Processus Stochastique III: Simulation de diffusion.

Rémi Vaucher

Doctorant, et prof à ses heures perdues
Laboratoire ERIC UR 3083

Un certain jour de l'année

Contents

Mouvement brownien en 1D

Avertissement: comme toute simulation numérique, il est impossible de simuler réellement le continu. Nous allons donc simuler pour une discrétisation temporelle donnée.

Le mouvement brownien (en dimensiion quelconque) est régi par une équation différentielle stochastique (simple)

$$dB(t) = \sigma\xi(t), \qquad \xi(t) \sim \mathcal{N}(0,1)$$

avec $\sigma > 0$ la variance souhaitée.

En utilisant le schéma d'Euler classique qui approxime $df(t)$ par $f(t+h) - f(t)$, on obtient le schéma suivant:

$$B_{t+1} = B_t + \xi, \qquad \xi \sim \mathcal{N}(0,1)$$

Ce schéma est un **processus de Markov** en temps discret!

Généralisation en haute dimension

Pour généraliser en haut dimension, il suffit d'appliquer le même schéma avec:

- $B_t \in \mathbb{R}^n$
- $\xi \sim \mathcal{N}(0, \Sigma)$, où $\Sigma \in \mathbb{R}^{n \times n}$ est une matrice diagonale définie positive (chaque coefficient de la diagonale représente la variance d'une dimension, généralement 1).

Le schéma d'Euler

On rappelle qu'une équation différentielle stochastique est une équation fonctionnelle de la forme:

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dB_t$$

Nous allons réaliser les approximations suivantes (pour un pas de temps h):

- $dX_t = X_t - X_{t-h}$
- $dt = h$
- $dB_t = B_t - B_{t-h} \sim \mathcal{N}(0, Id)$

Cela amène naturellement au schéma suivant:

$$X_t = X_{t-h} + h.b(t-h, X_{t-h}) + \sigma(t-h, X_{t-h})\xi, \qquad \xi \sim \mathcal{N}(0, Id)$$

Exemple: Equation de Langevin

On considère $b, \sigma > 0$. L'équation de Langevin est donnée par:

$$dV(t) = -bV(t) + \sigma dB(t)$$

Une solution de cette équation est appelée **processus d'Ornstein-Uhlenbeck**. Une solution explicite est donnée par:

$$V(t) = e^{-bt}V(0) + \int_0^t e^{-b(t-s)}\sigma dB(s)$$

De l'utilité de l'équation de Langevin

Propriété

On considère que $V(0) \sim \mathcal{L}_{V()}$ une loi quelconque. Alors, la suite (sous entendu continue ici) de variables aléatoire $V(t)$ **converge en loi** vers la loi normale $\mathcal{N}(0, \frac{\sigma^2}{2b}$

Approximation numérique

En utilisant le schéma d'Euler, on peut très facilement approximer une solution de l'équation de Langevin grâce au schéma suivant:

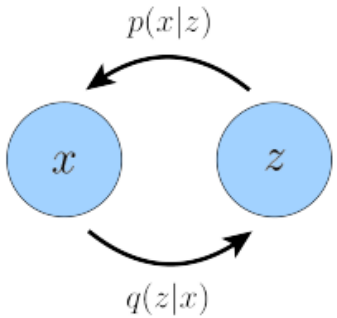
$$V(t) = (1-b)V(t-h) + \sigma\xi, \qquad \mathcal{N}(0,1)$$

La aussi, on se retrouve avec une chaîne de Markov. Autant il est intéressant de connaitre la variation du processus X au cours du temps, autant, il est encore plus intéressant (dans notre cas) de comprendre la variation de **la loi** de chaque étape X_t.

C'est sur ce principe que repose les algorithmes Génératifs par Modèle de Diffusion

Generative Diffusion Model: Les grands principe

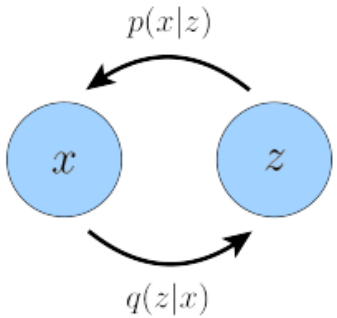
Pour bien comprendre le principe de la Stable Diffusion, il faut pratiquer un peu le bayésien. Dans l'illustration suivante, x et z représentent des étapes X_t et X_s:



- $p(x|z)$ désigne

Generative Diffusion Model: Les grands principe

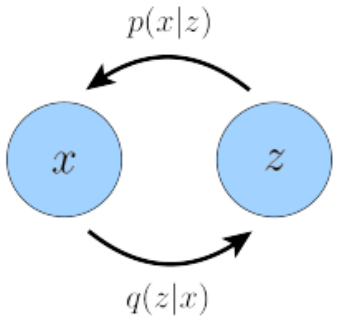
Pour bien comprendre le principe de la Stable Diffusion, il faut pratiquer un peu le bayésien. Dans l'illustration suivante, x et z représentent des étapes X_t et X_s:



- $p(x|z)$ désigne la loi de x conditionnellement à z.
- $q(z|x)$ désigne, sur le même principe, la loi de z conditionnellement à x.
- Si l'on se calque sur ce que l'on a vu en bayésien, cela représente

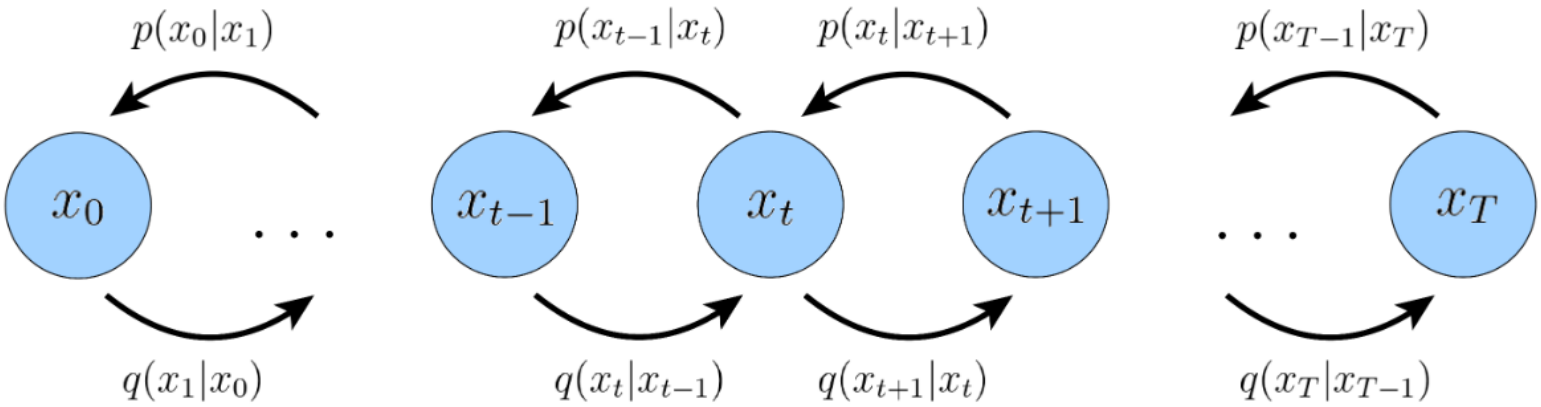
Generative Diffusion Model: Les grands principe

Pour bien comprendre le principe de la Stable Diffusion, il faut pratiquer un peu le bayésien. Dans l'illustration suivante, x et z représentent des étapes X_t et X_s:



- $p(x|z)$ désigne la loi de x conditionnellement à z.
- $q(z|x)$ désigne, sur le même principe, la loi de z conditionnellement à x.
- Si l'on se calque sur ce que l'on a vu en bayésien, cela représente les lois de **proposition** de x (resp. z) connaissant z (resp. x)

Maintenant, reprenons le contexte du processus stochastique, et appliquons ce principe à chaque étape:



Que peut on dire de la loi $p(x_T|x_0)$?

Si le processus étudié est un processus d'Ornstein-Uhlenbeck, alors nous savons! Pour T assez grand, x_T converge en loi vers une Gaussienne dépendante de b et σ.
Le principe du modèle génératif par diffusion est le suivant:

"*Considérant x_0 suivant une loi inconnu, il est possible de l'envoyer sur une loi connu par un processus de Langevin. En parcourant le même processus en temps renversé, il est possible de générer des données selon la loi de x_0.*"

L'algorithme "noising process"

Prenons une donnée x_0 (disons, une photo de votre petit chien/chat/cochon d'inde). Nous allons la bruiter de manière à suivre une dynamique de Langevin:

$$\begin{aligned} x_t &= \sqrt{\sigma} x_{t-1} + \sqrt{1-\sigma}\xi \\ &\sim \mathcal{N}(\sqrt{\sigma} x_{t-1}, (1-\sigma)I) \end{aligned}$$

Nous pouvons même exprimer x_t en fonction de x_0:

$$x_t = \sqrt{\prod_{i=0}^{t-1} \sigma_i x_0} + \sqrt{1 - \prod_{i=0}^{t-1} \sigma_i \xi}$$

$$\sim \mathcal{N}\left(\sqrt{\prod_{i=0}^{t-1} \sigma_i x_0}, 1 - \prod_{i=0}^{t-1} \sigma_i I\right) = \mathcal{N}(\sqrt{\overline{\sigma}_t} x_0, 1 - \overline{\sigma}_t I)$$

On obtient quelque chose comme ça:

Mais soyons clair: ceci est la partie facile.

The denoising process

L'objectif est très simple: le processus inverse **doit** être aussi régit par un mouvement Brownien.

Autant pour le bruitage, c'est simple, on génère x_t selon une loi normale centrée en $\overline{\sigma}_t x_0$. Facile.

Autant pour le débruitage, ça se complique. On va devoir générer x_{t-1} selon une loi normale centrée en... bah justement, on ne sait pas trop!

- C'est impossible de centrer en x_t, car on pourrais partir dans le mauvais sens
- Et en fait, on a pas d'autre choix possible...

Nous allons donc devoir générer x_{t-1} selon une loi $\mathcal{N}(\mu(x_t, t), \alpha^2 I)$

Problème: On ne connait pas $\mu(x_t, t)$.

De l'usage des réseaux de neurones.

C'est ici que l'on passe des probas/stats au machine learning.
Nous allons apprendre à un réseau de neurones à prédire $\mu(x_t, t)$ étant donné x_t et t.

De l'usage des réseaux de neurones.

C'est ici que l'on passe des probas/stats au machine learning. Nous allons apprendre à un réseau de neurones à prédire $\mu(x_t, t)$ étant donné x_t et t.

Enfin pas exactement. Je vous passe les détails (8 lignes de calculs) mais si

$$x_t = \sqrt{\overline{\sigma}_t} x_0 + \sqrt{1 - \overline{\sigma}_t} \xi_0$$

alors:

$$\mu(x_t, t) = \frac{1}{\sigma_t} x_t - \frac{1 - \sigma_t}{\sqrt{1 - \overline{\sigma_t}} \sqrt{\sigma_t}} \xi_0$$

Nous allons donc apprendre à notre réseau ξ_θ à prédire ξ_0

L'algorithme. Enfin.

Algorithm 1 Algorithme d'entrainement

1: Sélectionner $x_0 \sim p(x_0)$
2: Tirer aléatoirement $t \sim \mathcal{U}(\{1, ..., T\})$
3: Tirer aléatoirement $\xi_0 \sim \mathcal{N}(0, I)$
4: **while** `loss`$>$seuil **do**
5: Descente de gradient sur $\|\xi_0 - \xi_\theta(x_t, t)\|_2^2$
6: **end while**

Algorithm 2 Algorithme de génération

1: Tirer aléatoirement $x_T \sim \mathcal{N}(0, I)$
2: **for** $t \in \{T, ..., 1\}$ **do**
3: Si $t > 1$, tirer aléatoirement $z \sim \mathcal{N}(0, I)$, sinon $z = 0$.
4: $x_{t-1} = \frac{1}{\sigma_t}\left(x_t - \frac{1-\sigma_t}{\sqrt{1-\overline{\sigma_t}}}\xi_\theta(x_t, t\right) + \beta_t z$
5: **end for**

Pour plus d'infos

- Les premiers algos:
  Denoising Diffusion Probabilistic Models, Ho, Jain, Abbeel
  Denoising Diffusion Implicit Models, Song, Meng, Ermon.
- Pour mieux comprendre la théorie: *Understanding Diffusion Models: A Unified Perspective*, Luo
- Pour le choix des σ et autre hyperparamètres: *Elucidating the Design Space of Diffusion-Based Generative Models*, Kairas, Aittala, Aila, Laine