Machine Learning

Master 2 Informatique - BI&A

Guillaume Metzler

Université Lumière Lyon 2
Laboratoire ERIC, UR 3083, Lyon

guillaume.metzler@univ-lyon2.fr

Automne 2022

Avant Propos I

Le cours se décompose de la façon suivante :

- **Deux séances CM de 3h** sur le Machine Learning en général : Introduction à la problématique général, processus d'apprentissage et présentation des algorithmes classiques en apprentissage machine.
- **5 séances de TD de 3h** pour mettre en oeuvre, tester les différents algorithmes et effectuer des expériences de façon optimale.

Avant Propos II

Concernant les évaluations

Vous aurez deux évaluations :

- Un examen : il portera essentiellement sur le cours avec des questions plus ou moins proches du cours mais aussi une partie pratique.
- Un projet : l'idée sera de vous proposer de travailler sur un jeu de données réelles et de vous pratiquer les différents algorithmes, de les analyser et de faire un rapport sur les approches testées.

L'examen comptera pour environ 30-40% de la note finale et 60-70% pour le projet.

Déroulement des séances de TD

Vous aurez donc 5 séances de TD et voilà le contenu que je souhaite aborder avec vous au cours de ces séances :

- TD1 : mise en place et application d'un protocole en Machine Learning
- TD2 : travail sur des tâches de régression et de classification
- TD3 : classification binaire dans un contexte déséquilibré
- TD4 : boosting, adaboost et gradient boosting
- TD5 : cette dernière séance sera déidée à votre examen.

Qu'est-ce que le Machine Learning ?

Introduction

Qu'est-ce que le Machine Learning ? I

L'apprentissage Machine est une sous-branche de l'*Intelligence artificielle*. Elle se trouve à la frontière de l'Informatique et des Mathématiques Appliquées (comme les statistiques ou l'optimisation). Cette discipline recoupe également la *Science des Données* car elle nécessite préalablement de récolter/nettoyer et d'analyser des données afin d'en extraire les informations importantes pour l'application souhaitée.

Qu'est-ce que le Machine Learning ? II

Cette définition met en évidence des éléments importants en Machine Learning pour la résolution d'un problème

- La tâche : quel est le problème que je souhaite résoudre à l'aide d'un outil informatique ? Il est important de bien identifier et définir le problème.
- Les données : quelles sont les données dont je dispose pour m'aider à développer une intelligence capable d'effectuer cette tâche.
- L'algorithme : quel type de problème est-ce que je cherche à résoudre ? Mettre le problème sous forme *mathématiques*, développer l'algorithme pour le résoudre et l'implémenter.

Qu'est-ce que le Machine Learning ? III

Au fait, quelles sont pour vous les deux grandes phases en Apprentissage Machine ?

Qu'est-ce que le Machine Learning ? IV

Apprentissage/Entraînement

Cette première va consister à déterminer les paramètres d'un modèle préalablement défini et qui a pour objectif de répondre à une tache fixée.

Elle utilise un ensemble d'apprentissage, donc un ensemble limité de données.

Les tâches sont variées, comme la classification document, l'identification de chat ou de chien dans les images ou la prévision et l'études des cours d'un produit sur le marché boursier.

Cette phase est qualifiée de préparatoire dans le sens où le modèle est appris avant sa mise en production.

Qu'est-ce que le Machine Learning ? V

Test

On peut aussi retrouver cette phase là sous ne nom de phase de *mise en production*.

Le système a été préalablement construit, validé et testé et il peut maintenant être déployé à grande échelle à des fins pratiques !

Lors de cette phase là, le modèle est constamment surveillé afin que son efficacité reste conforme aux observations effectuées lors de son apprentissage. Parfois, des mises à jour sont nécessaires (ré-entrainement du modèle sur des données plus récentes) afin de maintenir son efficacité (on peut parler d'apprentissage *online*).

Données : Chat-Chien I

Regardons de plus prêt ce que sont nos données

Données : Chat-Chien II

Images : ce sont des matrices (une matrice dont chaque entrée correspond à la valeur d'un pixel) de nombres où chaque matrice représente une intensité de couleur pour un pixel donné –¿ superposition de trois matrices pour trois canaux de couleurs

$$R = \begin{bmatrix} 1 \\ 1 \\ 0.9 \\ 0 \\ .. \end{bmatrix}, \qquad G = \begin{bmatrix} 1 \\ 1 \\ 0.1 \\ 0.3 \\ .. \end{bmatrix}, \qquad B = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0.7 \\ .. \end{bmatrix}.$$

Données : Chat-Chien III

Elles sont au cœur des algorithmes d'apprentissage et peuvent donc être de différentes natures :

- brutes
- transformées - créées
- apprises

Nous verrons cependant un problème majeur lié à ces données et qui est propre à l'apprentissage : ces données sont issues d'une distribution inconnue ce qui ouvre des perspectives de recherche en Machine Learning !

Regardons maintenant leur usage d'un point de vue pratique

Données : Chat-Chien IV

Les descripteurs de nos images ne sont pas invariants et peuvent dépendre de plusieurs paramètres ou des conditions de récolte des données.

Exemples : formes ou motifs - diamètres - couleurs

- le diamètre varie avec la notion de distance
- les formes sont complexes à représenter (point de vue)
- la couleur varie avec l'exposition à la lumière

Ainsi, une même image, prise dans des conditions différentes peut avoir des caractéristiques différentes. (Il existe cependant des algorithmes qui permettant d'identifier des éléments identiques dans images "différentes", comme *SIFT*).

Données : Chat-Chien V

On se contente rarement, et c'est d'autant plus vrai dans un contexte industriel, de travailler sur les données brutes. On va souvent chercher à les transformer pour **créer de nouvelles variables** ou transformer **les variables de bases** :

- A l'aide de la connaissance métier et d'experts qui connaissent les informations importantes ou les facteurs discriminants.
- A l'aide de transformation mathématiques ou de modèles en apprentissage : réseaux de neurones, auto-encodeurs, apprentissage de métrique, noyaux, ...

Données : Chat-Chien VI

Elles sont au cœur des algorithmes d'apprentissage et peuvent donc être de différentes natures :

- brutes
- transformées - créées
- apprises

Nous verrons cependant un problème majeur lié à ces données et qui est propre à l'apprentissage : ces données sont issues d'une distribution inconnue, ce qui complexifie l'apprentissage.

Données : Chat-Chien VII

Regardons maintenant ces données sur le plan pratique, *i.e.* comment les utiliser. Si on adopte une présentation algorithmique, nous avons :

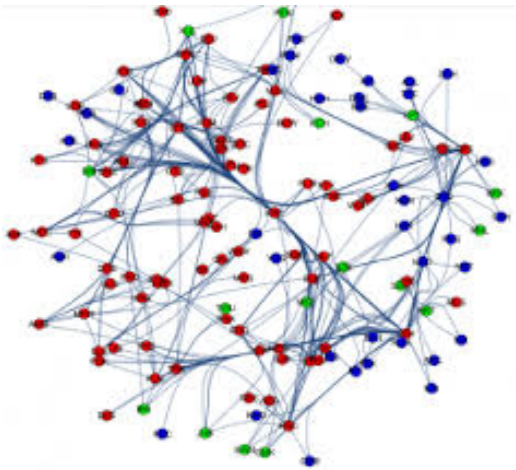
- **Input :** nos images (sous forme vectorielle)
- **Output :** classe de l'image (chien ou chat)
- **Ensemble d'apprentissage :** notre ensemble d'images + label

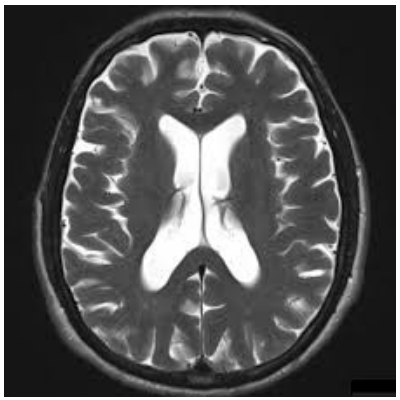
Données : Chat-Chien VIII

Une fois l'entraînement terminé, il est bon de savoir ce qui se passe si notre modèle peut être mis en production. On va donc reproduire les conditions d'une mise en production.

On cherche maintenant à savoir si notre modèle est performant sur de nouvelles données ? Est-ce qu'il est capable d'identifier le genre des images non rencontrées pendant sa phase d'entraînement ?

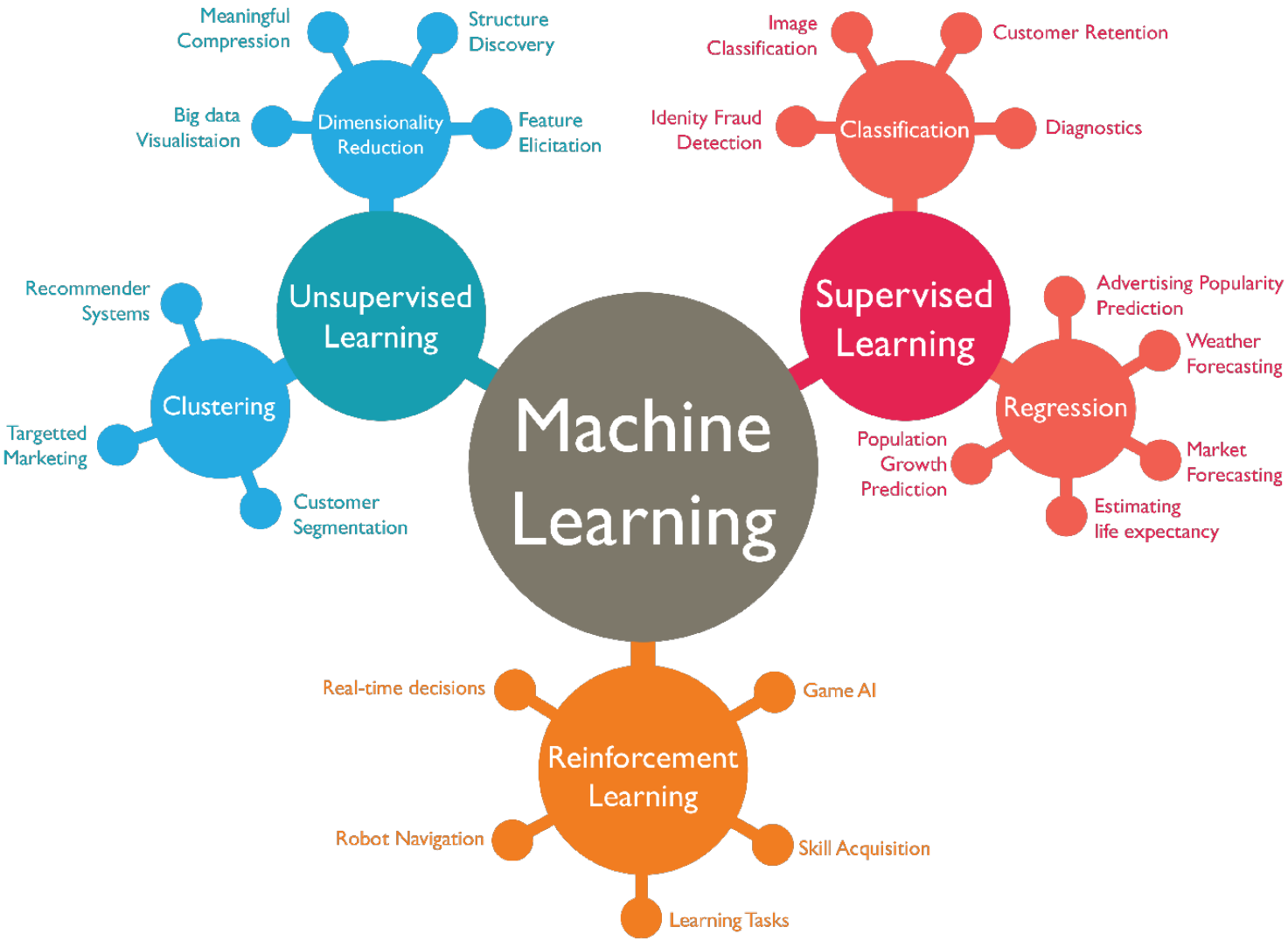
Des données sous toutes leurs formes





104	11,8	0,1	0,13	15,78	0,6	
126	10,3	0,3	0,00	16,31	0,0	
166	11,8	1,1	-0,06	10,56	0,4	
105	13,2	1,9	-0,03	11,89	1,8	
15	16,9	0,9	0,00	12,81	1,2	
6	18,7	0,4	0,12	10,92	0	

Différents contextes en Machine Learning I



Différents contextes en Machine Learning II

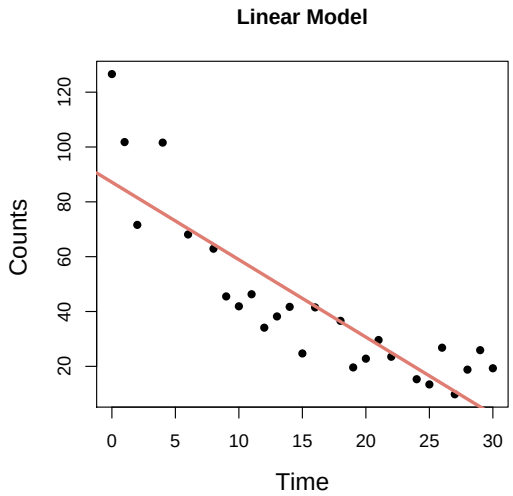
Nous pouvons décomposer le *Machine Learning* en trois grandes catégories principales

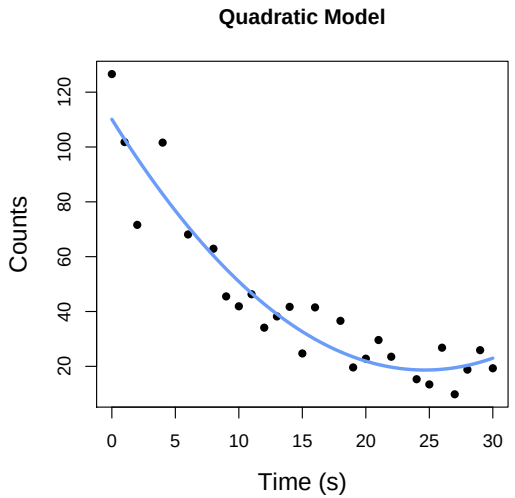
- **Apprentissage supervisé :** on y inclut une grande partie des algorithmes de classification (SVM - k-NN, ...) mais aussi d'apprentissage de similarité (Metric Learning, Transfert Learning) ou encore des tâches de régression
- **Apprentissage non supervisé :** on y retrouve les algorithmes de clustering (K-means ou HCA) mais aussi de l'apprentissage par transfert non supervisé
- **Apprentissage par renforcement :** ou comment apprendre de ses expériences

Différents contextes en Machine Learning III

Apprentissage supervisé : **Régression**

Le système a accès un ensemble de données décrites à la fois par des descripteurs mais également par une étiquette fournie par un "*oracle*". L'objectif sera d'apprendre une "règle" à partir des descripteurs et permettant de prédire la valeur de la donnée mais aussi celle des nouvelles données.

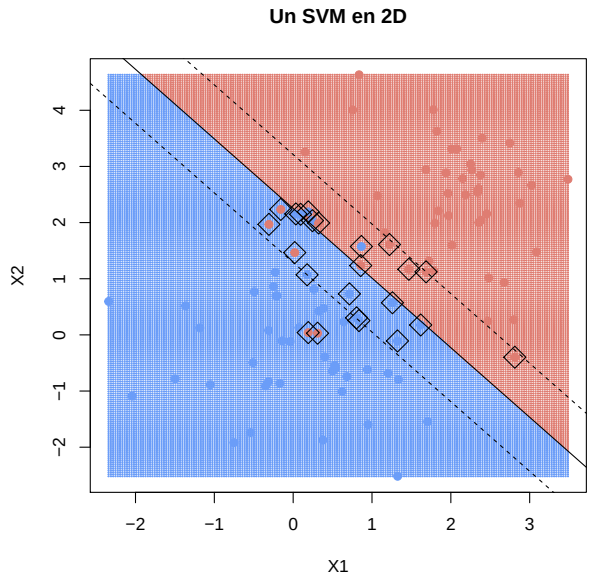


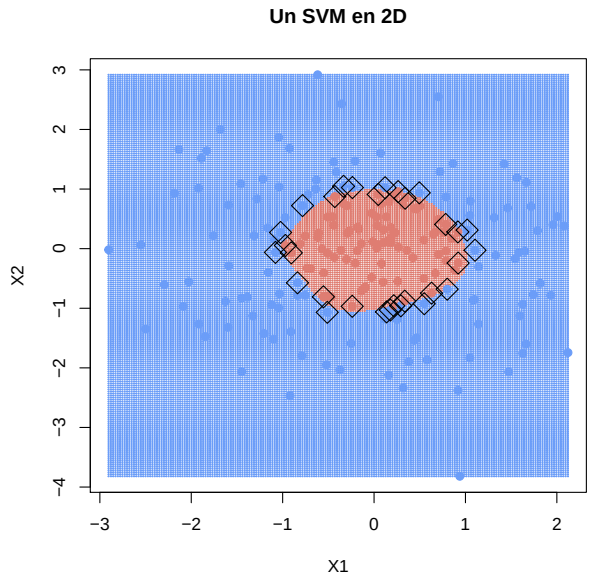


Différents contextes en Machine Learning IV

Apprentissage supervisé : **Classification**

Le système a accès un ensemble de données décrites à la fois par des descripteurs mais également par une étiquette fournie par un "*oracle*". L'objectif sera d'apprendre une "règle" à partir des descripteurs et permettant de prédire l'étiquette de la donnée mais aussi celle de nouvelles données..

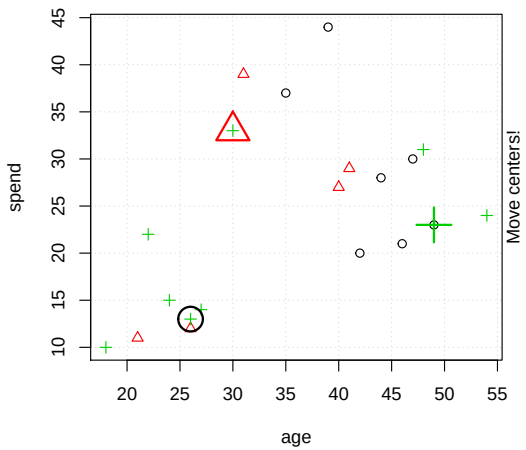


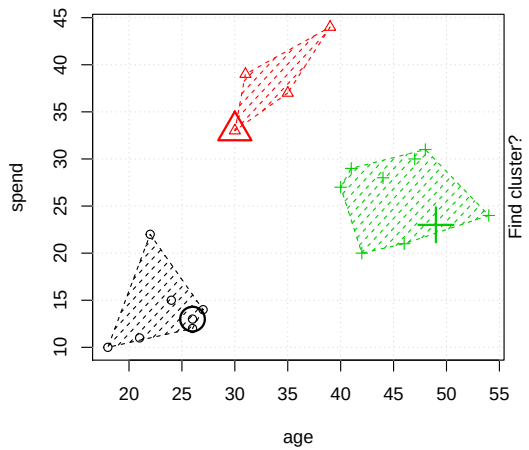


Différents contextes en Machine Learning V

Apprentissage non-supervisé : **Clustering**

On ne dispose pas d'étiquette lors de la phase d'apprentissage. L'idée principale est de regrouper les exemples présentants des **similarités** ou des **ressemblances** afin de créer des groupes (**clustering**). On peut également utiliser cela pour faire de l'**estimation de densité** ou apprendre une **nouvelle représentation des données**.

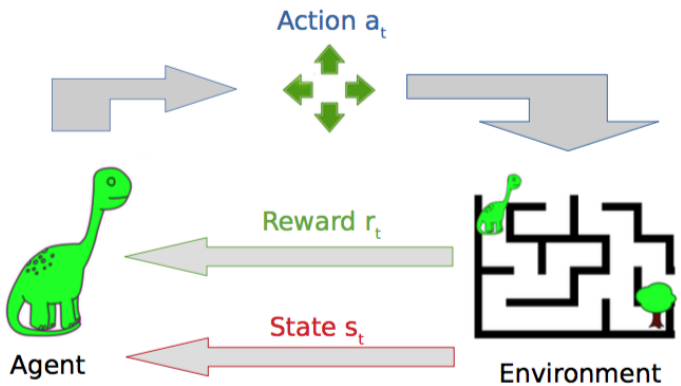




Différents contextes en Machine Learning VI

Apprentissage par renforcement

Le modèle doit apprendre à effectuer les bonnes **actions** en fonction du **contexte**, *i.e.* il doit prendre les bonnes **décisions** en fonction des **observations effectuées**. Cela se fait à l'aide d'un système de **récompenses**. Dans ce type d'apprentissage, on ne dispose de données lui indiquant les actions optimales à entreprendre selon la situation. On le laisse découvrir par lui même ce qu'il doit faire avec une méthode d'**essai-erreur**.



Données et représentation I

Les exemples précédents montrent bien que les données sont au cœur des algorithmes d'apprentissage *Machine Learning*. Elles peuvent se trouver sous différentes formes et formats, elles peuvent être structurées ou non, parfois contenir des anomalies ou des valeurs manquantes ...

En *Machine Learning*, il est d'usage de voir ces données comme un ensemble de m exemples ayant la même nature. La représentation la plus naturelle qui est choisie est sous forme d'un vecteur $\mathbf{x}$ en dimension d : $\mathbf{x} = (x_1, x_2, ..., x_d)$ où $\mathbf{x} \in \mathbb{R}^d$

Données et représentation II

La représentation choisie, *i.e.* ce que représente la vecteur, va dépendre de la nature des données.

Les images peuvent se représenter par des vecteurs (après transformation, CNN et "vectorisation")
données génétiques pour une séquence de gènes (dimension = longueur du gène, vecteur = encodage)
sons comme des suites de signaux (dimension = fréquence, vecteur des amplitudes)
documents textuels comme ensemble de mots (dimension = nombre de mots dans le corpus, vecteurs des occurrences)
métadonnées : auteurs, dates, ...

Les données réelles sont parfois très complexes, avec une grande redondance et présentent une grande variabilité.

Normalisation des données I

Outre l'aspect représentation, il faut aussi s'intéresser aux valeurs prises par chaque descripteur de nos données afin de ne pas accorder une importance fortuite à un sous-ensemble. Cela pourrait se révéler particulièrement nocif pour certains algorithmes basés sur la notion de distance !

Exemple : Soit $\mathbf{x} = (x_1, x_2), \mathbf{x}' = (x'_1, x'_2)$ où $x_1, x'_1 \in [0, 1]$ et $x_2, x'_2 \in [500, 1000]$, alors la distance Euclidienne

$$d(\mathbf{x}, \mathbf{x}') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2}$$

est semblable à la distance entre x_2 et x'_2.

Une solution consistance alors à normaliser les données, ce qui, dans la majorité des cas, permet d'augmenter les performances des algorithmes en accordant le même poids à chaque variable.

Normalisation des données II

Il existe plusieurs processus de normalisation des données dont les plus connus/utilisés sont les suivants :

- **mise à l'échelle** ou **normialisation min-max** pour que les valeurs se trouvent dans $[0, 1]$ ou $[-1, 1]$

$$x = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \text{ou} \quad x = 1 - 2 \times \frac{x - \min(x)}{\max(x) - \min(x)},$$

- **standardisation** : faire en sorte que chaque feature suivent une loi normale centrée réduite

$$x = \frac{x - \mu(x)}{\sigma(x)},$$

Normalisation des données III

- **normalisation** : diviser chaque vecteur par sa norme de telle sorte que $\|\mathbf{x}\| = 1$

$$\mathbf{x} = \frac{\mathbf{x}}{\|\mathbf{x}\|}.$$

Une hypothèse importante

Comme évoqué plus tôt, nous n'avons jamais accès à l'ensemble de la distribution lors de l'apprentissage d'un modèle, mais seulement à un **petit échantillon**. Pour s'assurer que le modèle appris sur ces données est capable de **généraliser sur de nouvelles données**, nous devons supposer que ces dernières sont *i.i.d.*

Définition 1.1: Distribution des données

Un ensemble d'apprentissage S est dit *i.i.d.* si tous les exemples sont issus d'une même distribution de probabilité **inconnue** $\mathcal{D}$ et qu'ils sont mutuellement indépendants.

$\rightarrow$ une hypothèse fondamentale pour la théorie en Machine Learning

Données et dimension

Si on dispose de suffisamment d'exemples dans un espace de dimension "limitée", on montre que nos modèles sont capables de généraliser, car les données permettent d'approximer correctement la distribution.

Quid dans le cas contraire ? Malédiction de la dimension !

Cela se traduit par l'apparition de phénomènes qui n'apparaissent pas en dimension raisonnable : comme des volumes qui deviennent anormalement faibles, et qui peuvent avoir des conséquences sur certains algorithmes.

$\rightarrow$Réduire la dimension de l'espace des données !

Apprentissage

Apprentissage I

Considérons $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ la distribution inconnue de nos données. L'espace $\mathcal{X}$ est appelé **input space** ou encore **feature space**, en général $\mathcal{X} \subset \mathbb{R}^d$. L'espace $\mathcal{Y}$ est **l'espace des étiquettes** ou encore **l'output space** :

- $\mathcal{Y} = \emptyset$: apprentissage non supervisé
- $\mathcal{Y} = \mathbb{R}$: régression
- $\mathcal{Y} = \{0, 1\}$: classification binaire
- $\mathcal{Y} = \{1, \ldots, C\}$: classification multi-classe
- $\mathcal{Y} = \{0, 1\}^C$: classification multi-label

Objectif : trouver un algorithme $\mathcal{A}$, générant une hypothèse h capable d'effectuer la tâche souhaitée.
Problème : en pratique $\mathcal{D}$ est inconnue.

Apprentissage II

On dispose uniquement d'un échantillon de taille finie $S = \{\mathbf{x}_i, y_i\}_{i=1}^m \underset{i.i.d.}{\sim} \mathcal{D} = \mathcal{X} \times \mathcal{Y}$ où $\mathcal{X} \subset \mathbb{R}^d$ et $\mathcal{Y} \subset \mathbb{Z}$ (classification) ou $\subset \mathbb{R}$ (régression).
Dans la suite, on supposera que nos données $\mathbf{x}_i \in \mathbb{R}^d$ et on notera

$$X = (\mathbf{x}_1, ..., \mathbf{x}_m) = \begin{pmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1d} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{m1} & \cdots & \mathbf{x}_{md} \end{pmatrix}.$$

On souhaite donc trouver une **hypothèse** h, telle que $h(\mathbf{x}) = y$ qui soit performante sur notre échantillon S mais aussi sur tout nouvel exemple $(\mathbf{x}, y)$.

Comment apprendre une telle hypothèse ?

Apprentissage III

Reprenons notre exemple de classification d'image de chats et de chiens.

On cherche à minimiser les erreurs de l'algorithme, ou plus généralement on minimise ce que l'on appelle un **risque**.

Apprentissage IV

Afin d'apprendre et de trouver la meilleur hypothèse $h^\star$, on a besoin de définir un critère qui permet de quantifier la qualité de l'hypothèse apprise. Ce critère va prendre la forme de **Mesure de performance** (que l'on va alors chercher à maximiser) ou plus traditionnellement de **Risque** (que l'on va chercher à minimiser).
Idéalement, on va chercher à minimiser la risque sur l'**ensemble de la distribution des données**, ce que l'on appelle le **Risque réel**.

Le risque réel $\mathcal{R}(h)$, encore appelé **risque en généralisation** d'une hypothèse h correspond à l'erreur moyenne (donc l'espérance) de l'hypothèse h sur toute la distribution

$$\mathcal{R}(h) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\mathbb{1}_{\{h(\mathbf{x})\neq y\}}\right].$$

Apprentissage V

En apprentissage supervisé, l'objectif est donc d'apprendre une hypothèse h qui va minimiser le risque réel, *i.e.* le risque sur notre distribution toute entière. Malheureusement, ce risque réel $\mathcal{R}(h)$ ne peut pas être calculé étant donné le caractère inconnu de $\mathcal{D}$.

Nous pouvons uniquement l'**estimer** ou le **mesurer** sur notre ensemble d'apprentissage. Cette estimation est appelée **Risque empirique**, noté $\mathcal{R}_S(h)$.

D'un point de vue pratique c'est donc cette quantité que l'on va chercher à minimiser ... enfin ... pas tout à fait comme nous le verrons après ! Regardons déjà sa définition.

Apprentissage VI

En apprentissage non-supervisé cette notion de risque se retrouve également mais elle ne fait plus référence à un terme d'erreur.

Elle est simplement employée pour faire référence à la quantité que l'on souhaite minimiser dans un problème minimisation, *e.g.* une somme de variances dans le cas du *clustering* ou encore une distance entre distributions lorsque l'on fait de l'*Adapatation de Domaine non supervisé*.

Retournons à la notion de risque en Apprentissage Supervisé

Apprentissage VII

Définition 2.1: Risque Empirique

Soit $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ un ensemble d'entraînement. Le risque empirique $\mathcal{R}_S(h)$, appelé aussi risque d'erreur, d'une hypothèse h correspond à l'erreur moyenne sur notre ensemble d'apprentissage, ou encore l'espérance de cette erreur sur S

$$\mathcal{R}_S(h) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\mathbb{1}_{\{h(\mathbf{x})\neq y\}}\right] = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{\{h(\mathbf{x}_i)\neq y\}} = \mathbb{P}\left[h(\mathbf{x}) \neq y\right].$$

Apprentissage VIII

Ce risque empirique mesure donc, dans le cas présent la probabilité que l'hypothèse h se trompe dans la prédiction effectuée pour l'exemple $\mathbf{x}$.

Mais minimiser les erreurs (de façon binaire) est un problème difficile pour les algorithmes (NP-hard). On cherche donc rarement à minimiser tel quel le taux d'erreur. En pratique, on va plutôt chercher à minimiser un risque basé sur un substitut du taux d'erreur via l'utilisation de **fonctions de coûts/pertes** (**loss functions** en anglais).

Fonctions de loss I

Cette notion de risque d'erreur est spécifique à ce qu'on appelle la *0-1 loss* (prédiction correcte 0, erreur de prédiction 1). En revanche, ce ne sont pas les seuls loss qui sont utilisées, d'ailleurs cette dernière n'est que **très rarement utilisée en pratique**.

Définition 2.2: Loss function

Une fonction de loss ℓ est une fonction $\mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ qui sert à mesurer le désaccord entre la prédiction effectuée par l'hypothèse h, $h(\mathbf{x})$ et la valeur à prédire y. $\mathcal{H}$ est appelé espace d'hypothèse.

Fonctions de loss II

Erreur en classification ou 0-1 loss

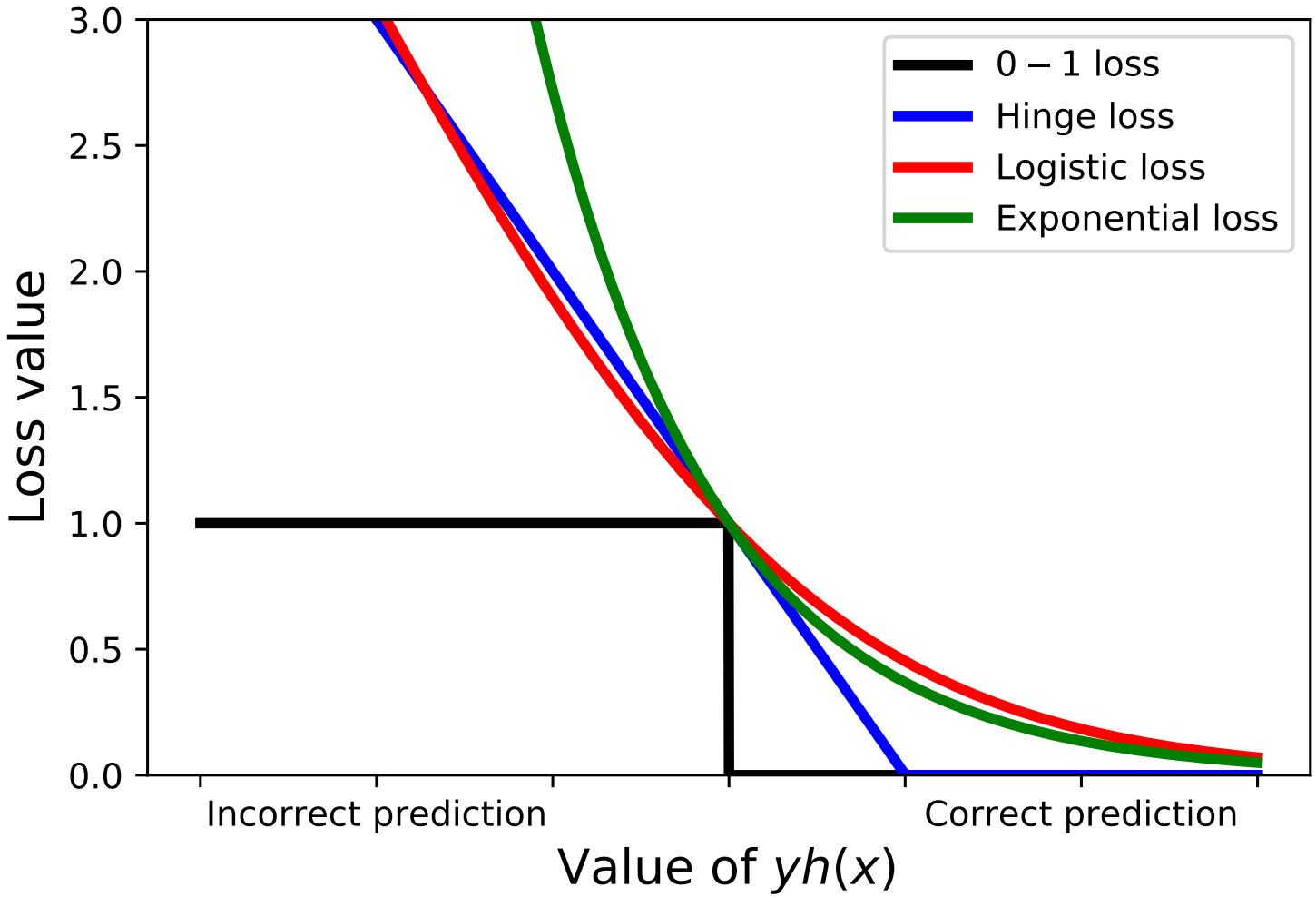
$$\ell(h(\mathbf{x}), y) = \begin{cases} 1 & \text{if } h(x) \times y < 0, \\ 0 & \text{sinon.} \end{cases}$$

Cette loss présente de nombreux inconvénients : elle n'est pas convexe, elle n'est pas dérivable, donc peu intéressante pour des **algorithmes d'optimisation**.

$\rightarrow$ utilisation de **surrogate**, des fonctions dites de substitutions, de la 0-1 loss qui présentent l'avantage d'être **convexe** et parfois même **différentiable**.

On utilisera des surrogates différentes en fonction de l'algorithme que l'on souhaite utiliser.
Ces surrogates se présentent comme des bornes supérieures de la 0-1 loss.

Fonctions de loss III



Fonctions de loss IV

- **la hinge loss :** on la rencontre plus particulièrement lors de l'apprentissage de modèles comme les *SVM* (classification ou régression)

$$\ell(h(\mathbf{x}), y) = \max(0, 1 - yh(\mathbf{x})),$$

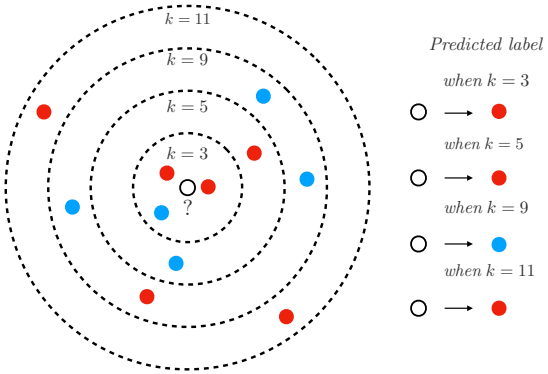
- **la loss logistique :** que l'on rencontre lors de l'utilisation d'un modèle de *régression logistique* (classification)

$$\ell(h(\mathbf{x}), y) = \frac{1}{\ln(2)} \ln\left(1 + \exp(-yh(\mathbf{x}))\right),$$

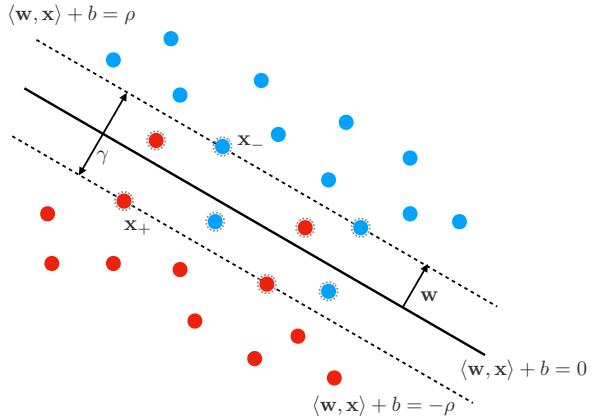
- l**la loss exponentielle :** on l'a rencontre surtout dans des contextes de *boosting*

$$\ell(h(\mathbf{x}), y) = \exp(-yh(\mathbf{x})).$$

Des algorithmes pour du supervisé



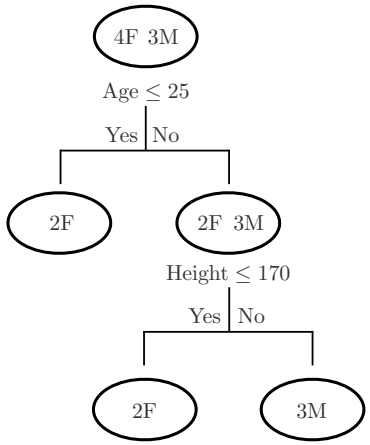
Plus proches - voisins (k-NN)



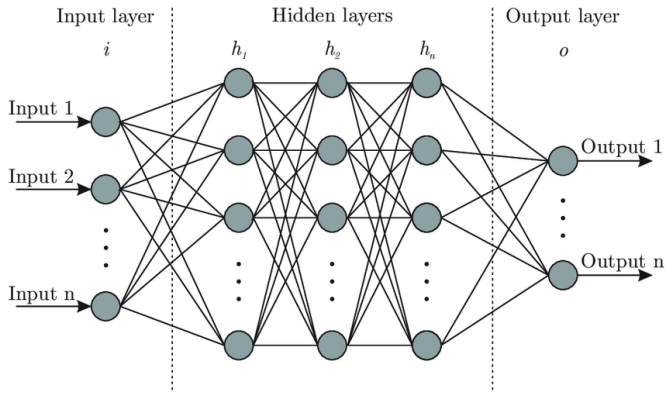
Séparateur à Vaste Marge (SVM)

Toy dataset

Age	Height	Sex
20	175	F
32	180	M
40	175	M
28	172	M
22	165	F
40	169	F
70	170	F



Arbres de décisions (DT)



Réseaux de neurones (NN)

Des algorithmes pour du non supervisé

- **Clustering** qui regroupe la classification descendante hiérarchique ou encore
- **Réduction de dimension** comme l'Analyse en Composantes Principales ou les méthodes factorielles de façon générale, on pourrait aussi citer des les auto-encodeurs
- **Estimation de densité** : modèles de mélanges de façon générale (algorithme statistique de type EM) ou encore celles fondées sur les méthodes à noyaux.

Qu'est-ce qu'un bon modèle ?

Maintenant que l'on dispose de données S, d'une loss ℓ et d'un algorithme, comment savoir si notre modèle est performant ?
Pour cela on va chercher à minimiser le risque empirique $\mathcal{R}_S(h)$

$$\mathcal{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} \ell(h(\mathbf{x}_i), y).$$

et tenter d'estimer le risque réel $\mathcal{R}(h)$ ou risque en généralisation

$$\mathcal{R}(h) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\ell(h(\mathbf{x}), y)\right].$$

Mais on ne connait pas la distribution ... donc va utiliser un processus de validation !

Cross validation I

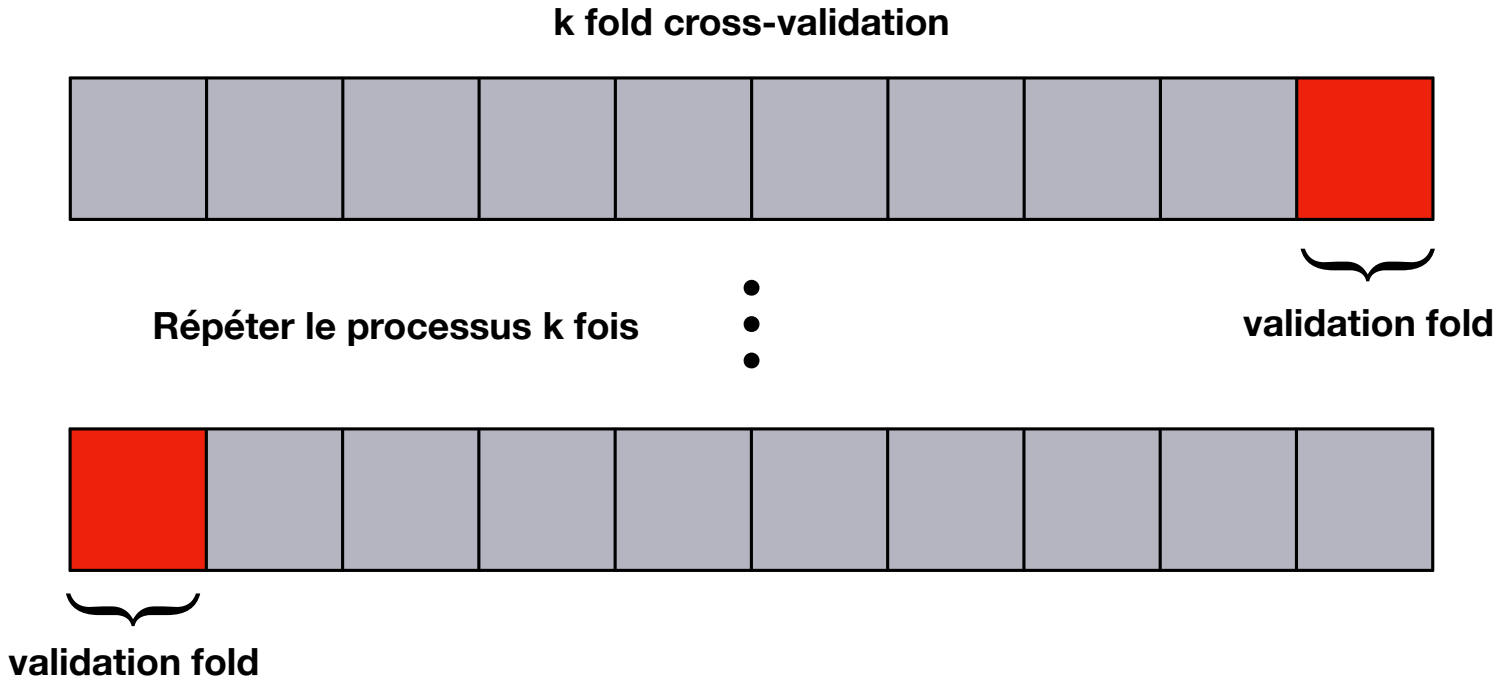
Il s'agit de conserver une partie des données de l'ensemble d'entraînement qui ne sera pas utiliser pour apprendre les paramètres de notre modèle, on les utilsera pour faire une première évaluation (entraînement - validation) mais on peut faire encore mieux.

Définition 2.3: k-fold Cross-Validation

l s'agit d'un moyen algorithmique permettant d'estimer les performances en généralisation d'un algorithme donné sur des données inconnues.
Le principe est simple, on sépare nos données en k groupes et on utilise $k-1$ groupes pour apprendre notre modèle et le dernier groupe pour l'évaluer.

Cross validation II

On effectue une série de k expériences en utilisant une validation classique, mais on change l'apprentissage et la validation à chaque run.



Au final, notre algorithme sera donc évalué sur l'ensemble des données disponibles, ce qui permet une estimation plus fiable.

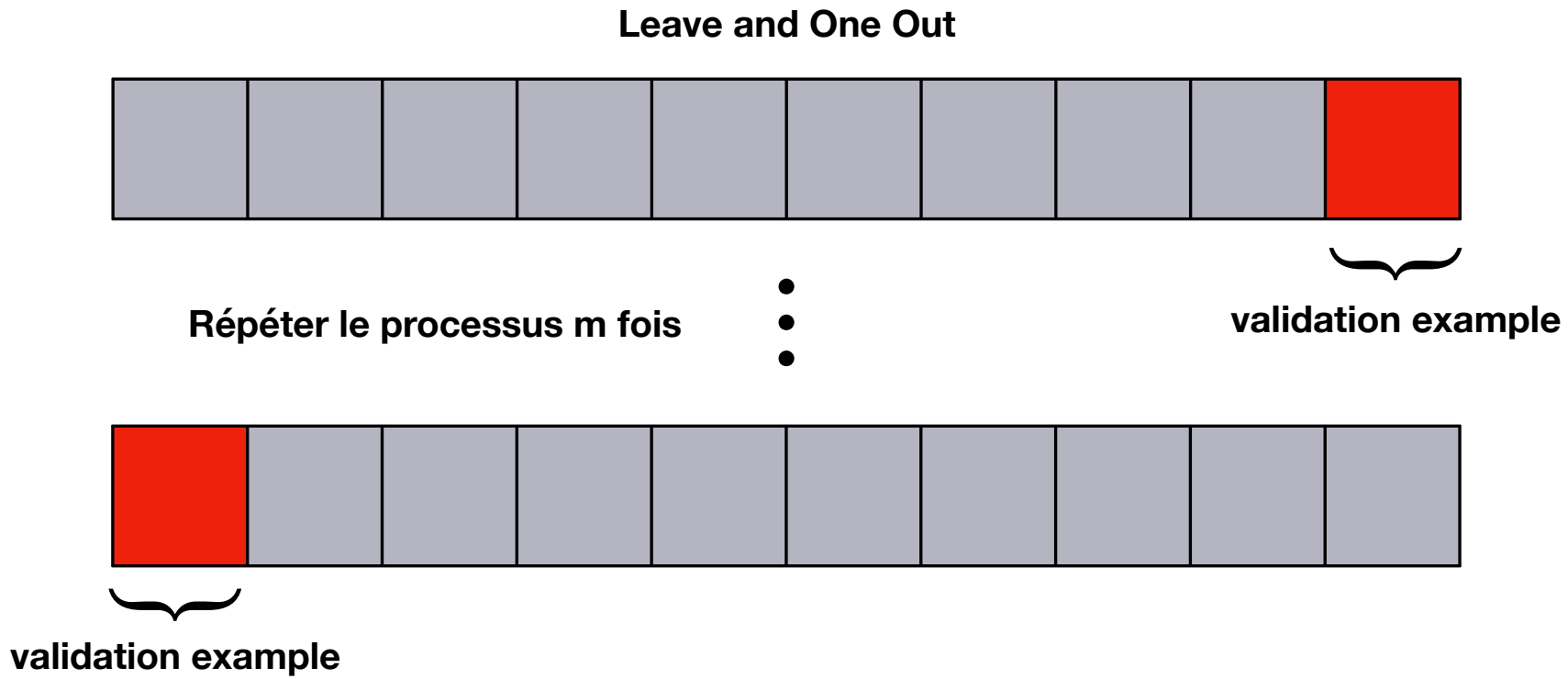
Cross validation III

Erreur moyenne en CV.

L'évaluation de $\frac{1}{k}\sum_{j=1}^{k}\mathcal{R}_{S_j}(h)$, appelée erreur moyenne en cross-validation permet de quantifier à quel point notre algorithme est capable de généraliser sur des nouvelles données issues de la même distribution.
C'est donc un bon indicateur pour nous permettre de vérifier si on peut potentiellement mettre notre algorithme en production ou non. On verra que ce n'est pas son seul usage ... De plus, il serait également bon de pouvoir tester notre procédure sur des données non rencontrées par le modèle jusqu'à présent.

Leave and One Out

Le principe reste le même mais on utilise cette fois-ci un seul exemple pour valider à chaque run, on doit effectuer un plus grand nombre d'apprentissage (m au lieu de k).



On retient le modèle ayant les meilleurs résultats en moyenne sur les m-folds qui servent à la validation

En ensuite ... ?

On dispose de données S, d'une loss ℓ et d'un algorithme et d'un processus permettant de sélectionner le meilleur modèle ... donc a priori on a tout ce qu'il faut pour se lancer dans la pratique (modulo la présentation des modèles).

En fait non ... avant il nous reste une dernière chose à regarder/étudier ! Qu'est-ce qui se passe si l'erreur observée sur l'ensemble d'apprentissage est sensiblement différente de l'erreur observée en validation ?

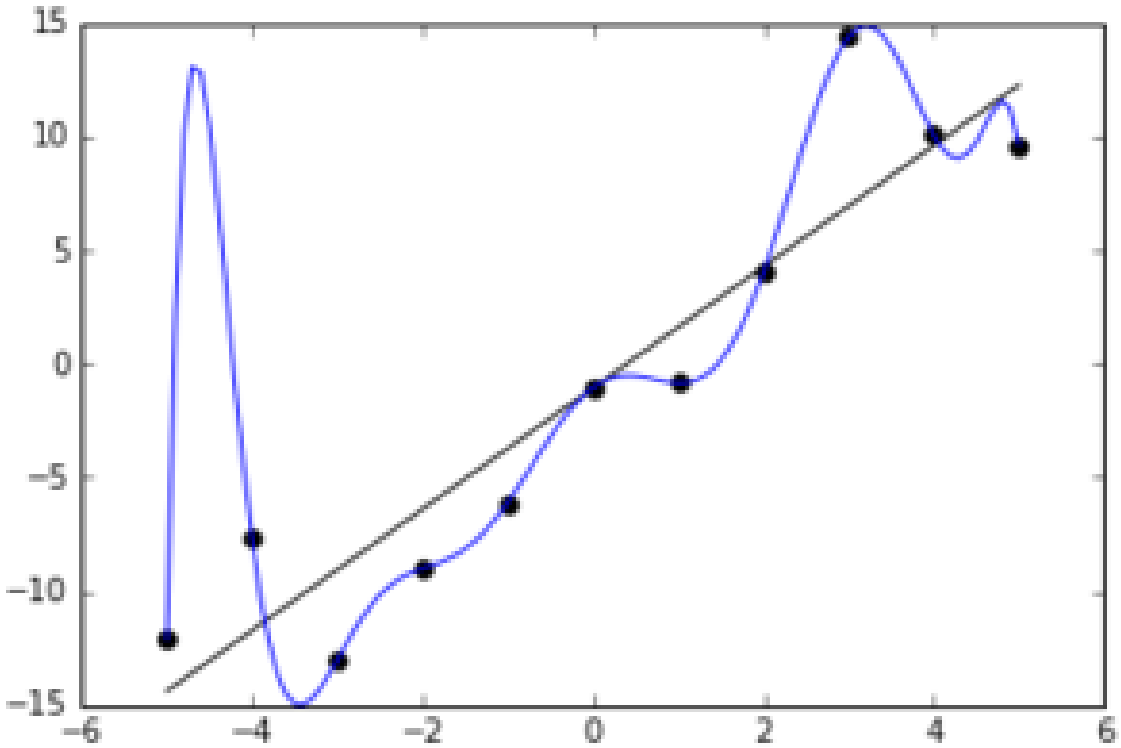
- si l'erreur en validation est **plus faible** que l'erreur sur l'ensemble d'entraînement,
- ou inversement, si l'erreur en validation **est plus** grande que l'erreur sur l'ensemble d'entraînement.

Overfitting ou Sur-Apprentissage I

Définition 2.4: Overfitting

En statistiques, l'overfitting survient quand le modèle se focalise tellement sur les données qu'il cherche également à apprendre le bruit présent. Ce phénomène apparaît lorsque **le modèle appris est trop complexe** ou lorsque **l'ensemble d'entraînement est trop petit** par rapport au nombre de paramètres de votre modèle (degrés de liberté).

Overfitting ou Sur-Apprentissage II

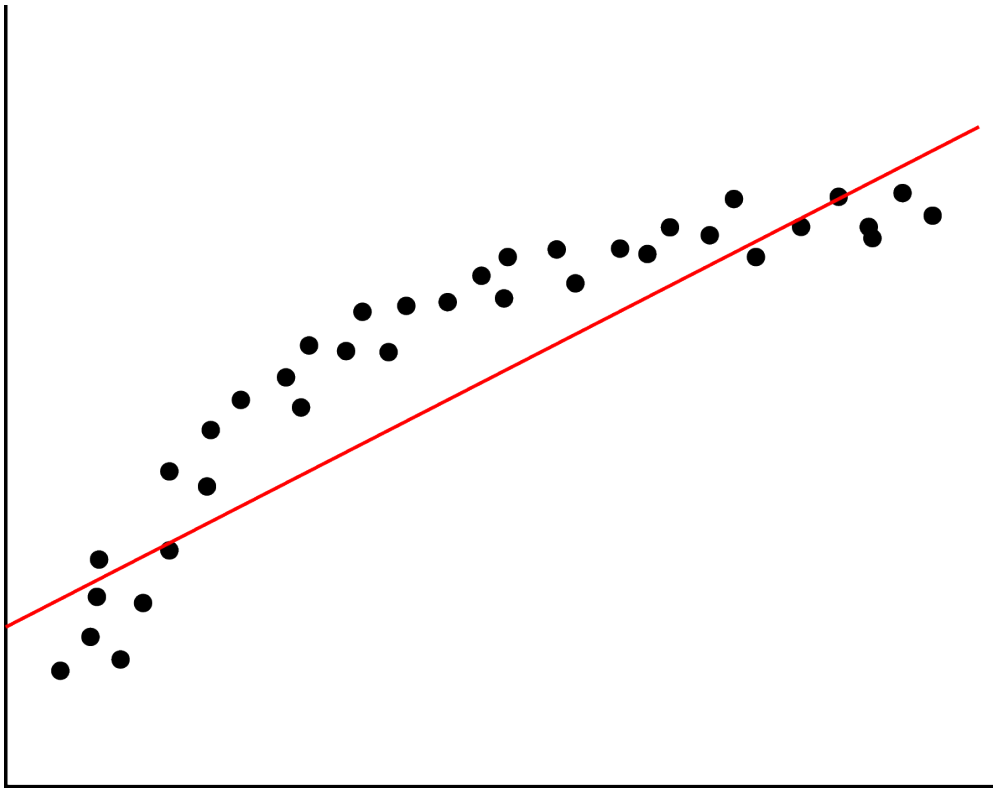


Underfitting ou Sous-Apprentissage I

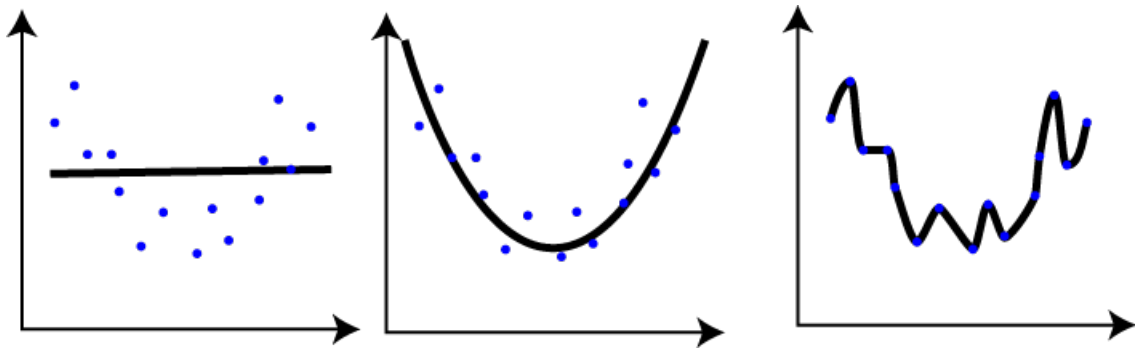
Définition 2.5: Underfitting

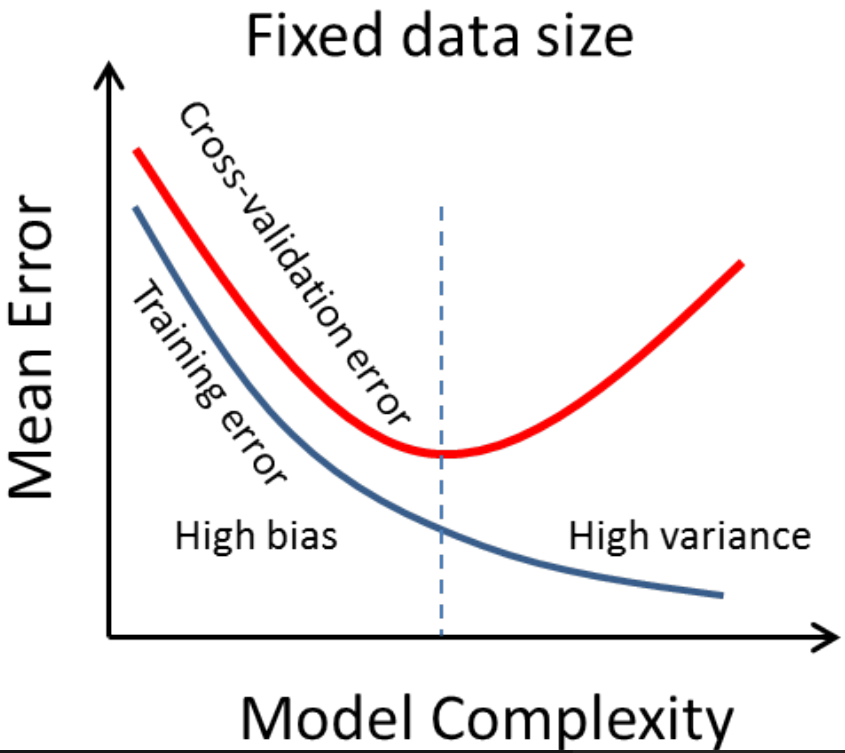
L'underfitting apparaît lorsque le modèle utilisé n'est pas capable d'appréhender la tendance présente dans les données. Cela survient surtout lorsque le modèle utilisé est beaucoup trop simple, comme l'utilisation d'un modèle linéaire pour approximer une courbe quadratique

Underfitting ou Sous-Apprentissage II



Underfitting-Overfitting I

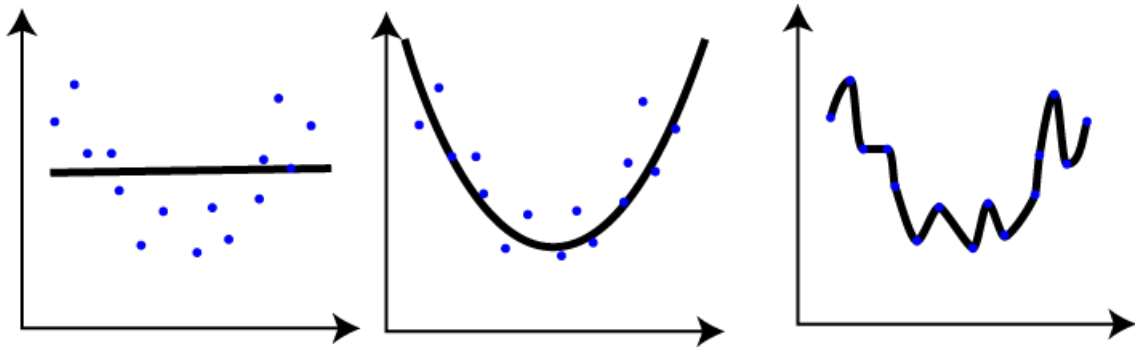




Comment savoir s'il y overfitting ou underfitting ?

- si l'erreur d'entraînement << erreur en CV, **overfitting**
- si l'erreur d'entraînement >> erreur en CV ou simplement élevée, **underfitting**

Underfitting-Overfitting II



Considérons que nos hypothèses sont issues d'un ensemble $\mathcal{H}$, appelé espace d'hypothèses

Overfitting : il intervient lorsque l'espace d'hypothèses est **trop riche**, *i.e.* si notre espace d'hypothèses est beaucoup **trop grand** et contient **des modèles potentiellement complexes**.

Unerfitting : il intervient lorsque l'espace d'hypothèses est **pauvre**, *i.e.* si notre espace d'hypothèses est beaucoup **trop petit** et contient des modèles qui sont **simples**.

Que faire ?

Pour traiter l'underfitting on peut tout simplement changer la classe de modèle que l'on apprend. En revanche pour l'overfitting, il va falloir trouver un moyen de *simplifier le modèle appris*.

Le rasoir d'Occam Il est également appelé principe de simplicité ou de parcimonie, il consiste à trouver l'explication (le modèle) le plus simple permettant d'expliquer les données :

no sunt multiplicanda entia praeter necessitatem" (William of Ockham)

Cela passe par l'ajout d'un terme de **Régularisation** dans notre problème d'optimisation.

Régularisation

L'introduction d'un terme de régularisation dans notre problème d'optimisation

$$h^\star = \arg\min_{h \in \mathcal{H}} \mathcal{R}_S(h).$$

Définition 2.6: Régularisation

Une régularisation est un terme, en statistiques et plus particulièrement en Machine Learning qui fait référence à l'**ajout d'un terme additionnel dans un problème d'optimisation** permettant d'éviter le sur-apprentissage. Il prend la forme d'une fonction qui va **pénaliser les modèles trop complexes** : il va donc chercher à lisser les modèles pour les rendre plus lisses ou encore restreindre les valeurs que peuvent prendre les paramètres d'un modèle.

Risque empirique régularisé

Le nouveau problème d'optimisation s'écrit alors :

$$h^\star = \arg\min_{h \in \mathcal{H}} \mathcal{R}_S(h) + \lambda \|h\|,$$

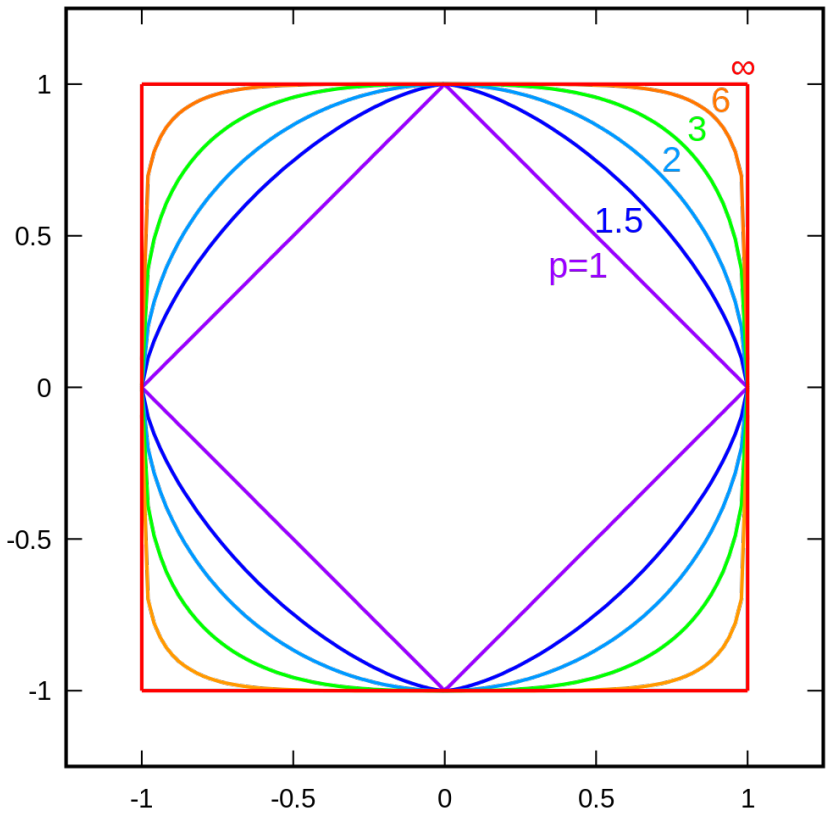
où

- λ est un paramètre de régularisation (on l'appelle aussi hyper-paramètre pour les distinguer des paramètres du modèle h),
- $\|\cdot\|$ est une fonction norme.

On va donc retourner l'hypothèse qui est capable d'atteindre le meilleur compromis entre performance et complexité.

Normes usuelles

Quelques exemples de normes : $\|h\|_p = \left(\sum_{i=1}^{d} h_i^p\right)^{\frac{1}{p}}$.

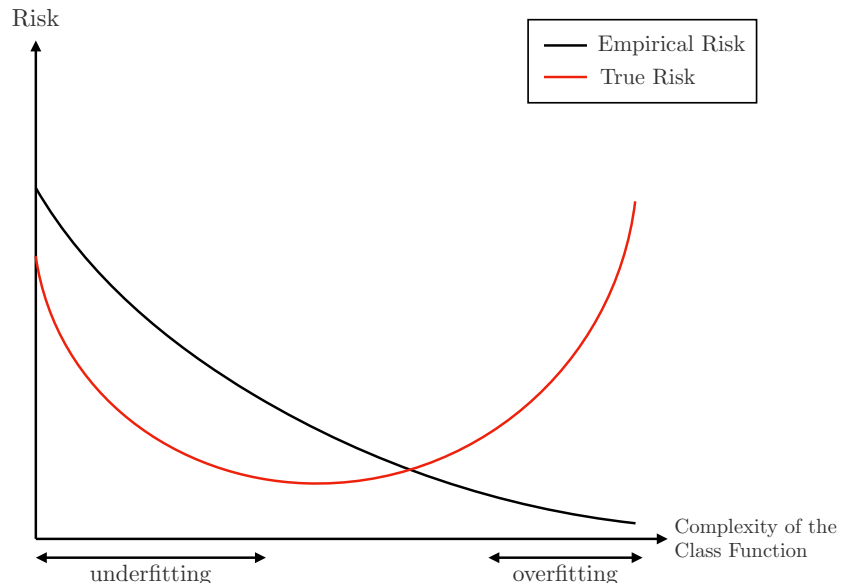


Rôle et choix du paramètre λ

Retournons à notre problème d'optimisation

$$h^\star = \arg\min_h \sum_{i=1}^{m} \ell(h(\mathbf{x}), y) + \lambda \|h\|,$$

où λ : paramètre de régularisation, contrôle la complexité de notre hypothèse, indirectement, on dit aussi qu'il contrôle la richesse de notre espace d'hypothèses $\mathcal{H}$.



Choisir λ qui se trouve à l'intersection des deux courbes ! Mais comment ?

Choix de λ et ... retour à la validation croisée

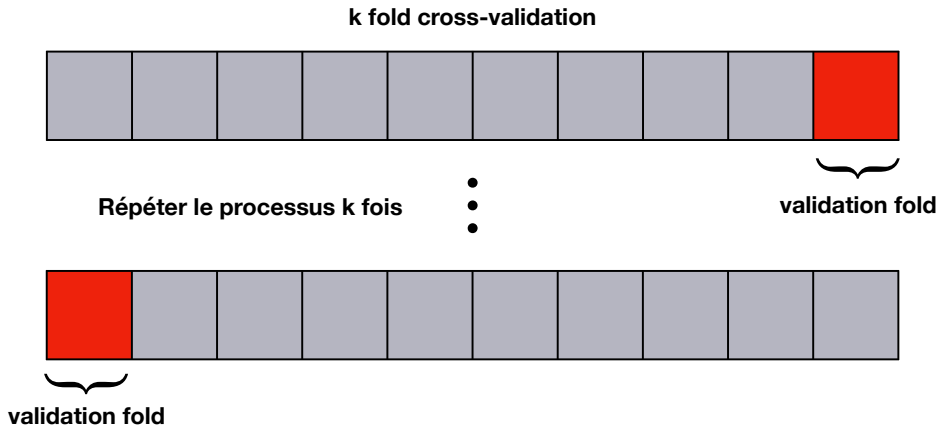
Notre ensemble d'apprentissage peut aussi servir à vérifier les capacités qu'à notre modèle **à généraliser pour le choix d'un hyper-paramètre en particulier**

- Validation standard : on sépare notre échantillon d'apprentissage en deux ensembles *train/valid* (2/3 - 1/3)
- k-fold cross validation : partition de l'échantillon en k ensembles. $k-1$ servent à apprendre et 1 sert à la validation
- Leave and One Out : on apprend sur tous les exemples sauf 1 qui sert à valider le modèle

k-fold cross validation comme processus de tunage

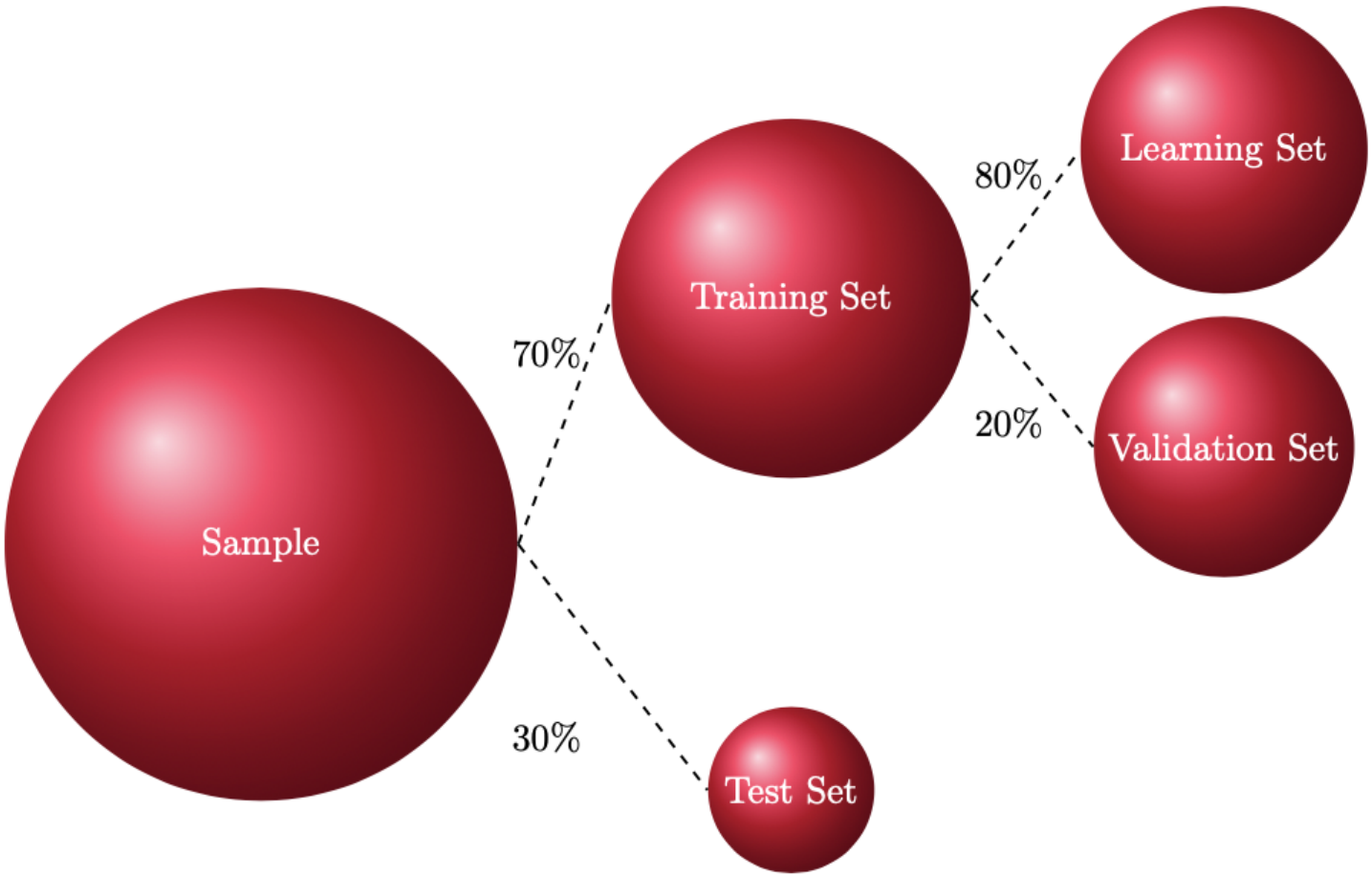
Le processus se fait de la façon suivante :

- on se fixe une grille de valeurs pour l'hyper-paramètre à optimiser : $\lambda \in (\lambda_1, \lambda_2, ..., \lambda_p)$
- pour chaque valeur de λ_i on calcule notre erreur moyenne de cross validation à l'aide d'une k-CV : $\frac{1}{k} \sum_{j=1}^{k} \mathcal{R}_{S_j}(h, \lambda_i)$



- on conserve la valeur de λ_i pour laquelle l'erreur moyenne en CV est la plus faible.

Découpage du jeu de données



Quelques remarques

- Pour éviter le problème de sur-apprentissage on utilise à la fois un (ou plusieurs) terme de régularisation dans notre problème d'optimisation
- On combine cela à de la k-fold cross validation (on prendra communément $k = 10$) afin de s'assurer que le modèle généralise bien
- Attention : ces approches sont empiriques ! Pour bien faire, il faudrait étudier les garanties en généralisation

$$|\underset{(\mathbf{x},y)\sim\mathcal{D}}{\mathbb{E}} \ell(h(\mathbf{x}), y) - \underset{(\mathbf{x},y)\sim S}{\mathbb{E}} \ell(h(\mathbf{x}), y)| \leq \mathcal{O}(\sqrt{1/m}, \lambda^{-1})$$

- Il faut parfois faire la distinction entre le problème d'optimisation et le critère de performance que vous souhaitez optimiser.