

PROJET S7

RÉGRESSION PAR PROCESSUS

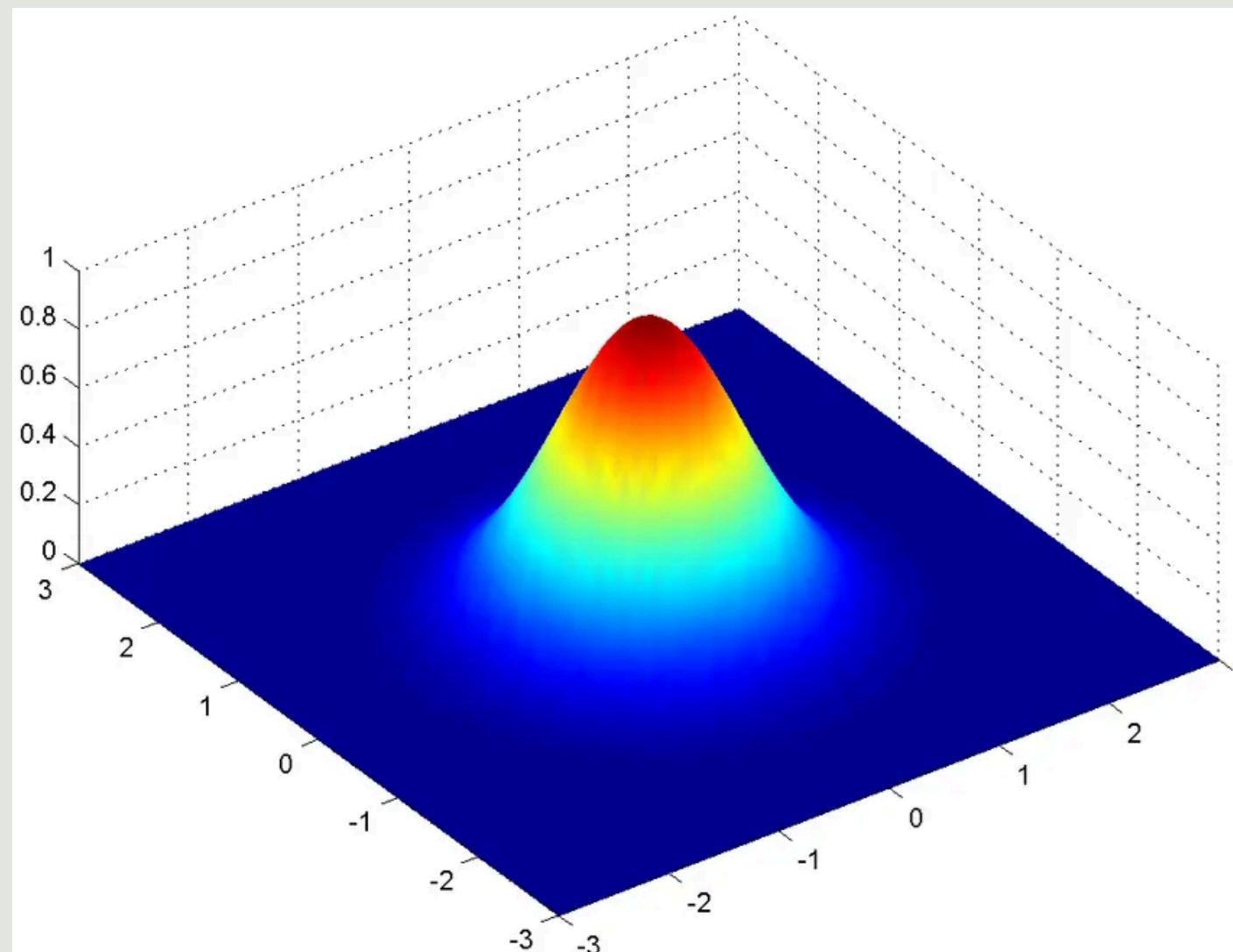
GAUSSIEN SCALABLE

Présenté par Thierry Zhang

SOMMAIRE

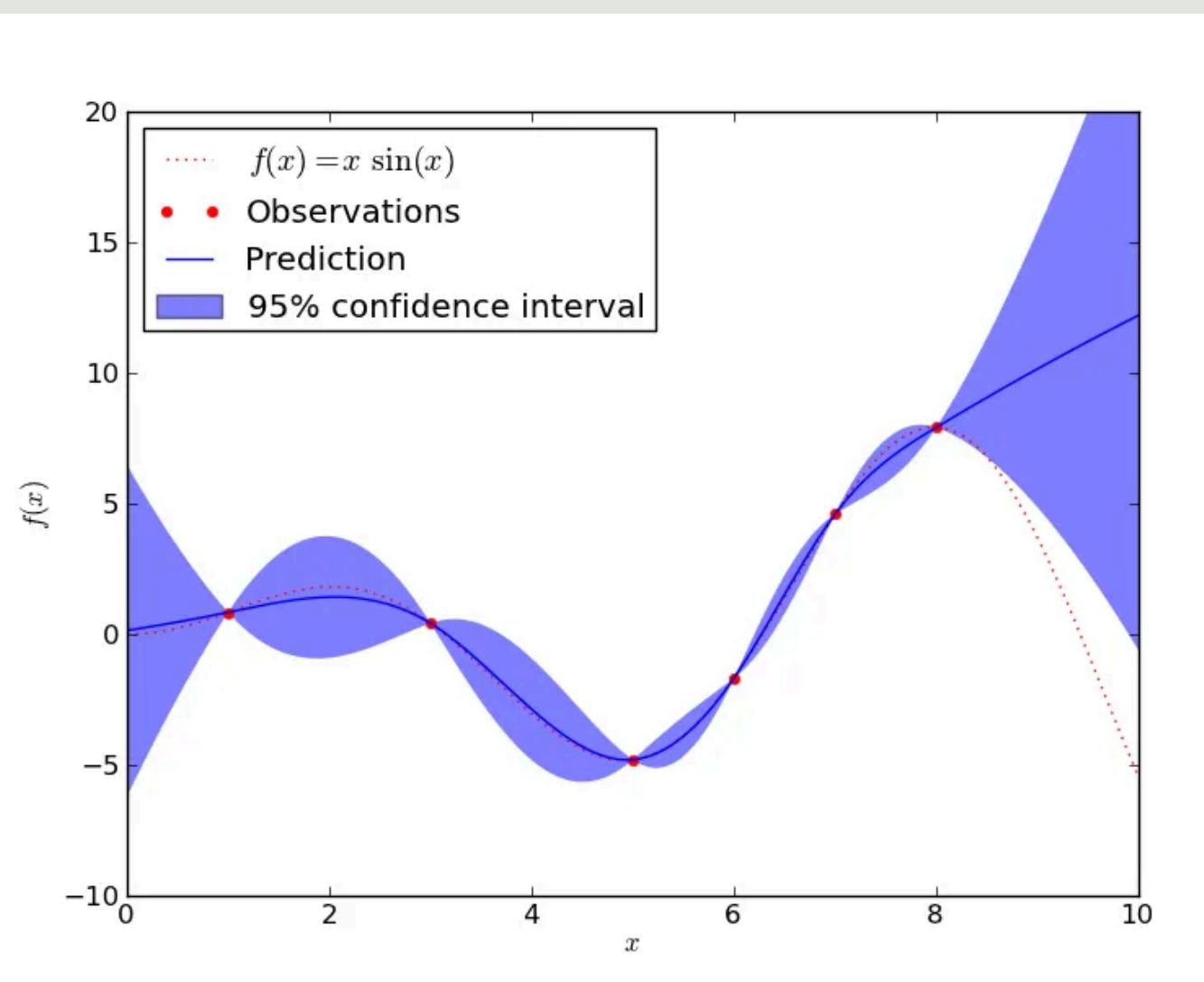
- Introduction
- Processus Gaussiens
- Régréssion par processus gaussiens
- Méthodes Scalables
- Resultats
- Conclusion

INTRODUCTION - RÉGRESSION PAR PROCESSUS GAUSSIEN



- **Méthode de régression bayésienne**
- **Fournit une estimation de la fonction cible avec une mesure d'incertitude.**
- **Utilisation de fonctions de covariance (kernels) pour capturer les relations entre les données.**

INTRODUCTION - RÉGRESSION PAR PROCESSUS GAUSSIEN



- Avantages:
Donne une prédiction et une incertitude
Peu sensible au sur apprentissage avec un bon
choix de kernel
- Inconvénient:
Complexité temporelle ($O(n^3)$)
Complexité Spatiale ($O(n^2)$)

INTRODUCTION - RÉGRESSION PAR PROCESSUS GAUSSIEN



Objectifs:

- Explorer les fondements théoriques de la régression par processus gaussiens (et comprendre le coût computationnel)
- Rechercher des méthodes permettant de réduire les coûts computationnels tout en gardant une précision satisfaisante
- Comparer les performances des différentes approches existantes

PROCESSUS GAUSSIENS

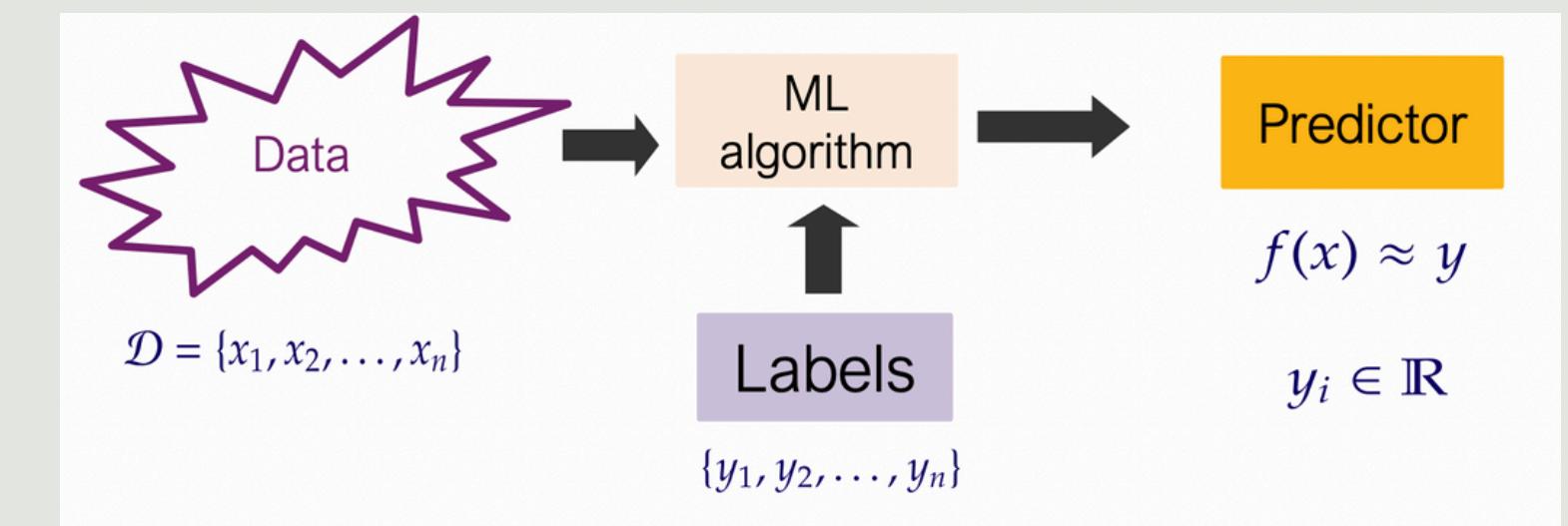
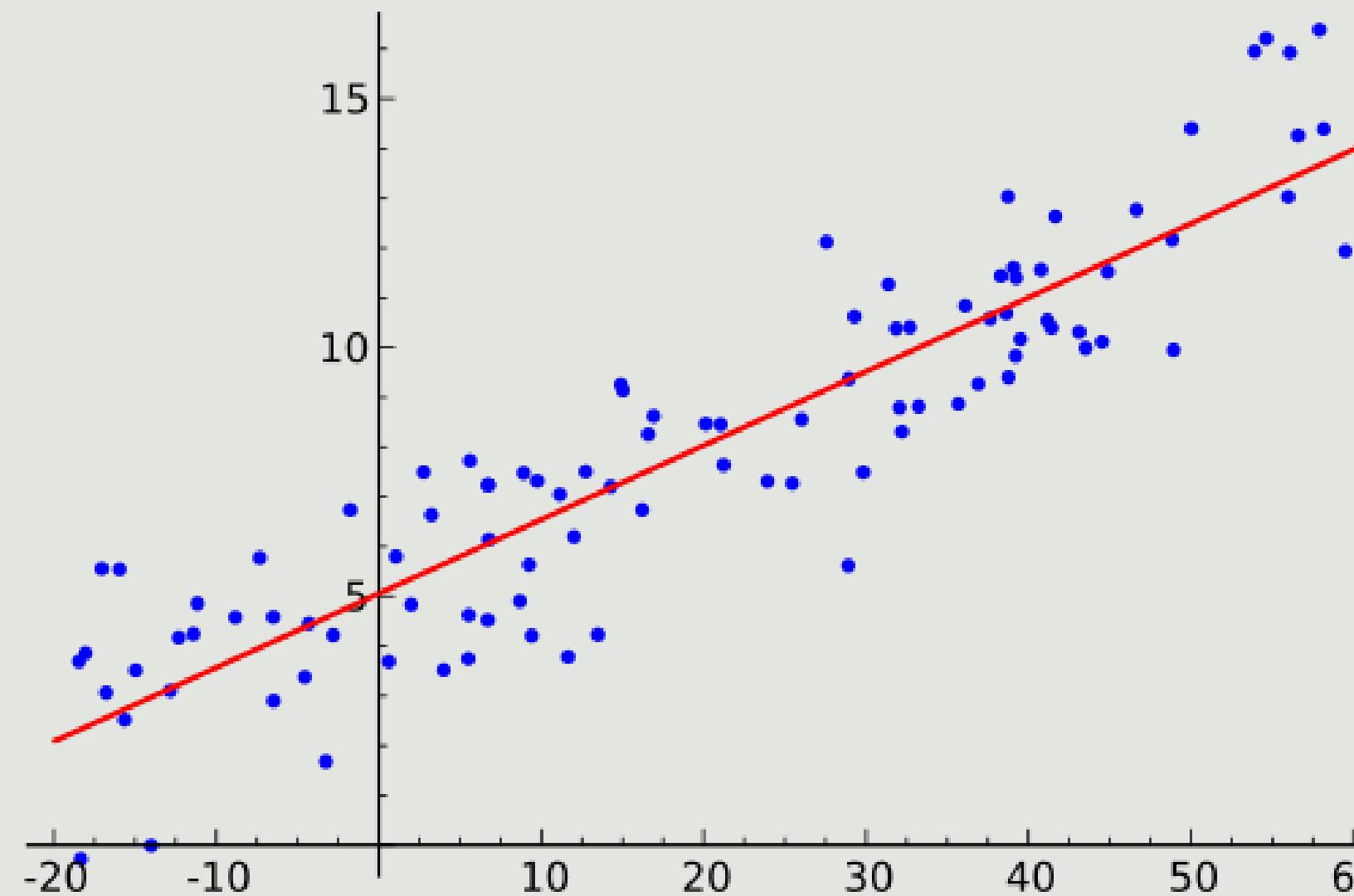
Définition: Un processus gaussien est un ensemble de variables aléatoires dont tout sous-ensemble fini suit une distribution conjointe gaussienne.

$$\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

Propriété: Un processus gaussien est entièrement caractérisé par :

- Sa fonction moyenne (mean function) : $m(t) = E[X_t]$,
- Sa fonction de covariance (kernel ou noyau) : $K(s, t) = \text{cov}(X_s, X_t)$

RÉGRESSION PAR PROCESSUS GAUSSIENS



RÉGRESSION PAR PROCESSUS GAUSSIENS

Cadre de la régression:

- $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, n\}$
- $y_i = f(x_i) + \varepsilon_i$

avec $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$

$f \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$

Choix de \mathbf{m} et de \mathbf{K} pour f :

- $\mathbf{m} = 0$
- \mathbf{K} kernel dépendant

d'hyperparamètres par exemple:

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

Finalement $\mathbf{Y} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$

RÉGRESSION PAR PROCESSUS GAUSSIENS

Training avec le dataset trouver les hyperparamètres θ dans K et σ_n

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

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

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

- **Complexité en $O(n^3)$** pour le calcul de l'inverse de K (par Cholesky par exemple)
- **Complexité en $O(n^3)$** pour la recherche des hyperparamètres

RÉGRESSION PAR PROCESSUS GAUSSIENS

Predicting

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

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

$$f \sim GP(\mu(\cdot), K(\cdot, \cdot))$$



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

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

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

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

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

- **Complexité:** $O(n)$ pour le calcul du vecteur moyenne
 $O(n^2)$ pour le calcul de la matrice de covariance

MÉTHODES SCALABLES

Sparse Gaussian Processes

Local Approximate Gaussian
Processes

Stochastic Variational Inference

SPARSE GAUSSIAN PROCESSES

Cadre de la régression:

- $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, n\}$
- $y_i = f(x_i) + \varepsilon_i$
avec $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$
 $f \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$

On a en plus M (< n) points d'induction

$$\mathcal{D} = \{(\bar{x}_i, \bar{y}_i) \mid i = 1, \dots, M\}$$

Choisis de manière random ou avec l'algorithme des k-mean

Avec un peu de théorie, on peut montrer que:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \sim \mathcal{N}(0, K_{\text{spgp}} + \sigma_n^2 I)$$

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

SPARSE GAUSSIAN PROCESSES

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

Même principe pour le training, trouver les hyperparamètres

Même principe pour la prédiction

Néanmoins l'inversion de K est beaucoup plus simple:

Complexité pour le training en

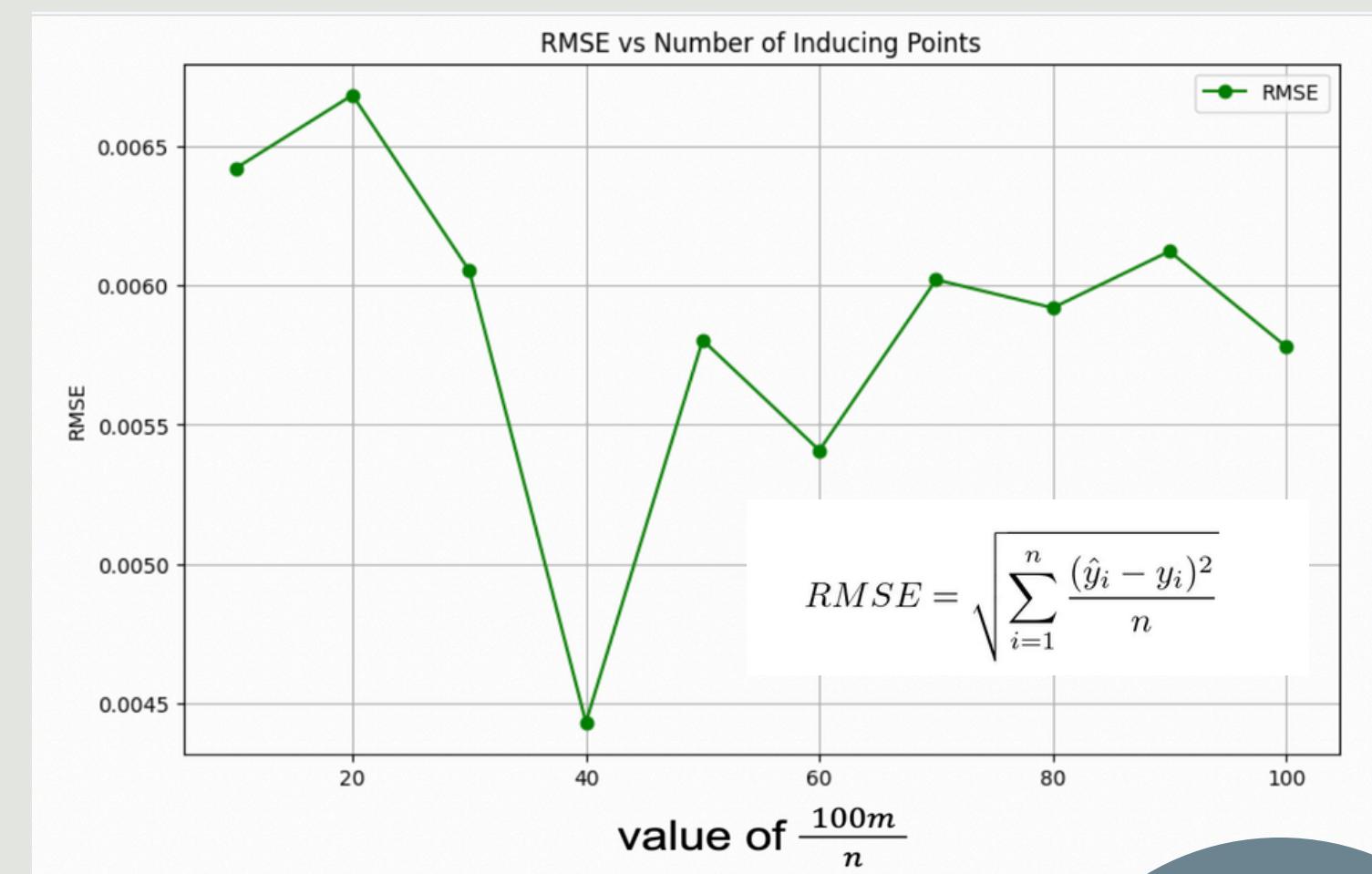
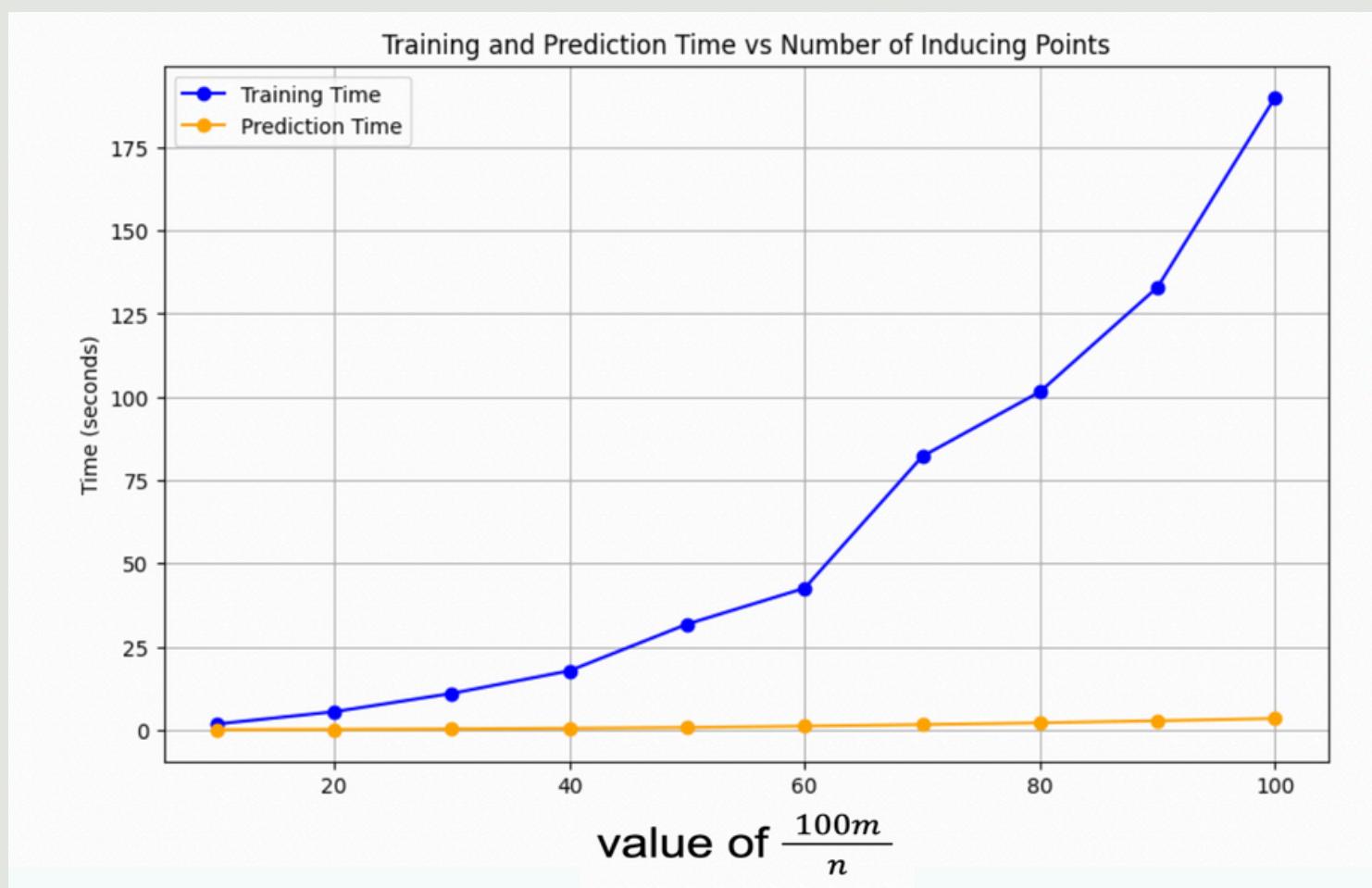
$$O(nM^2)$$

Complexité pour la prédiction en

$$O(M^2)$$

SPARSE GAUSSIAN PROCESSES NUMÉRIQUE

Influence du nombre de points induits M Choisis de manière random



SPARSE GAUSSIAN PROCESSES - NUMÉRIQUE

Résultats sur des datasets réels

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

LOCAL APPROXIMATE GP

Cadre de la régression:

- $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, N\}$
- $y_i = f(x_i) + \varepsilon_i$
avec $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$
- $f \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$

Trouver un sous ensemble

$$D_n(x) \subseteq D_N(x)$$

$$n < N$$

2 méthodes principales :

- ACL (Active Learning Cohn) : repose sur la minimisation de la variance prédictive à un point x
- KNN (k plus proches voisins)
- On utilise le sous Dataset pour le training
- Complexité pour l'ACL: $O(n^3)$
- Complexité pour NN: $O(N \log(n))$

LOCAL APPROXIMATE GP - NUMÉRIQUE

Résultats:

- Méthode ALC
Temps d'entraînement : 49.5 secondes (\downarrow 82.01% par rapport à GP).
RMSE : 295 (\uparrow 30.76% par rapport à GP)
- Méthode NN
Temps d'entraînement : 1.1 secondes (\downarrow 99.96% par rapport à GP).
RMSE : 242 (\uparrow 7.23% par rapport à GP).

STOCHASTIC VARIATIONAL INFERENCE

Cadre de la régression:

- $\mathcal{D} = \{(x_i, y_i) \mid i = 1, \dots, n\}$
- $y_i = f(x_i) + \varepsilon_i$
avec $\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$
- $f \sim \mathcal{N}(\mathbf{m}, \mathbf{K})$

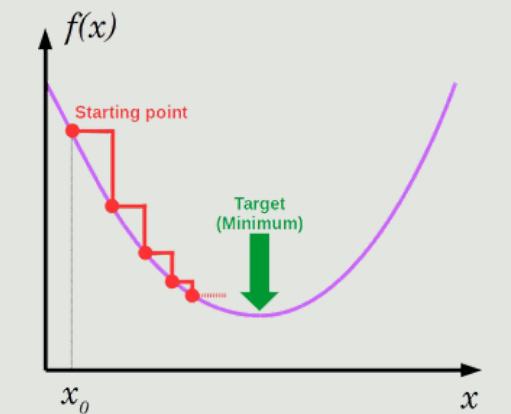
On a en plus M points d'induction globaux notés u

$$\mathcal{D} = \{(\bar{x}_i, \bar{y}_i) \mid i = 1, \dots, M\}$$

A maximiser

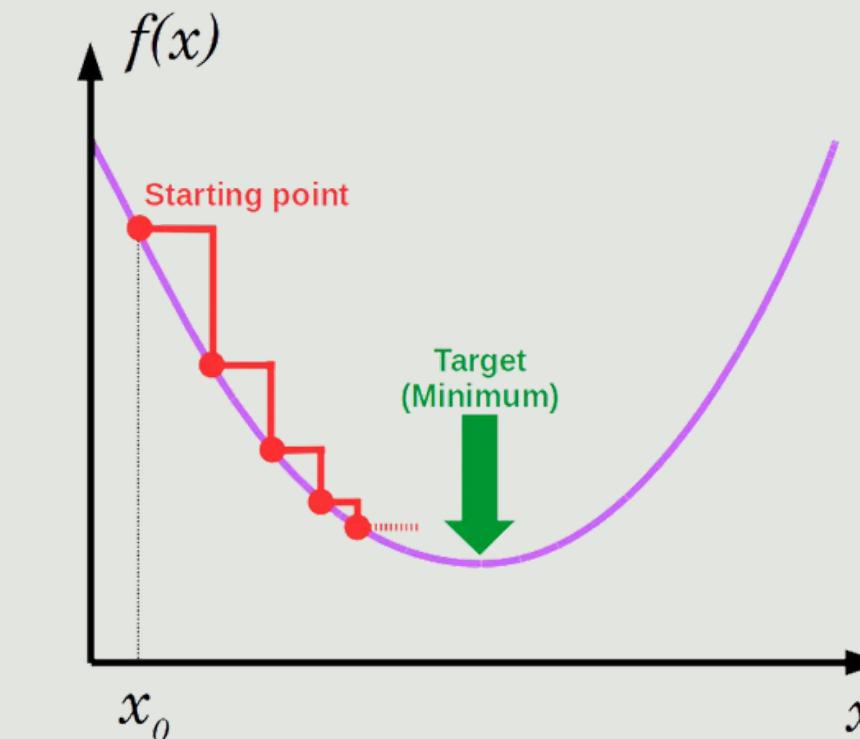
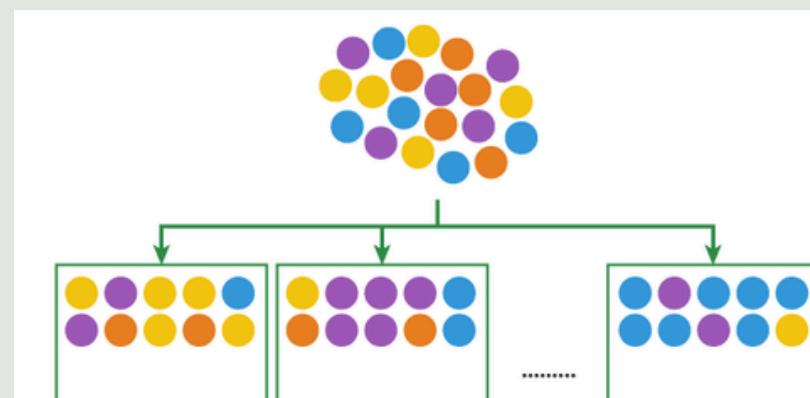
$$\log p(y|X) \geq \mathbb{E}_{q(u)} [\log p(y|u) + \log p(u) - \log q(u)]$$
$$q(u) = \mathcal{N}(u|m, S)$$

- Maximisation pendant l'entraînement par descente de gradient (le nombre d'itération est choisi par l'utilisateur)



STOCHASTIC VARIATIONAL INFERENCE

Séparation de D minibatch



$$\log p(y|X) \geq \mathbb{E}_{q(u)} [\log p(y|u) + \log p(u) - \log q(u)]$$

A maximiser

$$q(u) = \mathcal{N}(u|m, S)$$

Complexité pour le training $O(T(M^3 + BM^2))$

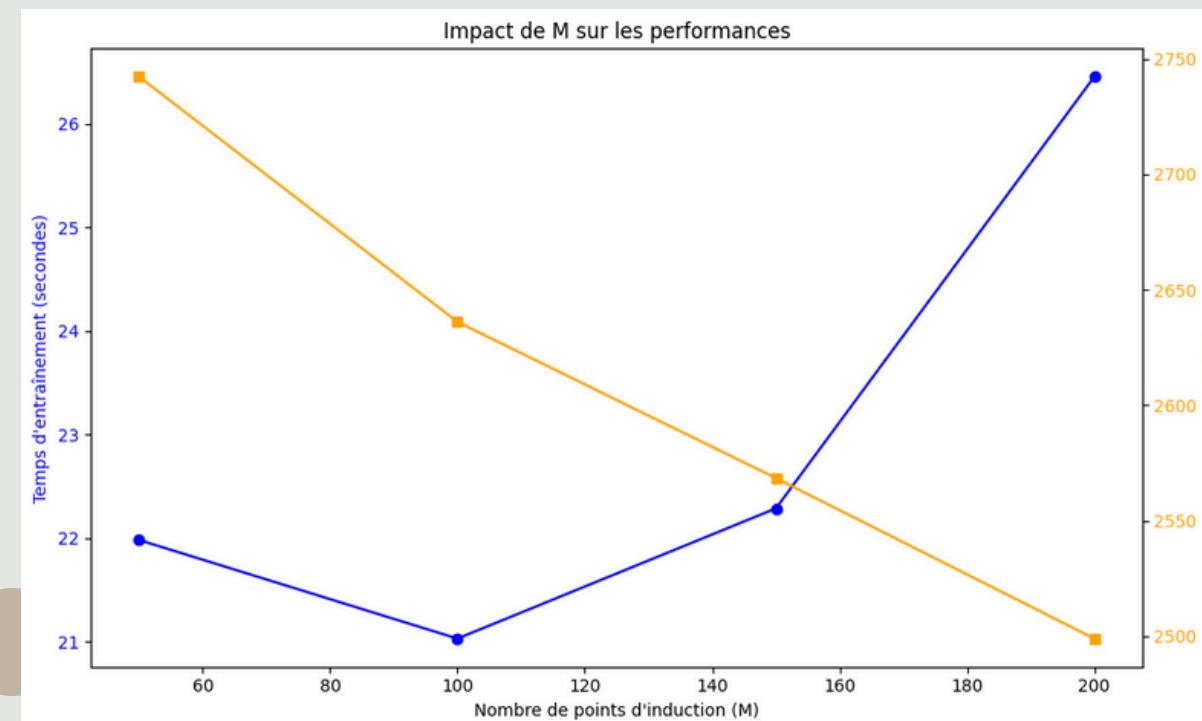
avec T le nombre d'itérations
et B le nombre de mini-batchs
et M le nombre de points induits

Pour la prédiction c'est comme avant

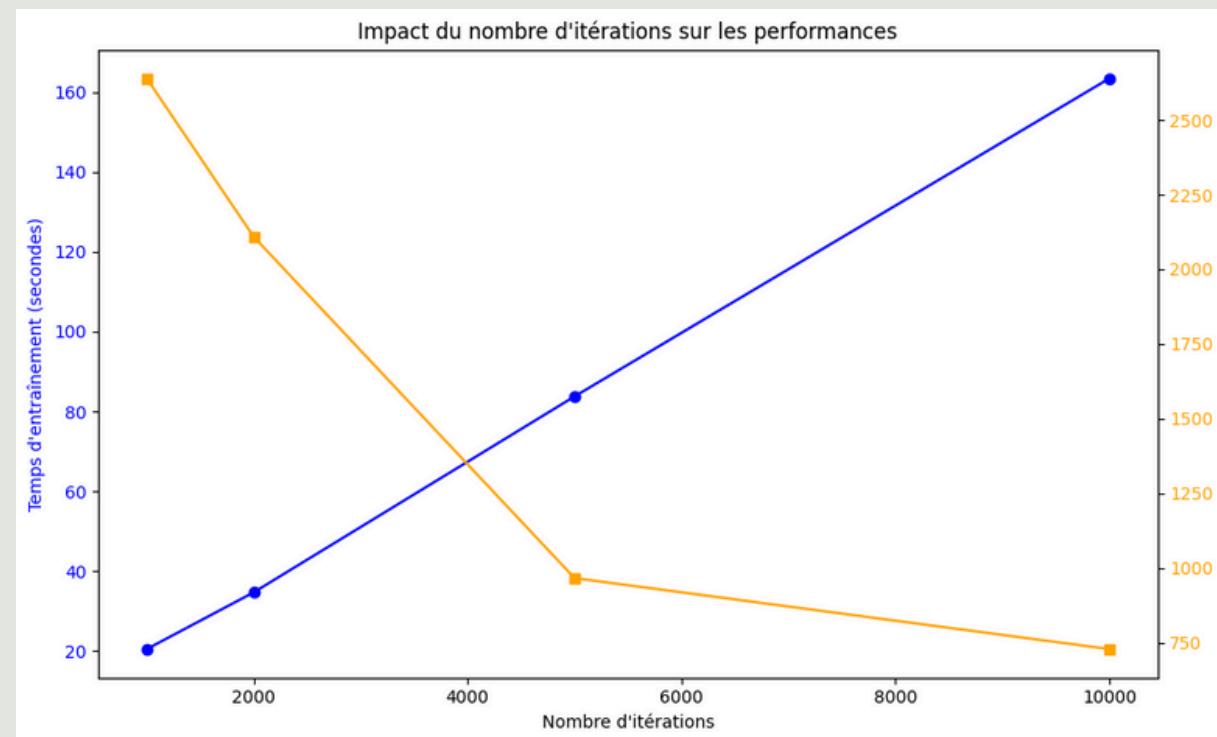
STOCHASTIC VARIATIONAL INFERENCE - NUMÉRIQUE

Impact des paramètres

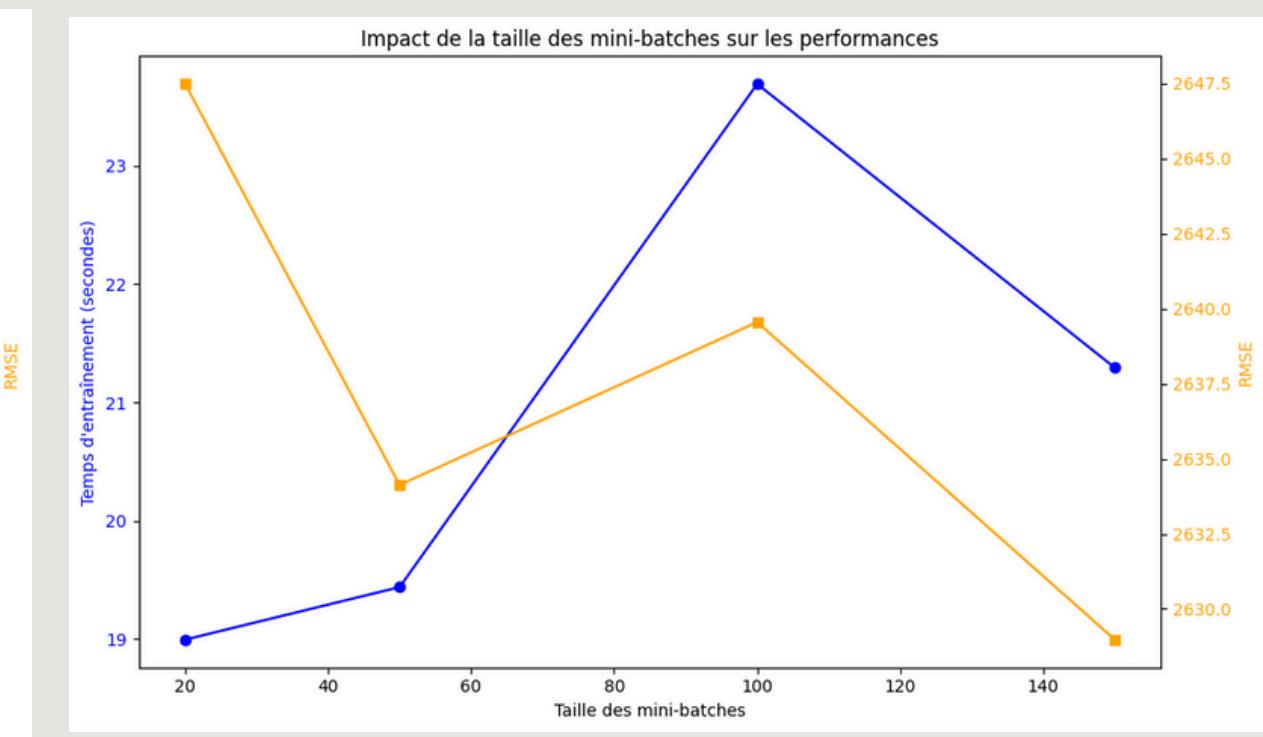
M



Nombre d'itérations



Nombre de minibatchs



STOCHASTIC VARIATIONAL INFERENCE - NUMÉRIQUE

Résultat sur le dataset Diamonds:
(M=100, avec 20 000 itérations avec une taille de mini-batch de 50)

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

RÉSULTATS FINAUX

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

CONCLUSION

- La régression par processus gaussien offre une précision supérieure aux méthodes de régression classiques
- Son principal inconvénient réside dans son coût computationnel élevé, qui peut limiter son utilisation sur de grands ensembles de données.
- Les méthodes scalables (SPGP, laGP, SVI) ont été développées pour pallier cette limitation :
- Elles permettent de réduire significativement les temps de calcul
- Elles maintiennent un niveau de performance et de précision satisfaisant

Merci
pour votre attention