Introduction to Statistical Supervised Machine Learning

Examen 2022 - 2023

Master 1 MIASHS

Guillaume Metzler

Institut de Communication (ICOM)
Université de Lyon, Université Lumière Lyon 2
Laboratoire ERIC UR 3083, Lyon, France

guillaume.metzler@univ-lyon2.fr

Durée : 1h30

Les documents personnels, notes de cours, téléphones et ordinateurs ne sont pas autorisés pour cet examen.

Abstract

Les exercices sont tous indépendants et peuvent être traités dans l'ordre qui vous conviendra. On prendra cependant soin de bien indiquer les numéros des questions traitées ainsi que les exercices correspondants. Pour certaines questions, vous pouvez illustrer vos propos à l'aide de graphiques, dessins ou autres tableaux si cela vous semble pertinent.

Autour de l'apprentissage

Soit $S = \{\mathbf{x}_i, y_i\}_{i=1}^m$ un échantillon d'apprentissage, ℓ une loss et $\mathbf{w}$ les paramètres d'un modèle. On considère un algorithme de classification quelconque qui cherche à résoudre le problème d'optimisation suivant :

$$\min_{\mathbf{w}} \frac{1}{m} \sum_{i=1}^{m} \ell(h_{\mathbf{w}}(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w}\|, \tag{1}$$

où $h_{\mathbf{w}}$ représente une hypothèse (un classifieur ou un régresseur) qui dépend du paramètre $\mathbf{w}$.

1. Dans le problème d'optimisation (1), identifier le terme que l'on appelle *risque empirique* et le *terme de régularisation*.

   Le **risque empirique** est le terme défini par $\frac{1}{m} \sum_{i=1}^{m} \ell(h_{\mathbf{w}}(\mathbf{x}_i), y_i)$ et le terme de régularisation est le terme $\lambda \|\mathbf{w}\|$.

2. Quel est le nom de λ et quel est son rôle ?

   λ est un hyper-paramètre de notre modèle et il permet de trouver un compromis entre la complexité du modèle et ses performances sur le jeux de données d'entraînement.
   Une faible valeur de λ signifie que le modèle sera peu régularisé et que ce dernier se concentrera sur la minimisation du risque et sera potentiellement plus sujet au sur-apprentissage. A l'inverse une forte régularisation, *i.e.*, une valeur élevée de λ, entraîne l'apprentissage de modèles trop simples et donc peu performants.

3. On se concentre maintenant sur le terme $\|\mathbf{w}\|$

   (a) Quel est l'impact, sur le modèle appris, en choisissant $\|\mathbf{w}\| = \|\mathbf{w}\|_1$?

   Il s'agit d'un terme de pénalité ℓ_1 connu sous le nom de *lasso*. Cette pénalité est souvent employée lorsque l'on travaille en grande dimension, *i.e.*, avec beaucoup de variables. Cette pénalité va permettre d'effectuer de la *sélection de variables* pour apprendre des modèles parcimonieux, reposant sur les variables les plus importantes.

   (b) Quel est l'impact, sur le modèle appris, en choisissant $\|\mathbf{w}\| = \|\mathbf{w}\|_2$?

Ce terme *ridge*, va aussi faire en sorte que certains paramètres s'annulent, mais en premier lieu, il va d'abord éviter que certains paramètres ne prennent des valeurs disproportionnées par rapport aux paramètres du modèle.

4. Rappeler quel est l'objectif général en Machine Learning (notamment vis à vis du risque empirique et du risque en généralisation).

   De façon générale, nous cherchons un modèle qui soit performant sur les données, *i.e.* qui minimise notre risque empirique, avec un biais faible, mais qui soit aussi performant sur des nouvelles données qui suivent la même distribution que les données d'entraînement (c'est notre fameux risque en généralisation). C'est notamment via ce terme de régularisation que nous serons en mesure de contrôler cet équilibre entre biais et variance du modèle, *i.e.* de garantir une borne performance sur l'ensemble d'entraînement mais aussi sur les données tests.

5. Décrire précisément la procédure que vous devriez mettre en place pour résoudre le problème (1) permettant de répondre à la question précédente.

   Pour résoudre ce problème là, nous supposons que l'on dispose d'un échantillon S et on se donne des valeurs $\lambda_1, \lambda_2, \ldots, \lambda_p$ de l'hyper-paramètre.

   On commence par séparer notre échantillon S en deux ensembles disjoints S_{train} et S_{test}. Une k- fold cross validation est effectuée à l'aide de l'ensemble S_{train} afin de trouver la meilleure valeur de l'hyper-paramètre λ parmis les valeurs prédéfénies.

   Une fois que cette valeur est trouvée, l'ensemble des données d'entraînement sont utiliser pour réapprendre un modèle qui sera ensuite évaluée sur les données tests.

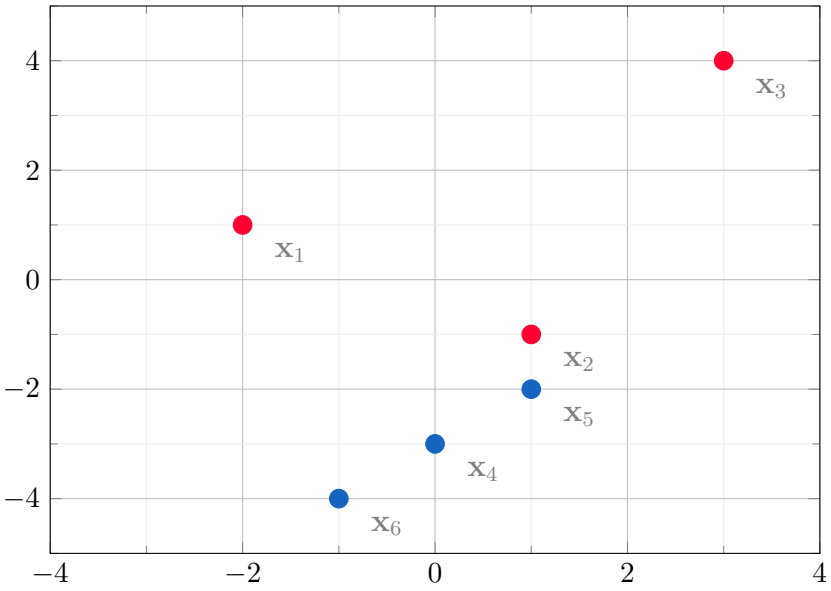
Algorithmes

Cet exercice se concentre sur l'étude de certains algorithmes, on va considérer que l'on dispose du jeu d'entraînement S suivant

Individu	x_1	x_2	y
1	-2	1	1
2	1	-1	1
3	3	4	1
4	0	-3	-1
5	1	-2	-1
6	-1	-4	-1

Algorithme du plus proche voisin

1. Représenter les individus $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_6$ sur un dessin.



2. Expliquer le fonctionnement de l'algorithme du plus proche voisin (*i.e.* k-NN lorsque $k = 1$)

   L'algorithme consiste à attribuer à une nouvelle donnée, l'étiquette de son plus proche dans l'ensemble d'entraînement.

   Pour cela, nous commençons par calculer les distances entre le nouvel exemple et l'ensemble des données d'entraînement, on les ordonnes ensuite par ordre croissant et on conserve uniquement l'exemple le plus proche pour la classification.

3. On considère les points $\mathbf{z}_1 = (1, 1), \mathbf{z}_2 = (0, 3)$ et $\mathbf{z}_3 = (-1, 1)$. Déterminer leur étiquette à l'aide de l'algorithme du plus proche voisin.

   Il faut pour cela commencer par calculer la distance des différents points $\mathbf{z}_i$ à l'ensemble des points d'entraînement. La table ci-dessous donne la distance euclidienne au carrée entre les points $\mathbf{z}_i$ et $\mathbf{x}_j$. La distance notée en gras correspond à la distance la plus petite pour un $\mathbf{z}_i$ donné.

	$\mathbf{z}_1 = (1, 1)$	$\mathbf{z}_2 = (0, 3)$	$\mathbf{z}_3 = (-1, 1)$
$\mathbf{x}_1 = (-2, 1)$	9	**8**	**1**
$\mathbf{x}_2 = (1, -1)$	**4**	17	8
$\mathbf{x}_3 = (3, 4)$	13	10	25
$\mathbf{x}_4 = (0, -3)$	17	36	17
$\mathbf{x}_5 = (1, -2)$	9	26	13
$\mathbf{x}_6 = (-1, -4)$	29	50	25
Prédiction	$+1$	$+1$	$+1$

4. Sur le plan de l'apprentissage, quel est le spécificité de cet algorithme ?

   Cet algorithme est non paramétrique, il n'y a donc pas réellement de phase d'apprentissage et tous les exemples d'entraînement sont nécessaires pour prédire l'étiquette d'une nouvelle donnée.

5. Quel est le principal inconvénient de cet algorithme, notamment lorsque la taille de l'échantillon d'apprentissage est grande ?

   L'inconvénient de cet algorithme est qu'il est nécessaire de garder tous les exemples d'ensemble d'entraînement en mémoire pour la prédiction de l'étiquette d'une nouvelle donnée, ce qui est un complexité d'ordre $\mathcal{O}(md)$. En outre, il faudra ensuite calculer, pour chaque exemple test, sa distance à l'ensemble des exemples d'entraînement puis les ordonnées. Cela entraîne donc des calculs qui peuvent être coûteux mais nécessite aussi un espace de stockage suffisamment important.

Algorithme du SVM

On considère un jeu d'entraînement S' qui a conduit à l'obtention du SVM linéaire représenté en Figure 1 et dont les paramètres sont approximativement les suivants :

$$\mathbf{w} = (-1.5, -0.5) \quad \text{et} \quad b = -1.$$

1. Rappeler la règle de classification d'un SVM linéaire.

   Pour étiqueter un nouvel exemple, on regardera de quel côté de l'hyerplan ce dernier se trouve. Plus précisément si on note $(\mathbf{w}, b)$ les paramètres de l'hyperplan séparateurs alors

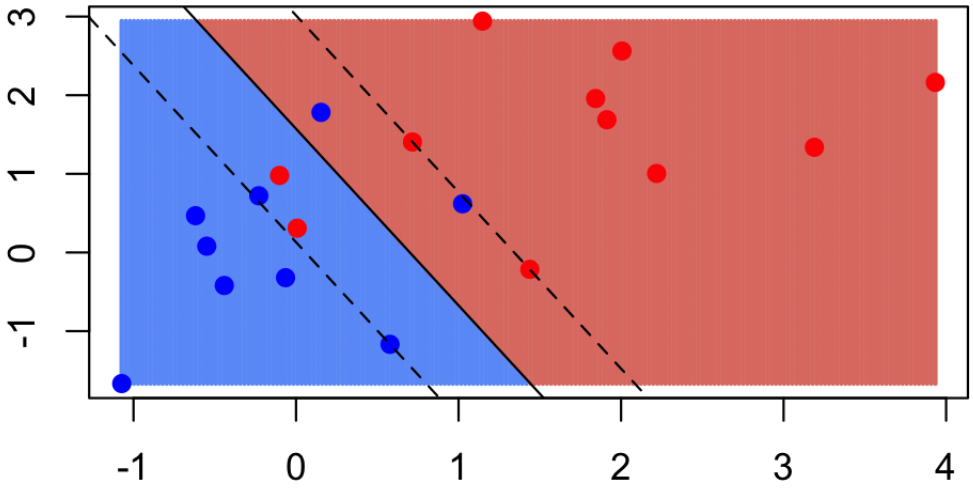


Figure 1: Classifieur SVM linéaire. La zone bleue représente la zone de prédiction négative, *i.e.* $y = -1$ et la zone rouge représente la zone de prédiction positive, *i.e.* $y = +1$.

$$h(\mathbf{x}) = \begin{cases} 1 & \text{si } \langle \mathbf{w}, \mathbf{x} \rangle + b > 0, \\ -1 & \text{si } \langle \mathbf{w}, \mathbf{x} \rangle + b < 0. \end{cases}$$

2. Donner la définition de la *hinge loss* et énoncer le problème d'optimisation à résoudre pour un SVM linéaire.

   La hinge loss est définie par

   $$\ell(h(\mathbf{x}), y) = [1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)]_+ = \max\left(0, 1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)\right).$$

   Le problème d'optimisation que l'on cherche alors à résoudre est alors, en posant $\mathbf{w} = (\mathbf{w}, b)$

   $$\min_{\mathbf{w} \in \mathbb{R}^{d+1}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{m} \sum_{i=1}^{m} [1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b)]_+ ,$$

   où $C > 0$ est un hyper-paramètre du modèle.

3. Sur la Figure 1, identifier :

(a) l'hyperplan séparateur

Il s'agit de la droite en trait plein.

(b) les marges du SVM

Ces dernières sont représentées en trait pointillé sur le graphique.

(c) les vecteurs (ou points) supports

Les point supports sont l'ensemble des points rouges et bleus qui se trouvent du mauvais côté de l'hyperplan ou à l'intérieur de la marge du SVM.

4. Prédire l'étiquette des individus $\mathbf{x}_1, \mathbf{x}_2$ et $\mathbf{x}_6$ à l'aide du SVM (on demande de le faire par un calcul et non graphiquement).

   On rappelle que l'on a $\mathbf{w} = (-1.5, -0.5)$ et $b = -1$

	$\mathbf{x}_1 = (-2, 1)$	$\mathbf{x}_2 = (1, -1)$	$\mathbf{x}_6 = (-1, -4)$
$\langle \mathbf{w}, \mathbf{x} \rangle + b$	1.5	-2	2.5
$h(\mathbf{x})$	$+1$	-1	$+1$

5. Quelle est la valeur de la loss pour le point $\mathbf{x}'$ de coordonnées $(1, -1)$ qui est un point dont le label est négatif, *i.e.* $y = -1$.

   Nous avons $\langle \mathbf{w}, \mathbf{x} \rangle + b = -2$, donc $1 - y(\langle \mathbf{w}, \mathbf{x} \rangle + b) = -1$, la loss est donc nulle pour ce point. Il est correctement classé et se trouve en dehors de la marge.

Régression logistique

1. Rappeler ce qu'est la régression logistique. On expliquera le principe et on rappellera la règle de classification.

   La régression logistique est un *modèle linéaire généralisé*, dont le but n'est pas d'effectuer une tâche de *régression* mais une tâche de *classification*, *i.e.*, on cherche à prédire une classe et non un nombre réel. Pour être plus précis, ce modèle cherchera à estimer la probabilité d'appartenance à une classe.

   Les paramètres du modèle sont estimés par maximum de vraisemblance.

Les exemples sont affectés à la classe pour laquelle la probabilité d'appartenance est la plus élevée.

2. Quelle est la loss que l'on cherche à optimiser dans la phase d'apprentissage. On rappellera la définition de cette loss.

   La loss que l'on cherche à optimiser dans le cadre de la régression logistique est la *loss logistic*. Cette dernière est définie par la log-vraisemblance de notre modèle

$$\ell(\boldsymbol{\theta}, S) = -\sum_{i=1}^{n} \left[y_i \ln \left(\frac{1}{1+\exp(-\boldsymbol{\theta}^T \mathbf{x}_i)} \right) + (1-y_i) \ln \left(1 - \frac{1}{1+\exp(-\boldsymbol{\theta}^T \mathbf{x}_i)} \right) \right]$$

3. On rappelle que le modèle logistique repose sur l'utilisation d'une fonction logistique

$$s(x) = \frac{1}{1+\exp(-x)}.$$

   Etudier la convexité de cette fonction.

   Cette fonction est deux fois dérivables et la dérivée première est donnée par

$$\begin{aligned} s'(x) &= \frac{\exp(-x)}{(1+\exp(-x))^2}, \\ &= \frac{}{1+\exp(-x)} \times \frac{\exp(-x)}{1+\exp(-x)}, \\ &= \frac{1}{1+\exp(-x)} \left(1 - \frac{1}{1+\exp(-x)} \right), \\ &= s(x)(1-s(x)). \end{aligned}$$

   La dérivée seconde, en se basant sur les calculs précédents, est alors donnée par :

$$\begin{aligned} s''(x) &= s'(x)(1-s(x)) - s'(x)s(x), \\ &= s(x)(1-s(x))(1-s(x)) - s(x)(1-s(x))s(x), \\ &= s(x)(1-s(x))[1-s(x)-s(x)], \\ &= s(x)(1-s(x))(1-2s(x)) \end{aligned}$$

   La dérivée seconde est positive est donc positive pour tout x tel que $1-2(x) > 0$, donc pour tout x tel que $s(x) < 1/2$, soit $x < 0$. La fonction est donc convexe sur $\mathbb{R}_-$ et concave sur $\mathbb{R}_+$.

Méthodes ensemblistes

1. Quelles sont les deux grandes méthodes ensemblistes vues en cours ? On décrira brièvement les principes de ces deux approches. On pourra également citer un algorithme emblématique des différentes approches ensemblistes.

   Nous avons étudier deux grandes méthodes ensemblistes dans le cadre de ce cours.

   - le **bagging** qui consiste à construire une série de modèles à l'aide d'échantillons bootstrap. Ces modèles sont donc appris de façon indépendante à l'aide de données partiellement différentes d'un modèle à un autre.
     Un modèle emblématique de ce type d'approche est la *forêt aléatoire*.
   - le *boosting* qui consiste à construire une série de modèles à l'aide d'un même échantillon mais dont les exemples seront pondérés à chaque itération en fonction des résultats des apprenants. Ces modèles sont donc appris de façon itérative.
     Un modèle emblématique de ce type d'approche est *Adaboost*, que l'on retrouve ci-dessous.

2. Décrire les différentes étapes de l'algorithme Adaboost ci-dessous :

Input: Echantillon d'apprentissage S de taille m,
un nombre T de modèles
Output: Un modèle $H_T = \sum_{t=0}^{T} \alpha_t h_t$
begin
  Distribution uniforme $w_i^{(0)} = \dfrac{1}{m}$
  for $t = 1, ..., T$ **do**
    Apprendre un classifieur h_t à partir d'un algorithme $\mathcal{A}$
    Calculer l'erreur ε_t de l'algorithme.
    if $\varepsilon_t > 1/2$ **then**
      Stop
    else
      Calculer $\alpha_{(t)} = \dfrac{1}{2} \ln \left(\dfrac{1 - \varepsilon_t}{\varepsilon_t} \right)$
      $w_i^{(t)} = w_i^{(t-1)} \dfrac{\exp(-\alpha_t y_i h_t(\mathbf{x}_i))}{Z_t}$
  Poser $H_T = \sum_{t=0}^{T} \alpha_t h_t$
  return H_T

Au départ, nous attribuons le même poids à l'ensemble de nos exemples. De façon itérative nous allons ensuite :

- apprendre un classifieur sur les données d'entraînement avec la pondération courante.
- estimer l'erreur du modèle ε du classifieur appris
- si cette erreur est plus grande que 1/2, la classifieur n'est pas pris en compte et la procédure s'achève. Sinon
- on calcule le poids du modèle dans la combinaison. Ce poids α est donné par

$$\alpha = \frac{1}{2} \ln \left(\frac{1-\varepsilon}{\varepsilon} \right).$$

- le poids des exemples est ensuite modifié de façon à ce qu'un exemple mal classé ait un poids plus important et un exemple bien classé verra son poids diminuer.

3. Quelle est loss que l'on cherche à minimiser avec l'algorithme adaboost ?

   L'algorithme se concentre sur la minimisation de la loss exponentielle.

   On considère le modèle suivant :

$$H(\mathbf{x}) = sign(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x}) + \alpha_3 h_3(\mathbf{x})),$$

   où $(\alpha_1, \alpha_2, \alpha_3) = (1, 2, 3)$ et $h_1(\mathbf{x}) = sign(2), h_2(\mathbf{x}) = sign(2x_1)$ et $h_3(\mathbf{x}) = sign(-x_1 + x_2 - 1)$.

   (a) En repartant de la description de l'algorithme Adaboost, quel est le *weak learner* qui a l'erreur la plus faible ?

   Etant donnés les poids des différents classifieurs et lien qui existe entre l'erreur d'un classfieur h_t, noté ε_t, et son poids α_t. Le classifieur h_t avec le poids le plus important est celui avec l'erreur la plus faible. Donc c'est ici le classifieur h_3 qui a l'erreur la plus faible et donc les meilleurs performances.

   (b) Prédire l'étiquette des données $\mathbf{x}_1, \mathbf{x}_2$ et $\mathbf{x}_3$ de l'ensemble S défini à l'exercice précédent.

   On cherche à prédire les étiquettes de

$$\mathbf{x}_1 = (-2, 1), \quad \mathbf{x}_2 = (1, -1) \quad \text{et} \quad \mathbf{x}_3 = (3, 4).$$

Les prédictions du modèles sur les différentes données sont présentées dans la table ci-dessous

	$\mathbf{x}_1 = (-2, 1)$	$\mathbf{x}_2 = (1, -1)$	$\mathbf{x}_3 = (3, 4)$
$h_1(\mathbf{x})$	$+1$	$+1$	$+1$
$h_2(\mathbf{x})$	-1	$+1$	$+1$
$h_3(\mathbf{x})$	$+1$	$+1$	-1
$H(\mathbf{x})$	$+1$	$+1$	± 1