Spécifications techniques de « Hawkes_Process.py »

Introduction

Le but de ce document est d'expliqué le fonctionnement du fichier python « Hawkes_Process ».

Ce script vise à estimer les paramètres d'un processus de Hawkes à noyau gaussien en utilisant l'estimateur de maximum de vraisemblance.

On rappelle la formule d'un processus d'Hawkes :

$$\lambda^*(t) = \lambda(t) + \sum_{t_i<t} \mu(m_i, t - t_i)$$

Où :

- $\mu_{p,\beta}(m,t) = p * m * \beta * \exp(-\beta * t)$

Les paramètres à estimer sont p et β.

Architecture fonctionnelle du script « Hawkes_Process »

Le script est composé de 7 fonctions python :

1. MAP : qui permet d'estimer les valeurs de p et β via une méthode d'apostériori
2. loglikelihood : qui calcule la log-vraisemblance de notre modèle
3. prediction : qui calcule le nombre de retweet potentiel d'un tweet
4. identifiant_tweet : qui donne l'identifiant du tweet d'un message Kafka
5. time_series : qui donne la série partielle d'une cascade
6. simulate_marked_exp_hawkes_process : permettant de simuler la formule ci-dessus (cette fonction fait partie du test d'intégration, non utile pour la prediction et le calcul des paramètres du noyau gaussien)
7. neg_power_law : utilisé dans la fonction « simulate_marked_exp_hawkes_process » pour simplifier les calculs (non utile pour la prediction et le calcul des paramètres du noyau gaussien)

Une huitième fonction : « test_MLE » permet de vérifier que la fonction MLE fonctionne

Cette fonction fait office de test d'intégration, aucun test unitaire n'est réalisé dans ce projet. On considère que si le test d'intégration est OK, alors les différentes fonctions de ce script sont par défaut OK.

MAP

Variables d'entrées : • cascade : Processus de Hawkes où les paramètres sont à estimés • t : instant à partir duquel on veut estimer p et β • alpha • mu • init_params : valeurs initiales de p et β pour l'optimisation • max_n_star :
Variables de sorties : • -res.fun : valeur de la fonction d'objectif • res.x : valeur de p et β permettant de maximiser la vraisemblance
Algorithme :

1. On calcule les moments des priors
2. On applique la méthode des moments pour calculer l'espérance et la covariance pour p et β
3. On calcule l'inverse de la matrice de covariance
4. On utilise le module « minimize » de scipy.optimize pour maximiser la log-vraisemblance et obtenir les paramètres optimaux de p et β

loglikelihood

Variables d'entrées :
- p : probabilité qu'un follower retweet
- beta : temps de réponse du follower
- cascade
- t

Variable de sortie :
- La valeur de la log-vraisemblance à l'instant t d'un processus de Hawkes

Algorithme :
1. On vérifie les valeurs de p et β pour ne pas lever une exception
2. On calcule la valeur de la log-vraisemblance (formule annexe)

Prediction

Variables d'entrées :
- p
- β
- cascade
- alpha
- mu
- t

Variable de sortie :
- N : le nombre total, estimé de retweet que va générer un tweet

Algorithme :
1. On calcul la valeur de n*
2. On calcule la valeur de N (formule en annexe)

Identifiant_tweet

Variable d'entrée :
- String : un message kafka

Variable de sortie :
- Identifiant : l'identifiant du tweet initiale

Algorithme :
1. On fait une décomposition du message
2. On récupère une sous-chaîne de caractère qui est l'identifiant du tweet initiale

Time_series

Variable d'entrée :
- String : un message kafka

Variable de sortie :
- Identifiant : l'identifiant du tweet initiale

Algorithme :
1. On fait une décomposition du message
2. On récupère une sous-chaîne de caractère qui est la série partielle
3. On transforme cette str en une matrice de flottant

simulate_marked_exp_hawkes_process

Variables d'entrées : • p • beta • m0 : magnitude initiale du tweet • alpha : paramètre de la loi puissance négative • mu : valeur minimum la loi puissance
Variable de sortie : • Un tableau contenant les données d'un processus de Hawkes

neg_power_law

Variables d'entrées : • alpha • mu
Variable de sortie : • Valeur de la loi puissance négative

Annexe :

Log-vraisemblance :

$$L(\theta) = (n-1) * log(p * \beta) + \sum_{i=2}^{n} \log\left(\sum_{j=1}^{i-1} m_j * \exp\left(-\beta(t_i - t_j)\right)\right) - p * \sum_{i=1}^{n} m_i(1 - \exp(-\beta(t - t_i)))$$

Prediction :

$$N = n + \frac{p * \sum_{i=1}^{n} m_i * \exp\left(-\beta(t - t_i)\right)}{1 - p * \mu * \frac{\alpha - 1}{\alpha - 2}}$$