rapport projet ocaml

Valentin Capieu

May 2024

1 choix de résolution du problème

Le problème consiste à créer un arbre de prévision de pronfondeur maximale fixe sur un jeux de données et de faire en sorte qu'il soit le plus précis possible.
Pour cela la 1ère partie consiste à créer ce **type tree**. Étant donnée que la profondeur est fixe et que le jeu de données est restraint dans le carré délimité par les points $(0,0)$ et $(1,1)$, j'ai décide de donner à mes noeuds 4 sous arbres qui sont les sous-abres de prévisions restraint en coupant horizontalement et verticalement au milieu du carré. Cela m'évite au passage de devoir gérer l'alternance de coupure horizontale et verticale qui est présente dans les arbres binaires. Chanque noeud contient de plus une position **pos** correspondant à l'endroit ou la coupure verticale et horizontale à été effectué. L'autre partie de mon **type tree** sont les feuilles qui ne peuvent contenir qu'une couleur.

```
type tree =
   |Leaf of color
   |Node of pos * tree * tree * tree * tree ;;
```

Ainsi, aller à la profondeur 10 dans mon arbre reviens à aller jusqu'à la profondeur 20 dans un arbre binaire. Ce qui me permet de ne pas être limité la profondeur maximale de mon arbre.

2 problèmes rencontrées et solutions

le premier problème que j'ai rencontré fut celui de la gestion des listes vides dans ma fonction **build_tree**. En effet, le jeux de données ne contenant que 1000 points, mon programme en découpant la zone initiale en 4^{10} petits carrés se retrouvait avec beaucoup de petits carrés vides (99.9% pour être précis). Cela implique que les feuilles de mes premiers abres de prévisions étaient toutes de la même couleur.

La solution que j'ai trouvé consiste à arrêter le découpage dès lors que la taille de l'échantillon sur lequel mon programme travaille est inférieure où égale à 4. Ainsi ça réduit fortement les probabilités que mon programme coupe un carré qui n'a pas besoin de l'être.

```
if (List.length set)<5 then Leaf(couleur_predominante set)
```

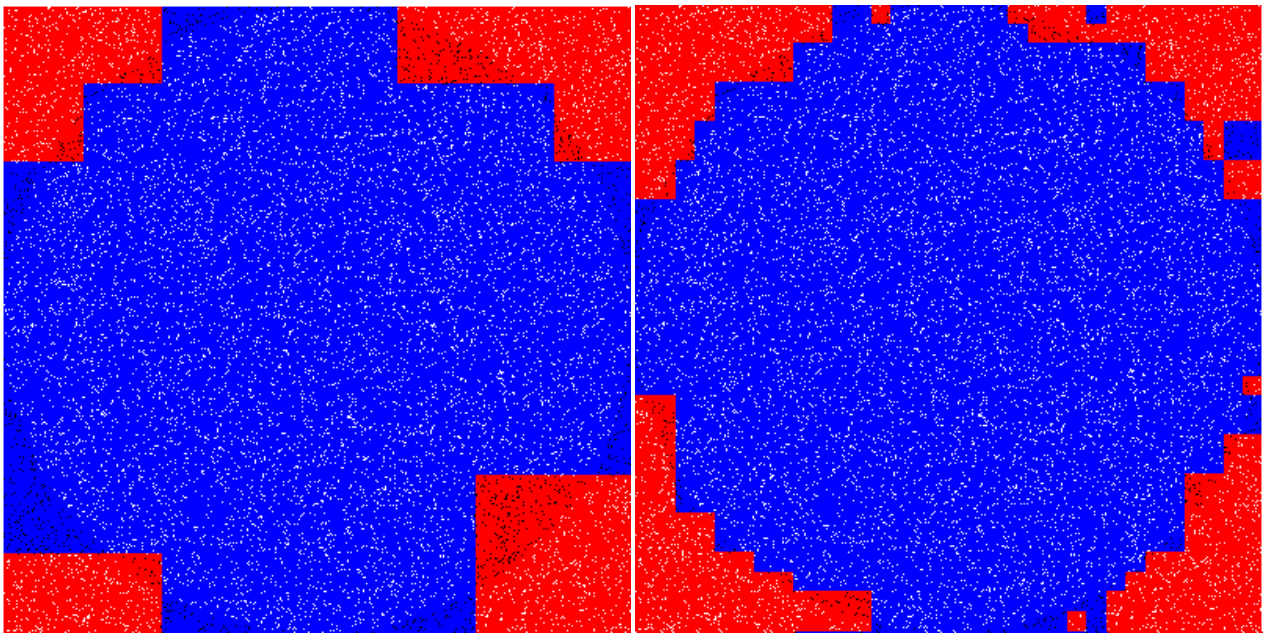
Mais même s'il y a plus de 4 éléments, rien ne dit que tous les éléments vont être réparti dans chacun des 4 carrés si l'on relance un découpage. Il se peut que l'un des carrées fils se retrouve sans aucun point et donc se voit attribuer une couleur par défaut. Cette situation est difficilement gérable dans le cas ou l'échantillon contient des points de couleurs différentes, mais l'est moins lorsque le programme travaille sur une zone où les points sont de couleurs identiques.

J'ai donc décidé d'implémenter un nouveau test, qui consitste à compter le nombre de point rouge et le nombre de point bleu à l'intérieur d'un dataset. Ainsi, dès qu'une liste ne contient qu'une seule couleur, pas besoin d'effectuer un découpage.

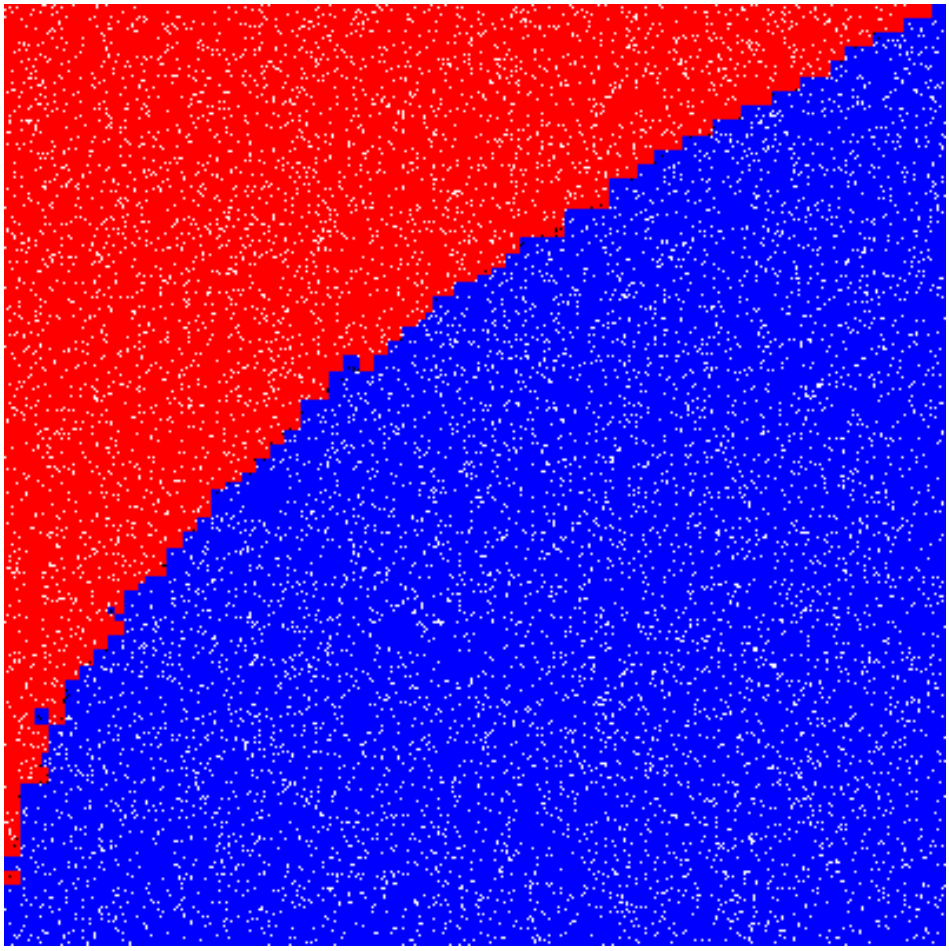
```
let (r,b)= compteur set in if (r==0) then Leaf(blue) else if (b==0) then Leaf(red)
```

3 limites du programme

Mon programme n'est pas très sensible à la profondeur maximale. Il l'est néanmoins beacoup plus sur la taille de l'échantillon donné à la fonction **build_tree**. en effet la précision lorsque l'échantillon est de taille 100 est d'environ 90% alors que sur un échantillon de taille 1000 la précision est d'environ 95%.



Mais cette sensibilité marche dans les deux sens : Avec un échantillon de taille 10000, j'arrive à dépasser les 99% de précision.



Concernant la complexiter temporelle de mon programme, il tourne et renvoie ses résultats en moins d'une seconde. On peut aussi s'amuser à regarder combien de temps il mettrait si l'on rentre une grande valeur pour la taille de l'échantillon.
Au dela de 200 000, j'ai une erreur de stack overflow (sûrement que l'optimisation de la complexité spatiale doit laisser à désirer). Mais par contre en dessous de 200 000 il met toujours à peine 1 ou 2 secondes à s'effectuer, donc j'en conclut que la complexité temporelle est assez faible.

J'ai aussi fait d'autre tests; par exemple (attention ce que j'ai fait est illégal j'ai modifié temporairement la partie après le "AUCUNE MODIFICATION NE DOIT ETRE FAITE A PARTIR DE CE COMMENTAIRE") j'ai réduit la profondeur maximale autorisé (de 10 à 5) juste pour voir si ma méthode consistant à couper en 4 plutôt qu'en 2 sert à quelque chose. Et au final la précision ne change pas sur tous mes test; ma précotion s'avère donc inutile.

4 Conclusion

En conclusion, mon programme tourne rapidement avec une présion décente (environ 95% sur toutes les fonctions de test avec une profondeur de 10 et un échantillon de départ de taille 1000).

Mais j'ai investit beacoup de temps pour rendre mon algorithme insensible à la profondeur maximale par peur que celle-ci soit un facteur déterminant de l'éfficacité de mon programme alors qu'une profondeur 10 suffisait largement, même pour les arbres binaires.

J'aurais donc mieux fait d'investir ce temps à chercher une autre méthode pour séparer mon échantillon que simplement couper en 4 au milieu (qui est la méthode naïve par exelence) par exemple essayer en regardant les coordonnées médianes des points d'une couleur, ou faire la moyenne entre le barycentre des points rouge et le barycentre des points bleus.