

Statistique en Grande Dimension et Apprentissage

Fabien Panloup

Notes de Cours

Master 2 Data Science - Université d'Angers

Année universitaire 2023-2024

Introduction

Ce cours est consacré à la présentation des méthodes d'apprentissage statistique en général et à leur mise en pratique notamment en grande dimension, *i.e* dans le cadre où le nombre de paramètres du problème est grand. On s'intéressera dans un premier temps à quelques propriétés théoriques et aux fondements permettant d'estimer le *risque* relatif à un problème statistique donné et on tentera de présenter plusieurs familles d'algorithmes fondamentaux.

Ce cours venant à la suite de celui de “Data Mining” de M1, il se focalisera principalement sur les algorithmes non traités dans ce cours, et en particulier, sur ceux qui sont spécifiques à la grande dimension.

Enfin, il est important de noter que ce polycopié contient des notes de cours perfectibles et non exhaustives. En particulier, s'il reste du temps, nous reviendrons également sur certains algorithmes de classification non supervisée et évoquerons des développements dans des domaines tels que le traitement spécifique des données textuelles ou bien celui des systèmes de recommandation.

Table des matières

1	Changements	9
2	Apprentissage Statistique : Fondements	11
2.1	Introduction	11
2.1.1	Généralités	11
2.1.2	Supervisé/Non Supervisé	12
2.1.3	Adaptatif/Non adaptatif	15
2.1.4	Paramétrique/Non paramétrique	15
2.2	Classification supervisée : modélisation du problème	15
2.2.1	Loi et Espérance conditionnelles : rappels rapides	16
2.2.2	Risque	17
2.2.3	Algorithmes de prévision	19
2.2.4	Consistance et algorithme par moyennisation locale	20
2.2.5	Comment sélectionner un algorithme/un modèle	20
2.2.6	Estimation du risque	21
2.2.7	Overfitting	21
2.2.8	Estimation du risque	23
2.2.9	Validation croisée	24
3	Fléau de la dimension	27
3.1	Qu'est-ce que la grande dimension ?	27
3.1.1	Exemples	27
3.1.2	A propos de n et p	27
3.2	De la statistique classique à la statistique en grande dimension	28
3.2.1	n , p et statistique	28
3.3	Fléau de la dimension (Curse of dimensionality)	29
3.3.1	Quelques exemples de difficultés liées à la grande dimension	29
3.3.2	Moyennage local ?	30
3.3.3	Exemple de la régression linéaire	31
3.3.4	Matrice de covariance	31
3.3.5	Et l'ACP ?	32
3.4	Que peut-on espérer en grande dimension ?	32
4	Arbres Binaires de décision et extensions agrégées	35
4.1	Arbres binaires de décision	35
4.1.1	Construction d'un arbre binaire	35
4.1.2	Forme générale du résultat	35
4.1.3	Un premier exemple	36
4.1.4	Partitionnement de l'espace	36
4.1.5	Un exemple	36

4.1.6	Un exemple	38
4.1.7	Un exemple	38
4.1.8	Un exemple	38
4.1.9	Remarques/questions avant une construction effective	38
4.1.10	Méthodes CART/C4.5	39
4.1.11	Construction d'un arbre binaire maximal	39
4.1.12	Critère d'homogénéité et Règle de division	39
4.1.13	A propos des divisions	40
4.1.14	Q_t pour la classification	40
4.1.15	Q_t pour la régression	40
4.1.16	Elagage	41
4.1.17	Comment élaguer ?	41
4.1.18	Avantages/Inconvénients	41
4.2	Bagging, Random Forests, Boosting	42
4.2.1	Agrégation de modèles	42
4.2.2	Bootstrap	42
4.2.3	Bagging	42
4.2.4	Forêts aléatoires	43
4.2.5	Boosting	44
4.2.6	Adaboost	45
4.2.7	Gradient Boosting	45
5	Régression linéaire en Grande Dimension	49
5.1	Régression Linéaire Classique : Rappels	49
5.1.1	Modèle	49
5.1.2	Régression linéaire et Apprentissage	49
5.1.3	Estimation de θ^*	50
5.1.4	Premières propriétés	50
5.1.5	Qualité du modèle linéaire classique	51
5.1.6	Régressions PCR/PLS	54
5.2	LASSO/RIDGE/ELASTIC NET	55
5.2.1	La pénalisation par la "norme" de θ	55
5.2.2	Rappels succincts sur le Lagrangien	56
5.2.3	A propos de la pénalisation ℓ_0	57
5.2.4	Lasso vs Ridge et Elastic Net	58
5.2.5	Evolution avec λ	58
5.2.6	Un résultat simple pour le Ridge	58
5.2.7	"Calcul" de $\hat{\theta}_\lambda$ pour le LASSO	60
5.2.8	En pratique	61
5.3	Prédiction/Estimation : Résultats	61
5.3.1	Un premier résultat de prédiction	61
5.3.2	La condition RE	63
5.3.3	Estimation	63
5.3.4	Prédiction II	63
5.3.5	Recouvrement du support	64
5.3.6	Extensions/Remarques	64
6	Support Vector Machines	67
6.1	Introduction	67
6.2	Linéairement/non linéairement séparables	67
6.3	Classificateurs linéaires à Vaste Séparateur de Marge	68
6.3.1	Construction du SVM linéaire	68

6.3.2	Hyperplans	68
6.3.3	Hyperplans de séparation	70
6.3.4	Définition du SVM (dans le cas linéairement séparable)	70
6.3.5	Reformulation du problème d'optimisation	70
6.3.6	Problème Dual	71
6.3.7	Calcul de β^* , β_0^*	72
6.3.8	Règle de Décision	72
6.4	SVM linéaires à marge flexible	72
6.4.1	SVM linéaire pour données non linéairement séparables ?	72
6.4.2	Problème d'optimisation associé (aux marges flexibles)	73
6.4.3	Résolution du problème aux marges flexibles	73
6.4.4	Vecteurs supports	74
6.5	SVM non linéaires	74
6.5.1	Principe des SVMs à noyau	74
6.5.2	Autres méthodes à noyau	76
7	Réseaux de Neurones et Deep Learning	79
7.1	Introduction	79
7.2	Réseaux de Neurones Artificiels	80
7.2.1	Neurone Artificiel	81
7.2.2	Le perceptron MultiCouche	83
7.3	Théorème d'approximation universelle	85
7.4	Rétropropagation/Estimation des paramètres	87
7.4.1	Problématique	87
7.4.2	Descente de Gradient/Descente de Gradient Stochastique	88
7.4.3	Rétropropagation	89
7.5	Réseaux de neurones à convolution	91
7.5.1	Convolution	91
7.5.2	Pooling	93
7.5.3	Fully Connected Network	93
7.6	RNN et GAN	93
7.7	Conclusion	94

Chapitre 1

Changements

Remplacer quelques exos de la feuille LASSO. Certains ne sont plus d'actualité. Inclure l'exercice de Giraud page 92 (sur l'unicite du LASSO). Construction a la main de variantes du LASSO : Group-Lasso Régression logistique /Fused-Lasso

Chapitre 2

Introduction à l'Apprentissage Statistique

2.1 Introduction

2.1.1 Généralités

L'apprentissage statistique (machine learning en anglais) désigne la science dédiée à l'exploitation des données issues d'un phénomène aléatoire. Le terme “exploitation” peut avoir plusieurs sens. On peut chercher à

- Décrire un phénomène : explorer/vérifier/décrire les relations entre les différentes variables au vu des observations
- Expliquer : comprendre/tester l'influence d'une variable ou d'un ou plusieurs facteurs dans un modèle statistique.
- Prédire : Prévoir un résultat, une réponse pour une nouvelle observation.
- Sélectionner les variables qui sont les plus influentes sur le phénomène
- Classer des individus ou des variables,...

Néanmoins, les objectifs généraux précédents et la définition elle-même ne permettent pas réellement de distinguer le terme “apprentissage” qui met en avant le caractère automatique et algorithmique de l'exploitation des données. Quelques tentatives de définitions du machine learning en vrac :

- “Field of Study that gives computers the ability to learn without being explicitly programmed” (Samuel, 1959).
- “The goal of machine learning is to build computer systems that can adapt and learn from their experience” (Dietterich, 1999)
- “A computer program is said to learn from experience E with some respect to some class of tasks T and performance measure P if its performance at tasks in T as measured by P improves with experience E (T. Mitchell, “Machine Learning”, 1997).

Exemples

Afin d'illustrer les différents objectifs décrits ci-dessus, voici quelques situations diverses :

- Identifier les mécanismes de résistance à un traitement du cancer/Sélectionner le meilleur traitement au vu des données du patient/Prévoir sa réaction au traitement

- Identifier les gènes impliqués dans le développement d'une maladie
- Reconnaître des images (chiffres/formes/nodules,...)
- Reconnaître un spam
- Prévoir la consommation électrique d'un ensemble de foyers
- Classer les clients d'une assurance selon leurs données personnelles.
- Optimiser le choix de publicité sur un site web,...

2.1.2 Supervisé/Non Supervisé

La plupart des problèmes d'apprentissage statistique peuvent être classés en deux groupes : les problèmes **supervisés** et **non supervisés**:

Supervisé :

Pour chaque individu, on peut distinguer une “réponse” ou un “label” (étiquette, spécifique aux phénomènes à réponses qualitatives) que l'on notera généralement Y . Les autres variables sont appelées “prédicteurs” ou “variables explicatives” et sont souvent notées \mathbf{X} .

- On dispose donc d'un ensemble de données $(X_i, Y_i)_{i=1}^n$, où n est le nombre d'observations (images, patients,...), $X_i = (X_i^1, \dots, X_i^p)$ est le vecteur (ligne ou colonne) des variables (caractéristiques) par individu.
- Dans ce cadre, l'objectif naturel est de déterminer la “meilleure fonction” permettant d'approcher au mieux la vraie réponse y étant donné un vecteur d'entrée $x = (x_1, \dots, x_p)$.
- Lorsque la réponse est qualitative, on parle de classification. Dans un cadre quantitatif, on parle généralement de régression (même si ce terme est aussi utilisé dans le cadre précis de la classification : la régression logistique par exemple permet de faire de la classification). Lorsque le bruit est additif, le modèle de régression prend la forme générale suivante : $Y = f(\mathbf{X}) + \varepsilon$ où f est une fonction appartenant à une classe de fonctions fixée au départ.

Ci-dessous (Figure 2.1), un exemple simple de classification binaire. Dans ce cas, le but est de déterminer une règle permettant de classer la nouvelle observation dans le bon groupe. Notons que dans cet exemple,

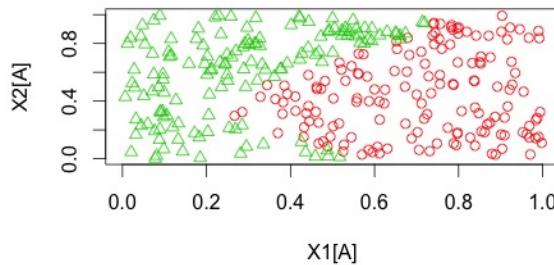


Figure 2.1: Deux groupes de couleur différente

$\mathbf{X} = (X_1, X_2)$ appartient à $[0, 1]^2$ tandis que Y appartient à $\{\circ, \triangle\}$. On reviendra sur cet exemple de base dans la suite. Un exemple d'algorithme : Les k -plus proches voisins : une règle naturelle. Pour un point donné, je

considère les k plus proches voisins (k à déterminer) et je choisis ma réponse au vu de celles de ses k voisins les plus proches.

- Dans un cadre qualitatif, on pourra choisir la réponse la plus fréquente.
- Dans un cadre quantitatif, on fera plutôt une moyenne de ces réponses.
- Ces règles peuvent bien sûr être raffinées (par exemple, en pondérant selon la distance du voisin).

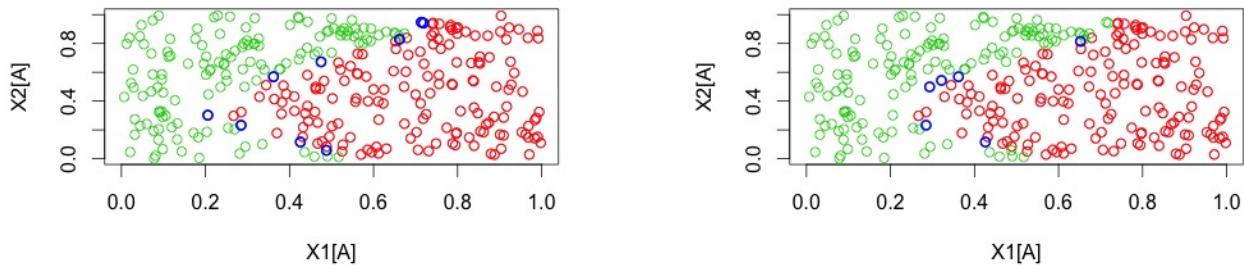


Figure 2.2: $k = 1$ à gauche, $k = 5$ à droite

Non Supervisé

On dispose d'un ensemble de mesures pour chaque individu mais pas de réponse naturelle. On souhaite néanmoins tirer de l'information de cet ensemble d'observations. On peut alors chercher à comprendre les relations entre les variables et/ou les observations, identifier les (groupes de) variables les plus sensibles,.... Ces méthodes peuvent être vues comme un moyen de pallier l'absence de réponse. Elles peuvent être utilisées dans un but préliminaire mais sont de plus en plus présentes pour l'apprentissage de données complexes.

On peut identifier deux ou trois types importants d'algorithmes dans cette classe :

- ceux qui permettent de “constituer des groupes de variables” ou plus précisément de “déterminer les variables ou combinaisons de variables les plus variantes” : c'est l'objectif de l'**analyse en composantes principales** ou de certaines de ses variantes pour le “clustering de variables”.
- ceux qui sont destinés à fabriquer des groupes d'observations. On parle alors de “clustering d'individus”. Les méthodes de **K-means** ou de **clustering hiérarchique** en sont des exemples classiques.
- ceux qui permettent d'approximer la loi de probabilité (estimation de densité/mélange...).

Dans les exemples ci-dessus, on remarque que l'on est plus dans un mode “explicatif”. Néanmoins, l'apprentissage non supervisé peut aujourd'hui être utilisé pour “créer des réponses”. On parle de modèles/algorithmes génératifs. Par exemple, il peut s'agir de créer une nouvelle image (ex: portrait d'une personne non existante mais qui a bien les caractéristiques “voulues”), de créer une nouvelle manière de jouer à un jeu. Dans ce cadre, on constate que le point important est que l'on peut a posteriori faire apprendre l'algorithme en lui “donnant un bon point” lorsque le résultat est satisfaisant (image réussie, partie gagnée contre un adversaire). On parle alors d'apprentissage par renforcement (ce terme peut aussi correspondre à des situations supervisées). Dans ce cadre, l'apprentissage purement supervisé devient en quelque sorte obsolète. Sur ce point, le lecteur intéressé pourra écouter le cours de Y. Lecun au Collège de France dont le résumé écrit est accessible ici :

https://www.college-de-france.fr/media/yann-lecun/UPL4485925235409209505_Intelligence_Artificielle_____Y._LeCun.pdf.

Semi-supervisé/Etiquettes manquantes

Bien entendu, la pratique comporte des exemples d'applications qui ne sont pas forcément parfaitement classées dans l'une ou l'autre des classes (ce point est d'ailleurs déjà présent dans la discussion précédente). On parle alors d'apprentissage semi-supervisé. C'est le cas par exemple lorsque,

- Seule une partie des données possède une réponse. Par exemple, si la réponse est la durée de vie après traitement, les personnes encore en vie n'ont pas d'étiquette !! Cette situation peut aussi intervenir lorsque l'étiquetage humain¹ est trop "coûteux" (en un sens variable). Dans ce cas, on peut envisager de labelliser une partie des données puis d'utiliser un mélange d'apprentissage supervisé et non supervisé pour apprendre la *règle de décision* (cf terminologie à venir).
- les réponses ne sont pas les mêmes: en médecine, on peut penser à des patients traités dans des instituts de recherche dans plusieurs endroits du monde où l'on n'a pas considéré les mêmes réponses (parce qu'ils n'ont pas reçu le même traitement). Néanmoins, les données étant difficiles à obtenir, on souhaiterait être capable de mettre en commun ces données).

Pour terminer cette section non exhaustive, il est important de noter que le classement supervisé/non supervisé/semi-supervisé a vocation à tenter d'ordonner les problèmes ainsi que leurs classes de solutions mais que ce point de vue est de toute façon erroné. Les problèmes d'apprentissage de données procurent aujourd'hui des situations diverses qui génèrent elles-mêmes des questions diverses.

Parmi ces problèmes divers, on peut évoquer celui des **systèmes de recommandation** qui sont un outil majeur du **commerce en ligne**. La question posée est grossièrement la suivante : comment proposer le meilleur produit au client au vu de ses précédentes actions sur le web ? On trouve dans le même registre l'exemple de Netflix où les produits sont des films et les précédentes actions, les films qui ont été vus avant (et leur note). La résolution de ce type de problème s'appuie sur un système de **filtrage collaboratif**. En gros, il s'agit de tenter d'établir des similarités entre individus à partir de films vus mais qui ne sont généralement pas les mêmes. Il s'agit bien d'une forme d'apprentissage non supervisé, mais dont la forme ne ressemble pas aux problèmes évoqués précédemment et dont les réponses apportées sont donc également différentes.

Imperfection des données

Ci-dessus, on a évoqué les diverses situations d'apprentissage. Ci-dessous, on va s'intéresser plutôt aux situations "classiques" mais où l'apprentissage est rendu difficile par l'imperfection des données. :

- les "**vraies**" **données manquantes** : comment gérer l'absence de certaines données pour une partie des individus (notamment dans les paramètres d'entrée) ? Plusieurs réponses selon les situations :
 - Exclure l'individu; dépend de l'importance des variables manquantes, du nombre d'individus...
 - Remplacer la donnée absente par une valeur raisonnable. Au niveau 0, on peut remplacer par la médiane de l'ensemble des individus (pour une variable quantitative) mais ceci reste très arbitraire. Au niveau 1, on peut choisir de prendre la médiane d'une partie des individus (les plus proches par exemple). Cela signifie que l'on parie sur le fait que la donnée manquante de l'individu sera proche des voisins les plus proches. Cela reste encore discutable. Au niveau 2, on peut tenter d'apprendre : on regarde la donnée manquante comme une variable réponse relativement aux autres variables et on tente d'optimiser la prédiction de cette valeur manquante à l'aide des méthodes d'apprentissage (que l'on va voir dans la suite). Cette dernière méthode peut parfois générer du *surapprentissage*.
 - On pourra consulter le cours <https://perso.univ-rennes1.fr/valerie.monbet/doc/cours/IntroDM/Chapitre4.pdf> pour plus de détails sur le sujet.

¹On parle aussi d'*annotation*.

- Les classes **mal équilibrées** : Dans de nombreuses situations, les classes peuvent être mal “balancées”. On peut penser par exemple au problème de détection de fraude. La classe “non frauduleuse” est par nature mieux représentée. Ce problème peut biaiser l’apprentissage. Dans ce cas, il peut être nécessaire de
 - Adapter la *fonction de perte* en attribuant un poids plus élevé aux classes mal représentées (voir aussi précision, rappel, F_1 -score, MCC, …, cf TD).
 - Générer artificiellement de nouveaux individus dans les classes où l’effectif est trop faible en clonant les individus existants ou en les interpolant… (**SMOTE** par exemple).
 - Attention, cloner ou interpoler fait perdre la propriété d’indépendance et de loi identique entre les observations. Ainsi, ce type de méthode est à considérer avec prudence.

2.1.3 Adaptatif/Non adaptatif

Une autre information importante à prendre en compte dans l’étude d’un problème est la manière dont les données arrivent.

- Dispose-t-on de toutes les données à un instant fixé ?
- Les données arrivent-elles au fil du temps (“on the fly”) ?

Dans la deuxième situation, le caractère adaptatif de la méthode d’apprentissage est un élément important notamment pour le calcul effectif des prédictions. Par adaptatif, on entend : la faculté de l’algorithme à être mis à jour lors de l’arrivée d’une nouvelle donnée sans devoir tout recalculer. L’algorithme des k -plus proches voisins peut être programmé de manière adaptative par exemple (Exercice).

2.1.4 Paramétrique/Non paramétrique

Considérons le problème de base où $(Z_1, \dots, Z_n)_{n \geq 1}$ est issu d’une loi \mathbb{P} inconnue que l’on cherche à estimer. On parle de Statistique

- Paramétrique : lorsque la loi de probabilité $\mathbb{P} \in \{\mathbb{P}_\theta, \theta \in \Theta\}$ où $\Theta \subset \mathbb{R}^d$.
- Non paramétrique lorsque \mathbb{P} vit dans un espace de dimension infinie (ou finie mais tendant vers $+\infty$ avec n). Deux exemples :
 - Estimation de la densité f de la loi \mathbb{P} (relativement à une mesure donnée, mesure de Lebesgue par exemple). Dans ce cas, l’approche non paramétrique consiste à supposer que f vit dans un espace de fonctions fixé (par exemple, l’ensemble des fonctions positives, \mathcal{C}^2 d’intégrale égale à 1).
 - Régression non paramétrique : $Y = f(\mathbf{X}) + \varepsilon$ où f vit dans un espace de fonctions de dimension infinie (alors que la régression linéaire par exemple est clairement paramétrique).

2.2 Classification supervisée : modélisation du problème

Soit $(X_1, Y_1), \dots (X_n, Y_n)$, un échantillon d’apprentissage issu d’une loi conjointe \mathbb{P} sur $\mathcal{X} \times \mathcal{Y}$.

Définition 2.2.1. 1. Une règle de prévision/régression/décision/discrimination est une fonction $f : \mathcal{X} \rightarrow \mathcal{Y}$ qui à \mathbf{x} associe la sortie $f(\mathbf{x})$.

2. Une fonction de perte ℓ est une fonction positive définie sur $\mathcal{Y} \times \mathcal{Y}$ telle que $\ell(y, y') > 0$ dès que $y \neq y'$.

Exemples importants : En régression, $\ell(y, y') = |y - y'|^2$ ou plus généralement, $\ell(y, y') = |y - y'|^\alpha$, $\alpha > 0$, et en classification : $\ell(y, y') = \mathbf{1}_{y \neq y'}$ (voir TD pour d’autres exemples).

2.2.1 Loi et Espérance conditionnelles : rappels rapides

Loi conditionnelle, espérance conditionnelle sont des notions qui jouent un rôle important en apprentissage. Commençons par en rappeler les définitions dans le cas où X est à valeurs discrètes. Notons \mathcal{X} un ensemble discret, X une variable aléatoire à valeurs dans \mathcal{X} et Y une variable aléatoire réelle. Supposons pour simplifier que $\mathbb{P}(X = x) > 0$ pour tout $x \in \mathcal{X}$. Dans ce cas, pour toute fonction f borélienne bornée,

$$\mathbb{E}[f(Y)|X = x] = \frac{1}{\mathbb{P}(X = x)} \mathbb{E}[f(Y)1_{\{X=x\}}] =: \Phi_f(x)$$

est l'espérance conditionnelle de $f(Y)$ sachant $X = x$. Remarquons que dans le cas où $f(y) = 1_{\{y \in A\}}$, on retrouve la probabilité conditionnelle (classique) de l'événement $\{Y \in A\}$ relativement à $\{X = x\}$. On appelle alors *espérance conditionnelle de $f(Y)$ sachant X* la variable aléatoire définie par :

$$\mathbb{E}[f(Y)|X] = \Phi_f(X).$$

On appelle aussi loi conditionnelle de Y sachant X $\mathcal{L}(Y|X)$, la loi de probabilité $\mu_{Y|X}$ définie par : pour toute fonction f borélienne bornée,

$$\int f d\mu_{Y|X} = \mathbb{E}[f(Y)|X].$$

N.B. Dans le cas où $\mathcal{Y} = \{y_1, \dots, y_r\}$, $\mathcal{L}(Y|X)$ est simplement la donnée de $\mathbb{P}(Y = y_k|X)$ pour tout $k \in \{1, \dots, r\}$. On note que la mesure μ est aléatoire et $\sigma(X)$ -mesurable. Plus simplement, on utilisera la loi conditionnelle de Y sachant $X = x$ que l'on notera $\mathcal{L}(Y|X = x)$, la loi $\mu_{Y|X=x}$ définie par : pour toute fonction f borélienne bornée,

$$\int f d\mu_{Y|X=x} = \mathbb{E}[f(Y)|X = x].$$

L'extension au cas général (celui des variables continues) est plus délicate car l'événement $\{X = x\}$ est de probabilité nulle mais le sens intuitif reste le même. Notons que le cadre standard est celui de $\mathbb{E}[Y|\mathcal{F}]$, i.e. de l'espérance relativement à la tribu \mathcal{F} (ici, on se limite au cas où $\mathcal{F} = \sigma(X)$). Nous donnons ici quelques propriétés importantes sans les démontrer.

Proposition 2.2.2. 1. $\mathbb{E}[f(Y)g(X)] = \mathbb{E}[\mathbb{E}[f(Y)|X]g(X)]$ pour toute fonction g mesurable bornée (c'est la “vraie définition”).

2. En particulier, $\mathbb{E}[\mathbb{E}[f(Y)|X]] = \mathbb{E}[f(Y)]$ (prendre $g = 1$).

3. Si Y est $\sigma(X)$ -mesurable, $\mathbb{E}[f(Y)|X] = f(Y)$.

4. Si Y est indépendant de X , alors $\mathbb{E}[f(Y)|X] = \mathbb{E}[f(Y)]$.

5. Sous des hypothèses d'intégrabilité adéquates,

$$\mathbb{E}[Y|X] = \inf_{Z \text{ } \sigma(X)\text{-mesurable tq } \mathbb{E}[|Z|^2] < +\infty} \mathbb{E}[(Y - Z)^2]. \quad (2.2.1)$$

En d'autres termes, l'espérance conditionnelle est la meilleure estimation de $f(Y)$ (au sens L^2) conditionnellement à l'information apportée par $\sigma(X)$.

On trouvera des détails sur le sujet dans de nombreux ouvrages classiques de probabilité.

Remarque 2.2.3. Le point 5 est en fait une conséquence du fait que $\mathbb{E}[Y|X]$ est une projection orthogonale. Plus précisément, notons $E = \mathbb{L}^2(\Omega, \mathcal{F}, \mathbb{P})$, l'espace vectoriel des variables aléatoires \mathcal{F} -mesurables telles que $\mathbb{E}[Y^2] < +\infty$, muni du produit scalaire suivant :

$$\langle Y_1, Y_2 \rangle = \mathbb{E}[Y_1 Y_2].$$

$F := \mathbb{L}^2(\Omega, \sigma(X), \mathbb{P})$ est un sous-espace vectoriel de E et par définition pour tout vecteur Z de F ,

$$\langle Y - \mathbb{E}[Y|X], Z \rangle = \mathbb{E}[YZ] - \mathbb{E}[\mathbb{E}[Y|X]Z] = 0.$$

C'est donc bien une projection orthogonale. On retrouve alors classiquement la propriété (2.2.1) en écrivant:

$$\mathbb{E}[(Y-Z)^2] = \mathbb{E}[(Y-\mathbb{E}[Y|X])^2] + 2\langle Y - \mathbb{E}[Y|X], \mathbb{E}[Y|X] - Z \rangle + \mathbb{E}[(\mathbb{E}[Y|X] - Z)^2] = \mathbb{E}[(Y-\mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - Z)^2]$$

où l'on a utilisé que le second terme était nul puisque $Y - \mathbb{E}[Y|X]$ est orthogonal à l'espace vectoriel F (et que $\mathbb{E}[Y|X] - Z$ appartient à F par définition). On voit alors que le terme de droite est minimisé lorsque $Z = \mathbb{E}[Y|X]$.

2.2.2 Risque

Définition 2.2.4. Pour une fonction de perte donnée, on appelle risque ou erreur de généralisation d'une règle de prévision f la quantité :

$$R_f = \mathbb{E}[\ell(Y, f(\mathbf{X}))].$$

On dit qu'une règle f^* est optimale si,

$$R_{f^*} = \inf_{f \in \mathcal{F}} R_f.$$

Une telle règle est appelée prédicteur/règle de Bayes. Pour une règle de prévision f , la quantité $R_f - R_{f^*}$ est appelée l'excès de risque tandis que le risque minimal est appelé risque de Bayes.

Remarque 2.2.5. En pratique, on cherche souvent la règle de décision au sein d'une sous-classe \mathcal{S} de l'ensemble des fonctions. Par exemple, lorsque l'on fait de la régression linéaire, on cherche à trouver la meilleure fonction f parmi les fonctions de la forme $f(x) = \langle x, \theta \rangle$, $\theta \in \mathbb{R}^p$. Evidemment, en pratique, la fonction f^* n'appartient pas nécessairement à cet ensemble. Cela génère donc du *bias* et la quantité

$$\inf_{f \in \mathcal{S}} R_f - R_{f^*}$$

est appelée *erreur d'approximation*.

On est capable pour certaines pertes de définir de manière formelle (mais explicite) les règles optimales :

Théorème 2.2.6. (i) En régression avec $\mathcal{Y} = \mathbb{R}$ et $\ell(y, y') = |y - y'|^2$, $f^*(\mathbf{x}) = \mathbb{E}[Y|X = \mathbf{x}]$ est la règle de Bayes.

(ii) En régression avec $\mathcal{Y} = \mathbb{R}$ et $\ell(y, y') = |y - y'|$, la fonction $f^*(\mathbf{x}) = \text{médiane}(\mathcal{L}(Y|X = \mathbf{x}))$ est la règle de Bayes.

(iii) En classification (avec \mathcal{Y} de cardinal fini) et $\ell(y, y') = 1_{y \neq y'}$, la fonction f^* définie pour tout $\mathbf{x} \in \mathcal{X}$ par

$$f^*(\mathbf{x}) = \operatorname{argmax}_{y \in \mathcal{Y}} \mathbb{P}(Y = y | \mathbf{X} = \mathbf{x}).$$

Notons

$$\eta(x, y) = \mathbb{P}(Y = y | \mathbf{X} = x).$$

A x fixé, la probabilité de se tromper avec la règle f^* est égale à $1 - \max_{y \in \mathcal{Y}} \eta(x, y)$. On a donc

$$R_{f^*} = \mathbb{E}[1 - \max_{y \in \mathcal{Y}} \eta(X, y)] = \inf_f R_f. \quad (R_{f^*} \leq 1 - \frac{1}{\operatorname{Card}(\mathcal{Y})}).$$

Preuve. (i) Exercice.

(ii) Admis.

(iii) On considère seulement le cas binaire : $\mathcal{Y} = \{-1, 1\}$,

$$f^*(\mathbf{x}) = \operatorname{Argmax}_{y \in \{-1, 1\}} \mathbb{P}(Y = y | \mathbf{X} = \mathbf{x}).$$

Soit $f : \mathcal{X} \rightarrow \mathcal{Y}$ une règle de prévision. Elle est donc de la forme :

$$f(x) = \begin{cases} 1 & \text{si } x \in A \\ -1 & \text{si } x \in A^c \end{cases}$$

On cherche à prouver que $R_{f^*} = \min_{f: \mathcal{X} \rightarrow \mathcal{Y}} R_f$. Pour simplifier, on note $\eta(x) = \mathbb{P}(Y = 1 | X = x)$ de sorte que $\mathbb{P}(Y = -1 | X = x) = 1 - \eta(x)$.

$$\begin{aligned} R_f &= \mathbb{P}(Y \neq f(\mathbf{X})) = \mathbb{P}(Y = 1, f(\mathbf{X}) = -1) + \mathbb{P}(Y = -1, f(\mathbf{X}) = 1) \\ &= \mathbb{P}(\mathbf{X} \in A^c, Y = 1) + \mathbb{P}(\mathbf{X} \in A, Y = 1) \\ &= \mathbb{E}[\mathbb{P}(Y = 1 | \mathbf{X}) \mathbf{1}_{\{\mathbf{X} \in A^c\}}] + \mathbb{E}[\mathbb{P}(Y = -1 | \mathbf{X}) \mathbf{1}_{\{\mathbf{X} \in A\}}] \\ &= \mathbb{E}[\eta(X) \mathbf{1}_{\{\mathbf{X} \in A^c\}}] + \mathbb{E}[(1 - \eta(X)) \mathbf{1}_{\{\mathbf{X} \in A\}}] \\ &\geq \mathbb{E}[\min(\eta(X), 1 - \eta(X))]. \end{aligned}$$

Or, lorsque $f = f^*$, l'ensemble A satisfait :

$$A = \{x \in \mathcal{X}, \eta(x) \geq 1 - \eta(x)\} \quad (= \{x \in \mathcal{X}, \eta(x) \geq 1/2\}).$$

Ainsi,

$$\begin{cases} \eta(x) = \min(\eta(x), 1 - \eta(x)) & \text{si } x \in A^c \\ 1 - \eta(x) = \min(\eta(x), 1 - \eta(x)) & \text{si } x \in A \end{cases}$$

de sorte que

$$\mathbb{E}[\eta(X) \mathbf{1}_{\{\mathbf{X} \in A^c\}}] + \mathbb{E}[(1 - \eta(X)) \mathbf{1}_{\{\mathbf{X} \in A\}}] = \mathbb{E}[\min(\eta(X), 1 - \eta(X))].$$

Ceci conclut la preuve. ■

Remarque 2.2.7. En classification binaire, la règle de Bayes est à comprendre de la manière suivante. Selon les zones de l'espace, la valeur de la réponse Y est tirée selon un jeu de pile ou face. Si la probabilité de faire “Pile” est supérieure à 1/2, on choisit 1, si elle est plus faible que 1/2, on choisit -1.

Remarque 2.2.8. Dans les algorithmes les plus récents (XGBOOST, Réseaux de neurones), l'approche utilisée en classification est moins basée sur l'erreur dite de classification (*accuracy* en anglais). L'erreur est maintenant souvent étudiée au niveau de la probabilité $\mathbb{P}(Y = y | X = x)$ plutôt que sur la règle de prévision elle-même (qui s'en déduit par maximisation). Les mots-clés associés sont la fonction *softmax* et l'*entropie croisée*. On reviendra sur ces notions plus tard. Par ailleurs, dans les approches classiques, comme cela a été signalé précédemment, l'erreur de classification est généralement complétée par d'autres mesures telles que le F_1 -score afin de mieux mesurer la prédiction associée à chaque classe (et d'éviter par exemple des désagréments liés au déséquilibre des classes...).

A ce stade, il est important de comprendre que ce qui précède permet simplement de savoir ce que l'on cherche. Plus exactement, si l'on avait accès à la loi du couple (X, Y) , alors prédire au mieux le modèle (à minima pour les fonctions de perte considérées ici) consisterait à choisir la règle de Bayes. Malheureusement, en pratique, la loi du couple (X, Y) est évidemment inconnue. Il nous faudra donc à partir des observations d'essayer en quelque sorte de mimer la règle de Bayes (via la loi des grands nombres). Plus généralement et plus précisément, on essaiera, pour un algorithme donné d'*estimer* la fonction minimisant l'excès de risque défini plus haut.

2.2.3 Algorithmes de prévision

Définition 2.2.9. Etant donné un échantillon de taille n , un *algorithme de prévision* (ou **prédicteur**) est une application qui à $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$ associe une fonction notée \hat{f}_n de \mathcal{X} vers \mathcal{Y} .

Par exemple, dans le cas des k plus proches voisins, on a :

- en discrimination binaire :

$$\hat{f}_n(\mathbf{x}) = \text{sgn}(\eta_n(\mathbf{x})) \text{ où } \eta_n(\mathbf{x}) = \frac{\text{Card}\{\text{"voisins de } \mathbf{x}\", Y_i = 1\}}{k} - \frac{1}{2}.$$

- en régression :

$$\hat{f}_n(\mathbf{x}) = \frac{1}{k} \sum_{k \text{ ppv de } \mathbf{x}} Y_i.$$

Autre exemple : lorsque l'on utilise la régression linéaire standard, on pose $\hat{f}_n(\mathbf{x}) = \langle \mathbf{x}, \hat{\theta}_n \rangle$ où

$$\hat{\theta}_n = \underset{\theta \in \mathbb{R}^d}{\text{Argmin}} \sum_{i=1}^n (Y_i - \langle \mathbf{X}_i, \theta \rangle)^2.$$

Exercice. On pourra vérifier que dans les exemples ci-dessus, $\hat{f}_n = f_n(\mathcal{D}_n)$ où f_n est une fonction déterministe.

Risque et Algorithmes de prévision

On cherche maintenant à mesurer la qualité de l'algorithme de prévision.

Définition 2.2.10. Le risque (moyen) d'un algorithme de prévision est défini par

$$\mathbb{E}_{(\mathbf{X}, Y)}[R_{\hat{f}_n}] = \mathbb{E}_{(\mathbf{X}, Y)}[\ell(Y, \hat{f}_n(\mathbf{X}))].$$

Quelques exemples :

- discrimination binaire, fonction de perte $1_{y \neq y'}$:

$$\mathbb{E}_{(\mathbf{X}, Y)}[R_{\hat{f}_n}] = \mathbb{P}_{(\mathbf{X}, Y)}(Y \neq \hat{f}_n(\mathbf{X})).$$

- Quantitatif, fonction de perte $\ell(y, y') = (y - y')^2$:

$$\mathbb{E}_{(\mathbf{X}, Y)}[R_{\hat{f}_n}] = \mathbb{E}[(Y - \langle \mathbf{X}, \hat{\theta}_n \rangle)^2].$$

Important ! \hat{f}_n est construit à partir de l'échantillon $\{(\mathbf{X}_1, Y_1), \dots, (\mathbf{X}_n, Y_n)\}$ et (\mathbf{X}, Y) représente un échantillon indépendant de cette suite de v.a. Ainsi, on a deux niveaux d'aléas :

- l'aléa contenu dans \mathcal{D}_n qui sert à construire \hat{f}_n
- l'aléa relatif au couple (\mathbf{X}, Y) indépendant utilisé pour le calcul de R_f avec $f = \hat{f}_n$ (voir Section 2.2.5 pour plus de détails).

Dans la suite, le risque sera estimé via l'utilisation d'un *échantillon test*, indépendant de celui dit *d'apprentissage* ou *d'entraînement* qui a servi à la construction du prédicteur \hat{f}_n .

2.2.4 Consistance et algorithme par moyennisation locale

Définition 2.2.11. L'algorithme est (faiblement) **consistant** si

$$\mathbb{E}_{(\mathbf{X}, Y)}[R_{\hat{f}_n}] \xrightarrow{n \rightarrow +\infty} \inf_{f \in \mathcal{F}} R_f.$$

Il est fortement consistant si $(R_{\hat{f}_n})_n$ converge *p.s.* vers $\inf_{f \in \mathcal{F}} R_f$.

Par défaut, le mot consistance fera dans la suite référence à la consistance faible.

Intéressons-nous à la consistance des algorithmes de prévision les plus naturels :

Définition 2.2.12. On appelle algorithme par moyennisation locale un algorithme basé sur une moyenne "locale" des observations. Attention, le terme local ici est à comprendre comme "avec une pondération décroissant avec la distance".

L'algorithme des k plus proches voisins en est un (pondération $1/k$ ou 0). Dans un cadre quantitatif, l'algorithme est défini par :

$$\hat{f}_n(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^n Y_i \mathbf{1}_{\{\mathbf{x}_i \text{ parmi les } k \text{ p.p.v. de } \mathbf{x}\}}.$$

Exercice. Expliquez pourquoi, l'algorithme ci-dessus est une approximation de la règle optimale (de Bayes) définie précédemment.

Si l'on revient maintenant à la définition des règles optimales, on peut montrer sous des conditions assez générales (cf TD) que :

Théorème 2.2.13. *Les algorithmes par moyennage local sont universellement consistants (Par exemple, pour les k -ppv, ça marche si $k_n \rightarrow +\infty$ et que $k_n/n \rightarrow 0$).*

Le résultat ci-dessus est volontairement énoncé de manière informelle (mais bien entendu, ce résultat peut s'énoncer sous des hypothèses précises). En réalité, la consistance n'est pas une notion satisfaisante en pratique car elle est seulement asymptotique. On souhaiterait évaluer l'erreur de prévision pour un échantillon fixé. Outre l'étude de la qualité de l'algorithme, le but est souvent de pouvoir choisir dans une famille celui qui est le plus efficace. Ce choix pourra dans un premier temps être établi par minimisation du risque (dont le calcul devra s'appuyer sur l'échantillon d'observation).

2.2.5 Comment sélectionner un algorithme/un modèle

En apprentissage statistique, les choix du modèle et de l'algorithme sont des étapes fondamentales pour optimiser la qualité de la prévision.

- Par modèle, il s'agit de choisir la classe de probabilités \mathbb{P} dans laquelle on cherche à approcher la loi de (\mathbf{X}, Y) . De la même manière, il peut s'agir de choisir la classe de fonctions f qui est destinée à approcher la règle optimale. Par exemple, si on choisit un modèle de régression linéaire (paramétrique), on cherche à approcher $\mathbb{E}[Y|\mathbf{X} = x]$ par des fonctions de la forme $f(x) = \langle x, \theta \rangle$.

De même, si l'on choisit un modèle de *partitionnement*, on cherchera à approcher la règle de Bayes par des fonctions de la forme :

$$f(x) = \sum_{i=1}^R a_i \mathbf{1}_{B_i}(x)$$

où les a_i sont des coefficients à estimer/déterminer et $\{B_1, \dots, B_R\}$ désigne une partition de l'ensemble \mathcal{X} . Notons que l'on cherche évidemment ici dans une sous-classe de l'ensemble des partitions (C'est le cas des arbres de décision binaires par exemple).

N.B. Comme on pourra le voir plus tard, un modèle peut être plus ou moins "riche/flexible" selon la taille de la classe de fonctions.

- Par algorithme, on entend la méthode utilisée pour estimer la meilleure fonction dans la classe de fonctions choisies. Notons que si la distinction entre modèle et algorithme est claire dans certains contextes, elle l'est moins dans d'autres. Par exemple, dans le cas des SVM/arbres/réseaux de neurones, cette distinction est claire. Par contre, pour les k plus proches voisins, les fonctions de prévision étant définies directement sur les données, il n'est pas possible de définir simplement la classe de fonctions considérée. Néanmoins, il est “moralement” clair que la richesse du “pseudo-ensemble” de fonctions associées à cet algorithme (donc du modèle) est décroissante avec le nombre de voisins.

Pour décider, l'approche la plus naturelle consiste à minimiser le risque. Pour cela, il faut commencer par être capable de l'estimer.

2.2.6 Estimation du risque

On rappelle que le risque moyen est défini par $\mathbb{E}_{\mathbf{X}, Y}[R_{\hat{f}_n}]$. Pour une fonction f fixée, la quantité R_f s'approche par le *risque empirique* :

$$\hat{R}_n(f) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f(X_i)).$$

En effet, par la loi des grands nombres (et si $\mathbb{E}[\ell(Y, f(X))] < +\infty$),

$$\hat{R}_n(f) \xrightarrow{n \rightarrow +\infty} R_f \quad p.s.$$

Ainsi, pour estimer le risque moyen, on peut être tenté d'utiliser

$$\hat{R}_n(\hat{f}_n) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, \hat{f}_n(X_i)).$$

Ceci donnerait

- la **proportion** de mal classés dans le cas de la classification (avec fonction de perte $\ell(y, y') = 1_{y \neq y'}$) : $\frac{1}{n} \sum_{k=1}^n 1_{Y_k \neq \hat{f}_n(\mathbf{X}_k)}$,
- $\frac{1}{n} \sum_{k=1}^n (Y_k - \hat{f}_n(\mathbf{X}_k))^2$ dans le cas de la régression (avec fonction de perte “moindres carrés”).

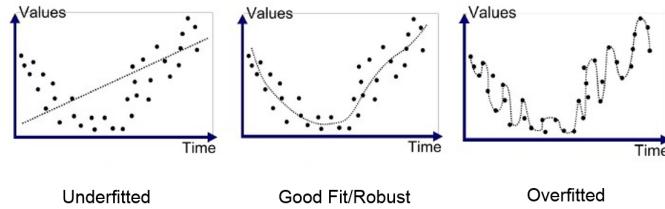
La quantité $\hat{R}_n(\hat{f}_n)$ est appelée *erreur d'entraînement* (“train error”) mais,

Attention : Elle ne peut être considérée comme une approximation du risque moyen car elles est, comme son nom l'indique, mesurée sur l'échantillon d'entraînement, *i.e.* sur l'échantillon qui a servi à construire \hat{f}_n . Mathématiquement, la loi des grands nombres ne s'applique pas directement car \hat{f}_n n'est pas indépendant de l'échantillon.

Du point de vue “apprentissage”, le problème mathématique se comprend très bien dans le cas d'un modèle (ou d'une classe de modèles) très **flexible**. Dans ce cas l'erreur peut être beaucoup plus faible que la véritable erreur. Pour se faire une intuition, on peut penser aux modèles de régression polynomiale (plus le degré du polynôme est grand, plus le modèle peut approcher/interpoler les points de l'échantillon). On peut également penser aux modèles construits par partition que l'on a évoqués précédemment. Dans ce cadre, plus le nombre de parties K est grand (ou plus l'arbre est *profond* dans le cas particulier des modèles d'arbres), plus le modèle est flexible (par exemple, si $K = n$, on peut évidemment fabriquer un modèle dont l'erreur d'entraînement est nulle).

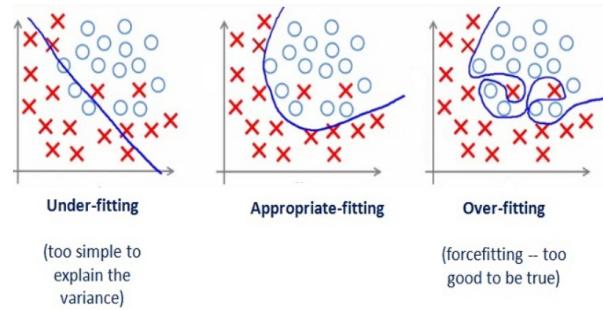
2.2.7 Overfitting

La notion de flexibilité du modèle est souvent associée à celle de “surapprentissage” (overfitting) ou “sous-apprentissage” (underfitting). Ci-dessous, deux situations simples pour bien comprendre la situation de surapprentissage. Dans la figure 2.3, on considère un problème de régression. On constate que selon la richesse de la classe de fonctions considérée, on a plus ou moins de capacité à interpoler les points de l'échantillon.

Figure 2.3: (issu de <https://medium.com/greyatom>)

On sent également que si dans le cas de gauche, la classe de fonctions envisagée est trop restreinte pour estimer le comportement de la relation entre \mathbf{X} et Y , le fait de trop vouloir coller aux observations ne permet pas non plus de dégager de l'information. L'erreur d'entraînement sur la figure de droite est nulle mais pour autant, il est probable que si l'on teste ce prédicteur sur un autre échantillon, l'erreur (dite de test) sera bien plus importante. Dans la figure du milieu, on dirait avec un langage “expert” (souvent teinté d'anglissimes) : “Le modèle fitte bien les données”.

Dans la figure 2.4, le même type de phénomène est considéré dans un cadre de classification binaire.

Figure 2.4: (issu de <https://medium.com/greyatom>)

Erreur d'estimation vs Erreur d'approximation : Plus le modèle est flexible, plus il est difficile à estimer, *i.e.* plus l'approximation de la meilleure fonction dans le modèle choisi nécessite de données. Plus précisément, considérons l'approche “simple” où l'on cherche la meilleure fonction dans une sous-classe \mathcal{S} de l'ensemble des fonctions. On rappelle que l'erreur dite d'approximation (ou biais du modèle) est définie par

$$\inf_{f \in \mathcal{S}} R_f - R_{f^*}.$$

A cette erreur doit s'ajouter l'erreur d'estimation, *i.e.* la “distance” entre la fonction \hat{f}_n et la meilleure fonction f_S^* de la classe \mathcal{S} définie par $f_S^* = \min_{f \in \mathcal{S}} R_f$ (pour simplifier, on suppose que le minimum existe). Plus précisément, à supposer que l'on puisse calculer :

$$\hat{f}_n = \operatorname{Argmin}_{f \in \mathcal{S}} \frac{1}{N} \sum_{k=1}^N \ell(Y_k, f(X_k)),$$

quelle erreur commet-on vis-à-vis de f_S^* ou plus précisément que vaut l'*erreur d'estimation*, *i.e.* la quantité

$$\mathbb{E}[R_{\hat{f}_n}] - R_{f_S^*}?$$

Cette quantité est généralement vue comme un terme de variance. Cette variance est très dépendante de la flexibilité du modèle et/ou de la dimension du problème (voir chapitre sur le “fléau de la dimension” pour plus

d'éléments à ce sujet).

Exercice. Supposons que $\mathbf{X} = (X_1, \dots, X_p)$ est que les variables sont indépendantes de loi uniforme sur l'intervalle $[-1, 1]$. Calculez $\text{Var}(\langle \mathbf{X}, \theta \rangle)$. Expliquez le lien entre ce calcul et l'erreur d'estimation dans un cadre bien choisi.

En résumé simplifié, l'erreur se divise en deux, l'une d'estimation, l'autre d'approximation. “Bien apprendre”, c'est savoir trouver un bon compromis entre ces deux erreurs. On parle ainsi de *compromis biais-variance*.

2.2.8 Estimation du risque

Reprendons : comme indiqué plus haut, pour une fonction f donnée, on a par la loi des grands nombres

$$R_f = \lim_{N \rightarrow +\infty} \frac{1}{N} \sum_{k=1}^N \ell(Y_k, f(X_k))$$

où $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ est une suite de v.a. i.i.d. de même loi que (\mathbf{X}, Y) . Lorsque f est remplacé par \hat{f}_n , on retrouve la question suivante déjà évoquée plus haut :

- **Problème :** \hat{f}_n est aléatoire puisqu'elle est construite à partir de l'échantillon.
- **Conséquence :** Deux niveaux de “moyennisation”.
- **Conséquence sur l'estimation du risque :** Division de l'échantillon en deux parties (a minima), l'une pour construire \hat{f}_n (échantillon d'apprentissage, *train* en anglais), l'autre pour calculer une approximation de l'espérance (échantillon **test** ou de **validation** selon la situation²).

Divisons l'échantillon en deux parties. Notons \mathcal{D}_{n_1} la partie consacrée à l'apprentissage (train) et $\tilde{\mathcal{D}}_{n_2}$ la partie consacrée au test (On la note $\tilde{\mathcal{D}}$ pour éviter les confusions). L'erreur “test” (ou de validation si l'on effectue une sélection de modèle, cf suite) est alors la quantité

$$\frac{1}{n_2} \sum_{i=1}^{n_2} \ell(\tilde{Y}_i, \hat{f}_{n_1}(\tilde{X}_i))$$

A échantillon \mathcal{D}_{n_1} fixé, on a alors par la LGN (sous des hypothèses appropriées) :

$$\frac{1}{n_2} \sum_{i=1}^{n_2} \ell(\tilde{Y}_i, \hat{f}_{n_1}(\tilde{X}_i)) \xrightarrow{n_2 \rightarrow +\infty} \Phi(\mathcal{D}_{n_1}) = \mathbb{E}[\ell(Y, \hat{f}_{n_1}(\mathbf{X})) | \mathcal{D}_{n_1}],$$

i.e. le risque conditionnellement à l'information contenue dans l'échantillon \mathcal{D}_{n_1} . Notons ici que pour montrer ce résultat, on a utilisé que par définition, \hat{f}_{n_1} est une fonction mesurable de \mathcal{D}_{n_1} . De manière équivalente lorsque les espaces sont discrets, cela peut s'interpréter comme la loi des grands nombres classique (et non conditionnelle) sous la probabilité conditionnelle $\mathbb{P}(\cdot | \{\mathbf{X}_1 = x_1, Y_1 = y_1, \dots, \mathbf{X}_{n_1} = x_{n_1}, Y_{n_1} = y_{n_1}\})$.

Dépendance à l'échantillon d'apprentissage

- Dans ce qui précède, on constate que le risque est “conditionnel à l'échantillon d'apprentissage”. Il est donc “biaisé”. Une manière de réduire ce biais relatif à la base d'apprentissage sera de faire de la validation croisée (voir plus loin).

²Par “situation”, on évoque ici les deux cas suivants : soit on a construit un prédicteur de manière définitive et on souhaite estimer son risque, on parle alors d'échantillon test...), soit on est dans une démarche de recherche du meilleur modèle et/ou des meilleurs paramètres et dans ce cas, on parlera plutôt de validation.

- Néanmoins, que se passe-t-il quand n_1 est grand ? Difficile de donner une réponse générale. De façon intuitive, on peut supposer que l'échantillon devient de plus en plus représentatif de la loi de (\mathbf{X}, Y) de sorte que l'espérance conditionnelle est assez peu sensible à \mathcal{D}_{n_1} . Plus précisément,
- Faisons par exemple l'hypothèse que \mathbf{X} et Y sont liés par la relation

$$Y = f_\theta(\mathbf{X}) + \varepsilon \quad \theta \text{ inconnu},$$

et que $\hat{f}_n = f_{\hat{\theta}_n}$ (modèle linéaire par exemple). Alors, si $\hat{\theta}_n \rightarrow \theta$ (consistance de l'estimateur), on a (sous des hypothèses de continuité de $\theta \mapsto f_\theta$), $\hat{f}_{n_1} \rightarrow f$. Ainsi, par des théorèmes de convergence (type cv dominée), la quantité converge vers R_{f_θ} (déterministe).

- Ceci est une manière de traduire cette non-dépendance asymptotique à l'échantillon.

2.2.9 Validation croisée

Pour estimer le risque, on utilise de manière plus générale la validation croisée (pour limiter le “biais d’apprentissage”). Comme son nom l’indique, la validation croisée est surtout utilisée pour la partie “validation”, *i.e.* pour le choix du meilleur modèle.

Il s’agit de diviser l’échantillon en K parties (K folds) de même taille **aléatoirement** notées I_1, \dots, I_K puis

- utiliser successivement chaque échantillon $\mathcal{D}_{-I_k} = \{(\mathbf{x}_i, y_i), i \in \{1, \dots, n\} \setminus I_k\}$, comme échantillon d’entraînement puis \mathcal{D}_{I_k} comme échantillon de validation.
- Calculer le risque sur chaque sous-échantillon.
- Le risque obtenu par validation croisée notée R_{CV} est alors obtenu comme une moyenne de tous ces risques.

Remarque 1 : Par exemple, dans le cas $K = 2$, on coupe l’échantillon en 2 comme précédemment et on fait la moyenne du risque en utilisant la première puis la seconde partie de l’échantillon comme échantillon d’entraînement. Schématiquement, dans le cas $K = 5$ (K -folds cross-validation),

Iteration 1	Test	Train	Train	Train	Train
Iteration 2	Train	Test	Train	Train	Train
Iteration 3	Train	Train	Test	Train	Train
Iteration 4	Train	Train	Train	Test	Train
Iteration 5	Train	Train	Train	Train	Test

Plus précisément, notons \hat{f}_{-I_k} , l’algorithme de prévision associé à \mathcal{D}_{-I_k} . On a

$$\hat{R}_{CV} = \frac{1}{K} \sum_{k=1}^K \frac{1}{|I_k|} \sum_{i \in I_k} \ell(y_i, \hat{f}_{-I_k}(\mathbf{x}_i)).$$

Remarques :

- Méthode générale pouvant s’appliquer dans la plupart des contextes.
- Peu de résultats théoriques sur le sujet malgré une utilisation très répandue.

- Intuitivement, l'idée est de **réduire la dépendance à l'échantillon d'entraînement** en moyennisant sur K échantillons d'entraînement différents.
- Très important quand l'échantillon est petit.
- $K = 5$ ou $K = 10$ sont des choix usuels (lorsque K est trop important, coût de calcul souvent élevé)
- Le cas limite est le “Leave-One Out”, cas où $K = n$. (A chaque itération, l'échantillon d'entraînement est constitué de $n - 1$ individus).

Surapprentissage et nécessité de diviser en 3

Question : Supposons que l'on ait sélectionné un modèle à l'aide de l'une des méthodes précédentes. L'erreur \hat{R}_{CV} est-elle une bonne approximation du vrai risque ?

Réponse : Pas si clair. On peut encore avoir sur-estimation ou sous-estimation du risque selon la flexibilité du modèle et le nombre de données. De manière plus précise, les échantillons de validation rentrent de près ou de loin dans le choix du modèle puisque l'on choisit le meilleur dans une famille de possibles. Ce phénomène est d'ailleurs amplifié si l'on empile plusieurs méthodes. Supposons par exemple que l'on *agrège* les deux meilleurs modèles de deux familles différentes et que l'on cherche la “meilleure” agrégation. Dans ce cas, les échantillons de validation servent à la fabrication de l'algorithme final.

Une autre erreur usuelle apparaît lorsque l'on fait de la sélection de variables. Imaginons que l'on fabrique un premier prédicteur et que l'on décide de réduire le nombre de variables en ne sélectionnant que les variables les plus importantes (penser par exemple à la régression linéaire). Une fois ces variables sélectionnées, on peut à nouveau recalculer le meilleur prédicteur dans la famille de modèles considérés avec pour critère de décision, l'erreur de validation croisée et dans certains cas, constater que l'erreur a diminué (si le nombre de variables p est important relativement au nombre d'observations, ce phénomène risque de se produire). On peut réitérer ce processus plusieurs fois et choisir à la fin celui qui minimise l'erreur de validation croisée. Néanmoins, dans ce cas, il est hautement probable que l'erreur de validation croisée sous-estime nettement le risque réel. Pour éviter ce phénomène, la marche à suivre est la suivante :

Règle à suivre :

- Laisser une (petite) partie de l'échantillon en dehors du processus de choix de modèle (Echantillon “Test”).
- Sélectionner via une ou plusieurs méthodes le ou les meilleurs algorithmes de différentes familles sur les échantillons “Train/Validation”
- Comparer les algorithmes sur l'échantillon “vierge” à la fin du processus.

Remarque : Exemple du Kaggle (<http://gregpark.io/blog/Kaggle-Psychopathy-Postmortem/>)

Conclusions/Extensions

- Chapitre dédié aux principales notions liées à l'apprentissage statistique.
- Estimation du risque = Opération délicate.
- Peu de règles générales ; Conclusions “universelles” à envisager avec prudence.
- Présentation des méthodes de sélection de modèles via l'approche “minimisation empirique du risque”. Néanmoins, pour certaines familles de modèles il existe des méthodes plus élaborées, basées sur des arguments théoriques (type AIC par exemple). Par ailleurs, dans certains problèmes, la minimisation du risque de prédiction n'est pas la seule question fondamentale d'où l'utilisation d'autres outils de mesure (Courbe ROC par exemple, F_1 -score,... voir TD).

Chapitre 3

Fléau de la dimension

Bien que ce cours soit consacré à l'apprentissage de manière générale et pas seulement à “la grande dimension”, nous choisissons ici de consacrer dès maintenant un chapitre à cet aspect afin de le conserver à l'esprit dans toute la suite du cours.

3.1 Qu'est-ce que la grande dimension ?

Pour le mathématicien, le mot “grande dimension” fait surtout référence au nombre important de variables p que peut contenir le problème car la qualité de l'apprentissage en dépend fortement. Plus précisément, la dimension implique des spécificités statistiques dans la manière d'appréhender le problème. Le but de ce chapitre est d'en présenter quelques aspects.

3.1.1 Exemples

Ci-dessous, quelques situations typiques où la grande dimension apparaît de manière naturelle.

Données biotech: mesure des milliers de quantités par “individu”. On peut penser en particulier aux données “omiques” où l'on cherche à appréhender une maladie par exemple via un ensemble d'informations par individu de l'ordre de plusieurs dizaines à centaines de milliers de variables (cf Figure 3.1 issu de <http://www.ipubli.inserm.fr/>)

Traitement d'images: images médicales, astrophysique, video surveillance, etc. Chaque image est constituée de milliers ou millions de pixels ou voxels.

Marketing : les sites web et les programmes de fidélité collectent de grandes quantités d'information sur les préférences et comportements des clients. Ex: systèmes de recommandation...

Business : l'exploitation des données internes et externes de l'entreprise devient primordiale

3.1.2 A propos de n et p

- Dans tous ces champs d'applications, l'objectif est de décrire de manière plus précise un phénomène (penser au traitement d'images ou à la médecine par exemple). Néanmoins, sous quelles conditions l'accumulation de données peut permettre d'aller dans cette direction ?
- Quantités fondamentales : n le nombre d'observations/individus (patients/images/“clics”...) et p , le nombre de variables/mesures par “individu” (éviter le terme “données” qui peut porter à confusion).
- On parle de Grande Dimension lorsque p est grand, *i.e.* lorsque l'espace sur lequel “vivent” les observations est de grande dimension.

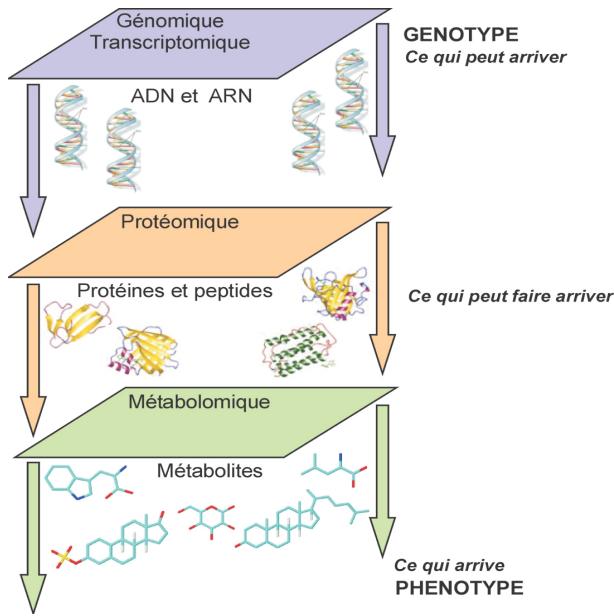


Figure 3.1: Données omiques

- Du point de vue informatique, le caractère n grand est lui AUSSI un facteur de “Big Data” (puisque le temps de calcul s’en trouve impacté) mais on comprendra que du point de vue mathématique, “plus n est grand, plus on est content”.
- Plusieurs situations : en médecine, $n \ll p$, en traitement d’images récoltées sur Internet (cf Facebook par exemple), on peut imaginer n et p grands.
- Ces deux types de situations sont totalement différents. En particulier, les objectifs d’apprentissage sont totalement dépendants du lien entre ces deux valeurs.

3.2 De la statistique classique à la statistique en grande dimension

3.2.1 n, p et statistique

Statistique Classique = p petit et n grand. Dans ce cas, on peut étudier le problème via les résultats classiques type TCL. Soit $f : \mathbb{R}^p \rightarrow \mathbb{R}$, (X_n) suite de variables *i.i.d.*

$$\sqrt{n} \frac{\left(\frac{1}{n} \sum_{i=1}^n f(X_i) - \mathbb{E}[f(X_1)] \right)}{\sqrt{\text{Var}(f(X_1))}} \xrightarrow{n \rightarrow +\infty} \mathcal{N}(0, 1),$$

ou en version L^2 ,

$$\text{Var}\left(\frac{1}{n} \sum_{i=1}^n f(X_i)\right) = \frac{\text{Var}(f(X_1))}{n}.$$

Lorsque p est fixé et que $n \rightarrow +\infty$, la moyenne empirique se concentre sur la moyenne théorique.

- Le problème est que $\text{Var}(f(X_1))$ augmente avec p . Supposons que f soit Lipschitzienne. Dans ce cas, on a

$$\text{Var}(f(X_1)) = \mathbb{E}[(f(X_1) - \mathbb{E}[f(X_1)])^2] \leq C \mathbb{E}[\|X_1\|^2] = C \sum_{i=1}^p \mathbb{E}[(X_1^i)^2] \propto p,$$

si les coordonnées ont la même loi par exemple. Ce qui précède n'est pas tout à fait convaincant car la justification de la croissance en p nécessiterait en toute rigueur une égalité ou une minoration. Néanmoins, on peut noter que si $f = \text{Id}$ et que X_1 est une variable à valeurs dans \mathbb{R}^p centrée, alors, on a bien une égalité.

- Dans un cadre général, le ratio $\sqrt{\frac{p}{n}}$ est donc fondamental dans le calcul de l'erreur.

Statistique en grande dimension : $n << p$ ou $n \propto p$. De manière générale, il faut repenser la statistique dans un mode non asymptotique en n (Chebyshev/Concentration) mais aussi et surtout adapter les objectifs.

- Par exemple, travailler sur un sous-espace/une sous-variété de \mathbb{R}^p qui soit de dimension adaptée au nombre d'observations (LASSO).
- Utiliser des méthodes de classification qui “supportent la dimension” (arbres/SVM...)
- Réduire la dimension du problème...

3.3 Fléau de la dimension (Curse of dimensionality)

3.3.1 Quelques exemples de difficultés liées à la grande dimension

Grande Dimension et Méthodes à moyennage local :

On a vu dans le chapitre 1 que les méthodes à moyennage local (type KNN) sont des candidats non paramétriques naturels pour l'apprentissage. Qu'en est-il en grande dimension ? **Exemple** : Supposons que l'on ait à apprendre une relation de la forme $Y = f(\mathbf{X}, \varepsilon)$ où \mathbf{X} suit la loi uniforme sur l'hypercube $[0, 1]^p$.

Supposons même pour simplifier que $Y = f(\mathbf{X})$ (*i.e.* que sachant \mathbf{X} la réponse est déterministe).

Supposons même pour simplifier encore plus que l'on subdivise chaque dimension en 10 (sur chaque dimension, on divise l'intervalle $[0, 1]$ en 10 intervalles) et que f est constante sur les hypercubes de la forme

$$\prod_{k=1}^p \left[\frac{i_k}{10}, \frac{i_{k+1}}{10} \right].$$

Dans ce cas, le nombre minimal d'observations pour apprendre la relation $Y = f(\mathbf{X})$ est égal à

$$10^p!! \quad (\text{Croissance exponentielle avec la dimension}).$$

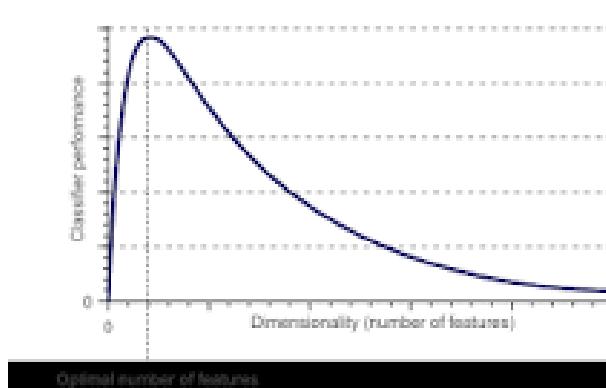
Boule unité en grande dimension :

En langage plus élaboré, la propriété ci-dessus peut s'interpréter comme la décroissance exponentielle du volume de la boule unité avec la dimension. On a le résultat suivant :

Proposition 3.3.1.

$$V_p(1) = \text{Vol}(B_p(0, 1)) = \frac{\pi^{\frac{p}{2}}}{\Gamma(\frac{p}{2} + 1)} \underset{p \rightarrow +\infty}{\sim} \left(\frac{2\pi e}{p} \right)^{\frac{p}{2}} (p\pi)^{-\frac{1}{2}}.$$

La proposition 3.3.1 est illustrée par la figure 3.2.

Figure 3.2: $p \rightarrow \text{Vol}(B_p(0, 1))$

Boule unité en grande dimension et nombre d'observations

La proposition 3.3.1 signifie que l'on a besoin d'un nombre exponentiel (en p) de boules de rayon 1 pour remplir l'hypercube $[0, 1]^p$. Ainsi, supposons que $x^{(1)}, \dots, x^{(n)}$ soient n observations telles que pour tout $x \in [0, 1]^p$, il existe au moins un point parmi les n tel que $\|x^{(i)} - x\| \leq 1$. Autrement dit, supposons que $x^{(1)}, \dots, x^{(n)}$ soient telles que

$$[0, 1]^p \subset \bigcup_{i=1}^n B(x^{(i)}, 1),$$

alors on a nécessairement $1 \leq nV_p(1)$. On retrouve que le nombre de points nécessaires pour remplir l'hypercube satisfait l'inégalité :

$$n \geq \frac{\Gamma(\frac{p}{2} + 1)}{\pi^{\frac{p}{2}}} \sim \left(\frac{p}{2\pi e}\right)^{\frac{p}{2}} (p\pi)^{\frac{1}{2}}$$

En dimension 100, on trouve déjà $n \geq 42.10^{39}$!!

Répartition de la masse d'une boule en grande dimension

Dans le même sens, on peut remarquer que la boule unité en grande dimension peut s'avérer peu intuitive. Considérons la boule de rayon r et la couronne

$$C_p(r) = \{x, 0.99r \leq \|x\| \leq r\}.$$

On a :

$$\frac{\text{Vol}(C_p(r))}{\text{Vol}(B_p(r))} = 1 - 0.99^p \xrightarrow{p \rightarrow +\infty} 1.$$

Ainsi, très rapidement, la masse de la boule unité se trouve concentrée dans sa “croûte”. La répartition des points dans l'espace est finalement assez surprenante.

3.3.2 Moyennage local ?

1. Au vu de ce qui précède, on ne peut clairement pas envisager d'utiliser des méthodes à moyennage local en grande dimension.
2. Utiliser les k -plus proches voisins n'a donc pas de sens en général car il n'y a plus de voisins en grande dimension.

3. Ceci est confirmé par la borne théorique obtenue par Gadat *et. al.*:

$$\sup_{(\mathbf{X}, Y)} |\mathcal{R}(\hat{f}_n) - \mathcal{R}(f^*)| \leq Cn^{-\frac{1+\alpha}{2+p}}.$$

où $\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\mathbf{X}))]$.

4. Néanmoins, la borne ci-dessus peut être vue comme pessimiste car elle est “universelle” (pour l’ensemble des lois (\mathbf{X}, Y) telles que $x \mapsto \eta(x) := \mathbb{P}(Y = 1 | \mathbf{X} = x)$ varie raisonnablement par exemple).
5. Si la fonction η est très peu sensible ou si le support “réel” de X est de dimension plus faible, cela peut encore fonctionner. C’est ce qu’on peut par exemple observer dans le cas des données MNIST (données images de chiffres manuscrits en $28 \times 28 = 784$ pixels).

3.3.3 Exemple de la régression linéaire

Supposons qu’il existe $\theta^* \in \mathbb{R}^p$ tel que pour tout $i \in \{1, \dots, n\}$,

$$Y_i = \langle \mathbf{x}_i, \theta^* \rangle + \varepsilon_i,$$

où $Y_i \in \mathbb{R}$, $\mathbf{x}_i \in \mathbb{R}^p$ et $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. Posons $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)^T$. L’estimateur des moindres carrés $\hat{\theta}$ satisfait (lorsque $\mathbf{x}^T \mathbf{x}$ est inversible):

$$\hat{\theta} = \operatorname{Argmin}_{\theta \in \mathbb{R}^p} \|Y - \mathbf{x}\theta\|^2 = (\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T Y.$$

De plus, si $\mathbf{x}^T \mathbf{x}$ est inversible (matrice $n \times p$),

$$\mathbb{E}[\|\hat{\theta} - \theta_0\|^2] = \mathbb{E}[\|(\mathbf{x}^T \mathbf{x})^{-1} \mathbf{x}^T \varepsilon\|^2] = \operatorname{Tr}((\mathbf{x}^T \mathbf{x})^{-1}) \sigma^2.$$

Supposons que les colonnes C_1, \dots, C_p de \mathbf{x} soient orthonormales (ce qui implique que $p \leq n$). Dans ce cas,

$$(\mathbf{x}^T \mathbf{x})_{i,j} = \langle C_i, C_j \rangle = \delta_{i,j} \implies \mathbf{x}^T \mathbf{x} = I_p.$$

Par conséquent,

$$\mathbb{E}[\|\hat{\theta} - \theta_0\|^2] = p\sigma^2.$$

On retrouve à nouveau la croissance linéaire de l’erreur de la MSE avec la dimension.

3.3.4 Matrice de covariance

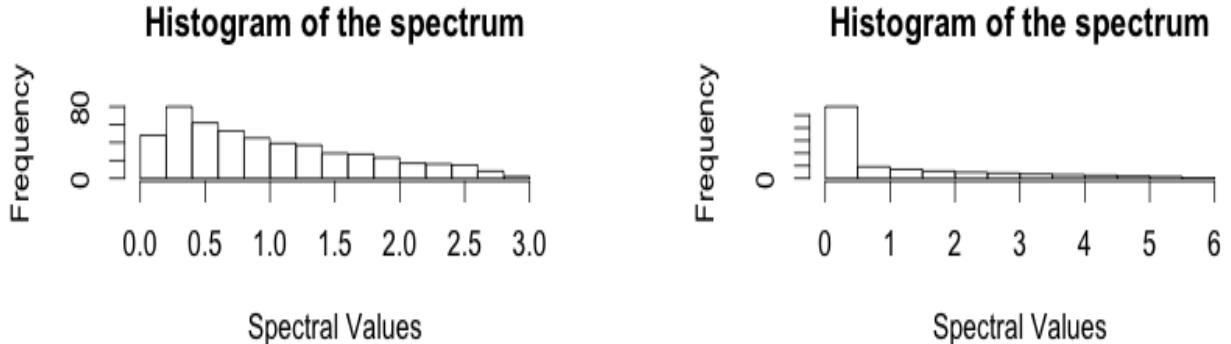
Soit \mathbf{X} une variable aléatoire à valeurs dans \mathbb{R}^p de matrice de covariance Σ (matrice $p \times p$). Pour simplifier, supposons que \mathbf{X} est centrée. Dans ce cas, $\Sigma_{i,j} = \mathbb{E}[\mathbf{X}_i \mathbf{X}_j]$ et notons $(X^{(1)}, \dots, X^{(n)})$ un échantillon *i.i.d.* issu de \mathbf{X} . Soit $\Sigma^{(n)}$ la matrice de covariance empirique associée :

$$\Sigma_{i,j}^{(n)} = \frac{1}{n} \sum_{k=1}^n X_i^{(k)} X_j^{(k)}.$$

Par la loi des grands nombres, si p est fixé et $n \rightarrow +\infty$,

$$\Sigma^{(n)} \rightarrow \Sigma.$$

Question : Si p n’est pas petit devant n , peut-on encore espérer que $\Sigma^{(n)}$ soit une bonne approximation de Σ ?
Réponse : Non, en général. Par exemple, on peut le voir sur le spectre de la matrice de covariance empirique. Dans les graphes de la figure 3.3, on suppose que $\mathbf{X} \sim \mathcal{N}(0, I_p)$. Dans ce cas, l’unique valeur propre de la matrice $\mathbb{E}[\mathbf{X} \mathbf{X}^T]$, *i.e.* de la “vraie” matrice de covariance est 1. Or, on remarque que le spectre de la matrice de covariance empirique ne ressemble pas du tout au spectre de la matrice cible.

Figure 3.3: $n = 1000$, $p = 500$ (left), $p = 2000$ (right)

3.3.5 Et l'ACP ?

On rappelle que l'Analyse en Composantes Principales consiste pour un d fixé à déterminer le meilleur sous-espace vectoriel V_d de dimension d au sens suivant :

$$V_d^{(n)} \in \operatorname{Argmin}_{\dim V \leq d} \sum_{i=1}^n \|X^{(i)} - P_V(X^{(i)})\|^2,$$

où $\| \cdot \|$ désigne la norme euclidienne, avec l'objectif sous-jacent d'estimer le meilleur sous-espace vectoriel au sens suivant :

$$V_d \in \operatorname{Argmin}_{\dim V \leq d} \mathbb{E}[\|\mathbf{X} - P_V(\mathbf{X})\|^2],$$

i.e. du sous-espace vectoriel de dimension d , telle que la projection de \mathbf{X} sur cet espace est la plus proche au sens L^2 de la variable d'origine \mathbf{X} . Via la loi des grands nombres, on peut montrer que " $V_d^{(n)} \rightarrow V_d$ " lorsque $n \rightarrow +\infty$.

Question : L'intérêt de l'ACP est ainsi de réduire "optimalement" la dimension de l'espace mais cette réduction de dimension est-elle fiable statistiquement ? Sous quelles conditions ? On rappelle le résultat suivant (lorsque les variables sont centrées, sinon il faut recentrer).

Théorème 3.3.2. *Supposons que l'échantillon soit issu d'une v.a. X centrée de matrice de covariance $\mathbb{E}[XX^T]$ de rang supérieur à d . Dans ce cas, $V_d^{(n)}$ est le sous-espace vectoriel de dimension d engendré par les d vecteurs propres associés aux plus grandes valeurs propres de $\Sigma^{(n)}$, la matrice de covariance empirique définie par :*

$$\Sigma_{i,j}^{(n)} = \frac{1}{n} \sum_{k=1}^n X_i^{(k)} X_j^{(k)}.$$

Pour rappel, les matrices de covariance (empirique ou non) sont symétriques positives. Ainsi, quitte à normaliser, les vecteurs propres forment une base orthonormée de V_d .

Au vu de ce qu'on a dit sur la matrice de covariance précédemment, on peut donc en conclure l'ACP ne donne pas d'information fiable en grande dimension et en toute généralité, seulement sous la condition $n >> p$.

Néanmoins, si la matrice de covariance est de rang faible ou contient un grand nombre de petites valeurs propres, l'ACP peut s'avérer encore efficace (A voir en pratique).

3.4 Que peut-on espérer en grande dimension ?

Les messages ci-dessous (qui ne sont que des exemples de difficultés rencontrées) ne sont pas très rassurants pour aborder des problèmes en grande dimension. Néanmoins, comme l'indique la suggestion dans le cadre de

l'ACP pour les matrices de faible rang, on peut espérer tirer de l'information si :

- il existe des structures “cachées” dans les espaces de grande dimension qui sont de petite dimension: les données en grande dimension sont concentrées autour de structures de faible dimension reflétant la faible complexité des données. On peut penser à la structure géométrique des images par exemple. En médecine, ce type d'hypothèse n'est pas clair pour certaines maladies complexes mais le faible nombre de données nous astreint à une telle hypothèse.
- La structure est plus complexe mais le nombre de données est lui aussi élevé ce qui permet via des algorithmes performants d'aller plus loin dans la flexibilité et donc d'augmenter la prédiction, l'analyse...

Chapitre 4

Arbres Binaires de décision et extensions agrégées : Bagging, Random Forests et Boosting

On s'intéresse dans ce chapitre à une autre famille fondamentale d'algorithmes d'apprentissage : les arbres de décisions et leurs extensions basées sur des agrégations d'arbres rendant les algorithmes plus stables.

Les arbres de décision ont l'avantage de se baser sur une hiérarchisation de l'information très intuitive tout en permettant de produire des résultats très compétitifs (surtout dans leurs versions agrégées).

On s'intéressera d'abord aux arbres CART introduits par Léo Breiman (1984) puis on introduira l'agrégation la plus simple que constitue le “bagging”. On présentera ensuite en détail les forêts aléatoires ou random forests (introduits par Breiman et Cutler) pour enfin terminer par les méthodes de boosting qui constituent une méthode d'agrégation applicable aux arbres de décision mais aussi à d'autres classes algorithmes.

4.1 Arbres binaires de décision

4.1.1 Construction d'un arbre binaire

La segmentation par arbre est une approche non-paramétrique de la classification ou de la régression.

- But : expliquer une variable réponse (qualitative ou quantitative) à l'aide d'autres variables.
- Principe : construire un arbre à l'aide de divisions successives de l'échantillon en deux segments (appelés aussi **noeuds**) homogènes par rapport à la réponse Y (qualitative ou quantitative) en utilisant l'information de p variables X_1, \dots, X_p (qualitatives ou quantitatives, ordonnables ou non).

4.1.2 Forme générale du résultat

L'arbre ainsi construit est structuré comme suit :

- Racine comportant l'échantillon complet.
- Branches ou segments divisant successivement les sous-échantillons.
- Feuilles ou Noeuds terminaux formant une partition de l'échantillon.

4.1.3 Un premier exemple

- Variables : Age, Revenu et Sexe
- Chaque division/coupe sépare chaque noeud en deux noeuds fils *plus homogènes que le noeud père*
- Homogènes en quel sens ? Définir un critère... (selon le type de réponse)

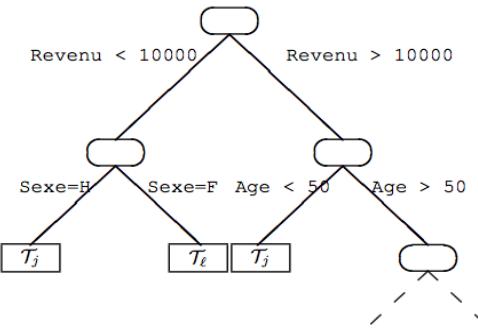


FIGURE 1 – Exemple élémentaire d’arbre de classification.

4.1.4 Partitionnement de l'espace

- Les feuilles de l’arbre (noeuds terminaux) forment une partition de l’espace.
- Cette partition va constituer la règle de classification ou de régression.
- Si X est quantitatif, les feuilles sont représentables via un partitionnement dyadique de l’espace (représentant l’échantillon global) :

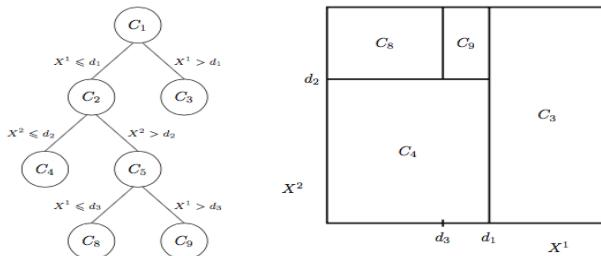
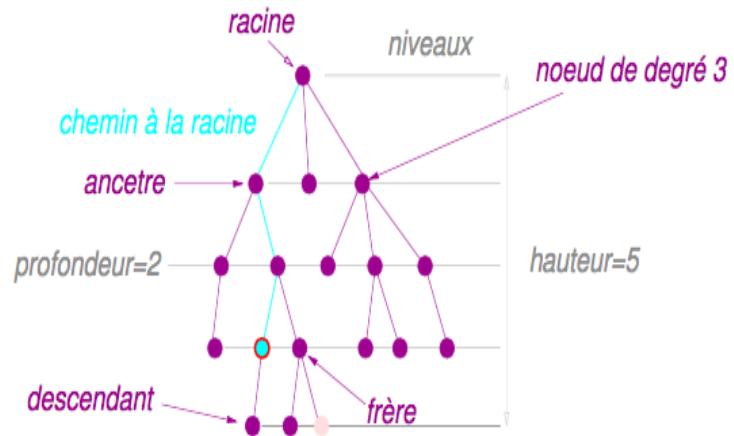


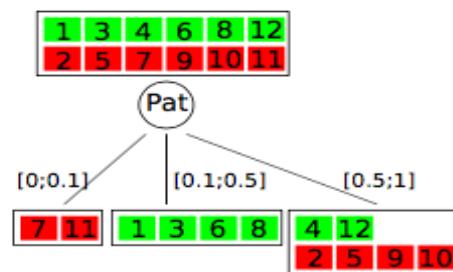
FIGURE 2 – Construction d’un arbre avec variables explicatives quantitatives et pavage dyadique de l’espace. Chaque noeud père engendre deux fils conséquence d’une division ou coupe définie par le choix d’une variable : X^1 ou X^2 et d’une valeur seuil, successivement d_1, d_2, d_3 . À chaque pavé de l’espace ainsi découpé, est finalement associée une valeur ou une classe de Y .

4.1.5 Un exemple

But : expliquer la variable *Wai* (pour “wait”) (ou déterminer un prédicteur de *Wai*).



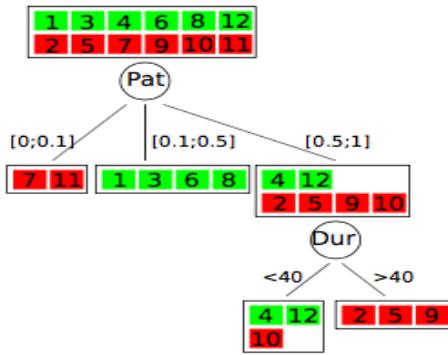
	Alt	Bar	F/S	Hun	Pat	Pri	Rai	Res	Typ	Dur	Wai
x_1	Y	N	N	Y	0.38	\$\$\$	N	Y	French	8	Y
x_2	Y	N	N	Y	0.83	\$	N	N	Thai	41	N
x_3	N	Y	N	N	0.12	\$	N	N	Burger	4	Y
x_4	Y	N	Y	Y	0.75	\$	Y	N	Thai	12	Y
x_5	Y	N	Y	N	0.91	\$\$\$	N	Y	French	75	N
x_6	N	Y	N	Y	0.34	\$\$	Y	Y	Italian	8	Y
x_7	N	Y	N	N	0.09	\$	Y	N	Burger	7	N
x_8	N	N	N	Y	0.15	\$\$	Y	Y	Thai	10	Y
x_9	N	Y	Y	N	0.84	\$	Y	N	Burger	80	N
x_{10}	Y	Y	Y	Y	0.78	\$\$\$	N	Y	Italian	25	N
x_{11}	N	N	N	N	0.05	\$	N	N	Thai	3	N
x_{12}	Y	Y	Y	Y	0.89	\$	N	N	Burger	38	Y



4.1.6 Un exemple

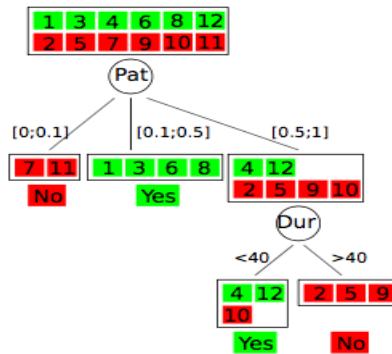
On choisit de séparer les données via la variable Pat en premier lieu (dans cet exemple, on autorise la division en 3, dans la suite, il n'y aura que des arbres binaires).

4.1.7 Un exemple



Les deux noeuds de niveau 1 à gauche ne sont plus divisibles. On continue sur le groupe de droite. On arrête la division à l'étape suivante (même s'il reste un noeud non *homogène*).

4.1.8 Un exemple



Règle de décision : on affecte la réponse “yes” si le noeud terminal (feuille) contient une proportion plus importante de “yes”, “no” sinon. On a fabriqué un prédicteur !

4.1.9 Remarques/questions avant une construction effective

- Une fois l'arbre construit, on affecte à chaque sous-groupe (feuille) une classe si Y qualitatif : la plus représentée dans la feuille si on se base sur l'erreur de classification usuelle, une valeur Y quelconque si Y quantitatif : par exemple, la moyenne des Y_i appartenant à la feuille semble un choix naturel. Formellement, si $(R_t)_t$ désigne les régions de \mathbb{R}^p relatives à la partition finale, la règle de décision s'écrit :

$$\hat{f}(x) = \sum_t \hat{y}_t 1_{x \in R_t}.$$

- Doit-on utiliser toutes les variables explicatives ? Non en général . . . (le surapprentissage guette. . .). N.B. On peut utiliser plusieurs fois la même variable (ce qui permet de comprendre la limitation au cas binaire).
- En pratique, l'ensemble des arbres binaires constructibles à partir de p variables croît (au moins) exponentiellement avec p . Il faut donc trouver une stratégie raisonnable (en coût de calcul) permettant de choisir un “bon” arbre au sens usuel du terme (*i.e.* avec un bon compromis biais/variance).

4.1.10 Méthodes CART/C4.5

- CART : Classification and Regression Tree, introduit par Breiman (1984).
- C4.5 : Quinlan (1993), même principe de construction mais basé sur un critère différent.
- Principe général : (X_1, \dots, X_p) l'ensemble des variables quantitatives ou qualitatives explicatives, Y réponse quantitative ou qualitative,
- 1 - Construire un arbre maximal noté A_{\max} .
- 2 - Ordonner les sous-arbres selon une séquence emboîtée suivant la décroissance d'un critère pénalisé de déviance ou de taux de mal-classés.
- 3 - Sélectionner “le” sous-arbre optimal : c'est la procédure d'*élagage* (Pruning).

4.1.11 Construction d'un arbre binaire maximal

- Critère de division : basé sur la définition d'une fonction d'hétérogénéité ou de désordre.
- L'hétérogénéité d'un noeud t se mesure par une fonction positive notée Q_t qui doit être
 - nulle si le noeud est homogène : tous les individus appartiennent à la même modalité ou prennent la même valeur de Y .
 - “maximale” lorsque les valeurs de Y sont équiprobables ou très dispersées.
- Admissibilité : Une division est dite admissible si aucun des noeuds descendants n'est vide, *i.e.* si la règle de division n'a pas conduit à classer tous les individus dans un même noeud fils.
- La croissance de l'arbre maximal s'arrête à un noeud donné, qui devient donc terminal selon un critère d'homogénéité : soit il est parfaitement homogène, soit on fixe un seuil, sur la non-homogénéité ou sur le nombre d'individus (ex: moins de 2 individus dans le noeud), sous lequel on arrête la division.

4.1.12 Critère d'homogénéité et Règle de division

A chaque noeud, la règle consiste à choisir la variable et le choix du découpage qui fait le plus diminuer l'hétérogénéité. Notons \mathcal{D} l'ensemble des divisions admissibles. Notons le noeud père t ,

- t_l et t_r , ses noeuds fils,
- Q_t l'hétérogénéité au noeud t , N_t le nombre d'individus au noeud t
- pour une division admissible $d \in \mathcal{D}$, $Q_{t_l}(d)$ et $Q_{t_r}(d)$ les hétérogénéités associées.

La division se fait alors en déterminant

$$\text{Argmax}_{d \in \mathcal{D}} \left(Q_t - \frac{N_{t_l}(d)}{N_t} Q_{t_l}(d) - \frac{N_{t_r}(d)}{N_t} Q_{t_r}(d) \right)$$

ou de manière équivalente

$$\text{Argmin}_{d \in \mathcal{D}} (N_{t_l}(d)Q_{t_l}(d) + N_{t_r}(d)Q_{t_r}(d)).$$

4.1.13 A propos des divisions

- dans le cadre quantitatif, *i.e.* ordonné, continu ou discret à valeurs dans \mathcal{X} (\mathbb{R} si continu), l'ensemble des divisions possibles est :
$$\mathcal{D} = \{\{X_j \leq s\}, \{X_j > s\}, j \in \{1, \dots, p\}, s \in \mathcal{X}\}.$$
- dans le cadre qualitatif, on a a priori $\frac{1}{2}(2^{m_j} - 2) = 2^{m_j-1} - 1$ choix par variable j à valeurs dans un ensemble à m_j éléments (Le nombre de partitions en deux parties d'un ensemble à m éléments est égal au nombre de parties non triviales de cet ensemble divisé par 2).
- Exemple : si X_j est à valeurs dans A, B, C , alors, on peut tester A vs $\{B, C\}$, B vs $\{A, C\}$, et C vs $\{A, B\}$ (soit $2^2 - 1$ choix).
- Attention au surapprentissage : une variable catégorielle (qualitative) avec un trop grand nombre de modalités peut générer du sur-apprentissage. On pourra réduire le nombre de modalités.
- Comme toute variable catégorielle, celle-ci peut être recodée en variable numérique binaire en considérant les m variables $\tilde{X}^i = 1_{X=i}$ (à supposer que l'ensemble des modalités soit $\{1, \dots, m\}$).

4.1.14 Q_t pour la classification

Il existe plusieurs choix possibles. On suppose que Y est à valeurs dans $\{1, \dots, m\}$ et on note pour un noeud fixé t , $\hat{p}_{t,k}$ la proportion d'individus de type k au noeud t .

1. Erreur de classification : si on affecte une classe au noeud t , alors elle correspondra à celle qui est la plus représentée. Notons-la

$$Q_t = 1 - \hat{p}_{t,k(t)}$$

où $k(t) = \text{Argmax}_k \hat{p}_{t,k}$. Excepté le cas de classification binaire, ce critère n'est pas compétitif car il ne mesure pas l'hétérogénéité (voir TD). On lui préfère l'indice de Gini (CART) ou l'entropie de Shannon (C4.5) :

2. Indice de Gini :

$$Q_t = \sum_{k \neq k'} \hat{p}_{t,k} \hat{p}_{t,k'} = \sum_{k=1}^m \hat{p}_{t,k} (1 - \hat{p}_{t,k}).$$

3. Entropie de Shannon.

$$Q_t = - \sum_{k=1}^m \hat{p}_{t,k} \log \hat{p}_{t,k}.$$

(voir TD pour interprétation de Gini et entropie).

4.1.15 Q_t pour la régression

Supposons Y quantitative. Dans ce cas, le choix usuel consiste à minimiser la variance intra-noeuds : notons \mathcal{I}_t l'ensemble des individus au noeud t .

$$V(t) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} (y_i - \bar{y}_t)^2$$

où

$$\bar{y}_t = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} y_i.$$

Remarque: Dans les différents cas, la décroissance de l'hétérogénéité est une quantité visualisable sur le graphe via la longueur du segment qui relie le noeud père au noeuds fils.

4.1.16 Elagage

Pour un arbre A , on note \mathcal{F}_A l'ensemble des feuilles (noeuds terminaux) et K_A le nombre de feuilles. La qualité d'ajustement de A est définie par :

$$Q(A) = \sum_{t \in \mathcal{F}_A} N_t Q_t(A).$$

A nouveau Q_t désigne une mesure de l'hétérogénéité (erreur de classif., Gini, entropie, variance intra-noeuds, . . .).

Proposition 4.1.1. *Pour tout sous-arbre A de A_{\max} (construit via la règle de division) satisfait $Q(A) \geq Q(A_{\max})$. En d'autres termes, la quantité $Q(A)$ décroît avec les divisions.*

Afin de pouvoir sélectionner un modèle en évitant le surapprentissage, on pénalise cette quantité par le nombre de feuilles :

$$C_\alpha(A) = Q(A) + \alpha K_A.$$

L'élagage est alors possible.

4.1.17 Comment élaguer ?

Etape 1 : Sélection de sous-arbres emboités. Il s'agit d'une procédure itérative où à la fois

1. La taille de l'arbre va réduire jusqu'à retour à l'état racine.
2. Le paramètre α va être calibré à chaque itération.

On initialise à $A_0 = A_{\max}$ et $\alpha_0 = 0$ puis à chaque itération :

1. On pose

$$\alpha_k = \min_{t \text{ noeud de } A_k} \frac{Q_t - Q(A_k(t))}{|A_k(t)| - 1}$$

où $A_k(t)$ désigne le sous-arbre de A_{k-1} issu de t . Le(s) noeud(s) qui minimise(nt) le critère ci-dessus satisfait(ont) par construction $C_{\alpha_k}(t) = C_{\alpha_k}(A_k(t))$.

2. On élague les branches issues du noeud t minimiseur.

On a ainsi fabriqué une suite de sous-arbres candidats à être optimaux.

Etape 2 : Parmi ces candidats, on choisit le meilleur par estimation de l'erreur de prédiction sur un échantillon test ou par validation croisée.

4.1.18 Avantages/Inconvénients

1. Avantages :

- Non-paramétrique, capable de gérer tous types de variables
- D'interprétation simple et visualisable.

2. Inconvénients :

- Pas de performances garanties théoriquement.
- Manque de régularité dans les résultats dans le cas de la régression.
- Instabilité importante lorsque la complexité du problème augmente

3. Solutions/Améliorations pour rendre cette approche plus robuste.

- Boosting/Bagging
- Random Forests
- XGBoost

4.2 Bagging, Random Forests, Boosting

4.2.1 Agrégation de modèles

Les méthodes décrites ci-après entrent dans la famille des méthodes d'agrégation de modèles. Qu'est-ce que l'agrégation de modèles ? Comme son nom l'indique, l'idée est d'agréger, *i.e.* de combiner, des modèles/algorithmes afin d'en augmenter les qualités prédictives. Dans la suite, nous présentons 3 méthodes d'agrégation dans le cadre des arbres de décision. Les idées relatives au boosting et au bagging (et même celles de la construction des random forests) sont néanmoins transposables à d'autres problèmes.

4.2.2 Bootstrap

Les méthodes ci-après font usage de la méthode dite de *bootstrap*, qui consiste à faire usage de sous-échantillons tirés aléatoirement et avec remise de l'échantillon complet (introduit par Efron au début des années 80) :

- Soit $(Z_1, \dots, Z_n)_n$ un échantillon.
- (i_1, \dots, i_k) une suite d'indices tirés aléatoirement avec remise dans $\{1, \dots, n\}$.
- $(Z_{i_1}, \dots, Z_{i_k})$ est un échantillon bootstrap issu de $(Z_1, \dots, Z_n)_n$ (on peut choisir $k = n$).
- On le note plus simplement $(Z_1^{*,b}, \dots, Z_k^{*,b})$, $b \in \{1, \dots, B\}$ où B le nombre d'échantillons bootstrap.
- Supposons par exemple que l'on cherche à estimer un paramètre θ . Chaque échantillon produit un estimateur $\hat{\theta}^*(b)$.
- La quantité $\hat{\theta}^* = \frac{1}{B} \sum_{b=1}^B \hat{\theta}^*(b)$ fournit alors un nouvel estimateur de θ qui apporte de l'information. Par exemple,
- La variance empirique bootstrap

$$\hat{\sigma}_B^2 = \frac{1}{B-1} \sum_{b=1}^B (\hat{\theta}^*(b) - \hat{\theta}^*)^2$$

fournit un estimateur de la variance.

- Malgré le rééchantillonnage qui implique évidemment l'absence d'indépendance entre les échantillons bootstrap, ce procédé permet de produire une estimation de la variance sans utiliser de nouvelles variables.
- Dans le cadre des arbres, ce procédé va être utilisé pour agréger des arbres et améliorer les performances de classification en réduisant généralement la variance.

4.2.3 Bagging

Il s'agit de la version la plus simple d'agrégation par moyennisation bootstrap.

Principe :

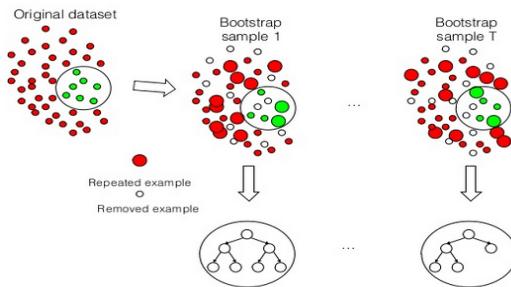
- On tire B échantillons bootstrap issus de l'échantillon (d'entraînement). Chaque échantillon produit un prédicteur $\hat{f}_b(\cdot)$.
- Dans le cadre quantitatif, on note $\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}_b(x)$.
- Dans le cadre qualitatif usuel, on note $\hat{f}_{bag}(x) = \text{Argmax}_{k=1}^m \sum_{b=1}^B 1_{\{\hat{f}_b(x)=k\}}$ (Autrement dit, on procède par vote majoritaire).
- \hat{f}_{bag} est l'estimateur obtenu par bagging.

- L'estimation OOB (Out-Of-Bag) de l'erreur de prédiction est obtenue en calculant pour chaque indice i , la prédiction bootstrap \hat{Y}_i issue de l'ensemble des échantillons bootstrap ne contenant pas l'indice i . L'erreur OOB est alors
 - $\frac{1}{n} \sum_{k=1}^n (Y_k - \hat{Y}_k)^2$ dans le cas quantitatif (pour la fonction de perte usuelle)
 - $\frac{1}{n} \sum_{k=1}^n 1_{Y_k \neq \hat{Y}_k}$ dans le cadre qualitatif (avec la fonction de perte usuelle).

Bagging pour les arbres de décision

Principe/Remarques :

- Construire B arbres de décision issus des B échantillons bootstrap.
- Associer un prédicteur à chaque arbre “bootstrappé”.
- Moyenniser comme présenté dans le slide précédent.
- En général, seule la première phase de CART est conservée : il n'y a pas d'élagage. L'idée est que le surapprentissage potentiel est compensé par la moyennisation bootstrap.
- L'erreur out-of-bag peut être calculée au fil de l'ajout d'échantillons bootstrap mais ce procédé est coûteux.
- Ce principe permet en pratique d'améliorer les performances mais réduit l'interprétabilité (ou le caractère explicatif) puisque la structure d'arbre est remplacée par une moyenne (cette remarque vaut aussi pour les algorithmes présentés dans la suite). On peut néanmoins envisager de quantifier l'importance des variables en comptabilisant leurs (et leurs niveaux) d'apparitions dans chacun des arbres mis en jeu.
- Généralement, le bootstrap fonctionne lorsque les arbres en jeu ne sont pas trop (positivement) corrélés (voir TD).



4.2.4 Forêts aléatoires

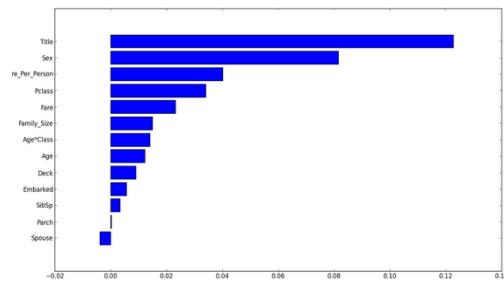
Afin de limiter la corrélation entre les arbres issus du bagging, le principe des forêts aléatoires est de randomiser les variables mises en jeu dans la division et de n'en choisir que m parmi les p disponibles. **Détails :**

- On tire B échantillons bootstrap issus de l'échantillon.
- Pour chaque échantillon bootstrap, on construit l'arbre de décision maximal avec la règle suivante : à chaque noeud, on choisit la meilleure division possible basée sur m variables tirées aléatoirement parmi les p (variantes possibles).
- On tire un prédicteur \hat{f}_b de chaque échantillon bootstrap.
- On conclut comme pour le bagging.

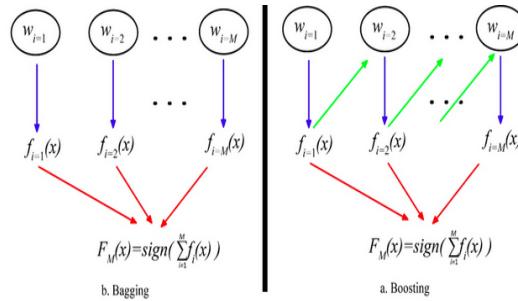
Remarques

- Intérêt : réduire la corrélation entre les arbres : par exemple, si une variable joue un rôle prédictif trop important, elle apparaîtra dans la plupart des arbres baggés dans le haut de l'arbre.
- En pratique $m \approx \sqrt{p}$.
- Méthode très performante, quelques résultats théoriques récents (*e.g.* Arlot).

On peut enfin mesurer l'importance de chaque variable (fonction `importance` dans R, package `randomForest`) afin de “récupérer” un peu d'interprétabilité.



4.2.5 Boosting



A l'origine le boosting a été introduit pour améliorer les performances de faibles classificateurs en les agrégeant intelligemment. Par rapport au bagging qui n'est conçu “que” pour réduire la variance, le boosting a pour objectif de réduire également le biais.

Principe

1. Construction itérative d'une agrégation de modèles.
2. A chaque étape, on augmente le poids des observations mal ajustées/mal prédites.
3. On voit donc a priori le double-effet potentiel : réduction de la variance par agrégation et diminution biais via 2.

Pour l'instant, le propos est vague. Les algorithmes ayant ce point de vue sont nombreux. On se concentre ici sur la présentation de l'algorithme historique : *Adaboost*.

4.2.6 Adaboost

Soit $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ un échantillon d'apprentissage. On note $\mathbf{w} = \{w_i, i = 1, \dots, n\}$ les poids relatifs à chaque observation. On initialise les poids w_i associés à chaque observation à $1/n$:

Algorithme dans le cas de la classification : A l'étape m ,

- On fabrique un prédicteur \hat{f}_m sur le modèle m en minimisant une erreur de prédiction pondérée :

$$\hat{f}_m = \operatorname{Argmin}_{f \in \mathcal{F}} \sum_{i=1}^n w_i 1_{y_i \neq f(x_i)}.$$

- On calcule l'erreur pondérée

$$\mathcal{E}_m = \frac{\sum_{i=1}^n w_i 1_{\{\hat{f}_m(x_i) \neq y_i\}}}{\sum_{i=1}^n w_i}.$$

- On calcule le logit $\alpha_m = \frac{1}{2} \log((1 - \mathcal{E}_m)/\mathcal{E}_m) \vee 0$.

- On actualise les poids de la manière suivante :

$$w_i \leftarrow w_i \exp(\alpha_m 1_{\{\hat{f}_m(x_i) \neq y_i\}}), \dots, i = 1, \dots, n.$$

Le classifieur final est (dans le cas où les classes sont -1 et 1) :

$$\hat{f}(x) = \operatorname{sgn} \left(\sum_{m=1}^M \alpha_m \hat{f}_m(x) \right).$$

Remarque 4.2.1. La règle d'actualisation des poids est fabriquée pour que :

- Le poids soit augmenté/diminué si mal classé/bien classé.
- Après plusieurs itérations, le poids va donc se concentrer sur les observations difficiles à classer.
- Le nouveau modèle joue donc le rôle d'ajusteur de ces “dernières” observations.

Ce principe peut donc s'appliquer à l'agrégation d'arbres. La façon la plus simple d'imaginer les choses est la suivante. On a à disposition un certain nombre d'arbres construits par bootstrap et donc de prédicteurs associés. L'ensemble \mathcal{F} est donc constitué des B prédicteurs \hat{f}_b , $b = 1, \dots, B$. On dispose d'un autre échantillon (idéalement indépendant mais ça n'est pas obligatoire) $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$. On choisit alors successivement les prédicteurs par la méthode décrite ci-dessus.

En réalité, la construction peut être itérative. A chaque itération, on peut, à partir d'un nouvel échantillon bootstrap, fabriquer un modèle qui va fitter les *résidus*. Plus précisément, l'Adaboost peut être vu comme une forme de Gradient Boosting (associé à la fonction de perte exponentielle) et on va justement voir dans la suite que la construction du Gradient Boosting est complètement itérative.

Remarque 4.2.2. Le choix de α_m peut sembler mystérieux. Il est là encore lié à la fonction de perte exponentielle (adaptée à la classification binaire avec classes -1 et 1) (voir TD pour plus de détails).

4.2.7 Gradient Boosting

Dans sa version plus moderne, le boosting s'interprète généralement comme un problème d'optimisation. Considérons une fonction de perte Ψ . Dans le cas quantitatif, on supposera $\Psi(y, y') = (y' - y)^2$. Dans le cas de la classification binaire (avec $\mathcal{Y} = \{-1, 1\}$), on supposera que

$$\Psi(y, F(x)) = e^{-yF(x)} \quad \text{ou} \quad \Psi(y, F(x)) = \log(1 + e^{-yF(x)}),$$

qui correspondent soit à la fonction de perte exponentielle ou à la fonction de perte logit. Dans le cas de la classification, on remarque en particulier que la fonction de perte usuelle (erreur de classification) n'est pas considérée. On lui préfère des fonctions Ψ régulières (pour pouvoir dériver).

Dans le cas multilabels (K classes, incluant le cas $K = 2$!), on travaillera avec la fonction (parfois appelée déviance/deviation dans la littérature),

$$\Psi(y, F(x)) = - \sum_{k=1}^K 1_{y=k} \log(\pi^k(F(x)))$$

où

$$\pi^k(F(x)) = \frac{e^{F(X_i)_k}}{\sum_{j=1}^K e^{F(X_i)_j}}$$

Notez que F est une fonction à valeurs dans \mathbb{R}^K de sorte que $F(x)_k$ désigne sa k -ième coordonnée. Il faut aussi comprendre que cette fonction est une entropie croisée dont on rappelle la définition. Si p et q sont deux probas à valeurs dans $\{1, \dots, K\}$, l'entropie croisée H s'écrit :

$$H(p, q) = - \sum_{k=1}^K p(k) \log(q(k))^1.$$

Ici,

$$\Psi(y, F(x)) = H(\delta_y, \pi(F(x))),$$

c'est-à-dire une sorte de distance entre la loi (triviale) de la réponse et le vecteur de probabilités donné par la fonction F .

Un second point de vue est celui de la log-vraisemblance. Si l'on fait l'hypothèse que $\mathbb{P}(Y = y|X = x) = \pi^F(x)$, alors la log-vraisemblance (conditionnelle) associée à l'échantillon d'apprentissage s'écrit :

$$\begin{aligned} \log(\mathbb{P}(Y_1 = y_1, \dots, Y_N = y_N | X_1 = x_1, \dots, X_N = x_N)) &= \sum_{i=1}^N \log(\mathbb{P}(Y_i = y_i | X_i = x_i)) \\ &= \sum_{i=1}^N \sum_{k=1}^K \log(\pi^k(F(x_i))) 1_{y_i=k} = - \sum_{i=1}^N \Psi(y_i, F(x_i)). \end{aligned}$$

Ainsi, minimiser $\sum_{i=1}^N \Psi(y_i, F(x_i))$ sur l'ensemble des fonctions F revient à maximiser la log-vraisemblance du modèle.

Présentons maintenant l'idée générale du gradient boosting. On a à notre disposition un ensemble de fonctions $f : \mathcal{X} \rightarrow \mathcal{Y}$, disons d'un ensemble de règles de prévisions. On appelle \mathcal{F} l'ensemble de ces fonctions. Notons que dans le cas des arbres, ces fonctions sont de la forme $\sum_{j=1}^J \beta_j 1_{A_j}$ où $(A_j)_{j=1}^J$ désigne un partitionnement de l'espace et J désigne l'ensemble des feuilles.

Pour un échantillon $\{(X_1, Y_1), \dots, (X_n, Y_n)\}$ et une fonction F , on introduit

$$C_n(F) = \frac{1}{n} \sum_{i=1}^n \Psi(Y_i, F(X_i)).$$

Cas quantitatif, fonction de perte “moindres carrés”. On construit dans un premier temps f_1 en cherchant $f_1 \in \text{Argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n (Y_i - f(X_i))^2$. On pose également $F_1 = f_1$ et $\eta_1 = 1$. On cherche ensuite

¹On rappelle que $H(p, q) = H(p) + KL(p||q)$ où KL désigne la divergence de Kullback-Leibler entre p et q (pseudo-distance entre p et q).

$f_2 \in \mathcal{F}$ telle que $C_n(F_1 + \eta f_2)$ est minimal parmi les $\eta \in (0, +\infty)$ et fonctions $f_2 \in \mathcal{F}$. On ne présente ici que le choix de f_2 et on suppose η petit. On a alors :

$$C_n(F_1 + \eta f_2) = \frac{1}{n} \sum_{i=1}^n (Y_i - F_1(X_i))^2 - \frac{2\eta}{n} \sum_{i=1}^n (Y_i - F_1(X_i)) f_2(X_i) + O(\eta^2).$$

En d'autres termes, le développement au premier ordre fait apparaître :

$$\langle Y - F_1, f_2 \rangle$$

où $\langle Y - g, f \rangle = \sum_{i=1}^n Y_i g(X_i), f(X_i) \rangle$. Par un argument de Cauchy-Schwarz, on en déduit qu'il faut choisir $(f_2(X_i))_{i=1}^n$ colinéaire à $(Y_i - F_1(X_i))_{i=1}^n$. Le principe est alors de construire f_2 via un arbre qui a pour but de "fitter" le mieux possible $\{(X_i, Z_i), i = 1, \dots, n\}$ où $Z_i = Y_i - F_1(X_i)$. En d'autres termes, on ajuste donc le nouvel arbre sur les résidus. Une fois choisi le f_2 , on peut ensuite ajuster le η en minimisant la fonction $\eta \mapsto b\eta^2 - 2a\eta$ avec

$$a = \sum_{i=1}^n (Y_i - F_1(X_i)) f_2(X_i) \quad \text{et} \quad b = \sum_{i=1}^n (f_2(X_i))^2.$$

Il suffit de poser

$$\eta_2 = \frac{a}{b} = \frac{\sum_{i=1}^n (Y_i - F_1(X_i)) f_2(X_i)}{\sum_{i=1}^n (f_2(X_i))^2}.$$

On itère ensuite cette construction pour finalement obtenir le prédicteur

$$F = \sum_{t=1}^T \eta_t f_t.$$

Cadre général, cadre de la classification. Si maintenant, on considère

$$C_n(F) = \frac{1}{n} \sum_{i=1}^n \Psi(Y_i, F(X_i))$$

avec Ψ générale, minimiser $C_n(F + \eta f)$ lorsque η est petit parmi les $f \in \mathcal{F}$ consiste à faire une descente de gradient :

$$\Psi(y, F(x) + \eta f(x)) = \Psi(y, F(x)) + \eta \langle \partial_2 \Psi(y, F(x)), f(x) \rangle + O(\eta^2),$$

où $\partial_2 \Psi(y, z) = (\partial_{z_1} \Psi(y, z), \dots, \partial_{z_k} \Psi(y, z))$. Suivant la même idée que celle décrite dans le cas quantitatif, on en déduit qu'idéalement, la fonction f doit vérifier

$$f(X_i) = -\partial_2 \Psi(Y_i, F(X_i)), \quad i = 1, \dots, n$$

pour faire le plus décroître $C_n(F)$ dans son voisinage. Remarquons que dans le cas $\Psi(y, y') = (y - y')^2$, on retrouve $-\partial_2 \Psi(Y_i, F(X_i)) = 2(Y_i - F(X_i))$.

Dans le cas de la classification, on va plutôt considérer le prédicteur de score plutôt que le prédicteur de classe pour pouvoir utiliser une approche plus quantitative. Par exemple, avec la fonction de perte exponentielle,

$$-\partial_2 \Psi(Y_i, F(X_i)) = Y_i e^{-Y_i F(X_i)}.$$

On va donc ajuster un nouvel arbre sur cet échantillon. La difficulté ici est de comprendre ce que signifie ajuster un arbre sur $\{(X_i, Z_i), i = 1, \dots, n\}$ où $Z_i = Y_i e^{-Y_i F(X_i)}$ puisqu'il s'agit de classification. L'idée est donc d'attribuer à chaque observation de l'échantillon bootstrap sélectionné pour la construction de ce nouvel arbre un poids proportionnel à Z_i . On retrouve alors le principe de l'Adaboost. A l'issue de la construction de ce nouvel arbre, on sélectionne le prédicteur (de score) pour l'ajouter au précédent. Finalement,

$$F = \text{sgn} \left(\sum_{t=1}^T \eta_t f_t \right).$$

Terminons le cas multiclasses. Pour $y \in \{1, \dots, K\}$ et $z = (z_1, \dots, z_k)$ appartenant à \mathbb{R}^K , notons :

$$\Psi(y, z) = - \sum_{k=1}^K 1_{y=k} \log \left(\frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}} \right)$$

En remarquant que

$$-\Psi(y, z) = \sum_{k=1}^K 1_{y=k} z_k - \log \left(\sum_{k=1}^K e^{z_k} \right),$$

on en déduit que

$$-\partial_{z_k} \Psi(y, z) = 1_{y=k} - \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}} = 1_{y=k} - \pi^k(z).$$

Ainsi, l'idée est d'utiliser une fonction f satisfaisant

$$(f(X_i))_k = 1_{Y_i=k} - \pi^k(F(X_i)), \quad \forall (k, i) \in \{1, \dots, K\} \times \{1, \dots, N\}.$$

Formellement, on reconnaît une forme de résidu. En pratique, à nouveau, ces résidus vont fournir des poids à chaque nouvel individu. On ajustera ensuite "l'arbre" (voir plus bas) sur le nouveau jeu pondéré par ces poids issus des résidus. L'algorithme est alors le suivant pour un pas fixé η :

- Etape 1 : on initialise F_0 à 0 de sorte que $\pi^k(F_0(x)) = \frac{1}{K}$ pour tous k et x . On fabrique $f_1 = (f_1^1, \dots, f_1^K)$ ajusté sur l'échantillon d'apprentissage $\{(X_i, Z_i), i = 1 \dots, n\}$ où $Z_i = (Z_i^1, \dots, Z_i^K)$ avec $Z_i^k = 1_{Y_i=k} - \frac{1}{K}$. On pose $F_1 = \eta f_1$.
- Etape t : A cette étape, on a $F_t = \sum_{u=1}^t \eta f_u$. On fabrique $f_{t+1} = (f_{t+1}^1, \dots, f_{t+1}^K)$ ajusté sur l'échantillon d'apprentissage $\{(X_i, Z_i), i = 1 \dots, n\}$ où $Z_i = (Z_i^1, \dots, Z_i^K)$ avec $Z_i^k = 1_{Y_i=k} - \pi^k(F_t(X_i))$. On pose $F_{t+1} = F_t + \eta f_{t+1}$.

Le "vecteur-score" final est donc $\pi_T = (\pi^k(F_T(X_i)))_{k=1}^K$ et la classe prédictive par l'algorithme est donc le score maximal.

N.B. On a laissé de côté un point important ici : on a à chaque étape besoin de traiter un problème de régression vectorielle. En effet, la sortie est un vecteur de taille K . En pratique, il semble que dans le cadre des arbres, on traite ce problème à l'aide de K arbres de régression : chaque arbre est ajusté sur l'échantillon d'apprentissage $\{(X_i, Z_i^k), i = 1 \dots, n\}$ où $Z_i^k = 1_{Y_i=k} - \pi^k(F_t(X_i))$. Il pourrait peut-être être envisagé de traiter le problème avec un seul arbre en considérant une variance intra-noeuds vectorielle

$$V(t) = \frac{1}{|\mathcal{I}_t|} \sum_{i \in \mathcal{I}_t} \|y_i - \bar{y}_t\|^2$$

mais cette approche ne semble pas développée actuellement.

On doit enfin terminer ce chapitre en évoquant le **XGBoost** (Extrem Gradient Boosting) qui est une méthode assez récente et donnant généralement d'excellents résultats basée sur le gradient boosting combinée à une pénalisation limitant le surapprentissage. Cet algorithme sera testé en TP.

On pourra trouver une description rapide de XGBoost dans <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-agreg.pdf>

et des détails précis sur les choix de paramètres dans <https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/>.

Le rapport de projet annuel d'Adrien Maitammar (2021-2022) donne également beaucoup d'indications à ce sujet. Vous le trouverez sur Moodle.

Chapitre 5

Régression linéaire en Grande Dimension

5.1 Régression Linéaire Classique : Rappels

5.1.1 Modèle

On considère une variable à expliquer Y (supposée réelle pour l'instant) et p covariables (ou variables explicatives) X_1, \dots, X_p . L'objectif de la *régression linéaire* est d'expliquer Y à l'aide d'une combinaison linéaire bruitée de X_1, \dots, X_p de la forme :

$$Y = \theta_0 + \sum_{i=1}^p \theta_i X_i + \varepsilon$$

où $\theta = (\theta_0, \theta_1, \dots, \theta_p)^T$ appartient à \mathbb{R}^{p+1} et ε est une variable aléatoire centrée (bruit) indépendante de la variable aléatoire X . Quitte à considérer le vecteur $(1, X_1, \dots, X_p)$, on remarque que l'on peut mettre ce modèle sous la forme plus synthétique,

$$Y = \langle X, \theta \rangle + \varepsilon.$$

Dans la suite, on notera simplement $\theta = (\theta_1, \dots, \theta_p)$ et $X = (X_1, \dots, X_p)$. Pour un couple (X, Y) donné, le (un des) but(s) est alors de déterminer :

$$\theta^* = \operatorname{Argmin}_{\theta \in \mathbb{R}^{p+1}} \mathbb{E}[(Y - \langle X, \theta \rangle)^2].$$

5.1.2 Régression linéaire et Apprentissage

Si on fait l'hypothèse que le “vrai” modèle est de la forme

$$Y = \langle X, \theta^* \rangle + \varepsilon, \tag{5.1.1}$$

alors le prédicteur de Bayes (pour la fonction de perte $\ell(y, y') = (y - y')^2$) est donné par

$$f_{\text{Bayes}}(x) = \mathbb{E}[Y|X = x] = \langle x, \theta^* \rangle.$$

Dans ce cas, si $\operatorname{Var}(\varepsilon) = \sigma^2$, alors

$$\inf_{f: \mathbb{R}^p \rightarrow \mathbb{R}} \mathcal{R}_f = \mathbb{E}[(Y - \langle X, \theta^* \rangle)^2] = \mathbb{E}[\varepsilon^2] = \sigma^2.$$

Soit $\mathcal{D}_n = \{(X^{(1)}, Y^{(1)}), \dots, (X^{(n)}, Y^{(n)})\}$ un échantillon d'apprentissage (en particulier *i.i.d.*). On suppose dans la suite que pour $k \in \{1, \dots, n\}$,

$$Y^{(k)} = \langle X^{(k)}, \theta^* \rangle + \varepsilon_k.$$

où θ^* est une quantité “à apprendre”. Sauf mention contraire, on fera aussi l'hypothèse que $\varepsilon_k \sim \mathcal{N}(0, \sigma^2)$.

5.1.3 Estimation de θ^*

Pour estimer θ^* , on utilise l'estimateur des moindres carrés (qui est aussi l'estimateur du maximum de vraisemblance dans le cas où le bruit est gaussien)

$$\hat{\theta}_n = \operatorname{Argmin}_{\theta \in \mathbb{R}^p} \sum_{i=1}^n (Y^{(i)} - \langle X^{(i)}, \theta \rangle)^2 = \operatorname{Argmin}_{\theta \in \mathbb{R}^p} \|\mathbf{Y} - \mathbf{X}\theta\|^2$$

où $\| \cdot \|$ désigne la norme euclidienne,

$$\mathbf{Y} = \begin{pmatrix} Y^{(1)} \\ \dots \\ Y^{(n)} \end{pmatrix} \text{ et } \mathbf{X} = \begin{pmatrix} X_1^{(1)} & \dots & X_p^{(1)} \\ \dots & \dots & \dots \\ X_1^{(n)} & \dots & X_p^{(n)} \end{pmatrix}$$

Attention aux notations : X et Y désignent des variables aléatoires. \mathbf{X} et \mathbf{Y} désignent la matrice de “design” (ou des covariables) et le vecteur des réponses associées à l'échantillon. Dans la suite, ε désigne le vecteur $(\varepsilon_1, \dots, \varepsilon_n)$.

5.1.4 Premières propriétés

A partir de maintenant, on fait l'hypothèse que le vrai modèle s'écrit sous la forme (5.1.1). Ceci signifie que l'on fait l'hypothèse que $\mathcal{L}(Y|X) = \mathcal{N}(\langle X, \theta^* \rangle, \sigma^2)$. Avec les notations précédemment introduites, cela se traduit sur l'échantillon d'observations par l'égalité (dans \mathbb{R}^n):

$$\mathbf{Y} = \mathbf{X}\theta^* + \varepsilon.$$

En particulier, comme $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_n)$, on en déduit que conditionnellement à \mathbf{X} (*i.e.* connaissant \mathbf{X})

$$\mathbf{Y} \sim \mathcal{N}(\mathbf{X}\theta^*, \sigma^2 I_n).$$

Dans la suite, on raisonnera toujours “conditionnellement à \mathbf{X} ” de sorte que \mathbf{X} est vue comme une matrice déterministe.

Dans le modèle linéaire classique, on fait généralement l'hypothèse que $\mathbf{X}^T \mathbf{X}$ est inversible, ce qui revient à dire que $\operatorname{rg}(\mathbf{X}) = p$ et donc que $p \leq n$. Dans ce cadre, on obtient une expression explicite $\hat{\theta}$:

Proposition 5.1.1. *Si $\mathbf{X}^T \mathbf{X}$ est inversible, on a :*

$$\hat{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}.$$

De plus,

$$\hat{\theta} \sim \mathcal{N}(\theta^*, \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1}).$$

Remarque 5.1.2. Pour n observations, le prédicteur associé est défini pour tout $x \in \mathbb{R}^p$ par :

$$\hat{f}_n(x) = x^T \hat{\theta} = \langle x, \hat{\theta} \rangle.$$

Si $\mathbf{X}^T \mathbf{X}$ est inversible alors,

$$x^T \hat{\theta} \sim \mathcal{N}(x^T \theta^*, \sigma^2 x^T (\mathbf{X}^T \mathbf{X})^{-1} x).$$

Propriétés de $\hat{\mathbf{Y}}$: Rappelons dans un premier temps que si l'on note X_i la i -ème colonne de \mathbf{X} et $V_{\mathbf{X}} = \operatorname{Vect}(X_1, \dots, X_p)$, alors la projection orthogonale sur $V_{\mathbf{X}}$ est donnée par : $\forall U \in \mathbb{R}^n$,

$$\operatorname{Proj}_{V_{\mathbf{X}}}(U) = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T U \quad (\text{lorsque } \mathbf{X}^T \mathbf{X} \text{ est inversible}).$$

Dans ce cas, si on note $\hat{\mathbf{Y}} = \mathbf{X}\hat{\theta}$, *i.e.* si $\hat{\mathbf{Y}}$ est le vecteur des prédictions relatives à l'échantillon d'apprentissage, alors

Proposition 5.1.3. Si $\mathbf{X}^T \mathbf{X}$ est inversible,

$$\hat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y} = \text{Proj}_{V_{\mathbf{X}}}(\mathbf{Y}).$$

De plus,

$$\|\hat{\mathbf{Y}} - \mathbf{X}\theta^*\|^2 = \|\mathbf{X}(\hat{\theta} - \theta^*)\|^2 \sim \sigma^2 \chi_p^2$$

Par conséquent,

$$\mathbb{E} \left[\frac{\|\hat{\mathbf{Y}} - \mathbf{X}\theta^*\|^2}{n} \right] = \sigma^2 \frac{p}{n}.$$

Preuve : On a vu que

$$\hat{\theta} - \theta^* \sim \mathcal{N}(0, \sigma^2 (\mathbf{X}^T \mathbf{X})^{-1})$$

de sorte que

$$\mathbf{X}(\hat{\theta} - \theta^*) \sim \mathcal{N}(0, \sigma^2 \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T) = \mathcal{N}(0, \sigma^2 P_{V_{\mathbf{X}}}).$$

Le théorème de Cochran nous permet alors d'en déduire le résultat (On peut par exemple le vérifier facilement dans le cas où $\mathbf{X}^T \mathbf{X}$ est la matrice I_p).

Remarque 5.1.4. Lorsque p augmente, on constate que l'erreur d'estimation augmente linéairement avec p . Par ailleurs, ce calcul est effectué sur l'échantillon qui a servi à construire $\hat{\theta}$. C'est donc une erreur d'entraînement (erreur généralement "optimiste"). Se posent alors plusieurs questions :

- Dans un cadre classique, comment estimer la qualité du modèle et si besoin comment peut-on sélectionner des variables ?
- Comment envisager le cadre $p > n$ (et donc en particulier lorsque $p \gg n$) où tout ce qui précède n'a pas de sens car $\mathbf{X}^T \mathbf{X}$ ne peut être inversible car de rang inférieur ou égal à $\min(n, p)$.

5.1.5 Qualité du modèle linéaire classique

Dans le modèle linéaire classique, une quantité classique est le R^2 basé sur la décomposition suivante de la somme des carrés (SC) :

$$SCT = SCM + SCR$$

où

$$SCT = \sum_{i=1}^n (\mathbf{Y}_k - \bar{\mathbf{Y}}_n)^2, \quad SCM = \sum_{i=1}^n (\hat{\mathbf{Y}}_k - \bar{\mathbf{Y}}_n)^2, \quad SCR = \sum_{i=1}^n (\mathbf{Y}_k - \hat{\mathbf{Y}}_k)^2.$$

Remarque 5.1.5. Cette décomposition de la variance empirique s'écrit sous forme condensée de la manière suivante :

$$\|\mathbf{Y} - \bar{\mathbf{Y}}\|^2 = \|\hat{\mathbf{Y}} - \bar{\mathbf{Y}}\|^2 + \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2.$$

On remarque en particulier que le "double-produit" a disparu.

Ceci est dû à la propriété suivante :

Proposition 5.1.6. On a :

$$\bar{\mathbf{Y}} = \frac{1}{n} \sum_{k=1}^n \hat{\mathbf{Y}}_k.$$

En particulier, $\bar{\mathbf{Y}} \in V_{\mathbf{X}}$ et $\hat{\mathbf{Y}} - \bar{\mathbf{Y}} \in V_{\mathbf{X}}$. Par conséquent,

$$\langle \hat{\mathbf{Y}} - \bar{\mathbf{Y}}, \mathbf{Y} - \hat{\mathbf{Y}} \rangle = 0.$$

Pour prouver ce résultat, le point principal est de se souvenir que $\mathbf{1}$ appartient à $V_{\mathbf{X}}$. La suite est laissée au lecteur.

Coefficient de détermination : Le coefficient de détermination est alors défini par :

$$R^2 = \frac{SCR}{SCT} = 1 - \frac{SST}{SCR} \quad (\text{en anglais } SCR=\text{RSS}).$$

C'est en fait le rapport de carrés de deux longueurs : si l'on "oublie" la moyenne $\bar{\mathbf{Y}}$, c'est le cosinus entre \mathbf{Y} et sa projection $\hat{\mathbf{Y}}$.

Remarque 5.1.7. De cette interprétation géométrique, on comprend facilement que $0 \leq R^2 \leq 1$ et que l'ajout de variables explicatives augmente le R^2 . Plus R^2 est proche de 1 plus l'ajustement sur les données observées est bon.

Coefficient de détermination ajusté :

Proposition 5.1.8. Soient \mathcal{M}_1 et \mathcal{M}_2 deux modèles de régression linéaire de la forme :

$$\begin{aligned} \mathcal{M}_1 : Y &= \beta_0 + \sum_{i=1}^p \beta_i X_i + \varepsilon \\ \mathcal{M}_2 : Y &= \beta_0 + \sum_{i=1}^p \beta_i X_i + \beta_{p+1} X_{p+1} + \varepsilon. \end{aligned}$$

Leurs coefficients de détermination satisfont $R_1^2 \leq R_2^2$.

Cette propriété implique que le R^2 ne détecte pas l'overfitting. On lui préfère souvent le R^2 ajusté défini par :

$$R_{adj}^2 = 1 - \frac{\frac{SCR}{n-p-1}}{\frac{SST}{n-1}}.$$

Le R^2 ajusté est en fait un coefficient de détermination normalisé en fonction des paramètres p et n . Pour rappel, si $\text{Vect}(1, \mathbf{X}_1, \dots, \mathbf{X}_p) = p+1$, alors, on obtient via le théorème de Cochran ($Y - \hat{Y} = P_{V_{\mathbf{X}}^\perp} Y = P_{V_{\mathbf{X}}^\perp} \varepsilon$),

$$\frac{SCR}{\sigma^2} \sim \chi^2(n-p-1).$$

Sous l'hypothèse $(H_0) := \theta_1 = \dots = \theta_p = 0$, on a également que

$$\frac{SCT}{\sigma^2} \sim \chi^2(n-1).$$

Autres mesures de la qualité du modèle

Le R^2 ajusté prend donc plus en compte la fiabilité du modèle et pose la question : vaut-il mieux un modèle complet moins fiable statistiquement qu'un modèle réduit biaisé mais d'estimation plus précise (question du compromis Biais-Variance) ? On parle alors de *sélection de modèles*. Dans la continuité du R^2 ajusté, il existe des indicateurs permettant de choisir le modèle le plus adapté aux données. Il s'agit en particulier d'éviter l'overfitting. On en présente ici succinctement quelques-uns (dont la définition peut varier légèrement selon les sources) :

- Le C_p de Mallows défini par

$$\left(\frac{SCR}{\hat{\sigma}^2} + 2p - n \right) \quad \text{que l'on cherche à minimiser.}$$

- Le critère AIC (Akaike Information Criterion),

$$AIC = \frac{1}{\hat{\sigma}^2} \left(n \frac{\log SCR}{n} + 2p + 1 \right)$$

- Le critère BIC (Bayesian) :

$$BIC = (SCR + \log(n)\hat{\sigma}^2)$$

Là encore, on cherche à minimiser ce critère.

Validation

- Validation : on a pu constater que jusqu'à présent, les aspects "erreur test" et "erreur de validation" n'ont pas été considérés. Les outils présentés plus hauts peuvent être vus comme des alternatives à la validation. Ces outils ont en particulier été développés pour pallier la difficulté computationnelle générée par la validation croisée. Néanmoins, aujourd'hui, ce problème est moins prégnant avec l'accélération des calculateurs.
- Pratique : qu'on utilise validation croisée, R_{adj}^2 , C_p , AIC ou BIC , une question se pose, comment comparer l'ensemble des modèles que l'on peut fabriquer avec p variables ? En effet, il en a 2^p !!
- Réponse : Algorithmes de sélections de variables "Pas à Pas", "Mixtes", "Globaux" selon les situations.

Algorithmes de sélection de variables

Présentons quelques méthodes (qui possèdent toutes leurs variantes) :

- Forward : A chaque pas, une variable est ajoutée au modèle. C'est celle dont le R^2 est maximal (ou de manière équivalente celle dont le SCR est minimal). On obtient ainsi p modèles contenant 1, 2, 3, ..., p variables. On choisit ensuite celui qui minimise un critère pénalisé (R_{adj}^2, \dots).
- Backward : A chaque pas, une variable est retirée au modèle : celle qui augmente le moins le SCR (ou qui diminue le plus moins le R^2).
- On a aussi des approches hybrides (qui mélangeant ces deux approches).

Les méthodes présentées ci-dessus souffrent de deux problèmes :

- Lorsque p est grand, leur mise en oeuvre est coûteuse et l'exploration des modèles reste nécessairement très partielle.
- Lorsque $p > n$, elles ne sont a priori pas applicables puisque l'estimateur $\hat{\theta}$ n'est pas bien défini. En effet, comme $rg(\mathbf{X}) < p$, $\text{Ker } \mathbf{X}$ est non réduit à $0_{\mathbb{R}^p}$. Ainsi,

$$\text{Argmin}_{\theta} \|\mathbf{Y} - \mathbf{X}\theta\|^2$$

n'est pas unique.

- Une théorie assez récente permet de pallier le second problème en introduisant une pénalisation dans l'estimateur. Il permet aussi de donner une alternative à la sélection de modèles dans le cadre classique.

Il existe plusieurs manières de pallier ce problème. Nous évoquons dans un premier temps deux variations de la régression linéaire standard utilisées dans la pratique puis nous étudierons en détail les méthodes dites régularisées/pénalisées.

5.1.6 Régressions PCR/PLS

Avant d'aborder les méthodes pénalisées, évoquons ici deux méthodes de régression linéaire avec réduction de dimension qui sont assez fréquemment utilisées.

Régression PCR

La régression sur composantes principales ou PCR est basée sur le principe simple suivant : réduire le nombre de paramètres par ACP puis appliquer une régression linéaire standard. Plus précisément, l'idée est de “résumer” les variables X_1, \dots, X_p par un sous-ensemble de variables Z_1, \dots, Z_r deux à deux orthogonales et combinaisons linéaires des variables X_1, \dots, X_p . A r fixé ($r \leq p$), les variables Z_1, \dots, Z_p sont simplement les composantes principales associées des variables obtenues par l'analyse en composantes principales de la matrice \mathbf{X} des données. Généralement, les variables sont centrées réduites.

La régression PCR peut avoir des effets intéressants lorsque les variables d'entrée sont trop corrélées. Par ailleurs, cette méthode peut être vue comme un moyen de considérer des situations où $n < p$ (en commençant par sélectionner r variables avec $r \leq n$). En revanche, au vu des limites de l'ACP évoquées précédemment, on peut avoir des doutes quant à sa robustesse à la dimension.

Régression PLS

La régression PLS (“Partial Least Squares” ou “moindres carrés partiels”) a en commun avec la régression PCR d'avoir comme objectif de réduire la dimension, en cherchant un ensemble de r composantes orthogonales optimales en un certain sens. En revanche, ici, le principe de la construction des composantes est optimisé pour que celles-ci maximisent leur lien avec à la variable Y au sens de la covariance empirique.

Pour rappel, dans le cas de la régression PCR, qui se base au préalable sur une ACP, on cherche une matrice $V^* = (V_1^* | V_2^* | \dots | V_d^*)$ de taille $p \times d$ où les V_1, \dots, V_d forment une base orthonormée, définie par :

$$V^* = \operatorname{Argmax}_{V, V^T V = I_d} V^T \mathbf{X}^T \mathbf{X} V,$$

qui de manière équivalente, garantit que V^* minimise $\sum_{i=1}^n \| \mathbf{X}^{(i)} - P_V(\mathbf{X}^{(i)}) \|^2$. La régression PCR consiste alors à poser $\tilde{\mathbf{X}} = \mathbf{X} V^*$, matrice $n \times d$ puis à résoudre le problème de régression linéaire $\mathbf{Y} = \tilde{\mathbf{X}} \theta + \varepsilon$. Dit autrement, une donnée d'entrée $x = (x_1, \dots, x_p)$ est remplacée par $\tilde{x} = (\langle x, V_1^* \rangle, \dots, \langle x, V_d^* \rangle)$. Notons également que d'un point de vue vectoriel, l'ACP résout le problème suivant. Supposons construits V_1, \dots, V_{m-1} . Alors, pour le choix de la m -ième direction, on cherche le vecteur v solution de

$$\begin{aligned} v &= \operatorname{Argmax}_{\|\alpha\|=1} \alpha^T \mathbf{X}^T \mathbf{X} \alpha \\ &\text{parmi les } \alpha \text{ tels que } \alpha^T \mathbf{X}^T \mathbf{X} v_\ell = 0 \text{ pour tout } \ell \in \{1, \dots, m-1\}. \end{aligned}$$

Notons que la condition sur α garantit que $\mathbf{X}v$ est orthogonal à $\mathbf{X}V_\ell$ pour tout $\ell \in \{1, \dots, m-1\}$. Pour la régression PLS, le principe est similaire (a minima dans sa version la plus simple) mais basé sur la maximisation de la covariance avec Y . Notons que cette méthode, comme la régression PCR n'est pas invariante par changement d'échelle. De ce fait, on suppose généralement dans ces méthodes que \mathbf{X} est centrée réduite. On résout dans ce cas le problème d'optimisation suivant. Pour le choix de la m -ième direction, on cherche le vecteur v solution de

$$\begin{aligned} v &= \operatorname{Argmax}_{\|\alpha\|=1} \alpha^T \mathbf{X}^T \mathbf{Y} \mathbf{Y}^T \mathbf{X} \alpha \\ &\text{parmi les } \alpha \text{ tels que } \alpha^T \mathbf{X}^T \mathbf{X} v_\ell = 0 \text{ pour tout } \ell \in \{1, \dots, m-1\}. \end{aligned}$$

La condition sur α garantit à nouveau que $\mathbf{X}v$ est orthogonal à $\mathbf{X}V_\ell$ pour tout $\ell \in \{1, \dots, m-1\}$. Notons en revanche ici que lorsque \mathbf{Y} est unidimensionnel, $\mathbf{X}^T \mathbf{Y}$ est un vecteur de taille p de sorte que pour $m = 1$, un vecteur maximisant $v = \operatorname{Argmax}_{\|\alpha\|=1} \alpha^T \mathbf{X}^T \mathbf{Y} \mathbf{Y}^T \mathbf{X} \alpha$ est un vecteur colinéaire à $\mathbf{X}^T \mathbf{Y}$ de norme 1. Cela signifie que $V_1 = \frac{\mathbf{X}^T \mathbf{Y}}{\|\mathbf{X}^T \mathbf{Y}\|}$. On trouve ensuite le second vecteur via le procédé de Gram-Schmidt. Plus précisément, pour

la deuxième direction, on cherche le vecteur u parmi les $\mathbf{X}u$ orthogonaux à $\mathbf{X}V_1$ qui maximise à nouveau la covariance au carré entre $\mathbf{X}u$ et Y . Or, il est facile de montrer que

$$\{\mathbf{X}u, \mathbf{X}u \text{ orthogonal à } \mathbf{X}V_1\} = \{\mathbf{X}v - \frac{\langle \mathbf{X}v, \mathbf{X}V_1 \rangle}{\|\mathbf{X}V_1\|^2} \mathbf{X}V_1, v \in \mathbb{R}^d\}.$$

Comme

$$\mathbf{X}v - \frac{\langle \mathbf{X}v, \mathbf{X}V_1 \rangle}{\|\mathbf{X}V_1\|^2} \mathbf{X}V_1 = \tilde{\mathbf{X}}v \quad \text{avec } \tilde{\mathbf{X}} = \mathbf{X} - \frac{1}{\|\mathbf{X}V_1\|^2} \mathbf{X}V_1 (\mathbf{X}V_1)^T \mathbf{X},$$

on peut alors en déduire que $V_2 = \frac{\tilde{\mathbf{X}}^T \mathbf{Y}}{\|\tilde{\mathbf{X}}^T \mathbf{Y}\|}$ puis un algorithme en remplaçant à chaque fois la matrice courante notée abusivement \mathbf{X} par $\mathbf{X} - \frac{1}{\|\mathbf{X}V_{k-1}\|^2} \mathbf{X}V_{k-1} (\mathbf{X}V_{k-1})^T \mathbf{X}$. C'est ce qu'on appelle la *déflation*.

Pour plus de détails, on pourra consulter le cours <https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-sparse-pls.pdf> qui présente de manière assez complète cette famille d'algorithmes très utilisée dans la pratique.

5.2 LASSO/RIDGE/ELASTIC NET

Ces méthodes font partie de la famille des régressions pénalisées : on pénalise la fonction de perte afin de favoriser certains θ .

5.2.1 La pénalisation par la “norme” de θ

Principe général de la régression pénalisée

- Pour un $\lambda > 0$, on cherche à minimiser la fonction

$$\Phi_{\mathcal{P}}(\theta) = \frac{\|\mathbf{Y} - \mathbf{X}\theta\|^2}{2n} + \lambda \mathcal{P}(\theta).$$

\mathcal{P} désigne la *pénalité*.

- Quelques exemples de pénalités :

- $\mathcal{P}(\theta) = \|\theta\|_0 := \text{Card}\{i, \theta_i \neq 0\}$. On parle de pénalisation ℓ_0 . Attention $\|\cdot\|_0$ n'est pas une norme (voir plus loin).
- $\mathcal{P}(\theta) = \|\theta\|_1 = \sum_{i=1}^p |\theta_i|$ (LASSO, pénalisation ℓ_1).
- $\mathcal{P}(\theta) = \frac{\|\theta\|_2^2}{2} = \frac{1}{2} \sum_{i=1}^p \theta_i^2$ (Ridge).
- $\mathcal{P}(\theta) = (1 - \alpha) \frac{\|\theta\|_2^2}{2} + \alpha \|\theta\|_1$, $\alpha \in [0, 1]$ (Elastic Net).

Rôle de la pénalisation

- On favorise les solutions pour lesquelles $\mathcal{P}(\theta)$ est petit.
- Ainsi, par exemple, en norme $\|\cdot\|_0$, on pénalise le nombre de variables. On favorise les solutions qui ont peu de variables “allumées”.
- **Remarque** : les C_p , AIC et BIC ressemblent à des pénalisations par cette norme sauf que la pénalisation est “uniforme”.

- Interprétation Lagrangienne : la fonction $\Phi_{\mathcal{P}}(\theta)$ est à comprendre comme un *Lagrangien*, i.e. une fonction fabriquée pour *régulariser* le problème d'optimisation sous contrainte

$$\min_{\mathcal{P}(\theta) \leq s(\lambda)} \frac{\|\mathbf{Y} - \mathbf{X}\theta\|^2}{2n}.$$

(voir plus loin pour quelques rappels)

Représentation graphique

Représentons ici le problème sous contrainte (en dimension 2) :

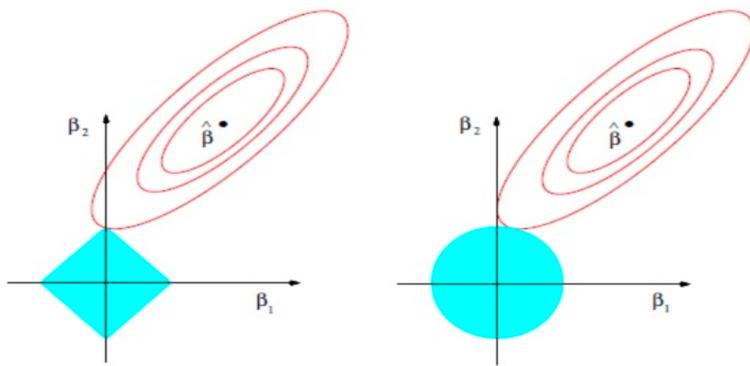


Figure 5.1: Gauche : en norme 1, Droite, en norme 2 ($\hat{\beta} = \operatorname{Argmin}_{\beta \in \mathbb{R}^2} \|Y - X\beta\|^2$, en rouge, les lignes de niveau de $\|Y - X\beta\|^2$ autour du minimum)

5.2.2 Rappels succincts sur le Lagrangien

On considère le problème suivant : minimiser la fonction J sur un sous-ensemble G défini par $G = \{\theta \in \mathbb{R}^p, f(\theta) = 0\}$ (f supposée régulière ici).

- Exemple : $J(\theta) = \|Y - X\theta\|^2$ et $f(\theta) = \|\theta\|_2^2 - R$.
- Dans ce cas, le théorème des multiplicateurs de Lagrange donne la condition nécessaire suivante :

Théorème 5.2.1. *Si θ^* est un point de G où J atteint son minimum, alors il existe λ^* tel que*

$$\nabla J(\theta^*) + \lambda^* \nabla f(\theta^*) = 0.$$

Ainsi, si l'on note L la fonction définie par

$$L(\lambda, \theta) = J(\theta) + \lambda f(\theta)$$

alors, (λ^*, θ^*) est solution de

$$\partial_\theta L(\lambda^*, \theta^*) = \partial_\lambda L(\lambda^*, \theta^*) = 0.$$

La première partie du théorème indique en particulier que là où le minimum est atteint $\nabla J(\theta^*)$ et $\nabla f(\theta^*)$ sont colinéaires. Admettons maintenant que J et f sont convexes. Dans ce cas,

- $\mathcal{C} = \{\theta, f(\theta) \leq 0\}$ est convexe.
- Si $\min_{\theta \in \mathbb{R}^p} J(\theta)$ est en dehors de \mathcal{C} , on peut alors vérifier que $\min_{\theta \in \mathcal{C}} J(\theta)$ est nécessairement sur le bord du convexe.

- Les points critiques de $\theta \mapsto L(\lambda, \theta)$ sont des minima.
- **Pour aller plus loin :** Pour rendre la condition nécessaire du théorème précédent suffisante, on s'appuie souvent sur les conditions “KKT”.
- **Important :** Considérer la fonction $\theta \mapsto L(\lambda, \theta)$ permet de considérer un problème d'optimisation non contraint. Lorsque la fonction L est convexe, alors, on peut mettre en oeuvre des algorithmes d'optimisation (type descente de gradient) pour déterminer le minimum de L
- **En pratique :** on cherche le minimum θ_λ^* de $\theta \mapsto L(\lambda, \theta)$ (à λ fixé) puis on fait varier λ en étudiant l'évolution de $J(\theta^*)$.
- **Cas limites :** lorsque $\lambda = 0$, on regarde le problème non pénalisé tandis que lorsque $\lambda \rightarrow +\infty$, on cherche à déterminer la meilleure approximation sur une boule de rayon tendant vers 0.

5.2.3 A propos de la pénalisation ℓ_0

- Si l'on veut pénaliser le nombre de variables pour réduire la variance du modèle, le choix naturel est la pseudo-norme $\|\cdot\|_0$. On parle de pseudo-norme car celle-ci n'est pas une norme au sens mathématique (2 des 3 propriétés définissant une norme ne sont pas satisfaites).
- **Parcimonie :** on parle dans ce cas d'hypothèse de parcimonie (“sparsity”): on fait l'hypothèse que peu de variables sont réellement explicatives. Si tel est le cas, *i.e.* si le vrai modèle est de la forme

$$Y = X\theta^* + \varepsilon$$

avec $\|\theta^*\|_0 = \text{Card}\{j, \theta_j^* \neq 0\} = s_0$ petit devant p (et si possible devant n également).

- Notons \mathbf{X}_{s_0} la sous-matrice de \mathbf{X} de taille $n \times s_0$ constituée des colonnes “allumées” de \mathbf{X} . Si l'on suppose que $\mathbf{X}_{s_0}^T \mathbf{X}_{s_0}$ est inversible, alors, une fois “déterminé ce support”, l'application du modèle linéaire standard permet d'obtenir une erreur en $\frac{s_0}{n}$.
- **Problème :** La (pseudo)-norme $\|\cdot\|_0$ n'est pas une fonction convexe !! Chercher le minimum de la fonction $\Phi_{\|\cdot\|_0}$ n'est pas un problème soluble numériquement car il faut a priori explorer tout l'espace de manière exhaustive.

De la pénalisation ℓ_0 à la pénalisation ℓ_1

- **Idée :** Remplacer la (pseudo)-norme $\|\cdot\|_0$ par une pénalisation par la fonction convexe “minimale”: la norme $\|\cdot\|_1$ (pour comprendre la notion de minimalité de convexité, penser à l'application $x \mapsto |x|$ en dimension 1).
- **LASSO :** Acronyme de Least Absolute Shrinkage and Selection Operator (introduit par Tibshirani en 1996).
- **Objectifs du LASSO :** Obtenir avec une norme moins “sélective” que la pénalisation ℓ_0 des résultats qui sont sensiblement similaires à ceux que l'on obtiendrait avec la norme “idéale”. Notons

$$\hat{\theta}_\lambda = \underset{\theta \in \mathbb{R}^p}{\text{Argmin}} \frac{\|\mathbf{Y} - \mathbf{X}\theta\|^2}{2n} + \lambda\|\theta\|_1.$$

- Estimation : $\hat{\theta}_\lambda$ proche de θ^* ?
- Sélection : si J_0 est le “vrai” support, obtenir avec grande probabilité $J(\hat{\theta}_\lambda^*)$ proche de J_0 ?
- Prédiction : $\frac{1}{n}\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2 = O(\frac{s_0}{n})$ avec grande probabilité ?

5.2.4 Lasso vs Ridge et Elastic Net

Ridge et Elastic Net sont aussi basés sur des pénalisations convexes (même strictement convexes). Ridge a de plus une solution explicite. Quel est leur rôle statistique ?

- LASSO : tente de mimer la norme $\|\cdot\|_0$ en tentant d'extraire des variables d'intérêt et en annulant les autres.
- Ridge est une méthode plus “douce” qui réduit (“shrinks”) la taille des variables, notamment celles qui sont corrélées.
- Elastic Net tente donc de mélanger ces deux approches.
- Dans la suite, on parlera principalement du LASSO et du Ridge.

5.2.5 Evolution avec λ

- La fonction $\Phi_{\mathcal{P}}$ est croissante avec λ .
- Comment choisir λ ? Il s’agit d’un compromis biais-variance. Plus λ est petit, plus la variance est élevée. Plus λ est grand, moins le modèle est flexible et donc plus le biais est élevé.
- On choisit donc λ en minimisant (l'estimation de) l'erreur de prédiction.
- Ce choix est fait par validation (simple ou croisée) (voir figure slide suivant).
- On trace aussi généralement les coefficients sélectionnés en fonction de λ (voir figure slide suivant).

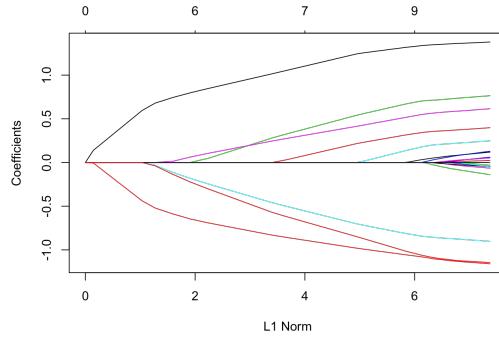


Figure 5.2: Apparition des coefficients dans le LASSO en fonction de $\|\hat{\theta}_\lambda\|_1$ (remarque : à l'origine $\lambda = +\infty$)

5.2.6 Un résultat simple pour le Ridge

- si $\mathcal{P}(\theta) = \frac{\|\theta\|_2^2}{2}$, alors la fonction $\Phi_{\mathcal{P}}$ définie par

$$\Phi_{\mathcal{P}}(\theta) = \frac{\|\mathbf{Y} - \mathbf{X}\theta\|^2}{2n} + \lambda\mathcal{P}(\theta)$$

est strictement convexe de classe \mathcal{C}^2 et coercive (tend vers $+\infty$ lorsque $\|\theta\|_2 \rightarrow +\infty$). On a alors le résultat suivant :

Proposition 5.2.2. Si $\mathcal{P}(\theta) = \frac{\|\theta\|_2^2}{2}$, alors $\Phi_{\mathcal{P}}$ admet un unique minimum en

$$\hat{\theta}_\lambda = (\mathbf{X}^T \mathbf{X} + \lambda n I_p)^{-1} \mathbf{X}^T \mathbf{Y}.$$

Preuve. Il suffit de chercher l'unique point critique de la fonction $\Phi_{\mathcal{P}}$. ■

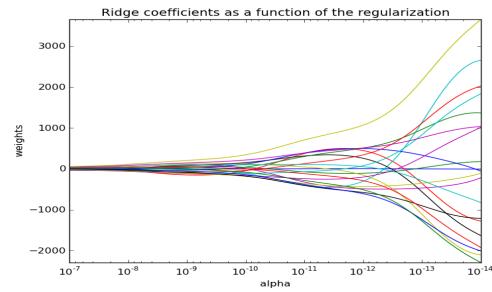


Figure 5.3: Apparition des coefficients dans le RIDGE en fonction de λ (plus “smooth”)/“Chemin de régularisation”

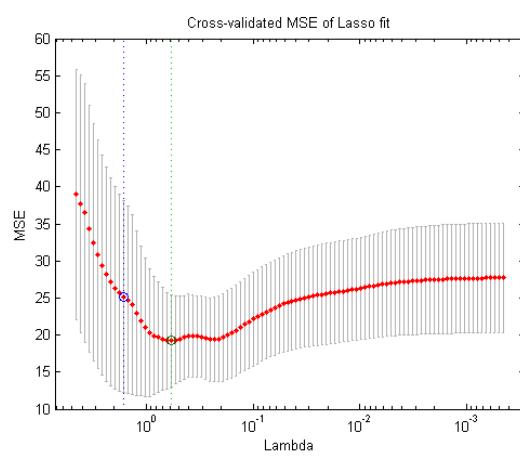


Figure 5.4: Estimation de la MSE en fonction de λ

Remarque 5.2.3. • Inverse bien définie car $\mathbf{X}^T \mathbf{X} + \lambda n I_p$ est une matrice définie positive.

- On voit l'effet de la pénalisation qui apparaît au “dénominateur” et réduit la taille des coefficients. On voit également que l'on ne “tue” pas les coefficients.

5.2.7 “Calcul” de $\hat{\theta}_\lambda$ pour le LASSO

Le problème est plus difficile car $\|\theta\|_1$ est non strictement convexe et non dérivable sur les axes.

- **Un cas particulier :** $\mathbf{X}^T \mathbf{X} = I_p$ (ce qui implique que $p \leq n$). Dans ce cas, la première forme quadratique est définie positive donc $\Phi_{\mathcal{P}}$ est strictement convexe et coercive. Elle admet donc un unique minimum. On a de plus

Proposition 5.2.4 (Seuillage doux (Soft-Thresholding)). *Si $\mathbf{X}^T \mathbf{X} = I_p$, alors si l'on note $\hat{\theta} = X^T Y$ la solution du modèle linéaire non pénalisé, on a :*

$$\hat{\theta}_\lambda(j) = \text{sgn}(\hat{\theta}(j)) \left(\frac{|\hat{\theta}(j)|}{n} - \lambda \right)_+$$

Remarque : On voit donc sur cet exemple que le LASSO sélectionne les coordonnées de la solution classique qui sont supérieures à λ mais les “shrink” (les réduit) aussi.

Dans le cas général, il n'y a pas nécessairement unicité à la solution du LASSO. De plus, “la” solution n'est pas explicite. On doit faire recours à un algorithme d'optimisation pour déterminer $\hat{\theta}_\lambda$. On peut néanmoins affirmer que :

Proposition 5.2.5. $\theta \in \mathbb{R}^p$ est solution du LASSO de paramètre λ si et seulement si les conditions de “stationnarité” suivantes sont satisfaites : pour tout $j \in \{1, \dots, p\}$

$$\begin{cases} \mathbf{X}_j^T(Y - \mathbf{X}\theta) = n\lambda \text{sgn}(\theta_j) & \text{si } \theta_j \neq 0 \\ |\mathbf{X}_j^T(Y - \mathbf{X}\theta)| \leq n\lambda & \text{sinon.} \end{cases}$$

Preuve. Pour démontrer ce résultat, on fait appel aux outils d'optimisation d'une fonction convexe non différentiable. Cette théorie s'appuie sur la notion de sous-différentiel qui est une extension de la notion de dérivée usuelle aux fonctions convexes. Cette notion sera détaillée en TD mais en voici quelques rudiments. ■

Rappel : Sous-Différentiel/Sous-Gradient. Une fonction $F : \mathbb{R}^p \rightarrow \mathbb{R}$ convexe, lorsqu'elle est différentiable en $x \in \mathbb{R}^p$, satisfait

$$F(y) \geq F(x) + \langle \nabla F(x), y - x \rangle \quad \forall y \in \mathbb{R}^p,$$

où ∇F désigne le gradient usuel de F . Le sous-différentiel est une notion définie pour préserver cette inégalité dans le cas non différentiable.

Définition 5.2.6. On appelle sous-différentiel de F en x , l'ensemble défini par

$$\partial F(x) = \{w \in \mathbb{R}^p, F(y) \geq F(x) + \langle w, y - x \rangle \quad \forall x \in \mathbb{R}^p\}.$$

Un élément w de $\partial F(x)$ est appelé un sous-différentiel de F en x .

Lorsque F est convexe, le sous-différentiel est un ensemble non vide (voir TD). De plus, si F est différentiable, alors $\partial F(x) = \{\nabla F(x)\}$, de sorte que l'on retrouve bien le calcul différentiel usuel dans ce cas.

L'exemple le plus simple est $F(x) = |x|_1 = \sum_{i=1}^p |x_i|$. F est différentiable sauf en les axes, i.e. en

$$\bigcup_{i=1}^p \{x \in \mathbb{R}^p, x_i = 0\}.$$

On peut alors vérifier que

$$\partial F(x) = \{w \in \mathbb{R}^p, w_j = \text{sign}(x_j), \text{ si } x_j \neq 0 \text{ et } w_j \in [-1, 1] \text{ si } x_j = 0\}.$$

Ci-dessous, deux propriétés importantes des sous-différentielles :

Proposition 5.2.7. Soit $F : \mathbb{R}^p \rightarrow \mathbb{R}$ une fonction convexe. Alors,

- Pour tous $x, y \in \mathbb{R}^p$,

$$\langle w_x - w_y, x - y \rangle \geq 0, \quad \forall w_x \in \partial F(x), \forall w_y \in \partial F(y).$$
- On a aussi l'équivalence suivante :

$$x \in \operatorname{Argmin} F \iff 0 \in \partial F(x).$$

Cette dernière propriété est justement l'outil qui permet de démontrer la proposition 5.2.5.

Unicité de la solution du LASSO : La solution du LASSO n'est a priori pas unique puisque la seconde condition est définie par une inégalité. Néanmoins, voici un résultat plus précis à ce sujet :

Proposition 5.2.8. • Soient $\hat{\theta}^{(1)}$ et $\hat{\theta}^{(2)}$ deux minimiseurs de F_λ définie par

$$F_\lambda(\theta) = \|\mathbf{Y} - \mathbf{X}\theta\|^2 + \lambda\|\theta\|_1.$$

On a toujours $\mathbf{X}\hat{\theta}^{(1)} = \mathbf{X}\hat{\theta}^{(2)}$.

- Soit $J := \{j \in \{1, \dots, p\} \mid |\mathbf{X}_j^T(\mathbf{Y} - \mathbf{X}\hat{\theta})| = \lambda\}$ (ensemble qui ne dépend pas du minimiseur par le point précédent). Alors, si $\mathbf{X}_J^T \mathbf{X}_J$ est inversible, $\operatorname{Argmin} F_\lambda$ est réduit à un point. En d'autres termes, la solution du LASSO est dans ce cas unique. L'ensemble J coïncide alors le support de $\hat{\theta}$.

Pour la preuve, on pourra consulter [6], p.92.

Exercice. Démontrer la première propriété. Proposer quelques exemples permettant d'illustrer la situation ci-dessus.

5.2.8 En pratique

Voici 2 types d'algorithmes pour calculer $\hat{\theta}_{LASSO}$ (voir TD pour plus de détails)

- **L'algorithme LARS** (Least Angle Regression) : calcul par palier des variables allumées (cf graphe d'apparition des coefficients). On fait décroître λ jusqu'à l'apparition d'une première variable (la plus corrélée avec y). Ensuite, on cherche le seuil pour lequel la deuxième variable apparaît (toujours par un principe de corrélation ou d'angle)
- **La descente coordonnées par coordonnées.** Il s'agit d'un algorithme d'optimisation où l'on minimise coordonnées par coordonnées. Etape 1 : on fige $\theta_2, \dots, \theta_p$ et on regarde la fonction

$$\theta_1 \mapsto \frac{1}{2n} \sum_{i=1}^n (y_i - \sum_{k \neq j} x_{ik} \theta_k - x_{i1} \theta_1)^2 + \lambda \sum_{k \neq 1} |\theta_k| + \lambda |\theta_1|.$$

On en tire un minimum (explicite dans le cas du LASSO) $\hat{\theta}_1$ puis on réitère la minimisation sur la deuxième coordonnée en conservant la valeur de $\hat{\theta}_1, \dots$. Par construction, il s'agit d'une suite décroissante. Sous conditions standard, l'algorithme converge vers un minimiseur (même principe que EM).

5.3 Prédition/Estimation : Résultats

5.3.1 Un premier résultat de prédition

On commence par un premier résultat général et sans condition de parcimonie (qu'on retrouvera dans la littérature sous la terminologie "slow LASSO").

Théorème 5.3.1. *On suppose que $\mathbf{Y} = \mathbf{X}\theta^* + \varepsilon$. On note $\hat{\theta}_\lambda = \operatorname{Argmin}_\theta \left(\frac{\|\mathbf{Y} - \mathbf{X}\theta\|_2^2}{2n} + \lambda\|\theta\|_1 \right)$.*

(i) (*Résultat algébrique*) *Si $\|\frac{\mathbf{X}^T \varepsilon}{n}\|_\infty \leq \lambda$,*

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{2n} \leq 2\lambda\|\theta^*\|_1.$$

(ii) (*Résultat probabiliste*) *Supposons que les ε_i sont i.i.d. de loi $\mathcal{N}(0, \sigma^2)$ et que les colonnes de \mathbf{X} ont été centrées et standardisées de sorte pour tout $j \in \{1, \dots, p\}$, $n^{-\frac{1}{2}}\|\mathbf{X}_j\|_2 \leq C$. Alors, si $\lambda = 2C\sqrt{\frac{\log p + \log(\frac{1}{\delta})}{n}}$, on a avec probabilité $1 - \delta$,*

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{2n} \lesssim \sqrt{\frac{\log p + \log(\frac{1}{\delta})}{n}}\|\theta^*\|_1.$$

Preuve. (i) Par définition de $\hat{\theta}_\lambda$, on a

$$\frac{\|\mathbf{Y} - \mathbf{X}\hat{\theta}_\lambda\|^2}{2n} + \lambda\|\hat{\theta}_\lambda\|_1 \leq \frac{\|\mathbf{Y} - \mathbf{X}\theta^*\|^2}{2n} + \lambda\|\theta^*\|_1.$$

Comme $\mathbf{Y} = \mathbf{X}\theta^* + \varepsilon$, on a également :

$$\|\mathbf{Y} - \mathbf{X}\hat{\theta}_\lambda\|^2 = \|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2 - 2\langle \mathbf{X}(\hat{\theta}_\lambda - \theta^*), \varepsilon \rangle + \|\varepsilon\|^2$$

tandis que

$$\|\mathbf{Y} - \mathbf{X}\theta^*\|^2 = \|\varepsilon\|^2.$$

Ainsi,

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{2n} \leq \frac{\langle \hat{\theta}_\lambda - \theta^*, \mathbf{X}^T \varepsilon \rangle}{n} + \lambda(\|\theta^*\|_1 - \|\hat{\theta}_\lambda\|_1).$$

En utilisant que

$$\langle \hat{\theta}_\lambda - \theta^*, \mathbf{X}^T \varepsilon \rangle \leq \|\mathbf{X}^T \varepsilon\|_\infty \|\hat{\theta}_\lambda - \theta^*\|_1 \leq \|\mathbf{X}^T \varepsilon\|_\infty (\|\hat{\theta}_\lambda\|_1 + \|\theta^*\|_1),$$

il vient

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{2n} \leq \|\hat{\theta}_\lambda\|_1 \left(\frac{\|\mathbf{X}^T \varepsilon\|_\infty}{n} - \lambda \right) + \|\theta^*\|_1 \left(\frac{\|\mathbf{X}^T \varepsilon\|_\infty}{n} + \lambda \right).$$

Sous l'hypothèse $\lambda \geq \|\frac{\mathbf{X}^T \varepsilon}{n}\|_\infty$, on en déduit

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{2n} \leq 2\lambda\|\theta^*\|_1.$$

(ii) Pour en déduire des bornes probabilistes, il nous faut maintenant contrôler

$$\mathbb{P}\left(\left\|\frac{\mathbf{X}^T \varepsilon}{n}\right\|_\infty > \lambda\right) = \mathbb{P}\left(\max_{j=1}^p \frac{|\sum_{i=1}^n \mathbf{X}_{i,j} \varepsilon_i|}{n} > \lambda\right).$$

On fait la preuve avec $\sigma = 1$. On rappelle que les colonnes ont été recentrées et que $n^{-1/2}\|\mathbf{X}_j\|_2 \leq C$. Alors pour tout j , $\sum_{i=1}^p \mathbf{X}_{i,j} \varepsilon_i$ suit une loi $\mathcal{N}(0, \|\mathbf{X}_j\|_2^2) = \mathcal{N}(0, \sigma_j^2)$ avec $\sigma_j^2 \leq C^2 n$. Ainsi, en utilisant que la probabilité de l'union est plus petite que la somme des probabilités, il vient

$$\mathbb{P}\left(\left\|\frac{\mathbf{X}^T \varepsilon}{n}\right\|_\infty > \lambda\right) \leq \sum_{j=1}^p \mathbb{P}\left(|Z| > \frac{n\lambda}{\sigma_j}\right)$$

où Z suit une loi $\mathcal{N}(0, 1)$. Or,

$$\mathbb{P}(|Z| > t) \leq e^{-\frac{t^2}{4}}$$

de sorte que

$$\mathbb{P}\left(\left\|\frac{\mathbf{X}^T \varepsilon}{n}\right\|_\infty > \lambda\right) \leq \sum_{j=1}^p e^{-\frac{n^2 \lambda^2}{4\sigma_j^2}} \leq p e^{-\frac{n\lambda^2}{4C^2}}.$$

Pour que cette probabilité soit plus petite qu'un $\delta > 0$, il faut donc

$$e^{-\frac{n\lambda^2}{4C^2}} \leq \frac{\delta}{p} \iff \lambda \geq 2C \sqrt{\frac{\log p + \log(\frac{1}{\delta})}{n}}$$

ce qui donne le résultat. □

5.3.2 La condition RE

Pour obtenir des résultats d'estimation et améliorer les résultats de prédiction, on introduit une condition supplémentaire.

Définition 5.3.2. Soit $\alpha > 0$ et J_0 le support de θ^* . On note

$$C_\alpha(J_0) = \{\Delta \in \mathbb{R}^p, \|\Delta_{J_0}^c\|_1 \leq \alpha \|\Delta_{J_0}\|_1\}.$$

La matrice de design \mathbf{X} vérifie la Restricted Eigenvalue (RE) condition sur J_0 avec les paramètres (κ, α) si

$$\|\mathbf{X}\Delta\|_2 \geq \kappa \|\Delta\|_2 \quad \forall \Delta \in C_\alpha(J_0).$$

$C_\alpha(J_0)$ désigne l'ensemble des vecteurs de \mathbb{R}^p qui ont plus de "masse" sur J_0 . En fait, $\hat{\Delta} = \hat{\theta}_\lambda - \theta^*$ a vocation (sous des hypothèses adéquates) à vivre dans $C_\alpha(J_0)$. La condition (RE) garantit alors que $\|\mathbf{X}\hat{\Delta}\|_2 \geq \kappa \|\hat{\Delta}\|_2$ de sorte que l'on pouvoir déduire des résultats d'estimation, *i.e.* des bornes sur $\|\hat{\theta}_\lambda - \theta^*\|_2$ de celles obtenues pour $\|\mathbf{X}\hat{\Delta}\|_2$.

5.3.3 Estimation

Théorème 5.3.3. Supposons que la condition RE vérifiée avec $\alpha = 3$ et $\kappa > 0$.

(i) (Résultat algébrique) Alors, si $\lambda > \|\frac{\mathbf{X}^T \varepsilon}{n}\|_\infty$,

$$\|\hat{\theta} - \theta^*\|_2 \leq \frac{3\lambda\sqrt{s_0}}{\kappa}.$$

(ii) (Résultat probabiliste) Supposons que les ϵ_i sont i.i.d. de loi $\mathcal{N}(0, \sigma^2)$ et que les colonnes de \mathbf{X} ont été centrées et standardisées de sorte pour tout $j \in \{1, \dots, p\}$, $n^{-\frac{1}{2}} \|\mathbf{X}_j\| \leq C$. Alors, si $\delta > 0$ et $\lambda = 2C\sigma\sqrt{\frac{2\log p + \delta^2}{n}}$, le point (i) est vérifié avec probabilité $1 - 2e^{-\frac{\delta^2}{2}}$. Ainsi, avec grande probabilité,

$$\|\hat{\theta} - \theta^*\|_2^2 \lesssim \frac{s_0 \log p}{n}$$

où $s_0 = \text{Card}\{i, \beta_i^* \neq 0\}$.

5.3.4 Prédiction II

Sous (RE), on peut aussi améliorer les résultats de prédiction obtenus précédemment :

Théorème 5.3.4 (Fast Lasso). *Supposons que les ϵ_i sont i.i.d. de loi $\mathcal{N}(0, \sigma^2)$ et que les colonnes de \mathbf{X} ont été standardisées de sorte pour tout $j \in \{1, \dots, p\}$, $n^{-\frac{1}{2}}\|\mathbf{X}_j\| \leq C$. Alors, si $\delta > 0$ et $\lambda = 2C\sigma\sqrt{\frac{2\log p + \delta^2}{n}}$, Si la condition RE est satisfaite, alors*

$$\frac{\|\mathbf{X}(\hat{\theta}_\lambda - \theta^*)\|_2^2}{n} \lesssim \frac{s_0 \log p}{n}.$$

Pour les deux résultats ci-dessus, on renvoie à http://www.stat.cmu.edu/~arinaldo/Teaching/36710/F18/Scribed_Lectures/Oct22.pdf ou à [5].

5.3.5 Recouvrement du support

On termine par le problème de recouvrement du support. Plus exactement, la question posée est la suivante : le Lasso a-t-il la capacité à ne pas générer de faux positifs ? Un élément de réponse est le théorème suivant dont on trouvera un énoncé plus précis dans [5].

Théorème 5.3.5. *Notons \mathbf{X}_J la sous-matrice obtenue en conservant les colonnes actives de θ^* . Supposons que $\mathbf{X}_J^T \mathbf{X}_J$ est inversible et qu'une condition d'incohérence mutuelle est satisfaite. Alors, si les colonnes sont normalisées comme précédemment le choix $\lambda \approx \sqrt{\frac{\log p}{n}}$ permet d'assurer avec grande probabilité que le support de $\hat{\theta}_\lambda$ est contenu dans celui de θ^* .*

5.3.6 Extensions/Remarques

Remarques. En résumé, les principaux avantages des méthodes pénalisées sont :

- Grande dimension : ces méthodes fonctionnent dans les cas où le nombre d'individus est inférieur au nombre de variables ($n < p$).
- Sélection parcimonieuse : cet avantage concerne principalement le Lasso qui permet de sélectionner un sous-ensemble restreint de variables (dépendant du paramètre λ). Cette sélection restreinte permet souvent de mieux interpréter un modèle.
- Consistance de la sélection : d'après ce qu'on a vu, lorsque le vrai vecteur est creux, le Lasso a en théorie la capacité à sélectionner les variables associées.

En revanche, le Lasso possède (bien entendu) certaines limites :

- Les fortes corrélations : si des variables sont fortement corrélées entre elles et qu'elles sont importantes pour la prédiction, le lasso en privilégiera une au détriment des autres. Un autre cas, où les corrélations posent problème, est quand les variables d'intérêts sont corrélées avec d'autres variables. Dans ce cas, la consistance de la sélection du lasso n'est plus assurée.
- La très grande dimension : lorsque notamment la dimension est trop élevée, l'hypothèse “ β creux” n'a plus vraiment de sens et le Lasso n'est alors plus en mesure de sélectionner les variables en jeu,

mais cette dernière remarque semble évoquer un problème peu surmontable.

Extensions. Les méthodes de type LASSO ont été très développées. On trouvera dans la littérature le *Fused-Lasso* qui prend en compte la spatialité des variables. Plus précisément, dans certains problèmes, les variables peuvent être liées à des zones géographiques proches. On peut penser au traitement d'images ou à l'analyse en différents points de l'activité électrique du cerveau. Dans ce cas, on peut penser que l'influence de deux variables “proches” sera similaire. Ainsi, il est alors naturel de pénaliser la fonction de perte de manière à favoriser les solutions β dont les coordonnées “proches” génèrent des β_j “proches”.

On minimisera alors par exemple

$$\frac{\|\mathbf{Y} - \mathbf{X}\theta\|^2}{2n} + \lambda_1 \|\theta\|_1 + \lambda_2 \sum_{j=1}^{p-1} |\theta_j - \theta_{j-1}|.$$

Dans un autre genre, on trouvera aussi le *Group-Lasso* qui fournit une sélection parcimonieuse de groupes de variables (si possible, fournies auparavant). Ce type d'extension est particulièrement au cas de variables qui sont de différents types (omiques d'un côté, “environnementales” de l'autre...).

Enfin, les méthodes de sélection parcimonieuse se sont aussi étendues à d'autres types de problèmes. On pensera par exemple à l'analyse en composantes principales qui possède sa version “sparse” (afin de pallier le problème évoqué dans le chapitre 2).

Chapitre 6

Machines à Vecteurs Supports

6.1 Introduction

Les Support Vector Machines souvent traduits par “Séparateur à Vaste Marge” (SVM) sont une classe d’algorithmes d’apprentissage initialement définis pour la discrimination c'est-à-dire la prévision d'une variable qualitative binaire. Ils ont été ensuite généralisés à la prévision d'une variable quantitative. Dans le cas de la discrimination (classification), ils sont basés sur la recherche de l’hyperplan de marge optimale qui, lorsque c'est possible, classe ou sépare correctement les données tout en étant le plus éloigné possible de toutes les observations. Le principe est de trouver un classifieur, ou une fonction de discrimination, dont la capacité de généralisation (qualité de prévision) est la plus grande possible. L’objectif est ainsi de fabriquer une règle de décision “robuste” par séparation de l'espace (en deux parties dans le cadre binaire).

Les séparateurs à vaste marge ont été développés dans les années 1990 à partir des considérations théoriques de Vladimir Vapnik sur le développement d'une théorie statistique de l'apprentissage : la théorie de Vapnik-Tchervonenkis. Ils ont rapidement été adoptés pour leur capacité à travailler avec des données de grandes dimensions, le faible nombre d'hyperparamètres, leurs garanties théoriques, et leurs bons résultats en pratique. Les SVM apparaissent dans de nombreux domaines d’application (classification d’images, reconnaissance de caractères, médecine,...).

Avant d’aller plus loin, les points importants à retenir pour la suite sont les suivants :

1. La construction de la fonction de prédiction se fait via la résolution d'un problème d'optimisation quadratique que l'on détaillera dans le cas des données *linéairement séparables*.
2. A l'aide de l'introduction de *noyaux*, les SVM pourront s'appliquer aussi à des données *non linéairement séparables*.

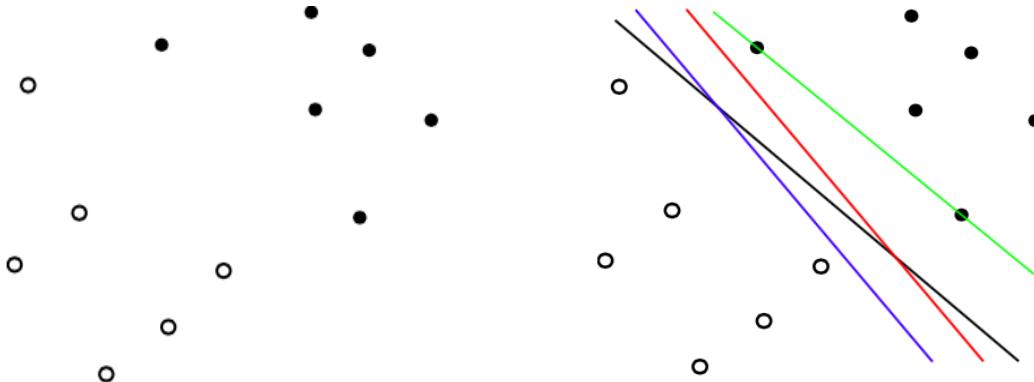
La suite de ce chapitre est donc structurée comme suit :

- Présentation en détail du cadre des “Données Linéairement Séparables”
- Extension au cadre des données “presque linéairement séparables” (*i.e.* où la séparation par un hyperplan reste un choix efficace)
- Présentation des machines à noyau permettant d’étendre largement le principe à des données non linéairement séparables.

6.2 Linéairement/non linéairement séparables

Question : Intuitivement, comment sépareriez-vous ces points ?

Si l'on choisit de séparer par une droite comme sur la figure de droite, laquelle choisir ?



Principe : Dans ce cadre linéairement séparable, le principe des SVM est de fabriquer l'**hyperplan** maximisant la **marge**, *i.e.* la distance aux observations les plus proches (voir Figure 6.1).

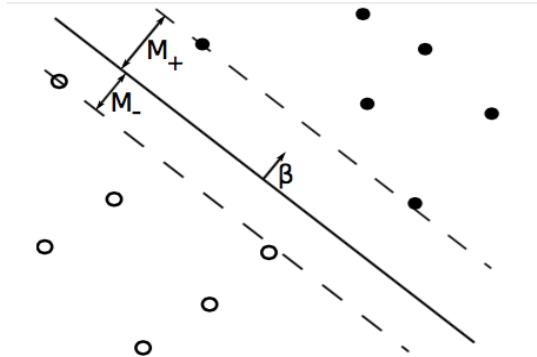


Figure 6.1: Hyperplan et marge

Comme indiqué plus haut, ce principe peut s'étendre à des cadres non linéairement séparables (machines à noyau). On reviendra sur ce type d'exemple plus tard mais afin de donner un aperçu de ces méthodes, voici ci-dessous (Figure 6.2 un exemple de classification (hautement) non linéaire traité par SVM (**à noyau**) :

6.3 Classificateurs linéaires à Vaste Séparateur de Marge

6.3.1 Construction du SVM linéaire

But : Trouver un classifieur linéaire (hyperplan) qui va séparer les données et maximiser la distances entre ces 2 classes.

Vecteurs Supports : la détermination de l'hyperplan est réalisée (uniquement) via les *vecteurs supports* qui sont les points de chaque classe les plus proches de l'hyperplan).

6.3.2 Hyperplans

- Un hyperplan affine \mathcal{H} de \mathbb{R}^p est un sous-espace affine de dimension $p - 1$.
- Ainsi, il est caractérisé par un point M_0 appartenant à \mathcal{H} et un vecteur normal β :

$$\mathcal{H} = \{M \in \mathbb{R}^p, \langle \beta, \overrightarrow{M_0 M} \rangle = 0\}$$

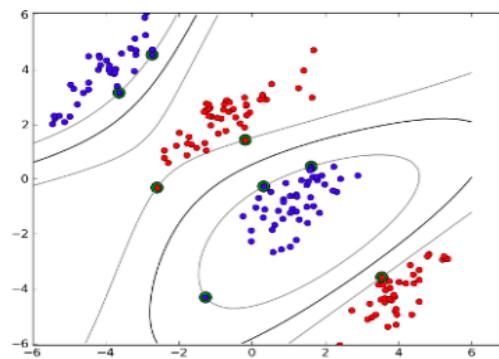
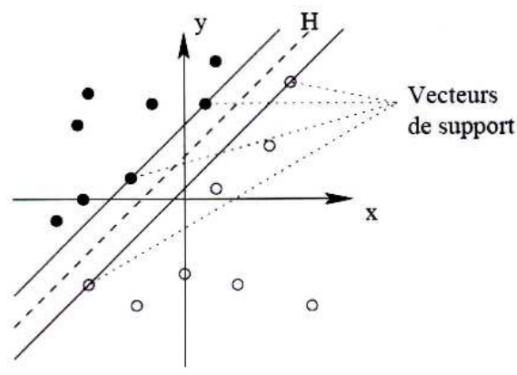


Figure 6.2: Exemple (multi-classes) non linéaire.



- Son équation s'écrit :

$$\langle \beta, x \rangle + \beta_0 = 0.$$

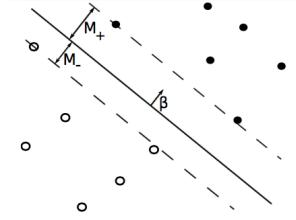
- L'hyperplan sépare l'espace en deux parties \mathcal{C}_+ et \mathcal{C}_- telles que $\mathcal{C}_+ = \{x, \langle \beta, x \rangle + \beta_0 > 0\}$ et $\mathcal{C}_- = \{x, \langle \beta, x \rangle + \beta_0 < 0\}$

6.3.3 Hyperplans de séparation

Supposons que $\{(x_1, y_1), \dots, (x_n, y_n)\}$ est un échantillon tel que $(x_i, y_i) \in \mathbb{R}^p \times \{-1, 1\}$. On dit que \mathcal{H} d'équation $\langle \beta, x \rangle + \beta_0 = 0$ est un **hyperplan de séparation** pour l'échantillon si

$$\forall i \in \{1, \dots, N\}, \quad \langle \beta, x_i \rangle + \beta_0 \begin{cases} > 0 & \text{si } y_i = 1 \\ < 0 & \text{si } y_i = -1 \end{cases}$$

i.e. $\forall i \in \{1, \dots, N\}, y_i (\langle \beta, x_i \rangle + \beta_0) > 0$.



6.3.4 Définition du SVM (dans le cas linéairement séparable)

Principe : Parmi les hyperplans de séparation de l'échantillon d'apprentissage, on veut choisir celui qui a la plus **vaste marge**, i.e. telle que $\min_{i=1}^N d(x_i, \mathcal{H})$ est maximal. On fait ici l'hypothèse que celui-ci existe.

Remarque : Lorsque $\|\beta\| = 1$, $d(x_i, \mathcal{H}) = |\langle \beta, x_i \rangle + \beta_0| = y_i (\langle \beta, x_i \rangle + \beta_0)$ (Exercice). Ainsi, le problème de détermination du meilleur hyperplan pour la marge se définit comme suit :

Hyperplan de séparation optimal : solution du problème

$$\left\{ \begin{array}{l} \max_{\beta, \beta_0} M \text{ sous la contrainte} \\ \forall i \in \{1, \dots, N\}, \quad y_i (\langle \beta, x_i \rangle + \beta_0) \geq M \text{ et } \|\beta\| = 1. \end{array} \right. \quad (6.3.1)$$

6.3.5 Reformulation du problème d'optimisation

Si l'on ne constraint plus β à être de norme 1, dans ce cas :

$$d(x_i, \mathcal{H}) = \frac{y_i (\langle \beta, x_i \rangle + \beta_0)}{\|\beta\|}$$

de sorte que la contrainte s'écrit

$$y_i (\langle \beta, x_i \rangle + \beta_0) \geq M \|\beta\|.$$

Remarque : On peut choisir $C = \|\beta\|$ en fonction de M ! Prenons $C = 1/M$. Le problème d'optimisation peut alors se reformuler :

$$\left\{ \begin{array}{l} \max_{\beta, \beta_0} M \text{ sous la contrainte} \\ \forall i \in \{1, \dots, N\}, \quad y_i (\langle \beta, x_i \rangle + \beta_0) \geq 1 \text{ et } \|\beta\| = \frac{1}{M}. \end{array} \right. \quad (6.3.2)$$

Mais dans ce cas,

$$\max_{\beta, \beta_0} M = \max_{\beta, \beta_0} \frac{1}{\|\beta\|} = \min_{\beta, \beta_0} \|\beta\|$$

de sorte que l'on peut finalement écrire le problème sous la forme :

$$\begin{cases} \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 \text{ sous la contrainte} \\ \forall i \in \{1, \dots, N\}, \quad y_i(\langle \beta, x_i \rangle + \beta_0) \geq 1. \end{cases} \quad (6.3.3)$$

Il s'agit d'un *problème d'optimisation quadratique* sous contraintes d'inégalités affines (c'est en particulier un problème (fortement) convexe). Ce type de problème se résout assez bien en petite dimension via des méthodes d'optimisation convexe (Gauss-Seidel/gradient projeté/point intérieur...) mais lorsque p augmente, on lui préfère généralement le "problème dual" en introduisant le lagrangien :

$$L(\beta, \beta_0, \alpha) = \frac{1}{2} \|\beta\|^2 - \sum_{i=1}^N \alpha_i (y_i(\langle \beta, x_i \rangle + \beta_0) - 1).$$

Plus précisément, le premier problème est dit *primal* tandis que sa reformulation via le passage au Lagrangien est dite *duale*. Les α_i sont appelées *variables duales*.

6.3.6 Problème Dual

Le problème dual consiste à déterminer $(\beta^*, \beta_0^*, \alpha^*)$ tel que

$$L(\beta^*, \beta_0^*, \alpha^*) = \max_{\alpha_1, \dots, \alpha_N \geq 0} \min_{\beta, \beta_0} L(\beta, \beta_0, \alpha).$$

Pour cela, on minimise d'abord en (β, β_0) puis on maximise en α .

- Points critiques de $(\beta, \beta_0) \mapsto L(\beta, \beta_0, \alpha)$ solutions de $\partial_\beta L = 0$ et $\partial_{\beta_0} L = 0$ ce qui donne :

$$\begin{cases} \beta(\alpha) = \sum_{i=1}^N \alpha_i y_i x_i \\ \sum_{i=1}^N \alpha_i y_i = 0. \end{cases}$$

- Si on réinjecte ces conditions dans L , on trouve (voir TD pour détails) :

$$\tilde{L}(\alpha) = L(\beta(\alpha), \beta_0(\alpha), \alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle.$$

Le problème dual s'écrit alors :

$$\begin{cases} \min_{\alpha \in (\mathbb{R}^+)^N} \frac{1}{2} \alpha^T G \alpha - \langle \alpha, \mathbf{e} \rangle \\ \text{avec } \langle \mathbf{y}, \alpha \rangle = 0. \end{cases}$$

où G est la matrice définie par $G_{i,j} = y_i y_j \langle x_i, x_j \rangle$, $\mathbf{e} = (1, \dots, 1)^T$ et $\mathbf{y} = (y_1, \dots, y_n)^T$. Il s'agit à nouveau d'un problème quadratique mais cette fois en dimension n , ce qui rend l'approche particulièrement intéressante numériquement lorsque n est petit devant p . En résumé, on a donc deux formulations (primale et duale) du problème d'optimisation qui du point de vue numérique sont concurrentes. En pratique, la seconde approche est la plus couramment utilisée (pour la raison qui suit par exemple).

Remarque 6.3.1. Un point important pour la suite : le problème dual s'exprime sous la forme d'un problème d'optimisation ne dépendant des x_i que par le produit scalaire ! Cette remarque jouera un rôle fondamental dans la section sur les SVMs à noyau.

6.3.7 Calcul de β^* , β_0^*

On suppose maintenant que l'on a réussi à calculer α^* défini par

$$\alpha^* = (\alpha_1^*, \dots, \alpha_N^*) = \operatorname{Argmax}_{\alpha \in \mathbb{R}^N} \tilde{L}(\alpha).$$

Comment retrouver ensuite l'équation de l'hyperplan de séparation optimal ?

- Par construction, $\beta^* = \sum_{i=1}^N \alpha_i^* y_i x_i$.
- Pour β_0^* , on s'appuie sur les *conditions complémentaires de KKT* qui garantissent que

$$\forall i \in \{1, \dots, N\}, \quad \alpha_i^*(y_i (\langle \beta^*, x_i \rangle + \beta_0^*) - 1) = 0.$$

En d'autres termes, soit le coefficient α_i^* est nul, soit le point considéré atteint la frontière. Dans ce cas, c'est un *vecteur support*. On peut donc retrouver β_0^* en considérant un (x_i, y_i) vecteur support (il en existe au moins 2). On a alors

$$\beta_0^* = y_i - \langle \beta^*, x_i \rangle.$$

Remarque 6.3.2. ▷ En prenant un vecteur support “positif” et un “négatif” (voir TD), on peut aussi obtenir la formule :

$$\beta_0^* = -\frac{1}{2} \left(\min_{y_i=1} \langle \beta^*, x_i \rangle + \min_{y_i=-1} \langle \beta^*, x_i \rangle \right).$$

▷ L'hyperplan solution ne s'exprime qu'à partir des vecteurs supports. En effet, dès que le vecteur support, il ne *sature* pas les conditions de KKT et $\alpha_i = 0$. C'est donc en quelque sorte une compression de l'information. Pour terminer, il est néanmoins important de noter que dans les SVMs à marge flexible (à venir), le nombre de vecteurs supports sera plus important.

6.3.8 Règle de Décision

A l'issue de ces calculs, on a donc fabriqué un **prédicteur** \hat{f} de la classe d'un point x :

$$\hat{f}(x) = \operatorname{sgn}(\langle \beta^*, x \rangle + \beta_0^*) = \operatorname{sgn}\left(\sum_{i=1}^N y_i \alpha_i^* \langle x_i, x \rangle + \beta_0^*\right).$$

Remarque 6.3.3. ▷ En lien avec la remarque 6.3.1, on constate ci-dessus que la règle de décision ne dépend des variables d'entrée que par le produit scalaire... ▷ Marge ? Pour rappel, l'idée était de fabriquer l'hyperplan de séparation ayant la marge maximale. Si l'on revient à la construction du problème d'optimisation, on constate que la marge M^* associée à l'hyperplan optimal est donnée par :

$$M^* = \frac{1}{\|\beta^*\|}.$$

▷ Comme d'habitude, si l'on veut étudier l'erreur de classification de cet algorithme de prévision, on fabriquera l'hyperplan de séparation sur un sous-ensemble d'entraînement et on calculera l'erreur de classification sur un échantillon test. ▷ En pratique, les données sont rarement linéairement séparables. On va donc dans la suite voir comment injecter de la “souplesse” dans cet algorithme.

6.4 SVM linéaires à marge flexible

6.4.1 SVM linéaire pour données non linéairement séparables ?

Question : Comment fabriquer un classifieur linéaire basé sur la même approche lorsqu'il n'existe pas d'hyperplan de séparation ? (ou lorsqu'il en existe un mais qu'il est clairement trop dépendant de données *outliers*)



Réponse : Fabriquer un SVM linéaire à marge souple : La condition $y_i(\langle \beta, x_i \rangle + \beta_0) \geq 1$ est remplacée par $y_i(\langle \beta, x_i \rangle + \beta_0) \geq 1 - \xi_i$ avec $\xi_i \geq 0$.

- Si $\xi_i \in]0, 1]$, le point est bien classé mais “sous” la marge du SVM précédent.
- Si $\xi_i > 1]$, le point est mal classé (voir figure en haut à droite).

6.4.2 Problème d'optimisation associé (aux marges flexibles)

- Si l'on fait trop souvent recours à $\xi_i > 0$ voir grand, cela signifie que le classifieur qui serait créé n'est pas très bon.
- Ainsi, l'idée est de pénaliser la variable ξ_i de la manière suivante : on résout ici

$$\begin{cases} \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^n \xi_i \text{ sous les contraintes} \\ \forall i \in \{1, \dots, N\}, \quad y_i(\langle \beta, x_i \rangle + \beta_0) \geq 1 - \xi_i \\ \xi_i \geq 0. \end{cases} \quad (6.4.1)$$

où C est un paramètre à régler (*tuning parameter*). Il règle le taux de souplesse. Plus C est grand, moins l'on “tolère” les erreurs de classification. On est donc dans un paradigme “Biais-Variance” : il faut trouver un compromis entre le surapprentissage qui guette lorsque C est petit et le fort biais lorsque C est grand (Le coefficient C pourra se choisir par validation croisée).

6.4.3 Résolution du problème aux marges flexibles

Là encore, l'idée est de passer à la formulation duale. Le Lagrangien s'écrit ici :

$$L(\beta, \beta_0, \xi, \alpha, \mu) = \frac{1}{2} \|\beta\|^2 + C \sum_{i=1}^N \xi_i - \sum_{i=1}^N \alpha_i (y_i(\langle \beta, x_i \rangle + \beta_0) - (1 - \xi_i)) - \sum_{i=1}^N \mu_i \xi_i.$$

Le deuxième terme permet de gérer la contrainte $\xi_i \geq 0$, $i \in \{1, \dots, N\}$. A nouveau, on peut écrire les conditions KKT :

$$\text{Stationnarité : } \begin{cases} \partial_\beta L = 0 \implies \beta = \sum_{i=1}^N \alpha_i y_i x_i, \\ \partial_{\beta_0} L = 0 \implies \sum_{i=1}^N \alpha_i y_i = 0, \\ \partial_\xi L = 0 \implies \alpha_i = C - \mu_i, \end{cases} \quad (6.4.2)$$

de sorte qu'en réinjectant dans L , on a encore à maximiser

$$\tilde{L}(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

sous contrainte $\sum_{i=1}^N \alpha_i y_i = 0$ et $0 \leq \alpha_i \leq C$.

6.4.4 Vecteurs supports

Notons $(\alpha^*, \beta^*, \beta_0^*, \xi^*, \mu_i^*)$ la solution du problème. Les conditions complémentaires de KKT donnent

$$\forall i \in \{1, \dots, N\}, \begin{cases} \alpha_i^* (\langle \beta^*, x_i \rangle + \beta_0^*) - (1 - \xi_i^*) = 0, \\ \mu_i^* \xi_i^* = 0 \iff \xi_i^* (\alpha_i^* - C) = 0. \end{cases}$$

Les vecteurs supports sont à nouveau les x_i pour lesquels $\alpha_i^* > 0$. Néanmoins, on a deux classes :

1. Ceux pour lesquels $\xi_i^* = 0$ (on ne fait pas usage de la flexibilité : vecteur supports du précédent modèle).
2. Ceux pour lesquels $\xi_i^* > 0$. Dans ce cas, $\alpha_i^* = C$. Ces points sont les nouveaux vecteurs supports issus de la flexibilisation du modèle. On peut remarquer qu'ils sont plus “suspects”, *i.e.* leur présence dans la construction du prédicteur est la plus susceptible d'être remise en cause.

6.5 SVM non linéaires

6.5.1 Principe des SVMs à noyau

Constat : Les SVMs linéaires constituent des classificateurs linéaires robustes. Néanmoins, la contrainte de linéarité est clairement restreignante. Dans un cadre où les points ont une répartition dans l'espace ne permettant absolument pas une séparation par un hyperplan, que faire ?

Question : Peut-on adapter les approches précédentes pour fabriquer des *variétés* de dimension $p - 1$ permettant de séparer les points de la même manière que les SVMs linéaires ?

Réponse : Oui, en introduisant des noyaux. L'idée est de déformer l'espace ou plus précisément les vecteurs d'entrée x_i dans un nouvel espace appelé **espace de représentation (feature space)**. Cette idée est due à Boser, Guyon et Vapnik (1992). **Exemple.** Considérons $p = \mathbb{R}^2$ et l'ensemble de points de la figure 6.3. L'ensemble de ces points n'est clairement pas linéairement séparable. Néanmoins, si l'on note

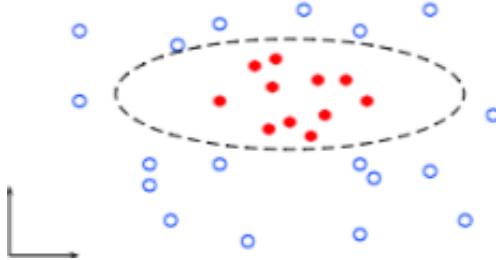


Figure 6.3: Données séparables par une ellipse

$\phi(x) = \phi(a, b) = (a^2, b^2, a, b)$, alors, l'ensemble des points $\{\phi(x_i), y_i, i = 1, \dots, N\}$ l'est dans $E = \mathbb{R}^4$.

Problème d'optimisation associé (et astuce du noyau)

Considérons $\phi : \mathbb{R}^p \mapsto E$ où E est l'*espace de représentation* (“feature space”) et supposons que E est muni d'un produit scalaire $\langle \cdot, \cdot \rangle_E$. Dans ce nouvel espace, le problème d'optimisation dual consiste à déterminer la solution de :

$$\begin{cases} \min_{\alpha \in (\mathbb{R}^+)^N} \frac{1}{2} \alpha^T \tilde{G} \alpha - \langle \alpha, \mathbf{e} \rangle \\ \text{avec } \langle \mathbf{y}, \alpha \rangle = 0 \text{ et } 0 \leq \alpha_i \leq C. \end{cases}$$

où G est la matrice définie par $G_{i,j} = y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$, $\mathbf{e} = (1, \dots, 1)^T$ et $\mathbf{y} = (y_1, \dots, y_n)^T$.

En d'autres termes, partant de points dans l'espace de départ \mathbb{R}^p , on les plonge dans un espace plus gros dans lesquels ils peuvent être séparés par un hyperplan (dans un sens flexible) et donc bénéficier de l'algorithme linéaire présenté plus haut. En suivant le raisonnement précédent (dans le cas flexible), le problème d'optimisation peut encore être résolu par la même méthode et mène au meilleur hyperplan dans E pour les points $\{\phi(x_i), y_i, i = 1, \dots, N\}$. La règle de décision associée est :

$$\hat{f}(x) = \text{sgn} \left(\sum_{i=1}^N y_i \alpha_i^* \langle \phi(x_i), \phi(x) \rangle_E + \beta_0^* \right).$$

IMPORTANT : On n'a pas besoin de fabriquer ϕ explicitement puisqu'on ne fait usage que de $\langle \phi(x), \phi(x') \rangle$. On a simplement besoin de $k(x, x') = \langle \phi(x), \phi(x') \rangle$. Ce point est souvent appelé "astuce du noyau" ou "kernel trick" car il permet de plonger les données dans un espace quelconque sans même se soucier de sa structure. La seule connaissance du noyau est suffisante.

Noyau : définition

La fonction k est appelé un noyau :

Définition 6.5.1. Une fonction $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ telle que $k(x, x') = \langle \phi(x), \phi(x') \rangle_E$ pour une fonction $\phi : \mathcal{X} \rightarrow E$ est appelée un noyau.

Par exemple, $k(x, x') = \langle x, x' \rangle_{\mathbb{R}^2}^2$ est un noyau. Un espace E associé est \mathbb{R}^3 avec la fonction ϕ définie par $\phi(x) = \phi(a, b) = (a^2, \sqrt{2}ab, b^2)$.

Proposition 6.5.2. (*Condition de Mercer*). Si la fonction $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ est continue symétrique et si pour tout $n \geq 2$ et tous points x_1, \dots, x_n , $k(x_i, x_j)_{1 \leq i, j \leq n}$ est définie positive (sur \mathbb{R}^n), alors il existe un espace de Hilbert E et une fonction $\phi : \mathcal{X} \rightarrow E$ tels que $k(x, x') = \langle \phi(x), \phi(x') \rangle_E$. On dit alors que k est un noyau positif et l'espace E est appelé espace à noyau reproduisant.

Définition 6.5.3. La matrice $K = (k(x_i, x_j))_{1 \leq i, j \leq n}$ est appelée matrice de Gram.

Exemples de Noyaux

Supposons que $\mathcal{X} = \mathbb{R}^p$. Donnons quelques exemples de noyaux usuels.

- Noyau polynomial de degré r : $k(x, x') = (c_0 + \gamma \langle x, x' \rangle)^r$ (où $\langle x, x' \rangle$ est le produit scalaire usuel sur \mathbb{R}^p).
- Noyau Gaussien : $k(x, x') = \exp(-\gamma \|x - x'\|^2)$.
- Noyau de Laplace : $k(x, x') = \exp(-\gamma \|x - x'\|)$.
- Noyau Sigmoïde : $k(x, x') = \tanh(\gamma \langle x, x' \rangle + c_0)$.

Remarques : Les noyaux Gaussiens et Laplaciens sont dits radiaux (dépendent de la norme de la différence) tandis que les deux autres basés sur le produit scalaire sont dits projectifs.

L'espace E n'est pas précisé. Dans le cas gaussien, l'espace E est un espace de fonctions (en particulier de dimension infinie).

Dans le cadre à noyau, l'idée est de plonger la variable x dans un espace de Hilbert, appelé espace des représentations dans lequel les points sont linéairement séparables puis d'utiliser les méthodes développées dans le cadre linéaire. Comme pour le thème des régressions pénalisées, le sujet est très large et de nombreuses extensions/variations (selon les contextes), résultats théoriques, existent sur le sujet. Parmi ces extensions, on notera en particulier que les SVM s'appliquent dans un cadre **multi-classes** et dans le cadre de la **régression**. Enfin, les méthodes à noyau présentées plus haut ne se limitent d'ailleurs pas au cadre de la SVM. On les trouvera également dans d'autres contextes où l'on cherche à étendre des méthodes "linéaires" à un cadre non linéaire.

6.5.2 Autres méthodes à noyau

Pour terminer ce chapitre, évoquons donc de manière un peu plus générale les méthodes dites à noyau. De manière un peu imprécise, les méthodes à noyau peuvent s'appliquer dans le contexte suivant :

Théorème 6.5.4. *Tout algorithme pour vecteurs qui puisse ne s'exprimer qu'en termes de produits scalaires entre vecteurs peut être effectué implicitement dans un espace de Hilbert en remplaçant chaque produit scalaire par l'évaluation d'un noyau défini positif sur un espace quelconque.*

Voici ci-dessous un autre exemple où l'on peut appliquer ce “théorème”.

Kernel PCA

Trouver la k -ième direction de l'ACP, c'est résoudre le problème suivant : déterminer

$$w = \operatorname{Argmin}_{w \perp (w_1, \dots, w_{k-1})} \frac{1}{n} \sum_{i=1}^n \frac{\langle x_i, w \rangle^2}{\|w\|^2}.$$

C'est donc bien un problème qui s'exprime en fonction du produit scalaire. D'un point de vue fonctionnel, résoudre l'ACP revient à déterminer

$$f = \operatorname{Argmin}_{f \in \mathcal{H}, f \perp (f_1, \dots, f_{k-1})} \frac{1}{n} \sum_{i=1}^n \frac{f(x_i)^2}{\|f\|_{\mathcal{H}}^2}. \quad (6.5.1)$$

où \mathcal{H} est l'ensemble des fonctions $f(x) = \langle w, x \rangle$ muni du produit scalaire $\langle f, \tilde{f} \rangle_{\mathcal{H}} = \langle w, \tilde{w} \rangle_{\mathbb{R}^d}$. On peut transposer ce problème à un espace de fonctions différent via l'astuce du noyau. Par exemple, si

$$k(x, y) = \langle x, y \rangle^2.$$

Alors, on peut montrer que l'espace \mathcal{H} associé est l'ensemble des fonctions f de la forme

$$f_S(x) = x^T S x$$

où S est une matrice symétrique, muni du produit scalaire

$$\langle f_{S_1}, f_{S_2} \rangle_{\mathcal{H}} = \langle S_1, S_2 \rangle_F = \sum_{i,j} S_1(i, j) S_2(i, j).$$

Par le *théorème du représentant*, la solution f_k^* du problème de minimisation (6.5.1) sur l'espace \mathcal{H} (associé au noyau k) s'écrit nécessairement

$$f_k^* = \sum_{i=1}^n \alpha_i k_{x_i}$$

où $k_{x_i} : x \mapsto k(x_i, x)$. On a $\langle k_{x_i}, k_{x_\ell} \rangle_{\mathcal{H}} = k(x_i, x_\ell)$. Ainsi,

$$\|f_k^*\|_{\mathcal{H}}^2 = \sum_{i,\ell} \alpha_i \alpha_\ell k(x_i, x_\ell) = \alpha^T \mathbf{K} \alpha$$

où $\mathbf{K} = (k(x_i, x_\ell))_{i,\ell}$ et

$$\sum_{i=1}^n f_k^*(x_i)^2 = \sum_{i=1}^n (\sum_{i=1}^n \alpha_i k(x_i, x_k))^2 = \dots = \alpha^T \mathbf{K}^2 \alpha$$

de sorte que le vecteur α_k^* des coefficients associés à f_k^* satisfait

$$\alpha^* = \operatorname{Argmin}_{\alpha} \frac{\alpha^T \mathbf{K}^2 \alpha}{\alpha^T \mathbf{K} \alpha}.$$

sous la contrainte $(\alpha_k^*)^T \mathbf{K} \alpha_j^* = 0$ pour tout $j \leq k-1$ où les α_j^* sont les coefficients des fonctions f_j^* , $j \in \{1, \dots, k-1\}$, autrement dit, les $k-1$ premières directions de l'ACP à noyau.

On diagonalise alors \mathbf{K} (la matrice de Gram) : $K = U \Delta U^T$ avec $\Delta_1 \geq \Delta_2 \dots \geq \Delta_n \geq 0$. Ensuite, on pose $\beta_k = \mathbf{K}^{\frac{1}{2}} \alpha$ et on cherche

$$\operatorname{Argmin}_{\beta} \beta^T \mathbf{K} \beta \quad (= \operatorname{Argmin}_{\alpha} \frac{\alpha^T \mathbf{K}^2 \alpha}{\alpha^T \mathbf{K} \alpha}).$$

sous les contraintes $\beta_k^T \beta_j = 0$ pour $j \leq k-1$ et $\beta_k^T \beta_k = 1$. Ce sont exactement les vecteurs propres de \mathbf{K} . Il suffit alors d'inverser pour obtenir les α_k . On trouve

$$\alpha_k = \frac{1}{\Delta_k} \beta_k.$$

Finalement, la projection d'un point x sur la k -ième direction, *i.e.* pour son image $\Phi(x)$ s'écrit :

$$\langle \Phi(x), f_k \rangle_{\mathcal{H}} = \sum_{i=1}^n \alpha_{i,k} \langle \Phi(x), \Phi(x_i) \rangle_{\mathcal{H}} = \sum_{i=1}^n \alpha_{i,k} K(x, x_i).$$

On peut alors remplacer les points dans la base $\{f_1, \dots, f_d\}$.

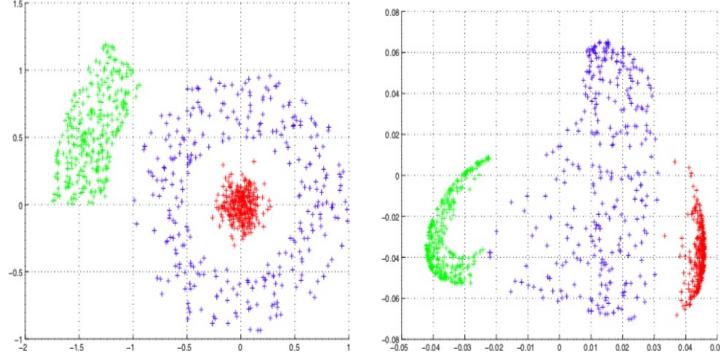


Figure 6.4: ACP à noyau (gaussien)

D'autres méthodes à noyau seront expérimentées en TP.

Chapitre 7

Réseaux de Neurones et Deep Learning

7.1 Introduction

Les réseaux de neurones dits artificiels ou formels ont d'abord été initialement développés pour modéliser l'activité du cerveau. On trouve à la fin des années 1950 des travaux des neurologues Warren McCulloch et Walter Pitts intitulé "What the frog's eye tells the frog's brain" dans laquelle ils tentent de modéliser l'activité d'un ensemble de neurones.

Le fonctionnement d'un neurone seul est alors modélisé à l'aide une fonction de transfert qui prend en entrée une combinaison linéaire d'informations et qui renvoie un signal lorsque l'information d'entrée dépasse un certain seuil. Le fonctionnement du cerveau peut alors être compris comme un empilement de structures simples (voir Figures 7.1 et 7.2 pour des illustrations de neurones sur un plan biologique ou formel) et que l'empilement

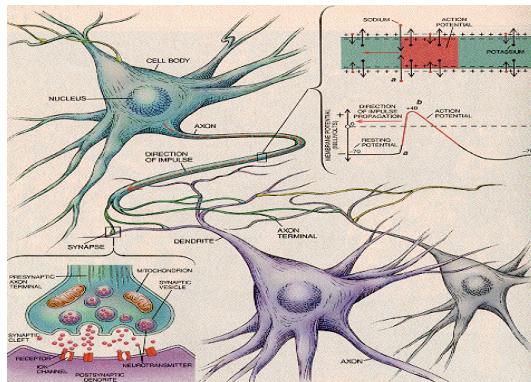


Figure 7.1: Neurones biologiques, axones et dendrites, figure issue de <http://www.lacim.uqam.ca/~chauve/Enseignement/BIF7002/Rapports/Simon-Beaulne/neurones.htm>

de ces briques élémentaires peut modéliser un système complexe. On appelle *réseau de neurones* une telle structure, constituée de plusieurs *couches* constituées de plusieurs neurones. La terminologie *Deep Learning* fait quant à elle référence à la notion d'apprentissage profond qui peut être comprise comme une structure de réseaux de neurones constituée d'un nombre important de couches. La notion d'apprentissage profond fait aussi référence à la structure même des couches allant du neurone formel à des couches *convolutionnelles* voire *multi-convolutionnelles*. L'idée générale consiste à adapter la structure des réseaux au problème considéré.

D'un point de vue général, on peut considérer à ce stade les réseaux de neurones comme une famille de fonctions ayant une bonne capacité d'approximation de phénomènes non linéaires. S'il existe aujourd'hui peu de résultats

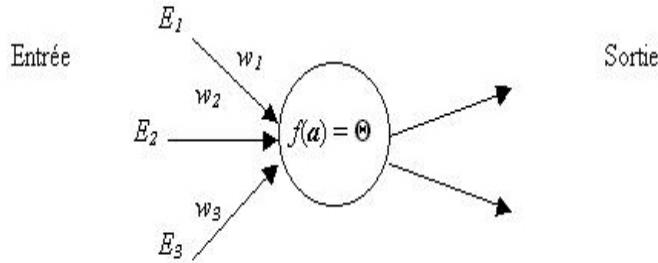


Figure 7.2: Neurone formel : figure issue de <http://www.lacim.uqam.ca/~chauve/Enseignement/BIF7002/Rapports/Simon-Beaulne/neurones.htm>

quantifiant mathématiquement ces qualités, les résultats numériques sont eux, suffisamment remarquables pour expliquer le succès actuel de ces structures.

On peut citer en premier lieu la reconnaissance d’images. En 2012, [7] écrase le Challenge *Image-Net Large Scale Visual Recognition Challenge* (INLSVRC) en obtenant moins de 16% d’erreurs contre 26% au minimum pour ses concurrents. Le modèle utilisé, appelé *Alexnet*, est un réseau de neurones à *convolution* constitué de 9 couches et près de 60 millions de paramètres à régler.

Viennent ensuite la reconnaissance de textes, la reconnaissance vocale, la robotique, la modélisation du jeu de Go.... Il est important de noter qu’au delà de la structure des réseaux, la force de ces méthodes d’apprentissage repose généralement sur une combinaison d’apprentissages supervisé et non supervisé. Dans le jeu de Go par exemple, il est facile de comprendre que l’on peut pré-entraîner le modèle sur des données existantes puis améliorer ses résultats en confrontant des “joueurs” obtenus par simulation (en modifiant légèrement les paramètres). La confrontation de ces modèles fabriqués par simulation permet alors d’améliorer les performances sans faire intervenir de données réelles nouvelles. Notons que ces notions sont connectées à celle d’apprentissage par renforcement (*Reinforcement Learning*).

Comme indiqué plus haut, différentes types de réseaux de neurones sont à distinguer :

1. Perceptron Multicouches (modèle “historique”)
2. CNN : Convolutional Neural Networks (Réseaux de Neurones à Convolution adaptés pour le traitement d’images en particulier)
3. RNN : Recurrent Neural Networks (Réseaux de neurones récurrents, adaptés pour les données séquentielles telles que les données textuelles par exemple).
4. GAN : Generative Adversarial Neural Networks (Réseaux de Neurones Antagonistes Génératifs, utilisés pour la génération d’images et plus généralement pour la “création” artificielle)
5.

7.2 Réseaux de Neurones Artificiels

Un réseau de neurones artificiel est une famille de fonctions paramétriques que l’on notera

$$\{f_\theta : \mathbb{R}^p \mapsto \mathbb{R}, \theta \in \Theta\},$$

où $\Theta \subset \mathbb{R}^m$ (m grand en général). Remarquons que l'on fait ici l'hypothèse que l'entrée x est un vecteur de \mathbb{R}^p et que la sortie $y = f_\theta(x)$ est un réel. Cette hypothèse, en particulier sur la sortie n'est pas adaptée à tous les contextes mais nous conserverons dans la suite ce formalisme pour simplifier.

Cadre : Les réseaux de neurones seront envisagés pour des problèmes de régression ou de classification.

7.2.1 Neurone Artificiel

Définition

On s'intéresse à la structure d'un neurone dit artificiel ou formel qui a vocation à tenter de modéliser le fonctionnement d'un neurone biologique basé sur :

- des synapses : points de connexion entre neurones,
- des dendrites : entrées du neurones,
- des axones : sorties du neurone vers d'autres neurones,
- un noyau : activateur des sorties en fonction des stimulations en entrée.

Définition 7.2.1. Un neurone artificiel (d'indice j) est une fonction h de la variable d'entrée $x = (x_1, \dots, x_d)$ construite à l'aide de poids $w = (w_1, \dots, w_d)$, d'un biais $b \in \mathbb{R}$ et d'une fonction d'activation ϕ , définie par :

$$\forall x \in \mathbb{R}^d, \quad h(x) = \phi(\langle w, x \rangle + b)$$

où $\langle w, x \rangle = \sum_{i=1}^d w_i x_i$.

La figure 7.3 résume la définition ci-dessus.

Remarque 7.2.2. Lorsque $\phi = Id$, alors on retrouve la régression linéaire classique.

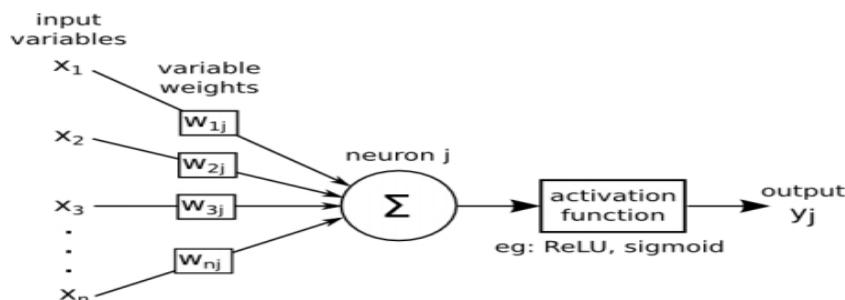


Figure 1: source: andrewjamesturner.co.uk

Figure 7.3: Structure d'un neurone

Fonctions d'activation

Ces fonctions ont vocation à reproduire le fonctionnement d'un neurone. La fonction la plus naturelle est donc

- La fonction seuil (threshold), définie par : $\phi(x) = \mathbf{1}_{x \geq 0}$. Néanmoins, en pratique, on utilise plutôt les fonctions suivantes :
- La fonction ReLU (Rectified Linear Unit), définie par : $\phi(x) = \max(0, x)$ qui est un peu moins "binaire" que la fonction seuil.

- La fonction Sigmoïde (réciproque de *logit*), définie par : $\phi(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}$ (qui croît de 0 et 1)
- La fonction Tangente Hyperbolique : $\phi(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ qui croît de -1 à 1 .

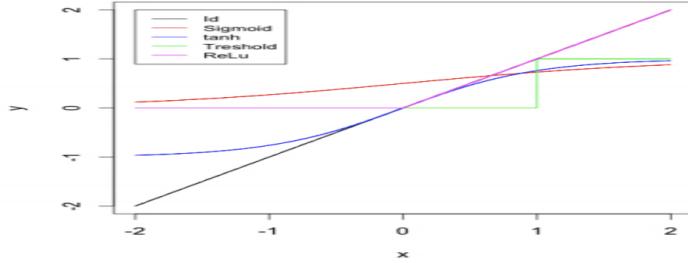


Figure 7.4: Exemples de fonctions d'activation

Remarque 7.2.3. Historiquement, la sigmoïde était très utilisée en tant que fonction régulière. C'est aussi son défaut. Plus exactement, le caractère trop “mou” du gradient rend la recherche de paramètres optimaux difficile. On lui préfère aujourd’hui la fonction *ReLU* (bien que non différentiable en 0). On peut ajouter un léger biais à la fonction *ReLU*

$$\phi(x) = \max(x, 0) + \alpha \min(x, 0)$$

afin d'éviter que le gradient soit nul sur une partie de l'espace.

On notera que pour la dernière couche (voir section suivante pour la notion de dernière couche), on utilisera une fonction d'activation adaptée au problème, *i.e.* qui renvoie une réponse de type régression ou classification selon les cas. Par exemple,

- En classification binaire : la fonction Sigmoïde (qui génère un *Score* d'où l'on déduit un classifieur).
- En classification multilabels : la fonction *Softmax* à valeurs dans l'ensemble des probabilités sur $\{1, \dots, K\}$, définie pour $x \in \mathbb{R}^K$ par :

$$\text{softmax}(x) = \left(\frac{e^{x_k}}{\sum_{i=1}^K e^{x_i}} \right)_{k \in \{1, \dots, K\}}.$$

(voir Figure 7.5 pour une illustration)

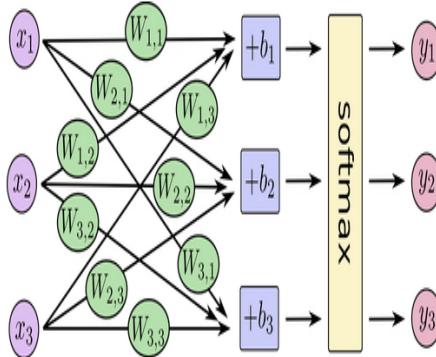


Figure 7.5: Dernière couche avec application de softmax

7.2.2 Le perceptron MultiCouche

Définition 7.2.4. Le perceptron est une structure composée d'une ou plusieurs couches cachées de neurones où la sortie d'un neurone devient l'entrée du suivant. De telle structures sont représentées dans les figures 7.6 et 7.7.

Remarque 7.2.5. On peut aussi envisager des modifications où la sortie d'un neurone est l'entrée d'un neurone de la même couche ou d'un neurone inférieur.

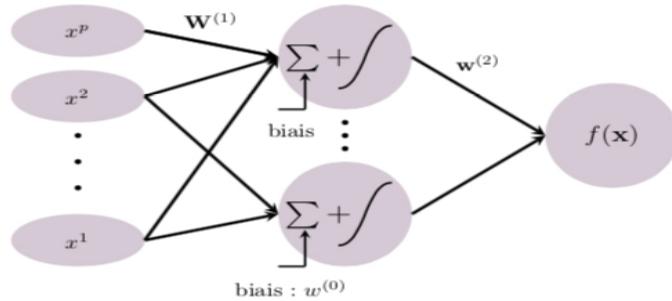


Figure 7.6: Exemple de perceptron à une couche cachée (et deux neurones dans la couche cachée)

On constate que la structure est la suivante :

- Entrée : vecteur $x = (x_1, \dots, x_p)$.
- Notons k l'indice d'un neurone de la première couche cachée (intermédiaire). Celui-ci peut être défini par :

$$h^{(k,1)}(x) = \phi_1(\langle W^{(k,1)}, x \rangle + b_{k,1}).$$

Généralement, on choisit une fonction d'activation par couche cachée. Celle-ci est donc notée ϕ_1 pour fonction d'activation associée à la première couche. En revanche, les poids $W^{(k,1)}$ et le biais $b_{k,1}$ dépendent bien entendu du neurone afin que ces derniers se "déclenchent" selon des règles différentes.

- La sortie de la couche 1 devient l'entrée de la couche suivante. L'entrée est de dimension le nombre de neurones de la couche précédente. On fabrique alors la nouvelle couche de la même manière, mais avec potentiellement une fonction d'activation différente. Ces différentes entités peuvent être vues comme des filtres. On tente d'isoler des caractéristiques simples du "signal" x puis de ses transformées.
- La dernière couche, elle, à pour rôle de délivrer une réponse $\hat{f}(x)$, i.e. le prédicteur (ou auparavant le score) associé à ce modèle.

Voici maintenant une formulation plus algorithmique :

Construction itérative du réseau pour des paramètres fixés :

- $h^{(0)}(x) = x \in \mathbb{R}^d$. Nombre de couches cachées $L \in \mathbb{N}$, nombre de neurones par couche d_k ($d_0 = d$).
- Pour $k = 1, \dots, L$,

$$a^{(k)}(x) := b^{(k)} + W^{(k)} h^{(k-1)}(x) \quad \text{où}$$

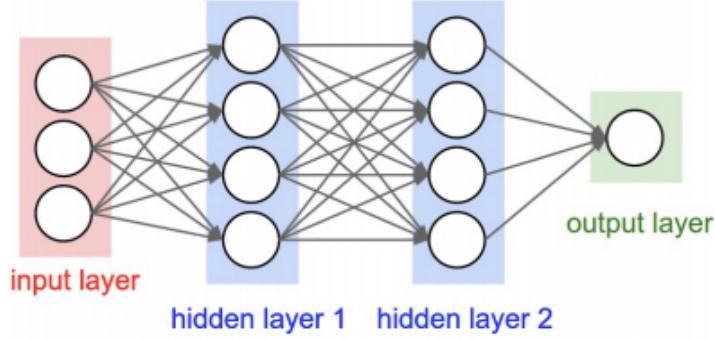


Figure 7.7: Perceptron à deux couches

$$\begin{aligned} b^{(k)} &\in \mathbb{R}^{d_k}, \quad W^{(k)} \in \mathcal{M}(d_k, d_{k-1}), \\ h^{(k)}(x) &:= \phi_k(a^{(k)}(x)) := (\phi_k(a_1^{(k)}(x)), \dots, \phi_k(a_{d_k}^{(k)}(x))), \\ \text{où } \phi_k &\text{ est une fonction d'activation.} \end{aligned}$$

- Pour $k = L + 1$, i.e. pour la dernière couche, c'est le même principe mais on choisit de distinguer les notations afin de mettre en avant que la sortie doit être en adéquation avec la forme de la réponse (régression/classification binaire, multiconcasses...). On a alors

$$\begin{aligned} a^{(L+1)}(x) &:= b^{(L+1)} + W^{(L+1)}h^{(L)}(x) \quad \text{où} \\ h^{(L+1)}(x) &:= \Psi(a^{(L+1)}(x)), \end{aligned}$$

où Ψ produit généralement une estimée de y dans le cas de la régression et une estimée de $\mathbb{P}(Y|X = x)$ dans le cas de la classification. On peut donc imaginer dans le cadre de la classification multiconcasses que $a^{(L+1)}(x) = (z_1, \dots, z_K) \in \mathbb{R}^K$ et que Ψ est la fonction *softmax*.

Pour une suite de fonctions d'activation fixée, on a donc une classe de fonctions f_θ , $\theta \in \Theta$ où Θ est défini par

$$\Theta = \{(W^{(k)}, b^{(k)}), k = 1, \dots, L + 1\}.$$

Remarque 7.2.6. On voit ici la taille potentiellement gigantesque de l'espace considéré. On peut noter que les hyperparamètres d_1, \dots, d_L (nombre de neurones par couche) ainsi que le nombre de couches lui-même n'a pas été considéré dans l'ensemble Θ . En pratique, comme dans les autres modèles (KNN, Arbres, ...), on peut dans un premier temps fixer les hyperparamètres pour la modélisation puis tenter de comparer différents modèles (i.e. avec des hyperparamètres ou des fonctions d'activation différentes). On peut aussi noter que le choix des hyperparamètres peut aussi être fait de manière adaptative mais nous ne détaillerons pas ce point.

Remarque 7.2.7. Le réseau est donc construit par compositions successives de fonctions (On retrouve ainsi le principe de base des connexions neuronales faisant de l'empilement de ces compositions une architecture potentiellement complexe).

Exercice : Considérons un réseau à une couche cachée avec 3 neurones et une sortie quantitative et une entrée de dimension 2. Précisez les paramètres à estimer ainsi que la dimension de l'espace Θ .

On peut se douter que la taille de la famille de réseaux peut avoir comme conséquence des problèmes importants de sur-apprentissage. Pour cette raison, les réseaux (perceptron ou autres) sont généralement

implémentés avec des ajouts de méthodes de régularisation pour limiter le **surapprentissage**. Un premier moyen est d'augmenter le nombre d'observations en les simulant à partir de la combinaison de données existantes et de transformations déterministes ou aléatoires (translations, rotations, convolution avec un bruit dans le cas des images par exemple). Une autre méthode consiste à mettre un alea sur l'utilisation d'un neurone :

Dropout : Le dropout ([8]) est une technique de régularisation (qui a été mise en oeuvre sur Alexnet par exemple) qui consiste à annuler la réponse d'un neurone avec une certaine probabilité p (typiquement, 1 chance sur 2). Les neurones ainsi ignorés ne contribuent pas au modèle. A chaque étape de l'apprentissage, c'est donc un modèle légèrement différent qui est utilisé. Lors de la phase de test, une solution consiste à utiliser tous les neurones mais à réduire leur activation d'un facteur p ce qui peut s'interpréter comme une approximation de l'effet moyen d'un neurone activé avec probabilité $1 - p$ (penser à l'application pour la fonction ReLU). On peut aussi conserver le dropout dans la phase de test quitte à calculer plusieurs évaluations d'un même individu (voir les paramètres par défaut des fonctions implémentées pour plus de détails).

Remarque 7.2.8. Les raffinements ou tentatives d'amélioration numérique sont nombreuses. Il n'est évidemment pas possible de les présenter toutes ici. Le lecteur devra simplement garder à l'esprit que du principe de base des réseaux de neurones, on peut envisager de nombreuses modifications pour améliorer leurs capacités prédictives. On peut évoquer par exemple le *Dropconnect* qui consiste à annihiler des connexions aléatoirement (mettre les poids à 0 avec une certaine probabilité) ou les méthodes de pénalisation (similaires au Lasso et au Ridge) qui "shrinkent" ou "seuillent" les paramètres à estimer.

7.3 Théorème d'approximation universelle

Question : Quels types de fonctions peuvent-elles être approximées par des réseaux de neurones ?

Comprendre les performances numériques du deep learning revient en partie¹ à être capable de répondre à ce type de question. Quelques résultats commencent à exister sur le sujet mais à ma connaissance, il n'existe pas encore à ce jour (notes écrites fin 2019) de résultats remarquables différenciant d'autres classes usuelles, les classes de fonctions générées par les réseaux de neurones.

Les résultats les plus nombreux sont connus sous le nom de "théorèmes d'approximation universelle" et se situent, au moins sur un plan terminologique (mais aussi sur la preuve), dans la continuité de résultats de type "théorème d'approximation de Weierstrass". On énonce ici un résultat de Hornik (1991).

Théorème 7.3.1. Soit $\phi : \mathbb{R} \mapsto \mathbb{R}$ une fonction bornée continue croissante et non-constante. Soit K un compact de \mathbb{R}^d . Alors, l'ensemble \mathcal{F} des fonctions $F : K \rightarrow \mathbb{R}$ de la forme

$$F(x) = \sum_{i=1}^N v_i \phi(\langle w_i, x \rangle + b_i), \quad N \in \mathbb{N}^*$$

est dense dans $\mathcal{C}(K, \mathbb{R})$ (ensemble des fonctions continues de K dans \mathbb{R}). Autrement dit, pour tout $f : K \mapsto \mathbb{R}$ continue, pour tout $\varepsilon > 0$, il existe $N \in \mathbb{N}^*$, il existe des réels $v_i, w_i, b_i, i = 1, \dots, N$ tels que la fonction F associée satisfasse :

$$\forall x \in K, \quad |F(x) - f(x)| \leq \varepsilon.$$

Remarque 7.3.2. Les fonctions F ainsi définies sont exactement les perceptrons à une couche cachée lorsque la fonction ψ de sortie est l'identité et que le coefficients de biais dans la couche finale sont nuls. Notons que l'ensemble des fonctions F peut aussi être vu comme l'espace vectoriel engendré par les fonctions de la forme

$$\phi(\langle w, x \rangle + b), \quad w \in \mathbb{R}^d, b \in \mathbb{R}.$$

Il existe des extensions de ce résultat aux espaces L^p ce qui permet de gérer les fonctions qui ne sont pas régulières.

¹L'aspect "rétropropagation" joue également un rôle très important, voir section suivante.

Ce résultat n'est pas quantitatif malheureusement (Il ne donne pas d'estimations de l'erreur en fonction du nombre de neurones utilisés...)

Un résultat plus récent de Pinkus (1999) montre que le résultat ci-dessus est vrai si et seulement si ϕ n'est pas une fonction polynomiale.

Exercice :

- Montrez que le résultat de Pinkus est compatible avec celui de Hornik.
- Expliquez rapidement pourquoi la condition de Pinkus est clairement nécessaire.

Idées de preuve : Considérons le cas de l'intervalle $[0, 1]$ et de la fonction seuil $\phi(x) = 1_{x \geq 0}$. On peut facilement vérifier que les fonctions de la forme $g(x) = 1_{x \in [a, b]}$ appartiennent à l'ensemble considéré car

$$g(x) = 1_{x \leq b} - 1_{x \leq a} = \phi(-x + b) - \phi(-a + x).$$

Ainsi, on peut reconstruire toutes les fonctions étagées. Dit autrement, avec cette fonction d'activation, les fonctions étagées sont des réseaux de neurones à une couche cachée. Or, il est clair que l'ensemble des fonctions étagées est dense dans l'ensemble des fonctions continues (application du théorème de Stone-Weierstrass pour les treillis par exemple).

Dans le cas de la fonction ReLU, le principe de preuve est assez proche. On peut montrer que l'ensemble des fonctions continues affines par morceaux est inclus dans l'ensemble des réseaux de neurones à une couche associés à cette fonction d'activation. On applique ensuite le théorème de Stone-Weierstrass.

Dans le cas général (en dimension 1 uniquement et lorsque ϕ est \mathcal{C}^∞), le problème se ramène à montrer que l'espace vectoriel $\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})$ engendré par l'ensemble

$$\{\phi(\lambda x + b), \lambda \in \mathbb{R}, b \in \mathbb{R}\}$$

est dense dans l'ensemble des fonctions continues. On peut alors commencer par montrer que puisque

$$x \mapsto \frac{\phi((\lambda + h)x + b) - \phi(\lambda x + b)}{h}$$

appartient à $\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})$ pour tout $h > 0$ sa limite (en tant que suite de fonctions, pour la topologie de la convergence uniforme) est dans l'adhérence de $\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})$. On a donc pour tout λ, b ,

$$x \mapsto \frac{\partial}{\partial \lambda} \phi(\lambda x + b) = \phi'(\lambda x + b)x \in \overline{\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})}.$$

En prenant la fonction ci-dessus avec $\lambda = 0$ et b_0 tel que $\phi'(b_0) \neq 0$ (ce dernier existe car la fonction n'est pas constante), on obtient que

$$x \mapsto \phi'(b_0)x \in \overline{\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})}.$$

Par conséquent, $x \mapsto x \in \overline{\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})}$. On peut reproduire ce raisonnement pour finalement montrer que pour tout k , $x \mapsto \phi^{(k)}(b)x^k$ appartient à $\overline{\mathcal{N}(\phi, \mathbb{R}, \mathbb{R})}$ (il existe pour tout k un réel b_k tel que $\phi^{(k)}(b_k) \neq 0$ lorsque ϕ n'est pas un polynôme). L'adhérence contient donc tous les monômes donc tous les polynômes. On conclut ensuite par le théorème de Weierstrass pour les polynômes (voir Pinkus, 1999, Proposition 3.4 pour plus de détails).

Pour ceux qui voudraient aller plus loin sur le sujet, consulter le cours https://www.math.univ-toulouse.fr/~7Efmalgouy/enseignement/downloadMva_deep_L/theorie_deep_learning.pdf.

7.4 Rétropropagation/Estimation des paramètres

Question : Comment trouver les paramètres optimaux ?

Réponse (très) approximative : En faisant une descente de gradient (stochastique).

Que signifie gradient stochastique ? Comment calcule-t-on le gradient ? On tentera de répondre à ces questions dans cette section.

7.4.1 Problématique

Comme dans tout problème d'apprentissage, on a à minimiser une certaine erreur définie formellement comme

$$L(\theta) = \mathbb{E}_{(X,Y)}[\ell(Y, f_\theta(X))]$$

où $f_\theta(x)$ désigne pour une entrée $x \in \mathbb{R}^p$, la sortie du réseau de neurone relative au paramètre θ (constituée elle-même de la suite de poids et de biais relatives aux différentes couches). La fonction ℓ est une fonction de perte qui peut être $\ell(y, y') = (y - y')^2$ dans le cas de la régression, l'erreur de classification dans le cas de la classification binaire ou multiconcasses. Notons que cette fonction de perte peut être pénalisée pour limiter le surapprentissage. On peut également noter que dans le cadre multiconcasses, on utilise souvent l'entropie croisée définie par

$$\ell(Y, f_\theta(X)) = - \sum_{k=1}^K \mathbb{1}_{Y=k} \log(f_\theta^k(x))$$

où $f_\theta^k(x)$ désigne l'estimation de $\mathbb{P}(Y = k | X = x)$ produite par la fonction softmax. On peut noter que lorsque la classe 1 est prédite avec probabilité proche de 1, i.e. $f_\theta(x) \approx (1, \dots, 0)$ alors

$$\mathbb{E}[\ell(Y, f_\theta(X)) | X = x] = \sum_{k=1}^K f_k(x) \log(f_\theta^k(x)) \approx 0$$

avec la convention $0 \log 0 = 0$ (obtenue en prolongeant par continuité la fonction $u \mapsto u \log u$ en 0^+).

Pour un telle fonction L , on cherche en général $\theta^* = \text{Argmin} L$. Notons dès à présent, que la fonction L dans ce cadre des réseaux de neurones n'est en général pas convexe (comme peut l'être la fonction de perte quadratique pour la régression linéaire). Il faut imaginer une fonction L telle que celle représentée en figure 7.8.

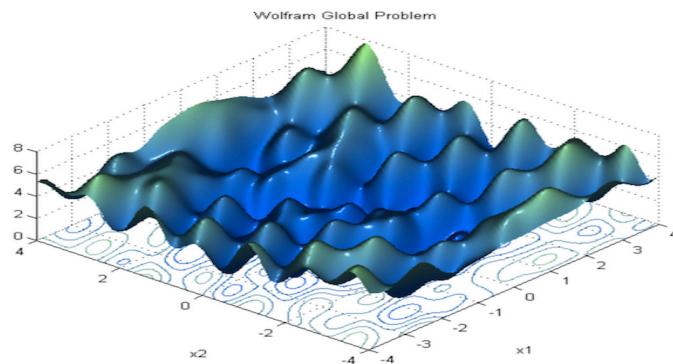


Figure 7.8: Potentiel non convexe !, Figure issue de <https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0>

“L'apprentissage de θ ” mène alors à des minima locaux en général. Cet apprentissage se fera par “descente de gradient stochastique sur les données”.

Remarque 7.4.1. On peut noter que même si des méthodes numériques existent pour chercher les minima globaux, les méthodes d'apprentissage privilégiées dans ce contexte seront conçues pour approcher les minima locaux (quitte à renouveler la procédure avec différents points de départ pour sélectionner le “meilleur” minimum local détecté par l'algorithme).

En pratique, estimer L consiste idéalement à calculer

$$L_N(\theta) = \frac{1}{N} \sum_{k=1}^N \ell(Y_k, f_\theta(X_k))$$

où l'échantillon $\mathcal{D}_N = \{(X_1, Y_1), \dots, (X_N, Y_N)\}$ est un échantillon test. Néanmoins, ici on est en train d'entraîner le modèle donc on va plus ou moins considérer l'erreur d'entraînement (en utilisant différentes astuces pour limiter le surapprentissage qui guette avec cette approche).

Le second problème réside dans la capacité à déterminer des minima (locaux) de L_N . Il s'agit d'un problème d'optimisation en grande dimension. Ainsi, on est réduit à espérer des propriétés de convexité locale de la fonction L_N pour chercher à optimiser le choix de θ .

7.4.2 Descente de Gradient/Descente de Gradient Stochastique

Descente de gradient

La manière la plus naturelle de chercher à explorer l'espace des paramètres en diminuant la valeur de la fonction L_N est de faire une descente de gradient :

$$\vartheta_{n+1} = \vartheta_n - \gamma_{n+1} \nabla_\theta L_N(\vartheta_n).$$

N.B. Dans la suite, γ_n sera fixé constant égal à $\gamma > 0$ mais on peut envisager de le faire décroître vers 0. Notons que γ est appelé le pas de l'algorithme. Remarquons également que dans des cadres simples, on tente souvent des méthodes plus performantes telles que Newton-Raphson par exemple.

Néanmoins, ce dernier nécessite pour un ϑ fixé de calculer

$$\nabla_\theta L_N(\vartheta) = \frac{1}{N} \sum_{k=1}^N \nabla_\theta \ell(y_k, f_{\vartheta_n}(x_k)).$$

Notons qu'à partir de maintenant, on choisit de noter l'échantillon $\{(x_1, y_1), \dots, (x_N, y_N)\}$ de manière à bien indiquer que l'échantillon est fixé. Notons également que par composition, on a pour tout j ,

$$\partial_{\theta_j} (\ell(y_k, f_{\vartheta_n}(x_k))) = \partial_2 \ell(y_k, f_{\vartheta_n}(x_k)) \partial_{\theta_j} f_{\vartheta_n}(x_k).$$

Par exemple, dans le cas $\ell(y, y') = (y - y')^2$,

$$\partial_{\theta_j} (\ell(y_k, f_{\vartheta_n}(x_k))) = 2(f_{\vartheta_n}(x_k) - y_k) \partial_{\theta_j} f_{\vartheta_n}(x_k).$$

Il nous faut donc calculer $\partial_{\theta_j} f_{\vartheta_n}(x_k)$ pour tout j et pour tout $k \in \{1, \dots, N\}$. Cette contrainte génère un problème numérique trop coûteux.

Descente de gradient stochastique

Une idée consiste à “randomiser le gradient”. Le principe est le suivant : on remarque que si I désigne une variable aléatoire uniforme à valeurs dans $\{1, \dots, N\}$, alors

$$\nabla_\theta L_N(\theta) = \mathbb{E}[\nabla_\theta \ell(y_I, f_{\vartheta_n}(x_I))].$$

L'idée de la descente de gradient stochastique consiste alors à remplacer $\nabla_{\theta} L_N(\theta)$ par $\nabla_{\theta} \ell(y_I, f_{\theta_n}(x_I))$ dans l'algorithme. On obtient alors l'algorithme

$$\theta_{n+1} = \theta_n - \gamma \nabla_{\theta} \ell(y_{I_{n+1}}, f_{\theta_n}(x_{I_{n+1}})) \quad (7.4.1)$$

où (I_n) désigne une suite de variables aléatoires *i.i.d.* de loi uniforme sur $\{1, \dots, N\}$ (*i.e.* une suite de tirages aléatoires avec remise dans l'ensemble $\{1, \dots, N\}$).

Exemple : Pour comprendre cette idée sur un exemple plus simple, on peut considérer l'exemple de la régression linéaire. Supposons que l'on cherche

$$\theta^* = \operatorname{Argmin}_{\theta} \frac{1}{N} \sum_{k=1}^N (y_k - \langle \theta, x_k \rangle)^2,$$

i.e. que l'on tente de trouver le minimiseur des moindres carrés dans le cadre de la régression linéaire. On sait résoudre ce problème explicitement lorsque que la matrice de design est inversible mais faisons comme si l'on ne connaissait pas la solution exacte. L'approche "optimisation stochastique" consiste ici alors à faire la descente stochastique suivante :

$$\theta_{n+1} = \theta_n - 2\gamma x_{I_{n+1}}^T (\langle \theta_n, x_{I_{n+1}} \rangle - y_{I_{n+1}})$$

où à nouveau I_n désigne un tirage aléatoire dans $\{1, \dots, N\}$. En d'autres termes, on calcule le gradient sur un point de l'échantillon en espérant que l'effet de moyennisation de la loi des grands nombres permette d'aller en moyenne dans la direction de θ^* (Il existe de nombreux résultats théoriques qui confirment cette intuition).

Revenons maintenant à l'algorithme 7.4.1. Ce dernier propose donc une alternative numériquement plus raisonnable a priori à la descente de gradient déterministe. Il nous faut cependant faire les remarques suivantes :

- Afin de réduire la dimension de l'espace, on va en réalité proposer une optimisation de la fonction L (ou L_N) en ajoutant un terme de pénalisation : typiquement, quelque chose comme $\Omega(\theta) = \|\theta\|^{\alpha}$ (plus exactement, on travaille avec des normes Frobenius de matrices). La descente de gradient stochastique se fera donc avec ce terme de pénalisation supplémentaire (1 pour une approche Lasso (sparse), 2 pour une approche Ridge).
- La descente de gradient stochastique est généralement implémentée en mode *mini-batch* ce qui signifie qu'au lieu de calculer le gradient sur une seule observation, on le calcule sur un petit groupe d'observations que l'on moyennise ensuite. Le *mini-batch* peut être vu comme une sorte de compromis raisonnable entre le "tout aléatoire" et le "tout déterministe". Ce paramètre est généralement calibrable sur les fonctions Python et R prévues pour le deep learning.
- A supposer que l'on soit dans un mode mini-batch ou "purement" stochastique, il nous faut quand même calculer le gradient à chaque itération et en particulier

$$\nabla_{\theta} f_{\theta_n}(x_{I_{n+1}}).$$

Ce calcul fait l'objet de la section suivante :

7.4.3 Rétropropagation

La rétropropagation est la dénomination relative au calcul par récurrence descendante du gradient d'un réseau de neurones. Ce procédé répond à la question : pour une entrée x et un paramètre θ donnés, comment calculer $\nabla_{\theta} f_{\theta}(x)$?

Une réponse vague consisterait à indiquer que la fonction f_θ est une composition de fonctions, donc en partant de la “fin”, on doit pouvoir remonter progressivement le réseau pour calculer toutes les dérivées partielles.

Considérons d’abord un exemple. Supposons que l’on considère un réseau de neurones à 1 couche cachée : dans ce cas, on peut écrire f_θ sous la forme :

$$f_\theta(x) = \psi \left(\beta + \sum_{i=1}^N v_i \phi(\langle w_i, x \rangle + b_i) \right).$$

Les paramètres v_i et β sont les poids et les constantes relatifs à la connexion entre la couche cachée et la sortie. Les w_i et b_i sont les paramètres qui lient l’entrée à la couche cachée. Chaque w_i vit dans \mathbb{R}^p (correspondant aux poids associés à l’ensemble des flèches menant au neurone i). On doit commencer par calculer les dérivées par rapport aux paramètres les plus récents :

- Notons $a^{(2)}(x) = \beta + \sum_{i=1}^N v_i \phi(\langle w_i, x \rangle + b_i)$.

$$\partial_\beta f_\theta(x) = \psi'(a^{(2)}(x)) \partial_\beta(a^{(2)}(x)) = \psi'(a^{(2)}(x)).$$

De même,

$$\partial_{v_i} f_\theta(x) = \psi'(a^{(2)}(x)) \partial_{v_i}(a^{(2)}(x)) = \psi'(a^{(2)}(x)) \phi(\langle w_i, x \rangle + b_i).$$

- Calculons maintenant les dérivées par rapport aux paramètres plus anciens. Notons $a_i^{(1)}(x) = \langle w_i, x \rangle + b_i$. Commençons par les b_i :

$$\partial_{b_i} f_\theta(x) = \psi'(a^{(2)}(x)) \partial_{b_i}(a^{(2)}(x)) = \psi'(a^{(2)}(x)) v_i \phi'(a_i^{(1)}(x)).$$

Regardons enfin les dérivées par rapport à w_i^j , i.e. le poids associé à la connexion entre l’entrée x_j et le neurone i . On a

$$\partial_{w_i^j} f_\theta(x) = \psi'(a^{(2)}(x)) \partial_{w_i^j}(a^{(2)}(x)) = \psi'(a^{(2)}(x)) v_i \phi'(a_i^{(1)}(x)) x_j.$$

On peut remarquer sur l’exemple précédent que pour calculer les dérivées partielles par rapport aux w_i ou aux b_i , on a eu recours à $\psi'(a^{(2)}(x)) = \frac{\partial f_\theta}{\partial a^{(2)}}$ qui a été calculé à la première étape. Plus généralement, par la formule de dérivation des fonctions composées, on aurait pu écrire :

$$\frac{\partial f_\theta}{\partial w_i} = \frac{\partial f_\theta}{\partial a^{(2)}} \times \frac{\partial a^{(2)}}{\partial a^{(1)}} \times \frac{\partial a^{(1)}}{\partial v_i}.$$

Remarquons que cette formulation est en général à comprendre comme un produit de matrices. Néanmoins, on peut au moins formellement se représenter la récurrence. Pour calculer les dérivées partielles par rapport à la $L+1-\ell$ -ième couche, on suppose calculées les dérivées par rapport à la couche précédente. On calcule les dernières (dans notre exemple, il s’agit $\frac{\partial a^{(1)}}{\partial v_i}$ qui est un vecteur de taille N car $a^{(1)}$ l’est) et on obtient finalement à la couche k : finalement utiliser une formule du type :

$$\frac{\partial f_\theta}{\partial W_{i,j}^{(k)}} = \frac{\partial f_\theta}{\partial a^{(k)}} \frac{\partial a^{(k)}}{\partial W_{i,j}^{(k)}},$$

Pour faire avancer la récurrence, on calcule

$$\frac{\partial f_\theta}{\partial a^{(k-1)}}$$

qui va nous servir à calculer les dérivées suivantes...

7.5 Réseaux de neurones à convolution

Comme indiqué plus haut, les réseaux de neurones sont devenus célèbres en partie grâce à leurs performances dans le domaine de la reconnaissance d'images. Dans ce domaine, les réseaux de neurones utilisés ont une structure particulière. On les appelle des réseaux de neurones à convolution.

L'idée générale de ces réseaux est d'éviter de considérer directement une image comme un vecteur de \mathbb{R}^{p^2} par exemple (lorsque l'on a une image Noir et Blanc $p \times p^2$) et d'appliquer un certain nombre de filtres qui vont permettre de dégager des caractéristiques de l'image qui faciliteront sa reconnaissance.

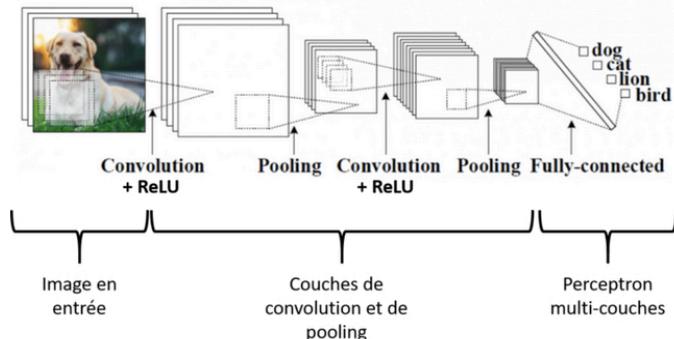


Figure 7.9: Architecture d'un CNN

Dans l'exemple 7.9, on reconnaît une architecture en certains points semblable au perceptron multi-couches. En réalité, toute la première partie est constituée de filtrages de deux types (en tout cas, dans le cadre usuel) : convolution³ et pooling. A l'issue de cette suite d'opérations, on a fabriqué une famille d'images captant chacune une part de l'information contenue dans l'image. Cet ensemble d'images constitue alors un vecteur d'entrée de la dernière partie de l'architecture qui, elle, est de type perceptron multi-couches classique (on parle de “fully connected network” pour mettre en avant le fait que toutes les couches précédentes (*i.e.* toutes les images fabriquées auparavant) sont interconnectées à l'aide du perceptron. Détailons maintenant la construction de ces différentes couches.

7.5.1 Convolution

La convolution constitue une des étapes de filtrage de l'image. Elle fait bien référence à la notion de convolution au sens mathématique du terme. Plus précisément, si l'on considère uniquement le cas discret, alors :

La convolution discrète de deux fonction f et g est formellement définie par

$$f * g(x) = \sum_t f(t)g(x+t)$$

. **Exemple :** Considérons pour commencer l'exemple de l'espace des fonctions de \mathbb{Z} dans \mathbb{Z} et notons $f(z) = \frac{1}{3}1_{z \in -1,0,1}$. Dans ce cas,

$$f * g(x) = \frac{1}{3}(g(x-1) + g(x) + g(x+1)).$$

Dans ce cas, la fonction f est appelée un noyau de convolution qui s'applique une fonction g quelconque. On voit que sur l'exemple ci-dessus, la convolution a pour effet de faire une des valeurs de la fonction g au voisinage de x .

²Dans le cas de l'image couleur, on a à considérer la décomposition RGB...

³A l'issue de la convolution, il y a généralement une étape ReLU pour annuler les pixels négatifs.

Dans le cas des images, la convolution va avoir pour rôle de transformer les petites zones de l'image. Formellement, encore si l'on considère une image I comme un élément de \mathbb{Z}^2 (quitte à mettre des zéros en dehors du carré $\{0, \dots, p\} \times \{0, \dots, p\}$), alors :

Définition 7.5.1. Si K désigne une application de \mathbb{Z}^2 dans \mathbb{R} , la convolution de l'image I par (le noyau/filtre) K est définie par : pour tous i, j ,

$$K * I(i, j) = \sum_{n, m} K(n, m)I(i + n, j + m).$$

En pratique, la fonction K est généralement non nulle sur quelques pixels seulement. Voici un exemple de convolution par noyau :

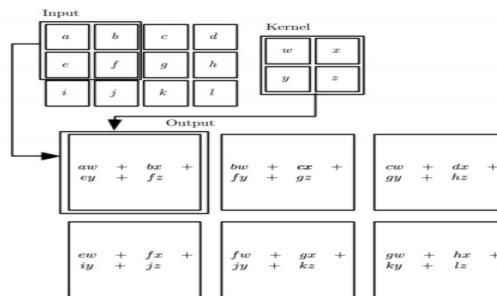


Figure 8: 2D convolution. Source :<https://i2.wp.com/khshim.files.wordpress.com/2016/10/2d-convolution-example.png>

Figure 7.10: Exemple de transformation

En d'autres termes, en pratique, les convolutions sont juste un ensemble fini de poids.

Les différents types d'effet de la convolution

Chaque manière de convoler l'image a vocation à avoir un effet particulier. Certaines convolutions augmentent le contraste, d'autres les contours, d'autres le flou.... Voici ci-dessous quelques exemples issus de la page <https://docs.gimp.org/2.6/fr/plug-in-convmatrix.html> :

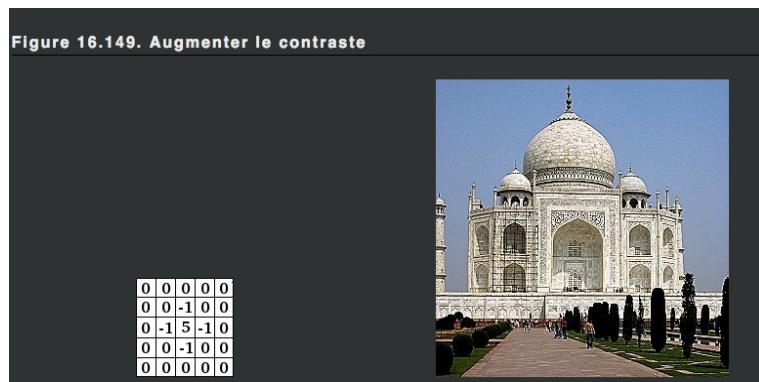


Figure 7.11: Augmentation des contrastes

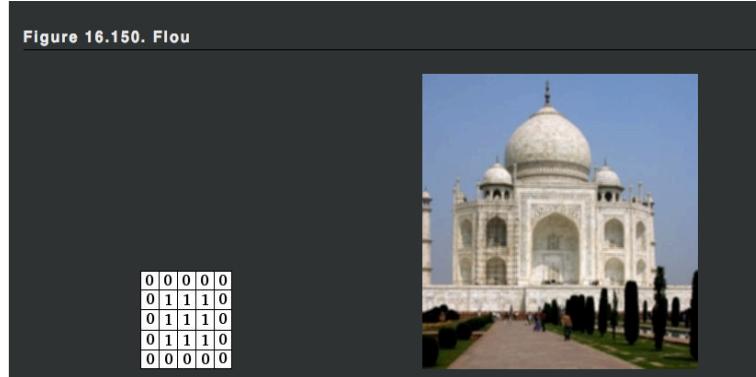


Figure 7.12: Convolution et flou

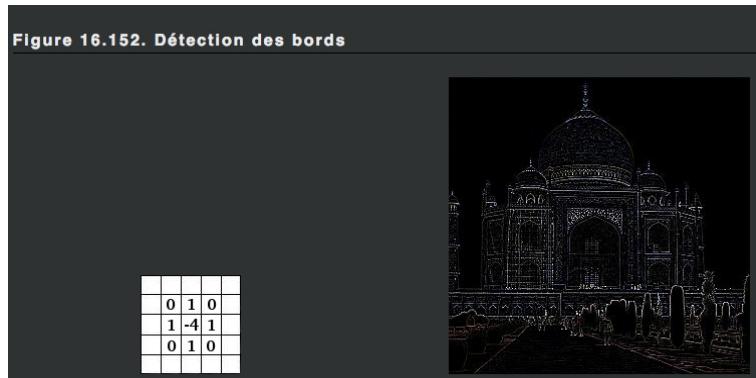


Figure 7.13: Convolution et bordures

7.5.2 Pooling

Le pooling constitue le deuxième type de filtrage. Il a vocation à réduire la dimension en remplaçant un ensemble de pixels par un seul pixel. On a ainsi :

- Le *max-pooling* qui sélectionne la valeur maximale sur un ensemble de pixels. En voici un exemple ci-dessous (Figure 7.14) :
- Le *mean-pooling* : remplace un ensemble de pixels par leur moyenne.

Les étapes de pooling sont aussi appelées “subsampling”.

7.5.3 Fully Connected Network

A l'issue de cette suite de filtrages, on reconstitue un objet contenant un des images ainsi construites à la dernière couche de convolution/pooling puis on lui applique un perceptron multi-couches (voir TP pour des mises en oeuvre). L'exemple de Alexnet est représenté dans la figure 7.15 <https://www.learnopencv.com/understanding-alexnet/>

7.6 RNN et GAN

Nous avons présenté en détail la structure d'un CNN. Les réseaux de neurones récurrents ainsi que les modèles génératifs ne seront pas présentés ici. Ces structures n'en demeurent pas moins des avancées importantes dans

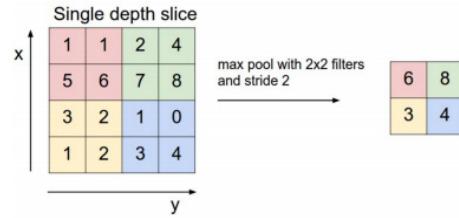


Figure 11: Maxpooling and effect on the dimension - Source :
<http://www.wildml.com/wp-content/uploads/2015/11/Screen-Shot-2015-11-05-at-2.18.38-PM.png>

Figure 7.14: Max-pooling

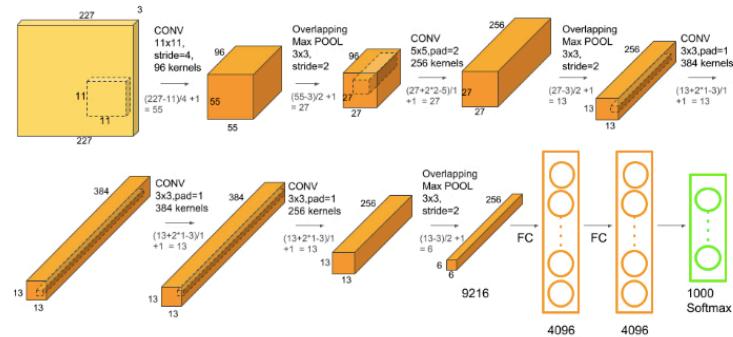


Figure 7.15: AlexNet Neural Network.

Figure issue de <https://www.learnopencv.com/understanding-alexnet/>

le domaine de la science des données. Pour les lecteurs curieux, on pourra par exemple consulter le cours suivant http://cedric.cnam.fr/~thomen/cours/MVA/Robust_deep_course.pdf pour aller un peu plus dans les détails.

7.7 Conclusion

A l'image d'une partie du cours, l'évocation des RNN et GAN est très limitée relativement à son utilisation actuelle. L'objectif de ce cours a donc été d'introduire dans leur ensemble les méthodes d'apprentissage allant de l'étude assez précise de méthodes basiques à l'évocation de méthodes complexes et récentes. Il est bien entendu qu'il vous appartient maintenant de continuer cet apprentissage à travers vos projets, stages et premières expériences professionnelles. Parmi les points non développés dans ce polycopié qui pourraient figurer dans ce cours, on peut citer les aspects calculs avec le calcul GPU (dont l'utilisation est primordiale dans le cadre des réseaux de neurones) ou les algorithmes de filtrage collaboratif utilisés pour fabriquer des systèmes de recommandation.

Bibliographie

- [1] Sylvain Arlot. Fondamentaux de l'apprentissage statistique. Disponible sur Moodle.
- [2] Christophe Giraud. Introduction to High-Dimensional Statistics. Chapman & Hall.
- [3] Trevor Hastie, Robert Tibshirani, Gareth James, Daniela Witten. Introduction to Statistical Learning. <http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf>
- [4] Trevor Hastie, Robert Tibshirani, Jerome Friedman. The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Second Edition. February 2009. <https://web.stanford.edu/~hastie/ElemStatLearn/>.
- [5] Statistical Learning with Sparsity: the Lasso and Generalizations. <https://web.stanford.edu/~hastie/StatLearnSparsity/>.
- [6] Christophe Giraud. Introduction to High-Dimensional Statistics. Chapman & Hall.
- [7] Krizhevsky, A., Sutskever, I. and Hinton, G. E. 2012. Image net classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.
- [8] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014. Dropout : a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929–1958.

Apprentissage Statistique en Grande Dimension-TD 1

Les exercices de ce premier chapitre sont en majorité consacrés à l'apprentissage supervisé.

Exercice 1. Dans les exemples suivants, dans quels cas pensez-vous qu'il est judicieux de choisir une méthode d'apprentissage flexible ?

1. Si le nombre n d'observations est grand et le nombre de variables p est petit
2. n est petit et p est grand.
3. La relation entre \mathbf{x} et y semble vraiment non-linéaire.
4. La variance du terme d'erreur ε est grande (Considérer un modèle de régression $Y = f(\mathbf{X}) + \varepsilon$).

Dans les situations 1, 3 et 4, on pourra tenter d'apporter un début d'explication théorique.

Exercice 2 (Règles de Bayes). 1. Démontrez le point (i) du théorème 1.2.5.

2. Démontrez le point (iii) du théorème 1.2.5 dans le cadre multiconfondus.

Exercice 3. On considère un modèle de régression sur \mathbb{R}^p et on note ℓ la fonction de perte définie par

$$\ell(y, y') = (y - y')^2.$$

Notons $f^*(X) = \mathbb{E}[Y|X]$.

1. Montrez que

$$Y = f^*(X) + \varepsilon \quad \text{avec } \mathbb{E}[\varepsilon] = 0 \text{ et } \text{Cov}(g(X), \varepsilon) = 0. \text{ pour toute fonction } g \text{ mesurable bornée.}$$

2. En déduire que pour une règle de décision $f : \mathbb{R}^p \rightarrow \mathbb{R}$, l'excès de risque satisfait

$$R_f - R_{f^*} = \mathbb{E}[(f(X) - f^*(X))^2].$$

3. En déduire que pour un prédicteur \hat{f}_n (bâti sur un échantillon d'apprentissage indépendant du couple (X, Y)), le risque moyen vaut

$$\mathbb{E}[R_{\hat{f}_n}] = \mathbb{E}[(\hat{f}_n(X) - f^*(X))^2] + \mathbb{E}[\varepsilon^2].$$

4. On suppose maintenant que l'on souhaite calculer le risque en un point \mathbf{x} . Montrez que

$$\mathbb{E}[\ell(Y, \hat{f}_n(X))|X] = \Phi(X) \quad \text{où } \Phi(\mathbf{x}) = \text{Var}((\hat{f}_n(\mathbf{x})) + \text{Var}(\varepsilon) + (\mathbb{E}[\hat{f}_n(\mathbf{x})] - f^*(\mathbf{x}))^2).$$

5. Sauriez-vous indiquer comment les quantités en jeu dans l'expression ci-dessus dépendent de la flexibilité de l'algorithme ? (On pourra considérer l'exemple des k -ppv).
6. Rappelez la définition du risque d'entraînement et du risque de test ces deux quantités (dans ce cadre). Expliquez dans le cadre du 1-ppv (quantitatif) pourquoi le risque d'entraînement tend vers 0 (on pourra supposer que X a une densité).

Exercice 4. Dans les exemples suivants, dégagerez les situations de régression ou de classification. Déterminez n et p .

1. On collecte les données de 500 entreprises. Pour chacune d'entre elles, on enregistre le chiffre d'affaires, le nombre d'employés, l'âge moyen des employés et le salaire moyen. On cherche ici à comprendre quels facteurs influent sur le salaire moyen par entreprise.

2. On considère un nouveau produit pour lequel on souhaite évaluer si sa mise en vente sera un succès ou un échec. Pour cela, on collecte les données de 20 produits similaires pour améliorer notre pronostic. Plus précisément, sont collectés : le prix de fabrication, le budget marketing, le prix le plus compétitif du marché ainsi que 10 autres variables. Par ailleurs, pour chaque produit, on sait également si sa mise en vente a été une réussite ou non.
3. On cherche ici à prédire le taux de change du Dollar pour la semaine suivante. Pour cela on collecte les données semaine par semaine sur l'année précédente du taux de change du Dollar ainsi que des prix de 100 produits boursiers.

Exercice 5 (Validation Croisée). 1. Le coût de calcul du prédicteur \hat{f}_n peut être très long dans certaines situations (si par exemple, construit comme la solution non explicite d'un problème de minimisation). Ainsi si le nombre de classes K utilisé dans la validation croisée est important, cette procédure peut alors être très coûteuse. Selon vous, quel procédé peut-on envisager pour réduire le temps de calcul (si la structure de calcul le permet) ?

2. Supposons que l'échantillon est de taille n et que les ensembles I_1, \dots, I_K sont de même cardinal r . Montrez que dans ce cas,

$$\mathbb{E}[\hat{R}_{CV}] = \mathbb{E}[\phi(\mathcal{D}_{n-r})]$$

où \mathcal{D}_{n-r} est un échantillon de taille $n-r$, ϕ est une fonction de $(\mathcal{X} \times \mathcal{Y})^{n-r}$ vers \mathbb{R} définie par :

$$\phi(\mathbf{d}_{n-r}) = \mathbb{E}[\ell(Y, \hat{f}_{\mathbf{d}_{n-r}}(\mathbf{X}))]$$

avec $\mathbf{d}_{n-r} = (\mathbf{x}_i, y_i)_{i=1}^{n-r}$ (déterministe).

3. On suppose que le modèle est de la forme $Y = f(\mathbf{X}) + \varepsilon$ (ε centrée indépendante de \mathbf{X}) et que $\ell(y, y') = (y - y')^2$. Explicitez la fonction ϕ .
4. On suppose dans cette question que le prédicteur est faiblement consistant. En déduire la limite de $\mathbb{E}[\hat{R}_{CV}]$ (on suppose que le nombre de folds K ne dépend pas de n).
5. Quels problèmes se poseraient-ils si l'on envisageait de calculer $\text{Var}(\hat{R}_{CV})$? (Ceci explique en partie les limites théoriques de cette méthode.)

Exercice 6 (Données manquantes/Données “mal balancées”). 1. On dispose de N individus, p variables et $m < N$ colonnes où il manque une variable.

- Que décidez-vous si N est grand devant p et m ?
- Que décidez-vous si vous n'êtes pas dans la première situation? Quels sont les points à considérer pour tenter de compléter le jeu de données de manière raisonnable?

2. — On dispose de données dont l'effectif est mal équilibré. Quelles peuvent être les conséquences lorsque l'on fait de la prédiction?

- Comment pourriez-vous modifier la fonction de perte usuelle en classification pour pallier ce problème?

Exercice 7 (1-ppv). Comme cela a été expliqué en cours, la manière la plus naturelle de mimer le prédicteur optimal de Bayes, consiste à approcher les probabilités conditionnelles $\mathbb{P}(Y = y|\mathbf{X} = x)$ (cas où \mathcal{Y} discret) par une approximation locale de cette quantité. L'algorithme des k -plus proches voisins en est une. On s'intéresse ici à sa mise en oeuvre sur un exemple simple de classification binaire puis nous focalisons sur la non-consistance du plus proche voisin (“1-ppv”). On pose $\mathcal{X} = [0, 1]$ et $\mathcal{Y} = \{0, 1\}$.

1. L'échantillon d'apprentissage est $(X_1 = 0.8, Y_1 = 1), (X_2 = 0.4, Y_2 = 0), (X_3 = 0.7, Y_3 = 1)$. Donnez la valeur prédite pour toute nouvelle entrée $x \in \mathcal{X}$
 - (a) par l'algorithme des 3-p.p.v.
 - (b) par l'algorithme du p.p.v.
2. Dans cette question, on suppose que la loi $\mathbb{P} = \mathcal{L}(X, Y)$ est la suivante : X suit la loi uniforme sur $[0, 1]$ et, $Y = 1$ si $X > 0.5$ et $Y = 0$ si $X \leq 0.5$.
 - (a) Donnez $\mathbb{P}(Y = 1|X = x)$ pour tout $x \in \mathcal{X}$. Qu'en pensez-vous?
 - (b) En déduire le prédicteur de Bayes (prédicteur optimal). Quel est son risque?

- (c) Soit \mathcal{D}_n un échantillon d'apprentissage (ou une base d'apprentissage). Soit E l'évènement : "tous les Y_i sont identiques". Que peut-on dire de l'algorithme du 1-ppv sur cet évènement ? Quelle est sa probabilité ?
- (d) En introduisant la statistique d'ordre $(X^{(i)})_{i=1}^n$ associée à l'échantillon $(X_i)_{i=1}^n$, donner une expression simple de l'algorithme du plus proche voisin lorsque $\omega \in E^c$ (noté à nouveau \hat{f}). Montrez également que sur E^c ,
- $$R_{\hat{f}} = \left| \frac{X^{(i^*)} + X^{(i^*+1)}}{2} - \frac{1}{2} \right|.$$
- Rappel : La statistique d'ordre associée à l'échantillon $(X_i)_{i=1}^n$ est la suite réordonnée des X_i : si $X_{k_1} < X_{k_2} < \dots < X_{k_n}$, alors $X^{(i)} = X_{k_i}$.*

- (e) Notons $Z_i = |X_i - \frac{1}{2}|$. Montrez que

$$\mathbb{E}[R_{\hat{f}}] \leq \mathbb{E}[Z^{(2)}] + \mathbb{P}(E),$$

où $(Z^{(i)})_{i=1}^n$ désigne à nouveau la statistique d'ordre associée à $(Z_i)_{i=1}^n$.

- (f) Montrez que si les X_i sont *i.i.d.* de loi $\mathcal{U}_{[0,1]}$, alors pour tout $r \in [0, 1/2]$,

$$\mathbb{P}(Z^{(2)} > r) = (1 - 2r)^n + 2nr(1 - 2r)^{n-1}.$$

En déduire que

$$\mathbb{E}[Z^{(2)}] = \frac{1}{n+1}.$$

- (g) Conclure sur la consistance du 1-ppv dans ce cadre.

Indication : on pourra utiliser le lemme de Wald dont l'énoncé est le suivant : pour une variable positive Z ,

$$\mathbb{E}[Z] = \int_0^{+\infty} \mathbb{P}(Z > r) dr.$$

3. Dans cette question, on suppose que la loi $\mathbb{P} = \mathcal{L}(X, Y)$ est la suivante : X suit la loi uniforme sur $[0, 1]$ et la loi conditionnelle de Y sachant $X = x$ est donnée par

$$\mathbb{P}(Y = 1|X = x) = \frac{2}{3} = 1 - \mathbb{P}(Y = 0|X = x).$$

- (a) Définir le prédicteur de Bayes dans ce cas. Quel est son risque ¹ ?
- (b) On cherche maintenant à estimer le risque de l'algorithme du plus proche voisin pour ce modèle.
- i. Définir l'algorithme dans ce cas (en fonction de \mathcal{D}_n , par définition d'un prédicteur). On pourra utiliser les *cellules de Voronoï* associées à l'échantillon :

$$\mathcal{C}_i = \{x \in [0, 1], \min_{k=1}^n (|X_k - x|) = |X_i - x|\}.$$

N.B. Les \mathcal{C}_i ne forment pas exactement une partition car les frontières sont communes mais comme la loi uniforme est continue, on peut raisonner en faisant comme tel.

- ii. Montrez que X et Y sont indépendants (On pourra calculer $\mathbb{P}(X \in [a, b], Y = 1)$).
- iii. Donnez une expression du risque de classification (à l'aide des cellules de Voronoï).
- iv. Montrez que le risque est égal à

$$2\mathbb{P}(Y = 1)\mathbb{P}(Y = 0).$$

1. On rappelle ici que ce risque est "indépassable" au sens où aucun algorithme ne pourra prétendre avoir un risque inférieur au risque de Bayes. Celui-ci est inhérent au modèle.

v. Qu'en déduit-on quant à la consistance du plus proche voisin ?

Exercice 8 (Théorème de Stone). On s'intéresse à une méthode par partition (version simplifiée des k -ppv), i.e. dans le cas où \mathcal{X} est un compact de \mathbb{R}^p . On note $(A_k^{(\varepsilon)})_{k=1}^{k_\varepsilon}$ une partition de \mathcal{X} telle que $\sup_{k=1}^{k_\varepsilon} \text{diam}(A_k^{(\varepsilon)}) \leq \varepsilon$. Notons $\mathcal{I}_k^\varepsilon = \{i \in \{1, \dots, n\}, X_i \in A_k^{(\varepsilon)}\}$ (cet ensemble est aléatoire). Dans le cas de la régression, le prédicteur est alors défini par :

$$\forall x \in A_k^{(\varepsilon)}, \quad \hat{f}(x) = \frac{1}{|\mathcal{I}_k^\varepsilon|} \sum_{i \in \mathcal{I}_k^\varepsilon} Y_i.$$

Discutez de la consistance de cette méthode (lorsque ε tend vers 0) (pour une démonstration précise et plus générale du théorème de Stone, voir le cours de S. Arlot accessible sur Moodle, on y trouvera également de nombreux exercices sur le sujet).

Exercice 9 (Classification binaire). Soit un problème d'apprentissage où Y est à valeurs dans $\{-1, 1\}$ et $Y|\mathbf{X} = x$ suit une loi de Rademacher de paramètre $p(x)$ où $p(x) \in [0, 1]$.

- On considère la fonction de perte $\ell(y, y') = \log(1 + e^{-yy'})$. Déterminez le prédicteur optimal dans ce cas, i.e. déterminez la fonction f (si elle existe) minimisant $\mathbb{E}[\ell(Y, f(\mathbf{X}))]$. Conclusion ?

Indication : On pourra commencer par montrer que $R_f = \mathbb{E}[\Psi_f(X)]$ avec

$$\Psi_f(x) = \log(1 + e^{-f(x)})p(x) + \log(1 + e^{f(x)})(1 - p(x)).$$

- On s'intéresse au problème de minimisation suivant : déterminez parmi les fonctions $f : \mathcal{X} \rightarrow \mathbb{R} \cup \{-\infty, +\infty\}$, celle qui minimise $\mathbb{E}[\ell(Y, f(\mathbf{X}))]$. Notons à nouveau f^* cette fonction. De quelle fonction usuelle s'agit-il ?

N.B. Cet exercice fait un lien avec la régression logistique. Dans ce cadre, on fait généralement l'hypothèse que la fonction p satisfait

$$\log\left(\frac{p(x)}{1 - p(x)}\right) = \langle \theta, x \rangle$$

où $\theta \in \mathbb{R}^p$. En d'autres termes, on fait une hypothèse de linéarité pour une transformation adéquate de la loi de Y sachant \mathbf{X} . Cette transformation est généralement appelée fonction de lien.

Exercice 10 (Risque asymétrique/Courbe ROC). Comme cela a été expliqué en cours, les fonctions de risque usuelles ne sont pas toujours adaptées au problème considéré. L'erreur de classification standard par exemple (en classification binaire) ne prend pas en compte l'importance des quantités en jeu. Par exemple, en médecine, il est usuellement plus grave de classer parmi les malades une personne saine plutôt qu'une personne saine parmi les malades. La courbe ROC (Receiver operating characteristic) est un outil venu du traitement du signal permettant de prendre en compte cette dissymétrie. On suppose que la sortie Y est égale à 1 si l'individu est malade et 0 sinon.

- On note $\eta(x) = \mathbb{P}(Y = 1|\mathbf{X} = x)$ et on considère la règle de décision suivante : pour un seuil s fixé appartenant à $[0, 1]$,

$$f_s^*(x) = \begin{cases} 1 & \text{si } \eta(x) > s \\ 0 & \text{sinon.} \end{cases} \quad (1)$$

Montrez que f_s^* minimise le risque associé à la fonction de perte :

$$\ell(y, y') = (1 - s)\mathbf{1}_{\{y=1, y'=0\}} + s\mathbf{1}_{\{y=0, y'=1\}}.$$

Indication : On pourra remarquer que dans le cas $s = 1/2$, on retrouve le cas classique.

- Dans cette partie, on se place dans le cas particulier de l'analyse discriminante linéaire :

$$\mathcal{L}(X|Y = i) = \mathcal{N}(m_i, \sigma_i^2), \quad i = 0, 1$$

et $\mathbb{P}(Y = 1) = p$. On suppose également que $m_0 < m_1$.

- Représentez graphiquement cette situation.

- (b) Pour s fixé, explicitez une règle de décision de la forme (1).
(c) On suppose dans cette question que $\sigma_0 = \sigma_1$. Dans ce cas, montrez que la règle peut s'écrire :

$$f(x) = \begin{cases} 1 & \text{si } x > \alpha_s \\ 0 & \text{sinon,} \end{cases}$$

où α_s est un seuil à définir.

- (d) Notons Se et Sp les fonctions (Sensibilité et Spécificité) définies par

$$\text{Se}(\alpha) = \mathbb{P}(X > \alpha | Y = 1) \quad \text{et} \quad \text{Sp}(\alpha) = \mathbb{P}(X \leq \alpha | Y = 0).$$

A quoi correspondent ces quantités (vrai/faux, positif/négatif...) ? Reformulez ces quantités en termes de risque de 1ère espèce/2ème espèce (en supposant que H_0 = "Individu malade").

- (e) Dans ce cadre (de la LDA), la courbe ROC est alors le graphe de

$$\{(1 - \text{Sp}(\alpha), \text{Se}(\alpha), \alpha \in \mathbb{R})\}.$$

Tracez cette courbe avec $m_0 = 1$ et $m_1 = 2$ ou $m_1 = 3$ avec dans chaque cas $\sigma_0 = \sigma_1 = 1$.

- (f) Montrez que si F est la fonction de répartition de $\mathcal{L}(X|Y = 0)$ et G , celle de $\mathcal{L}(X|Y = 1)$, alors la courbe ROC est aussi le graphe de la fonction

$$\text{ROC}(t) = 1 - G(F^{-1}(1 - t)), \quad t \in]0, 1[.$$

- (g) Dans ce qui précède, les calculs sont effectués sous un point de vue "oracle", i.e. en travaillant sous l'hypothèse que la loi du couple (X, Y) est connue. A l'aide de ce qui précède, proposez un prédicteur basé sur un échantillon d'apprentissage \mathcal{D}_n (On pourra s'appuyer sur des estimateurs usuels de la moyenne et de la variance). Ce prédicteur est-il consistant ?

- (h) Pour un échantillon donné, tracez la courbe ROC correspondante (en remplaçant les fonctions de répartition par des fonctions de répartition empiriques).

N.B. La courbe ROC est un outil de mesure de la qualité de classification binaire assez usuel. Pour comparer différents classifiants, on peut mesurer l'aire sous la courbe appelée aire AUC.

3. Dans la partie précédente, on a fait des hypothèses fortes sur le modèle. En pratique, il n'est souvent pas envisageable d'utiliser cette approche. Construisez une courbe ROC basée sur les effectifs de faux/vrais positifs/négatifs.
4. Dans la littérature, on utilise aussi le F_1 -score comme mesure de la qualité d'un algorithme. Etudiez la construction du F_1 -score et discutez l'intérêt de cette mesure. Même question pour le coefficient de corrélation de Matthews.

Exercice 11. Ce dernier exercice a vocation à jouer le rôle de mémo sur les méthodes non supervisées classiques. Parcourez le document ci-joint <http://www2.agroparistech.fr/IMG/pdf/ClassificationNonSupervisee-AgroParisTech.pdf> et listez les méthodes connues.

Statistique en Grande Dimension et Apprentissage - TD 2

Ce TD a pour but d'illustrer le chapitre 2 sur les problèmes de la grande dimension puis de faire un rappel sur l'analyse en composantes principales sur laquelle on reviendra dans la suite du cours (dans sa version sparse).

Exercice 1 (Faire plus de mesures = perdre de l'information?). On s'intéresse dans ce problème à l'analyse des puces à ADN. Après un traitement approprié, les données de puces à ADN correspondent à un vecteur (Y_1, \dots, Y_p) de différences de log-intensités. Typiquement, le nombre p de gènes est de l'ordre de quelques milliers. Ces observations peuvent être modélisées comme suit :

$$Y_j = \theta_j + \varepsilon_j, \quad j = 1, \dots, p,$$

et (ε_j) suite de variables aléatoires *i.i.d* de loi $\mathcal{N}(0, \sigma_j^2)$. Pour simplifier, on suppose que l'on est dans un cadre *homoscédastique*, *i.e.* que $\sigma_j = \sigma$. Le but est ici de détecter les gènes “positifs”, *i.e.* ceux pour lesquels

$$\theta_j \neq 0.$$

En général, de 1 à 10% des gènes sont positifs. On suppose que l'on observe n individus : chaque observation est notée

$$Y^{(i)} = (Y_1^{(i)}, \dots, Y_p^{(i)}).$$

1. On s'intéresse à un gène j fixé. Proposez une règle de décision telle que la probabilité que le gène j soit déclaré positif à tort (faux positif/false discoverie) soit de au plus 5%. (On veut limiter les fausses découvertes pour ne faire des expérimentations biologiques (généralement coûteuses) que sur les gènes pour lesquels on “est sûr” qu’ils sont positifs).
2. Supposons que $p = 5000$ et que 4% des gènes sont réellement positifs. Quel est le nombre moyen de faux positifs que va engendrer la règle de décision ci-dessus ?
3. Sachant que

$$\mathbb{P}\left(\max_{j=1, \dots, p} \varepsilon_j^2 > 2\gamma \log p\right) \xrightarrow{p \rightarrow +\infty} 1_{\gamma < 1}, \quad (1)$$

proposez une règle de décision telle que la probabilité qu'il existe au moins un j déclaré positif à tort, soit asymptotiquement nulle.

4. Que se passe-t-il quand p devient très grand ? Comment pourriez-vous envisager de pallier ce problème ?
5. Dans cette dernière question, on se propose de prouver le résultat (1).

- (a) Pour un seuil q fixé, exprimez

$$\mathbb{P}\left(\max_{j=1, \dots, p} |\varepsilon_j| \leq q\right)$$

en fonction de la fonction $G(z) = \mathbb{P}(|Z| > z)$.

(b) Montrez que pour tout $z > 1$,

$$\begin{aligned}\mathbb{P}(|Z| > z) &= \sqrt{\frac{2}{\pi}} \frac{e^{-\frac{z^2}{2}}}{z} - \sqrt{\frac{2}{\pi}} \int_z^{+\infty} \frac{e^{-\frac{x^2}{2}}}{x^2} dx \\ &= \sqrt{\frac{2}{\pi}} \frac{e^{-\frac{z^2}{2}}}{z} \left(1 + O\left(\frac{1}{z^2}\right) \right).\end{aligned}$$

Indication : On pourra faire une IPP (et considérer en premier lieu $\mathbb{P}(Z > z)$).

(c) En déduire le résultat.

Exercice 2 (Analyse en composantes principales). L'analyse en composantes principales (ACP) est un moyen usuel de réduire la dimension en tentant d'extraire les directions qui contiennent le plus d'information. Dans cet exercice, on se propose de faire quelques rappels sur l'ACP, théoriques et pratiques et d'étudier numériquement la robustesse (ou plutôt la non-robustesse) de l'ACP à la dimension.

1. On souhaite ici redémontrer la décomposition en valeurs singulières (SVD). On considère A une matrice $n \times p$ de rang r .
 - (a) Montrez que $r \leq \min(n, p)$.
 - (b) Montrez que $\mathcal{Im}(A^T)$ et $\text{Ker } A$ sont orthogonaux puis que $\mathcal{Im}(A) = \mathcal{Im}(AA^T)$.
 - (c) Montrez que AA^T admet une décomposition de la forme suivante :

$$AA^T = \sum_{j=1}^r \lambda_j u_j u_j^T$$

où $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_r > 0$ et $\{u_1, \dots, u_r\}$ est une famille orthonormale.

- (d) Posons $v_j = \lambda_j^{-\frac{1}{2}} A^T u_j$, $j = 1, \dots, r$. Montrez que $\|v_j\|^2 = 1$ pour tout $j \in \{1, \dots, r\}$.
- (e) Montrez également que pour tout $j \in \{1, \dots, r\}$, v_j est vecteur propre de $A^T A$ associé à λ_j . En déduire que $\{v_1, \dots, v_r\}$ forme une famille orthonormale de vecteurs propres de $A^T A$.
- (f) En posant $\sigma_j = \sqrt{\lambda_j}$, montrez que

$$\sum_{j=1}^r \sigma_j u_j v_j^T = \left(\sum_{j=1}^r u_j u_j^T \right) A = UU^T A$$

où $U = (u_1, \dots, u_r)$ (matrice ayant comme colonnes les vecteurs u_1, \dots, u_r).

- (g) Vérifiez que u_1, \dots, u_r forment une base de $\mathcal{Im}(AA^T)$ puis déduisez-en que UU^T est la matrice de projection sur $\mathcal{Im}(AA^T)$.
- (h) En utilisant la question 1b, en déduire que pour tout $y \in \mathcal{Im}(A)$, $UU^T y = y$ puis que $UU^T A = A$.
- (i) En déduire la SVD :

$$A = \sum_{j=1}^r \sigma_j u_j v_j^T = UDV^T$$

où $D = \text{Diag}(\sigma_1, \dots, \sigma_r)$ et $V = (v_1, \dots, v_r)$. Les vecteurs v_1, \dots, v_r sont appelés les axes principaux.

Pour rappel, on a le théorème suivant (admis, voir *e.g.* [Giraud, p.23]) :

Théorème 0.1. Notons $\mathbf{X} = (X^{(1)}, \dots, X^{(n)})^T$ (matrice $n \times p$). Avec les notations précédentes, \mathbf{X} admet une SVD de la forme $\mathbf{X} = UDV^T$. Pour tout $d \leq r := \text{rg}(\mathbf{XX}^T)$, $V_d := \text{Vect}(v_1, \dots, v_d)$ appartient à

$$\underset{\dim(\tilde{V}) \leq d}{\text{Argmin}} \sum_{i=1}^n \|X^{(i)} - \text{Proj}_{\tilde{V}} X^{(i)}\|^2.$$

où $\text{Proj}_{\tilde{V}}$ désigne la projection orthogonale sur le sous-espace vectoriel V . De plus,

$$\sum_{i=1}^n \|X^{(i)} - \text{Proj}_{V_d} X^{(i)}\|^2 = \sum_{k=d+1}^r \sigma_k^2.$$

Quelques rappels : Les vecteurs v_1, \dots, v_r sont appelés les axes principaux tandis que les vecteurs $c_1 := \sigma_1 u_1, \dots, c_r := \sigma_r u_r$ sont les composantes principales. L'intérêt de la SVD est de pouvoir lire directement la projection sur les matrices U , D et V . Plus exactement, pour tout $i \in \{1, \dots, n\}$, on a $X^{(i)} = c_1(i)v_1 + \dots + c_r(i)v_r$. De même,

$$\text{Proj}_{V_d}(X^{(i)}) = c_1(i)v_1 + \dots + c_d(i)v_d.$$

En pratique, l'analyse en composantes principales se fait sur la matrice des observations recentrées :

$$\tilde{X}^{(i)} = X^{(i)} - \frac{1}{n} \sum_{i=1}^n X^{(i)}.$$

Exercice 3 (Quelques tests numériques). .

1. Sur l'exercice 1 :
 - (a) Simulez le modèle avec en entrée : σ , p , p_1 (les gènes 1 à p_1 sont supposés négatifs) et θ (Pour simplifier, on fait l'hypothèse que quand le gène est positif, $\theta_j = \theta \neq 0$).
On pourra fabriquer une matrice stockant les observations de n individus.
 - (b) Représentez l'histogramme des valeurs des statistiques de test.
 - (c) Faites varier le seuil de rejet r et étudiez l'évolution de la FDR (False Discovery Rate) définie par

$$FDR = \mathbb{E} \left[\frac{FP}{FP + TP} \mathbf{1}_{\{FP + TP \geq 1\}} \right].$$
 où FP désigne le nombre de faux positifs et TP , le nombre de vrais positifs. Conclusions ?
 - (d) Tracez la courbe ROC du modèle.
2. Sur l'exercice 2 : on rappelle que l'ACP peut être calculée à l'aide de la commande `prcomp` (voir aussi `princomp`). Il existe également un package (rennais) `FactoMineR` qui permet d'effectuer toutes les opérations usuelles de l'ACP (et de ses variantes).
 - (a) Tentez une ACP avec un sous-échantillon de la base de données MNIST. Projetez les individus sur le plan d'ACP et produisez une représentation graphique. Que constatez-vous ?
 - (b) Testez l'algorithme des k -plus proches voisins lorsque l'espace global est remplacé par le sous-espace vectoriel constitué des d directions principales.
 - (c) On pourra aussi explorer les ACP à noyau qui sont une variante non linéaire de l'ACP.

Statistique en Grande Dimension et Apprentissage - TD Arbres/Entropie/Bootstrap/Boosting

Ce TD/TP a pour but d'illustrer le chapitre 5 sur les arbres binaires de décision, le bagging, le boosting et les forêts aléatoires.

Exercice 1. On souhaite comparer l'indice de Gini, l'erreur de classification et l'entropie.

1. Calculez ces quantités dans le cas binaire ($\{0, 1\}$) en notant p la proportion de type 1.
2. Représentez les graphes associés (on pourra utiliser R).
3. Considérons maintenant le cas d'une variable Y à valeurs dans un ensemble à 3 éléments 0, 1 et 2. Calculez les quantités (Gini, ...) lorsque $p_0 = 0.7, p_1 = 0.15, p_2 = 0.15$ et $p_0 = 0.7, p_1 = 0.25, p_2 = 0.05$. Que constatez-vous ? Comment interprétez-vous ce résultat ?
4. Pour une variable à m classes, calculez l'indice de Gini dans le cas équiprobable.
5. Montrez que l'indice de Gini au noeud t , noté $G(t)$ satisfait :

$$G(t) = 1 - \mathbb{P}(T_1 = T_2)$$

où (T_1, T_2) désigne un couple de variables aléatoires indépendantes et de loi $\mu = (\hat{p}_{t,k})_{k=1}^K$.

6. En utilisant l'inégalité de Jensen, montrez que $H_\mu \geq -\log(1 - G_\mu)$ où H_μ et G_μ sont respectivement l'entropie de Shannon et l'indice de Gini relatifs à une mesure de probabilité μ .

Exercice 2 (Entropie). Le but de cet exercice est de s'intéresser à différentes notions liées à l'entropie puis à montrer que l'entropie peut s'interpréter à l'aide de distances entre probabilités. Pour simplifier, on supposera dans toute la suite que P et Q sont des probabilités sur $\{1, \dots, K\}$ qui sont "non dégénérées", i.e. que $P(k) > 0$ et $Q(k) > 0$ pour tout $k \in \{1, \dots, K\}$.

1. La divergence (ou dissemblance) de Kullback-Lleibler de P par rapport à Q ¹ est définie par :

$$D_{KL}(P||Q) = \sum_{k=1}^K P(k) \log \frac{P(k)}{Q(k)}.$$

- (a) Quel lien a-t-on entre l'entropie de Shannon et la divergence de Kullback-Lleibler (on pensera à la loi uniforme) ?
- (b) En assimilant la divergence KL à une distance (ça n'en est pas une !), interprétez l'entropie de Shannon comme une mesure de l'hétérogénéité ou de l'instabilité d'un système (point de vue utilisé en physique).
- (c) Quel lien a-t-on entre l'entropie croisée et la divergence de Kullback-Lleibler ?
- (d) Notons $\pi(x) = (\pi_1(x), \dots, \pi_K(x))$. Déterminez la limite de "l'entropie croisée empirique" :

$$-\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K \mathbf{1}_{Y_i=k} \log(\pi_k(X_i)).$$

Exprimer cette limite à l'aide de la divergence de Kullback-Lleibler.

- (e) Notons \mathcal{S} un ensemble de fonctions $F = (F_1, \dots, F_K)$ de \mathbb{R}^p dans \mathbb{R}^K . Notons π_F la fonction de \mathbb{R}^p dans $\mathcal{P}(\{1, \dots, K\})$ (l'ensemble des probabilités sur $\{1, \dots, K\}$) défini par :

$$\forall k \in \{1, \dots, K\}, \quad (\pi_F(x))_k = \frac{e^{F_k(x)}}{\sum_{j=1}^K e^{F_j(x)}} \quad (\text{Softmax})$$

1. On utilise aussi la terminologie "entropie relative".

Montrez que si \mathcal{S} est fini (pour simplifier), déterminer

$$\operatorname{Argmin}_{F \in \mathcal{S}} \left(-\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K 1_{Y_i=k} \log(\pi_k(X_i)) \right)$$

revient asymptotiquement (en N) à déterminer

$$\operatorname{Argmin}_{F \in \mathcal{S}} \mathbb{E}[D_{KL}(\mathcal{L}(Y|X), \pi_F(X))].$$

2. Propriétés de la divergence KL et liens avec la distance en variation totale.

- (a) Vérifiez à l'aide de l'inégalité de Jensen et de la convexité de $x \mapsto x \log x$ que $D(P||Q) \geq 0$ et que $D(P||Q) = 0$ si et seulement si $P = Q$.
- (b) Montrez que pour tous $x, y \in]0, 1[$,

$$\kappa(x, y) := x \ln \left(\frac{x}{y} \right) + (1-x) \ln \left(\frac{1-x}{1-y} \right) \geq 2(x-y)^2.$$

Plus précisément, on va faire le lien avec la distance en *variation totale*.

- (c) Notons $B_+ = \{k, P(k) > Q(k)\}$. Montrez que

$$\sum_k |P(k) - Q(k)| = 2 \left(\sum_{k \in B_+} P(k) - \sum_{k \in B_+} Q(k) \right).$$

- (d) Notons $P_{B_+} = \sum_{k \in B_+} P(k)$ et $Q_{B_+} = \sum_{k \in B_+} Q(k)$. Montrez que

$$D_{KL}(P||Q) \geq \kappa(P_{B_+}, Q_{B_+}).$$

- (e) Notons P et Q deux probabilités sur $\{1, \dots, m\}$. La distance en variation totale de P à Q est définie par :

$$d_{TV}(P, Q) = \frac{1}{2} \sum_{k=1}^m |P(k) - Q(k)|.$$

Déduisez l'inégalité de Pinsker de ce qui précède :

$$D(P||Q) \geq \frac{1}{2} d_{TV}(P, Q)^2.$$

Exercice 3 (Bootstrap). 1. Soit Z_1, \dots, Z_q des variables aléatoires de même loi telles que $\operatorname{Var}(Z_1) = \sigma^2$ et telles que $\operatorname{Corr}(Z_i, Z_j) = \rho$ (indépendant de i et j). Montrez que

$$\operatorname{Var} \left(\frac{1}{q} \sum_{k=1}^q Z_i \right) = \rho \sigma^2 + \frac{1-\rho}{q} \sigma^2.$$

En appliquant ce résultat au bagging, discuter des conditions importantes pour que le bagging soit efficace.

- 2. Montrez que la probabilité que l'observation j n'appartienne pas à un échantillon bootstrap de taille n est égale à $(1 - \frac{1}{n})^n$. En déduire la proportion asymptotique d'observations n'appartenant pas à l'échantillon bootstrap (quantité à l'origine de l'estimateur bootstrap 0.632, voir par exemple https://perso.univ-rennes2.fr/system/files/users/fromont_m/Atelier2_bis.pdf p.90).

Exercice 4 (Adaboost et fonction de perte exponentielle). Supposons que $\mathcal{Y} = \{-1, 1\}$ et considérons la fonction de perte :

$$\ell(y, y') = \exp(-yy').$$

Pour une fonction $F : \mathcal{X} \rightarrow \mathbb{R}$, notons R le risque empirique associé :

$$R(F) = \frac{1}{n} \sum_{i=1}^n \exp(-Y_i F(X_i)).$$

1. Considérons une fonction $h : \mathcal{X} \rightarrow \{-1, 1\}$ et notons $\omega_i = \exp(-y_i F(x_i))$. Exprimez

$$\frac{\partial R(F + \alpha h)}{\alpha}$$

en fonction de α et des ω_i .

2. En déduire que la fonction $\alpha \mapsto \frac{\partial R(F + \alpha h)}{\alpha}$ atteint son minimum en

$$\alpha^* = \frac{1}{2} \log \left(\frac{1 - \mathcal{E}(F)}{\mathcal{E}(F)} \right),$$

où

$$\mathcal{E}(F) = \frac{\sum_{i=1}^n \omega_i \mathbf{1}_{h(x_i) \neq y_i}}{\sum_{i=1}^n \omega_i}$$

3. Au vu de ce qui précède, interpréter le choix de α_m dans l'algorithme Adaboost.

Exercice 5 (Forêts aléatoires/Boosting).

1. Ecrire formellement l'algorithme des forêts aléatoires.
2. Même question avec une méthode de gradient boosting avec fonction de perte quadratique.

Statistique en Grande Dimension et Apprentissage - TD 3

Ce TD a pour but d'illustrer le chapitre 3 principalement sur la régression linéaire pénalisée.

Exercice 1 (Quelques propriétés standards sur le modèle linéaire classique). On considère le modèle $\mathbf{Y} = \mathbf{X}\theta^* + \varepsilon$ avec $\mathbf{Y} \in \mathbb{R}^n$, \mathbf{X} matrice $n \times p$, $\theta^* \in \mathbb{R}^p$ et $\varepsilon \sim \mathcal{N}(0, \sigma^2 I_p)$. et $F(\theta) = \frac{1}{2n} \|\mathbf{Y} - \mathbf{X}\theta\|^2$. On suppose que $\mathbf{X}^T \mathbf{X}$ est inversible.

1. Retrouvez l'expression de $\hat{\theta} = \text{Argmin}_\theta F(\theta)$.
2. Montrez que $\hat{\theta} = \theta^* + (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \varepsilon$. En déduire la loi de $\hat{\theta} - \theta^*$.
3. Quel résultat retrouve-t-on lorsque $\mathbf{X} = (1, \dots, 1)^T$?
4. On considère le modèle $Y = \theta_0 + \sum_{i=1}^p \theta_i X_i + \varepsilon$. A l'aide du théorème de Cochran, montrez que

$$\frac{SCR}{\sigma^2} \sim \chi^2(n - p - 1)$$

Sous l'hypothèse $(H_0) := \theta_1 = \dots = \theta_p = 0$, montrez que

$$\frac{SCT}{\sigma^2} \sim \chi^2(n - 1).$$

Exercice 2 (soft-thresholding). Le but de cet exercice est de comprendre l'expression de $\hat{\theta}^{LASSO}$ à partir de l'étude du cas unidimensionnel. On pose pour tout $\theta \in \mathbb{R}$,

$$\phi(\theta) = \frac{1}{2N} \sum_{i=1}^N (y_i - z_i \theta)^2 + \lambda |\theta|.$$

où y_i, z_i sont des réels et $\lambda > 0$. On fait également l'hypothèse que $\frac{1}{N} \sum_{i=1}^n z_i^2 = 1$.

1. Calculez la dérivée de $\theta \mapsto \phi(\theta)$ (sur \mathbb{R}^*).
2. Montrez qu'une fonction strictement convexe et coercive admet un unique minimum (illustrer par un dessin). On note $\hat{\theta}$ ce minimum dans la suite.
3. Etablir le tableau de variations de $\theta \mapsto \phi(\theta)$ lorsque $\frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} > \lambda$. En déduire $\hat{\theta}$ dans ce cas.
4. Montrez que

$$\hat{\theta} = \begin{cases} \frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} - \lambda & \text{si } \frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} > \lambda \\ 0 & \text{si } \frac{|\langle \mathbf{z}, \mathbf{y} \rangle|}{N} < \lambda, \\ \frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} + \lambda & \text{si } \frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} < -\lambda. \end{cases}$$

5. En déduire que

$$\hat{\theta} = \mathcal{S}_\lambda \left(\frac{\langle \mathbf{z}, \mathbf{y} \rangle}{N} \right)$$

où

$$\mathcal{S}_\lambda(x) = \text{sgn}(x)(|x| - \lambda)_+.$$

\mathcal{S}_λ est appelé l'opérateur de seuillage doux (soft-thresholding operator).

Exercice 3 (Un cas de non-unicité). Considérons la fonction ϕ définie sur \mathbb{R}^2 par

$$\phi(\theta) = \frac{1}{2} (1 - \theta_1 - \theta_2)^2 + \lambda \|\theta\|_1.$$

Montrez que ϕ admet une infinité de minima lorsque $\lambda < 1$.

Exercice 4 (Coordinate Descent). Pour rappel, la descente coordonnées par coordonnées est un algorithme de recherche de $\hat{\theta}_\lambda$ basé sur le principe de minimiser successivement la fonction L de l'exercice précédent sur une seule des variables. Plus précisément, considérons la fonction L définie par :

$$L(\theta) = \frac{1}{2N} \|\mathbf{y} - \mathbf{X}\theta\|_2^2 + \lambda \|\theta\|_1.$$

Remarque : Selon les exercices, le coefficient devant $\|\mathbf{y} - \mathbf{X}\theta\|_2^2$ peut changer (1 , $1/(2N)$, $1/N$). Notez que quitte à changer la valeur de λ , les problèmes de minimisation restent équivalents.

- On note $\theta^{-j} = (\theta_1, \dots, \theta_{j-1}, \theta_{j+1}, \dots, \theta_p)$. Pour un tel vecteur, on note $L_j^{\theta^{-j}}$ l'application de \mathbb{R} dans \mathbb{R} définie par : $L_j^{\theta^{-j}}(\beta) = L(\theta_1, \dots, \theta_{j-1}, \beta, \theta_{j+1}, \dots, \theta_p)$. Montrer que

$$L_j^{\theta^{-j}}(\beta) = \frac{1}{2N} \sum_{i=1}^N (r_i^{(j)} - z_i^{(j)}\beta)^2 + \lambda \sum_{k \neq j} |\theta_k| + \lambda |\beta|$$

où $r_i^{(j)}$ et $z_i^{(j)}$, $i = 1, \dots, n$ désignent des réels que l'on explicitera.

- Quitte à normaliser la matrice $\mathbf{X} = (x_{ij})_{i,j}$, on fait l'hypothèse que $\frac{1}{N} \sum_{j=1}^N x_{i,j}^2 = 1$ pour tout $i \in \{1, \dots, n\}$. A l'aide de l'exercice précédent, montrez que la fonction $L_j^{\theta^{-j}}$ atteint son minimum en

$$\hat{\beta} = \mathcal{S}_\lambda \left(\frac{1}{N} \langle \mathbf{x}_j, \mathbf{r}^{(j)} \rangle \right)$$

où $\mathbf{r}^{(j)} = (r_1^{(j)}, \dots, r_n^{(j)})^T$ et \mathbf{x}_j désigne la j -ième colonne de la matrice \mathbf{X} .

L'algorithme de descente coordonnées par coordonnées fonctionne alors de la manière suivante. On initialise θ . On pose $\theta^{(0)} = 0$ par exemple. On considère alors la fonction

$$\beta \mapsto L_1^{(\theta^{(0)})^{-1}}(\beta)$$

dont on calcule le minimum $\hat{\beta}$ (Il se trouve qu'il est explicite !!). On définit alors $\theta^{(1)}$ par $\theta_1^{(1)} = \hat{\beta}$ et pour tout $j \neq 1$, $\theta_j^{(1)} = \theta_j^{(0)}$. A l'étape $j \leq p$, on dispose d'un $\theta^{(j)}$ et on reproduit le même schéma en figeant les variables différentes de j et en cherchant le minimum de $j \mapsto L_j^{(\theta^{(j)})^{-1}}$. Une fois les p variables passées, on recommence avec la première. . . .

3. Montrez que la suite $(L(\theta^{(n)}))_n$ est décroissante.
4. Montrez que $(\theta^{(n)})$ admet une sous-suite convergente vers $\theta^{(\infty)} \in \mathbb{R}^p$.
5. Pourriez-vous expliquer sans le démontrer que $\theta^{(\infty)}$ est nécessairement un minimum ?
6. En déduire que $L(\theta^{(n)})$ converge vers $\min_{\theta} L(\theta)$.

Exercice 5. 1. Retrouvez la formule de $\hat{\theta}^{\text{Ridge}}$.

2. On considère la fonction L définie par $L(\theta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\theta\|^2 + \lambda\|\theta\|_1$. Calculer le sous-différentiel de L .
3. En déduire que $\hat{\theta}_\lambda$ est minimum de L si et seulement s'il est solution du système :

$$\begin{cases} \mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\theta) = \lambda \text{sgn}(\theta_j) & \text{si } j \in J(\theta) \\ |\mathbf{x}_j^T(\mathbf{y} - \mathbf{X}\theta)| \leq \lambda & \text{sinon.} \end{cases} \quad (1)$$

où $J(\theta) = \{j \in \{1, \dots, p\}, \theta_j \neq 0\}$. Montrez que dans le cas $\mathbf{X}^T \mathbf{X} = I_p$, alors

$$\hat{\theta}_\lambda(j) = \text{sgn}(\hat{\theta}_j)(|\hat{\theta}_j| - \lambda)_+$$

où $\hat{\theta} = \hat{\theta}_0$, i.e. désigne la solution du système non pénalisé (bien définie dans ce cas).

4. On suppose que $\frac{\mathbf{X}^T \mathbf{X}}{n} = I_p^{-1}$ et que le modèle est de la forme $\mathbf{Y} = \mathbf{X}\theta^* + \varepsilon$. Vérifiez que l'on peut écrire

$$(\hat{\theta}_\lambda)(j) = \mathcal{S}_{\frac{\lambda}{n}}\left(\frac{\langle \mathbf{x}_j, \mathbf{y} \rangle}{n}\right) = \mathcal{S}_{\frac{\lambda}{n}}\left(\theta_j^* + \frac{(\mathbf{X}^T \varepsilon)_j}{n}\right).$$

Exercice 6 (λ_{\max}). Dans cet exercice, on se pose la question suivante. Peut-on calculer la valeur de λ à partir de laquelle, le LASSO ne détecte aucune variable ? On pose

$$L(\theta) = \frac{1}{2}\|\mathbf{y} - \mathbf{X}\theta\|^2 + \lambda\|\theta\|_1.$$

1. Montrez que

$$L(\theta) - L(0) = \frac{1}{2}\|\mathbf{X}\theta\|_2^2 - \langle \mathbf{y}, \mathbf{X}\theta \rangle + \lambda\|\theta\|_1.$$

2. Montrez que

$$\langle \mathbf{y}, \mathbf{X}\theta \rangle = \sum_{j=1}^p \theta_j \langle \mathbf{x}_j, \mathbf{y} \rangle.$$

3. On pose

$$\lambda_{\max} = \max\{|\langle \mathbf{x}_j, \mathbf{y} \rangle|, j = 1, \dots, p\}.$$

Déduire de la question précédente que pour tout $\lambda > \lambda_{\max}$, $L(\theta) - L(0) > 0$ pour tout $\theta \in \mathbb{R}^p$. Que vaut $\hat{\theta}_\lambda$ dans ce cas ?

4. Supposons maintenant $\lambda < \lambda_{\max}$. Montrez que dans ce cas, il existe $j \in \{1, \dots, p\}$ tel que $\theta_j \mapsto L(0, \dots, 0, \theta_j, 0, \dots, 0)$ n'atteint pas son minimum en 0. Déterminez-le. Conclure que

$$\lambda_{\max} = \sup\{\lambda, \hat{\theta}_\lambda \neq 0\}.$$

1. On omet souvent la division par n pour simplifier les notations mais elle est naturelle : pensez par exemple au cas où $\mathbf{Y} = \theta^* + \varepsilon$ (Que vaut $\mathbf{X}^T \mathbf{X}$ dans ce cas?). Elle permet de faire apparaître la convergence vers θ^* .

5. Interprétation : que pensez-vous de la valeur de λ_{max} ? Cela vous semble-t-il naturel que le LASSO détecte d'abord la variable qui maximise $|\langle \mathbf{x}_j, \mathbf{y} \rangle|$?

Exercice 7. Calculer l'erreur L^2 du Ridge. Lorsque $\mathbf{X}^T \mathbf{X} = I_p$, en déduire des bornes (voir <https://www.imo.universite-paris-saclay.fr/~giraud/MAP433/PC7-8-correction.pdf>).

Statistique en Grande Dimension et Apprentissage - TD 4

Ce TD a pour but d'illustrer le chapitre 4 sur les SVM.

Exercice 1 (Hyperplans). 1. Rappelez pourquoi l'équation d'un hyperplan affine \mathcal{H} s'écrit

$$\langle \beta, x \rangle + \beta_0 = 0.$$

2. Montrez que pour un tel hyperplan \mathcal{H} la distance d'un point x à \mathcal{H} est égale à :

$$d(x, \mathcal{H}) = \frac{|\langle \beta, x \rangle + \beta_0|}{\|\beta\|}.$$

3. Quelle est l'équation d'un hyperplan parallèle à \mathcal{H} ?

Exercice 2 (Optimisation/Lagrangien). On se donne un échantillon $(x_i, y_i)_{i=1}^N$ où $y_i \in \{-1, 1\}$. On cherche à résoudre

$$\begin{cases} \min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2 \text{ sous la contrainte} \\ \forall i \in \{1, \dots, N\}, \quad y_i(\langle \beta, x_i \rangle + \beta_0) \geq 1. \end{cases} \quad (1)$$

1. Ecrire le Lagrangien $L(\beta, \beta_0, \alpha)$ associé à ce problème (où $\alpha = (\alpha_1, \dots, \alpha_N)$)
2. Ecrire les conditions de stationnarité : $\partial_\beta L = 0$ et $\partial_{\beta_0} L = 0$.
3. Notons $(\beta(\alpha), \beta_0(\alpha))$ un point critique de $(\beta, \beta_0) \mapsto L(\beta, \beta_0, \alpha)$. Pourquoi $(\beta(\alpha), \beta_0(\alpha))$ correspond-il à un minimum de $(\beta, \beta_0) \mapsto L(\beta, \beta_0, \alpha)$?
4. Montrez que

$$\tilde{L}(\alpha) = L(\beta(\alpha), \beta_0(\alpha), \alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle.$$

On rappelle que le problème (1) se ramène au problème suivant :

$$\begin{cases} \max_{\alpha \in (\mathbb{R}^+)^N} \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{avec } \sum_{i=1}^N \alpha_i y_i = 0. \end{cases}$$

Les conditions complémentaires de KKT garantissent de plus que si α^* désigne un tel maximiseur, alors pour tout i , soit $\alpha_i^* = 0$ soit $\alpha_i^* > 0$ et $y_i(\langle \beta^*, x_i \rangle + \beta_0^*) - 1 = 0$ (Les multiplicateurs de Lagrange non nuls correspondent aux vecteurs supports).

5. Montrez que $\beta^* = \sum_{i=1}^N \alpha_i^* y_i x_i$ puis que

$$\beta_0^* = -\frac{1}{2} \left(\min_{y_i=1} \langle \beta^*, x_i \rangle + \max_{y_i=-1} \langle \beta^*, x_i \rangle \right).$$

Exercice 3 (Noyau polynomial). Soit $r \in \mathbb{N}$ tel que $r \geq 2$. On pose pour tous $x, x' \in \mathbb{R}^2$,

$$k(x, x') = (1 + \langle x, x' \rangle)^r.$$

1. Montrez qu'il existe un entier $m \geq 2$ et une application $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^m$ telle que pour tous $x, x' \in \mathbb{R}^2$,

$$\langle \phi(x), \phi(x') \rangle_{\mathbb{R}^m} = k(x, x').$$

On précisera la valeur de m .

2. Si on reprend la question précédente avec x, x' dans \mathbb{R}^p , quelle est la dimension de l'espace \mathcal{H} associé (On pourra commencer par le cas $r = 2$) ?
3. Retrouver que k est un noyau sans construire ϕ (On s'appuiera sur l'exercice 4).

Exercice 4 (Opérations sur les noyaux). Montrez que la somme et le produit de deux noyaux sur un même espace est encore un noyau.

Exercice 5 (SVM Multi-Classes). Différentes approches existent pour traiter les SVM multi-classes.

1. Approche globale : Ecrire un problème d'optimisation (primal) permettant de formuler les exigences d'un SVM linéaire (non flexible) dans le cas d'un problème à m classes.
2. Approche *un contre un* : on entraîne des SVM bi-classes puis on fabrique une règle globale. Sauriez-vous décrire plus en détail une telle approche ?
3. Approche *un contre tous* : A votre avis, à quoi cette approche correspond-elle ?

Exercice 6. Soient $\{(x_i, y_i), i \in \{1, \dots, N\}\}$ un échantillon inclus dans $\mathbb{R}^2 \times \{-1, 1\}$. On suppose que si $y_i = 1$, alors $x_i \in B(x_0, 1) = \{(a, b), (a - a_0)^2 + (b - b_0)^2 < 1\}$ ($x_0 = (a_0, b_0)$) et si $y_i = -1$, alors $x_i \in \{(a, b), (a - a_0)^2 + (b - b_0)^2 > 1\}$. En utilisant l'application ϕ définie par $\phi(a, b) = (a^2, b^2, a, b)$, montrez que les points sont linéairement séparables dans \mathbb{R}^4 .

Statistique en Grande Dimension et Apprentissage - TD 6

Ce TD a pour but d'illustrer le chapitre 6 sur les réseaux de neurones.

Exercice 1. Montrez que softmax et sigmoïde “coïncident” en classification binaire.

Exercice 2. Considérons un perceptron à une couche cachée avec 3 neurones et une sortie quantitative et une entrée de dimension 2. Précisez les paramètres à estimer ainsi que la dimension de l'espace Θ .

Exercice 3 (Approximation Universelle). 1. On rappelle que le théorème de Pinkus indique que celui de Hornik (Théorème 6.3.1) est vrai si et seulement si ϕ est une fonction non polynomiale. Expliquez pourquoi lorsque ϕ est polynomiale, le théorème ne peut être vrai.

2. On se place en dimension 1. On pose $\mathcal{F} = \text{Vect}\{x \mapsto \phi(\lambda x + b), \lambda, b \in \mathbb{R}\}$. Montrez que si ϕ est une fonction \mathcal{C}^∞ qui n'est pas un polynôme, alors :

- (a) les fonctions constantes sont dans \mathcal{F} ,
- (b) les fonctions linéaires sont dans l'adhérence de \mathcal{F} pour la topologie de la convergence uniforme, *i.e.* les fonctions linéaires peuvent s'écrire comme limites (uniformes) de fonctions de \mathcal{F} (penser au taux d'accroissement)
- (c) en généralisant, l'idée précédente, montrer que pour tout k , les polynômes de degré k appartiennent à l'adhérence de \mathcal{F} .
- (d) En déduire le résultat par le théorème de Weierstrass (pour les polynômes).

3. Enoncer le théorème de Stone-Weierstrass.

Exercice 4 (Dropout/Bagging). 1. Expliquez le principe du dropout en phase d'entraînement.

2. Supposez que le dropout soit réalisé comme un bagging. Quel serait le prédicteur naturel de $\mathbb{P}(Y|X = x)$ lorsque la probabilité de désactiver un neurone est égale à $1/2$. Quel serait-il lorsque cette probabilité est égale à $p \neq 1/2$?

3. Comment interprétez-vous l'explication :“In Testing, Use all activations, but reduce them by a factor p (to account for the missing activations during training).” (phrase issue du blog de A. Budhiraja).

N.B. : la reconstruction utilisée dans les modules de Deep Learning n'est pas tout à fait claire sur ce point (il faudrait plus explorer la bibliographie, par exemple <https://www.deeplearningbook.org/contents/regularization.html> pour clarifier ce point).

Exercice 5 (Gradient/Gradient Sto./RMSprop/Adagrad/Adam). Soit L définie par :

$$L(\theta) = \frac{1}{N} \sum_{k=1}^N \ell(Y_k, f_\theta(X_k)).$$

1. Rappelez la forme de la descente de gradient standard.

2. Rappelez la forme de la descente de gradient stochastique.
 3. Rappelez la forme de la descente de gradient mini-batch.
 4. Dans les modules de Deep Learning (Keras/Pytorch...), la méthode utilisée est généralement une variation d'une descente de gradient où le pas est adaptatif.
- (a) Adagrad : notons $Q_k(\theta) = \ell(Y_k, f_\theta(X_k))$. Alors, Adagrad est défini par :

$$\theta^{\text{new}} = \theta^{\text{old}} - \frac{\eta}{\sqrt{\sum_{k=1}^N |\nabla Q_k(\theta^{\text{old}})|^2}} \nabla L(\theta^{\text{old}}).$$

A votre avis, quel est l'intérêt d'une telle modification de la descente de gradient ? Comment l'écririez-vous en mode mini-batch ?

- (b) RMSprop : RMSprop est une version du gradient stochastique où celui-ci est moyenisé sur son passé proche (à ne pas confondre avec l'Averaged Gradient Descent...). En version gradient standard, cela donne :

$$\theta_n = \theta_{n-1} - \frac{\eta}{\sqrt{v_{n-1}}} \nabla \ell(Y_{i_n}, f_{\theta_{n-1}}(X_{i_n}))$$

où i_n est tiré aléatoirement uniformément parmi les observations et :

$$v_n = (1 - \rho)v_{n-1} + \rho|\nabla Q_{i_n}(\theta_{n-1})|^2.$$

Commentez cet algorithme et l'intérêt de cette modification. On pourra aussi montrer que l'influence des gradients précédents décroît géométriquement avec la mémoire.

- (c) Adam : la plus utilisée en pratique. C'est une amélioration de RMSProp. Expliquez son fonctionnement (en vous aidant d'une documentation).

Exercice 6 (Convergence de la descente de gradient). Supposons $L \in \mathcal{C}^2$ uniformément strictement convexe coercive.

1. On considère l'équation différentielle $\dot{\theta}_t = -\nabla L(\theta_t)$. Montrez que $|\theta_t - \theta^*|^2 \rightarrow 0$ lorsque $t \rightarrow +\infty$ où θ^* désigne l'unique minimum de V et $|\cdot|$ désigne la norme euclidienne.
2. Considérons l'algorithme $\theta_{n+1} = \theta_n - \gamma \nabla L(\theta_n)$ avec $\gamma > 0$ telle que la matrice Hessienne est une fonction bornée. Montrez que la convergence a lieu si γ est assez petit.
3. Supposons que $L(\theta) = \mathbb{E}[\ell(\theta, Z)]$ où Z désigne une variable aléatoire. Expliquez intuitivement pourquoi, si L satisfait les hypothèses de la question 2, alors (θ_n) définie par $\theta_{n+1} = \theta_n - \gamma \nabla_\theta \ell(\theta_n, Z_{n+1})$ est proche de θ^* pour γ assez petit (pour pallier le problème du γ assez petit, on peut aussi prendre un pas γ_n décroissant vers 0 tel que $\sum_n \gamma_n = +\infty$).

Exercice 7. Détaillez la structure d'un réseau de neurones récurrent en général puis d'un LSTM (on pourra s'appuyer sur de la documentation en ligne).

Statistique en Grande Dimension et Apprentissage - TP 1

Ce TP est consacré aux k -plus proches voisins. On utilisera de préférence le logiciel R. A travers la mise en oeuvre de cet algorithme, les objectifs sont de se familiariser avec le calcul empirique du risque, la validation simple, la validation croisée ainsi que le choix de paramètres (ou hyper-paramètres). On abordera aussi la notion de surapprentissage.

Exercice 1 (Données simulées). On se place ici dans le cadre suivant. On suppose que \mathbf{X} suit la loi uniforme sur $[0, 1]^2$ et que Y est une variable aléatoire à valeurs dans $\{0, 1\}$ telle que

$$\mathbb{P}(Y = 1 | \mathbf{X} = x) = \begin{cases} \alpha & \text{si } |X_1 + 2X_2| \leq 1 \\ \beta & \text{si } |X_1 + 2X_2| > 1. \end{cases}$$

1. Ecrire une fonction pour générer un échantillon \mathcal{D}_n de taille n .
2. Simulez un échantillon d'entraînement ($\mathbf{X}_{\text{train}}, Y_{\text{train}}$) de taille 100 et un échantillon test de taille 100 également que vous noterez ($\mathbf{X}_{\text{test}}, Y_{\text{test}}$).
3. Faire une représentation graphique de l'échantillon d'entraînement en attribuant une couleur différente et un symbole différent selon la valeur de Y (paramètres, `col` et `pch` de la fonction `plot`).
4. Appliquez la fonction `knn` du package `class` (écrire `library(class)`), pour prédire les points de coordonnées $(1/2, 1/2)$ et $(1/4, 3/4)$ (avec par exemple $k = 5$).
5. Appliquez la fonction `knn`, pour prédire les labels de l'échantillon test avec $k = 1$ et $k = 20$.
6. Tracez un graphe affichant bien/mal classés et leur classes.
7. A l'aide de la fonction `table`, afficher la matrice de confusion (qui permet de visualiser les classements prédits et les classements réels) pour ce test.
8. En déduire (l'estimation de) l'erreur de classification associée.
9. Calculez l'erreur par validation simple pour $k = 1 : 20$ et tracez l'évolution sur un graphe. Que décidez-vous ?
10. Calculez l'erreur test pour ce choix optimal. Que constatez-vous ? (On comparera à l'erreur de validation simple)
11. Calculez l'erreur par validation croisée sur tout l'échantillon en utilisant la fonction `knn.cv` (validation croisée de type Leave-One-Out) puis en programmant une validation croisée 5-fold. Que constatez-vous ? Quel est le choix optimal de k dans le cas 5-fold ?
12. Comparez l'erreur de validation croisée à l'erreur test pour le choix optimal de l'hyper-paramètre k .

Exercice 2. Récupérer le jeux de données `pulsar` et les afficher dans R. Il s'agit d'un jeu de données astrophysique dont chaque ligne correspond à un objet du ciel nocturne. A partir de 8 paramètres, l'objectif est d'être capable de prévoir si cet objet est un pulsar ou non

1. Fabriquer aléatoirement un échantillon d'entraînement et un échantillon test (de même taille).
2. Représenter une projection en $2d$ des données d'entraînement (et de leur label). Qu'en pensez-vous ?
3. Fabriquer un algorithme de k -plus proches voisins adapté à ce problème (en choisissant l'hyperparamètre par une procédure de validation simple ou croisée avec les proportions de votre choix).
4. Erreur de classification “optimale”? F_1 -score ?
5. Affichez pour $k = 3$ et $k = 30$, le graphe projeté des bien/mal classés ainsi que des classes correspondantes.
6. On pourra aussi représenter la *frontière de décision* pour $k = 30$ puis $k = 15$, puis $k = 1$. Dans quels cas peut-on parler de sur ou sous-apprentissage?
Pour fabriquer ce graphe, calculez la prédiction sur une grille de points adaptée aux données, puis représentez la prédiction des points de cette grille en leur attribuant une couleur.

Exercice 3. Dans ce denier exercice, on s'intéresse à une base de données célèbre : la base de données MNIST qui contient 70000 images de chiffres en noir et blanc et en 28×28 pixels.

1. Chargez les jeux de données `mnist_train.csv` et `mnist_test.csv`.
2. En remarquant que la première colonne est celle des réponses, fabriquez des `data.frame`(s) `Xtrain`, `Ytrain`, `Xtest` et `Ytest`.
3. Dans ce jeu de données, on atteint un peu les limites algorithmiques des k -plus proches voisins dont la complexité est en $O(nkd)$ où d est la dimension du problème (ici, $d = 784$).
4. Représentez l'une des observations sous la forme d'un tableau 28×28 afin de vous familiariser avec l'objet.
5. Tentez la mise en oeuvre de l'algorithme avec seulement 10 données test. Précédez la commande de `t1=Sys.time()` et écrivez ensuite `t1=Sys.time(); difftime(t2,t1)` afin d'évaluer le temps de calcul. Que pouvez-vous en déduire sur le temps de calcul de la base de données test complète ?
6. On propose donc dans la suite de travailler seulement avec un sous-échantillon d'entraînement de 12000 observations. Fabriquez ce sous-échantillon en utilisant la fonction `sample`.
7. Mettez en oeuvre l'algorithme sur ce sous-échantillon d'entraînement avec une base test de 500 observations (pour limiter le temps de calcul) et $k = 10$ voisins.
8. Représentez la matrice de confusion et calculez l'erreur de classification.

Statistique en Grande Dimension et Apprentissage TP “Arbres/Random Forests/Boosting”

Exercice 1 (Illustration méthode CART sur des données spam).

1. Charger la librairie `rpart`. Consulter l'aide de ce package.
2. Diviser l'échantillon en une partie `train` et une partie `test` (1/4 de la population) aléatoirement.
3. Construction de l'arbre de décision :

```
spam.rpart=rpart(formula = yesno . ,method="class",control=rpart.control(cp=0.001),  
data=spam7)
```

4. A l'aide de la fonction `plotcp`, affichez l'évolution de l'erreur de prédiction (plus précisément, le pourcentage relatif à celle d'origine).
5. Quel rôle joue c_p ? Faites-le varier.
6. Affichez l'arbre maximal :

```
library(rpart.plot)  
prp(spam.rpart,extra=1)
```
7. Pour obtenir l'arbre élagué, utilisez la fonction `prune`.

```
prune(spam.rpart,cp=???)
```
8. Evaluation des performances : sur l'échantillon d'entraînement, puis sur l'échantillon test sur l'arbre optimal, puis sur l'arbre maximal.

Exercice 2 (Illustration Bagging/Random Forests/Boosting sur des données spam). Diviser l'échantillon en une partie test (1/3) et une partie entraînement (2/3).

1. Bagging.
 - (a) Utilisez la fonction `bagging` sur l'échantillon d'entraînement.

```
bagging(yesno . ,coob = TRUE,data = spam.train,nbagg = 500).
```

On pourra éventuellement faire varier le nombre d'arbres en jeu (attention au temps de calcul).
 - (b) Affichez l'objet.
 - (c) Estimer l'erreur de prédiction sur l'échantillon test.
 - (d) Là encore, on pourra regarder l'évolution (pour $nbagg = 100, 200, 300, 400, 500$ par exemple).
2. Random Forests.
 - (a) Charger la librairie Random Forests : `library(randomForest)`. Consulter l'aide.
 - (b) Utilisez la fonction `randomForest` avec l'échantillon d'entraînement (pour $ntree = 100, 200, 300, 400, 500$ par exemple).
 - (c) Affichez l'objet (ou faites un `summary`).
 - (d) Estimer l'erreur de prédiction sur l'échantillon test. Conclusions
 - (e) Tracez le graphe d'importance des variables à l'aide de la fonction `varImpPlot`.
3. Boosting.

- (a) Chargez la librairie `gbm`.
 - (b) Utilisez la fonction `gbm..`
 - (c) Mêmes questions que précédemment.
 - (d) Affichage des variables d'importance.
4. XGBoost.
- (a) Installez le package `xgboost`.
 - (b) Explorez l'aide et tentez de mettre en oeuvre **XGBoost** sur cette base de données.
 - (c) Comparez les résultats avec les méthodes précédentes.

Exercice 3. Testez les outils précédents sur une base de données “Optical Recognition of Handwritten Digits”, <http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits>.

Ce TP a pour but de terminer les illustrations numériques du TP sur les arbres et extensions agrégées puis d'enchaîner sur le chapitre 4 et les SVM.

Exercice 4 (Random Forest/XGBoost en Python). On s'intéresse ici à la mise en application de XGBoost pour l'apprentissage de données cancer (Base de données “Breast Cancer Wisconsin”). On pourra utiliser Python.

Objectif : mettre en oeuvre XGBoost sur ces données (utiliser le code prévu à cet effet issu d'un (excellent) rapport de TP de M. Cordier.

Statistique en Grande Dimension et Apprentissage - TP Chapitre 3

Ce TP a pour but d'illustrer le chapitre 3 principalement sur la régression pénalisée.

Exercice 1 (Exemple simulé). 1. Ecrire une fonction de n, p, θ et σ permettant de générer le vecteur \mathbf{Y} de taille n donné par

$$\mathbf{Y} = \mathbf{X}\theta + \varepsilon$$

où \mathbf{X} est une matrice $n \times p$ dont les coordonnées sont i.i.d de loi $\mathcal{N}(0, 1)$ et ε est un vecteur de taille n i.i.d. de loi $\mathcal{N}(0, \sigma^2)$.

2. On fixe $n = 100$,

$$\theta^* = \left(\underbrace{1, \dots, 1}_{s/2 \text{ fois}}, \underbrace{-1, \dots, -1}_{s/2 \text{ fois}}, \underbrace{0, \dots, 0}_{p-s \text{ fois}} \right).$$

$p = 200$ et $\sigma = 0.5$. Créez un objet LASSO_ex1 issu de l'application de la fonction `glmnet` à ce cadre avec une pénalisation LASSO.

3. Affichez les résultats et commentez-les dans les deux cas.
4. Tracez le chemin de régularisation (apparition des coefficients) via la commande `plot(LASSO_ex1)`.
5. Utilisez la fonction `coef` pour une ou plusieurs valeurs pour évaluer l'estimation de θ .
6. Testez la fonction `predict` : exemple. `predict(LASSO_t, newx=nx, s=c(0.1, 0.05))`.
7. Choix du λ : à l'aide de la fonction `cv.glmnet`, on peut effectuer une validation croisée pour différentes de valeurs de λ . Créez un objet `cv_Ex1` issue de l'application de cette fonction. La commande `plot(cv_Ex1)` permet de représenter l'évolution de l'erreur de prédiction de validation croisée en fonction de $\log(\lambda)$. Que représentent $\lambda.min$ et $\lambda.1se$?
8. On souhaite évaluer la qualité de l'erreur sur un échantillon test.
 - (a) Générez un échantillon test.
 - (b) A l'aide de la fonction `predict`, calculez la prédiction pour chaque noyau X_i simulé : `predict(cv_Ex1glmnet.fit, ValSetX, s=bestLambdaLasso)`.
 - (c) Que vaut la MSE empirique sur cet échantillon ?
 - (d) Comparez le R^2 obtenu via le LASSO avec le λ sélectionné par validation croisée et celui obtenu via le modèle linéaire classique en conservant uniquement les variables sélectionnées (dans le cas $p = 200$).
9. Reproduire certains des tests précédents en changeant la valeur de θ et la valeur de σ ?
10. Faire le même type de tests avec le Ridge.

Exercice 2 (Classification et Leucémie). Cet exercice est tiré d'une école d'été bioinformatique qui se déroule sur Angers/Nantes tous les ans.

1. Téléchargez les jeux de données `leukemia_big.csv` et `leukemia_small.csv`. Chargez la base `leukemia_small.csv` sur R.
2. Les colonnes indiquent les patients ALL (acute lymphoplastic) ou AML (acute myeloid). La matrice regroupe les *log-expressions* de $p = 3571$ gènes (resp. $p = 7128$) gènes par individu pour $n = 72$ individus.
3. Créez un vecteur réponse Y constitué de 0 si AML et 1 si ALL. Transposez la matrice de log-expressions pour obtenir une matrice X de taille $n \times p$.
4. Stat. descriptive.
 - (a) Tracez un histogramme de la matrice X . Que constatez-vous ?
 - (b) Refaites-le pour un gène pris au hasard.
 - (c) Calculez le vecteur des moyennes par colonne de X : `apply(X, 2, mean)`. Refaites-le même test par classe.
 - (d) Normalité : prendre une colonne au hasard et tracer la droite de Henry associée (`qqnorm(X[, j]), qqline(X[, j])`). Conclusions ? Testez à nouveau en conditionnant aux classes.
5. ACP (... pour voir !) :
 - (a)

```
PCALeuk = prcomp(X, center = TRUE, scale = TRUE)
summary(PCALeuk)
plot(PCALeuk$x[, 1], PCALeuk$x[, 2], pch = 19, xlab = "Pr Comp 1", ylab = "Pr Comp 2", col = 2+Y)
text(PCALeuk$x[, 1], PCALeuk$x[, 2], labels=rownames(PCALeuk$x), cex=0.7,
pos = 3, col = 2+Y)
```

Conclusions ?
 - (b) Testez à nouveau avec les première et troisième directions de l'ACP puis avec les deuxième et troisième directions de l'ACP.
 - (c) Si vous souhaitez créer une règle de décision à partir de l'ACP ci-dessus, comment procéderiez-vous (voir suite du cours sur la classification linéaire et non linéaire) ?
 - (d) Comment testeriez-vous sa pertinence ?

Du point de vue biologique, l'ACP n'est pas complètement satisfaisante car elle fait potentiellement intervenir toutes les composantes sur une seule direction (Par ailleurs, l'ACP n'est en général pas robuste à la grande dimension...). Une alternative est d'utiliser une ACP sparse. Néanmoins, ici, on regarde un problème supervisé donc il est plus naturel de faire appel à des méthodes supervisées. On peut penser à la LDA ou la SVM (cf suite du cours). Ici, on va tester la régression logistique pénalisée.

Rappel : La régression logistique fait partie des généralisations du modèles linéaire où “pour faire simple”, on fait l’hypothèse que $\mathcal{L}(Y|X)$ appartient à une famille paramétrique de lois ayant une forme “ressemblant” à celle du modèle linéaire. Dans le cas de la

régression logistique on fait l'hypothèse que $\mathcal{L}(Y|X = x)$ suit une loi de Bernoulli de paramètre

$$\pi(x, \theta) = \frac{e^{\langle \theta, x \rangle}}{1 + e^{\langle \theta, x \rangle}}.$$

N.B. Comme d'habitude, on fait ici abstraction du terme constant pour simplifier (quitte à supposer que la première coordonnée de x est 1). Le vrai modèle fait intervenir $\theta_0 + \langle \theta, x \rangle$.

Comme dans le cas du modèle linéaire classique, on choisit alors θ par maximisation de la log-vraisemblance du modèle (qui dans le cas du modèle linéaire classique, s'apparente à la minimisation des moindres carrés). Ici,

$$L(x_1, Y_1, \dots, x_n, \theta) = \prod_{i=1}^n \pi(x_i, \theta)^{Y_i} (1 - \pi(x_i, \theta))^{1-Y_i}$$

et

$$\mathcal{L}(\mathbf{x}, \mathbf{Y}, \theta) = \log L(x_1, Y_1, \dots, x_n, \theta) = \sum_{i=1}^n Y_i \log(\pi(x_i, \theta)) + (1 - Y_i) \log(1 - \pi(x_i, \theta)).$$

On maximise ensuite cette fonction de θ par une méthode d'optimisation numérique (déterministe ou stochastique). Comme dans le cas du modèle linéaire, on peut/doit envisager de pénaliser ce problème lorsque l'on est en grande dimension. En général, le problème prend la forme suivante : on cherche

$$\hat{\theta}_\lambda := \operatorname{Argmin}_\theta \left\{ -\frac{1}{n} \mathcal{L}(\mathbf{x}, \mathbf{Y}, \theta) + \lambda \|\theta\|_r \right\}$$

avec $r = 1$ ou 2 (LASSO ou Ridge) ou encore

$$\operatorname{Argmin}_{\{\theta, \|\theta\|_r \leq s\}} \{-\mathcal{L}(\mathbf{x}, \mathbf{Y}, \theta)\}.$$

Là encore, on peut envisager des extensions en modifiant la fonction de pénalité (type Elastic Net). On admet ici que ce type de méthode peut être mis en place en pratique dans ce cadre, en particulier, que des méthodes numériques efficaces permettent de calculer ces optimiseurs. On souhaite ici mettre la méthode à exécution sur R.

(a) Testez le modèle non pénalisé :

```
classic_glm=glm(Y ~ X-1, family = binomial(link = "logit"))
```

X-1 signifie “sans terme constant”. Qu'en pensez-vous ?

(b) Partagez la base de données en une base d'entraînement de taille $\frac{2n}{3} = 48$ et une base de validation de taille $\frac{n}{3} = 24$ pour tester la qualité du modèle. Pour rappel, on génère ces deux-sous échantillons de manière aléatoire.

```
IndTrain=sample(1:n)[1:(2*n/3)]; IndVal=setdiff(1:n,IndTrain).
```

(c) A l'aide de la fonction `glmnet` (option “binomial”), créez un objet `TrainLasso`, résultat de la régression logistique avec pénalisation L_1 (LASSO).

- (d) Même question avec Ridge et Elastic Net (pour alpha=0.25, 0.5, 0.75).
- (e) Faites un `summary(TrainLasso)`. Tentez d'analyser les résultats obtenus. Vous pourrez vous aider du site de T. Hastie dédié à `glmnet`.

`(https://web.stanford.edu/~hastie/glmnet/)`

- (f) Affichez le graphe des coefficients (“chemin de régularisation”).
- (g) Etudiez l'évolution de l'erreur de validation croisée en fonction de λ :

```
LambdaLasso = cv.glmnet(TrainSetX, TrainSetY, family="binomial",
                         type.measure="class", alpha=1).
```

- (h) Affichez $\hat{\theta}_\lambda$ pour la meilleure valeur de λ estimée par validation croisée. Relancez le programme. Le résultat est-il similaire ?
- (i) Prédire l'échantillon test avec le meilleur choix de λ . Quel résultat obtenez-vous ?
- (j) Comparez avec Ridge/Elastic Net.
- (k) Considérez la base de données `leukemia_big` et procédez à des tests similaires. Comparez vos résultats.

Exercice 3. Pour compléter, testez le modèle de régression multinomiale sur les données `handdigits` accessibles sur Moodle (Cela signifie de refaire toute la procédure).

Statistique en Grande Dimension et Apprentissage TP “Machines à Vecteurs Supports et Méthodes à noyaux”

Commencez par installer le package `e1071` sur R (peut-être est-il déjà installé) puis tapez `library(e1071)`. Dans ce package, on trouvera un certain nombre de fonctions très utiles pour l'utilisation des SVM sur R : `svm`, `plot.svm`, `tune.svm`, `predict.svm`....

Exercice 1 (Exemples simulés). **Exemple 1** Linéairement Séparable.

1. On fabrique à la main deux jeux de données dans \mathbb{R}^2 séparées par la droite $y = x$. Par exemple, on peut simuler aléatoirement x selon une loi uniforme sur $[-3, 3]$ puis simuler b selon une loi uniforme sur $[-3, 3]$ et classer +1 les points $(x_i, x_i + b_i)$ qui sont au-dessus de la droite $y = x$ et classer -1 ceux qui sont en dessous de la droite. En d'autres termes, on a ici clairement un jeu de données parfaitement linéairement séparables. Fabriquez ainsi un `data.frame` ayant pour colonnes 1 et 2 abscisse et ordonnée des points et pour 3eme colonne la classe de chacun des points (Veillez à ce que cette dernière colonne soit bien vue comme un facteur).

2. Testez la fonction `svm` sur cet objet avec le noyau linéaire

```
svm(classe~., data=nom_data.frame, kernel="linear")
```

3. A l'aide de la fonction `plot` (voir `plot.svm` dans l'aide), affichez le résultat.
4. Affichez les vecteurs supports et leurs indices, les coefficients associés et β_0^* , l'“intercept” (allez voir les sorties de `svm` dans l'aide). Quelle est l'équation de l'hyperplan séparateur ?
5. Quel rôle joue le paramètre `cost`? Comment faire pour retrouver un SVM à marge non flexible? Faites varier les valeurs de `cost` et regardez l'effet sur les vecteurs supports.
6. Utilisez la fonction `predict` pour prédire la classe de votre échantillon d'apprentissage. Affichez la matrice de confusion à l'aide de la fonction `table`. En déduire l'erreur d'entraînement.
7. Simulez un échantillon de 100 variables aléatoires uniformes sur $[-3, 3]^2$ puis calculez l'erreur test.
8. Que se passe-t-il si l'on enlève l'option sur le noyau? Quel est le noyau par défaut? Testez la méthode avec ce nouveau noyau.

Exemple 2. Non linéairement séparable.

1. Simulez un échantillon de $n = 100$ variables gaussiennes de loi $\mathcal{N}(0, I_p)$ où p est une dimension que l'on fera varier.
2. Attribuez la valeur 1 aux x_i dont la norme est plus petite que \sqrt{p} et -1 aux autres.
3. Dans les questions qui suivent, on suppose que $p = 2$.
 - (a) Testez la fonction `svm` et la fonction `plot` avec les noyaux polynomiaux et gaussiens (à votre avis, quel serait le bon choix de degré pour le polynôme?)
 - (b) On souhaite ici calibrer les paramètres. Utilisez la fonction `tune.svm` dans le cas du noyau gaussien en faisant varier γ et `cost`. Quel est le meilleur choix? Calculez l'erreur test sur ce choix de paramètres en simulant un nouvel échantillon indépendant de 100 variables.
 - (c) Faire la même chose avec le noyau polynomial en testant les degrés 2 et 3. Erreur Test optimale? *N.B. Un principe usuel pour améliorer les résultats consiste à agréger les modèles, i.e. à faire une combinaison linéaire de plusieurs modèles. Ce principe pourrait être employé ici (voir Adaboost).*

- Refaire rapidement avec $p = 10$ les deux premières étapes. Comment fait-on pour afficher la solution dans un plan (voir la fonction `plot.svm` et le paramètre `slice`) ?

Exercice 2 (Un exemple multi-classes). Le jeu de données Iris (dit de Fisher), disponible sur R est un cadre usuel pour tester les algorithmes de classification supervisée. Testez la fonction `svm` sur cet exemple (Attention, dans l'affichage, le fait que le nombre de variables soit de dimension 4 implique que l'on doit choisir le plan dans lequel on représente le résultat et la "tranche" que l'on utilise pour les deux autres variables (en pratique, on choisit la valeur moyenne)).

Exercice 3 (SVM et régression). Comme mentionné dans la partie cours, les SVM peuvent aussi permettre d'aborder des problèmes de régression. On propose dans cet exercice de tester cette méthode sur le jeu de données `ozone` accessible sur Moodle. L'objectif, sur ces données, est d'améliorer la prévision calculée par les services de MétéoFrance (MOCAGE) de la concentration d'ozone dans certaines stations de prélèvement à partir de cette prévision et en s'aidant d'autre variables également prévues par MétéoFrance. Il s'agit d'un problème dit d'adaptation statistique d'une prévision déterministe pour une amélioration locale de modèles à trop grande échelle. Plus précisément, deux variables peuvent être prédites : soit la concentration quantitative d'ozone, soit le dépassement (qualitatif) d'un certain seuil fixé à $150 \mu\text{g}$. On s'intéressera en priorité au premier problème.

Exercice 4 (Un second exemple multi-classes). On revient ici sur la base `MNIST` (digits numbers). A nouveau, on a un modèle multi-classes (qui sert également de base de données usuelle pour les tests d'algorithmes). Mettre en oeuvre la méthode `svm`. Comment fonctionne l'algorithme ? Peut-on ajuster les paramètres ? *N.B. Attention, la base MNIST contient un nombre de points importants. Le temps de calcul risque d'exploser avec n. Testez dans un premier temps des petites valeurs de n (en tirant au hasard un échantillon de taille n).*

Exercice 5. Comme mentionné en cours, les SVM sont applicables et appliqués à de nombreux jeux de données. Voici en complément quelques références :

- Reconnaissance faciale par SVM (et ACP) : <https://github.com/gouravaich/faces-recognition-pca>
- 20NewsGroups* et SVM : *20NewsGroups* fait partie des bases "textuelles" célèbres (collection d'environ 20000 articles de journaux). Le but est alors de les classer de manière automatique par thème (voir par exemple <https://github.com/sukilau/20-newsgroup-classification/blob/master/NB-SVM.py>
pour un développement sur le sujet.

Exercice 6 (Autres méthodes à noyau). 1. Expliquer pourquoi les méthodes à noyau fonctionnent sur la régression ridge. Mettre en application sur un exemple simple (par exemple $Y = \sin(X) + \varepsilon$).

- Même question pour la régression logistique.
- Expliquer pourquoi le problème de minimisation des K -means peut s'exprimer à l'aide d'un produit scalaire mais que l'algorithme ne nécessite pas que la connaissance du noyau. Néanmoins, ce problème a été contourné et des algorithmes existent, par exemple le module `KernelKMeans` de `tslearn.clustering`. Tester cet algorithme sur un exemple. Pour les curieux, voir la vidéo de Julien Mayral https://www.youtube.com/watch?v=7o2Hny_5lk0 en complément (et les liens avec le clustering spectral).

Statistique en Grande Dimension et Apprentissage - TP "Réseaux de Neurones"

Ce TP a pour but d'illustrer le chapitre d'introduction aux réseaux de neurones et au deep learning.

Exercice 1 (Perceptron simple).

1. Charger les données spam sur Moodle.
2. Fabriquer un échantillon d'entraînement et un échantillon test.
3. Construire un modèle de type "perceptron simple" en utilisant en fonction d'activation ψ la fonction sigmoïde.
4. Que signifient les différents paramètres du modèle (layers, units,...) ? Affichez les caractéristiques de ce modèle.
5. Entraînez le modèle sur l'échantillon d'entraînement (comprendre les paramètres epochs, batch_size, optimizer).
6. Calculez l'erreur test.

Exercice 2 (Perceptron Multicouches).

1. Charger les données Mnist_test sur Moodle (uniquement la base test pour limiter le temps de calcul).
2. Rediviser cet échantillon en un échantillon d'entraînement et un échantillon test.
3. Construire un perceptron à une couche en utilisant en fonction d'activation ψ la fonction "softmax". Paramétrier le "dropout" (paramètre d'élagage).
4. Que signifient les différents paramètres du modèle (layers, units,...) ? Affichez les caractéristiques de ce modèle.
5. Entraînez le modèle sur l'échantillon d'entraînement (comprendre les paramètres epochs, batch_size).
6. Calculez l'erreur test.
7. Construire un modèle à deux couches cachées et refaites la même procédure.
8. Testez l'erreur de prédiction sur les données spam avec un réseau à une couche cachée en réglant les hyperparamètres.
9. Comment pouvez-vous sauvegarder votre modèle ? (voir sur ce sujet la page https://keras.rstudio.com/articles/tutorial_save_and_restore.html)

Exercice 3 (Perceptron Multicouches pour la régression). Le but de cet exercice est de prédire le prix moyen de l'immobilier à Boston via un perceptron multicouches. On utilise une base de données issue de keras :

```
boston_housing= dataset_boston_housing()
```

1. Construire un modèle à 2 couches cachées avec 64 neurones par couche cachée et la fonction d'activation `relu`.
2. Entraîner et tester le modèle.

Exercice 4 (Réseau de neurones convolutif/Réseaux de neurones récurrents).

1. Utiliser le tutoriel https://keras.rstudio.com/articles/examples/mnist_cnn.html afin de tester les modèles de réseaux de neurones convolutifs sur la base MNIST.

2. Pour aller plus loin : Importation d'images/Génération aléatoire d'images : Dans l'exemple précédent, nous n'avons pas abordé le problème de l'importation d'images au format jpeg ou png et également celui de la génération aléatoire d'images. Sur ce sujet, on pourra consulter la page <https://towardsdatascience.com/r-vs-python-image-classification-with-keras-1fa99a8fef9b> pour un exemple de reconnaissance de fruits.
3. Pour aller plus loin : Utiliser le tutoriel https://keras.rstudio.com/articles/examples/lstm_text_generation.html afin de tester l'architecture LSTM des réseaux de neurones récurrents pour générer du texte "Nietzschien".