Projet de Machine Learning
A continuation method for Semi-Supervised SVMs

Marwan Akrouch et Cyril Nederveen

6 juin 2021

Introduction

L'objectif de cette étude est de reproduire l'approche continue de la méthode S^{3}VM introduite par Chapelle, Chi et Zien dans leur artcile "A Continuation Method for Semi-Supervised SVMs". Cette méthode s'applique aux jeux de données comprenant à la fois des données labelisées et des données non labelisées. L'approche continue dans ce cas est justifiée par le fait que dans le problème d'optimisation, la fonction à minimiser n'est pas convexe et possède plusieurs minimums locaux.

Après avoir implémenté la méthode, nous avons décidé d'appliquer cette méthode dans le cas des données MNIST pour évaluer ses performances.

1 Le problème d'optimisation

La méthode SVM plus classique s'applique lorsqu'on a n données $\mathbf{x}_i \in \mathcal{X}$ dont les labels $y_i \in \{-1, 1\}$ sont connus. Pour construire un modèle, on choisit $w \in \mathcal{X}$ et $b \in \mathbb{R}$, on s'intéresse à la fonction de décision f telle que pour $x \in \mathcal{X}$, $f(x) = sign(\mathbf{w}^\top \mathbf{x}_i + b)$. Le but étant alors de sélectionner w et b qui minimise, en un certain sens, l'erreur de prédiction.

Pour cela on fixe la loss telle que $l(t) = max(0, 1 - t)$ dont l'allure est illustrée sur la firgure 1.
Ainsi la fonction L à minimiser sur $\mathbf{w}$ et b est donné par :

$$L(\mathbf{w}, b) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C \sum_{i=1}^{n} l\left(y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right)$$

où C est un hyperparamètre qui permet d'accorder plus ou moins d'importance à la régularité de $\mathbf{w}$, plus C est élevé plus la marge $\frac{1}{||\mathbf{w}||}$ est petite.

À présent nous considéront m données non labelisées $(\mathbf{x}_i)_{n+1 \leq i \leq n+m}$ de l'ensemble $\mathcal{X}$. L'idée de S^{3}VM est de séparer ces données non labelisées en deux catégories qui se distinguent. Pour cela on introduit un terme $l\left(|\mathbf{w}^\top \mathbf{x}_i + b|\right)$ qui pénalise la situation où $\mathbf{w}^\top \mathbf{x}_i + b$ est proche de 0, ie quand le point x_i est dans la marge (cf figure 1). En pratique, on utilise la fonction $\tilde{l}(t) = exp(-st^2)$, avec s=3,

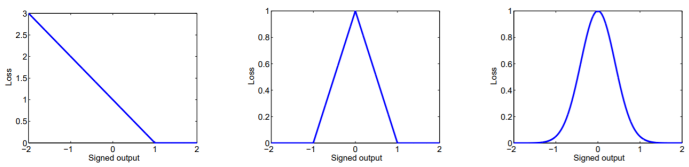

FIGURE 1 – courbes des losses mentionnés : gauche : $t \to max(0, 1 - t)$ - milieu : $t \to max(0, 1 - |t|)$ - droite : $t \to exp(-3t^2)$

pour avoir différentiabilité (cf figure 1). Ainsi la fonction L à minimiser sur et b est modifiée de la manière suivante :

$$L(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + C\sum_{i=1}^{n} l\left(y_i(\mathbf{w}^\top \mathbf{x}_i + b)\right) + C^* \sum_{i=n+1}^{n+m} l\left(|\mathbf{w}^\top \mathbf{x}_i + b|\right)$$

Afin de ne pas se retrouver dans la situation où les données non labelisées sont "rangées" que d'un seul côté i.e $\mathbf{w}^\top \mathbf{x}_i + b > 0$ pour tout $i \in [\![n+1, m]\!]$ ou $\mathbf{w}^\top \mathbf{x}_i + b < 0$ pour tout $i \in [\![n+1, m]\!]$, on décide d'ajouter la contrainte que le ratio de classification des données non labelisées soit à peu près égal à celui des données labelisées. Dans ce but, on peut imposer en plus dans le problème d'optimisation la contrainte suivante :

$$\frac{1}{m}\sum_{i=n+1}^{m+n} \mathbf{w}^\top \mathbf{x}_i + b = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Alors, en centrant les données non labelisées et en fixant b à $\frac{1}{n}\sum_{i=1}^{n} y_i$, on résout deux difficultés d'un coup car d'une part, il n'est plus nécessaire d'optimiser sous contraintes et d'autre part il ne reste plus que $\mathbf{w}$ comme variable.

2 Relaxation du problème

Dans le cas du SVM standard, une solution pour le problème d'optimisation est de plutôt résoudre le problème dual. En revanche, cette approche ne s'applique dans le cas du cS^{3}VMs par absence de convexité (et trés probablement absence de dualité forte) en raison de la présence du terme $\sum_{i=n+1}^{n+m} l\left(|\mathbf{w}^T \mathbf{x}_i + b|\right)$ dans l'expression de L. De plus, il existerait un certain nombre de minimums locaux. Pour y remédier, Chapelle, Chi et Zien ont considéré une approche qui consiste à approximer la fonction L par des fonctions L_γ plus régulières et convexes dépendant continuement du paramètre γ, et tel que $L_0 = L$. En choisissant une séquence $\gamma_0 > \cdots > \gamma_{end}$, à l'étape k, une fois qu'on a résolu le problème d'optimisation avec la fonciton $L_{\gamma_{k-1}}$ le paramètre optimal étant $\mathbf{w}_{k-1}$, on part de cette solution pour trouver le nouveau paramètre optimal $\mathbf{w}_k$ du problème avec la fonction L_{γ_k}.

Chapelle, Chi et Zien ont retenu pour L_γ la convolée de L par une gaussienne de variance $\gamma/2$. Cela donne en pratique :

$$L_\gamma(\mathbf{w}) = \frac{1}{\sqrt{\pi\gamma}^d}\int_{\mathbb{R}^d} L(\mathbf{w} - \mathbf{t})e^{-\|t\|^2/\gamma} d\mathbf{t}$$

L'algorithme est résumé ci-dessous.

Algorithm 1: Continuation method

Input: Fonction $L : \mathbb{R}^d \mapsto \mathbb{R}$, point initial $\mathbf{w}_0 \in \mathbb{R}^d$ et une séquence $\gamma_0 > \cdots > \gamma_{end}$
Soit $L_\gamma(\mathbf{w}) = (\pi\gamma)^{-d/2} \int L(\mathbf{w} - \mathbf{t}) exp(-\|t\|^2/\gamma) d\mathbf{t}$;
for $k \leftarrow 0$ ***to*** *end* **do**
  en partant de $\mathbf{w}_k$, trouver le minimiseur $\mathbf{w}_{k+1}$ de $L_{\gamma_{k+1}}$;
end

La raison pour laquelle Chapelle, Chi et Zien ont opté pour la convolée gaussienne de L réside dans le fait qu'il est possible d'établir une expression plus simple de L_γ. En effet, après calculs :

$$L_\gamma(\mathbf{w}) = \frac{1}{2}\mathbf{w}^\top \mathbf{w} + \frac{1}{2}\gamma d + C\sum_{i=1}^{n} \frac{\gamma\|\mathbf{x}_i\|}{\sqrt{2}}\left[\frac{\exp(-e_i^2)}{\sqrt{\pi}} - e_i \mathrm{erfc}(e_i)\right] + C^* \sum_{i=n+1}^{n+m} \frac{1}{\sqrt{a_i}} \exp\left(\frac{-s(\mathbf{w}^\top \mathbf{x}_i + b)^2}{a_i}\right)$$

Avec $a_i = 1 + 2\gamma s\|\mathbf{x}_i\|^2$ et $e_i = \frac{y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1}{\sqrt{2\gamma}\|\mathbf{x}_i\|}$. erfc est la complémentaire de la fonction d'erreur, définie par $\mathrm{erfc}(x) = 2/\sqrt{\pi}\int_x^\infty e^{-t^2} dt$.

D'autre part, le gradient de L_γ en $\mathbf{w}$ s'écrit :

$$\nabla L_\gamma(\mathbf{w}) = \mathbf{w}^\top + \frac{C}{2}\sum_{i=1}^{n} \operatorname{erfc}(e_i) y_i \mathbf{x}_i^\top - C^* \sum_{i=n+1}^{n+m} \frac{2s(\mathbf{w}^\top \mathbf{x}_i + b)}{a_i^{3/2}} \exp\left(\frac{-s(\mathbf{w}^\top \mathbf{x}_i + b)^2}{a_i}\right) \mathbf{x}_i^\top$$

Si l'allure de ces expressions semblent complexes, elles permettent néanmoins de réaliser une descente de gradient pour résoudre le primal.

Pour notre implémentation sous Python, nous avons repris le choix de Chapelle, Chi et Zien pour la séquence $\gamma_0 > \cdots > \gamma_{end}$ i.e :

$$\gamma_0 = \frac{(C^* \lambda_{max})^{2/3}}{(2s)^{1/3}} \quad \text{et} \quad \gamma_{end} = \frac{1}{10} \frac{1}{2s \max_i \|x_i\|^2}$$

où λ_{max} est la plus grande valeur propre de la matrice $\sum_{i=n+1}^{n+m} \frac{\mathbf{x}_i^\top \mathbf{x}_i}{\|\mathbf{x}_i\|^3}$. La séquence est de taille 10 et la décroissance est exponentielle :

$$\gamma_i = \left(\frac{\gamma_{end}}{\gamma_0}\right)^{\frac{i}{10}} \gamma_0, \quad 0 \leq i \leq 10.$$

Résultats

Classification à deux classes :

Après avoir implémenté le modèle, on a utilisé la dataset MNIST pour le tester.
Dans un premier temps on fait de la classification binaire en se basant sur les données de deux classes, par exemple les chiffres 0 et 3. On a pris 25% de données qui sont labelisé pour l'entrainement, ce qui fait 47 images labelisées et 143 non labelisées. on a obtenu les résulats illustrés par la figure 2. On a fait une cross validation pour estimer le meilleur choix de C et C^*, on a obtenu les résultats en figure 3. On a finalement choisi $C = C^* = 100$ comme dans le papier.

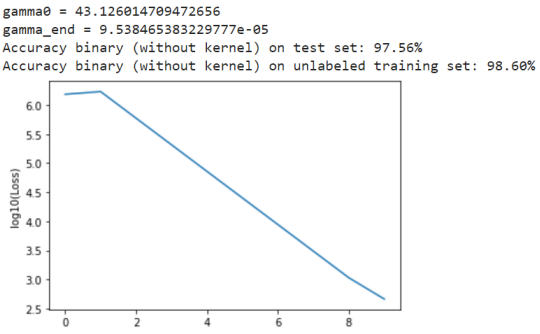

FIGURE 2 – résultats de classification pour les images ayant les labels 0 et 3, 25% des données sont labelisé. le plot représente la variation du log10 du convolved loss calculé après chaque étape de diminution de gamma.

Notons que l'intérêt de calculer l'erreur des prédictions sur les données non labelisées du training set est de chercher un modèle qui performe le mieux possible sur notre jeu de données sans se préoccuper de sa capacité à généraliser, c'est ce qu'on appelle du **transductive learning**.

On étudie maintenant la qualité du modèle selon le pourcentage des données labelisées. On fait varier ce dernier entre 1% et 40% et on calcule l'erreur sur le test set, les résultats sont illustrés dans la figure 4 : on remarque une tendance décroissante. Théoriquement, à mesure que nous augmentons le nombre d'étiquettes, nous obtenons une meilleure précision, ce qui est illustré empiriquement ci-dessous. Cependant, il faut tenir compte de la qualité des échantillons étiquetés (diversité et capacité à tirer des conclusions de ces échantillons).

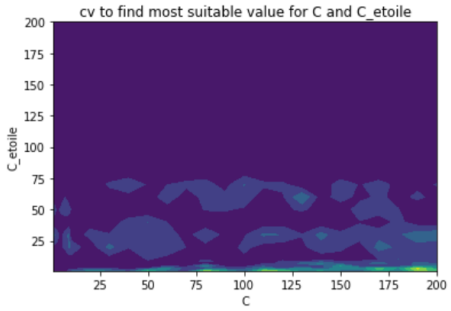

FIGURE 3 – Cross Validation pour estimer le meilleur paramètre C et C^* du modèle, dans le cadre d'une classification à deux classes (labels 0 et 3). Les zones foncées représentent une erreur faible.

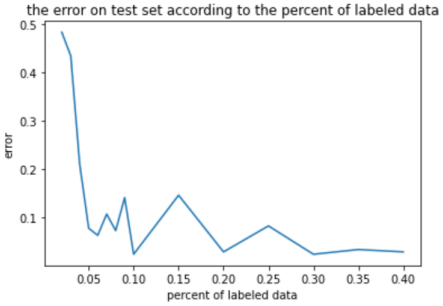

FIGURE 4 – Variation de l'erreur sur les données de test par rapport au poucentage des données labelisées dans le cadre d'une classification à deux classes (labels 0 et 3) sans kernel-PCA.

Classification multiclasses :

L'approche qu'on a adoptée pour une classification multiclasses est One Vs All : On entraine $10(10-1)/2 = 45$ classifieurs à deux classes pour chaque pair de labels, et pour prédire la classe d'une image on choisi le label ayant le plus grand score. Dans le cas des SVM, le score est la distance à l'hyperplan, i.e la valeur absolue de la fonction de décision $|w^T x + b|$.

Dans le cas de 25% des données labelisées, on a obtenue une accuracy sur le test set de 54.10%, alors que pour les prédictions sur les données non labelisées du training set on a une accuracy de 62%. On a fait varier le poucentage des données labelisée entre 1% et 40%, et on a obtenues les résultas dans la figure 5.

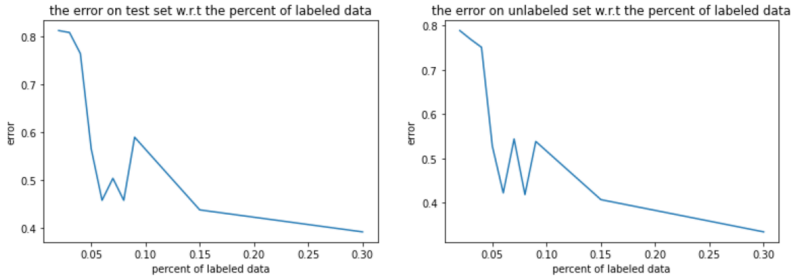

FIGURE 5 – Variation de l'erreur sur les données de test par rapport au poucentage des données labelisées dans le cadre d'une classification multiclasses One vs Rest (sans k-PCA).

.

On a également essayé une approche One vs One, qui est plus lente qu'une One vs Rest et en plus la précision du modèle n'a pas été améliorée.

3 Kernel trick et Analyse en Composantes Principales à noyeau

La méthode ci-dessus peut se généraliser au cas non linéaire en y appliquant le "kernel trick". Cela consiste à projeter les $\mathbf{x}_i$ dans un espace $\mathcal{H}$ de plus haute dimension que $\mathcal{X}$ puis de séparer les données dans cet espace plus descriptif des données.

On peut par ailleurs combiner à cette méthode, l'analyse en composante principale afin de réduire la complexité de l'algorithme présenté dans la section ci-dessus. En effet, la complexité annoncée par Chapelle, Chi et Zien s'élève à $O((n+m)^3)$, qui s'avére problématique pour des jeux de données à grande taille. L'ACP, qui réduit à maximum $q = n + m$ le nombre de features, permet de réduire considérablement la complexité.

En pratique, la projection des données dans $\mathcal{H}$ s'effectue avant l'ACP. Cela donne alors une transformation des données :

$$\psi(\mathbf{x}) = A^\top k(\mathbf{x}, \mathbf{x}_{tot})$$

où $k(\mathbf{x}, \mathbf{x}_{tot}) = (k(\mathbf{x}, \mathbf{x}_i))_{1 \leq i \leq n+m}$, A est la matrice obtenue en diagonalisant $K = (k(\mathbf{x}_i, \mathbf{x}_j))_{1 \leq i,j \leq n+m}$ sous la forme $K = U\Lambda U^\top$, puis en prennant $A = U\Lambda^{1/2}$.

On peut réduire la complexité encore plus en prennant $\mathbf{x}_{lab}$ au lieu de $\mathbf{x}_{tot}$ pour l'estimation de la projection ψ, dans ce cas on se limite aux n premières composantes principales :

$$\psi(\mathbf{x}) = \tilde{A}^\top k(\mathbf{x}, \mathbf{x}_{lab})$$

avec $k(\mathbf{x}, \mathbf{x}_{lab}) = (k(\mathbf{x}, \mathbf{x}_i))_{1 \leq i \leq n}$, et $\tilde{A}$ la matrice qui comporte les n premières lignes de la matrice A.

Résultats

La classification à deux classes (labels 0 et 3) avec 25% de données labelisées en utilisant un kernel RBF de paramètre $\sigma = 4$ donne les résultats en figures 6 et 7. Dans la figure 6, on a pris les $n + m$ premières composantes principales, càd qu'on considérer tout les données de training $\mathbf{x}_{tot}$ pour estimer la projection par ψ, $\psi(\mathbf{x}) = A^\top k(\mathbf{x}, \mathbf{x}_{tot})$. Puis, la figure 7 représente le cas où on réduit encore plus la dimension en considérant uniquement les données labelisées $\mathbf{x}_{lab}$ pour estimer la projection : $\psi(\mathbf{x}) = \tilde{A}^\top k(\mathbf{x}, \mathbf{x}_{lab})$ avec $k(\mathbf{x}, \mathbf{x}_{lab}) = (k(\mathbf{x}, \mathbf{x}_i))_{1 \leq i \leq n}$ et $\tilde{A}$ la matrice contenant les n premières lignes de A.

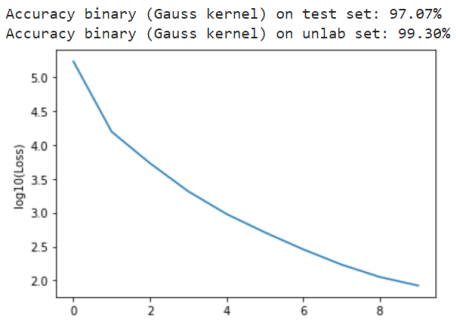

FIGURE 6 – résultats de classification pour les images ayant les labels 0 et 3 en utilisant un kernel RBF de paramètre $\sigma = 4$ **en considérant tout $\mathbf{x}_{tot}$ pour estimer la projection $\psi(\mathbf{x})$(on prend n+m composantes principales)**, on a pris 25% des données qui sont labelisées. le plot représente la variation du log10 du convolved loss calculé après chaque étape de diminution de gamma.

On remarque que la réduction supplémentaire de dimension à $n = 25\% \times (n+m)$ diminue légèrement la précision sur le test set par rapport au cas où on prend tout $\mathbf{x}_{tot}$ pour la projection k-PCA, mais on gagne en terme de complexité et de rapidité du modèle.

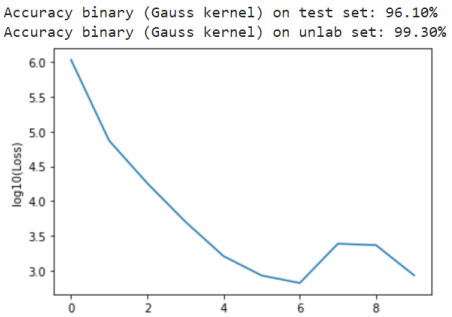

FIGURE 7 – résultats de classification pour les images ayant les labels 0 et 3 en utilisant un kernel RBF de paramètre $\sigma = 4$ **en considérant $\mathbf{x}_{lab}$ uniquement pour estimer la projection $\psi(\mathbf{x})$(on prend n composantes principales)**, on a pris 25% des données qui sont labelisées. le plot représente la variation du log10 du convolved loss calculé après chaque étape de diminution de gamma.

Pour la classification multiclasses, on a adopté l'approche One Vs All comme décrit précédemment. Avec un kernel RBF, la précision **sur le test set** est améliorée. Pour 25% des données qui sont labelisées et un kernel RBF de paramètre $\sigma = 3$, en considèrant uniquement les données labelisées pour la projection k-PCA, on a une accuracy de **68.7%**.

La valeur du paramètre σ du RBF a été choisie à travers une cross validation ; notons qu'on a fait la réduction supplémentaire de dimension en k-PCA (utiliser $\mathbf{x}_{lab}$ pour projeter) afin de dimunier la complexité. On a obtenue les résultats en figure 8 : il est préférable de choisir σ entre 2 et 4. Les σ faibles (entre 0 et 1) donnent de mauvaises résultats.

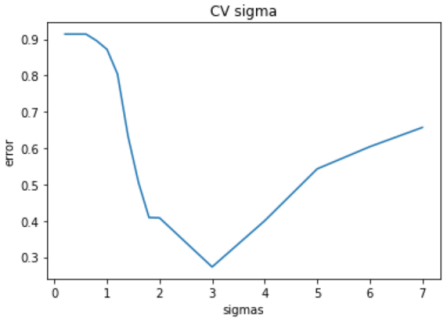

FIGURE 8 – Cross validation pour estimer le meilleur paramètre σ du RBF pour une classifiaction multiclasses one vs rest.On a pris 25% des données qui sont labelisées.

4 Conclusion

Les résultats obtenues en classification à deux classes sont très satisfaisants, et même pour des pourcentages de données labelisées faibles, soit avec la version linéaire soit avec un k-PCA : pour 25% de données labelisées, on a eu une précision $\approx 98\%$ sur le test set et $\approx 99\%$ sur les données non labelisées du training set. La performance du modèle sur l'ensemble des données non labelisées possède un intérêt particulier dans le cadre du transductive learning.

D'autre part, la classification multiclasses avec une approche One vs Rest donne de bons résultats aussi, et le k-PCA avec un kernel RBF améliore l'accuracy du modèle tout en dimuniant sa complexité : précision sur le test set $\approx 54\%$ dans le cas linéaire et $\approx 68\%$ en appliquant un k-PCA avec un kernel gaussien.