

## TP9

On considère le problème de calcul scientifique de la reconstruction de paramètres d'une interface droite

$$D = \{(x, y) \mid \cos \theta x + \sin \theta y = d\}.$$

Ici on a un patch de  $(2N + 1) \times (2N + 1)$  cellules carrées unités d'aire chacune égale à 1, avec  $N = 1$ .

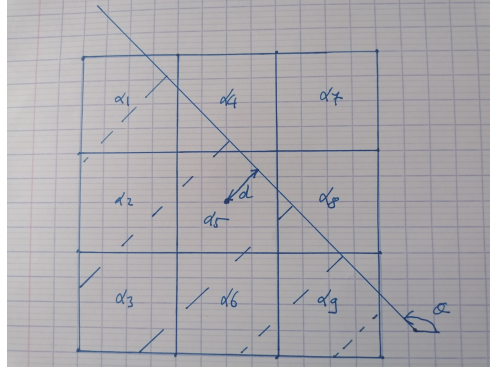


Figure 1:  $\alpha_2 = \alpha_3 = \alpha_6 = 1$  et  $\alpha_7 = 0$ .

Les fractions de volume son

$$0 \leq \alpha_i = \int_{C_i \cap \{a+bx+cy < 0\}} dx dy \leq 1.$$

Le problème est de reconstruire  $\theta$  à partir de la connaissance de  $\alpha_i$   $1 \leq i \leq (2N + 1)^2$ . Plus précisément, on cherchera à reconstruire un réseau de neurones qui réalise

$$\begin{aligned} f: \mathbb{R}^9 &\rightarrow \mathbb{R}^2 \\ (\alpha_i)_{i=1}^9 &\mapsto (\cos \theta, \sin \theta). \end{aligned}$$

. Nous allons voir comment créer cette fonction en Python/Keras/Tensorflow, puis comment exporter cette fonction dans un code C++.

- Charger `~despres/interface_create.py`, et analyser comment le soft construit par intégration numérique un dataset de type

$$\mathcal{D} = \{(\cos \theta, \sin \theta, \alpha_1, \dots, \alpha_9) \text{ pour une variété de } (\theta, d)\}.$$

Bien vérifier que l'échantillonnage en angle est tous les demi-degrés.

- Charger `~despres/interface_learn.py`, et analyser comment le soft "apprend"  $\Leftrightarrow$  faire du fit non-linéaire  $\Leftrightarrow$  fait de l'optimisation numérique pour construire  $f$ . Essayer d'obtenir une précision "moyenne" de l'ordre du 1%.

Activer les lignes

```
data = np.array([[1,1,1,0.5,0.5,0.5,0,0,0]])
prediction = model.predict(data)
print("prediction=",prediction)
```

et interpréter le résultat.

- Pour exporter le modèle en C++, on propose d'utiliser le soft *keras2cpp* <https://github.com/gosha20777/keras2cpp> dû à Georgy Perevozchikov. Installer *keras2cpp* sur votre ordinateur et faire l'exemple (il faut copier `example.model` dans *build*).

```
$ git clone https://github.com/gosha20777/keras2cpp.git
$ cd keras2cpp
$ mkdir build && cd build
$ python3 ../python_model.py
[[-1.85735667]]

$ cmake ..
$ cmake --build .
$ cp ../example.model .
$ ./keras2cpp
[ -1.857357 ]
```

Charger `~despres/cpp_interface.cc`, modifier la dernière ligne du fichier `CMakeLists.txt`, relancer la compilation, copier `interface.model` dans *build*, et lancer l'exécutable. Comparer le résultat avec la prédiction du point précédent.

- Construire le dataset avec une méthode d'Hypercube Latin, et comparer les résultats.
- Reprendre la chaîne de calcul du début et travailler avec des patches  $5 \times 5$ . Comparer les résultats (si possible).