Apprentissage par observation

IFT-17587
Concepts avancés pour systèmes intelligents
Luc Lamontagne

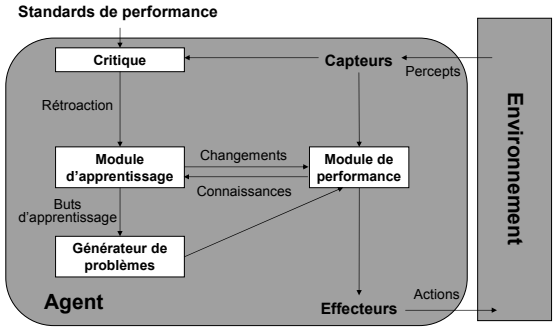
Plan

- Agent apprenant
- Apprentissage inductif
  - Arbres de décision
- Apprentissage d'ensembles d'hypothèses

Apprentissage par observation

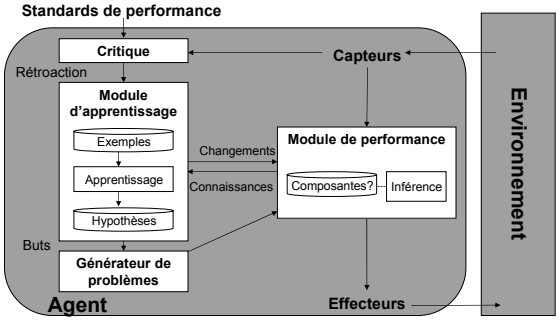
- Perceptions
  - Utilisées pour choisir une action.
  - Mais aussi comme exemples pour améliorer l'agent.
- Apprentissage
  - Essentiel pour des environnements inconnus.
  - Modifie les mécanismes de décision de l'agent.
- Agent apprenant
  - Simplifie la conception.
  - Permet à l'agent d'avoir plus de flexibilité.
  - Peut agir dans des environnements inconnus.
    - Et peut devenir meilleur avec le temps.

Agent apprenant



Agent apprenant :

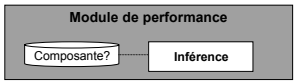
Apprendre quelle composante ?



Agent apprenant :

Apprendre quelle composante ?

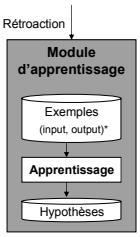
- Toutes les composantes peuvent être apprises
  - Réflexes
    - Règles condition-action : Si le fantôme me suit, je me sauve!
  - Modèles
    - Environnement : Un fantôme jaune à moins de 3 cellules est dangereux.
    - Action : Si je mange un *pacgum*, le fantôme devient bleu.
  - Buts ou utilités
    - Choix: Si un fantôme me poursuit, il vaut mieux protéger mon nombre de vie que d'augmenter mon nombre de points.



Agent apprenant :

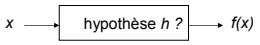
Type d'apprentissage

- Apprentissage supervisé
  - Pour chaque exemple, on a :
    - des *inputs* décrivant un état ou une situation (perceptions).
    - un *ouptut* indiquant le résultat attendu de l'inférence.
      - la bonne réponse!
      - elle provient d'une perception ou d'une rétroaction.
- Apprentissage non supervisé
  - Aucun indice sur l'*output* à produire.
  - L'agent apprend à partir des relations entre les inputs.
  - Il apprend à prédire les nouveaux inputs à partir de ceux du passé.
- Apprentissage par renforcement
  - L'*output* à produire est inconnu.
  - L'évaluation d'une action est faite par récompense ou punition.



Apprentissage inductif

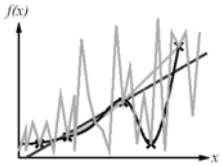
- La plus simple forme d'apprentissage
- Apprendre une fonction à partir d'exemples
  - On veut apprendre une fonction cible *f*.
  - On a des exemples qui sont des paires (x, $f(x)$).
  - Problème: trouver une hypothèse *h*
    - tel que $h \approx f$
    - étant donné un ensemble d'exemples d'entraînement



- Modèle très simplifié

Méthode d'apprentissage inductif

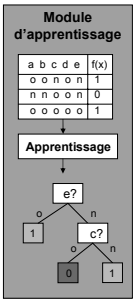
- *h* est compatible (*consistent*)
  - si elle donne la même chose que *f* sur tous les exemples.
- But : trouver la fonction *h* qui estime le mieux la fonction *f* sur les exemples.
  - Ex: Ajustement de courbes



- Laquelle choisir ?
- « *Ockham's razor* »: Trouver l'hypothèse la plus simple qui est compatible.

Apprentissage d'arbres de décision

- Une des formes les plus simples d'apprentissage inductif
  - Tout de même une de celles qui connaissent le plus de succès.
- À partir d'exemples, apprendre des structures d'arbre permettant de prendre des décisions.
  - Chaque nœud représente un test à faire.
  - Chaque branche représente une valeur possible résultant du test.
  - Une feuille correspond à la décision.



Arbres de décision :

Exemple du restaurant

- En arrivant au restaurant, devrait-on attendre qu'une table se libère ?

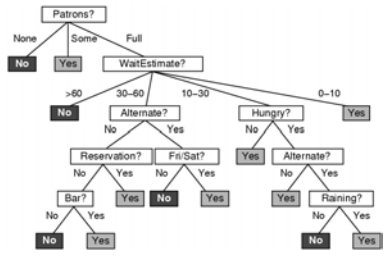
Example	Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	*Yes*
X_2	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	*No*
X_3	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	*Yes*
X_4	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	*Yes*
X_5	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	*No*
X_6	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	*Yes*
X_7	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	*No*
X_8	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	*Yes*
X_9	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	*No*
X_{10}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	*No*
X_{11}	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	*No*
X_{12}	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	*Yes*

- Les exemples décrits par des valeurs d'attributs (ex. booléen, discret, continu)
- La classification des exemples est positive (*yes*) ou négative (*no*)

Arbres de décision :

Exemple du restaurant

- En arrivant au restaurant, devrait-on attendre qu'une table se libère ?



Apprentissage d'arbres de décision

- Les instances représentés par des pairs (attribut, valeur)
  - Les valeurs peuvent être discrètes ou continues.
- La fonction visée est souvent binaire
  - Possible de les étendre pour avoir plusieurs valeurs en sorties.
- Peuvent donner en sortie des valeurs réelles
  - Surtout utilisé pour les problèmes d'optimisation.
- Utile pour des problèmes de classification
- Les exemples peuvent contenir du bruit.
  - Bruits = erreurs dans les exemples d'entraînement.
  - Il peut manquer des valeurs pour certains attributs.
  - Arbres de décision robustes aux erreurs.

Construire un arbre de décision

- Apprentissage : Approche diviser-pour-régner
  - On construit l'arbre de décision de haut en bas.
  - On utilise l'ensemble d'entraînement (i.e. les exemples).
  - On met à la racine de l'arbre l'attribut le plus important.
    - Celui qui sépare le mieux les exemples positifs et négatifs
  - On crée un nouveau nœud pour chaque valeur possible de l'attribut.
  - Pour chacun de ces nœuds :
    - On recommence récursivement la construction d'un sous-arbre.
- But
  - Trouver le plus petit arbre qui couvre l'ensemble d'entraînement.
- On ne veut pas uniquement mémoriser les observations
  - Extrapolation d'exemples qui n'ont pas déjà été vus.

Construire un arbre de décision :

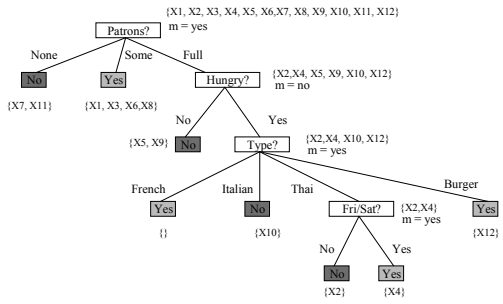
Algorithme

function DECISION-TREE-LEARNING(*examples*, *attributes*, *default*) **returns** a decision tree
  inputs: *examples*, set of examples
    attributes, set of attributes
    default, default value for the goal predicate

  if *examples* is empty **then return** *default*
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** MAJORITY-VALUE(*examples*)
  else
    best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
    tree ← a new decision tree with root test *best*
    m ← MAJORITY-VALUE(*examples*$_i$)
    for each value v_i of *best* **do**
      examples$_i$ ← {elements of *examples* with *best* = v_i}
      subtree ← DECISION-TREE-LEARNING(*examples*$_i$, *attributes* − *best*, *m*)
      add a branch to *tree* with label v_i and subtree *subtree*
    end
    return *tree*

Construire un arbre de décision :

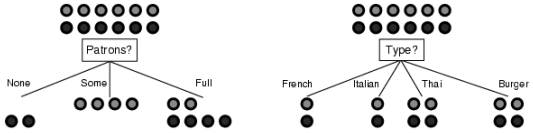
Exemple



Construire un arbre de décision :

Choisir un attribut

- Idée principale:
  - Un bon attribut sépare les exemples en des sous-ensembles qui sont (idéalement) soit « tous positifs » ou « tous négatifs ».



- *Patrons?* est un meilleur choix.

Choisir un attribut:

Le contenu d'information

- Une bonne mesure est la contenu d'information fourni par un attribut
- Défini à l'aide de l'entropie

$$I(P(v_1), \ldots, P(v_n)) = \Sigma_{i=1} -P(v_i) \log_2 P(v_i)$$

- Pour un ensemble d'entraînement constitué de *p* exemples positifs et *n* exemples négatifs :

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Choisir un attribut:

Le gain d'information

- Un attribut *A* divise l'ensemble d'entraînement *E* en sous-ensembles $E_1, \ldots, E_v$ selon les valeurs de *A*, où *A* a v valeurs distinctes.

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

- Le gain d'information (IG) ou la réduction d'entropie :

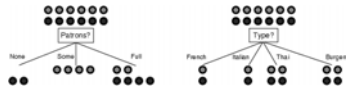
$$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - remainder(A)$$

- On choisi l'attribut avec le plus grand IG

Choisir un attribut:

Exemples de gain d'information

- Dans notre exemple précédent, $p = n = 6$, $I(6/12, 6/12) = 1$ bit
- Pour les attributs *Patrons* et *Type*:



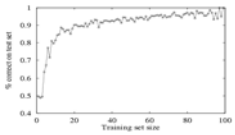
$$IG(Patrons) = 1 - [\frac{2}{12}I(0,1) + \frac{4}{12}I(1,0) + \frac{6}{12}I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{ bits}$$

$$IG(Type) = 1 - [\frac{2}{12}I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12}I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12}I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12}I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

- *Patrons* a le plus grand gain d'information de tous les attributs
- Il serait alors choisi par l'algorithme comme racine de l'arbre

Évaluation de la performance

1. Faire la collecte d'un grand ensemble d'exemples.
2. Les diviser en deux ensembles: entraînement et test.
3. Utiliser l'ensemble d'entraînement pour générer l'hypothèse *h*.
4. Mesurer le % de l'ensemble de test correctement classifiés par *h*.

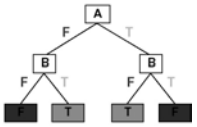


5. Répéter les étapes 1 à 4
   - Faire une sélection aléatoire d'exemples
   - Varier la taille des ensembles d'entraînement

Expressivité

- Toute fonction booléenne peut être représentée par un arbre de décision.
- Approche triviale
  - Faire correspondre chaque ligne de la table de vérité à un chemin dans l'arbre.
- Ceci donne un arbre de grandeur exponentiel
  - Mais on peut souvent faire beaucoup mieux.
  - Il est préférable de trouver des arbres plus compacts.

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Expressivité

- Les arbres de décisions ne sont pas efficaces (grandeur exponentielle) pour représenter certaines fonctions, comme:
  - Fonction de parité: retourne 1 s'il y a un nombre pair de 1 dans les entrées.
  - Fonction de majorité: retourne 1 s'il y a plus de la moitié des entrées à 1.
- Combien d'arbres possibles pour *n* attributs booléens ?
  = le nombre de fonctions booléennes
  = le nombre de table de vérité distinctes de 2^n rangées

Avec 6 attributs booléens, on obtient
18 446 774 073 709 551 616 arbres possibles!

Les arbres de décision en pratique

- Avantage de l'approche
  - Facile pour un humain de comprendre le résultat.
- Mais quelques points à considérer :
  - Il faut éviter le sur-entraînement (*overfitting*).
  - Quoi faire avec les données manquantes?
  - Certains attributs sont inutiles.
  - Des attributs (input) prennent des valeurs continues.
  - Des catégories sont parfois des valeurs continues.
- Des algorithmes prennent en compte ces facteurs.

Ensemble d'hypothèses

- Pourquoi classifier avec une seule hypothèse?
  - 2 têtes valent mieux qu'une! Et M têtes…
- *Ensemble learning*
  - On induit plusieurs hypothèses.
  - On combine leur prédiction individuelle par un vote.
  - Et on retourne une seule prédiction.
- Par ex. si on a 5 arbres de décisions
  - Pour évaluer une instance, on exécute les 5 arbres
  - On retourne la réponse qui revient le plus souvent.
    - Par ex. si 2 vrais et 3 faux, on retourne faux.
- On dénomme cette approche *bagging*.

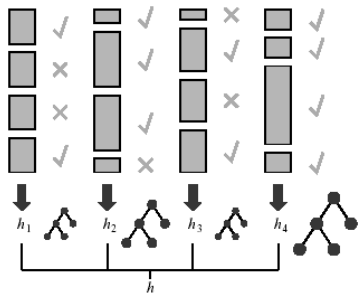
Ensemble d'hypothèses :

L'approche « *boosting* »

- Autre approche pour générer un ensemble d'hypothèses.
- Idée : attribuer un poids à chaque exemple.
  - On varie le poids des exemples pour mieux apprendre à classifier les exemples difficiles.
  - Plus un exemple a un grand poids élevé, plus il sera considéré important par l'algorithme d'apprentissage.
- Plusieurs versions de cette approche
  - *AdaBoost* en est un exemple.

Ensemble d'hypothèses :

Algorithme de *boosting*



Ensemble d'hypothèses :

Algorithme de *boosting*

- Affecter un poids de 1 à chaque exemple.
- Répéter *M* fois :
  - Générer une hypothèse à partir des exemples
  - Augmenter le poids des exemples mal classifiés.
  - Diminuer le poids des exemples bien classifiés.
  - Assigner un poids à l'hypothèse (un niveau de confiance)
- Résultat = un ensemble d'hypothèses
  - Une décision correspond à la somme pondérée des votes des *M* hypothèses
  - Le poids d'une hypothèse dépend de sa performance sur l'ensemble d'entraînement.
    - Les hypothèses plus performantes sont plus crédibles.

Ensemble d'hypothèses :

Algorithme de *boosting*

```
function ADABOOST(examples, L, M) returns a weighted-majority hypothesis
  inputs: examples, set of N labelled examples (x_1, y_1), ..., (x_N, y_N)
          L, a learning algorithm
          M, the number of hypotheses in the ensemble
  local variables: w, a vector of N example weights, initially 1/N
                   h, a vector of M hypotheses
                   z, a vector of M hypothesis weights

  for m = 1 to M do
      h[m] ← L(examples, w)
      error ← 0
      for j = 1 to N do
          if h[m](x_j) ≠ y_j then error ← error + w[j]
      for j = 1 to N do
          if h[m](x_j) = y_j then w[j] ← w[j] · error/(1 − error)
      w ← NORMALIZE(w)
      z[m] ← log (1 − error)/error
  return WEIGHTED-MAJORITY(h, z)
```


Conclusion

- Il existe différentes formes d'apprentissage.
- L'apprentissage supervisé apprend une fonction apprend une fonction à partir d'exemples.
- L'heuristique du gain d'information permettent un apprentissage efficace des arbres de décision.
- Les ensembles d'hypothèses offre souvent de meilleures performances que celle des hypothèses individuelles.