Pierre Marion - AMAL

Description des différents algorithmes de descente de gradient et de leurs grands principes

Batch : c'est la version la plus basique de descente de gradient. On calcule le gradient sur l'ensemble des données, et on descent selon un pas fixe. Cela présente plusieurs inconvénients : lenteur, impossibilité de calculer le gradient en ligne, nécessité de régler le pas de descente.

Stochastic Gradient Descent (SGD) : l'idée de la descente de gradient stochastique est de calculer le gradient en utilisant une seule donnée à chaque itération. Cette technique permet de supprimer les redondances associées au calcul du gradient par batch, et permet l'apprentissage en ligne. Néanmoins, elle possède une très grande variance, qui cause des fluctuations importantes de la fonction objectif.

La descente de gradient **mini-batch**, intermédiaire entre batch et SGD, vise à résoudre ce problème, en calculant le gradient sur un petit nombre de données.

Nesterov Accelerated Gradient (NAG) : cet algorithme vise à réduire les oscillations caractéristiques de SGD au niveau des points selles. Pour cela, la mise à jour de la direction de descente se fait avec de l'inertie (*momemtum*), c'est-à-dire qu'on ajoute au terme de mise à jour une fraction du terme à l'itération précédente. On peut faire une analogie avec une bille qui descent le long d'une pente et qui accumule de l'inertie.

De plus, pour éviter que l'inertie ne soit trop importante, on calcule le nouveau gradient à la position future approchée (calculée grâce au terme d'inertie) et non pas à la position actuelle. Dans notre analogie, cela permet d'éviter que la bille ne remonte trop haut lorsqu'elle atteint la fin de la pente.

Adagrad : les algorithmes présentés jusqu'à maintenant ont un inconvénient majeur : ils utilisent le même pas de descente pour tous les paramètres. Ce comportement est peu adapté au cas des données *sparse*, car on voudrait avoir un pas plus important pour les paramètres rares, et un pas plus faible pour les paramètres fréquents, c'est-à-dire ajouter une propriété d'adaptativité à l'algorithme de descente de gradient.

Pour cela, Adagrad divise pour chaque paramètre le pas de descente par la norme 2 des gradients déjà calculés pour ce paramètre.

RMSprop : cette extension d'Adagrad vise à résoudre son inconvénient principal, qui est la décroissance vers 0 des pas de gradient. Au lieu de diviser par la norme 2 des gradients, on divise par une moyenne des gradients au carré pondérés par un paramètre de *decay*. A noter que cette méthode issue d'un cours de Geoff Hinton n'a pas été publiée.

Adadelta : il s'agit d'une variante de RMSprop, qui permet d'éliminer la difficulté de régler le pas de descente, en remplaçant ce paramètre par une autre moyenne glissante, cette fois celle des termes de mises à jour.

Adaptive Moment Estimation (Adam) : comme son nom l'indique, cet algorithme combine les idées d'adaptativité (Adagrad, RMSprop, Adadelta) et d'inertie (momentum, NAG).