Lu Lin, Olivier Coudray

18 février 2018

DATA CAMP LAB6

Hyperparameters tuning on SVM models

Résumé

On travaille en binôme pour réaliser ce Lab6 en remplaçant le problème de *Digits* par *MNIST* et le problème de *Classification de chien/chat* par *Speech recognition*. Ce dernier étant notre projet final, on décide donc d'appliquer ces techniques de *Hyperparameters tuning* dans notre projet. On ne va pas soumettre les codes sur Moodle pour le moment, sachant que sans les données, il n'est pas possible de les exécuter. Toutefois, si vous les souhaitez, on pourrait vous les envoyer après.

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.

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

Description des données

MNIST

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

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 mots 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.

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

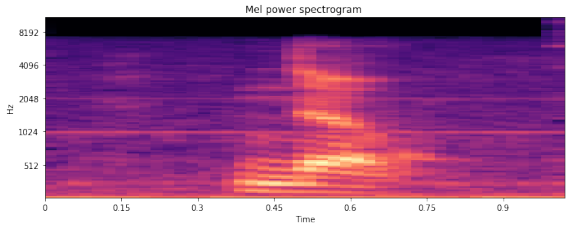
Dans la section 5, nous allons analyser la structure des données en utilisant cette remarque.

Speech recognition

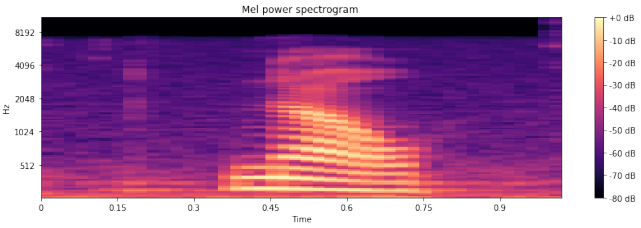
Le pré-traitement des enregistrements audio s'est fait en deux étapes. Dans un premier temps, il s'agit de recouper les extraits audio de façon à ne sélectionner que la partie où le mot est prononcé. Plus concrètement, on passe un filtre sur le signal pour couper tous les sons en deçà d'un certain seuil 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 mieux adaptée aux oreilles humaines) : *Mel-frequency cepstrum.*

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

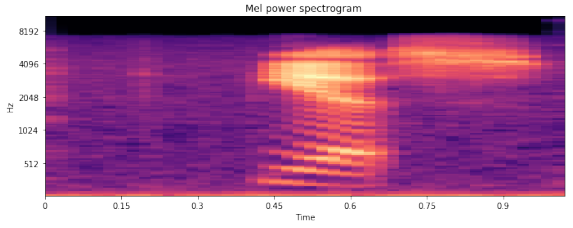
Notons que l'on peut considérer ou du moins visualiser ces spectrogrammes comme des images (matrices). A titre d'exemple, on peut observer ci-dessous, les spectrogrammes Mel pour deux mots différents.



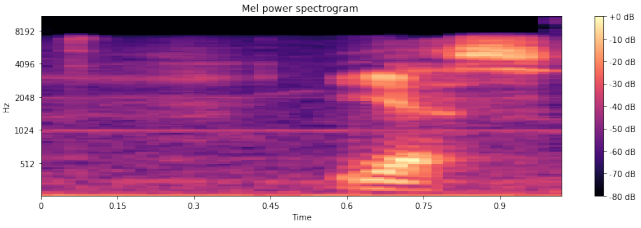
(a) Mel-Spectrogramme de "No" 1



(b) Mel-Spectrogramme de "No" 2



(a) Mel-Spectrogramme de "Yes" 1



(b) Mel-Spectrogramme de "Yes" 2

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 la même sensibilité (échelle), ni le même 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 qu'étant contraint par l'exponentiel du noyau gaussien, le résultat est très 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 (3) 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 adaptés 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 plate (constante) dans cette zone et 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 choisie empêche l'évaluation de dehors de $[-5, 5]^D$. Tout cela entraîne l'échec de DFO-TR.

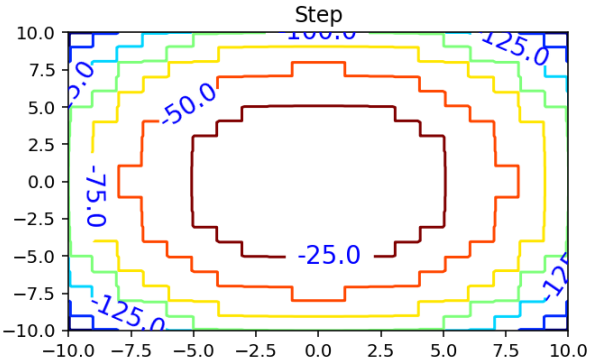


Figure 3 – Step function inverse

Handle the boundary

Dans notre problème, il y a deux contraintes, il faut que les deux hyperparamètres γ, C soient positifs. Il existe plusieurs façons de manipuler ces contraintes. Par exemple, on pourrait faire une transformation non linéaire $f : (x, y) \mapsto (x^2, y^2)$ de l'espace de recherche à l'espace des paramètres $[-5, 5]^2 \mapsto (\gamma, C)$ i.e. $(\gamma, C) = f(x, y)$. Il est aussi possible d'ajouter une pénalisation forte pour les points en dehors de la frontière. Cependant, la pénalisation risque de perturber la performance de la plupart des solvers. Notamment le *surrogate model* de DFO-TR, CMA-ES et BO repose sur les valeurs d'évaluations. Ici, on utilise seulement une transformation affine entre l'espace de recherche et l'espace des paramètres. Puis, selon les solvers, on fait des modifications nécessaires. Par exemple, dans le code de DFO-TR, on ajoute une fonction `constraint_shift` pour forcer les évaluations dans l'espace $[-5, 5]^2$. Pour CMA-ES, on utilise directement `options=`$\{bounds : [-5, 5]^2\}$. Enfin, le BO de scikit-learn prend les frontières comme l'argument.

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 supérieures à 98%. Le tableau suivant illustre les minima locaux trouvés :

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.

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 dispose de 500 données pour chaque classes, donc 5500 données au total. Pour la calibration des hyperparamètres, on n'utilise que 2000 données (tirées aléatoirement). Puis, on sépare le jeu de données en deux datasets : un d'entraînement et un de test avec la répartition $80\% - 20\%$. Ensuite, pour pouvoir comparer ces optimiseurs, on lance chaque algorithme trois fois (i.e. `restart`). Enfin, on faire le test sur l'ensemble des 5500 données avec la répartition $80\% - 20\%$ et on ne garde que les meilleurs paramètres. Comme le test set n'est pas fixé à la dernière étape, la précision fluctue.

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.

Remarque Bien que DFO-TR soit déterministe, le fait qu'on ait mélangé (shuffle) les données à chaque `restart`, les résultats finaux sont différents.

Optimizer	γ	C	Score	time (sec)	n eval
DFO-TR	15.0	265.0	$84.43 \pm 0.48\%$	≈ 600	≈ 40
CMA-ES	15.02	359.16	$83.93 \pm 0.76\%$	≈ 4000	≈ 300
Random Search	16.64	408.87	$84.39 \pm 0.64\%$	≈ 1200	100
BO	15.84	600	$84.14 \pm 0.70\%$	≈ 1500	100
Default	0.000372	1.0	$7.45 \pm 0.60\%$	–	–

Remarquons d'abord que la fonction objectif est non-bruitée pendant la calibration des hyperparamètres, mais elle est bruitée pour l'évaluation finale à cause du `shuffle`. Grâce à la calibration des hyperparamètres, on gagne énormément en précision. [5] En fait, d'après le leaderboard Kaggle, on est déjà proche du résultat state-of-the-art (avec la précision 91.06%)[**?**]. On observe également qu'il existe plusieurs optima locaux qui sont tout à fait comparables à ceux trouvés ci-dessus. Notamment CMA-ES a trouvé $(\gamma, C) = (3.88, 669.58)$, et $(\gamma, C) =$

5. La valeur de défaut de γ est définie par $\frac{1}{\texttt{n\_features}}$ avec `n_features` le nombre de caractéristiques. En l'occurrence, on a $128 * 21 = 2688$ pour chaque spectrogramme.

$(7.0, 551.1)$ avec la `popsize` $= 6$ par défaut. En fait, le résultat ci-dessus est obtenu avec `popsize` $= 15$.

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 (SVM avec noyau gaussien) est séparable (Figure (4) (5)), ce qui est très étonnant.

Faisons une remarque sur le lien entre l'échelle des paramètres γ et C et la structure des données (i.e. la difficulté du problème). On sait que plus γ et C sont élevés, plus que le modèle est complexe, car $\frac{1}{\gamma}$ correspond au *rayon* du noyau gaussien et C (complexity) contrôle le nombre de *support vector*. Pour le problème **MNIST** le(s) meilleur(s) paramètres est(sont) autour de $(0.024, 100)$, alors que pour **Speech recognition**, on trouve $(15.0, 400)$. Peut-on donc en conclure que le Speech recognition est beaucoup plus difficile que MNIST ? En fait, pour une comparaison plus juste, il faut tenir compte la gamme des features, d'après l'équation (1). Pour le problème de Speech recognition, on avait fait une normalisation en norme 2, et l'intervalle des valeurs est $[-0.04, 0]$ (le son en décibel `dB` est négatif). Ainsi, si on le re-normalise à l'ordre $[0, 1]$, il faut multiplier les features par 25, cela revient à diviser γ par $25^2 = 625$. Enfin, $\frac{15}{625} = 0.024$, on retrouve le même ordre de grandeur !

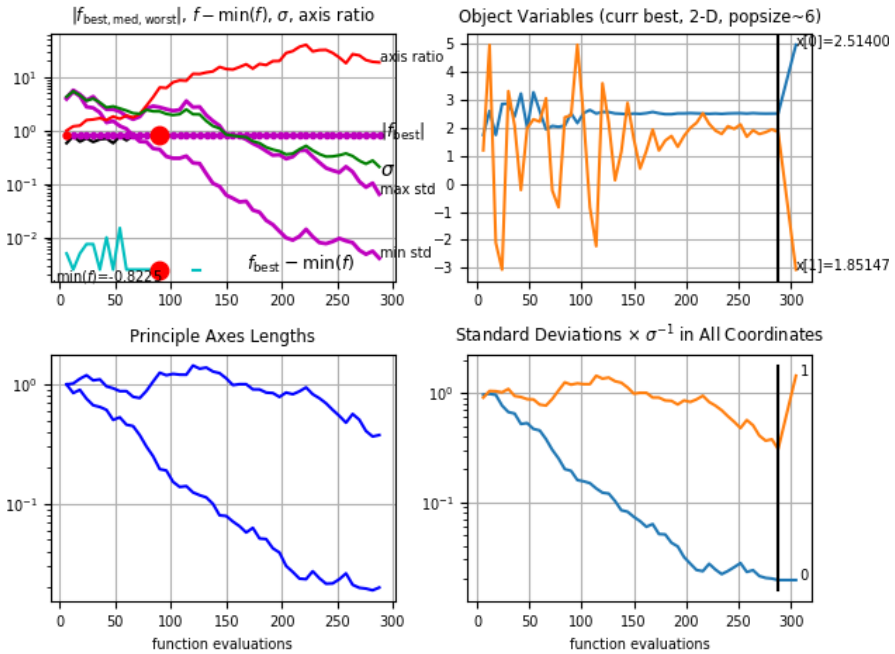


Figure 4 – CMA-ES plot popsize= 6

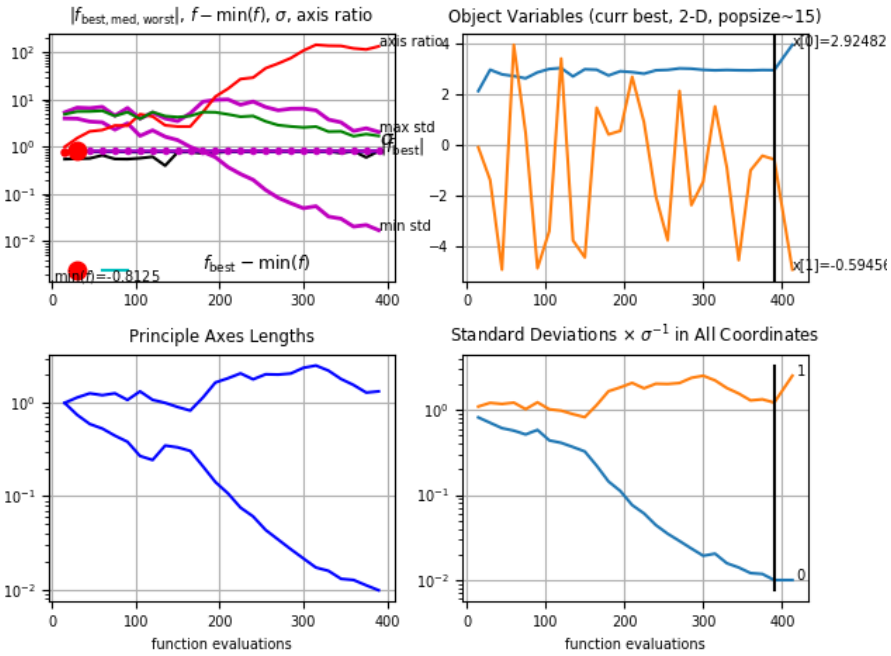


Figure 5 – CMA-ES plot popsize= 15