Leclercq Robin
Lambert Alexandre
Janssens Hugo

Analyse projet phase 1.1

Nous avons décidé de traiter les deux tableaux donnés (realClasses et estimateClasses) contenant chacun NB_TESTS cellules en un tableau à 2 dimensions (NB_CLASSES x NB_CLASSES) qui va réunir l'ensemble des résultats et qui va permettre de garder trace des différents choix qu'aura fait la machine en fonction de la classe correspondante. Cette étape sera réalisée dans la fonction traitementTableau2D().

Sa définition et son initialisation : `int resultsPerClass[NB_CLASSES][NB_CLASSES] = { 0 }`

Deux nouveaux tableaux seront ensuite créés : theoricalClassesPerClass et correctResultsPerClass correspondant respectivement au nombre d'apparences théoriques de chaque classe et le nombre de fois qu'une telle classe est apparue et a été correctement traitée. Ces deux tableaux contiennent NB_CLASSES cellules. La fonction resultsForEachClasses() va se charger d'utiliser le tableau resultsPerClass et de remplir ces deux tableaux :

Leurs définitions et initialisations :

```
int theoricalAppearencePerClass[NB_CLASSES] = { 0 };
int correctResultsPerClass[NB_CLASSES] = { 0 };
```

Pour terminer la partie traitement, un tableau contenant, pour chaque classe, le pourcentage correct de résultat sera créé ainsi qu'une variable de type double afin de connaitre la précision totale du système de mesure. La précision totale sera calculée à l'aide des deux tableaux présentés dans le paragraphe ci-dessus et non avec le tableau percentageCorrectResultPerClass afin d'éviter une perte de précision.

Voici leurs définitions :

```
double percentageCorrectResultPerClass[NB_CLASSES] = { 0 };
double averagePercentageCorrectResult;
```

Il est vrai qu'il a une redondance d'informations en créant ce dernier tableau (percentageCorrectResultPerClass) car il serait possible, dans une fonction d'affichage, d'utiliser les deux tableaux du deuxième paragraphe et d'appeler une fonction calculant la moyenne sur 2 nombres et de l'appeler sur les NB_CLASSES cellules. Nous avons fait ce choix car, de la sorte, les calculs de moyenne n'ont lieu qu'une seule fois. Ce procédé est intéressant si la fonction affichage en question est appelée un certain nombre de fois et permet ainsi d'économiser les ressources du processeur. Le point négatif est la création d'un tableau supplémentaire contenant un maximum de 20 double.

Leclercq Robin
Lambert Alexandre
Janssens Hugo

Ensemble des prototypes de fonctions :

```
 void  traitementTableau2D(int realClasses[], int estimateClasses[], int
resultsPerClass[NB_CLASSES][NB_CLASSES]);
 void  resultsForEachClasses(int theoricalAppearencePerClass[], int
correctResultsPerClass[], int resultsPerClass[NB_CLASSES][NB_CLASSES]);
 void  percentageAccuracyPerClass(double percentageCorrectResultPerClass[], int
theoricalResultsPerClass[], int measuredResultsPerClass[], double* overallAccuracy);
 void  displayResultsForEachClasses(int theoricalAppearencePerClass[], int
correctResultsPerClass[], double percentageCorrectResultPerClass[]);
 void  displayAccuracy(double* averagePercentageCorrectResult);
 void displayClass(int theoricalAppearencePerClass[], int resultsPerClass[NB_CLASSES]
[NB_CLASSES]);
```

Remarques : Nous avons utilisé des pointeurs lorsque cela était possible afin de privilégier le passage par adresse, moins couteux en mémoire. Nous avons également choisi de travailler avec des tableaux car l'utilisation de listes chainées nécessite, dans chaque chainon, de contenir la donnée ainsi qu'un pointeur vers le chainon suivant. Dans le prototype de la fonction displayClass(), il est possible de n'utiliser que le tableau à 2 dimensions mais dans l'optique de ne pas effectuer de calcul dans une fonction affichage et d'utiliser le tableau theoricalAppearencePerClass déjà créé, il est utilisé.