Ateliers Machine Learning Brest « ALLOHa »

Introduction au Reinforcement Learning

Yoann SOLA
ENSTA Bretagne

26/02/2019

Plan

I – Introduction : les sous-domaines du machine learning

II – Théorie du Reinforcement Learning

III – Application : Jeux Atari 2600

IV – Ma thèse

I – Introduction : les sous-domaines du machine learning

- **Machine Learning = Apprentissage automatique**

  Une machine apprend à exécuter une tâche pour laquelle elle n'a pas été explicitement programmée.

- **Différents types d'apprentissage :**
  - Apprentissage supervisé
  - Apprentissage non-supervisé
  - Apprentissage par renforcement

I – Introduction : les sous-domaines du machine learning

- **Apprentissage supervisé**

  Un superviseur guide la machine en lui fournissant des exemples à imiter. Les données d'apprentissage sont ainsi **« étiquetées » par le superviseur** et montrent le résultat attendu.

  Exemple : Reconnaissance de chiffres manuscrits.

- **Apprentissage non-supervisé**

  La machine n'est pas guidée par un superviseur et doit **trouver elle-même des similarités** entre les données, afin de les rassembler sous une même étiquette.

  Exemple : Système de recommandation

II – Théorie du Reinforcement Learning

- **Origines de l'apprentissage par renforcement**

  - **Automatique :** contrôle optimale = trouver les commandes d'un système qui maximisent ou minimisent un critère donné.

  - **Psychologie :** apprentissage par « essai et erreur » = découvrir et favoriser les actions menant à une récompense et éviter celles menant à une punition.

II – Théorie du Reinforcement Learning

- **Formulation du problème**

Utilisation des **« processus de décision markovien (MDP) »** comme outil théorique.

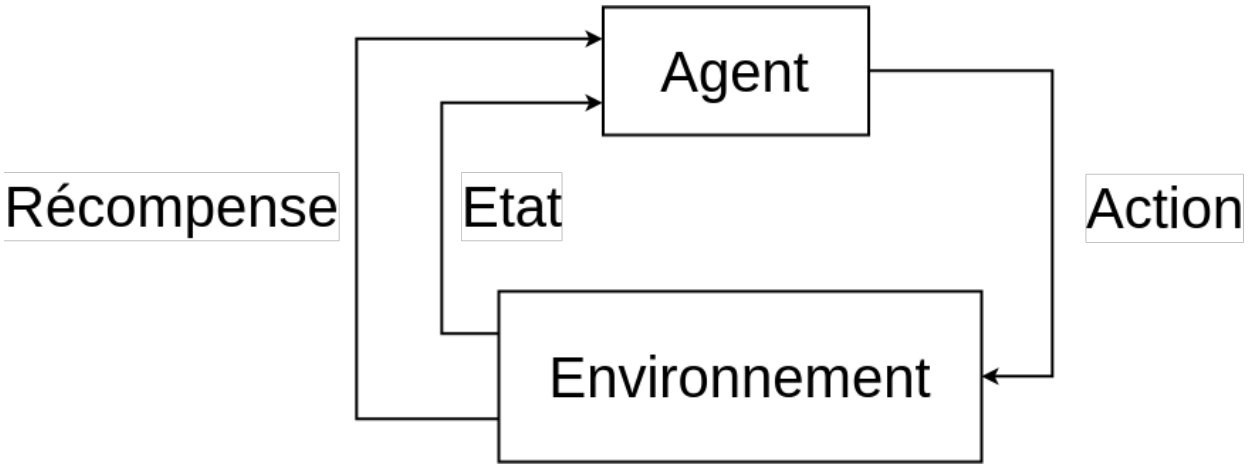
- Agent = Machine qui va apprendre une tâche donnée.

Exemple : Robot, IA d'un jeu vidéo, pendule inversé.

- Environnement = Tout ce que l'agent ne peut contrôler. Incertain.

Exemple : Rue, salle, batterie d'un robot.

II – Théorie du Reinforcement Learning



- **Etat** = perception de l'environnement par l'agent

- **Récompense** = évaluation des actions de l'agent, peut être positive ou négative (punition)

II – Théorie du Reinforcement Learning

- E0 → A1 → R1 → E1 → A2 → R2 → E2 …
- **Objectif de l'agent :** maximiser les récompenses attendues sur le long terme : R0 + R1 + … + Rk
- La récompense permettra à l'agent de connaître le **jugement** de ses actions (différent de la **correction** apportée en apprentissage supervisé).

II – Théorie du Reinforcement Learning

- **Exemple de problème de Reinforcement Learning :**

  Un robot ramasseur de déchets dans une cafétéria.

- Le robot peut décider de revenir vers une station afin de recharger ses batteries.

  - Agent = le robot
  - Environnement = la cafétéria
  - Action = avancer tout droit, pivoter sur place ou ramasser
  - État = décrit par les capteurs du robot : caméra RGB, radar, niveau de la batterie, etc.
  - Récompense = +1 pour chaque déchet ramassé, -3 si la batterie tombe à plat avant d'être revenu à la station.
  - Objectif : maximiser le nombre de déchets ramassés sans manquer d'énergie

II – Théorie du Reinforcement Learning

- **Politique et fonction de valeur**

  - **Politique d'un agent** = fonction définissant la **probabilité** d'une action d'être **choisie** par l'agent, dans un état donné : π(a|e)

    Elle dicte le **comportement** de l'agent.

    Exemple : robot recycleur

    Etat = {distance du déchet, niveau de la batterie}

    Action = {avancer vers un déchet OU revenir vers la station}

    - π(déchet | 2m , 68 %) = 0,95 et π(station | 2m , 68 %) = 0,05
    - π(déchet | 1m , 5 %) = 0,1 et π(station | 1m , 5 %) = 0,9

II – Théorie du Reinforcement Learning

- **Fonction de valeur** = fonction associant une **valeur Q** (Q comme Qualité) pour chaque combinaison d'état et d'action possible : Q(e,a)

- Plus Q(e,a) est élevé, plus l'**action a** va engendrer une quantité attendue de **récompense importante sur le long terme**, en partant de l'**état e**.

II – Théorie du Reinforcement Learning

- A un *instant t* et pour une *politique π* donnée, la *valeur Q* de l'*action a* à l'*état e* est l'**espérance de la somme des futures récompenses possibles** :

$$Q_\pi(e,a) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | E_t = e, A_t = a]$$

  0 ≤ ɣ ≤ 1 = discount rate = plus il est proche de 1, plus les futures récompenses attendues ont de l'importance dans la prise de décision.

II – Théorie du Reinforcement Learning

- Dans un état donné, la **meilleure action** à entreprendre est donc celle possédant **la plus grande valeur Q** : c'est l'action qui engendrera le plus de récompenses au cours du temps.

- **Politique optimale** : politique qui privilégie pour tous les états possibles l'action possédant la plus grande valeur Q.

- Le **problème est résolu** lorsque la politique optimale de l'agent est trouvée.

II – Théorie du Reinforcement Learning

- Pour trouver la politique optimale de l'agent, il faut **estimer les valeurs Q** de manière fiable.

- Au début de l'apprentissage, l'agent va **essayer aléatoirement** des actions afin d'explorer son environnement. Puis il va **construire sa politique** en **mettant à jour les valeurs** des actions grâce aux récompenses reçues.

II – Théorie du Reinforcement Learning

- **Plusieurs approches possibles estimer les valeurs Q :**

  - Si les espaces d'état et d'action ont **peu d'éléments** :

    Les combinaisons d'état et d'actions peuvent être rangés dans des tables et leurs valeurs respectives peuvent souvent être **déterminées de manière exacte**.

    Exemple d'algorithme : TD learning, Q-learning

  - Si les espaces d'état et d'action sont **trop grands** :

    Les valeurs doivent être **approchées**, car les calculs exactes prendraient trop de temps.

    Exemple d'algorithme : Deep Q-learning, Policy gradient

II – Théorie du Reinforcement Learning

- **Compromis à trouver entre exploration et exploitation :**

  ε = paramètre à régler

  - L'agent va suivre sa politique actuelle avec une probabilité 1-ε : il **exploite l'information** qu'il possède déjà (les valeurs Q)

  - L'agent va choisir une action aléatoirement avec une probabilité ε : il **explore son environnement** afin d'améliorer son estimation des valeurs des actions.

    Les valeurs des couples état-action peuvent varier dans le temps et il faut donc les mettre à jour en sélectionnant des actions qui ont été écartées par la politique actuelle.

II – Théorie du Reinforcement Learning

- Ouvrage théorique de référence dans le Reinforcement Learning (**la « bible » du RL**) :

  Richard S. Sutton and Andrew G. Barto, *« Reinforcement Learning – An Introduction – Second Edition »*, The MIT Press, 2018.

III – Application : Jeux Atari 2600

- Application présentée par **DeepMind** en 2013.

  Plusieurs jeux d'Atari 2600 utilisés pour tester leur algorithme de **Deep Q-network (DQN)** : Pong, Space Invaders, Breakout…

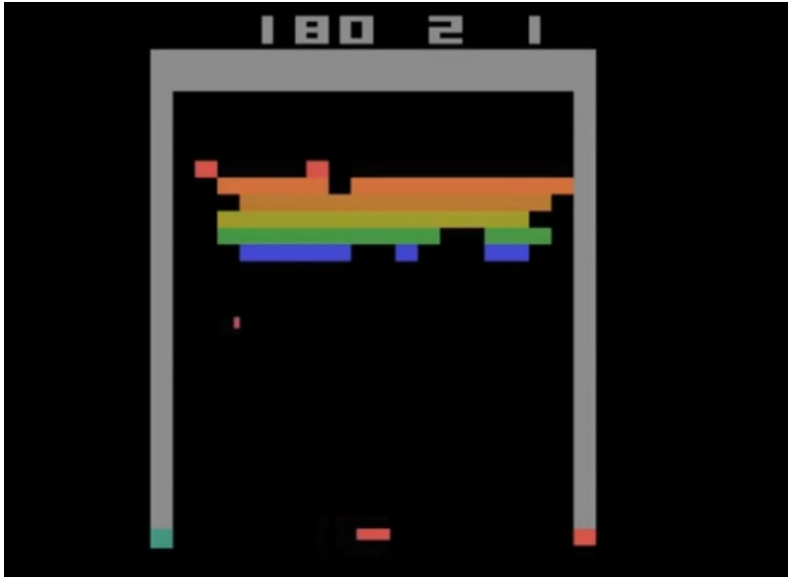
- Environnement très complexe : incertain et de **grandes dimensions**.

- L'apprentissage par renforcement se base **uniquement sur l'affichage** du jeu.

- **Aucune connaissance préalable** n'est utilisée : l'agent ne connaît pas les règles du jeu.

III – Application : Jeux Atari 2600

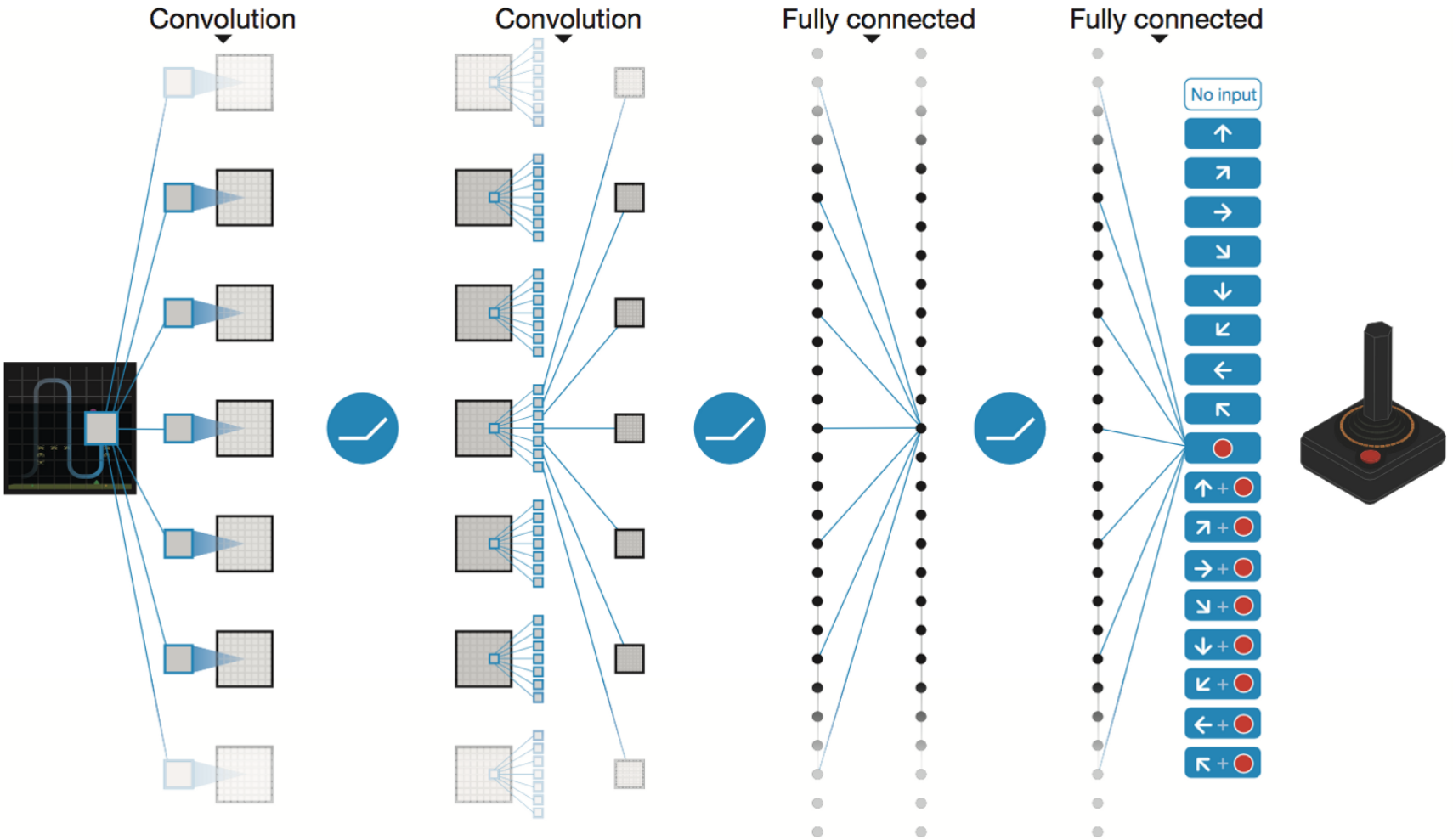
- Exemple de Breakout



Jeu vidéo de type casse-brique

- Agent = IA contrôlant le chariot rouge
- Environnement = jeu Breakout
- Etat = les 4 dernières images du jeu
- Action = gauche ou droite
- Récompense = +1 pour chaque brique cassée
- Objectif : maximiser le score du jeu

III – Application : Jeux Atari 2600



III – Application : Jeux Atari 2600

- **Vidéo de démonstration**

Google DeepMind's Deep Q-learning playing Atari Breakout

https://www.youtube.com/watch?v=V1eYniJ0Rnk

III – Application : Jeux Atari 2600

- Utilisation du framework **Gym** d'**OpenAI** :

  https://github.com/openai/gym

- Plusieurs algorithmes de Reinforcement Learning (dont DQN de DeepMind) implementés dans ensemble **Baselines** d'**OpenAI** :

  https://github.com/openai/baselines

III – Application : Jeux Atari 2600

- Ce sont deux **outils de référence** de l'aprentissage par renforcement.
  - Gym = **benchmark**
  - Baselines = implémentation d'**algorithmes célèbres**
- Nécessite d'avoir installé **Tensorflow**.

III – Application : Jeux Atari 2600

- **Publication scientifique originale** de DeepMind sur sa maîtrice des jeux Atari 2600 :

  Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M. (2013), *« Playing atari with deep reinforcement learning »,* arXiv preprint arXiv:1312.5602.

  https://arxiv.org/pdf/1312.5602v1.pdf

III – Application : Jeux Atari 2600

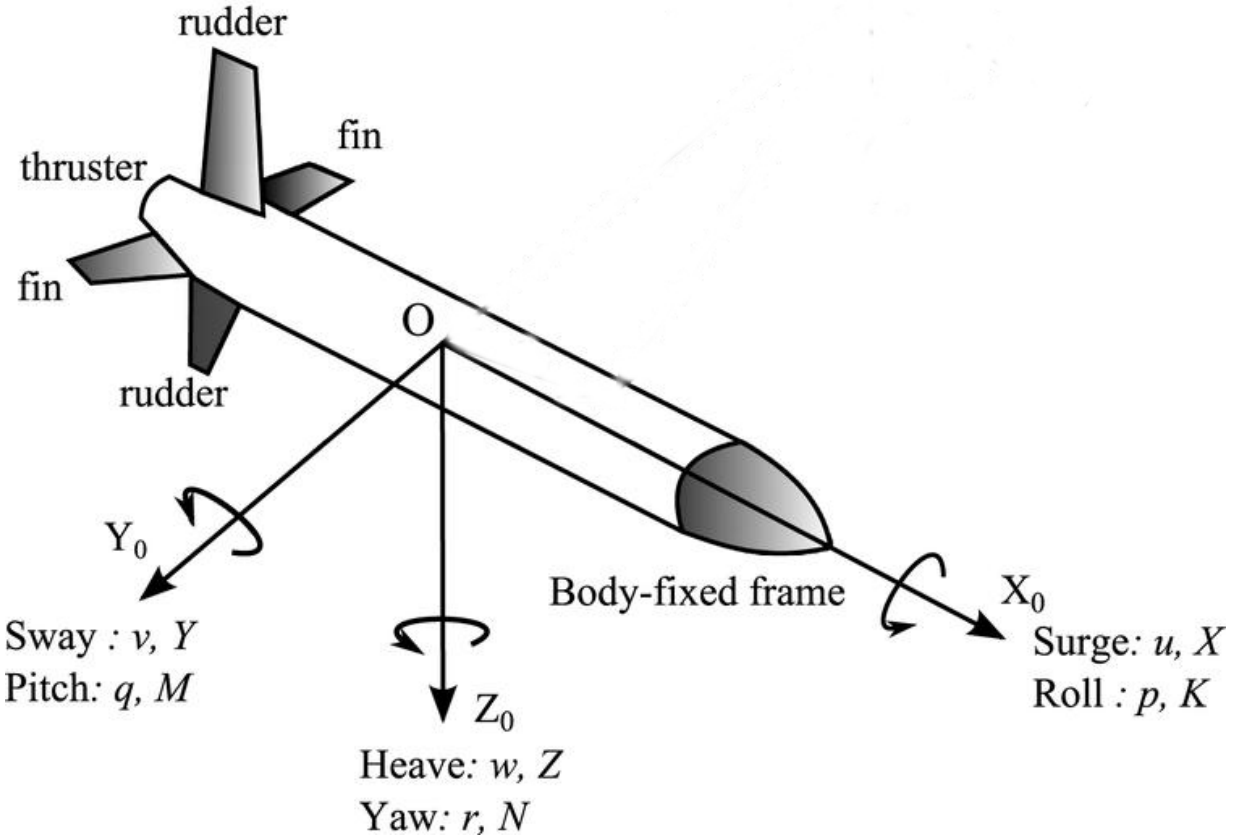
- **Autres applications** du Reinforcement Learning **très médiatisées** :
  - Les **échecs** et le jeu de **Go** (DeepMind)
  - Les jeux vidéo **Starcraft 2** (DeepMind) et **Dota 2** (OpenAI)
  - Le contrôle de **robots** : humanoïde, cheetah, bras robotique.

IV – Ma thèse

- **Sujet :** Machine learning pour le contrôle robuste de robots sous-marins.
- Effectuée à l'ENSTA Bretagne, au sein du Lab-STICC (UMR)
- Encadrée par Benoît CLEMENT, Gilles LE CHENADEC et Jordan NININ.
- Financée par la DGA et la région Bretagne.
- **Avant la thèse :** diplômé de l'ENSTA Bretagne, filière Robotique

IV – Ma thèse

- **AUV :** Autonomous Underwater Vehicle



IV – Ma thèse

- **Contraintes du milieu sous-marin :**

  - **Incertain :** des perturbations peuvent survenir (courant, rochers, autres véhicules)

  - **Non structuré :** pas de modèle fiable décrivant tous les phénomènes

- Les AUV doivent être piloter de manière **efficace ET sûre durant** leurs missions.

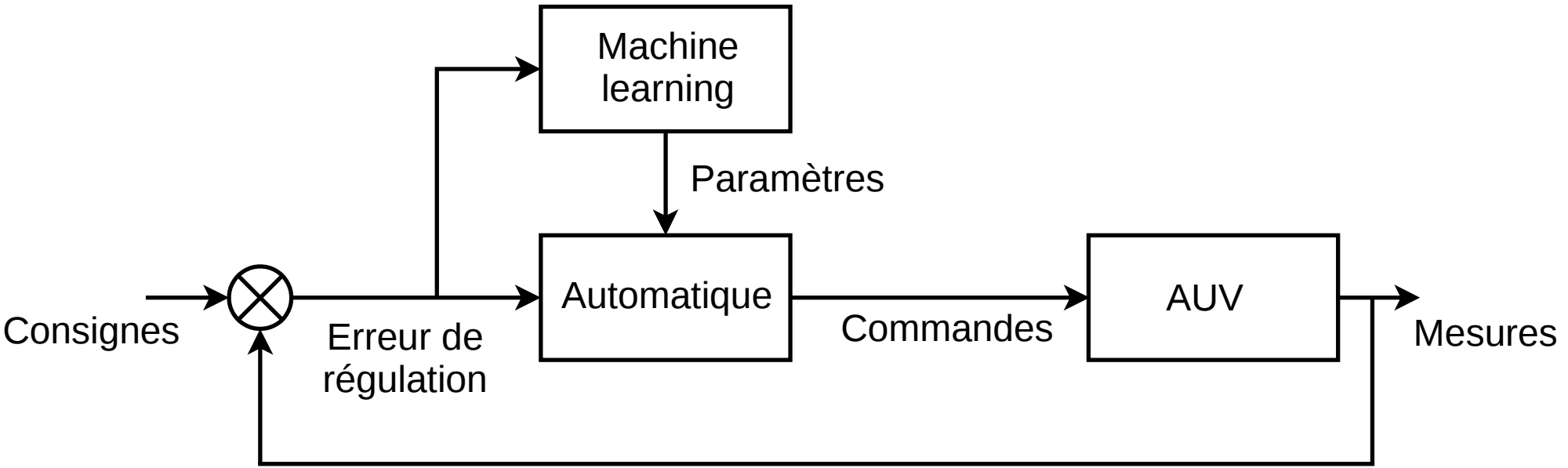
IV – Ma thèse

- Utiliser le **machine learning et l'automatique en parallèle** afin d'avoir le meilleur des deux mondes :

  - **Machine learning :** permet à l'AUV de **s'adapter à son environnement** incertain.

  - **Automatique :** permet de **garantir la stabilité** de la loi de commande de manière formelle.

IV – Ma thèse


Machine learning
Paramètres
Automatique
AUV
Consignes
Erreur de régulation
Commandes
Mesures

IV – Ma thèse

- **Algorithme utilisé en automatique :** régulateur PID (Proportionnel Intégral Dérivé).

- **Algorithmes testés en machine learning :**

  - PIDNN
  - RNN
  - RL

IV – Ma thèse

- Pour l'instant, algorithmes testés en **simulation**, mais **prévision de les embarquer** d'ici la fin de la thèse.

- **Outils utilisés :**
  - ROS : http://www.ros.org/
  - Gazebo : http://gazebosim.org/
  - UUV Simulator : https://uuvsimulator.github.io/

Github Ateliers Machine Learning Brest (ALLOHa)

https://github.com/amlb/amlb.github.io