Introduction à la méthode de Descente du Gradient

April 1, 2022

Outline

1. Introduction
2. Formulation
3. Variantes

Introduction

En effet une formulation mathématique d'un problème d'apprentissage automatique nous renvoi à un problème d'optimisation avec ou sans contrainte.

- Satisfaisante pour résoudre ce type de problème notamment pour les réseaux de neurones.
- Très populaire surtout dans les bibliothèques d'apprentissage profond.
- Mérite une attention particulière sur lui et ses variantes.
- La fonction objective $f : R^d \to R$ doit etre continue et totalement différentiable.

Formulation

Soit $f : R^d \rightarrow R$ une fonction continue et totalement différentiable et θ^0 la valeur initiale du paramètre θ de f. La descente du gradient procède itérativement comme suit (cas de minimisation):

- Calcule $\nabla_\theta f$.
- Met à jour θ:

$$\theta^{t+1} = \theta^t - \eta \frac{\partial f(\theta^t)}{\partial \theta^t} \tag{1}$$

  où η est le taux d'apprentissage.
- L'on répète les étapes précédente jusqu'à la convergence: θ^* tel que

$$\frac{\partial f(\theta^*)}{\partial \theta^*} = 0 \tag{2}$$

Formulation

- Pour la maximisation on peut calculer l'inverse.
- Le $f(\theta^*)$ de convergence est un minimum(maximum) global si la fonction est convexe sinon il est peut etre local.

Variantes

On distingue 3 vairantes de la descente du gradient:

- **Batch Gradient Descent.**
- **Stochastic Gradient Descent.**
- **Mini-Bacht Gradient Descent.**

Variantes

Batch Gradient Descent

Batch Gradient Descent

- la version standard de l'algorithme.
- l'erreur (gradient) est cumulé

$$\theta = \theta - \eta \nabla_{\theta} f(\theta) \tag{3}$$

- est assez lente.

Variantes

Bacht Gradient Descent

Algorithm 1 BGD($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta$)

Initialiser θ^0
$E_0 = 0$
for $i = 1$ **to** n_epoch **do**
  for all (x_d, y_d) in D **do**
    $y_{pred} = get\_prediction(\theta^i, x_d)$
    $E_t = E_{t-1} + get\_error(y_{pred}, y_d)$
  end for
  caluler $\nabla E(\theta)$
  $\theta^i = \theta^{i-1} - \eta \nabla E(\theta)$
end for
return θ^i

Variantes

Stochastic Gradient Descent

Stochastic Gradient Descent

- on calcule le gradient et met à jour θ pour chaque instance du jeu de données

$$\theta = \theta - \eta \nabla_\theta f(\theta, x_i, y_i) \tag{4}$$

- plus rapide mais assez instable surtout proche du minimum(maximum).

Variantes

Stochastic Gradient Descent

Algorithm 2 BGD($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta$)

Initialiser θ^0
$E_0 = 0$
for $i = 1$ **to** n_epoch **do**
  shuffle(D)
  for all (x_d, y_d) in D **do**
    $y_{pred} = get\_prediction(\theta^i, x_d)$
    $E_t = E_{t-1} + get\_error(y_{pred}, y_d)$
    caluler $\nabla E(\theta)$
    $\theta^i = \theta^{i-1} - \eta \nabla E(\theta)$
  end for
end for
return θ^i

Variantes

Mini-Bacht Gradient Descent

Mini-Bacht Gradient Descent

- la mise à jour ce fait après chaque lot de taille fixé, le gradient est cumulé tout au long du lot.

$$\theta = \theta - \eta \nabla_{\theta} f(\theta, x_{i:i+n}, y_{i:i+n}) \tag{5}$$

- A les advantages de la vitesse de la version stochastic et la stabilité de la version batch.
- Difficulté a trouvé la bonne taille pour le lot.

Variantes

Mini-Bacht Gradient Descent

Algorithm 3 mini_bacht($D = \{(x_1, y_1)...(x_n, y_n)\}, n\_epoch, \eta, bacht\_size$)

Initialiser θ^0
$E_0 = 0$
for $i = 1$ **to** n_epoch **do**
  shuffle(D)
  for bacht **in** get_bacht(D, bacht_size) **do**
    for all (x_d, y_d) in bacht **do**
      $y_{pred} = get\_prediction(\theta^i, x_d)$
      $E_t = E_{t-1} + get\_error(y_{pred}, y_d)$
      caluler $\nabla E(\theta)$
      $\theta^i = \theta^{i-1} - \eta \nabla E(\theta)$
    end for