Résumé des différents algorithmes de descente de gradient

Marc Treü & Karmim Yannis
Spécialité **DAC**

1 Les différentes variantes pour la descente de gradient.

La descente de gradient est un algorithme d'optimisation qui a pour but de minimiser une fonction réelle et différentiable. Le gradient a pour but de trouver la direction de la plus forte pente négative. Dans le cas des problèmes en apprentissage supervisé cette fonction à minimiser est la courbe des erreurs empiriques sur notre ensemble d'apprentissage.

Soit : $\mathcal{X} = \left\{\left(\mathbf{x}^i, \mathbf{y}^i\right) \in R^d \times R^c\right\}_{i=1}^N$ notre ensemble d'apprentissage, et L une fonction de coût. Pour un paramètre $\mathbf{w}$, le coût associé à l'ensemble d'apprentissage est :

$$L(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^N L\left(f_{\mathbf{w}}\left(\mathbf{x}^i\right), \mathbf{y}^i\right)$$

Et on cherche :

$$\mathbf{w}^* = argmin_{\mathbf{w}} L(\mathbf{w}) = argmin_{\mathbf{w}} \frac{1}{N}\sum_{i=1}^N L\left(f_{\mathbf{w}}\left(\mathbf{x}^i\right), \mathbf{y}^i\right)$$

On optimise donc notre paramètre $\mathbf{w}$ à l'aide de la descente de gradient :

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon \nabla_{\mathbf{w}} L(\mathbf{w})$$

Il existe néanmoins différentes manières d'effectuer cette descente et donc la mise à jour des poids.

1.1 Descente de gradient stochastique

Dans le cas d'une descente de gradient stochastique on tire un exemple aléatoirement et on pour chacun de ces exemples on effectue la mise à jour sur ces poids. La descente de gradient dans ce cas est minutieuse et précise, en revanche le temps de calcul est beaucoup plus long. Il est également plus difficile d'interpréter la valeur de notre fonction de coût, puisque elle est calculée pour chaque exemple.

1.2 Descente de gradient par batch

La descente de gradient par batch est la mise à jour de notre paramètre $\mathbf{w}$ en prenant en compte tout nos exemples d'apprentissage et en moyennant la fonction de coût sur tout notre ensemble. Cette méthode est plus rapide, cependant c'est souvent irréalisable du fait du grand nombre d'exemples qui ne peuvent tenir en mémoire. Le gradient peut être également plus imprécis, et le pas d'apprentissage doit être parfaitement adapté puisque l'on effectue une seule mise à jour.

1.3 Descente de gradient par mini-batch

La descente de gradient par mini-batch est un compromis des deux méthodes précédentes. On effectue la mise à jour des paramètres sur une partie de l'ensemble d'apprentissage, la descente est plus rapide mais plus imprécise que dans le cas stochastique. La taille de l'ensemble du mini-batch rajoute un hyper-paramètre à notre modèle.

2 Les algorithmes d'optimisations pour la descente de gradient.

2.1 Momentum

Pour la descente de gradient stochastique il est fréquent que l'on oscille autour du minimal local, et que l'on prenne donc plus de temps pour converger. Momentum accélère la convergence en conservant la mise à jour effectué au pas précédent et en la pondérant avec un terme $\gamma < 1$. Quand la nouvelle mise à jour est dans la même direction que la précédente alors on accélère la descente, sinon lorsque on est dans un sens différent ou opposé on va diminuer cette descente.

2.2 Adagrad

Adagrad est un algorithme de descente de gradient qui adapte son pas d'apprentissage en fonction de l'exemple qu'il reçoit, s'il rencontre un exemple qui apparaît fréquemment il va faire des pas plus petit, au contraire pour un exemple peu fréquent des pas plus grand. Il est adapté pour des ensembles d'apprentissages sparse.

2.3 Adam (Adaptive Moment Estimation)

Adam est également un algorithme qui adapte son pas d'apprentissage pour chaque paramètre, en prenant en compte la moyenne et la variance des mise à jour précédentes. C'est un des optimiseurs les plus efficace en pratique.

2.4 AMSGrad

Les algorithmes qui adaptent leurs pas d'apprentissages sont très utilisés dans les réseaux de neurones profonds, cependant dans certains cas il s'avère que ces algorithmes ne converge pas du fait d'un pas d'apprentissage trop petit.
Pour cela on regarde toujours si la mise à jour précédente est plus grande que la courante, si c'est le cas on utilise celle ci afin de ne pas trop diminuer le pas d'apprentissage.