

# Machine Learning: introduction opérationnelle; concepts et premiers exemples à l'aide de Jupyter

Jérémie Jakubowicz

[jeremie.jakubowicz@telecom-sudparis.eu](mailto:jeremie.jakubowicz@telecom-sudparis.eu)  
Télécom SudParis, Institut Mines-Télécom

Jeudi 24 novembre 2016  
École Polytechnique, Palaiseau

# The Home of Data Science

COMPETITIONS • CUSTOMER SOLUTIONS • JOBS BOARD

Get started »

# Data Science Challenges



DRIVEN DATA

COMPETITIONS FORUM ABOUT WORK WITH US SIGN UP LOG IN

## Data science competitions to save the world

I'm with an organization I'm a data scientist

# Data Science Challenges

 **datascience.net**

Accueil   Challenges   Classement   Aide    Se connecter

## Mettez votre talent de data scientist au service des entreprises

Participez à des challenges excitants, trouvez le meilleur modèle, gagnez des prix, soyez reconnus pour votre expertise, soyez visibles.

Entreprises : [Comment créer une compétition ?](#)



Je m'inscris

Nom complet (prénom nom)

E-mail

Mot de passe

J'ai lu et j'accepte les [conditions d'utilisation](#) du site datascience.net

[Je m'inscris](#)   [Je suis déjà inscrit](#)

Suggestions & support



# Exemple



Completed • \$5,000 • 925 teams

## Give Me Some Credit

Mon 19 Sep 2011 – Thu 15 Dec 2011 (4 years ago)

Dashboard	
Home	
Data	
Make a submission	
Information	
Description	
Evaluation	
Rules	
Prizes	
Forum	
Leaderboard	
Public	
Private	
My Team	
GitHub	
My Submissions	

Competition Details » Get the Data » Make a submission

**Improve on the state of the art in credit scoring by predicting the probability that somebody will experience financial distress in the next two years.**

Banks play a crucial role in market economies. They decide who can get finance and on what terms and can make or break investment decisions. For markets and society to function, individuals and companies need access to credit.

Credit scoring algorithms, which make a guess at the probability of default, are the method banks use to determine whether or not a loan should be granted. This competition requires participants to improve on the state of the art in credit scoring, by predicting the probability that somebody will experience financial distress in the next two years.

## Détails des données

La base de données dont on dispose consiste en  $n$  données de la forme  $(x_i, y_i)$  pour  $1 \leq i \leq n$ . Par exemple, pour  $x_1$ , i.e. la première entrée de la base, on lit :

RevolvingUtilizationofUnsecuredLines	0.766126609
age	45
NumberOfTime30-59DaysPastDueNotWorse	2
DebtRatio	0.802982129
MonthlyIncome	9120
NumberOfOpenCreditLinesAndLoans	13
NumberOfTimes90DaysLate	0
NumberRealEstateLoansOrLines	6
NumberOfTime60-89DaysPastDueNotWorse	0
NumberOfDependents	2

et pour  $y_1$  on lit :

SeriousDlqin2yrs 1

## Détails des données (suite)

Variable Name	Description	Type
SeriousDlqin2yrs	Person experienced 90 days past due delinquency or worse	Y/N
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits	percentage
age	Age of borrower in years	integer
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse	integer
DebtRatio	Monthly debt payments, alimony, living costs divided by monthly gross income	percentage
MonthlyIncome	Monthly income	real
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit	integer
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due.	integer
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines	integer
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse	integer
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children)	integer

# Première Partie

## Concepts fondamentaux du Machine Learning

# Cadre Formel

Quelques éléments de terminologie :

- ▶ Les  $x_i$  s'appellent **régresseurs** ou **covariables**. Formellement ce sont des vecteurs d'un certain  $\mathcal{X} = \mathbb{R}^p$ .
- ▶ Les  $y_i$  s'appellent **cible** ou **variable d'intérêt**. Formellement ce sont des éléments de l'ensemble  $\mathcal{Y} = \{0, 1\}$  (ou de  $\{-1, 1\}$  ou de  $\mathbb{R}$  si on préfère).
- ▶ Une **donnée** consiste donc en le couple  $(x, y) \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
- ▶ La **base de données** ou **l'échantillon** de  $n$  données est parfois noté  $D_n$  et est un élément de  $\mathcal{Z}^n$ .
- ▶ L'objectif est de fabriquer un bon (à *définir*) **prédicteur** ou **classifieur**, *i.e.* une fonction

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

- ▶ On verra qu'il est souhaitable de se restreindre à une sous-famille  $\mathcal{H}$  des fonctions de  $\mathcal{X} \rightarrow \mathcal{Y}$  appelé **modèle**

## Cadre Formel (suite)

On fait l'hypothèse que les  $(x_i, y_i)$  sont les réalisations de  $n$  variables aléatoires  $(X_i, Y_i)$  *indépendantes et identiquement distribuées*, suivant une loi  $\mathbb{P}$  sur  $\mathcal{Z}$ .

### Remarque 1

Ca ne veut pas dire que  $Y_i$  est indépendante de  $X_i$ ; ce qui rendrait tout espoir de prévision vain, mais que  $(X_i, Y_i)$  est indépendante du reste de la base.

### Remarque 2

Le cadre est différent de celui des *statistiques paramétriques*, où on suppose que les  $X_i, Y_i$  proviennent d'un modèle  $\mathbb{P}_\theta$  et où on cherche à retrouver ce  $\theta$ . Ici la loi de probabilité est supposée **fixe et inconnue**, c'est l'ensemble des prédicteurs qui constitue l'acte de modélisation.

# Mesures de performance

Quand dit-on qu'un prédicteur est meilleur qu'un autre ? Par le biais d'une **fonction de perte**  $\ell(y, y')$  qui correspond au coût de prédire  $y$  alors que la réalité est  $y'$ . Par exemple :

$$\ell(y, y') = \mathbb{1}\{y \neq y'\}$$

On peut alors définir le

- ▶ **Risque** d'un prédicteur  $h \in \mathcal{H} : R(h) = \mathbb{P}[\ell(h(X), Y)]$
- ▶ Et le risque du modèle  $\mathcal{H} : \inf_{h \in \mathcal{H}} R(h)$

## Remarque

Il faut bien avoir en tête que  $\mathbb{P}$  est inconnu. Donc les deux quantités précédentes sont malheureusement inaccessible dans la pratique.

## Mesures de performance (suite)

Il est naturel de s'intéresser au **risque empirique** d'un prédicteur  $h \in \mathcal{H}$  :

$$R_n(h) = \frac{1}{n} \sum \ell(h(x_i), y_i)$$

Qui, lui, est calculable, pour peu que  $h$  ne soit pas trop coûteux à évaluer et  $n$  pas trop grand (conditions pas toujours satisfaites dans un contexte de données massives mais c'est un autre sujet...)

Dans certains cas, on peut même espérer approcher aussi précisément qu'on veut le **risque empirique du modèle** :

$$\inf_{h \in \mathcal{H}} R_n(h)$$

## Mesures de performance (suite)

Dans le cas de l'exemple important  $\ell(y, y') = \mathbb{1}\{y \neq y'\}$ , l'expression  $R_n(h)$  s'interprète aisément : c'est le **nombre d'erreurs commises** par le classifieur  $h$  sur les données.

Le risque du modèle correspond à un classifieur qui atteint le **nombre minimal d'erreurs** sur la base (l'infimum est toujours atteint dans ce cas).

## Surapprentissage

Il est fondamental de constater que si il est légitime d'espérer que, pour un  $h$  générique,  $R_n(h)$  approche raisonnablement bien  $R(h)$ , avec grande probabilité, quand on a beaucoup de données à disposition, ce n'est pas nécessairement le cas de  $\inf_{h \in \mathcal{H}} R_n(h)$  et  $\inf_{h \in \mathcal{H}} R(h)$ .

La raison est la suivante : imaginons un jeu impliquant un **joueur** et la **Nature**. Le jeu est le suivant : la **Nature** suit invariablement la même stratégie, elle tire  $n$  données de façon iid suivant  $\mathbb{P}$ , qu'on suppose ici (et seulement ici !) **connue du joueur**. Le **joueur**, lui, doit choisir un prédicteur  $h \in \mathcal{H}$  du modèle. On évalue ensuite  $R_n(h)$ , le **joueur** cherche à minimiser cette erreur (la Nature a toujours la même stratégie donc peut importe son objectif).

Alors si  $\mathcal{H}$  est un modèle puissant, c'est-à-dire capable d'atteindre un petit risque empirique, quelles que soient les données, l'ordre dans lequel jouent le **joueur** et la **Nature** est très important.

# Surapprentissage (suite)

Jouons !

On prend  $\mathcal{H} = \mathbb{R} \rightarrow \{0, 1\}$ . Voici  $\mathbb{P}$  (par exemple  $\mathbb{P}[Y = 1|X = x] = (1 + \exp(x))^{-1}$  :



1



0

## Surapprentissage (suite)

Jouons !

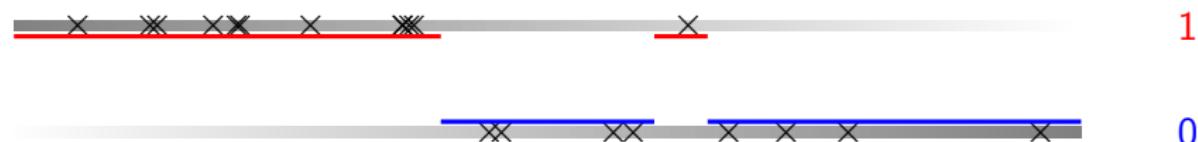
On prend  $\mathcal{H} = \mathbb{R} \rightarrow \{0, 1\}$ . Voici  $\mathbb{P}$  (par exemple  $\mathbb{P}[Y = 1|X = x] = (1 + \exp(x))^{-1}$  :  
Scénario 1, la Nature joue en premier :



## Surapprentissage (suite)

Jouons !

On prend  $\mathcal{H} = \mathbb{R} \rightarrow \{0, 1\}$ . Voici  $\mathbb{P}$  (par exemple  $\mathbb{P}[Y = 1|X = x] = (1 + \exp(x))^{-1}$  :  
Scénario 1, la Nature joue en premier :



Puis le joueur joue (en rouge et bleu) et fait un sans faute.

## Surapprentissage (suite)

Jouons !

On prend  $\mathcal{H} = \mathbb{R} \rightarrow \{0, 1\}$ . Voici  $\mathbb{P}$  (par exemple  $\mathbb{P}[Y = 1|X = x] = (1 + \exp(x))^{-1}$  :

Scénario 1, la Nature joue en premier :



Puis le joueur joue (en rouge et bleu) et fait un sans faute.

Scénario 2, le joueur joue la même chose mais en premier.



Puis la Nature joue et le joueur fait plusieurs fautes ! Il a fait du surapprentissage en généralisant à tort ce qu'il a observé de la Nature. En connaissant la loi  $\mathbb{P}$  il aurait dû jouer le classifieur  $h_*(x) = \mathbb{1}(x < 0)$ .

## Surapprentissage (suite)

L'exemple fait sentir le **compromis** qui existe entre la **complexité** du modèle  $\mathcal{H}$  et son pouvoir de **généralisation**. Il a plusieurs manières de reformuler ce compromis (biais/variance, approximation-fluctuations). On en retient les idées importantes suivantes :

- ▶ Plus le modèle est complexe, plus on risque le surapprentissage
- ▶ On ne peut pas comparer deux modèles  $\mathcal{H}_1$  et  $\mathcal{H}_2$  sur la seule base de leur risque empirique  $\inf_{h \in \mathcal{H}_1} R_n(h)$  et  $\inf_{h \in \mathcal{H}_2} R_n(h)$ . Sinon lorsque  $\mathcal{H}_1 \subset \mathcal{H}_2$  on privilégié nécessairement  $\mathcal{H}_2$  et c'est une erreur.

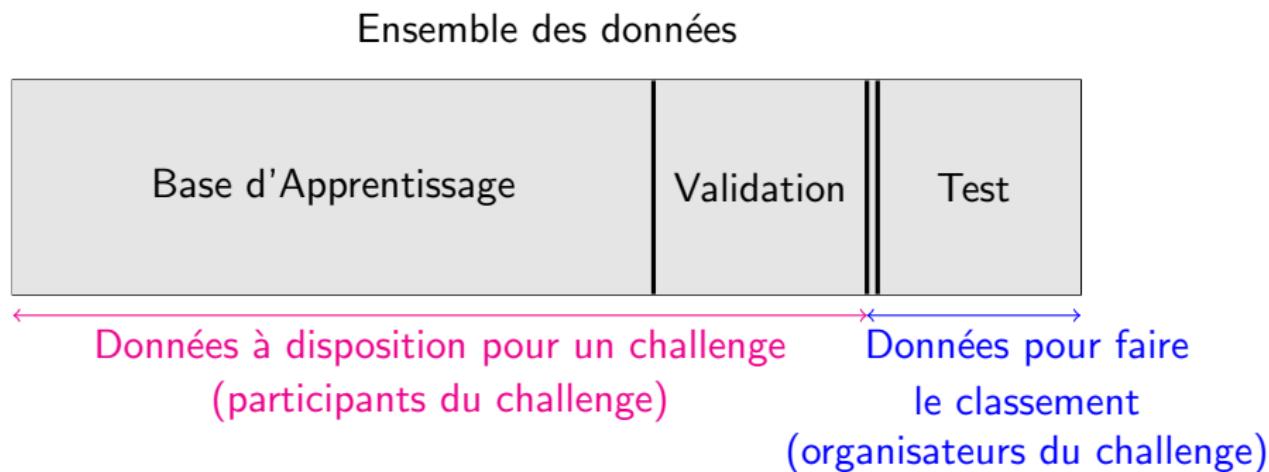
## Que faire pour détecter/éviter le surapprentissage ?

Une stratégie élégante mais difficile à mettre en œuvre consiste à calculer une borne  $\text{pen}(\mathcal{H})$  avec grande probabilité pour la quantité  $R(\mathcal{H}) - R_n(\mathcal{H})$ . Ces bornes font naturellement intervenir des mesures de la complexité de  $\mathcal{H}$ . Lorsque les calculs sont tractables, on utilise  $R_n(h) + \text{pen}(\mathcal{H})$  pour comparer les modèles entre eux.

Heureusement il existe une stratégie très simple qui permet de détecter/éviter le surapprentissage et qui consiste d'une certaine manière, on va le voir, à intervertir les ordres entre le joueur et la Nature, c'est la **validation croisée**.

## Validation Croisée

L'idée est la suivante. On commence par découper nos données en deux parties : la **base d'apprentissage** et la **base de validation**. Dans les challenges, il y aura une troisième partie, la **base de test** à laquelle on n'aura pas accès pendant la compétition.



## Validation Croisée (suite)

Formellement on découpe l'intervalle  $1 \leq i \leq n$  en deux sous-ensembles  $A$  et  $V$ . On ne parle plus de l'échantillon de test, dans la mesure où soit on n'en dispose pas, soit on le fait sortir une fois pour toute de la phase d'apprentissage. Imaginons qu'on cherche maintenant à tester la validité du modèle  $\mathcal{H}$  :

1. On approche

$$\hat{h} = \arg \min_{h \in \mathcal{H}} \sum_{i \in A} \ell(h(x_i), y_i)$$

2. On calcule

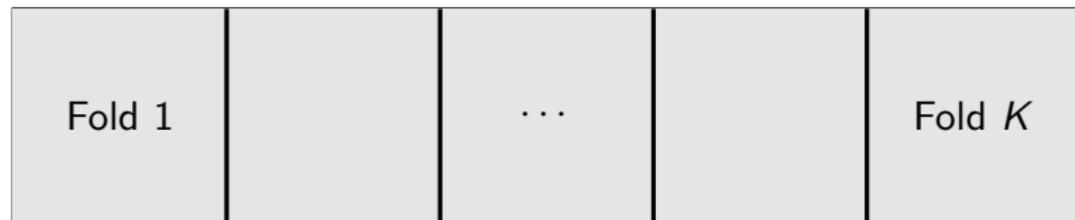
$$R_{CV}(\hat{h}) = \frac{1}{|V|} \sum_{i \in V} \ell(h(x_i), y_i)$$

Et on utilise  $R_{CV}(\hat{h})$  à la place de  $R_n(\mathcal{H})$  pour éviter le surapprentissage. Ainsi, si on doit comparer deux modèles, on compare leur " $R_{CV}$ " et pas leur " $R_n$ ".

## Validation en $K$ folds

En réalité on utilise souvent en pratique la variante suivante de la validation croisée, dite validation en  $K$ -folds :

Ensemble des données



1. On note  $I_1, \dots, I_K$  les  $K$  folds
2. On estime  $\hat{h}_k = \arg \min_{h \in \mathcal{H}} \sum_{i \notin I_k} \ell(h(x_i), y_i)$
3. On calcule  $R_{K-\text{Folds}}(\mathcal{H}) = \frac{1}{n} \sum_{1 \leq k \leq K} \sum_{i \in I_k} \ell(\hat{h}_k(x_i), y_i)$

Cette variante est plus symétrique que la précédente car chaque donnée interviendra à la fois pour estimer un  $\hat{h}_k$  et pour évaluer  $R_{K-\text{Folds}}(\mathcal{H})$ . On utilise  $R_{K-\text{Folds}}(\mathcal{H})$  à la place de  $R_{CV}(\mathcal{H})$  ou, pire,  $R_n(\mathcal{H})$  pour juger la validité du modèle  $\mathcal{H}$ .

## Retour sur la mesure de performance

Dans le cas de la classification, il peut être intéressant d'estimer  $p(X) = \mathbb{P}[Y = 1|X]$  au lieu, ou en plus, de prédire  $y = 0$  ou  $y = 1$  en fonction de  $x$ . Dans ce cas, la fonction de perte prend la forme :

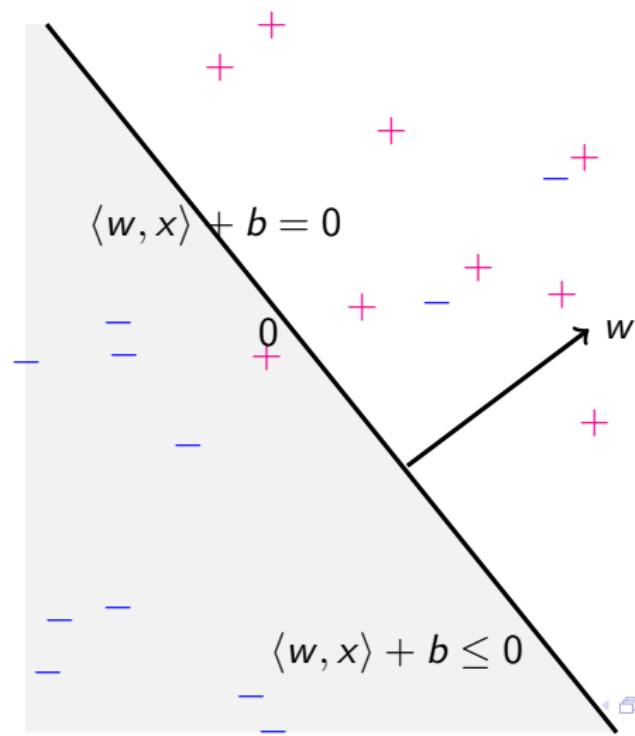
$$\ell(p, y) = y \log p + (1 - y) \log(1 - p)$$

qui correspond à la **log-vraisemblance** de l'observation  $y$  pour une loi de Bernouilli de paramètre  $p$ . Cette fonction de perte s'appelle aussi **perte logarithmique**.

En général on préfère  $p(x)$  à  $h(x)$  car il est toujours possible de passer de  $p(x)$  à  $h(x)$  par un seuil, tandis que l'inverse n'est pas possible. Dorénavant on confond  $h$  et  $p$ , en désignant toutes les prévisions qu'elles portent sur les probabilités ou la variable d'intérêt par  $h(x)$ .

## Un peu de géométrie pour finir cette première partie

Géométriquement, le problème de classification linéaire (ou plutôt précisément affine) correspond à trouver un “bon” hyperplan de séparation pour les données.



## Deuxième Partie

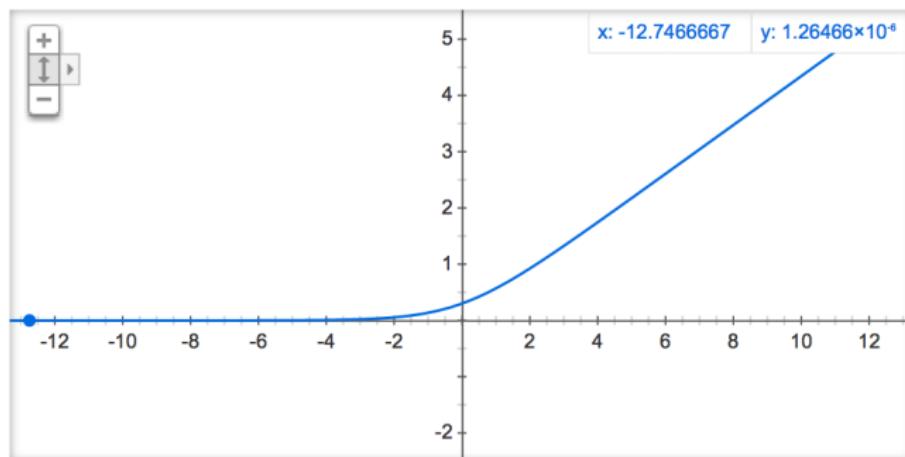
### Échantillon d'algorithmes du Machine Learning

## Régression Logistique

On cherche  $w$  (on ne mentionne plus le biais  $b$  en l'incorporant dans  $w$  quitte à rajouter l'entrée 1 à tous les vecteurs  $x_i$ ) qui minimise :

$$\sum_{i=1}^n \log(1 + \exp(-y_i \langle w, x_i \rangle))$$

Le terme  $-y_i \langle w, x_i \rangle$  a un sens très clair. Quant à la fonction  $t \mapsto \log(1 + \exp(t))$ , elle présente le graphe suivant :



## Régression Logistique Pénalisée

On peut modifier la fonction à minimiser de sorte à essayer de diminuer les coefficients du vecteur  $w$ , à travers le mécanisme de pénalisation :

$$\inf_w \sum_i \log(1 + \exp(-y_i \langle x_i, w \rangle)) + \lambda g(w)$$

où  $g(w)$  est une fonction positive qui croît quand les coordonnées de  $w$  croissent. Les trois exemples les plus fréquemment utilisés sont :

- ▶  $g(w) = \|w\|_1$  (dit Lasso)
- ▶  $g(w) = \|w\|_2^2$  (dit Ridge)
- ▶  $g(w) = \|w\|_1 + \mu \|w\|_2^2$  (dit Elastic Net)

# Machines à Vecteurs de Support (SVM)

Dans ce cas également il s'agit d'une fonction à minimiser qui prend une forme de somme de deux termes : un terme d'attaché aux données et un terme de regularization,

$$\inf \sum_i (1 - y_i \langle x_i, w \rangle)_+ + C \|w\|^2$$

## Les 7 différences

On se rend compte que ces fonctions à minimiser se ressemblent dans leur structure :

$$\inf_w \sum_i f(-y_i \langle x_i, w \rangle) + \lambda g(w)$$

Il y en a d'autres qui ont aussi cette même forme. Ce constat est à la base d'un certain nombre de méthodes bien adaptées aux données massives.

Néanmoins, il ne faut pas se laisser abuser par la ressemblance entre ces fonctions : changer  $\|w\|_2^2$  en  $\|w\|$  ou  $\log(1 + \exp(t))$  en  $(1 + t)_+$  peut avoir beaucoup d'impact sur la solution. Les raisons de ces différences profondes dépassent cette courte introduction et sont de nature géométrique.

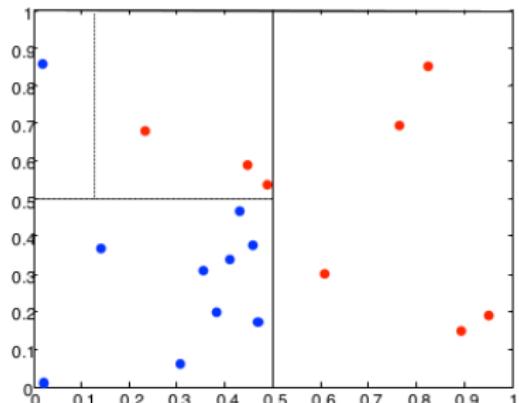
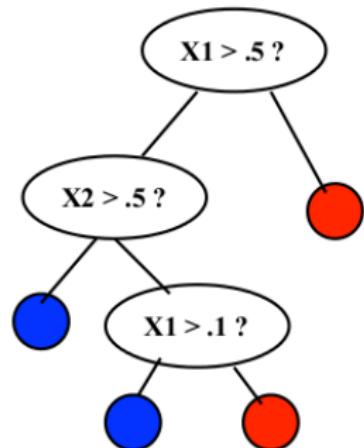
## Au delà du monde linéaire

Toutes les méthodes mentionnées jusqu'ici sont linéaires dans la mesure où elles reposent sur  $\langle w, x_i \rangle$ . Cette linéarité peut être une entrave si le lien entre  $y_i$  et  $x_i$  ne s'y prête pas bien. Il y a plusieurs options très utilisées :

- ▶ Les arbres/forêts
- ▶ Les noyaux
- ▶ Les réseaux de neurones
- ▶ Les plus proches voisins
- ▶ ...

Par faute de temps, on ne présentera que la première option dans cette introduction.

# Arbres de décision



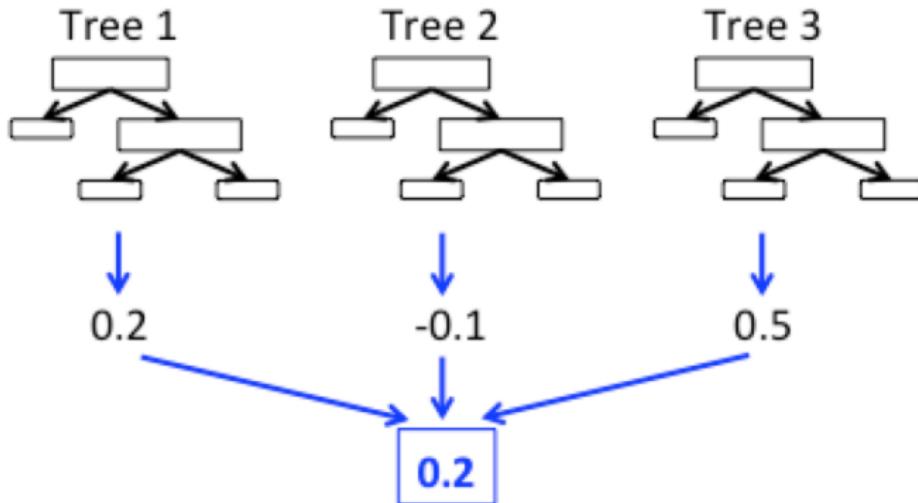
d'après Ankit Sharma sur quora.com

## Entraînement d'un arbre

Les arbres de profondeur 1 sont conceptuellement simples : ils correspondent à un seuil sur l'une des composantes des  $x_i$ . Pour choisir ce seuil et cette composante, on peut être exhaustif si la dimension des  $x_i$  est raisonnable : on examine toutes les coordonnées et tous les seuils (après avoir constaté que les valeurs des composantes de chaque  $x_i$  sont pivotales pour fixer ces seuils).

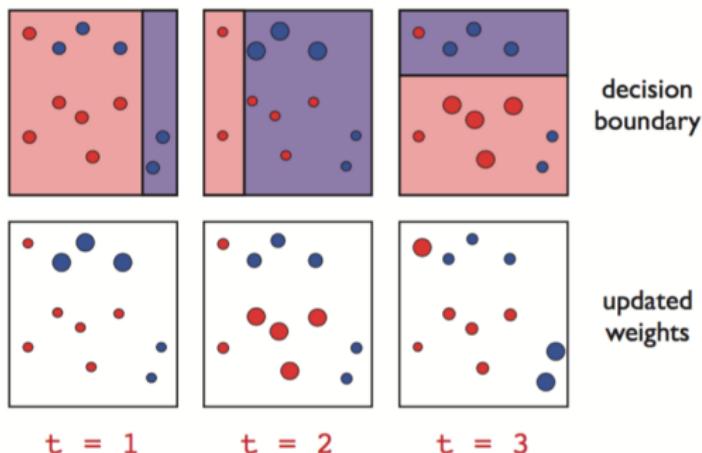
Pour les arbres de plus grande profondeur, on les entraîne de la même manière mais récursivement : on commence par trouver la première composante et la valeur du seuil pour on entraîne deux nouveaux arbres dans chacune des feuilles, etc. On stoppe quand un certain critère d'arrêt est satisfait (plus assez de points dans les feuilles en général).

## Forêts

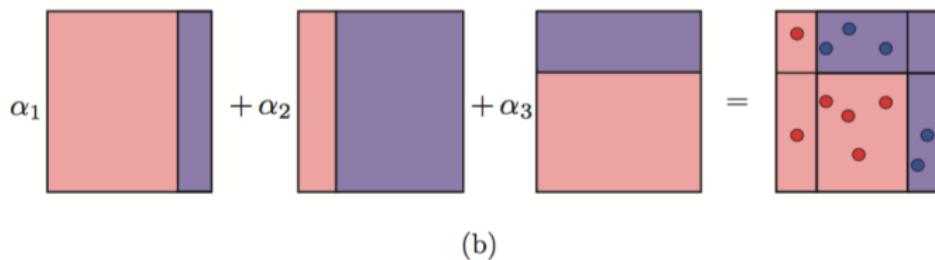


D'après <https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html>

# Boosting



(a)



(b)

D'après Mohri, Rostamizadeh et Talwalkar

# Mélange

Une approche fructueuse consiste à mélanger les modèles. On va le voir maintenant avec Jupyter

# Troisième Partie

## Expériences avec Jupyter