

## LE PERCEPTRON MULTICOUCHES (PMC)

Intéressons nous ici à un réseau “statique” (ou *feedforward*, i.e. pas de boucle rétroactive), dans un but d'apprentissage supervisé.

En voici quelques caractéristiques :

- ➊ **architecture** : PMC composé de couches successives, où 1 couche : ens. de neurones sans connexion entre eux ;
- ➋ **fonction de transfert** : un PMC réalise une transformation des variables d'entrée :

$$Y = \Phi(X_1, X_2, \dots, X_p; \alpha),$$

avec  $\alpha = (\alpha_{jkl})$  pr la  $j^{\text{è}}$  entrée ( $x_j$ ) du  $k^{\text{è}}$  neurone de  $l^{\text{è}}$  couche.

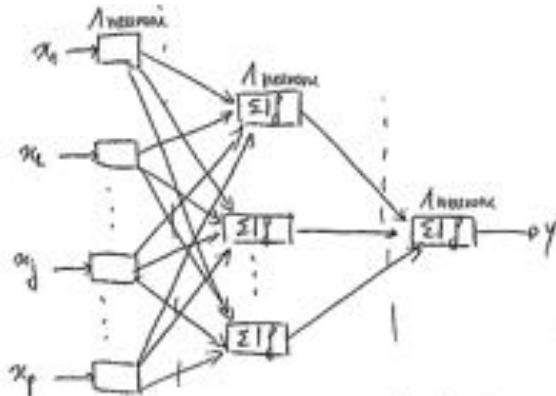
- ③ **généralisation** : cas de la régression, avec un perceptron à une couche cachée de  $q$  neurones, un neurone de sortie.  
⇒ La **fonction de transfert** s'écrit

$$Y = \Phi(x; \alpha, \beta) = \beta_0 + \beta^T z, \quad \text{avec } z_k = f(\alpha_{k0} + \alpha_k^T x),$$

pour  $k = 1, \dots, q$  (identifiant neurone ds couche cachée).

Usuellement, on a

- en rég. : dernière couche avec 1 seul neurone, avec  $f = Id$  ; tandis que neurones couche cachée ont une fonct. sigmoïde ;
- en classif. binaire : neurone de sortie muni de la fonction d'activation sigmoïde ;
- en discrimination à  $m$  classes :  $m$  neurones de sortie, munis de sigmoïde.



couche  
d'entrée

couche  
cachée

couche de  
sortie

non-complétude  
parmi les couches car  
ne modifie pas les  
entrées.

TRANSFERT

## 7 Réseau de neurones et Deep Learning

- Introduction
- Neurone formel et fonctionnement d'un perceptron multicouches
- Estimation des paramètres
- Paramétrage du réseau
- Deep learning et autres types de réseaux

## APPRENTISSAGE DU RÉSEAU

Supposons qu'on dispose d'une **base d'apprentissage** de  $n$  observations,  $(x_i^1, x_i^2, \dots, x_i^p; y_i)_{i=1, \dots, n}$ .

Prenons le cas de la régression (généralisable à tte fonction de perte dérivable, dc aussi à la discrimination cf Gini) et le réseau à

- une couche cachée à  $q$  neurones,
- une sortie linéaire.

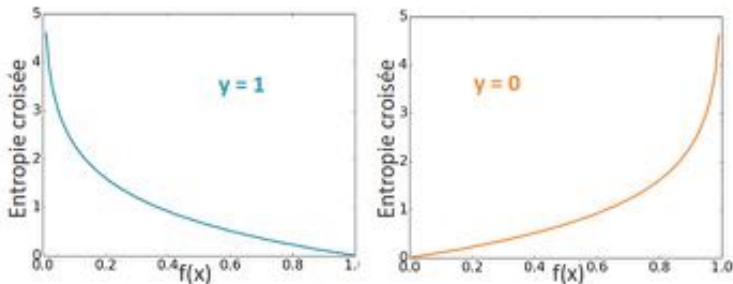
⇒ Les paramètres (poids) sont optimisés par **moindres carrés** :  
l'"apprentissage" minimise donc la perte quadratique

$$Q(\alpha, \beta) = \sum_{i=1}^n Q_i(\alpha, \beta) = \sum_{i=1}^n (y_i - \Phi(x_i; \alpha, \beta))^2,$$

avec  $\alpha = (\alpha_{jk})_{j=0, \dots, p; k=1, \dots, q}$ , et  $\beta = (\beta_k)_{k=0, \dots, q}$ .

# FONCTION DE PERTE CONVEXE - CLASSIFICATION

Dans le cas de la **classification**, nous allons choisir l'**entropie croisée**. Dans le cas binaire l'entropie croisée est définie par

$$\text{Erreur}(f(x^{(i)}), y^{(i)}) = -y^{(i)} \log f(x^{(i)}) - (1 - y^{(i)}) \log(1 - f(x^{(i)})).$$


Quand  $y=0$ , l'entropie croisée est d'autant plus élevée que  $f(x)$  est proche de 1.  
Réciproquement, quand  $y=1$ , l'entropie croisée est d'autant plus grande que la prédiction est proche de 0.

## ESTIMATION : EVALUATION DES GRADIENTS

Les algorithmes utilisés pour l'optimisation sont généralement basés sur une évaluation du gradient par rétropropagation.

On détaille l'algorithme le plus utilisé : la rétropropagation de l'erreur !

Consiste en évaluer la dérivée de la fonction de coût en **une seule observation** et par rapport à l'ensemble des paramètres, puis ajuster les paramètres, puis réévaluer avec les nouveaux paramètres sur une nvelle obs, et ainsi de suite.

Notons  $z_{ki} = f(\alpha_{k0} + \alpha_k^T x_i)$ , et  $z_i = (z_{i1}, \dots, z_{iq})$ .

Ainsi,  $z_i$  sont les valeurs pour l'individu  $i$  dans chaque neurone de la couche cachée, et  $z_{ki}$  la valeur de l'individu  $i$  dans le neurone  $k$ .

Etudions les dérivées partielles de l'erreur :

$$\begin{aligned}
 \frac{\partial Q_i}{\partial \beta_k} &= \frac{\partial (y_i - \Phi(x_i; \alpha, \beta))^2}{\partial \beta_k} = \frac{\partial (y_i - (\beta_0 + \beta^T z_i))^2}{\partial \beta_k} \\
 &= -2(y_i - \Phi(x_i)) (\beta^T z_i) z_{ki} \\
 &= \delta_i z_{ki}
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial Q_i}{\partial \alpha_{kj}} &= \frac{\partial (y_i - (\beta_0 + \beta^T z_i))^2}{\partial \alpha_{kj}} = \frac{\partial (y_i - (\beta_0 + \beta^T (f(\alpha_{k0} + \alpha_k^T x_i))))^2}{\partial \alpha_{kj}} \\
 &= -2(y_i - \Phi(x_i)) (\beta^T z_i) \beta_k f'(\alpha_k^T x_i) x_{ij} = \delta_i \beta_k f'(\alpha_k^T x_i) x_{ij} \\
 &= s_{ki} x_{ij}
 \end{aligned}$$

→  $s_{ki}$  : terme d'erreur sur chaque neurone caché pr l'indiv.  $i$ .

→  $\delta_i$  : terme d'erreur du modèle courant à la sortie pr l'indiv.  $i$ .

Ces deux termes vérifient les équations dites de rétropropagation de l'erreur. On pose

$$s_{ki} = f'(\alpha_k^T x_i) \beta_k \delta_i$$

⇒ Pour estimer les valeurs des gradients, on a donc besoin d'évaluer  $\delta_i$  et  $s_{ki}$ .

Cela se fait en 2 étapes :

- 1 une passe avant : valeurs courantes des poids permet de déterminer  $\hat{\Phi}(x_i)$  ;
- 2 puis une passe retour : avec  $\hat{\Phi}(x_i)$ , on peut évaluer  $\delta_i$  (en estimant les  $\beta$  par minimisation de la 1<sup>ère</sup> équation) puis  $s_{ki}$  par rétropropagation des  $\delta_i$ ...

## ALGORITHMES D'OPTIMISATION

Sait évaluer les gradients  $\Rightarrow$  reste à utiliser un algo adapté !

+ **simple** : utilisation itérative du gradient (e.g. Newton-Raphson) :  
en tout point de l'espace des paramètres, le vecteur gradient de  $Q$   
pointe dans la direction de l'erreur croissante  $\Rightarrow$  suffit de se  
déplacer dans le sens opposé pour  $\searrow Q$  ! Ainsi,

$$\begin{aligned}\beta_k^{(r+1)} &= \beta_k^{(r)} - \tau \sum_i \frac{\partial Q_i}{\partial \beta_k^{(r)}} \\ \alpha_{kj}^{(r+1)} &= \alpha_{kj}^{(r)} - \tau \sum_i \frac{\partial Q_i}{\partial \alpha_{kj}^{(r)}}\end{aligned}$$

$\tau$  : **taux d'apprentissage** (schéma minimisation  $f$  convexe).

# APPLICATION : ALGORITHME DE RÉTROPROPAGATION ÉLÉMENTAIRE DU GRADIENT

## Initialisation :

Tirage aléatoire uniforme sur  $[0, 1]$  pour les poids  $\alpha_{jkl}$  (normaliser dans  $[0, 1]$  les données d'apprentissage).

## Boucle :

- Tant que  $(Q > erreurMax)$  ou  $(niter < niterMax)$ , faire
  - ranger la base d'apprentissage dans un nouvel ordre aléatoire,
  - pour chaque indiv.  $i = 1, \dots, n$ , faire
    - calculer  $\epsilon(i) = y_i - \Phi(x_i^1, \dots, x_i^p; (\alpha)(i-1))$  en propageant les entrées vers l'avant ;
    - l'erreur est rétropropagée dans les  $\neq$  couches pour affecter à chaque entrée une "responsabilité" dans l'erreur globale ;
    - mise à jour de chaque poids  $\alpha_{jkl}(i) = \alpha_{jkl}(i-1) + \Delta\alpha_{jkl}(i)$ .

## 7 Réseau de neurones et Deep Learning

- Introduction
- Neurone formel et fonctionnement d'un perceptron multicouches
- Estimation des paramètres
- Paramétrage du réseau
- Deep learning et autres types de réseaux

## REMARQUE SUR LE TAUX D'APPRENTISSAGE

Le taux d'apprentissage (learning rate) est un paramètre de tuning.

Il peut

- soit être fixé par l'utilisateur au début de l'algorithme ;
- soit varier en cours d'exécution.

Si  $\tau$  est grand, alors on converge + vite vers une solution, mais elle est moins précise.

Et inversement.

# PARAMÈTRES D'UN RÉSEAU DE NEURONES

Si on récapitule, on doit spécifier/ déterminer...

- ① ...variables d'entrée et de sortie (leur faire subir d'éventuelles transformation de normalisation) ;
- ② ...architecture du réseau :
  - nb de couches cachées : aptitude à traiter des non-linéarités ;
  - nb de neurones par couche cachée ;⇒ Impacte le nb de param. à estimer !
- ③ ...3 autres paramètres : erreur max. tolérée, nb d'itérations max. de l'algo, un terme éventuel de **régularisation** ("decay", à intégrer dans la fonction de coût ⇒ Ridge, norme 2 des poids) ;
- ④ **taux d'apprentissage**  $\tau$ .

## COMPLEXITÉ D'UN RÉSEAU DE NEURONES

Les 2 choix sur le nombre de couches cachées, et le nombre de neurones par couche cachée, jouent sur la complexité du réseau.

⇒ Donc ces choix jouent sur la recherche du meilleur compromis biais-variance de l'estimateur par réseau neuronal...

⇒ Jouent donc également sur l'arbitrage qualité d'adéquation / qualité prédictive.

**En pratique**, on ne règle pas simultanément ces paramètres : on cherche à contrôler le phénomène de surapprentissage ⇒ on fera des échantillons bootstrap, ou des validations croisées, ou échantillon test, pour estimer l'erreur.

## RÉGLAGES

Pour ce qui concerne...

- ...la **durée d'apprentissage** (maxit dans R) : arrêter par ex l'apprentissage lorsque l'erreur de validation réaugmente.
- ...le **nb de couches** : d'après le théo. d'approx. univ., on peut se restreindre à un petit nb de couches cachées (1 ou 2 max.).
- ...le **nb de neurones** par couche cachée : minimiser l'estimation de l'erreur de prévision par validations croisées par exemple.

**Conclusion** : à chaque architecture spécifiée correspond un réseau de neurones optimal. On fait varier ensuite les param. : on choisit au final l'optimal des optimaux (comme CART avec l'élégage).

## 7 Réseau de neurones et Deep Learning

- Introduction
- Neurone formel et fonctionnement d'un perceptron multicouches
- Estimation des paramètres
- Paramétrage du réseau
- Deep learning et autres types de réseaux

## LE DEEP LEARNING ?

Le **Deep Learning** n'est rien d'autre qu'un réseau de neurones ultra complexe.

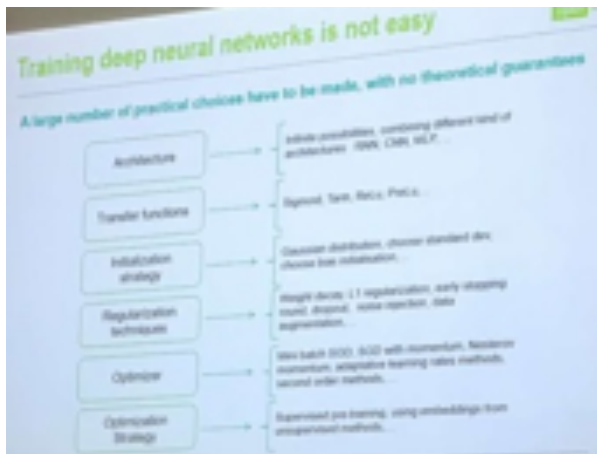
Il y a

- énormément de couches,
- et énormément de neurones.

Cela implique des **millions de paramètres** potentiels, et ne peut se calibrer qu'en cas de données gigantesque...

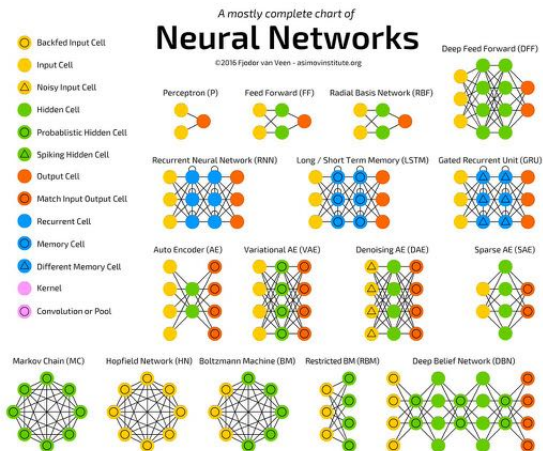
# CONCLUSION SUR LE DEEP LEARNING

Comme on peut s'en douter, il n'est en fait pas du tout facile de bien se servir d'un réseau Deep Learning...



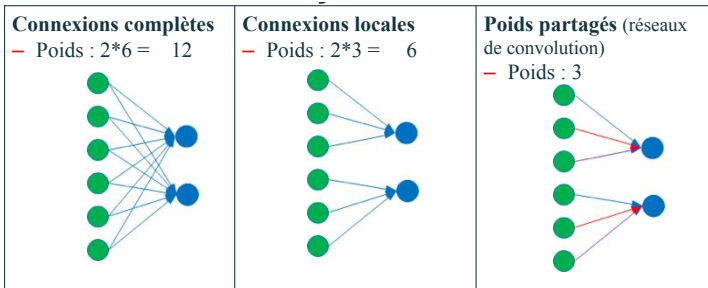
# TYPES DE RESEAUX NEURONAUX

∃ multitude de types de réseaux, avec des caractéristiques  $\neq$  :  
boucles de rétro-apprentissage, ...



## TYPES DE CONNEXION

∃ aussi une multitude de types de connexions :



# **CONCLUSION GENERALE**

Some characteristics of different learning methods.

Key: ● = good, ● = fair, and ● = poor.

Characteristic	Neural Nets	SVM	CART	GAM	KNN, Kernel	Gradient Boost
Natural handling of data of "mixed" type	●	●	●	●	●	●
Handling of missing values	●	●	●	●	●	●
Robustness to outliers in input space	●	●	●	●	●	●
Insensitive to monotone transformations of inputs	●	●	●	●	●	●
Computational scalability (large $N$ )	●	●	●	●	●	●
Ability to deal with irrelevant inputs	●	●	●	●	●	●
Ability to extract linear combinations of features	●	●	●	●	●	●
Interpretability	●	●	●	●	●	●
Predictive power	●	●	●	●	●	●

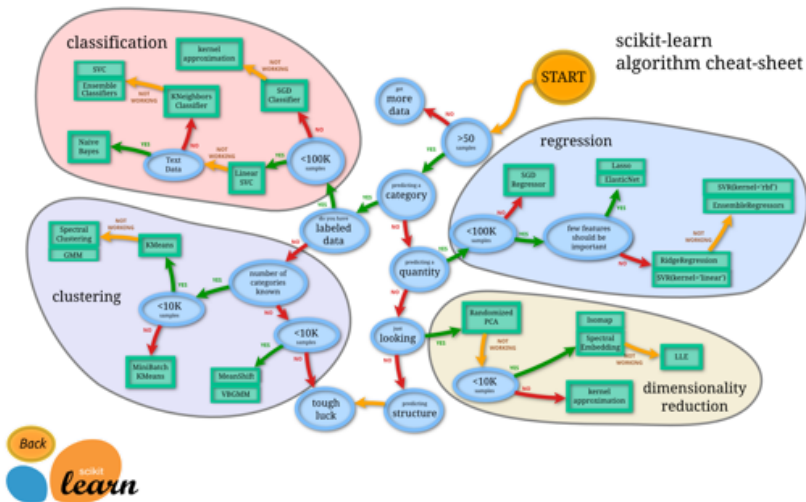
On peut aussi mentionner les points suivants, caractéristiques des méthodes d'apprentissage statistiques :

- + non-paramétrique,
- + peu d'hypothèses,
- + “data-driven”,
- + faible biais normalement,
- instabilité (potentielle large variance),
- gestion du surapprentissage,
- ressources informatiques,
- interprétabilité.

Notions-clefs retenus du cours ?

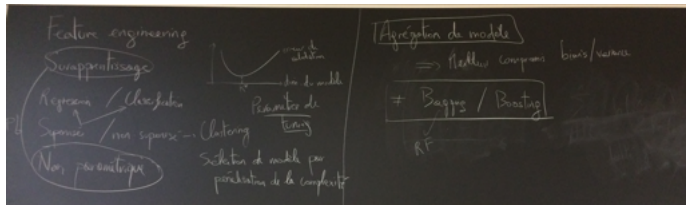
Retours sur le cours ? (contenu, TP, ...)

# QUELS MODELES POUR QUELLES APPLICATIONS ?



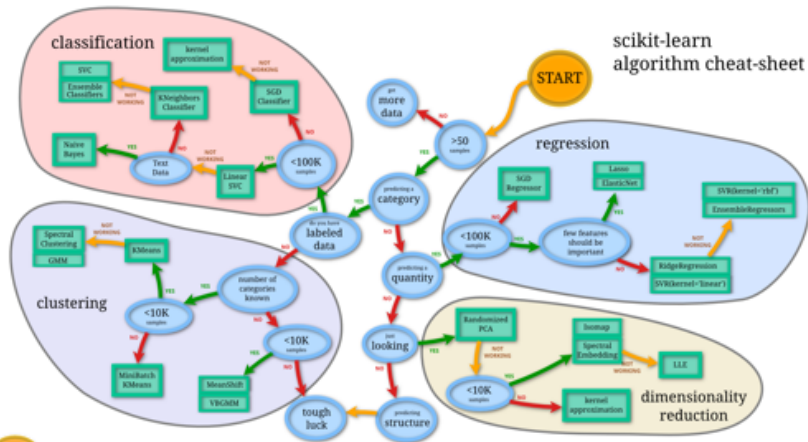
# NOTIONS PHARES DU COURS

Videos Deep Learning (cf dossier mac mes videos)



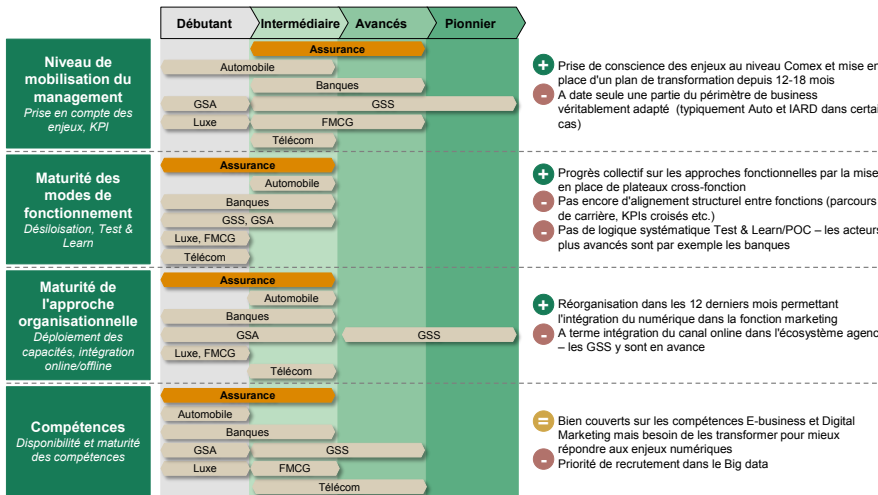
- $\rightarrow$  Contenu du cours
- $\rightarrow$  Mode d'enseignement
- $\rightarrow$  TP?
- $\rightarrow$  Enchaînement des séances (logos?)
- $\rightarrow$  Mode d'évaluation? Projet/Exam?

# scikit-learn algorithm cheat-sheet



# **ANNEXES**

# POSITION DES ASSUREURS



Source: entretiens, analyse BCG

# COMPÉTENCES BIG DATA

## Les compétences

1

### E-business

Activités de vente sur internet  
(web, site propre, sites tiers)

2

### Digital Customer Experience

Design des interfaces  
et parcours digitaux

3

### Digital Branding, Marketing

Activités marketing liées aux  
canaux digitaux  
(web, réseaux sociaux)

4

### Digital content

Création de contenu, de nouveaux  
produits/ services digitaux,  
digitalisation de produits/services  
existants

## Activités spécifiques

- E-commerce
- E-merchandising and site optimization
- Omnichannel/Multi-channel strategy

- UX designer / ergonomes
- Web developers

- Social media marketing (community mgr / E-reputation / Advocacy Marketing)
- Traffic acquisition (SEO, SEM, emailing, comparators, partnerships, affiliates)
- Digital branding (display, video)
- Programmatic / Real Time Bidding
- E-CRM

- Digital product or service manager
- Web / App Editor
- Digital Innovation / new digital product conception

## Les compétences

5

### Big Data & Analytics

Collecte, analyse  
et exploitation des données

6

### Mobile interfaces

Ensemble des interfaces propres  
aux canaux Smartphones et  
tablettes

7

### Digital tools

Développement et maintenance  
des outils et logiciels digitaux  
permettant la transformation  
numérique en interne comme en  
externe

8

### Digital support

Ressources en support  
des activités numériques

## Activités spécifiques

- Data scientist
- Web Analytics
- Data quality
- Business Intelligence

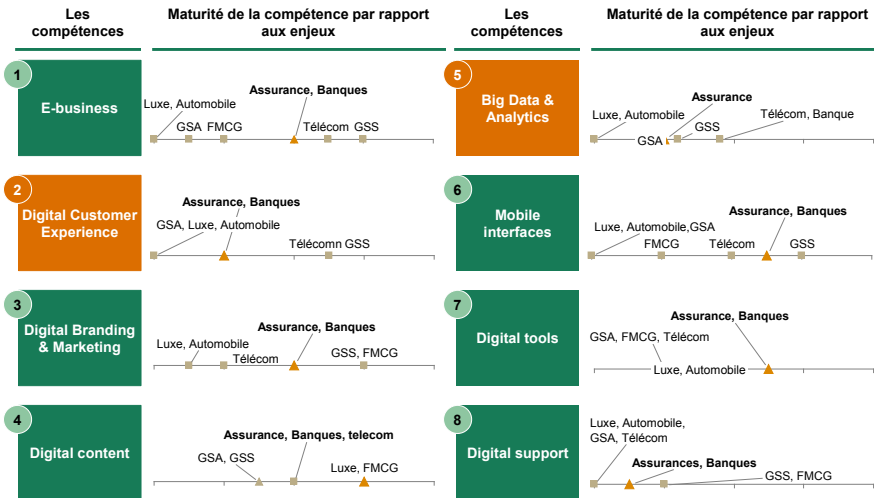
- Mobile app / msite developer
- Mobile UX
- Mobile data / geolocalisation

- Data technology (Hadoop, ...)
- E-CRM (Neolane, Unica, ...)
- Digital Front-Ends (Salesforce, ...)
- Cloud
- Digital security

- Digital Recruiting
- Digital legal
- Digital purchasing

Source: entretiens, analyse BCG

# ASSUREURS SUR CES COMPÉTENCES ?



Source: entretiens, analyse BCG

# Quelques articles de référence I



Y. Benjamini and Y. Hochberg.

Controlling the false discovery rate : a practical and powerful approach to multiple testing.

*Journal of the Royal Statistical Society, Series B*, 57 :289–300, 1995.



Leo Breiman.

Bagging predictors.

Technical report, UC Berkeley, 1994.



J.H. Friedman and P. Hall.

On bagging and nonlinear estimation.

pages –, 2000.



J. Friedman.

Greedy function approximation : A gradient boosting machine.

*The Annals of Statistics*, 29 :119–139, 2001.

# Quelques articles de référence II



Y. Freund and R.E. Shapire.

A decision-theoretic generalization of on-line learning and an application to boosting.

*J. Comput. Syst. Sci.*, 55(1) :119–139, 1997.



Olivier Lopez, Xavier Milhaud, and Pierre-Emmanuel Therond.

Tree-based censored regression with applications in insurance.

*Electron. J. Stat.*, 10 :2685–2716, 2016.



R.E. Shapire and Y. Freund.

*Boosting.*

MIT Press, Cambridge, 1st edition, 2012.



R.E. Shapire.

The boosting approach to machine learning : An overview.

Technical report, 2003.