Méthodes de classification de données par apprentissage

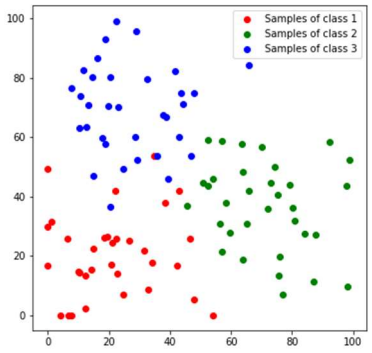
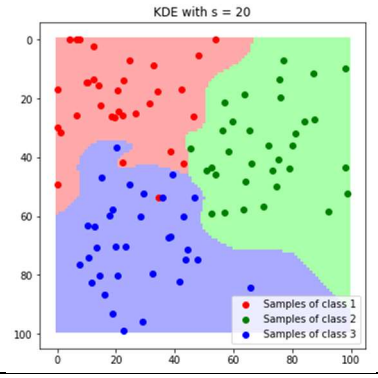
KNN

Méthode de classification à apprentissage supervisé

Vise à trouver à quelle famille appartient un nouvel élément en trouvant dans quelle famille majoritaire appartiennent ses k plus proches voisins

Soient $U_i^j = (u_0^j, u_1^j, \dots, u_{n-1}^j)$ des n-uplets représentant les données apprises U_i séparées en familles F_j.
La reconnaissance à l'aide de knn consiste à calculer la distance d d'un nouvel échantillon $E = (e_0, e_1, \dots, e_{n-1})$ à chacun des U_i : $d = \sqrt{\sum_{i=0}^{n-1}(e_i - u_i)^2}$

On établit alors une liste L_k des k plus proches voisins U_i^j de E_i et on trouve la famille j d'appartenance probable de E en trouvant la première famille majoritaire présente dans L_k

Situation initiale Points et couleurs par famille	
Estimation de l'appartenance à chaque famille de tous les points du plan avec k=3 plus proches voisins	

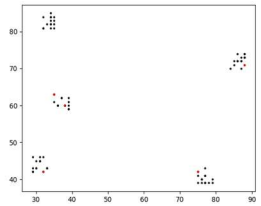
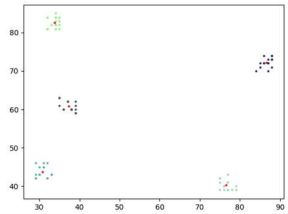
K-means

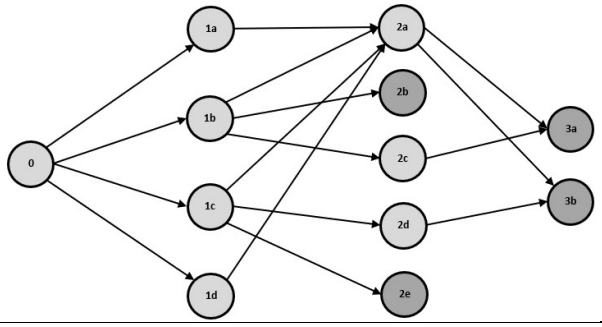
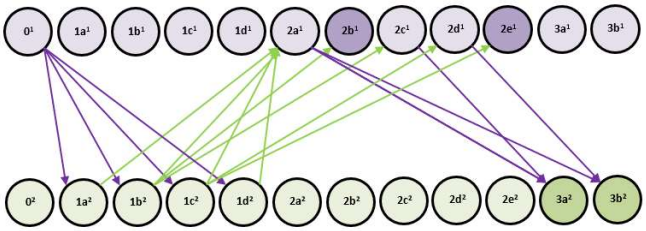
Algorithme à apprentissage non supervisé

Vise à regrouper des données en k familles les plus homogènes possibles.
Il fait appel à une méthode de partitionnement de données permettant de les regrouper en k groupes/clusters autour de k centres tels que chaque cluster soit l'ensemble des données de plus faible distance euclidienne au centre du cluster.

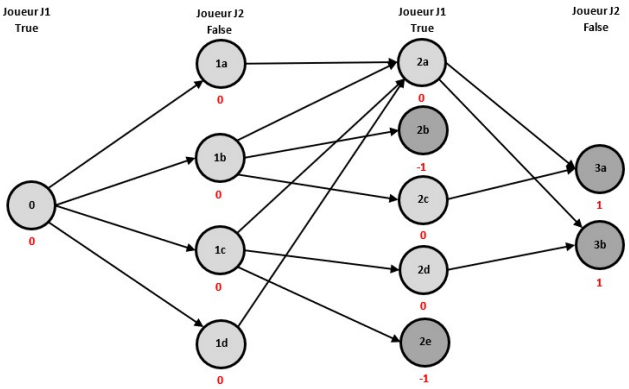
A partir d'un ensemble de n points $(x_1, x_2, .., x_n)$, l'algorithme vise à les partitionner en k clusters $(S_1, D, \dots, S_k)$ de centres respectifs $(c_1, c_2, \dots, c_k)$ avec $k \leq n$ de la manière suivante :

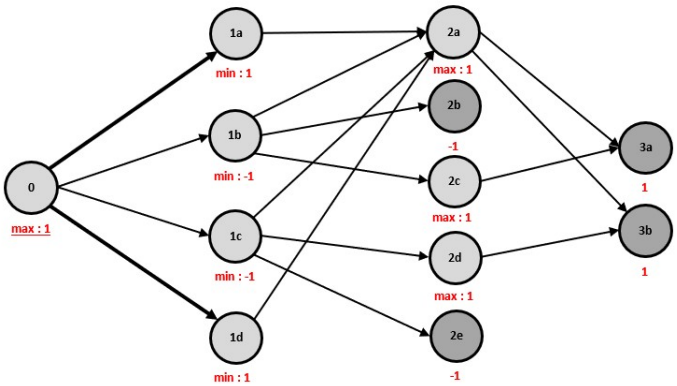
- Etape 0 : Choix aléatoire de k centres c_j distincts parmi les n points x_i
  - Etape 1 : En notant x_i^j les n_j points du cluster S_j, cette étape vise à affecter les n points x_i dans k clustersc S_j tels que $\|x_i^j - c_j\| = \min_l \|x_i^j - c_l\|$.
  - Etape 2 : Calcul des nouveaux centres c_j des k clusters S_j : $c_j = \frac{1}{n_j}\sum_{i=1}^{n_j} x_i^j$
- Itérations : Répétition des étapes 1 et 2 jusqu'à ce que les $(c_1, c_2, \dots, c_k)$ n'évoluent plus

Situation initiale Points et centres initiaux	Situation finale Clusters par couleurs et centres rouges
	

Les jeux d'accessibilité à 2 joueurs	
Position Situation Configuration	Une position, situation ou configuration est une étape de déroulement d'un jeu, c'est-à-dire l'état des éléments constitutifs de ce jeu.
Coups	Les coups sont les changements d'états successifs au cours du jeu
Graphe d'un jeu	On représente un jeu par un graphe G de ses positions G=(S,A) avec S les sommets et A les arêtes (coups) 
Partie	Une partie est un ensemble de coups menant au gain d'un joueur/à la défaite de l'autre, ou à un match nul
Jeu acyclique	Jeu qui ne présente pas de cycles. On ne repasse pas plusieurs fois à la même position
Jeu à information totale	A tout instant, chacun des joueurs à une information complète de l'état du jeu (rien n'est caché)
Jeu sans mémoire	Un coup ne dépend pas de ce qu'il s'est passé avant, dans la partie en cours ou à la partie précédente
Jeu à somme nulle	La somme des gains et des pertes est égale à 0
Jeu déterministe, sans hasard	Une décision amène toujours à la même situation
Arène	En appelant S1 et S2 les sommets atteignables par les joueurs J1 et J2 au cours de la partie sur le graphe G, on appelle arène le triplet (G,S1,S2)
Sommets gagnants Etats finals Etats terminaux	Les sommets gagnants, états finals ou terminaux d'un joueur, sont les états qui mènent ce joueur à gagner la partie. On les désigne par F1 et F2 avec F1⋂F2=Ø pour les joueurs 1 et 2
Jeu d'accessibilité à deux joueurs	Ensemble de l'arène et de F1 et F2 ((G,S1,S2),F1,F2).
Graphe biparti	Lorsque les deux joueurs jouent à tour de rôle, on définit un graphe biparti en dupliquant tous les sommets du jeu et en représentant les parties entre ces deux sous-ensembles  Le graphe bipartie divise ainsi l'ensemble des sommets S en deux sous-ensembles disjoints S1 et S2 des sommets atteignables par les joueurs 1 et 2 tel que les arêtes relient uniquement des sommets entre S1 et S2

Stratégie sans mémoire	Une stratégie sans mémoire est une méthode à appliquer à chaque coup
Stratégie gagnante	Une stratégie est dite gagnante pour un joueur si, en jouant cette stratégie, toute partie est finie et se termine dans un état gagnant pour ce joueur. On dit alors que la partie est jouée suivant cette stratégie.
Position gagnante (globalement)	Une position x est dite gagnante s'il existe une stratégie gagnante depuis ce sommet. Autrement dit, si le joueur qui y joue a beaucoup de mémoire, quels que soient les coups de son adversaire, il pourra forcer ce dernier à perdre. Cela ne veut pas dire que l'autre joueur est perdant/ne peut pas gagner, cela veut dire que si le joueur disposant d'une position gagnante ne commet aucune erreur, il gagnera. Disposer d'une position gagnante au départ garanti le gain du joueur concerné s'il connait l'intégralité du graphe G. Dans un graphe acyclique sans partie nulle, l'un des deux joueurs en dispose.
Position gagnante pour un joueur	Une position qui n'est pas gagnante pour le joueur qui y joue signifie que ses successeurs sont tous gagnants pour l'autre joueur. Les positions qui ne sont pas gagnantes sont donc des positions gagnantes pour l'adversaire du joueur qui y joue. Ainsi, si aucun match nul n'est possible, toute position est gagnante pour l'un des deux joueurs au cours de la partie. Les positions gagnantes (globales) sont les positions gagnantes pour J1 dans S1 et les positions gagnantes pour J2 dans S2 Les positions gagnantes pour J1 sont les positions gagnantes pour J1 dans S1 et les positions de S2 non gagnantes (globalement) (inversement pour J2).
Attracteurs Région gagnante	On définit ainsi les attracteurs d'un joueur (ou région gagnante), c'est-à-dire l'ensemble des positions (des deux joueurs) qui lui garantissent de pouvoir gagner (ce sont donc les positions gagnantes pour ce joueur). La procédure pour les déterminer est la suivante : Appelons F un ensemble de sommets. On appelle $Attr_1(F)$ la liste des attracteurs de F pour le joueur J1, c'est-à-dire la liste des sommets de S assurant le joueur J1 d'atteindre F. On appelle $Attr_1^k(F)$ l'ensemble des sommets à partir desquels le joueur J1 peut arriver dans F en au plus i coups. Le calcul des attracteurs se fait par itérations sur k (exemple pour le joueur J1) : - $Attr_1^0(F) = F$ - Tant que $Attr_1^{k+1}(F) \neq Attr_1^k(F)$ - $Attr_1^{k+1}(F) = \begin{cases} Attr_1^k(F) \\ U\{x \in S_1 \mid \exists x' \in Attr_1^k(F), (x,x') \in E\}\ (1) \\ U\{x \in S_2 \mid \forall x'/(x,x') \in E : x' \in Attr_1^k(F)\}\ (2) \end{cases}$ Autrement dit : ○ Les sommets actuellement trouvés menant à F ○ (1) Les sommets du graphe du joueur J1 ayant au moins une arête conduisant aux sommets menant à F (le joueur J1 choisira ce coup) ○ (2) Les sommets du graphe du joueur J2 dont les arêtes conduisent toutes à des sommets menant à F (le joueur J2 n'aura pas le choix) Avec : - $Attr_1^0(F) \in Attr_1^1(F) \in \cdots \in Attr_1(F)$ - $Attr_1^i(F)$ est une suite convergente avec au plus n itérations avec $n = \lvert S \rvert$ le nombre de sommets

Stratégie optimale	Un joueur joue une stratégie optimale si, à chaque coup, il choisit, lorsqu'elle existe, une position dans ses attracteurs (non gagnante pour l'adversaire)
Heuristique Fonction d'utilité	Une heuristique (ou fonction d'utilité) est une fonction qui dépend du joueur et de la position de jeu, et qui estime les chances de gain pour un joueur. Lorsque qu'il est impossible de départager les chances de gain hors des positions gagnantes pour un joueur ou perdantes (gagnantes pour l'autre), on choisit l'heuristique suivante (J joueur, A adversaire) : $h(x,J) = \begin{cases} +\infty \; si \; J \; gagne \; en \; x \\ -\infty \; si \; A \; gagne \; en \; x \\ 0 \; sinon \end{cases}$ Exemple (avec -1 et 1) où les positions grisées sont perdues pour le joueur qui y joue : 

Algorithme min-max	L'algorithme min-max permet de prédire la stratégie à jouer en ne créant pas tout le graphe du jeu (forte consommation de mémoire, sans parler des temps de calculs). Il étudie ainsi les coups possibles sur une profondeur p (p coups à l'avance). L'algorithme appliqué à une position x à la profondeur p procède ainsi :
	- Si la position x n'a pas de successeurs ou si la profondeur p=0 est atteinte :
	  ○ Renvoyer l'heuristique de la position x
	- Si la position x présente des successeurs :
	  ○ Si c'est un coup du joueur :
	  ▪ Calculer le résultat de l'algorithme min-max pour tous les successeurs de x à la profondeur p-1 pour l'adversaire
	  ▪ Renvoyer le maximum des résultats obtenus
	  ○ Si c'est un coup de l'adversaire :
	  ▪ Calculer le résultat de l'algorithme min-max pour tous les successeurs de x à la profondeur p-1 pour le joueur
	  ▪ Renvoyer le minimum des résultats obtenus
	
	Exemple pour la position initiale à la profondeur p=3 du joueur 1 :
	 
	Remarque : lorsque l'heuristique est choisie comme proposé ci-dessus, et lors d'une application de min-max sur toute la profondeur du graphe, min-max retourne le caractère gagnant(∞), perdant $(-\infty)$ ou nul (0) d'une position