Université de Paris

Perceptron moyenné

Guillaume Wisniewski
guillaume.wisniewski@u-paris.fr

décembre 2020

Nous allons décrire, dans ce document, un algorithme d'apprentissage permettant de déterminer les poids d'un classifieur linéaire à partir d'un ensemble d'apprentissage. Cet algorithme est une généralisation de l'algorithme du perceptron qui permet d'obtenir de très bonnes performances sur de très nombreuses tâches de TAL.

L'algorithme du perceptron, introduit dans le dernier cours, est un algorithme de *correction d'erreur* : lors de l'apprentissage, dès une étiquette erronée est prédite, les poids sont mis à jour afin de corriger cette erreur ou, du moins, réduire l'impact de celle-ci, en diminuant le score de l'étiquette prédite par erreur et en augmentant le score de l'étiquette qui aurait dû être prédite. Cette approche fonctionne lorsque les données sont linéairement séparables : dans ce cas, l'algorithme d'apprentissage converge après un nombre d'itérations polynomial par rapport au nombre d'exemples.

Si les données ne sont pas linéairement séparables (ce qui est en général le cas), il faut introduire un critère d'arrêt en spécifiant, par exemple, un nombre maximum de corrections devant être effectuées. Cette approche est toutefois problématique : si les 999 premiers exemples de l'ensemble d'apprentissage correctement classifiés mais que l'étiquette prédite pour 1 000^{e} exemple est fausse, les paramètres du perceptron seront corrigés sans tenir compte du fait que ceux-ci permettent de classifier correctement les 999 premiers exemples. Il est, de plus, tout à fait possible, que, après cette correction, ces exemples ne soient plus correctement classifiés : l'algorithme d'apprentissage du perceptron n'a pas de « mémoire » et, suivant sa position dans l'ensemble d'apprentissage, une erreur aura un impact plus ou moins grand sur le vecteur de poids.

Une généralisation simple de l'algorithme d'apprentissage du perceptron permet d'éviter ce problème : le perceptron moyenné. Cette généralisation, décrite dans l'algorithme 1 consiste à conserver le vecteur de poids à chaque itération et à utiliser, lors de l'inférence (c.-à-d. après l'apprentissage) un perceptron dont le vecteur de poids est constitué par la moyenne de tous les vecteurs de poids vus lors de l'apprentissage. En pratique, il n'est pas nécessaire de conserver les valeurs de tous les vecteurs de paramètres, mais uniquement la somme de ceux-ci (le vecteur $\mathbf{a}$ dans l'algorithme 1) et il n'est pas nécessaire de diviser celle-ci par le nombre d'itération

puisque un perceptron de paramètre $\mathbf{w}$ fera exactement les mêmes prédictions qu'un perceptron de paramètres $\alpha \times \mathbf{w}$ quelque soit le réel α. Il est primordial de noter que l'apprentissage du perceptron moyenné est exactement le même que l'apprentissage d'un perceptron « normal » et repose sur les corrections successives du vecteur de poids $\mathbf{w}$, le vecteur de poids moyenné $\mathbf{a}$ n'est utilisé que lors de l'inférence, une fois que l'apprentissage est terminée.

Le nombre d'itération MAX_EPOCH est un hyper-paramètre de l'algorithme qui est fixé avant l'exécution de celui-ci et dont la valeur est généralement déterminée par validation croisée.

Algorithm 1 Algorithme d'apprentissage du perceptron moyenné.

Require: a training set $\left(\mathbf{x}^{(i)}, y^{(i)}\right)_{i=1}^{n}$, hyper-parameter MAX_EPOCH

$\mathbf{w} \leftarrow 0$
$\mathbf{a} \leftarrow 0$
for $epoch = 1$ to MAX_EPOCH **do**
  shuffle training set
  for $i = 1$ to n **do**
    $y^* = \text{argmax}_{y \in \mathcal{Y}} \left\langle \mathbf{w}_y | \mathbf{x}^{(i)} \right\rangle$
    if $y^* \neq y^{(i)}$ **then**
      $\mathbf{w}_{y^{(i)}} \leftarrow \mathbf{w}_{y^{(i)}} + \mathbf{x}^{(i)}$
      $\mathbf{w}_{y^*} \leftarrow \mathbf{w}_{y^*} - \mathbf{x}^{(i)}$
    end if
    $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{w}$
  end for
end for
Use $\mathbf{a}$ as the weight vector.