Rapport projet OCaml

LEVEQUE Clément

2024

Table des matières

1 Choix des structures

Pour choisir l'implémentation du type `tree`, on a choisi trois types :
- `Empty`, représentant un arbre vide, qu'on utilisera lorsque la liste de points est vide.
- `Node of pos * orientation * tree * tree`, représentant donc un noeud que l'on utilisera lorsqu'on sépare la zone. On sépare donc la zone en fonction d'une position `pos` et d'une orientation `orientation`, et on a en résultat un arbre à gauche et un arbre à droite `tree * tree`
- `Leaf of color`, représentant une feuille qui nous donne la couleur de la position.

Comme décrit ci-dessus, nous avons utilisé un type `orientation` défini par `Horizontal / Vertical`

1.1 Prédiction

Cette implémentation nous permet un parcours d'arbre simple et clair. En effet, il suffit à chaque itération d'observer si l'on se trouve sur un arbre vide (au quel cas on renvoie par convention une prédiction blanche), si l'on se trouve sur une feuille (au quel cas on a atteint la couleur à prédire) ou si l'on se trouve sur un noeud et dans ce cas il suffit de comparer la position, et de vérifier l'orientation de la séparation pour savoir dans quel sous-arbre aller.

1.2 Construction

Cette manière de créer le type `tree` permet également de renvoyer simplement la couleur avec une `Leaf`, et de marquer de manière claire et consise une séparation de la liste avec la position et l'orientation. De plus, la création de `Empty` permet de manière logique de renvoyer un arbre même si le dataset est vide.
Finalement, cette implémentation permet de ne pas se perdre dans le code de création de l'arbre.

2 Choix de l'implémentation des fonctions

2.1 Build tree

Tout d'abord il a fallu faire la fonction de création de l'arbre `build_tree n training`
On a utilisé plusieurs fonctions auxiliaires que nous décrirons si nécessaire dans la partie qui suit. Avec la fonction `build_tree_aux`, nous procédons comme ceci :
- Tout d'abord nous vérifions que nous n'avions pas atteint la profondeur maximale, dans le cas où elle est atteinte, on renvoie une `Leaf` qui indique la couleur majoritaire de la section sur laquelle on se trouve avec la fonction `most_point_color`.
- Sinon, on vérifie si il y a une seule couleur dans la section en cours, si c'est le cas on renvoie cette couleur avec la fonction `only_one_color`.
- Si aucun des cas si dessus n'est vérifié, on commence le processus de séparation :
  - On calcule les positions moyennes pour les points rouges et pour les points bleus avec la fonction `avg_x_y`.
  - On teste les deux séparations (entre la séparation verticale et la séparation horizontale) avec la fonction `best_separation`.
  - On sépare en fonction du type de séparation retenu avec soit la fonction `separe_horizontal` soit `separe_vertical`.
  - On ajoute ces deux séparations à l'arbre en créant un `Node` où la position de la séparation est le milieu des positions moyennes rouge et bleu.

Les fonctions de séparation utilisent une fonction `sort_color`, qui sépare une liste en deux listes, une liste avec des points bleus et une autre avec des points rouges.
La fonction `best_separation` calcule les variances pour les deux types de séparation et renvoie la séparation ayant la plus petite variance.

2.2 Predict color

Pour faire la fonction `predict_color`, on utilise une fonction récursive `predict_color_aux` de parcours d'arbre.

- Si l'arbre est vide, alors par convention on retourne une prédiction blanche.
- Si on se situe sur une feuille, on renvoie la couleur prédite
- Sinon, on se situe sur un noeud.
  - On regarde le type de séparation, vertical ou horizontal.
  - On continue le parcours d'arbre en prenant celui qui correspond à la position du point recherché et à l'orientation de la séparation.

3 Problèmes techniques et solutions

3.1 Type tree

Le premier choix d'arbre de comportait pas de type `Empty`. Le problème rencontré est donc survenu lors de la construction de l'arbre. En effet, si le dataset est vide, il fallait quelque chose à retourner.
`Empty` fut donc ajouté au type `tree`.
Ensuite, lors de la première compilation du programme, même si le programme tournait, il affichait une précision de 66%. Après avoir imaginé d'autres manières de séparer une section, un problème très simple fut trouvé dans la fonction `avg_x_y` : lors du calcul de la moyenne, je ne divisais pas par `n`. Une fois ce problème corrigé, les compilations suivantes affichaient :

- 96.1% pour `generate_circle`
- 97.7% pour `generate_sqrt`
- 99.0% pour `generate_l_r` (une fonction qui met une couleur à gauche et l'autre à droite)

4 Limites

4.1 Profondeur maximale

Tout d'abord, il est important de noter que pour une raison inconnue, le paramètre de profondeur maximale de dépasse pas les 10 (soit la profondeur maximale par défaut). On peut la baisser, et les résultats changent en conséquence, mais l'augmenter au delà de 10 ne change rien, le programme fonctionne mais ne va pas au delà de `n=10`
Ensuite :

- Si la profondeur est égale à 1, alors le programme agit comme un intégrateur, en effet il donne l'aire de la figure tracée (78.9% pour la cercle, 77.4% pour la fonction racine, 49.8% pour la séparation verticale)
- Si la profondeur est inférieure ou égale à 3, on obtient des précisions inférieures à 90% (sauf évidemment pour la séparation verticale, qui donne une précision de 99% dès une profondeur de 2)

On en déduit donc que sans surprise, si la profondeur est trop faible, les prédictions ne sont pas fiables.

4.2 Taille dataset

De même, si la taille du dataset, est inférieure à 40, les précisions descendent en dessous des 90% (toujours en excluant la séparation verticale)