Arbres de décision

E. Gaussier

UGA - Lab. d'Informatique Grenoble

Exemple de données

jour	ciel	temp.	hum.	vent	jouer
1	soleil	chaud	élevée	faible	non
2	soleil	chaud	élevée	fort	non
3	couvert	chaud	élevée	faible	oui
4	pluie	doux	élévée	faible	oui
5	pluie	froid	normale	faible	oui
6	pluie	froid	normale	fort	non
7	couvert	froid	normale	fort	oui
8	soleil	doux	élevée	faible	non
9	soleil	froid	normale	faible	oui
10	pluie	doux	normale	faible	oui
11	soleil	doux	normale	fort	oui
12	couvert	doux	élevée	fort	oui
13	couvert	chaud	normale	faible	oui
14	pluie	doux	élevée	fort	non

Généralités

On notera $c(x)$ la fonction qui à chaque exemple x associe sa catégorie ("oui" ou "non" dans l'exemple *Jouer au tennis*)

Définition 1 : une hypothèse h est **cohérente** avec un ensemble d'apprentissage $\mathcal{D}$ *ssi* $h(x) = c(x)$ pour tous les éléments de $\mathcal{D}$

Définition 2 : l'**espace des versions**, noté $\mathcal{VS}_{h\mathcal{D}}$, d'un ensemble d'hypothèses $\mathcal{H}$ et d'un ensemble d'apprentissage $\mathcal{D}$ est le sous-ensemble de $\mathcal{H}$ formé des hypothèses cohérentes avec $\mathcal{D}$

Construction de l'espace des versions (1)

Première solution : l'algorithme **LIST-THEN-ELIMINATE**

1. $\mathcal{VS}_{h\mathcal{D}} \leftarrow$ contient toutes les hypothèses
2. Pour chaque élément de $\mathcal{D}$, supprimer de $\mathcal{VS}_{h\mathcal{D}}$ tous les éléments x tels que $h(x) \neq c(x)$

Construction de l'espace des versions (2)

Deuxième solution : l'algorithme CANDIDATE-ELIMINATION

Initialiser $\mathcal{G}$ et $\mathcal{S}$

Pour chaque élément x de l'ensemble d'apprentissage

- Si x est positif
  - Supprimer de $\mathcal{G}$ les hypothèses non cohérentes avec x
  - Pour chaque hypothèse s de $\mathcal{S}$ non cohérente avec x
    - Supprimer s de $\mathcal{S}$
    - Ajouter à $\mathcal{S}$ toutes les généralisations minimales h de s telles que h est cohérente avec x et il existe une hypothèse dans $\mathcal{G}$ plus générale que h
    - Supprimer de $\mathcal{S}$ toutes les hypothèses non spécifiques
- Si x est négatif, idem cas précédent en inversant les rôles de $\mathcal{S}$ et $\mathcal{G}$

Commentaires sur l'algorithme

L'espace des versions appris par l'algorithme CANDIDATE-ELIMINATION converge vers la "bonne" hypothèse si :

1. Il n'y a pas d'erreurs dans nos données d'apprentissage
2. La bonne hypothèse est dans $\mathcal{H}$

Si les données sont bruitées, l'algorithme converge vers l'ensemble vide.
Remarque : *cet algorithme trouve la bonne solution sous les conditions ci-dessus*

Les arbres de décision : représentation

À partir des données *Jouer au tennis*, apprendre des règles de décision de type :

Si ((A $\wedge$ B $\wedge$ D) $\vee$ (C $\wedge$ F)) alors ... sinon ...

- Disjonction de conjonctions (forme disjonctive normale)
- Réalise une catégorisation
- Règle de catégorisation est interprétable par un humain (si ...)
- Représentation sous forme d'arbres :
  - Les différentes branches d'un même nœud correspondent à des disjonctions
  - Les parcours descendants à des conjonctions

Exemple $(Outlook = Sunny \;\&\; Humidity = Normal) \vee (Outlook = Overcast \vee Humidity = High)$

Dans quels cas utiliser les arbres de décision ?

1. Les exemples peuvent se représenter sous forme de paires attributs-valeurs
2. La fonction cible est discrète
3. Les descriptions sous forme de disjonction sont utiles
4. L'ensemble d'apprentissage peut contenir des erreurs
5. Des valeurs d'attributs peuvent ne pas être repésentées dans l'ensemble d'apprentissage
6. Mais : nombre d'attributs limité

Algorithme de construction

Algorithme ID3 (Quinlan, 1986)

Créer un nœud racine (exemples positifs et négatifs, ensemble d'attributs $\mathcal{A}$ non vide).

ID3($\mathcal{D}$,*fonction_cible*,$\mathcal{A}$)

1. A $\leftarrow$ attribut qui classifie *le mieux* $\mathcal{D}$
2. Attribut de décision pour la racine $\leftarrow$ A
3. Pour chaque valeur possible v de A
   - Ajouter une branche sous la racine correspondant au test $A = v$
   - Soit $\mathcal{D}_v$ le sous-ensemble de $\mathcal{D}$ ayant pour valeur v pour A
   - Si $\mathcal{D}_v$ est vide ou ne sélectionne qu'une valeur de *fonction_cible*, alors ajouter une feuille dont l'étiquette est la valeur la plus fréquente de *fonction_cible* dans $\mathcal{D}$
   - Sinon, ajouter le sous-arbre ID3($\mathcal{D}_v$,*fonction_cible*,$\mathcal{A} - A$)

Déterminer le meilleur attribut (1)

L'entropie comme la mesure de l'impureté d'un ensemble
Cas binaire : $\mathcal{S}$ contient des exemples positifs et négatifs

$$E(\mathcal{S}) = -p(+)log(p(+)) - p(-)log(p(-))$$

Exemple sur 14 exemples, 9 sont positifs et 5 sont négatifs

$$E(\mathcal{S}) = 0.94$$

Forme de $E(\mathcal{S})$
****

****

Déterminer le meilleur attribut (2)

1. Généralisation : $E(\mathcal{S}) = \sum_{i=1}^{c} -p(i)log(p(i))$
2. Gain d'information : mesure la réduction d'entropie apportée par la partition d'un ensemble sur la base d'un attribut

$$\text{Gain}(\mathcal{S}, A) = E(\mathcal{S}) - \sum_{v} \frac{|\mathcal{S}_v|}{|\mathcal{S}|} E(\mathcal{S}_v)$$

Exemple $\mathcal{S} = (9+,5-)$. 6+ et 2- avec $Wind = Weak$ et le reste avec $Wind = Strong$
****

****

Déterminer le meilleur attribut (3)

Illustration sur les données *Jouer au tennis*

$Gain(S,Outlook) = 0.246$
$Gain(S,Humidity) = 0.151$
$Gain(S,Wind) = 0.048$
$Gain(S,Temperature) = 0.029$
****

****

Exercice (1)

Construire l'arbre de décision à partir de l'ensemble :

N^o exple	valeur att. 1	valeur att. 2	valeur att. 3	classe
1	0	1	0	+
2	0	1	1	+
3	0	0	0	+
4	1	0	1	-
5	1	0	0	-

Exercice (2)

Le problème du sur-apprentissage (1)

Définition 6 : une hypothèse h dans $\mathcal{H}$ *sur-apprend* sur $\mathcal{D}$ s'il existe h' dans $\mathcal{H}$ telle que h' a un plus grand taux d'erreur que h sur $\mathcal{D}$, mais un plus petit sur l'ensemble $\mathcal{X}$ de tous les exemples possibles.

1. Courbe taille de l'arbre/précision de la règle
2. Courbe complexité/précision

Solutions (apprentissage et validation) :

1. Arrêt précoce de croissance de l'arbre (mais difficile à mettre en œuvre)
2. Post-filtrage

Le problème du sur-apprentissage (2)

Un algorithme simple de post-filtrage :

1. Considérer tous les nœuds un par un (candidat au filtrage)
2. Un nœud (en fait son sous-arbre) est enlevé si l'arbre obtenu après filtrage n'est pas moins bon sur l'ensemble de validation

Variante de type filtrage de règles (permet de conserver une partie des sous-arbres) (C4.5)
Exemple IF (Outlook=Sunny) $\wedge$ (Humidity=High) THEN PlayTennis = No
Supprimer (Outlook=Sunny) ? $\rightarrow$ évaluation du résultat
Supprimer (Humidity=High) ? $\rightarrow$ évaluation du résultat