nabla_\lambda q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)] - q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta) d\theta \\ &= \int \nabla_\lambda q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)] d\theta - \int q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta) d\theta. \end{aligned}$$

En remarquant que,

$$\forall \theta \in \mathbb{R}, \quad q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta) = q_\lambda(\theta) \frac{\nabla_\lambda q_\lambda(\theta)}{q_\lambda(\theta)} = \nabla_\lambda q_\lambda(\theta)$$

Il vient alors,

$$\begin{aligned} \nabla_\lambda \mathcal{L}(\lambda) &= \int q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)] d\theta - \int \nabla_\lambda q_\lambda(\theta) d\theta \\ &= \int q_\lambda(\theta) \nabla_\lambda \log q_\lambda(\theta) \cdot [\log p(y, \theta) - \log q_\lambda(\theta)] d\theta - \underbrace{\nabla_\lambda \int q_\lambda(\theta) d\theta}_{=1} \\ &= \mathbb{E}_{q_\lambda} \left[\nabla_\lambda \log q_\lambda(\theta) \cdot \log \frac{p(y, \theta)}{q_\lambda(\theta)} \right] \end{aligned}$$

Le gradient sous cette forme est connu sous le nom de “score-function gradient”. Il suffit alors d'échantillonner $\theta \sim q_\lambda(\theta)$ pour construire un estimateur de cette dernière espérance par Monte-Carlo. Il est noté S le nombre d'échantillons générés.

$$\widehat{\nabla_\lambda \mathcal{L}(\lambda)} = \frac{1}{S} \sum_{i=1}^S \nabla_\lambda \log q_\lambda(\theta_i) \cdot \log \frac{p(y, \theta_i)}{q_\lambda(\theta_i)}$$

Pour que l'estimateur soit de bonne qualité, il doit être non biaisé et de variance faible. Il est noté, $X(\theta) = \nabla_{\lambda} \log q_{\lambda}(\theta) \cdot \log \frac{p(y, \theta)}{q_{\lambda}(\theta)}$ tel que, $\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)} = \frac{1}{S} \sum_{i=1}^S X(\theta_i)$. Il vient alors,

$$\begin{aligned} \mathbb{E}_{q_{\lambda}}[\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)}] &= \frac{1}{S} \sum_{i=1}^S \mathbb{E}_{q_{\lambda}}[X(\theta_i)] \\ &= \frac{1}{S} \cdot S \cdot \mathbb{E}_{q_{\lambda}}[X(\theta)], \quad \text{comme les } \theta_i \text{ i.i.d.} \\ &= \mathbb{E}_{q_{\lambda}}[\nabla_{\lambda} \log q_{\lambda}(\theta) \cdot \log \frac{p(y, \theta)}{q_{\lambda}(\theta)}] \end{aligned}$$

L'estimateur est donc non biaisé. Le calcul de la matrice de covariance dans le cas général n'apporte pas d'information supplémentaire. Alors, le calcul ne sera pas fait.

Inférence variationnelle stochastique

Jusqu'à présent, toutes les données y sont utilisées pour évaluer un échantillon du gradient. Ces données peuvent être à la fois nombreuses et de grandes dimensions. Ainsi, cela peut empêcher l'utilisation pratique de cette méthode. Pour pallier ce problème, l'inférence variationnelle stochastique propose d'utiliser un sous-ensemble de données $\{x_j, y_j\}_{j \in R}$ avec $R \subset [1, n]$. En pratique, un sous-ensemble de données est échantillonné pour une itération de la descente de gradient. Ainsi, le sous-ensemble d'indices des données pour l'itération t est noté R_t . L'ancien estimateur est développé,

$$\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)} = \frac{1}{S} \sum_{i=1}^S \nabla_{\lambda} \log q_{\lambda}(\theta_i) \cdot \{\log p(\theta_i) + \log p(y|\theta_i) - \log q_{\lambda}(\theta_i)\}$$

Les données étant supposées indépendantes, il peut être écrit,

$$\log p(y|\theta_i) = \sum_{j=1}^n \log p(x_j, y_j|\theta_i)$$

Un estimateur de cette quantité est,

$$\widehat{\log p(y|\theta_i)} = \frac{n}{|R_t|} \sum_{j \in R_t} \log p(x_j, y_j|\theta_i)$$

Puisque les données sont i.i.d. alors celles du sous-échantillon aussi. En exploitant cela, il vient que l'estimateur est non biaisé,

$$\begin{aligned} \mathbb{E}[\widehat{\log p(y|\theta_i)}] &= \frac{n}{|R_t|} \sum_{j \in R_t} \mathbb{E}[\log p(x_j, y_j|\theta_i)] \\ &= \frac{n}{|R_t|} \cdot |R_t| \cdot \mathbb{E}[\log p(x_1, y_1|\theta_i)] \\ &= \mathbb{E}\left[\sum_{j=1}^n \log p(x_j, y_j|\theta_i)\right] \\ &= \log p(y|\theta_i) \end{aligned}$$

Ainsi, le nouvel estimateur s'écrit,

$$\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)} = \frac{1}{S} \sum_{i=1}^S \nabla_{\lambda} \log q_{\lambda}(\theta_i) \cdot \left\{ \log p(\theta_i) + \frac{n}{|R_t|} \sum_{j \in R_t} \log p(x_j, y_j | \theta_i) - \log q_{\lambda}(\theta_i) \right\}$$

L'algorithme fonctionne alors pour de grands ensembles de données.

Réduction de variance

Dans le cadre de l'optimisation stochastique, les performances de l'algorithme dépendent de la variance de l'estimateur du gradient. L'estimateur naïf qui a été proposé est le suivant.

$$\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)} = \frac{1}{S} \sum_{i=1}^S \nabla_{\lambda} \log q_{\lambda}(\theta_i) \cdot \log \frac{p(y, \theta_i)}{q_{\lambda}(\theta_i)}$$

En pratique, cet estimateur a une variance très importante. C'est pourquoi il est nécessaire d'utiliser des méthodes de réduction de variance.

Variable de contrôle Une première approche est d'utiliser la méthode de la variable de contrôle. Il est noté $X(\theta) = \nabla_{\lambda} \log q_{\lambda}(\theta) \cdot \log \frac{p(y, \theta)}{q_{\lambda}(\theta)}$, la variable aléatoire dont l'espérance est à estimer. L'idée est d'introduire une seconde variable aléatoire, notée Y , qui possède une espérance nulle sous q_{λ} (i.e. $\mathbb{E}_{q_{\lambda}}[Y] = 0$). Pour un échantillon $\{\theta_i\}_{i=1}^S$ tiré selon q_{λ} , on modifie l'estimateur naïf en y soustrayant un terme dit de contrôle :

$$\widehat{\nabla_{\lambda} \mathcal{L}(\lambda)}_{\text{CV}} = \frac{1}{S} \sum_{i=1}^S \left[\nabla_{\lambda} \log q_{\lambda}(\theta_i) \log \frac{p(y, \theta_i)}{q_{\lambda}(\theta_i)} - c \cdot Y(\theta_i) \right].$$

Le coefficient c est choisi de manière à minimiser la variance de l'estimateur. Cet estimateur est bien non biaisé.

$$\mathbb{E}_{q_{\lambda}}[X - c \cdot Y(\theta)] = \mathbb{E}_{q_{\lambda}}[X] - c \cdot \underbrace{\mathbb{E}_{q_{\lambda}}[Y]}_{=0} = \mathbb{E}_{q_{\lambda}}[X]$$

La variance de l'estimateur est la suivante.

$$\mathbb{V}[X - c \cdot Y] = \mathbb{V}[X] - 2c \cdot \text{Cov}[X, Y] + c^2 \cdot \mathbb{V}[Y]$$

Il s'agit d'un polynôme en c qui atteint son minimum en $c = \frac{\text{Cov}(X, Y)}{\mathbb{V}(Y)}$ et dont le minimum vaut,

$$\mathbb{V}[X - c \cdot Y] = \mathbb{V}[X] - 2 \cdot \frac{\text{Cov}(X, Y)^2}{\mathbb{V}[Y]}$$

Le but est donc d'avoir un ratio $\frac{\text{Cov}(X, Y)^2}{\mathbb{V}[Y]}$ important, soit une corrélation entre X et Y relativement élevée à la variance de Y . D'autres méthodes issues de la méthode de Monte-Carlo (e.g. Important Sampling, Quasi Monte Carlo, Rao-Blackwellization) peuvent également être utiles suivant les cas.

Pathwise Gradient Une autre approche plus efficace, moyennant quelques hypothèses, est le “Pathwise Gradient”. Premièrement, cette approche nécessite l’existence d’une reparamétrisation de la loi variationnelle à partir d’une loi qui ne dépend pas des paramètres variationnels. Deuxièmement, le logarithme du ratio de la densité de la probabilité jointe du modèle et de la densité variationnelle doit être dérivable par rapport aux paramètres latents. Formellement,

- Il existe une application g_λ , un C^1 -difféomorphisme, telle que si ε est une variable aléatoire suivant une loi $\pi(\varepsilon)$ indépendante de λ , alors

$$\theta \sim q_\lambda \iff \theta = g_\lambda(\varepsilon) \text{ avec } \varepsilon \sim \pi(\varepsilon).$$

- $\theta \rightarrow \log \frac{p(y, \theta)}{q_\lambda(\theta)} \in D^1$

Par exemple, si θ suit une loi normale $\mathcal{N}(\mu, \sigma^2)$, la reparamétrisation suivante peut être exploitée :

$$\theta = \mu + \sigma \cdot \varepsilon, \text{ avec } \varepsilon \sim \mathcal{N}(0, 1),$$

où μ et σ sont les paramètres variationnels, c’est-à-dire $\lambda = (\mu, \sigma)$.

L’ELBO est re-paramétrisé,

$$\mathcal{L}(\lambda) = \mathbb{E}_{\theta \sim q_\lambda} \left[\log \frac{p(y, \theta)}{q_\lambda(\theta)} \right] = \mathbb{E}_{\varepsilon \sim \pi} \left[\log \frac{p(y, g_\lambda(\varepsilon))}{q_\lambda(g_\lambda(\varepsilon))} \right]$$

La loi ne dépendant plus de λ il vient,

$$\nabla_\lambda \mathcal{L}(\lambda) = \nabla_\lambda \mathbb{E}_{\varepsilon \sim \pi} \left[\log \frac{p(y, g_\lambda(\varepsilon))}{q_\lambda(g_\lambda(\varepsilon))} \right] = \mathbb{E}_{\varepsilon \sim \pi} \left[\nabla_\lambda \left(\log p(y, g_\lambda(\varepsilon)) - \log q_\lambda(g_\lambda(\varepsilon)) \right) \right]$$

D’une part,

$$\nabla_\lambda \log p(y, g_\lambda(\varepsilon)) = \nabla_\lambda g_\lambda(\varepsilon) \cdot \nabla_\theta \log p(y, \theta) \Big|_{\theta=g_\lambda(\varepsilon)}$$

D’autre part,

$$\nabla_\lambda \log q_\lambda(g_\lambda(\varepsilon)) = \nabla_\lambda g_\lambda(\varepsilon) \cdot \nabla_\theta \log q_\lambda(\theta) \Big|_{\theta=g_\lambda(\varepsilon)}$$

Finalement, il vient,

$$\nabla_\lambda \mathcal{L}(\lambda) = \mathbb{E}_{\varepsilon \sim \pi} \left[\nabla_\lambda g_\lambda(\varepsilon) \cdot \left(\nabla_\theta \log p(y, \theta) - \nabla_\theta \log q_\lambda(\theta) \right)_{\theta=g_\lambda(\varepsilon)} \right]$$

Ainsi, l’estimateur s’écrit,

$$\widehat{\nabla_\lambda \mathcal{L}(\lambda)}_{PG} = \frac{1}{S} \sum_{i=1}^S \left[\nabla_\lambda g_\lambda(\varepsilon_i) \cdot \left(\nabla_\theta \log p(y, \theta) - \nabla_\theta \log q_\lambda(\theta) \right)_{\theta=g_\lambda(\varepsilon_i)} \right]$$

En pratique, cette approche nécessite considérablement moins d’échantillons. Il est noté que la variance de cet estimateur n’est pas toujours plus faible, mais qu’il s’agit seulement d’une observation empirique pour les modèles usuels. De plus, les hypothèses restreignent les modèles auxquels cette méthode peut être appliquée.

4.4 Comparaison entre MCMC et Inférence Variationnelle

L'inférence variationnelle et les méthodes de Monte Carlo par Chaîne de Markov sont deux approches courantes pour l'approximation des distributions a posteriori. Chacune présente des avantages et des inconvénients, rendant leur utilisation dépendante du contexte.

Méthodes MCMC

Les méthodes de Monte Carlo par Chaîne de Markov, comme l'algorithme de Metropolis-Hastings ou l'échantillonnage de Gibbs, sont conçues pour produire des échantillons asymptotiquement exacts de la distribution a posteriori $p(\theta | y)$. Elles reposent sur la construction d'une chaîne de Markov dont la distribution stationnaire correspond à cette distribution cible. Lorsqu'un nombre suffisant d'itérations est réalisé, MCMC garantit en théorie un échantillonnage exact, ce qui en fait une approche de référence en inférence bayésienne. Ces méthodes sont particulièrement adaptées aux distributions complexes, notamment lorsque la posterior est multimodale ou présente de fortes corrélations entre les paramètres.

Cependant, la convergence des algorithmes MCMC peut être lente, en particulier en grande dimension ou lorsque les variables du modèle sont fortement dépendantes. De plus, la corrélation entre échantillons successifs implique qu'un grand nombre d'itérations est nécessaire pour obtenir des estimations fiables. Le diagnostic de convergence est une autre difficulté : il est souvent ardu de déterminer si la chaîne a atteint sa distribution stationnaire. Ces limitations rendent les méthodes MCMC coûteuses en calcul et parfois impraticables pour de grands ensembles de données.

Inférence Variationnelle

L'inférence variationnelle adopte une approche différente en reformulant le problème d'inférence bayésienne comme un problème d'optimisation. Plutôt que d'échantillonner directement depuis la distribution a posteriori, on approxime celle-ci par une distribution $q(\theta)$ que l'on ajuste pour minimiser la divergence de Kullback-Leibler $D_{KL}(q||p)$. Cette reformulation permet de remplacer un problème d'échantillonnage potentiellement coûteux par une optimisation plus efficace, exploitant les méthodes modernes de descente de gradient et l'astuce de reparamétrisation. Une fois l'optimisation terminée, l'évaluation de $q(\theta)$ est instantanée, ce qui constitue un avantage majeur par rapport aux chaînes MCMC, qui nécessitent de longues simulations pour générer des échantillons indépendants. De plus, l'inférence variationnelle est bien adaptée aux grands ensembles de données grâce à son extension stochastique (SVI), qui permet d'optimiser l'ELBO par mini-batches.

Toutefois, cette approche souffre de certaines limites. Contrairement aux méthodes MCMC, qui convergent asymptotiquement vers la distribution a posteriori, l'inférence variationnelle ne fournit qu'une approximation dont la qualité dépend du choix de la famille \mathcal{Q} . Une distribution variationnelle trop simple risque de ne pas capturer des caractéristiques complexes, comme la multimodalité ou des corrélations fortes. De plus, l'optimisation de l'ELBO peut être délicate, car celle-ci est souvent non convexe, ce qui

peut entraîner des problèmes de convergence. En pratique, le choix de la famille variationnelle constitue un compromis entre expressivité et efficacité computationnelle.

5 Parcimonie et Processus Gaussiens

Les processus gaussiens (GP) se distinguent par leur grande flexibilité ainsi que par leur capacité à fournir des estimations d'incertitude sur les prédictions. Toutefois, en présence de n données d'entraînement, la construction et l'inversion de la matrice de covariance K_{nn} , de taille $n \times n$, induisent un coût computationnel en $O(n^3)$. Cette complexité devient rapidement prohibitive lorsque n croît, limitant ainsi l'applicabilité des GP à des ensembles de données de taille modérée.

Une approche classique pour pallier cette limitation consiste à introduire un ensemble restreint de points induits, noté X_m , avec $m \ll n$. Ces points, sélectionnés ou optimisés, constituent une base réduite permettant d'approximer la matrice de covariance complète. L'inférence est alors réalisée dans un espace de dimension m , ce qui permet de ramener la complexité algorithmique à $O(nm^2)$. Ce gain de performance rend possible l'application des GP à des ensembles de données plus volumineux.

L'introduction des points induits soulève néanmoins plusieurs problématiques :

1. *Choix optimal des points induits :*

La position des points X_m a un impact direct sur la qualité de l'approximation obtenue pour la matrice de covariance.

2. *Compromis entre parcimonie et précision :*

Une réduction excessive de m peut dégrader significativement la capacité du modèle à capturer la structure des données.

3. *Effets sur l'apprentissage et la robustesse :*

L'introduction de paramètres supplémentaires liés aux points induits peut entraîner un risque de surapprentissage en l'absence de mécanismes de régularisation appropriés.

5.1 Inférence variationnelle pour les GPs

L'inférence variationnelle appliquée aux processus gaussiens vise à formuler l'approximation du modèle complet par un modèle parcimonieux comme un problème d'optimisation portant sur les points induits. Cette approche permet de quantifier l'erreur d'approximation tout en imposant une régularisation explicite, à la différence des méthodes antérieures basées sur des projections (Seeger et al., 2003; Snelson and Ghahramani, 2005). Elle autorise un apprentissage conjoint d'une distribution variationnelle sur les valeurs du processus au niveau des points induits, des hyperparamètres du noyau ainsi que de la position de ces points.

Dans ce cadre, l'approche proposée par Titsias (Titsias, 2009) introduit une borne inférieure de l'évidence (*Evidence Lower Bound*, ELBO) comme objectif d'optimisation. Cette ELBO comprend un terme de pénalisation des configurations excessivement complexes, ce qui agit comme une régularisation implicite et permet de limiter les phénomènes de surapprentissage.

5.2 Distribution prédictive et approximation variationnelle

La prédiction des valeurs du processus gaussien en de nouveaux points repose sur l'étude de la distribution prédictive conditionnelle des variables latentes inconnues. Soit z les valeurs du processus à prédire aux points de test X_* , et soit f la fonction latente sous-jacente, supposée continue et bruitée selon le modèle $y = f(x) + \varepsilon$, avec $f = (f(x_1), \dots, f(x_n))^T$ les valeurs de cette fonction aux points d'entraînement X . La distribution prédictive peut alors s'exprimer comme :

$$p(z | y) = \int p(z | f) p(f | y) df,$$

où $p(z | f)$ désigne la distribution du processus évalué aux nouveaux points X_* , conditionnellement aux valeurs latentes f , tandis que $p(f | y)$ représente la postérieure exacte sur les valeurs latentes conditionnellement aux observations bruitées.

Dans le cadre exact des processus gaussiens, cette intégrale est calculable analytiquement mais implique l'inversion de la matrice de covariance K_{nn} , entraînant un coût prohibitif en $O(n^3)$. Afin de pallier cette difficulté, une approche consiste à introduire un ensemble restreint de points induits X_m , avec $m \ll n$, et les variables associées f_m , représentant les valeurs du processus aux points X_m . En étendant l'expression précédente, la distribution prédictive s'écrit :

$$p(z | y) = \int \int p(z | f, f_m) p(f | f_m, y) p(f_m | y) df df_m.$$

L'hypothèse selon laquelle f_m constitue une statistique suffisante pour décrire f permet de supposer l'indépendance conditionnelle de z et f sachant f_m , ce qui se traduit par :

$$p(z | f, f_m) = p(z | f_m).$$

De même, puisque y représente une version bruitée de f , cette hypothèse conduit à :

$$p(f | f_m, y) = p(f | f_m).$$

Par conséquent, la distribution prédictive peut être reformulée comme :

$$\begin{aligned} p(z | y) &= \int \int p(z | f_m) p(f | f_m) p(f_m | y) df df_m \\ &= \int p(z | f_m) p(f_m | y) \underbrace{\int p(f | f_m) df}_{=1} df_m \\ &= \int p(z | f_m) p(f_m | y) df_m. \end{aligned} \tag{2}$$

Dans le cadre variationnel, la distribution postérieure exacte $p(f_m | y)$ est approchée par une distribution variationnelle $\phi(f_m)$, choisie au sein d'une famille simplifiée \mathcal{Q} . Cette famille correspond aux distributions gaussiennes de moyenne μ et de matrice de covariance Λ . Afin de faire le lien avec les notations de la section 4 les équivalences sont notés,

$$\begin{aligned}\theta &= f_m \\ \lambda &= X_m \\ q(\theta) &= \phi(f_m) \\ p(\theta | y) &= p(f_m | y).\end{aligned}$$

L'objectif consiste à minimiser la divergence de Kullback-Leibler entre $\phi(f_m)$ et $p(f_m | y)$, en optimisant les paramètres de ϕ . La dépendance entre ces paramètres et les points X_m sera explicitée ultérieurement.

La distribution variationnelle jointe et sa marginale sur z s'écrivent :

$$\begin{aligned}q(z, f_m) &= p(z | f_m) \phi(f_m), \\ q(z) &= \int q(z, f_m) df_m.\end{aligned}$$

En injectant cette approximation dans l'expression (2), on obtient la distribution prédictive approchée :

$$p(z | y) \approx \int p(z | f_m) \phi(f_m) df_m = q(z) \quad (3)$$

En appliquant les formules classiques de prédiction des processus gaussiens à l'expression (3), il est possible de calculer les fonctions de moyenne et de covariance associées à la distribution prédictive $q(z)$. La loi conditionnelle $z | f_m$ est donnée par :

$$z | f_m \sim \mathcal{N}(K_{xm}K_{mm}^{-1}f_m, k(\mathbf{x}, \mathbf{x}) - K_{\mathbf{x}m}K_{mm}^{-1}K_{m\mathbf{x}}).$$

La moyenne prédictive $m_y^q(\mathbf{x})$ s'obtient par :

$$\begin{aligned}m_y^q(\mathbf{x}) &= \mathbb{E}_{\phi(f_m)}[K_{xm}K_{mm}^{-1}f_m] = K_{xm}K_{mm}^{-1}\mu, \\ m_y^q(\mathbf{x}) &= K_{xm}K_{mm}^{-1}\mu.\end{aligned} \quad (4)$$

Quant à la covariance prédictive, elle se décompose comme suit :

$$\begin{aligned}\text{Var}_q[z] &= \mathbb{E}_{\phi(f_m)}[\text{Var}[z | f_m]] + \text{Var}_{\phi(f_m)}[\mathbb{E}[z | f_m]] \\ &= k(\mathbf{x}, \mathbf{x}) - K_{\mathbf{x}m}K_{mm}^{-1}K_{m\mathbf{x}} + K_{\mathbf{x}m}K_{mm}^{-1}\Lambda K_{mm}^{-1}K_{m\mathbf{x}}.\end{aligned}$$

La fonction de covariance associée s'écrit alors :

$$k_y^q(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}') - K_{\mathbf{x}m}K_{mm}^{-1}K_{m\mathbf{x}'} + K_{\mathbf{x}m}K_{mm}^{-1}\Lambda K_{mm}^{-1}K_{m\mathbf{x}'}. \quad (5)$$

5.3 Borne inférieure variationnelle (ELBO)

Dans cette section, l'objectif est de déterminer les expressions de la moyenne μ et de la variance Λ de la distribution variationnelle $\phi(f_m)$ minimisant la divergence $\text{KL}(\phi(f_m) \| p(f_m | y))$. Cela permet de spécifier intégralement le processus gaussien variationnel. La divergence de Kullback-Leibler est minimisée à l'aide de sa borne inférieure, l'ELBO, qui s'écrit :

$$\begin{aligned}\mathcal{L}(\phi(f_m)) &= \mathbb{E}_{\phi(f_m)}[\log p(y, f_m) - \log \phi(f_m)] \\ &= \int \phi(f_m) (\log p(y, f_m) - \log \phi(f_m)) df_m\end{aligned}$$

En développant $p(y, f_m)$, il vient :

$$p(y, f_m) = \int p(y | f) p(f | f_m) p(f_m) df$$

Puis, en appliquant le logarithme :

$$\log p(y, f_m) = \log \left(\int p(y | f) p(f | f_m) df \right) + \log p(f_m)$$

Cette expression est injectée dans l'ELBO :

$$\begin{aligned}\mathcal{L}(\phi(f_m)) &= \int \phi(f_m) \left(\log \left(\int p(y | f) p(f | f_m) df \right) + \log p(f_m) - \log \phi(f_m) \right) df_m \\ &= \int \phi(f_m) \left(\log \left(\int p(y | f) p(f | f_m) df \right) + \log \frac{p(f_m)}{\phi(f_m)} \right) df_m\end{aligned}$$

La fonction logarithme étant concave, l'inégalité de Jensen s'applique :

$$\mathcal{L}(\phi(f_m)) \geq \int \phi(f_m) \left(\int p(f | f_m) \log p(y | f) df \right) df_m + \int \phi(f_m) \log \frac{p(f_m)}{\phi(f_m)} df_m$$

L'ELBO constitue une borne inférieure de la divergence KL. L'expression précédente étant inférieure à l'ELBO, elle représente également une borne inférieure. Il est ainsi possible de réécrire :

$$\mathcal{L}(\phi(f_m)) = \int \phi(f_m) \left(\int p(f | f_m) \log p(y | f) df \right) df_m + \int \phi(f_m) \log \frac{p(f_m)}{\phi(f_m)} df_m$$

Démarche :

La borne variationnelle est désormais utilisée pour retrouver les paramètres μ (eq. (4)) et Λ (eq. (5)) de la distribution variationnelle optimale $\phi^*(f_m)$. Pour ce faire, la démarche suivante est suivie :

1. Exprimer l'intégrale

$$\int p(f | f_m) \log p(y | f) df$$

sous une forme explicite donnée par :

$$\log G(f_m, y) = [\log \mathcal{N}(y | \alpha, \sigma^2 I)] - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn}).$$

avec $Q_{nn} = K_{nm} K_{mm}^{-1} K_{mn}$. Cette expression permet d'obtenir une nouvelle formulation de la borne variationnelle :

$$\mathcal{L}(\phi(f_m)) = \int \phi(f_m) \log \left(\frac{G(f_m, y) p(f_m)}{\phi(f_m)} \right) df_m.$$

2. Introduire la distribution variationnelle optimale $\phi^*(f_m)$ en minimisant la divergence de Kullback-Leibler entre $\phi(f_m)$ et $\frac{Q(f_m)}{Z}$, où $Q(f_m)$ et Z seront définis ultérieurement, tels que :

$$\mathcal{L}(X_m, \phi) + \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn}) = \log(Z) - KL \left(\phi(f_m) || \frac{Q(f_m)}{Z} \right).$$

En annulant la divergence KL, il vient :

$$\phi^*(f_m) = \frac{Q(f_m)}{Z}.$$

3. En conséquence, une borne inférieure variationnelle optimale dépendant de ϕ^* est obtenue. Une forme simplifiée de cette borne sera ensuite recherchée.
4. Finalement, il est établi que

$$\phi^*(f_m) \propto Q(f_m),$$

ce qui permet d'identifier les paramètres μ et Λ .

1. Simplification de l'intégrale $\int p(f | f_m) \log p(y | f) df$:

Dans un premier temps, $\log p(y | f)$ est calculé. Dans un second temps, l'expression de $p(f | f_m)$ est obtenue, puis l'intégrale est évaluée.

$$\log p(y | f) = \log \left(\prod_{i=1}^n \mathcal{N}(y_i | f_i, \sigma^2) \right) = \log \left((2\pi\sigma^2)^{-\frac{n}{2}} \exp \left(-\frac{1}{2\sigma^2} (y - f)^\top (y - f) \right) \right).$$

Ce qui donne :

$$\log p(y | f) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y - f)^\top (y - f).$$

La distribution conditionnelle $p(f | f_m)$ est une loi normale de moyenne et de covariance :

$$\alpha = K_{nm}K_{mm}^{-1}f_m = \mathbb{E}_{p(f|f_m)}[f], \quad \Sigma = K_{nn} - Q_{nn} = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}.$$

Ainsi :

$$p(f | f_m) = \mathcal{N}(f | \alpha, \Sigma).$$

Il est alors possible de calculer l'intégrale $\int p(f | f_m) \log p(y | f) df = \log G(f_m, y)$.
L'expression $\log G(f_m, y)$ s'écrit :

$$\log G(f_m, y) = \int p(f | f_m) \left(-\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} (y - f)^\top (y - f) \right) df.$$

Cette expression contient : (i) un terme constant, indépendant de f , et (ii) un terme quadratique en f .

Terme constant :

$$\int p(f | f_m) \left(-\frac{n}{2} \log(2\pi\sigma^2) \right) df = -\frac{n}{2} \log(2\pi\sigma^2).$$

Terme quadratique :

L'espérance du terme quadratique est donnée par :

$$\mathbb{E}_{p(f|f_m)}[(y - f)^\top (y - f)] = (y - \alpha)^\top (y - \alpha) + \text{Tr}(K_{nn} - Q_{nn}).$$

En injectant les deux résultats :

$$\log G(f_m, y) = -\frac{n}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} ((y - \alpha)^\top (y - \alpha) + \text{Tr}(K_{nn} - Q_{nn})).$$

Ce qui conduit à :

$$\log G(f_m, y) = \log \mathcal{N}(y | \alpha, \sigma^2 I) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn}).$$

Ainsi, la borne variationnelle devient :

$$\mathcal{L}(\phi(f_m)) = \int \phi(f_m) \log \left(\frac{G(f_m, y) p(f_m)}{\phi(f_m)} \right) df_m.$$

2. Introduction de la distribution variationnelle optimale $\phi^*(f_m)$:

L'expression de la borne variationnelle $\mathcal{L}(\phi(f_m))$ est rappelée :

$$\mathcal{L}(\phi(f_m)) = \int \phi(f_m) \left[\log \mathcal{N}(y | \alpha, \sigma^2 I) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn}) + \log \frac{p(f_m)}{\phi(f_m)} \right] df_m.$$

Cette expression peut être décomposée comme suit :

$$\mathcal{L}(\phi(f_m)) = \underbrace{\int \phi(f_m) \log \left(\frac{\mathcal{N}(y | \alpha, \sigma^2 I) p(f_m)}{\phi(f_m)} \right) df_m}_S - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn}).$$

L'objectif est de maximiser cette borne variationnelle. L'attention est portée sur le premier terme, noté S , que l'on souhaite exprimer comme une divergence de Kullback-Leibler. Pour cela, les quantités suivantes sont introduites :

$$Q(f_m) = p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I), \quad S = \int \phi(f_m) \log \left(\frac{Q(f_m)}{\phi(f_m)} \right) df_m.$$

Le terme S est alors lié à la divergence de Kullback-Leibler entre $\phi(f_m)$ et $Q(f_m)/Z$, définie par :

$$KL \left(\phi(f_m) \parallel \frac{Q(f_m)}{Z} \right) = \int \phi(f_m) \log \left(\frac{\phi(f_m)}{Q(f_m)/Z} \right) df_m.$$

La constante de normalisation Z est donnée par :

$$Z = \int p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I) df_m.$$

Ainsi, il vient :

$$S = \log(Z) - KL \left(\phi(f_m) \parallel \frac{Q(f_m)}{Z} \right).$$

Cas optimal : Annulation de la divergence KL

La divergence de Kullback-Leibler est toujours positive et s'annule si, et seulement si :

$$\phi^*(f_m) = \frac{Q(f_m)}{Z} = \frac{p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I)}{Z}. \quad (6)$$

En remplaçant $\phi(f_m)$ par $\phi^*(f_m)$, on obtient :

$$\int \phi^*(f_m) \log \left(\frac{p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I)}{\phi^*(f_m)} \right) df_m = \log(Z).$$

Expression finale de la borne variationnelle optimale

Ainsi, pour $\phi(f_m) = \phi^*(f_m)$, la borne variationnelle devient :

$$\mathcal{L}(X_m, \phi^*) = \log(Z) - \frac{1}{2\sigma^2} \text{tr}(K_{nn} - Q_{nn}) = \log \left(\int p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I) df_m \right) - \frac{1}{2\sigma^2} \text{tr}(K_{nn} - Q_{nn}).$$

Cette expression correspond à l'optimisation variationnelle du modèle, dans laquelle la distribution $\phi^*(f_m)$ est choisie de façon à maximiser la borne variationnelle.

3. Forme simplifiée de la borne variationnelle maximale :

En observant que Z peut s'écrire comme $\int p(f_m) \tilde{q}(y \mid f_m) df_m$, avec \tilde{q} la densité conditionnelle de la variable $y = K_{nm} K_{mm}^{-1} f_m + \epsilon$, où $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ et $f_m \sim \mathcal{N}(0, K_{mm})$, il est possible d'établir :

$$Z = \mathcal{N}(y \mid 0, \sigma^2 I + Q_{nn}).$$

Ce qui revient à montrer que :

$$y \sim \mathcal{N}(0, \sigma^2 I + Q_{nn}).$$

1. **Espérance :**

$$\mathbb{E}[y] = K_{nm} K_{mm}^{-1} \mathbb{E}[f_m] = 0.$$

2. **Variance :**

$$\text{Cov}(y) = K_{nm} K_{mm}^{-1} \text{Cov}(f_m) (K_{nm} K_{mm}^{-1})^\top + \sigma^2 I,$$

avec :

$$\text{Cov}(f_m) = K_{mm}.$$

Ainsi, par symétrie et positivité de K_{nn} , il en résulte :

$$\text{Cov}(y) = K_{nm} K_{mm}^{-1} K_{mm} (K_{mm}^{-1})^\top K_{nm}^\top + \sigma^2 I = Q_{nn} + \sigma^2 I.$$

Par conséquent :

$$y \sim \mathcal{N}(0, \sigma^2 I + Q_{nn}), \quad \text{et} \quad Z = \mathcal{N}(y \mid 0, \sigma^2 I + Q_{nn}).$$

La borne variationnelle s'écrit alors :

$$\mathcal{L}(X_m) = \log \mathcal{N}(y \mid 0, \sigma^2 I + Q_{nn}) - \frac{1}{2\sigma^2} \text{tr}(K_{nn} - Q_{nn}).$$

La maximisation de cette fonction dépend directement des points induits et de leur nombre, ce qui permet d'optimiser ces paramètres.

Dans le cadre variationnel, la distribution q est définie comme $q(f, f_m) = p(f \mid f_m) \phi^*(f_m)$. Les points induits contrôlent la flexibilité de cette distribution variationnelle via la moyenne conditionnelle $K_{nm} K_{mm}^{-1} f_m$, qui définit comment l'information contenue dans f_m est projetée sur l'ensemble des f . Le choix des points induits affecte également la covariance conditionnelle $K_{nn} - Q_{nn}$, laquelle gouverne l'incertitude résiduelle sur f conditionnellement à f_m .

4. **Distribution variationnelle optimisée $\phi^*(f_m)$:**

Les points X_m influencent également la forme de la distribution optimisée $\phi^*(f_m)$. D'après eq. (6), il est établi :

$$\phi^*(f_m) = \frac{p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I)}{Z} \propto p(f_m) \mathcal{N}(y \mid \alpha, \sigma^2 I) = \mathcal{N}(f_m \mid 0, K_{mm}) \mathcal{N}(y \mid \alpha, \sigma^2 I),$$

où :

$$p(f_m) = \mathcal{N}(f_m \mid 0, K_{mm}) = \frac{1}{(2\pi)^{m/2} |K_{mm}|^{1/2}} \exp \left(-\frac{1}{2} f_m^\top K_{mm}^{-1} f_m \right),$$

et :

$$\mathcal{N}(y \mid \alpha, \sigma^2 I) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp \left(-\frac{1}{2\sigma^2} (y - \alpha)^\top (y - \alpha) \right),$$

avec $\alpha = K_{nm} K_{mm}^{-1} f_m$.

En développant le terme quadratique :

$$(y - \alpha)^\top (y - \alpha) = y^\top y - 2y^\top K_{nm} K_{mm}^{-1} f_m + f_m^\top K_{mm}^{-1} K_{mn} K_{nm} K_{mm}^{-1} f_m.$$

Cette expression permet de réécrire $\phi^*(f_m)$ sous la forme exponentielle :

$$\phi^*(f_m) \propto \exp \left(-\frac{1}{2} f_m^\top \Lambda f_m + f_m^\top b \right),$$

où :

$$\Lambda = K_{mm}^{-1} + \frac{1}{\sigma^2} K_{mm}^{-1} K_{mn} K_{nm} K_{mm}^{-1} \quad (7)$$

et :

$$b = \frac{1}{\sigma^2} K_{mm}^{-1} K_{mn} y.$$

En complétant le carré dans f_m , il vient :

$$-\frac{1}{2} f_m^\top \Lambda f_m + f_m^\top b = -\frac{1}{2} (f_m - \mu)^\top \Lambda (f_m - \mu) + \frac{1}{2} \mu^\top \Lambda \mu,$$

avec :

$$\mu = \Lambda^{-1} b.$$

Le terme $\frac{1}{2} \mu^\top \Lambda \mu$ ne dépendant pas de f_m , il peut être ignoré pour la forme normalisée de la distribution.

La matrice Λ , définie par l'équation (7), peut être réécrite comme suit :

$$\Lambda = K_{mm}^{-1} \left(I + \frac{1}{\sigma^2} K_{mn} K_{nm} \right) K_{mm}^{-1}.$$

En posant :

$$\Sigma = I + \frac{1}{\sigma^2} K_{mn} K_{nm},$$

il est obtenue :

$$\Lambda = K_{mm}^{-1} \Sigma K_{mm}^{-1} \Rightarrow \Lambda^{-1} = K_{mm} \Sigma^{-1} K_{mm}.$$

Le vecteur b s'écrit :

$$b = \frac{1}{\sigma^2} K_{mm}^{-1} K_{mn} y,$$

d'où :

$$\mu = \Lambda^{-1} b = \frac{1}{\sigma^2} K_{mm} \Sigma^{-1} K_{mn} y.$$

La distribution optimisée $\phi^*(f_m)$ prend alors la forme :

$$\phi^*(f_m) \propto \exp \left(-\frac{1}{2} (f_m - \mu)^\top \Lambda (f_m - \mu) \right),$$

ce qui correspond à :

$$\phi^*(f_m) = \mathcal{N}(f_m \mid \mu, \Lambda).$$

Le processus gaussien variationnel est ainsi complètement spécifié.

5.4 Sélection des points induits

Une méthode couramment utilisée pour sélectionner les points induits est l'optimisation par gradient. Cette approche consiste à maximiser la borne inférieure variationnelle $\mathcal{L}(\phi)$ par rapport aux positions des points induits X_m . Cependant, cette méthode peut être coûteuse en raison de la grande dimensionnalité du problème, surtout lorsque le nombre de points induits m est élevé.

Une alternative est la sélection gloutonne Titsias (2009). Cette méthode exploite le fait que $\mathcal{L}(\phi)$ est une fonction croissante lorsque de nouveaux points induits sont ajoutés. L'idée est de commencer avec un ensemble vide de points induits et d'ajouter itérativement les points qui maximisent l'augmentation de $\mathcal{L}(\phi)$. Cette approche est plus rapide que l'optimisation par gradient, mais elle peut conduire à des solutions sous-optimales.

Enfin, une autre méthode consiste à utiliser une approche basée sur l'échantillonnage. Par exemple, on peut sélectionner les points induits en échantillonnant aléatoirement un sous-ensemble des données d'entraînement ou en utilisant des techniques comme le clustering, par exemple k -means++ Arthur and Vassilvitskii (2006), pour identifier les points les plus représentatifs.

5.5 Inférence Variationnelle Stochastique (SVI)

Dans le but de rendre l'inférence variationnelle pour GP extensible à de plus grands jeux de données, Hensman et al. (2013) propose une extension de la méthode variationnelle introduite par Titsias (2009). Cette nouvelle approche autorise la Stochastic Variational Inference (SVI), c'est-à-dire qu'elle permet de mettre à jour les paramètres variationnels à partir de mini-lots de données, plutôt que d'utiliser la totalité de l'échantillon d'entraînement à chaque itération.

Bien que la borne variationnelle de Titsias (2009) fournisse une bonne approximation, elle est dite condensée dans la mesure où la fonction latente f est intégrée de manière globale. La borne variationnelle obtenue par Titsias est la suivante :

$$\mathcal{L}_{\text{Titsias}} = \underbrace{\log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, Q_{nn} + \sigma_n^2 \mathbf{I})}_{\text{terme de log-vraisemblance}} - \underbrace{\frac{1}{2\sigma_n^2} \text{tr}(K_{nn} - Q_{nn})}_{\text{terme correctif}},$$

Cette forme est dite condensée car elle ne permet pas une décomposition explicite de la vraisemblance en somme sur chaque observation. Il n'est pas possible de recourir efficacement à des mini-lots pour mettre à jour les paramètres avec cette intégration

globale, car la factorisation complète de la vraisemblance reste dépendante de l'ensemble entier des observations à chaque étape.

Le travail de Hensman et al. (2013) repose sur une reformulation de la borne variationnelle pour autoriser une factorisation permettant l'optimisation stochastique. En effet, on considère un processus gaussien muni de points induits X_m et f_m les valeurs de la fonction au points induits. Dans le cadre variationnel, une distribution gaussienne est introduite pour approximer le postérieur, paramétrée par une moyenne \mathbf{m} et une covariance \mathbf{S} :

$$\phi(\mathbf{f}_m) = \mathcal{N}(\mathbf{m}, \mathbf{S}).$$

En appliquant la construction standard de l'ELBO, on obtient :

$$\mathcal{L} = \mathbb{E}_{\phi(\mathbf{f}_m)} [\log p(\mathbf{y} | \mathbf{f}_m)] + \log p(\mathbf{f}_m) - \log \phi(\mathbf{f}_m)$$

En supposant que la vraisemblance se factorise sur les observations $\{(x_i, y_i)\}_{i=1}^n$, cette borne se réécrit comme une somme de termes locaux.

Pour chaque point x_i , on définit le vecteur de covariances croisées avec les inducing points :

$$\mathbf{k}_i \equiv \begin{bmatrix} k(x_i, X_{m_1}) \\ \vdots \\ k(x_i, X_{m_m}) \end{bmatrix} \in \mathbb{R}^m.$$

D'après la propriété de conditionnement d'un processus gaussien, la distribution de $f(x_i)$ conditionnellement à \mathbf{f}_m est :

$$f(x_i) | \mathbf{f}_m \sim \mathcal{N}(\mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{f}_m, k(x_i, x_i) - \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{k}_i).$$

Dans le cadre de la régression gaussienne, un bruit additif i.i.d. est ajouté : $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, d'où la distribution de l'observation :

$$y_i = f(x_i) + \varepsilon_i \Rightarrow y_i \sim \mathcal{N}(\mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{f}_m, \sigma_n^2).$$

L'espérance par rapport à $\phi(\mathbf{f}_m)$ permet alors de réécrire la borne variationnelle comme suit :

$$\mathcal{L}_2 = \sum_{i=1}^n \left\{ \log \mathcal{N}(y_i | \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{m}, \sigma_n^2) - \frac{1}{2} \sigma_n^2 \tilde{k}_{i,i} - \frac{1}{2} \text{tr}(\mathbf{S} \mathbf{\Lambda}_i) \right\} - \text{KL}(\phi(\mathbf{f}_m) \| p(\mathbf{f}_m)), \quad (8)$$

où :

$$\tilde{k}_{i,i} = \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{k}_i, \quad \mathbf{\Lambda}_i = \sigma_n^2 \mathbf{K}_{mm}^{-1} \mathbf{k}_i \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1}.$$

La divergence de Kullback-Leibler entre deux lois normales permet de calculer explicitement :

$$\text{KL}(\phi(\mathbf{f}_m) \| p(\mathbf{f}_m)) = \frac{1}{2} \left[\text{tr}(\mathbf{K}_{mm}^{-1} \mathbf{S}) + \mathbf{m}^\top \mathbf{K}_{mm}^{-1} \mathbf{m} - m + \log \frac{|\mathbf{K}_{mm}|}{|\mathbf{S}|} \right].$$

Le calcul des dérivées de \mathcal{L}_2 par rapport aux paramètres variationnels \mathbf{m} et \mathbf{S} donne :

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{m}} = \sigma_n^{-2} \mathbf{K}_{mm}^{-1} \mathbf{K}_{mn} \mathbf{y} - \mathbf{\Lambda} \mathbf{m}, \quad \frac{\partial \mathcal{L}_2}{\partial \mathbf{S}} = \frac{1}{2} \mathbf{S}^{-1} - \frac{1}{2} \mathbf{\Lambda}.$$

L'annulation de ces dérivées mène à l'optimum :

$$\mathbf{S} = \mathbf{\Lambda}^{-1}, \quad \mathbf{m} = \sigma_n^{-2} \mathbf{\Lambda}^{-1} \mathbf{K}_{mm}^{-1} \mathbf{K}_{mn} \mathbf{y}.$$

Dans la formulation \mathcal{L}_2 de l'équation (8), chaque terme de la somme pouvant être calculé indépendamment, il est possible d'utiliser un sous-ensemble $\mathcal{B} \subset \{1, \dots, n\}$ d'indices. Cela permet une mise à jour incrémentale des paramètres variationnels à l'aide d'échantillons (mini-lots). À chaque itération, un mini-lot \mathcal{B} est sélectionné, et l'estimateur suivant est utilisé :

$$\widehat{\mathcal{L}}_2 = \frac{n}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left\{ \log \mathcal{N}(y_i \mid \mathbf{k}_i^\top \mathbf{K}_{mm}^{-1} \mathbf{m}, \sigma_n^2) - \frac{1}{2} \sigma_n^2 \tilde{k}_{i,i} - \frac{1}{2} \text{tr}(\mathbf{S} \mathbf{\Lambda}_i) \right\} - \text{KL}(\phi(\mathbf{f}_m) \parallel p(\mathbf{f}_m)).$$

Cette estimation stochastique rend l'optimisation de \mathcal{L}_2 scalable et adaptée à l'apprentissage sur de très grands ensembles de données.

6 Deep Kernel Learning

6.1 Problème des GP en haute dimension

Les Processus Gaussiens sont largement utilisés en apprentissage automatique pour modéliser des relations complexes entre des données grâce à leur flexibilité et leur cadre probabiliste rigoureux. Cependant, ils souffrent d'une limitation majeure lorsqu'ils sont appliqués à des données de grande dimension : la difficulté de définir un noyau efficace qui capture correctement la structure sous-jacente non linéaire des données.

Pour pallier ce problème, plusieurs lignes de recherche proposent d'adapter ou d'élargir la structure du noyau de manière à mieux représenter la variété intrinsèque des données (Duvenaud et al. (2013), Wilson and Adams (2013)).

La méthode utilisée dans ce rapport pour répondre à la problématique est celle du Deep Kernel Learning Wilson et al. (2016b).

6.2 Principe du Deep Kernel Learning

Le Deep Kernel Learning (DKL) est une approche hybride qui combine les réseaux de neurones profonds avec les processus gaussiens pour surmonter limites des GPs. En effet, le réseau de neurones apprend à transformer les données brutes en une représentation plus riche et structurée, appelée espace latent. Cet espace latent est conçu pour capturer les aspects complexes des données. Ainsi, plutôt que d'appliquer directement le GP aux données brutes de haute dimension, le GP opère sur la représentation du réseau de neurones de plus faible dimension où les relations non linéaires sont mieux structurées.

Dans l'article Wilson et al. (2016b), l'utilisation de points induits est également introduite pour améliorer la parcimonie du processus gaussien. Bien que l'espace latent soit de dimension réduite par rapport aux données d'entrée, cette dimension reste suffisamment élevée pour que le coût computationnel de l'entraînement du processus gaussien demeure significatif. Par conséquent, ces points induits servent à résumer l'information contenue dans l'espace latent, réduisant ainsi le coût du calcul de la matrice de covariance. La sélection des points induits permet de construire une approximation qui conserve la structure sous-jacente tout en simplifiant considérablement le problème.

Par ailleurs, Wilson et al. (2016b) opte pour la méthode KISS-GP (Kernel Interpolation for Scalable Structured Gaussian Processes) Wilson and Nickisch (2015) plutôt que pour les approches classiques des sparse GPs. La méthode KISS-GP exploite la structure sous-jacente des grilles régulières pour interpoler efficacement les covariances, permettant ainsi une réduction significative de la complexité du calcul. Ainsi, KISS-GP construit une approximation de la matrice de covariance à partir d'une grille de points qui facilite l'interpolation de valeurs pour de nouveaux points et rend l'inversion de la matrice de covariance plus rapide et moins coûteuse.

6.3 Formulation mathématique

Il est considéré un GP de noyau k d'hyperparamètre $\theta = \{l, \sigma_f, \sigma_n\}$. De plus, il est considéré des données $Y = (y_1, \dots, y_n)^\top$ $X = (x_1, \dots, x_n)^\top$. Il est considéré un réseau de neurones ψ_w dont les poids sont notés w . Les entrées du noyau sont alors transformées par le réseau de neurones,

$$k(x_i, x_j | \theta) \rightarrow k(h_i | \theta, w),$$

avec $h_i = \psi_w(x_i)$. Il est alors noté,

$$K_{nn} = \left(k(h_i | \theta, w) \right)_{1 \leq i, j \leq n}$$

Le calcul de l'inverse de K_{nn} étant trop couteux pour un grand nombre de données d'entraînement, il est nécessaire d'approximer cette quantité.

Dans la version originale de (Wilson et al., 2016b) la matrice K_{nn} est approximé à l'aide de l'approche KISS-GP qui permet de réaliser une approximation directe de la matrice de covariance. Seulement, selon (Wilson et al., 2016a) cette méthode n'est pas applicable pour des données de grandes dimensions. La méthode employée pour les SVDKL est d'introduire des points induits f_m et de les optimiser. Il s'agit de la méthode des Sparse-GP introduite dans la partie précédente. Le même raisonnement s'applique à l'exception que l'on substitue $\psi_w(X)$ à X . Le formalisme est rappelé. Il s'agit de poser p le modèle, q le modèle variationnels et \mathcal{L} l'ELBO.

Le modèle probabiliste joint s'écrit,

$$p(y, f, f_m) = p(y | f) p(f | f_m) p(f_m)$$

avec,

- Distribution des observations conditionnelles à f :

$$p(y | f) = \prod_{i=1}^n p(y_i | f(x_i))$$

- Distribution du GP sur les observations conditionnées à f_m :

$$p(f | f_m) = \mathcal{N}(f | K_{nm} K_{mm}^{-1} f_m, K_{nn} - K_{nm} K_{mm}^{-1} K_{mn})$$

- Distribution a priori sur les points d'induction f_m :

$$p(f_m) = \mathcal{N}(f_m | 0, K_{mm})$$

La modèle variationnelle $q(f, f_m)$ s'écrit,

$$q(f, f_m) = p(f | f_m) \phi(f_m)$$

avec $\phi(f_m)$ une distribution variationnelle gaussienne paramétrée par $\lambda = (\mu, \Sigma)$,

$$\phi(f_m) = \mathcal{N}(f_m \mid \mu, \Sigma)$$

Le problème d'optimisation est le suivant,

$$q^* = \arg \min_q \text{KL}(q(f, f_m) \parallel p(f, f_m \mid y))$$

Finalement l'ELBO s'écrit,

$$\mathcal{L} = \log(\mathcal{N}(Y \mid 0, \sigma_n^2 I + Q_{nn})) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn})$$

avec,

$$Q_{nn} = K_{nm} K_{mm}^{-1} K_{mn}$$

$$K_{nn} = k(h, h; \theta)$$

$$K_{nm} = k(h, X_m; \theta)$$

$$K_{mm} = k(X_m, X_m; \theta)$$

6.4 Optimisation du modèle

L'objectif est d'optimiser l'ELBO à l'aide de la descente de gradient. Il est nécessaire de calculer le gradient de l'ELBO par rapport à ses paramètres. Trois groupes de paramètres sont identifiés :

θ : les hyperparamètres du noyau du GP

U : les points induits

w : les poids du réseau de neurones

Les dérivés partiels de \mathcal{L} sont calculés à partir de la règle de la chaîne.

6.5 Formalisation algorithmique

Il est fait trois propositions indépendantes d'algorithme pour l'optimisation de l'ELBO.

Le premier algorithme propose d'optimiser l'ensemble des paramètres par descente de gradient.

Algorithm 2 Optimisation du modèle DKL avec **Sparse GP** par descente de gradient

Require: Ensemble d'entraînement (X, Y) , réseau profond ψ_w , hyperparamètres du kernel θ , points induits X_m , taux d'apprentissage η , nombre d'itérations T

1: Initialiser les paramètres θ , X_m et w

2: **for** $t = 1$ **to** T **do**

3: **Propagation avant:**

4: Calculer la représentation profonde : $h \leftarrow \psi_w(X)$

5: Calculer les matrices de covariance :

$$K_{nn} \leftarrow k(h, h; \theta), \quad K_{nm} \leftarrow k(h, X_m; \theta), \quad K_{mm} \leftarrow k(X_m, X_m; \theta)$$

6: **Calcul de l'ELBO:**

$$Q_{nn} \leftarrow K_{nm} K_{mm}^{-1} K_{mn}$$

$$\mathcal{L} \leftarrow \log(\mathcal{N}(Y | 0, \sigma_n^2 I + Q_{nn})) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn})$$

7: **Calcul des gradients:**

$$g_\theta \leftarrow \nabla_\theta \mathcal{L}, \quad g_{X_m} \leftarrow \nabla_{X_m} \mathcal{L}, \quad g_w \leftarrow \nabla_w \mathcal{L}$$

8: **Mise à jour des paramètres:**

$$\theta \leftarrow \theta + \eta g_\theta, \quad X_m \leftarrow X_m + \eta g_{X_m}, \quad w \leftarrow w + \eta g_w$$

9: **end for**

10: **return** θ, X_m, w

Le deuxième algorithme propose à chaque itération de choisir les points induit à l'aide de l'algorithme K-means puis d'effectuer un pas de gradient.

Algorithm 3 Optimisation du modèle DKL avec **Sparse GP** par descente de gradient et sélection **K-means** des points induits

Require: Ensemble d'entraînement (X, Y) , réseau profond ψ_w , hyperparamètres du kernel θ , points induits X_m , taux d'apprentissage η , nombre d'itérations T

1: Initialiser les paramètres θ , X_m et w

2: **for** $t = 1$ **to** T **do**

3: **Propagation avant:**

4: Calculer la représentation profonde : $h \leftarrow \psi_w(X)$

5: Mettre à jour les points induits par k-means : $X_m \leftarrow \text{KMeans}(h, m)$

6: Calculer les matrices de covariance :

$$K_{nn} \leftarrow k(h, h; \theta), \quad K_{nm} \leftarrow k(h, X_m; \theta), \quad K_{mm} \leftarrow k(X_m, X_m; \theta)$$

7: **Calcul de l'ELBO:**

$$Q_{nn} \leftarrow K_{nm} K_{mm}^{-1} K_{mn}$$

$$\mathcal{L} \leftarrow \log(\mathcal{N}(Y | 0, \sigma_n^2 I + Q_{nn})) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn})$$

8: **Calcul des gradients:**

$$g_\theta \leftarrow \nabla_\theta \mathcal{L}, \quad g_w \leftarrow \nabla_w \mathcal{L}$$

9: **Mise à jour des paramètres:**

$$\theta \leftarrow \theta + \eta g_\theta, \quad w \leftarrow w + \eta g_w$$

10: **end for**

11: **return** θ, X_m, w

Le troisième algorithme propose à chaque itération de choisir les points induits à l'aide d'une sélection gloutonne puis d'effectuer un pas de gradient.

Algorithm 4 Optimisation du modèle DKL avec **Sparse GP** par descente de gradient et sélection greedy des points induits

Require: Ensemble d'entraînement (X, Y) , réseau profond ψ_w , hyperparamètres du kernel θ , points induits X_m , taux d'apprentissage η , nombre d'itérations T

```

1: Initialiser les paramètres  $\theta$ ,  $X_m$  et  $w$ 
2: for  $t = 1$  to  $T$  do
3:   Propagation avant:
4:     Calculer la représentation profonde :  $h \leftarrow \psi_w(X)$ 
5:     Sélection greedy des points induits :
6:        $X_m \leftarrow \emptyset$ 
7:       for  $i = 1$  to  $m$  do
8:          $meilleurELBO \leftarrow -\infty$ 
9:          $meilleurCandidat \leftarrow \text{None}$ 
10:        for chaque  $x$  dans  $h \setminus X_m$  do
11:           $Z_{candidat} \leftarrow X_m \cup \{x\}$ 
12:          Calculer l'ELBO candidate  $\mathcal{L}_{candidat}$  en utilisant  $Z_{candidat}$ 
13:          if  $\mathcal{L}_{candidat} > meilleurELBO$  then
14:             $meilleurELBO \leftarrow \mathcal{L}_{candidat}$ 
15:             $meilleurCandidat \leftarrow x$ 
16:          end if
17:        end for
18:         $X_m \leftarrow X_m \cup \{meilleurCandidat\}$ 
19:      end for
20:      Calculer les matrices de covariance :
```

$$K_{nn} \leftarrow k(h, h; \theta), \quad K_{nm} \leftarrow k(h, X_m; \theta), \quad K_{mm} \leftarrow k(X_m, X_m; \theta)$$

```

21:   Calcul de l'ELBO:
```

$$Q_{nn} \leftarrow K_{nm} K_{mm}^{-1} K_{mn}$$

$$\mathcal{L} \leftarrow \log(\mathcal{N}(Y | 0, \sigma_n^2 I + Q_{nn})) - \frac{1}{2\sigma^2} \text{Tr}(K_{nn} - Q_{nn})$$

```

22:   Calcul des gradients:
```

$$g_\theta \leftarrow \nabla_\theta \mathcal{L}, \quad g_w \leftarrow \nabla_w \mathcal{L}$$

```

23:   Mise à jour des paramètres:
```

$$\theta \leftarrow \theta + \eta g_\theta, \quad w \leftarrow w + \eta g_w$$

```

24: end for
```

```

25: return  $\theta$ ,  $X_m$ ,  $w$ 

```

Pour l'ensemble des algorithmes la mise à jour des paramètres a été conservé dans sa forme la plus simple par soucis de clarté. Cependant, les paramètres étant de nature très différente il est nécessaire d'utiliser une stratégie d'apprentissage adaptative. Le gradient naturel ne sera pas utilisé ici par soucis de simplicité de l'implémentation. Cependant,

il pourra être utilisé un optimiseur plus simple évoqué précédemment tel qu'ADAM ou AdaGrad.

6.6 Vers une extension scalable des DKL : les SVDKL

Comme pour les méthodes parcimonieuses en processus gaussiens, l'utilisation de réseaux de neurones dans les DKL peut rapidement devenir coûteuse lorsque la taille du jeu de données augmente. En effet, le calcul du noyau et l'optimisation conjointe du réseau et de la fonction de covariance peuvent s'avérer prohibitifs à grande échelle.

Pour remédier à cette difficulté, il est possible de recourir aux techniques d'inférence stochastique, dont l'objectif est de réduire le coût de calcul tout en préservant les performances prédictives. Dans le cadre des DKL, cette démarche se concrétise notamment par la méthode *Stochastic Variational Deep Kernel Learning* (SVDKL) introduite dans Wilson et al. (2016a). L'idée est d'ajuster la formulation du problème, à l'instar de ce qui est fait pour les processus gaussiens parcimonieux (voir la section 5), afin de disposer d'une version approchée des variables du modèle. Cela permet d'entraîner le réseau par mini-lots et de manipuler plus aisément un grand volume de données. De cette façon, la puissance des représentations issues des réseaux profonds se combine avec la scalabilité de l'inférence variationnelle stochastique.

Dans les SVDKL, on intègre une couche de processus gaussien parcimonieux *après* le réseau de neurones profond, selon un schéma similaire à celui décrit dans la section 5. Les points d'induits permettent de construire une approximation de la distribution a posteriori en utilisant moins de ressources que dans un processus gaussien complet. Cette approximation repose sur une procédure d'inférence variationnelle qui optimise simultanément la distribution a posteriori gaussienne et les paramètres du réseau. De plus, l'implémentation en mini-lots permet de limiter le temps de calcul car chaque itération de la descente de gradient ne traite qu'une fraction des données.

7 Expériences et Résultats

7.1 Jeu de données : Olivetti Faces

Le jeu de données *Olivetti Faces*, fourni par la bibliothèque `sklearn.datasets`, contient 400 images en niveaux de gris de résolution 64×64 pixels, représentant 40 individus différents. Chaque sujet est photographié sous des expressions faciales et des conditions d'éclairage variées, introduisant une diversité visuelle significative.

Afin de concevoir une tâche de régression non triviale, une stratégie d'augmentation des données a été mise en œuvre. Elle consiste à appliquer à chaque image une rotation aléatoire dans l'intervalle $[-45^\circ, 45^\circ]$, puis à définir comme variable cible l'angle exact de rotation. Cette tâche présente un intérêt méthodologique, dans la mesure où elle nécessite la prédiction d'une variable continue à partir d'entrées visuelles redondantes et structurellement similaires.

L'ensemble de données ainsi augmenté comporte 2000 échantillons. Une partition aléatoire en deux sous-ensembles distincts a été réalisée, en allouant 80 % des données à l'apprentissage et 20 % à l'évaluation des performances.

7.2 Protocole expérimental

L'évaluation porte sur les six modèles suivants :

- **Processus gaussien (GP)**
- **Réseau de neurones (NN)**
- **Sparse Gaussian Process (SGP)**
- **Stochastic Variational Gaussian Process (SVGP)**
- **Deep Kernel Learning (DKL)**
- **Stochastic Variational Deep Kernel Learning (SVDKL)**

Les performances de ces approches sont analysées à partir d'un protocole expérimental. Les sections suivantes présentent les principaux éléments méthodologiques mis en œuvre pour garantir une évaluation fiable et reproductible.

Dans un premier temps, l'architecture des réseaux de neurones utilisés est décrite en détail, en insistant sur leur rôle dans l'extraction de représentations intermédiaires à partir des données d'entrée. Ces représentations sont ensuite exploitées par les modèles à noyau dans le cadre du Deep Kernel Learning.

Les choix d'optimisation sont ensuite discutés. Les méthodes L-BFGS et Adam ont été sélectionnées pour leurs performances en termes de convergence et de stabilité. Les précautions spécifiques prises pour stabiliser l'entraînement, notamment en présence de modèles probabilistes complexes, sont également précisées.

Enfin, l'utilisation systématique de la différentiation automatique est soulignée. Cet outil joue un rôle central dans le calcul efficace des gradients.

7.3 Réseaux de neurones

Les réseaux de neurones artificiels (RNA) constituent une classe de modèles paramétriques particulièrement flexibles, capables d'approximer des fonctions complexes qui sont souvent difficiles à représenter analytiquement. Leur intérêt théorique repose notamment sur le théorème d'approximation universelle (Cybenko, 1989; Hornik, 1991), selon lequel un réseau à une seule couche cachée, muni d'une fonction d'activation non linéaire, peut approximer toute fonction continue sur un compact avec une précision arbitraire, à condition que le nombre de neurones de la couche cachée soit suffisant.

Structure d'un réseau de neurones entièrement connecté

Un réseau de neurones à une couche cachée peut être modélisé formellement comme la composition d'une transformation linéaire suivie d'une non-linéarité :

$$f_{\theta}(x) = W_2 \phi(W_1 x + b_1) + b_2$$

où :

- $x \in \mathbb{R}^{d_{in}}$ désigne l'entrée du modèle ;
- $W_1 \in \mathbb{R}^{d_h \times d_{in}}$ et $b_1 \in \mathbb{R}^{d_h}$ sont les paramètres de la couche cachée ;
- $W_2 \in \mathbb{R}^{d_{out} \times d_h}$ et $b_2 \in \mathbb{R}^{d_{out}}$ sont les paramètres de la couche de sortie ;
- $\phi(\cdot)$ représente une fonction d'activation non linéaire, typiquement la fonction ReLU définie par $\phi(x) = \max(0, x)$.

La présence de la non-linéarité est essentielle, car elle confère au réseau la capacité d'apprendre des représentations riches et non triviales, dépassant les limites imposées par les modèles purement linéaires.

Réseaux de neurones convolutifs (CNN)

Lorsque les données présentent une structure spatiale régulière, comme c'est le cas pour les images, il est avantageux d'exploiter cette structure à l'aide d'architectures spécialisées. Les réseaux de neurones convolutifs (CNN) ont été conçus pour tirer parti des invariances locales et des motifs récurrents présents dans les signaux visuels LeCun et al. (1998).

Une couche convolutionnelle effectue une opération discrète de convolution entre une image d'entrée X et un filtre W , définie par :

$$(X * W)(i, j) = \sum_{m=-k}^k \sum_{n=-k}^k X(i + m, j + n) W(m, n)$$

Cette opération permet l'extraction de caractéristiques locales, telles que les bords, textures ou motifs, tout en limitant le nombre de paramètres grâce au mécanisme de

partage des poids.

Les architectures modernes de CNN s'organisent généralement en blocs successifs composés de couches convolutionnelles, de fonctions d'activation non linéaires, d'opérations de sous-échantillonnage (*pooling*), puis d'un ou plusieurs modules entièrement connectés en sortie.

L'architecture utilisée dans le cadre de ce travail repose sur trois blocs convolutionnels, définis comme suit :

- Chaque bloc applique une convolution bidimensionnelle suivie d'une activation ReLU et d'un max-pooling, permettant de réduire progressivement la résolution spatiale ;
- Une couche entièrement connectée terminale projette les cartes de caractéristiques vers un espace latent de dimension réduite, typiquement 16.

Cette architecture vise à extraire une représentation compacte et informative des images, facilitant ainsi l'apprentissage ultérieur des modèles à noyau.

Apprentissage des représentations latentes

Afin d'obtenir une représentation latente pertinente des images d'entrée, une phase de pré-entraînement supervisé est réalisée. L'architecture considérée est composée d'un encodeur convolutif, suivi d'une couche linéaire dense de prédiction. L'encodeur projette chaque image $x \in \mathbb{R}^{64 \times 64}$ dans un espace latent de dimension 16, noté $f_\theta(x) \in \mathbb{R}^{16}$, tandis que la couche de sortie prédit l'angle de rotation associé sous forme scalaire.

L'entraînement de ce réseau est effectué en mode supervisé, à l'aide d'une fonction de perte quadratique :

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - w^\top f_\theta(x_i) - b)^2$$

où $w \in \mathbb{R}^{16}$ et $b \in \mathbb{R}$ sont les paramètres de la couche linéaire de sortie. Une fois cette phase d'entraînement achevée, la couche de prédiction est supprimée, et seul l'encodeur convolutif est conservé. Ce dernier est ensuite utilisé pour fournir les entrées latentes aux modèles hybrides, tels que DKL et SVDKL.

Cette procédure permet de fournir une initialisation informative et stable à l'encodeur, évitant ainsi les phénomènes d'effondrement des représentations (*representation collapse*), qui se traduisent par une convergence des vecteurs latents vers un point unique. Un tel effondrement conduit à une matrice de covariance dégénérée dans les modèles à noyau, rendant leur entraînement instable. La stratégie de pré-entraînement constitue donc une étape essentielle pour garantir la qualité de la représentation initiale et la bonne convergence des modèles probabilistes profonds.

7.4 Optimisation

L'optimisation des modèles considérés repose sur des techniques adaptées à la nature et à la structure de chaque approche. Deux grandes familles de méthodes sont employées : d'une part, les méthodes de type quasi-Newton, utilisées dans le cadre des modèles à processus gaussiens exacts ; d'autre part, les méthodes de descente de gradient stochastique adaptatives, appliquées aux modèles variationnels et hybrides.

Méthode quasi-Newton : L-BFGS

Dans le cas du processus gaussien exact (GP), l'entraînement du modèle consiste à maximiser la vraisemblance marginale, définie par :

$$\mathcal{L}(\theta) = \log p(y|X, \theta) = -\frac{1}{2}y^\top K_\theta^{-1}y - \frac{1}{2}\log |K_\theta| - \frac{n}{2}\log(2\pi),$$

où θ désigne l'ensemble des hyperparamètres du noyau, incluant notamment la longueur caractéristique, la variance du signal et la variance du bruit. Ce problème d'optimisation, de dimension modérée, est résolu à l'aide de la méthode L-BFGS (*Limited-memory Broyden-Fletcher-Goldfarb-Shanno*) (Wright, 2006), qui approxime la matrice hessienne H_k^{-1} à partir d'itérations précédentes, selon la mise à jour suivante :

$$\theta_{k+1} = \theta_k - \eta_k H_k^{-1} \nabla_\theta \mathcal{L}(\theta_k)$$

avec :

$$s_k = \theta_{k+1} - \theta_k, \quad y_k = \nabla \mathcal{L}(\theta_{k+1}) - \nabla \mathcal{L}(\theta_k), \quad \rho_k = \frac{1}{y_k^\top s_k}, \quad V_k = I - \rho_k y_k s_k^\top$$

et :

$$H_{k+1}^{-1} = V_k^\top H_k^{-1} V_k + \rho_k s_k s_k^\top$$

Cette méthode assure une convergence rapide vers un optimum local, tout en conservant une bonne stabilité numérique, à condition que le nombre d'hyperparamètres reste raisonnable.

Méthode de descente de gradient adaptative : Adam

Pour les modèles variationnels (SGP, SVGP) et hybrides (DKL, SVDKL), l'espace des paramètres est de dimension élevée en raison de la présence simultanée des poids du réseau neuronal, des points d'induction et des paramètres variationnels. Dans ce contexte, une optimisation stochastique par mini-batches est requise. L'algorithme Adam (*Adaptive Moment Estimation* (Kingma and Ba, 2014)) est utilisé pour sa robustesse et sa capacité à adapter dynamiquement les taux d'apprentissage. Les moments du gradient sont estimés selon les formules suivantes :

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_\theta \mathcal{L}(\theta_t) \tag{9}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) (\nabla_\theta \mathcal{L}(\theta_t))^2 \tag{10}$$

Ces quantités sont ensuite corrigées pour compenser le biais initial :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (11)$$

La mise à jour des paramètres s'effectue alors via la règle suivante :

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon}$$

Les hyperparamètres employés dans ce projet sont $\beta_1 = 0,9$, $\beta_2 = 0,999$, et $\varepsilon = 10^{-8}$.

Pré-entraînement et stabilisation pour DKL et SVDKL

L'entraînement conjoint des modèles hybrides profonds, tels que DKL et SVDKL, présente des difficultés particulières, notamment en raison des interactions complexes entre le réseau de neurones et le processus gaussien. Afin de stabiliser l'optimisation, une phase de pré-entraînement est d'abord réalisée : le réseau de neurones convolutif est initialement entraîné de manière supervisée sur la tâche de régression à l'aide d'une simple couche linéaire de sortie. Cette étape vise à produire une représentation latente informative et non dégénérée.

Une fois cette phase terminée, la couche linéaire de sortie est supprimée et l'ensemble du modèle (réseau convolutif et processus gaussien) est entraîné conjointement. Deux taux d'apprentissage distincts sont utilisés afin de favoriser la stabilité de l'optimisation : un taux plus faible pour les poids du réseau neuronal $\eta_{NN} = 10^{-3}$ et un taux plus élevé pour les hyperparamètres du processus gaussien $\eta_{GP} = 10^{-2}$. Cette différenciation permet de :

- préserver la stabilité des représentations extraites par le réseau convolutif ;
- accélérer la convergence des composantes gaussiennes du modèle, améliorant ainsi la précision prédictive.

7.5 Différentiation automatique

L'optimisation des modèles considérés dans cette étude repose sur l'évaluation efficace des gradients des fonctions objectifs, telles que la vraisemblance marginale ou l'ELBO, par rapport aux paramètres du modèle. Dans ce cadre *automatic differentiation* (AD), constitue une approche centrale. Elle permet de calculer numériquement les dérivées exactes de fonctions complexes, représentées comme des compositions d'opérations élémentaires.

Principe général

La différentiation automatique repose sur la règle de la chaîne appliquée à des fonctions composées. Soit une fonction $f : \mathbb{R}^n \rightarrow \mathbb{R}$ décomposée comme une composition de fonctions élémentaires :

$$f = f_K \circ f_{K-1} \circ \dots \circ f_1$$

où chaque fonction intermédiaire $f_k : \mathbb{R}^{d_k} \rightarrow \mathbb{R}^{d_{k+1}}$ est différentiable. La dérivée de f par rapport à l'entrée x est donnée par le produit successif des matrices jacobiniennes des fonctions intermédiaires :

$$\nabla_x f(x) = J_{f_K}(f_{K-1}(\dots f_1(x))) \cdot J_{f_{K-1}}(f_{K-2}(\dots f_1(x))) \dots J_{f_1}(x)$$

où chaque $J_{f_k}(x)$ désigne la matrice jacobienne de f_k en x , i.e. :

$$J_{f_k}(x) = \frac{\partial f_k(x)}{\partial x} \in \mathbb{R}^{d_{k+1} \times d_k}$$

Modes de différentiation automatique

Deux modes principaux de différentiation automatique sont utilisés en pratique, selon la direction de propagation des dérivées.

Mode avant (*forward mode*) Le mode avant calcule les dérivées en propageant les perturbations des entrées vers la sortie, dans l'ordre naturel des opérations :

$$\frac{\partial y}{\partial x} = J_{f_K}(f_{K-1}(\dots f_1(x))) \dots J_{f_2}(f_1(x)) J_{f_1}(x)$$

Ce mode est particulièrement adapté lorsque le nombre d'entrées est faible.

Mode inverse (*reverse mode*) Le mode inverse, ou rétropropagation (*backpropagation*), propage les dérivées en sens inverse, depuis la sortie vers les entrées :

$$\frac{\partial y}{\partial x} = J_{f_1}(x)^\top J_{f_2}(f_1(x))^\top \dots J_{f_K}(f_{K-1}(\dots f_1(x)))^\top$$

Le coût de cette méthode dépend du nombre de sorties (souvent égal à 1 pour une fonction de perte scalaire), et non du nombre d'entrées, ce qui la rend particulièrement adaptée à l'optimisation de modèles à grand nombre de paramètres.

Graphe computationnel dynamique dans PyTorch

Dans ce projet, la bibliothèque **PyTorch** est utilisée pour effectuer la différentiation automatique en mode inverse, en s'appuyant sur un graphe computationnel dynamique. Lors de l'évaluation de la fonction $f_\theta(x)$, un graphe orienté acyclique est construit implicitement, dans lequel :

- les nœuds correspondent aux variables intermédiaires issues d'opérations élémentaires (produits, sommes, fonctions non linéaires) ;
- les arcs représentent les dépendances entre ces opérations.

Par exemple, pour la fonction :

$$f(\theta, x) = \theta_1 \cdot x^2 + \theta_2 \cdot \sin(x)$$

le graphe computationnel sous-jacent se décompose en étapes intermédiaires :

$$\begin{aligned}
v_1 &= x^2 && \rightarrow \frac{\partial v_1}{\partial x} = 2x \\
v_2 &= \sin(x) && \rightarrow \frac{\partial v_2}{\partial x} = \cos(x) \\
f &= \theta_1 \cdot v_1 + \theta_2 \cdot v_2 && \rightarrow \frac{\partial f}{\partial v_1} = \theta_1, \quad \frac{\partial f}{\partial v_2} = \theta_2
\end{aligned}$$

La rétropropagation calcule alors la dérivée totale :

$$\frac{\partial f}{\partial x} = \theta_1 \cdot 2x + \theta_2 \cdot \cos(x)$$

Ce processus est entièrement automatisé par **PyTorch**. Lors de la passe directe (*forward pass*), les dérivées locales sont enregistrées. Lors de la passe inverse (*backward pass*), ces dérivées sont utilisées pour propager efficacement les gradients depuis la fonction de perte jusqu'aux paramètres du modèle.

Application à l'optimisation des modèles

La différentiation automatique permet l'optimisation conjointe des réseaux de neurones et des composantes gaussiennes en garantissant un calcul précis et différentiable des gradients. Cette approche autorise l'utilisation d'optimiseurs tels qu'Adam dans un cadre unifié, tout en assurant la stabilité et la rapidité de convergence nécessaires à l'entraînement de modèles à forte expressivité, comme les approches DKL et SVDKL.

7.6 Résultats

Les performances des modèles sont évaluées selon deux critères : l'erreur quadratique moyenne (RMSE) sur l'ensemble de test, et la qualité de l'incertitude prédictive, mesurée via la variance marginale fournie par les modèles à noyau. Les résultats numériques sont présentés dans le tableau 1.

| Modèle | RMSE |
|--------|-------|
| GP | 3.476 |
| NN | 2.570 |
| SGP | 4.775 |
| SVGP | 5.886 |
| DKL | 1.687 |
| SVDKL | 1.783 |

Table 1: Erreur quadratique moyenne des modèles évalués sur l'ensemble de test.

Le tableau indique que les approches hybrides combinant un réseau neuronal profond à un processus gaussien (DKL, SVDKL) surpassent les autres modèles en termes de précision prédictive. Le modèle DKL obtient notamment la meilleure performance avec une RMSE de 1.687, suivi de près par SVDKL. À l'inverse, les modèles variationnels sans encodeur neuronal préalable (SGP, SVGP) montrent une dégradation notable des performances.

La figure 1 illustre les prédictions obtenues sur l'ensemble de test, avec les intervalles de confiance à 95 % associés à chaque prédiction. Ces visualisations mettent en évidence des différences marquées dans la calibration des incertitudes entre les modèles.

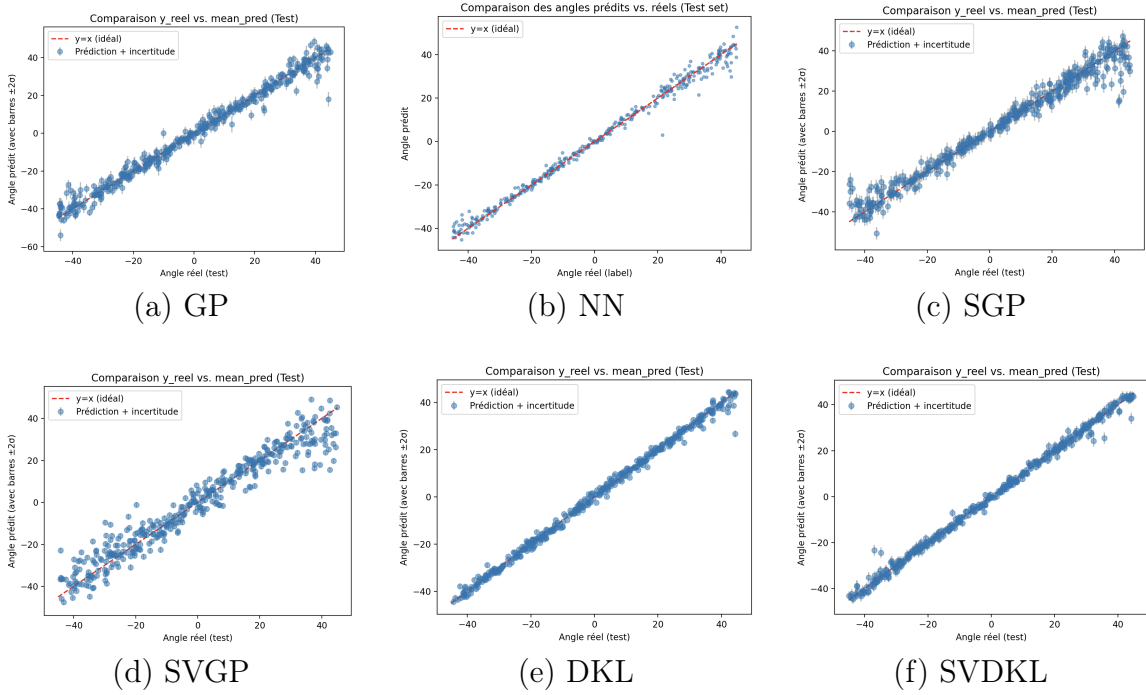


Figure 1: Comparaison visuelle des prédictions et des intervalles de confiance à 95 % pour les différents modèles évalués.

Bien que les approches DKL et SVDKL offrent des prédictions précises, les intervalles de confiance affichés semblent sous-estimer l'incertitude réelle. Ce phénomène, déjà souligné dans la littérature (Foong et al., 2019; Wilson et al., 2016b), s'explique par la structure même des DKL : le processus gaussien opère dans un espace latent appris, souvent trop contracté, ce qui entraîne une surestimation de la confiance prédictive, en particulier loin des points d'entraînement.

La qualité prédictive ne suffit donc pas à garantir une bonne calibration, et des stratégies complémentaires, telles que la régularisation de l'espace latent ou la calibration a posteriori, peuvent être envisagées pour corriger ce biais. Une évaluation plus poussée sur des tâches synthétiques avec incertitude contrôlée est également nécessaire pour confirmer ces observations.

8 Conclusion

Ce rapport souligne de manière approfondie l'efficacité des approches hybrides associant processus gaussiens et réseaux de neurones pour traiter des ensembles de données complexes et volumineux. L'analyse du krigeage, des méthodes bayésiennes paramétriques et de l'inférence variationnelle a permis d'identifier les forces des approches classiques ainsi que leurs limites, notamment en termes de complexité computationnelle et de gestion de l'incertitude. En combinant des techniques d'approximation, telles que l'utilisation de points induits, avec des stratégies d'optimisation avancées, l'étude révèle qu'il est possible de réduire significativement le coût de calcul tout en maintenant une précision élevée dans la quantification de l'incertitude.

Le passage aux méthodes de Deep Kernel Learning représente une avancée notable, en offrant une capacité accrue à modéliser des relations non linéaires complexes grâce à la transformation des données dans un espace latent adapté. Cette démarche permet d'exploiter pleinement les atouts des réseaux de neurones dans l'apprentissage de représentations pertinentes, tout en intégrant la robustesse des processus gaussiens pour l'estimation des incertitudes.

Par ailleurs, l'examen détaillé des méthodes d'inférence variationnelle met en évidence la nécessité d'une approche flexible dans l'optimisation des distributions postérieures. L'utilisation de techniques telles que la descente de gradient adaptée à la géométrie de l'espace des distributions ouvre des perspectives prometteuses pour le traitement de problèmes à haute dimensionnalité, tout en offrant des solutions pour contourner l'intractabilité du calcul exact de l'évidence.

L'ensemble des résultats présentés dans ce travail ouvre la voie à de nouvelles pistes de recherche, notamment en ce qui concerne le développement de modèles encore plus scalables et adaptés aux défis actuels de l'apprentissage automatique.

References

- Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2000.
- David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.
- Isabel Beichl and Francis Sullivan. The metropolis algorithm. *Computing in Science & Engineering*, 2(1):65–69, 2000.
- David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- George EP Box and George C Tiao. *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011.
- Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014.
- Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.
- Stéphane d’Ascoli, Maria Refinetti, and Giulio Biroli. Optimal learning rate schedules in high-dimensional non-convex optimization problems. *arXiv preprint arXiv:2202.04509*, 2022.
- Biswa Nath Datta. *Numerical linear algebra and applications*. SIAM, 2010.
- Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.
- John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.
- David Duvenaud, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, pages 1166–1174. PMLR, 2013.
- Runa Eschenhagen, Alexander Immer, Richard Turner, Frank Schneider, and Philipp Hennig. Kronecker-factored approximate curvature for modern neural network architectures. *Advances in Neural Information Processing Systems*, 36:33624–33655, 2023.
- Andrew YK Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E Turner. ‘in-between’ uncertainty in bayesian neural networks. *arXiv preprint arXiv:1906.11537*, 2019.

- Alan E Gelfand. Gibbs sampling. *Journal of the American statistical Association*, 95 (452):1300–1304, 2000.
- Mark Girolami and Ben Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(2):123–214, 2011.
- W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 1970.
- James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.
- Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.
- Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Pierre Simon Laplace. *Théorie analytique des probabilités*. Courcier, 1820.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- James Martens. New insights and perspectives on the natural gradient method. *Journal of Machine Learning Research*, 21(146):1–76, 2020.
- James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pages 2408–2417. PMLR, 2015.
- Georges Matheron. *Le krigeage universel*, volume 1. École nationale supérieure des mines de Paris Paris, 1969.
- Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- John L Nazareth. Conjugate gradient method. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(3):348–353, 2009.
- Victor M-H Ong, David J Nott, and Michael S Smith. Gaussian variational approximation with a factor covariance structure. *Journal of Computational and Graphical Statistics*, 27(3):465–478, 2018.
- Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 1999.

- Matthias W Seeger, Christopher KI Williams, and Neil D Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *International Workshop on Artificial Intelligence and Statistics*, pages 254–261. PMLR, 2003.
- Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.
- Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, 18, 2005.
- Michalis K Titsias. Variational model selection for sparse gaussian process regression. *Report, University of Manchester, UK*, 2009.
- M-N Tran, Nghia Nguyen, David Nott, and Robert Kohn. Bayesian deep net glm and glmm. *Journal of Computational and Graphical Statistics*, 29(1):97–113, 2020.
- Minh-Ngoc Tran, Trong-Nghia Nguyen, and Viet-Hung Dao. A practical tutorial on variational bayes. *arXiv preprint arXiv:2103.01327*, 2021.
- Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.
- C. Rasmussen & C. Williams. Gaussian processes for machine learning. *The MIT Press*, 2006.
- Andrew Wilson and Ryan Adams. Gaussian process kernels for pattern discovery and extrapolation. In *International conference on machine learning*, pages 1067–1075. PMLR, 2013.
- Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International conference on machine learning*, pages 1775–1784. PMLR, 2015.
- Andrew G Wilson, Zhiting Hu, Russ R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. *Advances in neural information processing systems*, 29, 2016a.
- Andrew Gordon Wilson, Christoph Dann, and Hannes Nickisch. Thoughts on massively scalable gaussian processes. *arXiv preprint arXiv:1511.01870*, 2015.
- Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016b.
- Stephen J Wright. Numerical optimization, 2006.
- Kaichao You, Mingsheng Long, Jianmin Wang, and Michael I Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.
- Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.