Apprentissage statistique et optimisation
Arbres de décision

Exercice 1 : Arbre de décision (sur feuille)

Vous avez observé des objets de deux classes (+ ou -).

forme	taille	couleur	classe
rond	petit	blanc	+
carré	petit	rouge	-
rond	petit	vert	+
carré	moyen	blanc	+
rond	petit	rouge	-
rond	petit	jaune	+
rond	moyen	jaune	-
rond	moyen	blanc	+
carré	grand	blanc	+
carré	petit	rouge	-
carré	grand	vert	-

1. Quelle quantité d'information est donnée par `couleur = blanc` ?
2. Quelle est la quantité d'information de chaque attribut ?
3. Déroulez l'algorithme d'arbre de décision et montrez l'arbre obtenu.

Solution:

1. Nous posons $p = \mathbb{P}(Class = 1|couleur = blanc) = 1$. La quantité d'information donnée par `couleur = blanc` notée $I(couleur = blanc)$ est, par définition :

$$\begin{aligned} I(couleur = blanc) &= -plog(p) - (1-p)log(1-p) \\ &= 0 \end{aligned}$$

   avec la convention $0\log(0) = 0$.
2. En notant val_i les différentes valeurs possibles pour l'attribut att, la quantité d'information de att est définie par $\sum_i \mathbb{P}(att = val_i) I(att = val_i)$.

Par exemple,

$$\begin{aligned} I(couleur) = \quad & \mathbb{P}(couleur = blanc) I(couleur = blanc) + \\ & \mathbb{P}(couleur = rouge) I(couleur = rouge) + \\ & \mathbb{P}(couleur = vert) I(couleur = vert) + \\ & \mathbb{P}(couleur = jaune) I(couleur = jaune) \\ = \quad & \tfrac{4}{11} \times 0 + \\ & \tfrac{3}{11} \times 0 + \\ & \tfrac{2}{11} \times \log(2) + \\ & \tfrac{2}{11} \times \log(2) \\ = \quad & \tfrac{4}{11} \log(2) \\ \approx \quad & 0.25205 \end{aligned}$$

Par des calculs analogues, nous obtenons :

$$\begin{aligned} I(forme) = \quad & -\tfrac{6}{11}(\tfrac{2}{3}\log(\tfrac{2}{3})\tfrac{1}{3}\log(\tfrac{1}{3})) - \tfrac{5}{11}(\tfrac{2}{5}\log(\tfrac{2}{5}) + \tfrac{3}{5}\log(\tfrac{3}{5}))) \\ \approx \quad & 0.6531 \\ I(taille) = \quad & \tfrac{6}{11}\log(2) - \tfrac{3}{11}(\tfrac{2}{3}\log(\tfrac{2}{3}) + \tfrac{1}{3}\log(\tfrac{1}{3}))) + \tfrac{2}{11}\log(2) \\ \approx \quad & 0.6777 \end{aligned}$$

3. On considère l'algorithme vu en cours :
   - si l'ensemble d'entraînement ne contient qu'une seule classe, on retourne cette classe et l'algorithme s'arrête.
   - sinon on cherche l'attribut *att* le plus informatif. Pour chaque valeur *val* possible de *att*, on exécute récursivement l'algorithme en restreignant l'ensemble d'apprentissage aux exemples tels que $att = val$.

   L'algorithme *att* le plus informatif est tel que $I(att)$ soit **minimal**.

   Explications (cf. cours *Decision Trees* d'Andrew W. Moore ou *Apprentissage automatique* d'Antoine Cornuéjols et Laurent Miclet) :

   On pose X et Y deux variables aléatoires telles que X prend les valeurs $(x_i)_{i=1}^n$ et Y les valeurs $(y_j)_{j=1}^m$.

   On définit

   (a) l'**entropie** de X par :

$$H(X) = -\sum_{i=1}^{n} \mathbb{P}(X = x_i) \log(\mathbb{P}(X = x_i)) \tag{1}$$

   (b) l'**entropie conditionnelle de Y sachant $\mathbf{X = x_i}$** par :

$$H(Y|X = x_i) = -\sum_{j=1}^{m} \mathbb{P}(Y = y_j | X = x_i) \log(\mathbb{P}(Y = y_j | X = x_i)) \tag{2}$$

   (c) l'**entropie conditionnelle de Y sachant X** par :

$$H(Y|X) = -\sum_{i=1}^{n} \mathbb{P}(X = x_i) H(Y|X = x_i) \tag{3}$$

(d) le **gain d'information de Y sachant X** par :

$$IG(Y|X) = H(Y) - H(Y|X) \tag{4}$$

L'entropie (éq. 1) représente la quantité d'information de la variable X (plus l'entropie de X est grande, plus la valeur de X est incertaine).

L'équation 2 donne l'incertitude sur Y sachant que $X = x_i$, et l'équation 3 mesure l'entropie résultante de Y une fois que X est connu.

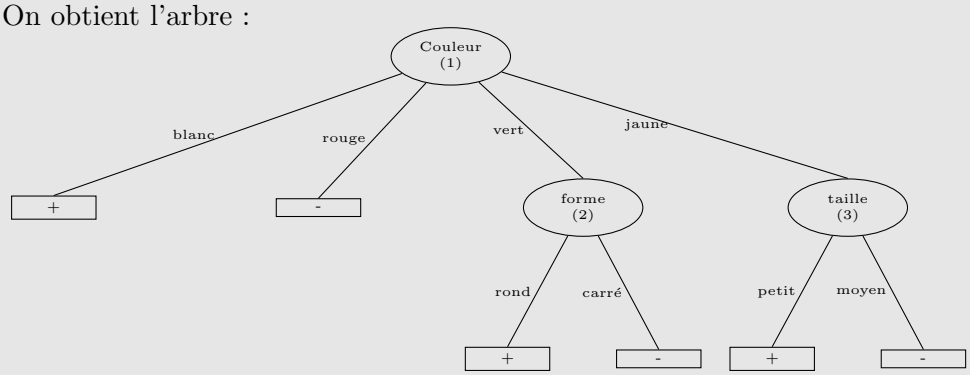
Le gain d'information (éq. 4) est alors la réduction d'entropie gagnée grâce à la connaissance de X.

En posant, comme dans le cours, $Y = Class \in \{0, 1\}$ et $X = att$, nous obtenons par (2) $I(att = val) = H(Class|att = val)$, et par (3) $I(val) = H(Class|att)$.

Nous choisissons l'attribut maximisant le gain d'information sur la classe $IG(Class|att)$. Par l'équation (4), cela revient à minimiser $H(Class|att) = I(att)$ ($H(Class)$ est constant).

Application sur l'exemple

On obtient l'arbre :



Avec (attribut choisi en gras) :

	$I(forme)$	$I(taille)$	$I(couleur)$
(1)	0.6531	0.6777	**0.25205**
(2)	**0**	**0**	-
(3)	log(2)	**0**	-

(On peut choisir indifféremment forme ou taille en (2))

Exercice 2 : Arbre de décision (implémentation)

Nous allons maintenant implémenter l'algorithme d'apprentissage d'arbre de décision vu en cours. Nous nous limiterons à une classification en deux classes, et où les attributs prennent un nombre fini de valeurs.

Pour l'implémentation Octave, chaque attribut, valeur d'attribut et classe sont représentés par un entier.

Exemple (exercice 1) :

1. `Forme`, `taille` et `couleur` sont respectivement les attributs 1, 2 et 3.
2. Pour l'attribut `forme`, `rond` est représenté par 1 et `carré` par 2. Pour l'attribut `taille` (. . .)
3. La classe + est la classe 1, la classe - la classe 0.

Nous proposons de représenter l'ensemble d'apprentissage à l'aide de deux matrices :

– une matrice X contenant les exemples. Chaque ligne de X représente une entrée de l'ensemble d'apprentissage, chaque colonne un attribut.
– un vecteur Y dont la i-ième composante est la classe de la i-ème ligne de X.

Exemple (exercice 1) : La deuxième ligne de X s'écrit (2 1 2) et $Y(2) = 0$.

On veut être capable, étant donnés X et Y, d'**afficher** l'arbre de décision correspondant. Nous décomposons pour cela le travail en deux fonctions :

1. Une fonction `[att_min info_min] = calculInfo(X, Y, selAtt)` qui, pour chaque attribut *att*, calcule $I(att)$ et renvoie l'attribut le plus informatif et la quantité d'information I associée.
   On prendra garde à ne pas considérer les attributs présents dans le vecteur `selAtt`.
2. Une fonction `constrArbre(X, Y, prof, selAtt)` qui, en se basant sur `calculInfo` affiche récursivement l'arbre de décision.

Pseudo-code de `calculInfo(X, Y, selAtt)` :

```
Fonction [att_min info_min] = calculInfo(X,Y, selAtt)
        att_min = 0, info_min = +inf, I=0
        pour att dans X non precedemment selectionne
                pour v valeur de att
                                p=Pr(Class=1|att=v)
                                si (0<p<1)
                                        Iv = -p*log(p) - (1-p)*log(1-p)
                                sinon
                                        Iv = 0
                                fin si
                                I = I + p(att=val)*Iv
                fin pour
                si I < info_min
                        info_min = I
                        att_min = att
                fin si
                I=0
        fin pour
        retourner [att_min info_min]
Fin Fonction
```

Remarques :
– On prend la convention $\log(0)0 = 0$.
– Lors de la construction du tableau, nous sélectionnons un attribut att pour un noeud (le plus informatif). Nous créeons ensuite un noeuds fils pour chaque valeur v que peut prendre att. Pour chacun de ces nouveaux noeuds, les valeurs de att sont fixées.
  Pour éviter de sélectionner plusieurs fois le même attribut, on tient à jour une liste des attributs sélectionnés que `calculInfo` doit ignorer.
– La fonction `unique(x)` renvoie un vecteur contenant les éléments du vecteur x par ordre croissant et en un unique exemplaire. On peut donc boucler sur chaque valeur de l'attribut j grâce aux lignes suivantes :

```
val=unique(x(:,j);
for v=val'
     (...)
endfor
```

Questions :

1. Implémenter `calculInfo`.
2. Implémenter `constrArbre` en vous basant sur l'algorithme exposé en cours.
3. Afficher l'arbre de décision de l'exercice 1.

Solution:

On propose la solution suivante :

```
%calcule pour chaque attribut (colonne de x) la quantite d'information relative
%retourne l'attribut minimisant cette quantite et la valeur d'information associee
function [att_min, info_min] = calculInfo(x,y,selAtt)
        att_min=0;
        info_min=inf;
        I=0;
        for j=1:size(x,2)%on itere sur les attributs.
                if (sum(selAtt==j)==0)%si l'attribut n'a pas ete deja selectionne
                        val=unique(x(:,j)); %val contient les valeurs possibles de l'attribut courant
                        for v=val'%calcul de I(att)
                                p=sum(y(x(:,j)==v))/sum(x(:,j)==v);
                                if (p>0 && p<1)
                                        Iv=-p*log(p)- (1-p)*log(1-p);
                                else
                                        Iv=0;
                                endif
                                I+=sum(x(:,j)==v)/size(x,1)*Iv;
                        endfor
                        %mise a jour du min
                        if info_min > I
                                info_min=I;
                                att_min=j;
                        endif
                        I=0;
                endif
        endfor
endfunction
```

```
%Affiche l'arbre de decision relatif a l'ensemble d'apprentissage X,Y.
function a=constrTree(X,Y,depth,selAtt)
        if (length(Y)<1)
                printf("Y vide !");
        elseif (size(unique(Y),1)==1)%si classe unique, on l'affiche
                printf("classe %d \n",Y(1));

        else %sinon on selectionne l'attribut le plus informatif, et on appelle recursivement l'algorithme

                [att, info]=calculInfo(X,Y,selAtt);
                printf("attribut %d profondeur %d info %f \n",att,depth,info);
                values=unique(X(:,att));
                for v=values'%pour toutes les valeurs possible de att
                        printf("profondeur %d valeur att %d \n",depth,v);
                        pos=find(X(:,att)==v);
                        constrTree(X(pos,:),Y(pos,:),depth+1,[selAtt att]);
                endfor
        endif
endfunction
```