AS - TP1 - Prise en main de Torch et régression linéaire

Ludovic Denoyer

Tutorial LUA

Lire le tutorial LUA : `http://tylerneylon.com/a/learn-lua/`

Les tenseurs Torch

Les tenseurs torch sont décrits ici :

1. `https://github.com/torch/torch7/blob/master/doc/tensor.md`
2. `https://github.com/torch/torch7/blob/master/doc/maths.md`

Modèle linéaire

Nous allons implémenter une descente de gradient sur une modèle linéaire classique $f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^m$ de deux façons différentes. La première consistera en l'implémentation algorithmique de l'algorithme de descente de gradient ; la second utilisera le package *optim* de torch et permettra l'utilisation de plusieurs méthodes d'optimisation.

Grands principes

Torch dispose de multiples *modules* (voir `https://github.com/torch/nn/blob/master/README.md`) permettant l'implémentation de méthodes dérivables de façon modulaires (voir "petit guide des réseaux de neurones profonds"). Le module *nn.Linear* permet d'effectuer un produit matricielle simple. Comme tous les modules, il est composé de 4 méthodes importantes :

- **zeroGradParameters()** qui permet de remettre à 0 la valeur du gradient (stocké en mémoire) $Grad \leftarrow 0$
- **forward(x)** qui permet de calculer la valeur $f_\theta(x)$
- **backward(x,delta)** qui permet de calculer le gradient de l'erreur en fonction du gradient calculé sur les sorties du module : $Grad \leftarrow Grad + \nabla_\theta \Delta(f_\theta(x), y)$
- **updateParameters(l)** qui permet de mettre à jour les paramètres du module $\theta \leftarrow \theta - l.Grad$

Torch dispose aussi de *criterions* permettant le calcul d'une fonction coût. Nous allons utiliser le critère des moindres carrés *nn.MSECriterion*. Tout comme les modules, un critère possède les méthodes suivantes :

- **forward(y',y)** qui permet de calculer la valeur $\Delta(y', y)$
- **backward(y',y)** qui permet renvoie la valeur de $\nabla_{y'}\Delta(y', y)$

Faire un "pas" de gradient sur une donnée (x, y) revient ainsi à effectuer les opérations suivantes :

```
module:zeroGradParameters()
output=module:forward(x)
loss=criterion:forward(output,y)
delta=criterion:backward(output,y)
module:backward(x,delta)
module:updateParameters(learning_rate)
```

Exercice

- A partir du fichier `example-linear-regression.lua`, implémentez la descente de gradient stochastique.
- Expliquez ce qu'il se passe dans le fichier `example-linear-regression-optim.lua`.

Tester la méthode avec d'autres dataset issu de `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` en tuilisant le package 'svm' de torch `https://github.com/koraykv/torch-svm`