Adrien
KERISIT
TD - J

Rapport de conception : arbre de décision

Types choisis pour manipuler l'arbre :

Matrice : les données en entrée du programme sont stockées dans cette structure.
La première colonne de cette matrice est la variable à prédire ou la variable « Y ». Les autres colonnes sont les variables de prédiction (Xi)

Nœud : une structure correspondante à un nœud de l'arbre y compris la racine et les feuilles
Elle permet de manipuler et parcourir l'arbre facilement. J'ai choisi de créer cette structure comme vu précédemment en TD.
Chaque nœud comporte un ensemble d'individus qui se trouvent dans la variable struct tab_lignes[int]. Cette dernière contient non pas les individus choisis pour le nœud, mais les index de ces individus dans la matrice de départ. La variable tab_lignes est Null pour la racine, car il s'agit de l'ensemble des individus de la matrice.

Fonctions et algorithmes :

Les fonctions applicables sur tous les nœuds sont les suivantes :

int ChoixVariable (nœud)*
Elle permet de choisir la variable la mieux adaptée pour diviser les individus d'un nœud.
A chaque nœud en entrée en commençant par la racine et pour chaque variable de division Xi, la fonction calcule d'abord la médiane corrigée. Ensuite en parcourant chaque individu du nœud, elle calcule la précision des nœuds fils qui découleront de la division et les stocke dans des variables locales accuracyLeft [double] et accuracyRight [double]. La précision des individus de la division est calculée en fonction du pourcentage des individus ayant la même espèce que celle entrée par l'utilisateur et stockée dans le nœud également.

C'est ensuite la précision la plus élevée qui indiquera la meilleur variable de division pour ce nœud.

A ce niveau, la fonction crée les deux fils gauche et droits du nœud.
Affecte les membres du parent :
    indexVariable, median
Ainsi que les membres des fils :
    nb_lignes, precision,

La fonction renvoie l'index de la colonne de la matrice.

*void split(noeud * parent, int index);*
Cette fonction parcourt tous les individus du nœud se trouvant dans la variable *tab_lignes* et opère réellement, à l'aide de la meilleure variable déjà choisie (index), la division des individus selon ces critères :

- Elle ajoute un individu dans fils_gauche si la valeur Y de l'individu est <= médiane corrigée Xchoisi (index).
- Elle ajoute un individu dans fils_ droite si la valeur Y de l'individu fils_droite si Y est > médiane corrigée Xchoisi (index)

A noter que l'ajout de l'individu se fait par l'ajout de l'index de l'individu au membre tab_lignes des fils.

Création de l'arbre :

*construire_Arbre (noeud * racine, int hauteurMax,double minAccuracy,double maxAccuracy,int compteur, double minEchantillon)*

La fonction récursive « construire_Arbre » appelle la fonction de création des fils gauche et droite (« split »). C'est cette dernière fonction qui exécute à son tour la fonction « ChoixVariable » pour une division des individus plus performante. C'est à l'aide de ces deux fonctions que les nœuds des fils sont construits, remplis d'informations et liés entre eux.

La création de l'arbre, à son tour, se fait via cette fonction récursive qui se répète tant que l'une des conditions d'arrêts ; à savoir : seuil précision min et max, hauteur max et pourcentage min d'un échantillon ; ne sont pas atteintes.

Fonctionnalités du menu :

Choix menu :
1-/ *Hauteur* : cette fonction récursive renvoie la hauteur d'un sous arbre ou alors 0 si le noeud est NULL. Cette dernière est la condition d'arrêt.
C'est donc un compteur récursif, qui à chaque rappel de la fonction, cherche la hauteur maximum des deux branches gauches et droites de l'arbre et s'incrémente par 1.

L'effet général est que le compteur comptera toujours la hauteur d'une des branches (jamais les deux) tant qu'au moins l'une des deux est pas NULL.

2-/ *Largeur* : cette fonction récursive renvoie de la même manière que la fonction hauteur la largeur d'un sous arbre ou alors 1 si le noeud est une feuille. En effet, ici la condition d'arrêt permet d'augmenter la largeur et la largeur finale n'est plus la largeur d'une des branches mais l'addition de celle des deux branches.

3-/ Affichage arborescent :
*void affichage_arborescence(noeud const *arbre,int decal)*

La fonction de l'affichage arborescent, utilise la même fonction que celle vue en TD6 sur les arbres. Elle affiche récursivement pour chaque nœud de l'arbre les informations qu'il contient. Elle appelle ensuite les fils gauche et droits en incréments à chaque appel le décalage de l'impression.

La condition d'arrêt est lorsque la fonction récursive atteigne une feuille i.e. les fils son NULL.

A noter qu'à la lecture également, l'ordre hiérarchique est basé sur le décalage de l'affichage d'un nœud par rapport au nœud qui le suit. Un nœud qui n'est pas suivi par un nœud décalé à droite est une feuille. En effet, le décalage est utilisé pour bien visualiser la parenté/fraternité des nœuds.

4-/ Affichage des feuilles:
*void print_feuille(noeud const * racine,char chemin[])*

Il s'agit ici d'une fonction récursive Print_feuille qui parcourt l'arbre jusqu'à ce qu'elle arrive à une feuille et affiche les informations qu'elle contient. En revanche les informations doivent être précédées par le chemin du nœud.

Pour renvoyer le chemin le paramètre « chemin » une chaîne de caractère contient l'information de chaque nœud et la concatène avec les informations des nœuds déjà parcourus.

De plus, l'utilisation d'un buffer de taille assez large facilite la tâche de l'allocation de mémoire à chaque appel de la fonction grâce aux fonctions sprintf et strcat.

La récursivité de la fonction permet de renvoyer comme chemin la nouvelle chaîne de caractères qui concatène le chemin passé en paramètre et le chemin contenu dans le buffer.

5-/ Prédiction :
*void Predict(noeud const * nœud, double * newUser, double * precisionVar)*

Au départ, les valeurs des variables Xi de l'individu que l'utilisateur doit saisir dans un tableau sont stockées dans un tableau de double.

Ensuite la fonction récursive Predict de type void parcours l'arbre de décision créé et pour chaque nœud compare la médiane du nœud avec la valeur de l'individu pour la variable de meilleure division du nœud (Xi) qui est stockée dans le nœud (le membre indexVariable).

Si la valeur de Xi est inférieure ou égal à la médiane, c'est la fonction récursive « Predict » pour le fils gauche qui est appelé. Dans le cas contraire, la fonction est appelée pour le fils droit.

La condition d'arrêt est lorsque les fils sont NULL et dans ce cas, la précision du nœud dont le fils est NULL est affectée au paramètre « precisionVar ».

A noter que le paramètre « precisionVar » est un paramètre de sortie. En effet, pour pouvoir récupérer la valeur de ce paramètre, c'est l'adresse de la variable « precisionVar » qui est utilisée à l'appel.

	précision pour Y=2	X1	X2	X3	X4
1	0.034483	7.7	3.0	6.1	2.3
2	0.034483	6.2	2.8	4.8	1.8
3	1.000000	5.5	2.5	4.0	1.3
4	0.000000	6.7	3.3	5.7	2.5
5	0.944444	6.0	2.2	5.0	1.5
6	0.034483	6.0	2.7	5.1	1.6
7	1.000000	5.7	2.6	3.5	1.0
8	1.000000	5.8	2.6	4.0	1.2
9	0.000000	5.1	3.4	1.5	0.2
10	0.000000	5.4	3.9	1.3	0.4