

Trading algorithmique sur cryptomonnaies

Conception d'un bot qui se connecte à Binance et effectue des ordres sur le Bitcoin et le XRP



BOUDJEMAÏ Illyas

NAJMAN Leonard

INTRODUCTION	1
I/ HISTOIRE DES TRADING BOTS	2
II/ LES TECHNOLOGIES UTILISÉS PAR UN ALGORITHMIC TRADING BOT	3
<u>DeepLearning</u>	<u>3</u>
L'architecture	3
Le processus d'apprentissage	4
<u>Algorithme Génétiques</u>	<u>4</u>
PARTIE PRATIQUE	4
CONCLUSION	4
REFERENCES	5

INTRODUCTION

L'automatisation décrit un large éventail de technologies qui réduisent l'intervention humaine dans les processus. L'intervention humaine est réduite par la prédétermination de critères de décision, de relations entre sous-processus et d'actions connexes, et par l'incorporation de ces pré déterminations dans des machines.

On peut détailler le fonctionnement des trading algorithmique en deux parties : le premier, la prise de décision fondée sur une stratégie et une appréciation des conditions du marché d'une part, la seconde, l'exécution de cette décision, qui va se traduire par l'émission d'ordres d'achat ou de vente vers une ou des plateformes de trading.

Un système de trading automatisé, un sous-ensemble de la trading algorithmique, utilise un programme informatique pour créer des ordres d'achat et de vente et soumettre automatiquement les ordres à un centre de marché ou à une bourse. Le programme informatique génère automatiquement des ordres sur la base d'un ensemble de règles prédéfinies en utilisant une stratégie de négociation basée sur une analyse technique, des calculs statistiques et mathématiques avancés ou des données provenant d'autres sources électroniques.

I/ HISTOIRE DES TRADING BOTS

L'origine des systèmes de négociation remonte à 1949, lorsque Richard Donchian a lancé Futures, Inc. l'un des premiers fonds de matières premières à capitaux publics, qui utilisait des règles fixes pour générer des signaux d'achat et de vente. Évidemment, sans Internet ni les ordinateurs..

L'idée de systèmes de trading basés sur des règles est devenue plus populaire parmi les traders dans les années 1980, lorsque des traders célèbres comme le trader tortue Richard Dennis et le propriétaire des Boston Red Sox John Henry ont commencé à appliquer des règles mathématiques d'entrée et de sortie aux marchés des matières premières.

L'avènement des contrats à terme du Chicago Mercantile Exchange à la fin des années 1990 a été la dernière impulsion pour que les systèmes de négociation entrent dans le

courant dominant, permettant aux négociants de contourner la salle des marchés avec des ordres acheminés vers une bourse électronique appelée Globex. Désormais, un ordinateur pouvait non seulement calculer où placer les ordres, mais aussi placer directement la transaction sur la bourse.

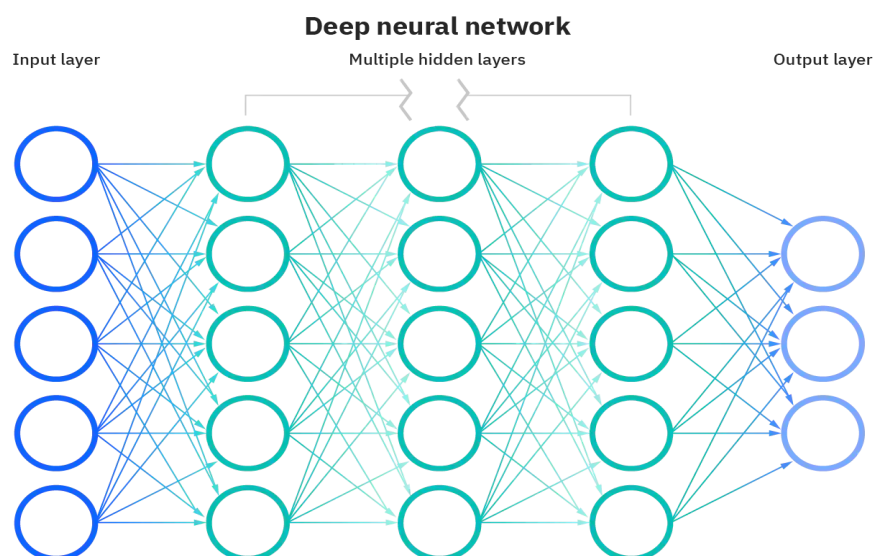
Les améliorations technologiques ont permis d'accroître l'accessibilité des investisseurs particuliers. Les premières formes de systèmes de négociation automatisés, des logiciels basés sur des algorithmes, ont été utilisées par les gestionnaires financiers et les courtiers. Ces types de logiciels étaient utilisés pour gérer automatiquement les portefeuilles des clients. Cependant, le premier service de marché libre sans aucune supervision a été lancé en 2008, il s'agit de Betterment par Jon Stein. Depuis lors, ce système s'est amélioré avec le développement de l'industrie informatique. Aujourd'hui, le système de négociation automatisée gère d'énormes actifs dans le monde entier. En 2014, plus de 75 % des actions négociées sur les bourses américaines provenaient d'ordres du système de négociation automatisée.

II/ LES TECHNOLOGIES UTILISÉS PAR UN ALGORITHMIC TRADING BOT

DeepLearning

L'architecture

Deep learning est une branche du machine learning qui utilise des réseaux neuronaux à plusieurs couches. Un réseau neuronal profond analyse les données à l'aide de représentations apprises, de la même manière qu'une personne examinerait un problème.



Comme on peut le voir dans le schéma, les neurones interconnectés à d'autres neurones forment un réseau. Les réseaux neuronaux peuvent être lus de gauche à droite. Ici, la première couche est la couche dans laquelle les entrées sont saisies. Il existe deux couches internes (appelées couches cachées) qui effectuent des calculs, et une dernière couche qui contient toutes les sorties possibles.

Le fonctionnement d'un réseau neuronal complet est simple : on entre des variables en entrée, et après quelques calculs, on obtient une sortie.

Le processus d'apprentissage

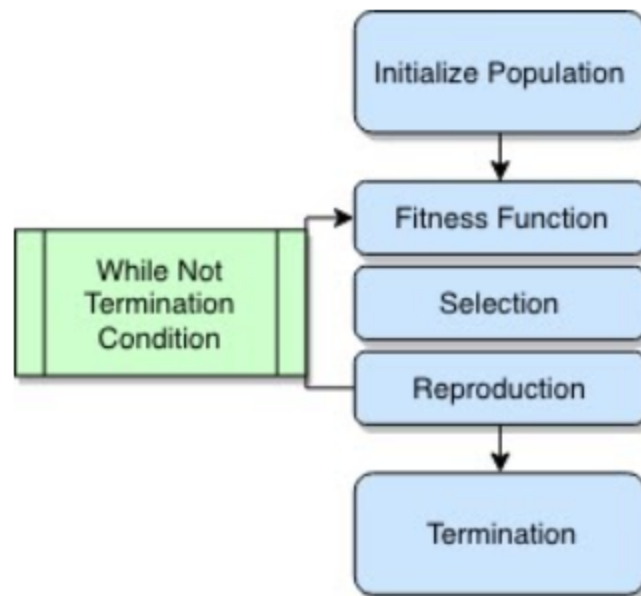
Tout d'abord, lorsqu'une entrée est donnée au réseau neuronal, il renvoie une sortie. Au premier essai, il ne peut pas obtenir la bonne sortie par lui-même et c'est pourquoi, pendant la phase d'apprentissage, chaque entrée est accompagnée de son nom, expliquant quelle sortie le réseau neuronal aurait dû deviner. Si le choix est le bon, les paramètres actuels sont conservés et l'entrée suivante est donnée. En revanche, si la sortie obtenue ne correspond pas à l'étiquette, les poids sont modifiés. Ce sont les seules variables qui peuvent être modifiées pendant la phase d'apprentissage.

Pour déterminer quel poids il est préférable de modifier, un processus particulier, appelé "backpropagation", est effectué.

Algorithme Génétiques

Architecture

Les algorithmes génétiques varient dans leur structure en fonction de leur objectif, mais ils ont tous quelques éléments en commun. L'algorithme commence par initialiser une population d'individus en utilisant des valeurs par défaut ou aléatoires. Ensuite, il fait passer chaque membre de cette population par une fonction d'adaptation. Il sélectionne les membres les plus aptes de la population pour les reproduire à l'aide d'une méthode définie dans la fonction de reproduction, puis répète l'évaluation et la reproduction jusqu'à ce qu'un nombre souhaité d'itérations se soit écoulé. À la fin, l'algorithme présente le ou les meilleurs membres de la population selon la fonction d'aptitude. Discutons plus en détail de chacun de ces concepts.



Il se divise en 4 fonctions:

Fitness function :

La fonction d'aptitude est le cœur d'un algorithme génétique. La fonction prend un individu et détermine dans quelle mesure il remplit les critères pour lesquels l'algorithme est optimisé.

Selection function:

La fonction de sélection prend la population et les résultats de la fonction d'aptitude pour déterminer qui doit se reproduire.

Reproduction function:

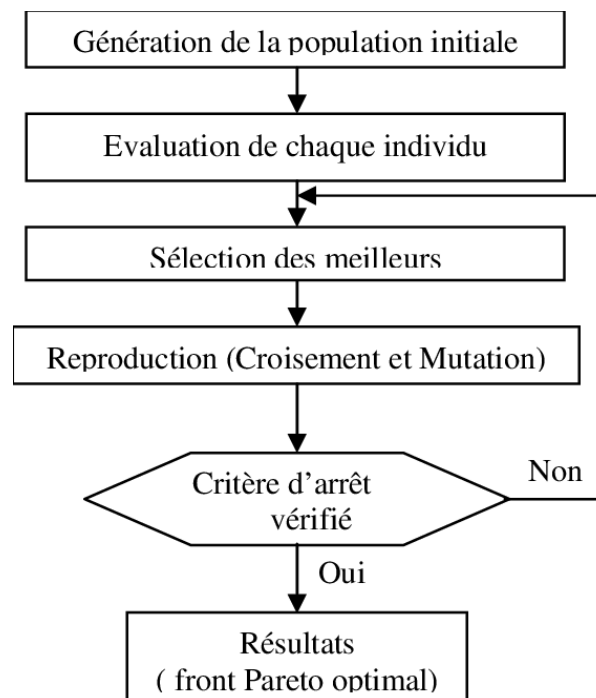
La fonction de reproduction détermine comment étendre la population en fonction des membres existants. La modification du comportement et des hyperparamètres de la fonction de reproduction est l'une des parties les plus complexes et les plus importantes de la création d'un algorithme génétique, car la fonction de reproduction détermine l'évolution de la population dans le temps.

Termination Function:

Une fois que les itérations souhaitées ont eu lieu, la fonction de terminaison prend la population finale et renvoie les meilleurs membres par score de fitness.

Le processus d'apprentissage

1. Une première population est créée aléatoirement.
2. On applique notre Fitness function afin d'attribuer une note à chaque individu.
3. Les individus présentant la meilleure note sur la population seront gardés pour la phase suivante.
4. Cette étape permet de générer une nouvelle population à partir des individus sélectionnés. Un croisement de gène est effectué entre deux individus afin de créer un « enfant ».
5. Ce processus est répété jusqu'à que l'évaluation soit satisfaisante.



PARTIE PRATIQUE

Run.cs

```
public static void Main(string[] args)
{
    //Démarrer Bot la première fois, avant le Timer
    AlgoTrading.Bot();

    //Debut Timer
    Timer timer = new Timer();
    timer.Interval = General.timerInterval * 60 * 1000; // Convertir ms en minutes
    timer.Elapsed += new ElapsedEventHandler(timer_Elapsed);
    timer.Enabled = true;

    timerFired.WaitOne();
}
```

Dans le main on commence par lancer l'algorithme de trading ,

```
public class AlgoTrading
{
    2 references
    public static void Bot()
    {
        string symbol = General.C1 + General.C2;

        //Obtenir les informations de compte
        var accInfo = API.GetAccountInformation();
    }
}
```

Dans **AlgoTrading.cs** l'algorithme va récupérer des informations du compte via l'API de la plateforme Binance.

```
public static bool Ping()
{
    string apiRequestUrl = $"{Url}v1/ping";

    string response = Server.WebRequest(apiRequestUrl, "GET", null);
    if (response == "{}")
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

Dans **API.cs** on va d'abord s'assurer que l'API est opérationnel et que l'on peut communiquer avec elle , une fois la vérification effectué on va envoyer des requêtes à l'API du site en utilisant deux clés (une privée et une public) pour que l'API puisse interagir avec notre compte de trading .

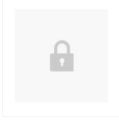
Liste des API

Supprimer toutes les API

Projet_tp3_API

Modifier les restrictions

Supprimer



API Key

dWmegTir831X0nvs0NGzA1vYIMBulwPNc31qF3nMlvTDCFqbelEq1gCshi0aAe9d

Copy

Secret Key

Restrictions API :

☒ Permettre la lecture
 ☒ Activer le trading au comptant et sur marge
 ☐ Activer les retraits

☐ Activer Marge
 ☐ Permet les transferts universels
 ☐ Activer les Options Vanilla

Restrictions d'accès IP :

☒ Sans restriction (moins sûr) Cette clé API permet l'accès à partir de n'importe quelle adresse IP. Ceci n'est pas recommandé.
 ☐ Restreindre l'accès aux adresses IP de confiance uniquement (recommandé)

Voyons maintenant comment ouvrir une position grâce à la fonction `PlaceMarketOrder` sur **API.css**

```
public static Order PlaceMarketOrder(string symbol, OrderSides side, double quantity)
{
    string apiRequestUrl = "";
    if (General.testCase == true)
        apiRequestUrl = $"{Url}v3/order/test";
    else
        apiRequestUrl = $"{Url}v3/order";

    string query = $"symbol={symbol}&side={side}&type={OrderTypes.MARKET}&quantity={quantity}";
    query = $"{query}&timestamp={Server.getTimeStamp()}";

    var signature = Server.getSignature(General.SecretKey, query);
    query += "&signature=" + signature;

    apiRequestUrl += "?" + query;
    var response = Server.webRequest(apiRequestUrl, "POST", General.ApiKey);

    var parsedResponse = JsonConvert.DeserializeObject<Order>(response);
    return parsedResponse;
}
```

Cette fonction place un ordre sur le marché, comme l'ordre est au prix actuel du marché il n'est pas nécessaire de lui envoyer en paramètre un prix. Elle prend donc en paramètre le symbole de la monnaie à trader, le sens (achat ou vente) ainsi que le montant à trader.

Le sens est défini grâce à la structure `OrderSides` contenus dans le Fichier **Info.css** :

8

```
public enum OrderSides
{
    4 references
    BUY,
    5 references
    SELL
}
```

Dans la fonction PlaceMarketOrder on cherche à obtenir l'url de la requête qui est composée d'une première demande qui fera appel aux paramètres passés dans la fonction .La 2ème partie de la requête est composée de l'Horodatage de la plateforme. Une fois que nous avons récupéré ces informations nous allons compléter l'Url avec la signature que l'on obtient du server grâce à la clé d'authentification secrète à l'aide de la fonction GetSignature contenu dans le fichier **Server.cs**:

```
public static string getSignature(string SecretKey, string query)
{
    Encoding encoding = Encoding.UTF8;
    var keyByte = encoding.GetBytes(SecretKey);
    using (var hmacsha256 = new HMACSHA256(keyByte))
    {
        hmacsha256.ComputeHash(encoding.GetBytes(query));
        return ByteToString(hmacsha256.Hash);
    }
}
```

Notre url pour la requête API est maintenant complète , nous pouvons donc l'envoyer sur le serveur API lié à notre compte grâce à la fonction webRequest contenu elle aussi dans **server.cs** . Cet url va placer un ordre sur le marché et ainsi réaliser l'action demandée . Une fois la requête envoyée, on s'assure que l'ordre a bien été exécuté en réceptionnant la réponse du serveur .

La réponse du serveur permet à l'algorithme de trading de nous informer si oui ou non l'ordre a été placé et lui permet d'agir en conséquence, par exemple si l'intervalle de temps pour placer un ordre a été dépassé alors l'algorithme ne retentera pas de placer le même ordre et poursuivra son analyse du marché .

CONCLUSION

Le trading algorithmique a ouvert une nouvelle ère pour les marchés, dont les avantages ne sont pas encore pleinement exploités. S'adapter à ce nouveau mode de négociation peut garantir de meilleurs résultats. Le trading algorithmique est désormais un "prérequis" pour survivre sur les marchés financiers de demain, car l'avenir du trading et de la négociation réside dans l'automatisation. Selon les rapports de l'industrie, la taille du marché mondial du trading algorithmique devrait passer de 11,1 milliards de dollars en 2019 à 18,8 milliards de dollars d'ici 2024.

Bien que le trading algo surpasse les styles traditionnels de trading sur de nombreux points, l'intervention humaine est toujours nécessaire dans une certaine mesure pour une meilleure tenue de marché avec des pensées prudentes pour assurer la stabilité des marchés financiers.

De plus on pourrait même voir un algorithme qui adapte sa stratégie en fonction du marché avec 0 intervention humaine, ou pour aller plus loin qui crée sa propre stratégie.

REFERENCES

1. <https://github.com/fengyang95/Awesome-Deep-Learning-Based-Time-Series-Forecasting>
2. <https://www.reddit.com/r/algotrading/>
3. <https://github.com/QuantConnect/Lean>
4. <https://github.com/aditid/TradingBot>
5. https://en.wikipedia.org/wiki/Algorithmic_trading
6. <https://blog.floydhub.com/introduction-to-genetic-algorithms/>