AS - TP5 - Graphes de calcul

Module nngraph

Nous allons nous intéresser à l'implémentation de structures de calculs plus complexes que de simples réseaux de neurones profonds. Pour cela, nous allons utiliser la bibliothèque *nngraph* (déjà installée sur les machines) et décrite à l'adresse `http://github.com/torch/nngraph`.

Modules complexes

A l'heure actuelle, nous avons utilisé des modules dont les entrées et les sorties sont des tenseurs. Cependant, il est possible d'utiliser des modules dont les entrées et/ou les sorties sont plusieurs tenseurs. Ces modules sont décrits ici : `https://github.com/torch/nn/blob/master/doc/table.md#nn.TableLayers`

Voici un exemple d'implémentation à tester vous même :

```
x=torch.randn(5)
y=torch.randn(5)
m=nn.CAddTable()
m:forward({x,y})
```

Ces modules permettent d'imaginer des architectures plus complexes.

0.1 Exercice

Nous souhaitons apprendre la fonction suivante : $f_W(x) = sigmoid(W.x + W_0) + x$. Le module $W.x + W_0$ correspond à un module *nn.Linear* classique.

- Implémentez (sans *nngraph*) l'algorithme de descente de gradient sur une base jouet (aléatoire par exemple) d'exemples (x, x) (de type auto-encodeur)

Modules complexes avec nngraph

Implémentation avec nngraph (voir explications aux tableau) - et `https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/practicals/practical5.pdf`

- Implémentez la fonction précédente en *nngraph*

– Sauvegardez la structure sous forme de fichier *dot* afin de visualiser le graphe construit
– Testez la module créée

Un module plus complexe

Implémentez le réseau suivant :

$$f_{\gamma,\theta}(x) = f_\theta(x \circ Relu(\gamma)) \tag{1}$$

où $\circ$ correspond au produit 'terme à terme', avec une régularization L_1 sur γ.

Quel est l'intéret de ce module ?