Chapitre 7: Optimisation par apprentissage

INF889B — Algorithmes d'optimisation combinatoire

Alexandre Blondin Massé

Université du Québec à Montréal

Hiver 2020

Partie a: Introduction

Contenu du chapitre

a. Introduction
b. Apprentissage par renforcement
c. Chaînes de Markov
d. Processus de décision markoviens
e. Réseaux de neurones
f. Recherche arborescente de Monte-Carlo
g. Méthode des différences temporelles
h. Alpha Zero

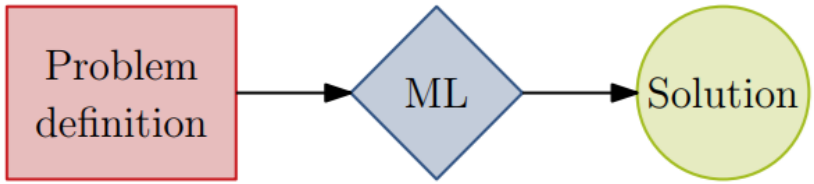
Plan

Optimisation par apprentissage

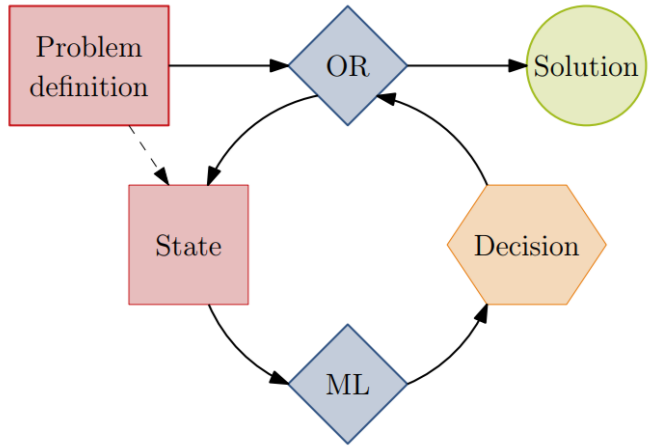
Optimisation par apprentissage

- *Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon*
- Version originale: 15 novembre **2018**
- Version révisée: 12 mars **2020**
- Par Bengio, Lodi et Prouvost
- Disponible sur ArXiV
- Survol de différentes tentatives d'utiliser l'**apprentissage automatique** pour l'optimisation combinatoire
- Par les **deux communautés** (AA et OC)

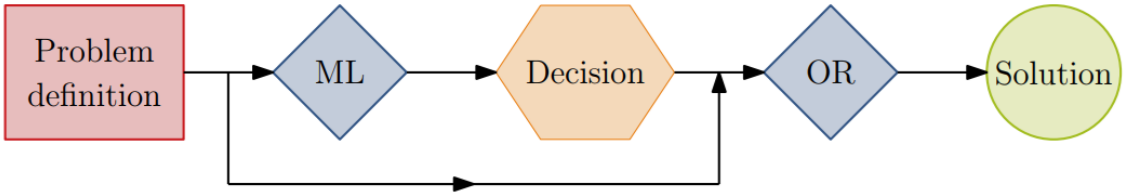
Trois approches générales



Bout-en-bout



Collaboration **répétée**



Détermination des **paramètres**

Images tirées de Bengio, Lodi et Prouvost

Algorithme tabou réactif

Rappels

Différentes stratégies

- Recherche locale
- Gloutonne
- Recuit simulé
- Recherche taboue
- Évolutionnaire

Mémoire

- Est-ce qu'on se rappelle des états **précédents**?
- Ou on ne se fie qu'à l'état **actuel**?
- **Sans** mémoire: recherche locale, recuit simulé, gloutonne, évolutionnaire
- **Avec** mémoire: recherche taboue, gloutonne

Paramétrisation

- Essentiellement toutes les stratégies demandent de **choisir** les valeurs de certains paramètres
- **Comment** faire ce choix?
- Quels sont les **meilleures** valeurs pour les paramètres?
- Essais et erreurs
- Difficile de **reproduire** les résultats

Utilisateur expert

- La qualité de la solution **dépend** de l'utilisateur
- En effet, celui-ci est l'**expert** qui sait ou devine quelles sont les valeurs les plus intéressantes
- Il cherche un équilibre entre la **diversification** et l'**intensification**

Réaction

Définition

Toute modification d'une valeur d'un paramètre de l'algorithme en réaction au comportement de l'algorithme durant son exécution est appelée **réaction**. (Par opposition à une modification entre deux exécutions consécutives de l'algorithme.)
Une heuristique est dite **réactive** si elle a des réactions pendant son exécution.

Remarques

- Une heuristique réactive est évidemment **plus complexe** à implémenter que sa version de base
- Le développeur a pour objectif de **formaliser** son expertise et de l'insérer directement dans l'implémentation

Deux conditions importantes

Afin de rendre une heuristique **réactive**, on doit s'assurer que les deux conditions suivantes sont respectées:

- **Documentation complète et non ambiguë**: il est essentiel de décrire de façon non ambiguë la façon dont les paramètres sont mis à jour, autant dans le manuscrit scientifique que dans le code

$\rightarrow$ garantit une meilleur reproductibilité

$\rightarrow$ paraît plus fiable et convaincant

- **Automatisation**: la phase d'amélioration de la paramétrisation est longue et fastidieuse et nécessite une procédure automatique

$\rightarrow$ car l'ajout de **réactions** dans un algorithme demande beaucoup plus de travail

$\rightarrow$ il faut donc que ça en vaille la peine

Rappel

1: **procedure** RechercheTaboue
2: Soit S une solution initiale
3: $B \leftarrow \emptyset$
4: **tant que** il reste des itérations **ou** pas de stagnation **faire**
5: $S' \leftarrow \emptyset$
6: **pour chaque** solution admissible S'' voisine de S **faire**
7: Si S'' est strictement meilleur que S' alors $S' \leftarrow S''$
8: Si S'' est strictement meilleur que B alors $B \leftarrow S''$
9: $S \leftarrow S'$
10: Mettre à jour les restrictions taboues

Temps d'interdiction

- Une restriction taboue classique est d'interdire une configuration pour une **durée** T fixe (en nombre d'itérations)
- **Expérimentation**: entre 5 et 12 semble optimal (7 étant le nombre magique)
- Comment **choisir**?

Exemple de stratégie

- Initialement, on prend $T = 1$
- On **augmente** T s'il y a une suggestion qu'une **diversification** est requise, par exemple, parce qu'on rencontre souvent les mêmes solutions
- On **diminue** T si la situation n'évolue pas et qu'on tombe sur des configurations diversifiées

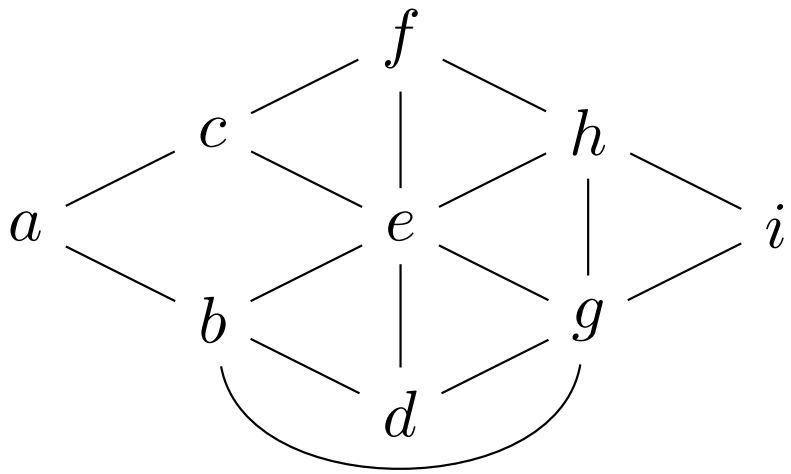
Échappement (*escape*)

- La recherche taboue permette d'éviter la **convergence** vers un optimal local
- Mais elle n'empêche pas de **boucler** sur les mêmes optimums locaux
- Différentes **stratégies**: mutations successives aléatoires, saut dans l'espace, etc.
- **Théoriquement**, cette stratégie garantit qu'on atteindra l'optimum (un jour)

Considérations pratiques

- Maintenir un **historique** de l'espace exploré entraîne généralement un coût important
- Besoin de **structures de données** en conséquence: tables de hachage, arbre radix, arbres binaires de recherche (équilibrés ou non), etc.
- Utilisation de structures de données **persistantes**

Exemple: le problème de la clique maximale



$\{b, d, e, g\}$ forme une **clique maximale**

- On prend un graphe simple G
- On cherche un sous-graph G de G **complet** de taille maximale
- C'est un problème **NP-difficile**

Algorithme réactif

- Proposé par **Battiti et Protasi**, dans *Reactive local search for the maximum clique problem*, Algorithmica, 29(4), 610, 2001.
- On **ajoute** ou **retire** des sommets d'une clique courante X selon une stratégie « réactive »

Variable	Description
t	itération actuelle (temps discret)
T	durée de l'interdiction
t_T	dernière itération où T a été modifiée
t_R	dernière itération où on a redémarré
X	ensemble de sommets courant
X_b	meilleur ensemble de sommets trouvé
k_b	cardinalité du meilleur ensemble trouvé
t_b	itération où X_b a été trouvé
$\ell[u]$	dernière itération concernant le sommet u

Pseudocode

1: **fonction** CLIQUE(G : graphe)
2: $t \leftarrow 0$, $T \leftarrow 1$, $t_T \leftarrow 0$, $t_R \leftarrow 0$
3: $X \leftarrow \emptyset$, $X_b \leftarrow \emptyset$, $k_b \leftarrow 0$, $t_b \leftarrow 0$
4: Pour tout $u \in V(G)$, soit $\ell[u] \leftarrow -\infty$
5: **faire**
6: $T \leftarrow$ RÉACTION(G, X, T)
7: $X \leftarrow$ VOISIN(G, X)
8: $t \leftarrow t + 1$
9: **si** $|X| > k_b$ **alors** $X_b \leftarrow X$, $k_b \leftarrow |X|$, $t_b \leftarrow t$
10: **si** $t - \max\{t_b, t_R\} > 100k_b$ **alors**
11: $t_R \leftarrow t$
12: REDÉMARRER
13: **tant que** k_b n'est pas acceptable
14: **retourner** X_b

Réaction

1: **fonction** RÉACTION(G : graphe, X : ensemble de sommets, T : temps)
2: Rechercher X dans l'historique
3: **si** X a été trouvé **alors**
4: $R \leftarrow t - \mathit{visité}[X]$
5: $\mathit{visité}[X] \leftarrow t$
6: **si** $R < 2(n-1)$ **alors**
7: $t_T \leftarrow t$
8: **retourner** T après l'avoir augmenté
9: **sinon**
10: Insérer X dans l'historique
11: $\mathit{visité}[X] \leftarrow t$
12: **si** $t - t_T > 10k_b$ **alors**
13: $t_T \leftarrow t$
14: **retourner** T après l'avoir diminué

Voisinage

- On dit d'un sommet $u \in V(G)$ qu'il est **interdit** si $\ell[u] \geq t - T$
- Sinon, on dit que u est **permis**

1: **fonction** VOISIN(G : graphe, X : ensemble de sommets)
2: **si** il existe un sommet permis u dans $V(G) - X$ adjacent à tout X **alors**
3: Mettre à jour $\ell[u]$
4: **retourner** $X \cup \{u\}$
5: **sinon si** il existe un sommet permis u dans X **alors**
6: Mettre à jour $\ell[u]$
7: **retourner** $X - \{u\}$
8: **sinon si** $|X|$ est non vide **alors**
9: Choisir $u \in X$ au hasard
10: Mettre à jour $\ell[u]$
11: **retourner** $X - \{u\}$
12: **sinon** ▷ X est vide
13: Choisir $u \in V(G)$ au hasard
14: Mettre à jour $\ell[u]$
15: **retourner** $\{u\}$

Redémarrage

1: **procedure** REDÉMARRER
2: $T \leftarrow 1, t_T \leftarrow t$
3: **si** il existe $v \in V(G)$ tel que $\ell[u] = -\infty$ **alors**
4: $X \leftarrow \{v\}$
5: **sinon**
6: Choisir v aléatoirement
7: $X \leftarrow \{v\}$

Caractéristiques de l'algorithme

- **Efficace**: complexité linéaire par itération
- **Convivial**: pas besoin de choisir la valeur du paramètre T
- **Simplicité**: relativement facile à implémenter
- **Performant**: empiriquement, les résultats obtenus sont parmi les meilleurs pour des métaheuristiques, algorithmes approximatifs

Apprentissage automatique

Trois approches possibles

Non supervisé

- Données **non étiquetées**
- Aucune **rétroaction**
- On recherche des informations **cachées** dans les données

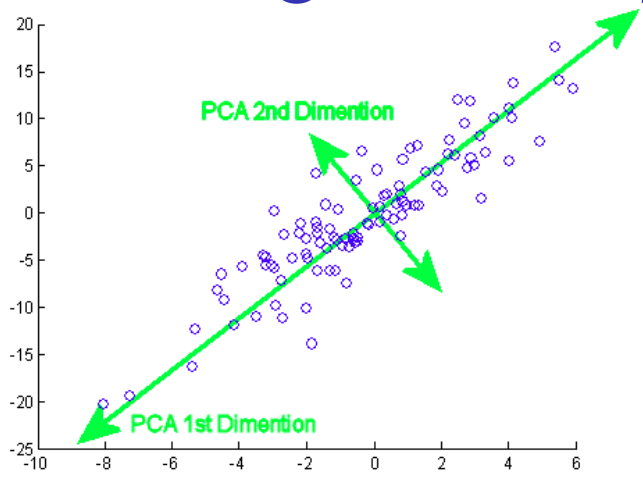
Supervisé

- Données **étiquetées**
- **Rétroaction directe**
- On souhaite **prédire** à partir de nouvelles entrées

Par renforcement

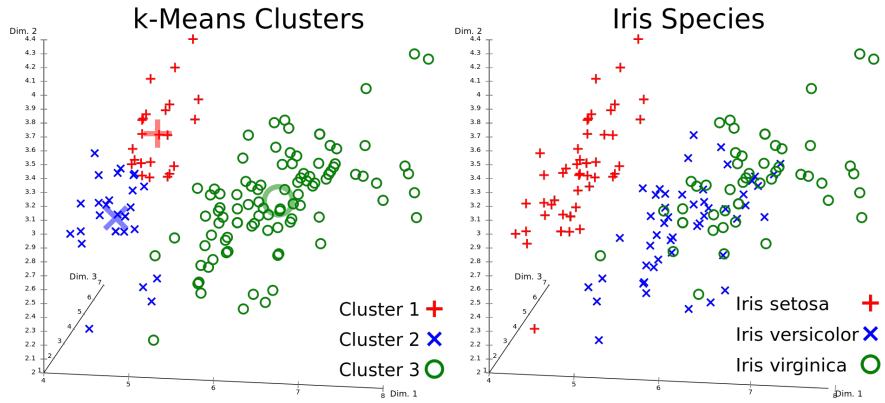
- **Agent** évoluant dans un **environnement**
- Mécanisme de **récompenses**
- On souhaite apprendre une série **d'actions**

Apprentissage non supervisé



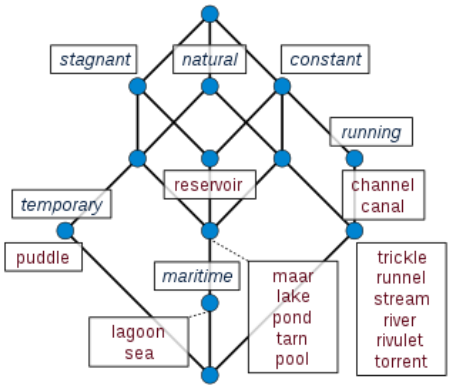
(Source: Machine Learning Course)

Analyse en composantes principales



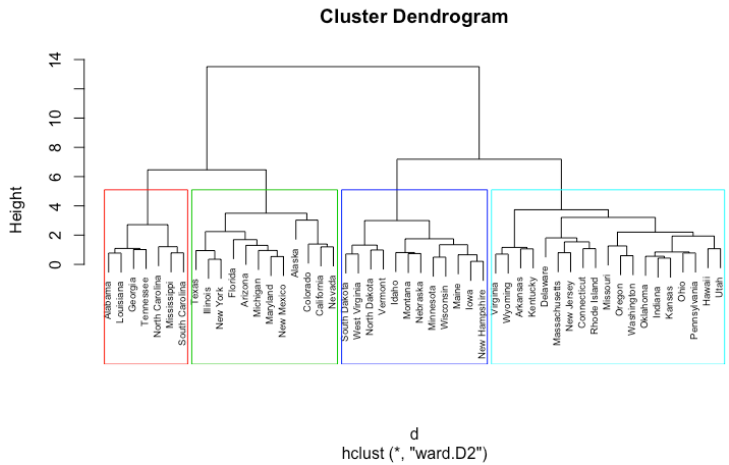
(Source: Wikipedia)

Partitionnement en k-moyennes



(Source: Wikipedia)

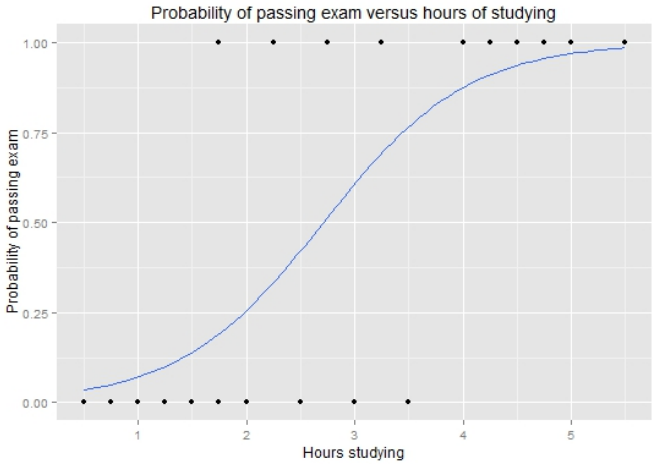
Analyse de concepts formels



(Source: UC Business Analytics)

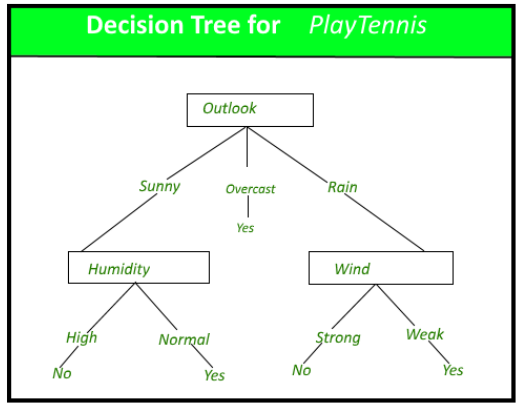
Partitionnement hiérarchique

Apprentissage supervisé



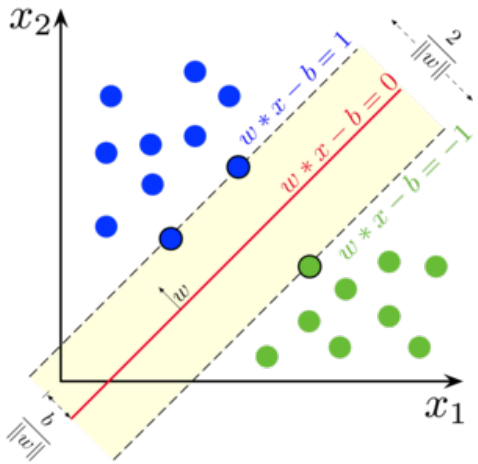
(Source: Wikipedia)

Régression



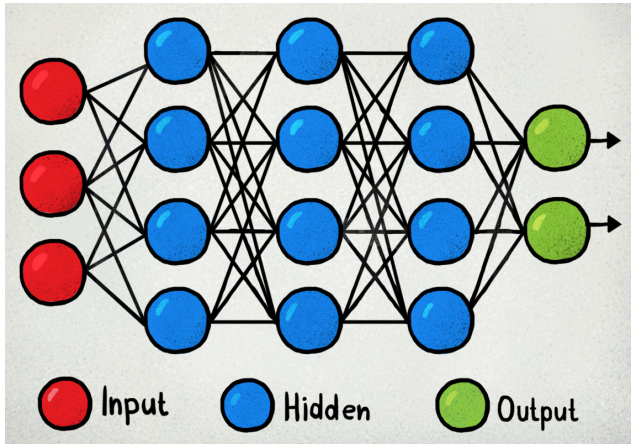
(Source: GeeksforGeeks)

Arbre de décision



(Source: Wikipedia)

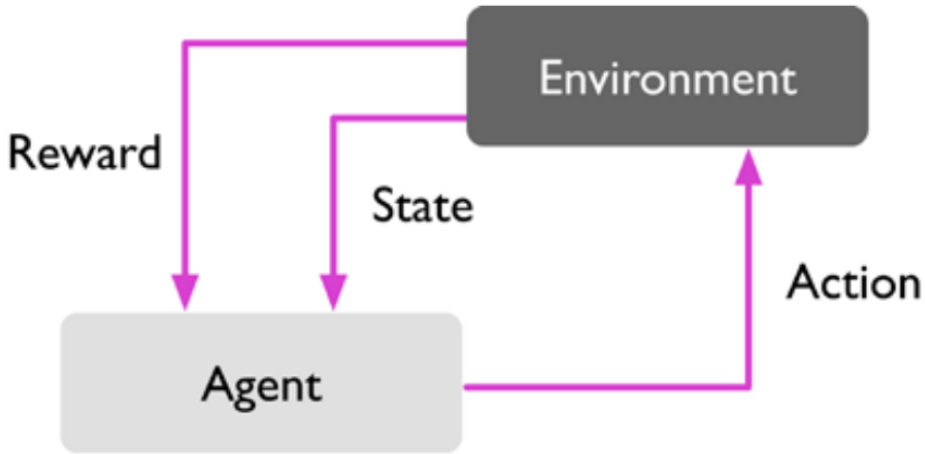
Machine à vecteurs de support



(Source: Medium.com)

Réseau de neurones artificiels

Apprentissage par renforcement



(Source: Python Machine Learning)

- L'agent effectue des **actions**
- L'environnement communique l'**état** résultant et...
- ...retourne une **récompense** (immédiate) à l'agent
- On souhaite **optimiser** le total des récompenses

AlphaGo

(Source: Youtube)

AlphaGo contre Lee Sedol

Jeux de plateaux

Ordinateur contre personne

- **Puissance 4**: 1989 (VICTOR)
- **Échecs**: 1996-1997 (Deepblue)
- **Atari/go/shogi/échecs**: 2017-2018 (AlphaGo, AlphaZero)

Prochaine étape?

> *« [La victoire d'AlphaGo] marque la fin d'une époque... les jeux de plateau sont plus ou moins réglés, et il est temps de passer à autre chose. »*
>
> — Murray Campbell (équipe de Deep Blue)

Références principales

- *Introduction to reinforcement learning*, R.S. Sutton, A.G. Barto et al., MIT Press Cambridge, 2018, disponible en ligne
- *Python Machine Learning, 3rd edition*, S. Raschka, Packt Publishing, 2019, site officiel
- *Deep Learning*, par I. Goodfellow, Y. Bengio and A. Courville, MIT Press, 2016, disponible en ligne
- *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*, par D. Silver et al., ArXiV, 2017, disponible en ligne
- *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*, par J. Schrittwieser et al., ArXiV, 2019, disponible en ligne