Kernel target alignment

Guillaume Pradel

27 juillet 2024

Table des matières

1 Introduction

1.1 Objectif de l'algorithme

Ce programme a pour objectif de calculer une pondération θ optimale pour un noyau gaussien de la forme :

$$k_\theta(x, x') = \exp\left(-\frac{\sum_{l=1}^{n} \theta_l^2 (\|x_l\|^2 + \|x'_l\|^2 - 2\langle x_l, x'_l \rangle)}{2\sigma^2} \right)$$

Le produit scalaire est défini à partir d'une matrice de Gram spécifique au problème de classification de polypeptides. Cette pondération optimale est calculée à partir de la fonction objectif [1] :

$$A(k_\theta, k_t) = \frac{\langle K_\theta, K_t \rangle_F}{\sqrt{\langle K_\theta, K_\theta \rangle_F \langle K_t, K_t \rangle_F}}$$

[1]Instinctivement, cette fonction s'apparente au cosinus de l'angle entre les deux vecteurs bi-dimensionnelles K_θ et K_t. On parle alors d'alignement de notre noyau de départ K_θ sur le noyau cible *(target kernel)* K_t qui est la matrice de Gram idéale pour la classification de nos données. Ici, nous avons donc pour un problème multi-classes :

$$k_t(x, x') = \begin{cases} 1 & \text{si } y = y' \\ 0 & \text{sinon.} \end{cases}$$

Cette alignement permet en théorie d'améliorer les performances de classification d'une M-SVM.

1.2 Problème et résolution

Le problème ici présent est donc un problème de maximisation car on cherche le θ qui maximise $A(k_\theta, k_t)$, ce qui donnera donc un meilleur alignement entre les 2 noyaux. Ce programme propose une optimisation par le biais d'une descente en gradient sur $-A(k_\theta, k_t)$. La taille des bases d'apprentissage sont généralement très importantes et pour pouvoir calculer une descente en gradient efficacement, il est nécessaire d'utiliser une méthode de décomposition. Celle utilisée ici est la méthode du `chunking` décrite dans [3]. Aussi, il est important que souligner que la composante centrale de θ est fixée à 1 et ce durant toute la descente en gradient.

1. La norme F est ici la norme de Frobenius

2 Utilisation du programme

2.1 Faire fonctionner le programme

Les makefiles sont déjà créés, voici les quelques commandes permettant la bonne utilisation du logiciel.

`calcul_theta` peut être compilée grâce à la commande :

`./compile_calcul_theta`

(Le makefile correspondant se trouve dans le sous-répertoire `make`).

Ensuite, pour pouvoir l'exécuter, il suffit d'utiliser la commande :

`./execute_calcul_theta`

2.2 Structure du fichier contenant les données

Les fichiers contenant les données doivent être des fichiers texte, avec une structure spécifique. Nous illustrons cette structure sur le calcul de l'angle omega de protéines. Le nom du fichier correspondant est `Data/omega.app`.

`76314` ⟵ nombre de données
`7` ⟵ nombre de composantes des vecteurs représentant les données
`4` ⟵ nombre de catégories
`...`

2.3 Paramètres du modèle

Il est possible de spécifier certains paramètres influant sur le calcul de θ, ces paramètres sont modifiables dans : `Fichcom/calcul_theta.com`.
Ce fichier contient les paramètres dans cette ordre :

`1e-3` ⟵ pas de gradient
`100 000` ⟵ nombre d'itérations lors de la descente en gradient
`10` ⟵ taille du *chunk*
`Data/omega.app` ⟵ nom du fichier où sont stockées les données d'apprentissage
`tableau_theta.txt` ⟵ nom du fichier où sont stockées les valeurs de θ calculées

Il existe également d'autres paramètres qui pèsent sur le calcul de θ. Entre autre, il est possible de changer la valeur du θ_0 faisant office de point de départ pour la descente en gradient. Le facteur σ peut également être modifié pour ajuster le résultat. Ces paramètres sont modifiables à l'intérieur du fichier `calcul_theta.c`, à l'endroit où sont assignées les variables.

3 Utilisation pratique pour les SVM

3.1 Utilisation pour les M-SVMs

Lors de l'utilisation du logiciel pour améliorer les performances des M-SVMs (ici [2]), il faut prendre soin de bien changer la fonction noyau avec les paramètres adéquats *(en l'occurence, σ)*. Aussi, il faut copier le θ se trouvant dans le fichier `tableau_theta.txt` et l'insérer dans le fichier `algebre.c`.

3.2 Observations du comportement de l'algorithme sur la base de données Omega

Les paramètres les plus déterminants dans le calcul de θ sont la valeur de σ ainsi que le pas de gradient. Il est conseillé de regarder la valeur du gradient ainsi que de θ pour pouvoir estimer un pas de gradient cohérent. En effet la formule de la descente de gradient est donnée par $\theta_{k+1} = \theta_k - \alpha \nabla A(k_\theta, k_t)$ où α est le pas de gradient. Et il faut nécessairement avoir un ordre de grandeur raisonnable pour α afin de ne pas trop modifier θ à chaque itération. Si l'on prend ces précautions il semble loisible de ne pas faire décroitre le pas de gradient au cours des itérations, car après quelques essais, l'incidence de cette décroissance semble être infime.

Le facteur σ est très important pour la classification. Dans le cas où $2\sigma^2 = 1$, la matrice de confusion obtenue lors de l'apprentissage semble raisonnable. Mais lorsque l'on passe à la base de test, la matrice de confusion indique beaucoup trop de mauvaises classifications dans la catégorie 3 (la plus représentée). Prendre $2\sigma^2 = 8 * dim\_input$ permet de diminuer ce phénomène mais donnera de moins bons taux de classification lors des tests.

θ_0 semble avoir peu d'impact sur la solution finale si l'on considère assez d'itérations (plus de 100000). Pour l'instant, les paramètres donnant les meilleurs résultats en phase de test sont : 500000 itérations, $1e-3$ en pas de gradient, $\theta_0 = \{0.5, 0.5, 0.5, 1, 0.5, 0.5, 0.5\}$, $2\sigma^2 = 1$ et une taille de chunk égale à 10.

3.3 Fonctions implémentées

D'autres fonctions sont implémentées dans ce logiciel. Entre autre, on peut considérer une nouvelle fonction objectif dont on chercherait le minimum :

$$||K_\theta - K_t||^2$$

Une descente en gradient est également implémentée pour cette fonction objectif, il suffit de remplacer le gradient dans la fonction `pas_de_gradient()`.

Aussi, pour faciliter la visualisation du processus d'optimisation, les deux fonctions objectifs sont disponibles et leurs résultats peuvent donc être affichés.

4 Commentaires

Ce programme a été fait en peu de temps et il est possible qu'il existe encore des coquilles dans le code. Pour toutes informations sur ce programme, envoyez un mail à `guillaume.pradel@telecom-sudparis.eu`

Remerciements Merci à Yann Guermeur et Thérèse Malliavin pour leur aide tout autant pour la compréhension du sujet que pour l'implémentation de certaines des fonctions utilisées ici.

Références

[1] Nello Cristianini, John Shawe-Taylor, Andre Elissee, and Jaz Kandola. On kernel-target alignment. *Innovations in Machine Learning*, 194, 01 2002.

[2] Y. Guermeur and E. Monfrini. A quadratic loss multi-class SVM for which a radius-margin bound applies. *INFORMATICA*, 2009. (submitted).

[3] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data : Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 1982.