Lu Lin, Olivier Coudray

16 février 2018

DATA CAMP LAB6

Hyperparameters tuning on SVM models

Résumé

Ce dossier contient un sous-dossiers nommé `DFO_src` et ....

Introduction

Dans ce rapport nous allons tester quatre algorithmes d'optimisation de type boîte-noire pour calibrer les hyperparamètres de l'algorithme de *Support Vector Machine* (SVM) appliqué à notre problème de classification : *Random Search*, *BO (Optimisation bayésienne)* de `scikit-learn`, *CMA-ES* et *DFO-TR*. L'objectif est de tester la performance de ces algorithmes d'optimisation dans un cas pratique. L'objectif n'est donc pas de résoudre directement notre problème mais de trouver la meilleure façon de le résoudre. Nous comparons les résultats obtenus avec le fameux problème de classification MNIST (Lab2 du Data Camp) en tenant compte des spécificités du SVM. Ici, nous considérons uniquement l'algorithme SVC avec le noyau gaussien de `scikit-learn`. Ainsi, la dimension de l'espace de recherche est 2 (γ et C). Sachant que ces deux paramètres sont strictement positifs d'échelles différentes, nous opérons une transformation (logarithmique/affine) [1] pour ramener l'espace de recherche à $[-5, 5]^D$ [2] comme celui de la plate-forme COCO. Nous montrons que cette transformation revient à diminuer le conditionnement (*condition number* κ) et rendre les solvers efficaces. Il est à remarquer que, sans cela, certains solvers seront incapable de fonctionner.

Description des données

MNIST

Il s'agit d'un problème de classification de 10 classes (les chiffres manuscrit de 0 à 9) avec les 70000 images de taille $28 \cdot 28$ (dont 60000 pour le train set et 10000 pour le test set).

1. Comme ce qu'on faisait pour le Lab2.
2. D étant la dimension de l'espace. Ici $D = 2$.

Speech recognition

Nous avons réduit notre problème de classification original de Kaggle de 30 classes à 11 classes. Il s'agit de classifier les mots à partir du son prononcé par un humain de durée environ 1 seconde. Ce sont des vocabulaires anglais simples, comme "yes", "no", "up", "down", "left", "right", "on", "off", "stop", "go" et les autres.

Pré-traitement des données (Preprocessing)

MNIST

Pour le problème MNIST, le seul pré-traitement est la normalisation des images. Il s'agit de diviser chaque pixel par 255 pour que les valeurs soient comprises entre 0 et 1. Faisons une remarque sur lien entre la normalisation et le paramètre γ en résumant le rapport de Lab2 [3]. En fait, la normalisation ici n'est pas obligatoire, mais elle a un impact direct sur la valeur du paramètre γ. En regardant la formule du noyau gaussien (1), on déduit que si on multiplie chaque pixel par 10, il faut diviser γ par 10^2 et garder la même constante C pour avoir la même précision (aka accuracy). Cela a été confirmé numériquement.

$$k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2} = e^{-\gamma'\|x'_i - x'_j\|^2} \tag{1}$$

Dans la section 5, nous allons analyser la structure des données en utilisant cette remarque.

Speech recognition

Le pré-traitement des enregistrements audio s'est fait en deux étapes. Dans un premier temps, il s'agit de recouper les extraits audio de façon à ne sélectionner que la partie où le mot est prononcé. Plus concrètement, on passe un filtre sur le signal pour couper tous les sons en deçà d'un certain seuil d'intensité (dans notre cas 15 dB). Dans un deuxième temps (c'est la partie la plus importante), il faut extraire les données de l'enregistrement. La technique standard consiste à considérer le spectrogramme de l'extrait audio. Dans le cadre de la reconnaissance vocale, il est conseillé d'utiliser un spectrogramme légèrement différent (échelle non linéaire en fréquence) : *Mel-frequency cepstrum*.

Notons que l'on peut considérer ou du moins visualiser ces spectrogrammes comme des images (matrices). A titre d'exemple, on peut observer ci-dessous, les spectrogrammes Mel pour deux mots différents.

Remarque **Lu** : Est-ce que cela tu pourrais compléter cette partie avec le pre-processing que t'as fait ? On pourrait mettre quelques images de spectrogramme. Par exemple, deux spectrogramme similaires de "go" et un spectrogramme très différent (comme celui de "yes" ou autres ?). Cela permet de leur convaincre que le spectrogramme suffit pour classifier ces mots.

3. Veuillez trouver une explication plus complète dans le rapport Lab2 de Lu Lin.

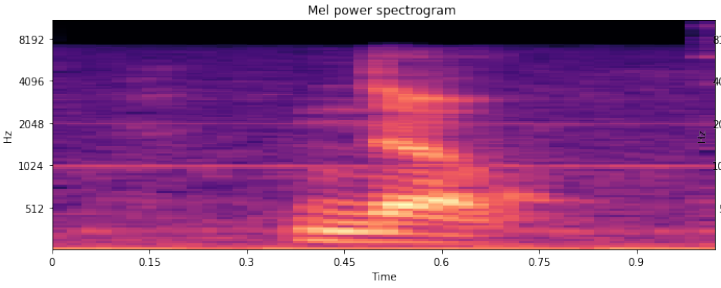

Figure 1 – "No"

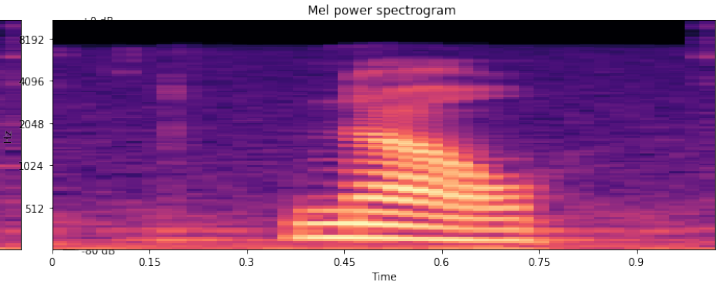

Figure 2 – "No"

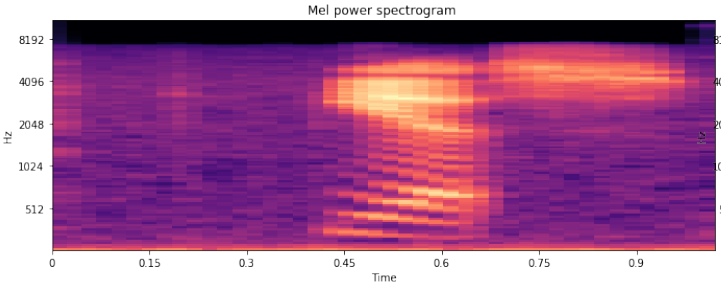

Figure 3 – "Yes"

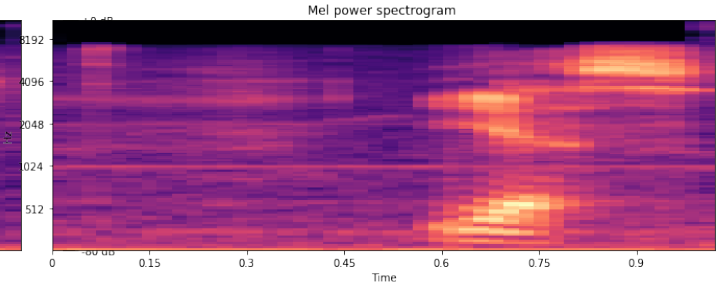

Figure 4 – "Yes"

Transformation de l'espace de recherche

Remarquons que pour la machine SVM avec le noyau gaussien, les deux paramètres γ et C n'ont pas ni les même sensibilités (l'échelle), ni le domaine de recherche. Par exemple, pour le MNIST, les paramètres state-of-the-art sont $\gamma = 0.025$, $C = 10$ [**?**]. D'après les résultats du Lab2, le paramètre C pourrait s'élever à l'ordre de 1000, alors que étant contraint par l'exponentiel du noyau gaussien, le résultat est sensible à la valeur de γ. D'où l'idée de faire au moins une transformation affine. Nous montrons que cela revient à diminuer le conditionnement dans un cas simple. Considérons la fonction ellipsoïde en dimension 2 suivante.

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Si $a > b$, alors le conditionnement $\kappa = (\frac{a}{b})^2$. Il est facile de vérifier que si on fait une transformation affine sur y pour le ramener au même l'ordre de grandeur, on diminue κ, ainsi on simplifie le problème. La Figure (5) explique pourquoi DFO-TR peut-être fortement influencé par ce genre de transformation. Remarquons que les paramètres de DFO-TR qu'on a choisi sont adapté pour la plate-forme COCO, c'est à dire que la zone de recherche est inclue dans $[-5, 5]^D$. Ici, la fonction Step est quasiment plat (constant) dans cette zone et que le minimum se trouve à la frontière. Or, la stratégie de DFO-TR consiste faire une grid-search avec $2D + 1$ points dans la phase initiale [4], en plus la taille de *delta* initiale Δ_0 qu'on avait choisi empêche

4. En partant de zéro, ce grid-search consiste à évaluer les points en perturbant une coordonnée à chaque fois i.e. $(x_1 + \delta, \ldots)$, $(x_1 - \delta, \ldots)$, $(x_1, x_2 + \delta, \ldots)$ ...etc.

l'évaluation de dehors de $[-5, 5]^D$. Tout cela entraîne l'échec de DFO-TR.

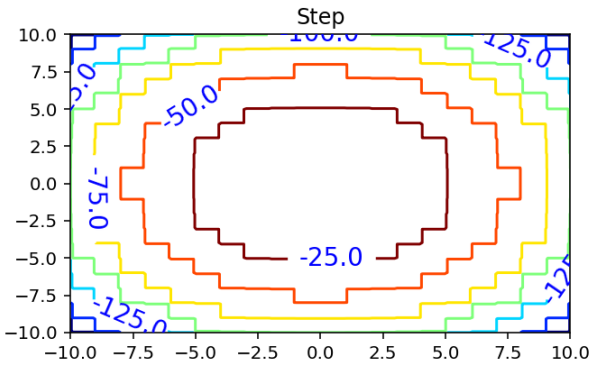

Figure 5 – Step function inverse

Handle the boundary

Dans notre problème

TODO Lu Je vais le faire.

Commentaires et les résultats

MNIST

De nos expériences dans le Lab2, DFO-TR converge au bout de 10 à 30 évaluations. Et les précisions obtenues sont supérieurs à 98%. Le tableau suivant illustre les minima locaux trouvés :

γ	C	size = 7000	size = 70000
0.0413	78.22	0.973	0.9851
0.0242	632.65	0.971	0.9848
0.02845	521.6	0.967	0.985
0.024	100	0.963	0.9864

Nous avons échantillonné 7000 images parmi 70000 pour le hyperparamters tuning. Nous entraînons SVM sur 6000 images et testons sur 1000 images. Les valeurs dans les deux dernières colonnes représentent les précisions sur le *test set*.

Remarque : *Scikit-Learn* a mélangé le *train set* et *test set* de MNIST. Donc, c'est normal d'avoir des précisions légèrement différentes par rapport aux données de *Keras*.

Speech recognition

Dans le traitement des données audio, on sépare le jeu de données en trois datasets : un d'entraînement, un de validation et un de test. Pour des raisons de complexité et de temps de calcul, on ne réalise l'entraînement que sur 1600 données dans cette étude comparative. La validation se fait sur un ensemble de 400 extraits audio.

Dans le cas de DFO-TR et CMA-ES, on note le nombre d'appels à la fonction d'évaluation ainsi que le temps d'exécution global. Dans le cas de *Random Search* et BO, on fixe au départ un budget de 100 évaluations et on note le temps d'exécution.

Optimizer	γ	C	Score	time (sec)	n eval
DFO-TR	5.75	774	0.823	553	35
CMA-ES	10.18	745	0.823	4865	288
Random Search	10.61	946	0.820	1560	100
BO	6.31	500	0.820	1794	100

L'étude montre que l'algorithme DFO-TR est le plus efficace aussi bien du point de vue de l'optimum trouvé mais également et surtout du point de vue du temps de calcul.

L'algorithme CMA-ES, en dépit du temps de calcul important fournit des renseignements intéressant sur la fonction de coût (voir graphe suivant). En particulier, on vérifie que la fonction est séparable.

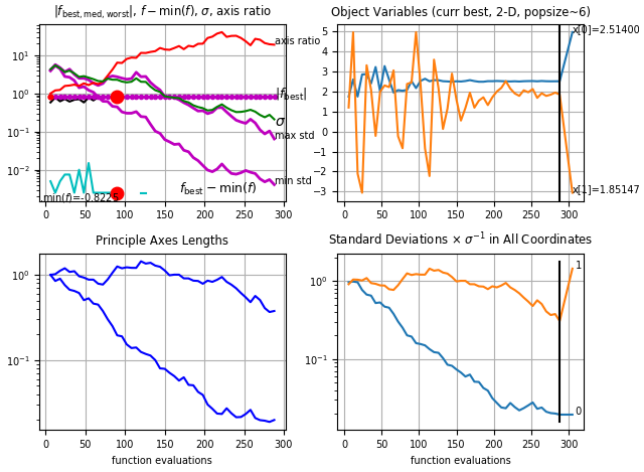

Figure 6 – CMA-ES plot