From Data Science to Machine Learning

De l'analyse de données à l'apprentissage automatique

Jean-Yves Ramel

ramel@univ-tours.fr

Laboratoire d'Informatique Fondamentale
et Appliquée de Tours

LABORATOIRE D'INFORMATIQUE FONDAMENTALE ET APPLIQUÉE DE TOURS

Prenons un exemple…

Prédiction de l'infarctus du myocarde

On sait que la probabilité de développer un infarctus du myocarde (IM) augmente avec l'âge et avec le taux de cholestérol LDL.

- Comment développer un programme qui prédise le risque d'IM à partir de ces deux variables ?

Exemple tiré et adapté de [Denoeux2018]

Approche à base de règles (système expert)

Prédiction de l'infarctus du myocarde

On sait que la probabilité de développer un infarctus du myocarde (IM) augmente avec l'âge et avec le taux de cholestérol LDL.

- Comment développer un programme qui prédise le risque d'IM à partir de ces deux variables ?

- **Approche de type « système expert » :**
  - Modéliser la connaissance d'un médecin par des règles de la forme :
  - SI âge > 60 ET ldl > 10 ALORS risque élevé
  - SI 50 < âge ET ldl > 8 ALORS risque moyen
  - ...

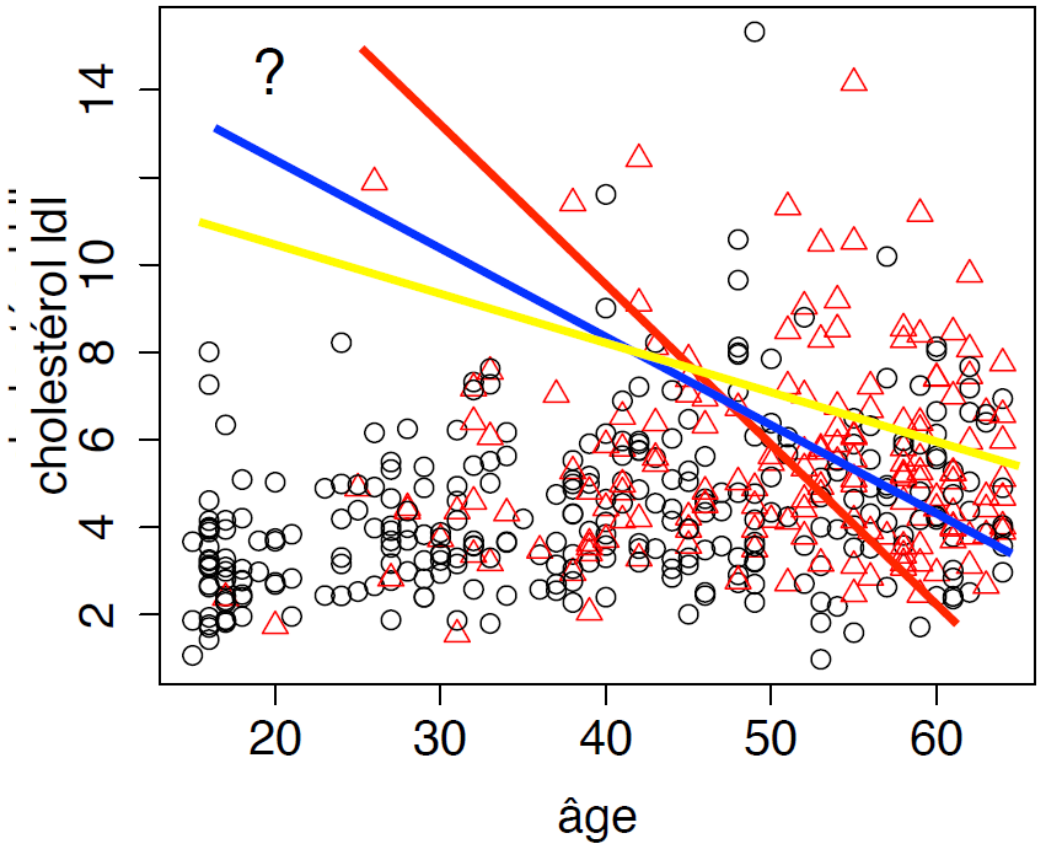
- Boite blanche → Mais approche difficile à mettre en oeuvrelorsque le nombre de variables explicatives devient important.

Approche par apprentissage

Approche de type ML ➔ Premier modèle : Régression logistique

Préparation des données

- Constituer une base d'apprentissage
- Comment distinguer / séparer au mieux les deux classes ?
- Trouver la frontière
- Modèle linéaire :
  la droite séparatrice optimale ?



Approche par apprentissage

Approche de type ML ➔ Premier modèle : Régression logistique

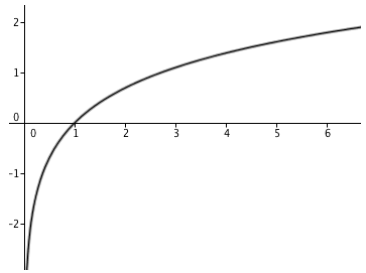
- On ne peut pas prédire à coup sûr l'occurence d'un IM à partir de l'âge et du taux de cholestérol, mais on peut chercher à estimer sa probabilité

  notée p(x) = $\mathbb{P}(\underbrace{\text{IM}}_{y=1} \mid \underbrace{\text{âge, Idl}}_{\mathbf{x}})$

- Modèle linéaire, on pose : $\ln \frac{p(\mathbf{x})}{1-p(\mathbf{x})} = w_0 + w_1 \times \text{âge} + w_2 \times \text{Idl}$

- Formulation équivalente : $p(\mathbf{x}) = \frac{1}{1+\exp(-w_0 - w_1 \times \text{âge} - w_2 \times \text{Idl})}$

- Problème : comment déterminer les coefficients w0, w1 et w2 ?
  - Si y = 1 (IM avéré), on veut avoir p(x) aussi grand que possible.
    On définit l'erreur dans ce cas par **-** *ln (p(x))* ➔ erreur grande qd p(x) est proche de 0
  - Symétriquement, si y = 0 (pas d'IM), on veut avoir p(x) aussi petit que possible.
    L'erreur est alors **-** *ln (1 - p(x))* ➔ erreur grande qd p(x) est proche de 1
  - Formule générale : $\text{erreur} = -y \ln p(\mathbf{x}) - (1-y) \ln (1-p(\mathbf{x}))$
  - Erreur totale pour un ensemble d'apprentissage {(x1; y1), … ,(xn; yn) } mesurée par l'**entropie-croisée** : $C(\underbrace{w_0, w_1, w_2}_{\mathbf{w}}) = \sum_{i=1}^{n} \text{erreur}_i$

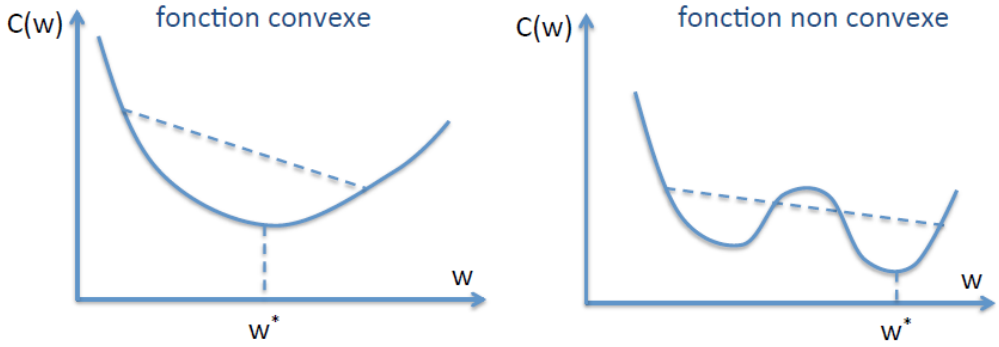


Approche par apprentissage

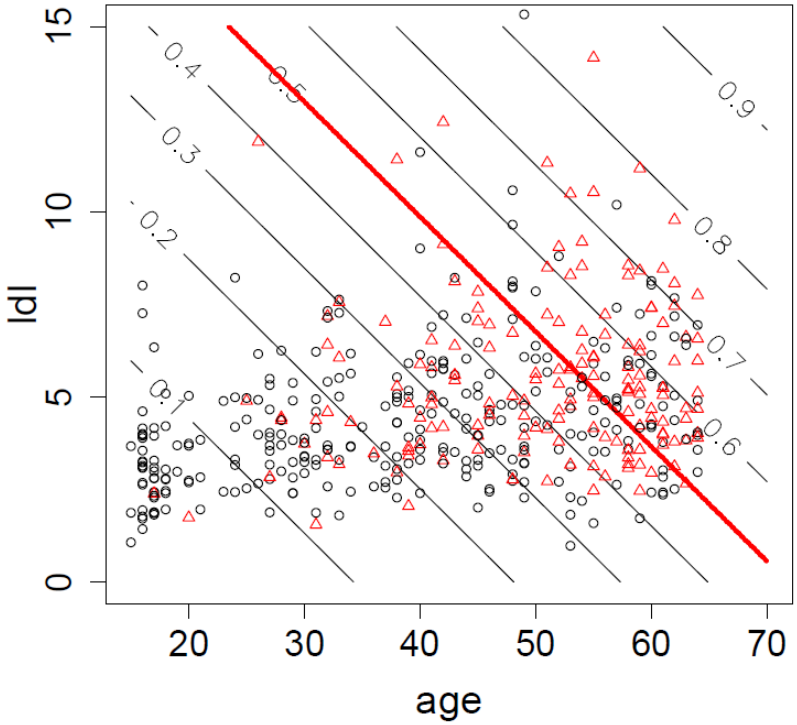
Approche de type ML ➔ Premier modèle : Régression logistique

Apprentissage

- Une fois définie une fonction d'erreur, le problème de l'apprentissage devient un problème d'optimisation : rechercher le vecteur de coefficient w* qui minimise l'erreur.
- Dans le cas de la régression logistique, ce vecteur est unique car la fonction d'erreur est convexe.



- Le vecteur solution w* peut être obtenu par un algorithme itératif.



Approche par apprentissage

Approche de type ML ➔ Premier modèle : Régression logistique

Exploitation

- Une fois déterminé le vecteur de coefficient w optimal, on dispose d'un **programme classifieur permettant de classer de nouveaux individus**.

Evaluation des performances

- Pour estimer la probabilité d'erreur du classifieur, il faut disposer d'un ensemble de test indépendant (**base de test**).
- On construit alors la **matrice de confusion** pour cet ensemble
- Avec 100 exemples

		Vraie classe	
		Positif	Négatif
Prediction	Positif	14	10
	Négatif	21	55

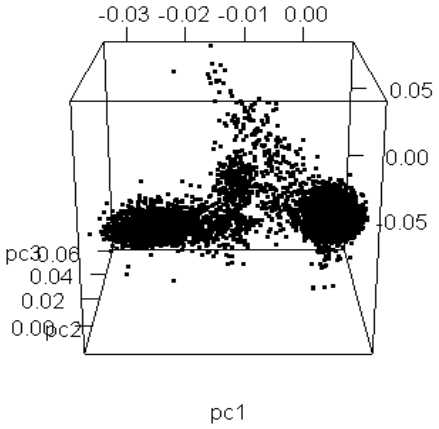
- Taux d'erreur = (10+21)/100=31%.

2 – Les Principaux modèles

RAPPEL ➔ Selon les données disponibles et les objectifs

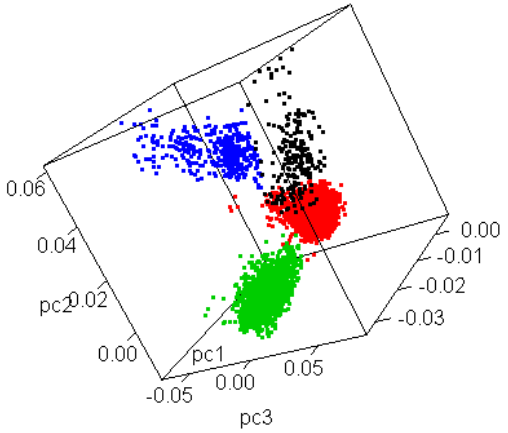
Apprentissage non supervisé

$\mathbf{Voxel_1} = [F_1\ F_2\ F_3\ \ldots]$
$\mathbf{Voxel_2} = [F_1\ F_2\ F_3\ \ldots]$
$\mathbf{Voxel_3} = [F_1\ F_2\ F_3\ \ldots]$
$\mathbf{Voxel_4} = [F_1\ F_2\ F_3\ \ldots]$
$\mathbf{Voxel_5} = [F_1\ F_2\ F_3\ \ldots]$
.....
$\mathbf{G} = [F_1\ F_2\ F_3\ \ldots]$



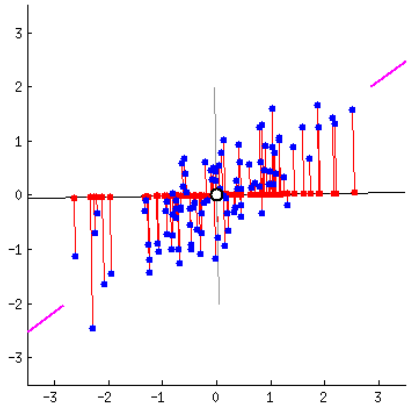
Apprentissage supervisé

$\mathbf{Voxel_1} = [F_1\ F_2\ F_3\ \ldots]$ [Tumor]
$\mathbf{Voxel_2} = [F_1\ F_2\ F_3\ \ldots]$ [Normal]
$\mathbf{Voxel_3} = [F_1\ F_2\ F_3\ \ldots]$ [Tumor]
$\mathbf{Voxel_4} = [F_1\ F_2\ F_3\ \ldots]$ [Tumor]
$\mathbf{Voxel_5} = [F_1\ F_2\ F_3\ \ldots]$ [Normal]
$\mathbf{G_G} = [F_1\ F_2\ F_3\ \ldots]$ ↑
$\mathbf{G_R} = [F_1\ F_2\ F_3\ \ldots]$ **Label**



Régression

$\mathbf{Voxel_1} = [F_1]$
$\mathbf{Voxel_2} = [F_1]$
$\mathbf{Voxel_3} = [F_1]$
$\mathbf{Voxel_4} = [F_1]$
$\mathbf{Voxel_5} = [F_1]$
.....
$\mathbf{G} = [F_1]$

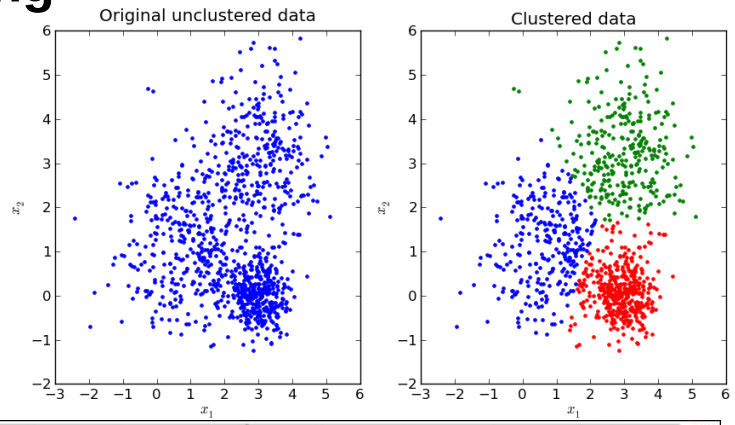


2 – Les Principaux modèles

Unsupervised classification : k-means clustering

- No tagged data available ⇨ learning impossible
- We look for k classes starting from k centers (G_i)

Objectif : minimising the intra-class variance



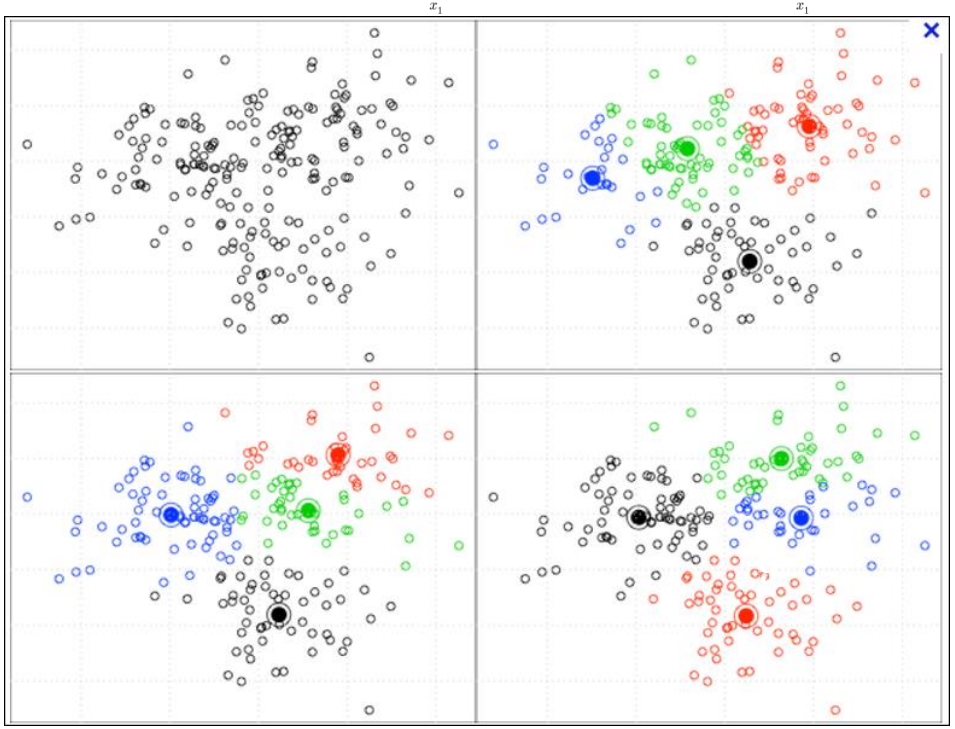
Algorithm:

1 - Choose K centers randomly

2 – Repeat :

a/ Allocate each x to the closest center Gi

b/ Compute the new Gi until stabilization



Dependent from initialization →

2 – Les Principaux modèles

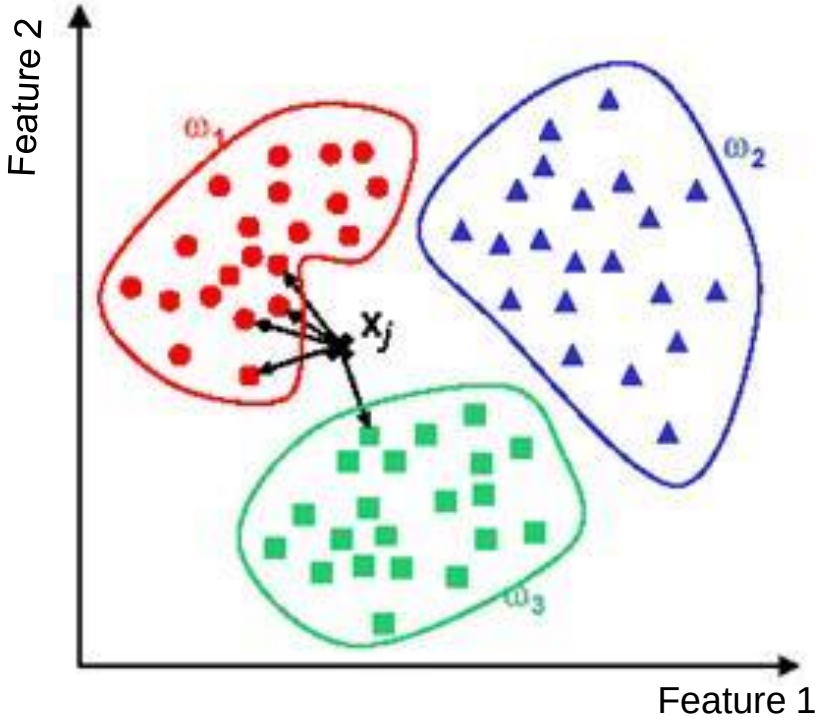
Clustering on images

Group together pixels by color, automatic segmentation: k-means, k = 2

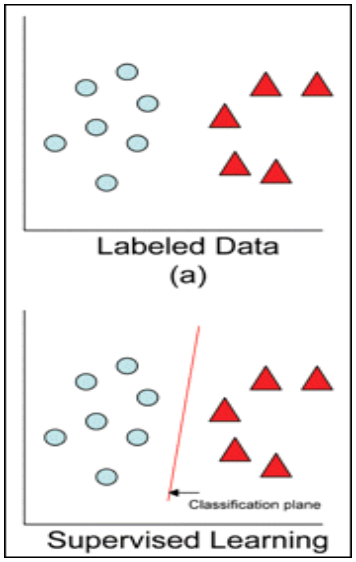
2 – Les Principaux modèles

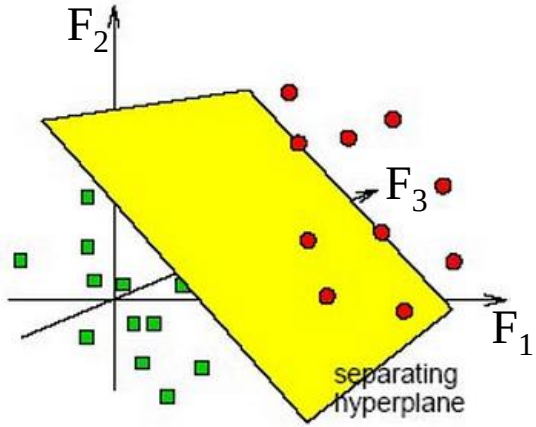
Supervised classification : k-Nearest-Neighbors (kNN)

- We have a training set with feature vectors tagged with the corresponding classes (w_i)
- The unknown vector X_j is classified with/inside the most represented class among its k nearest neighbors



Or other sophisticate techniques for classifier definition = ML





2 – Les Principaux modèles

Supervised classification : k-Nearest-Neighbors (kNN)

- We have a training set with feature vectors tagged with the corresponding classes (w_i)
- The unknown vector X_j is classified with/inside the most represented class among its k nearest neighbors

Algorithme 1 : Algorithme implémentant la règle des k-ppv

Données :
- *appbase* : base d'exemples de référence, contenant les valeurs des descripteurs
- *etiquettes* : étiquettes des exemples de la base de référence
- *individu* : exemple à classer
- k : nombre de voisins à prendre en compte

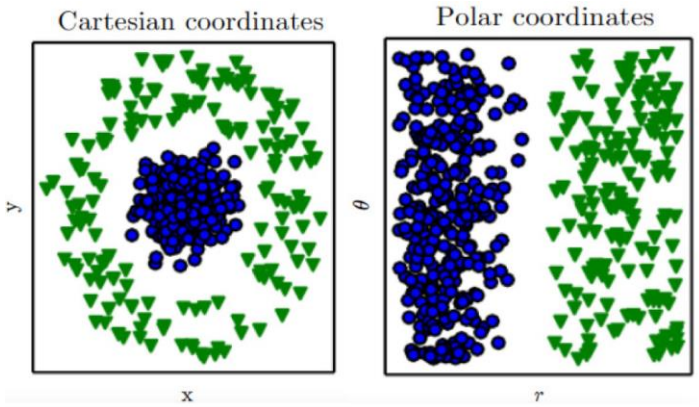
Résultat :
- *classe* : étiquette de la classe proposée

début
  pour *chaque individu* $I_r \in$ *dans appbase* **faire**
    dist[r] ← distance (individu, I_{app})
  end
  Trier conjointement (*dist*, *etiquettes*) sur la valeur de *dist* croissante
  classe ← étiquette majoritaire dans *etiquettes*$[0 : k]$
  retourner *classe*
end

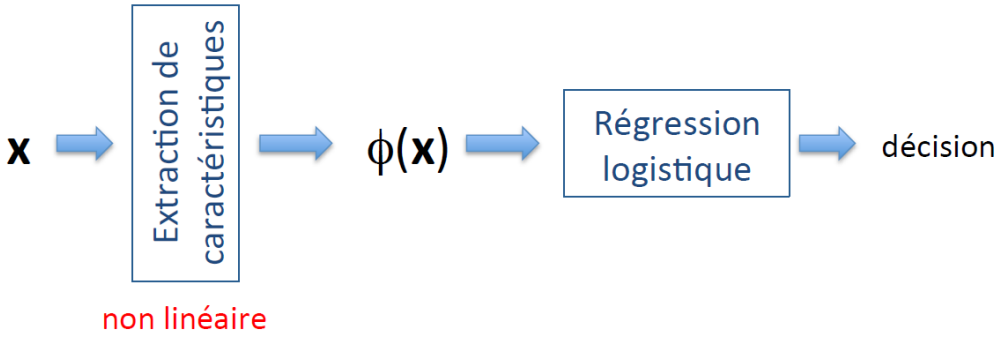
Problème du choix de l'espace de representation

➔ **Pour aller plus loin que les modèles linéaires…**

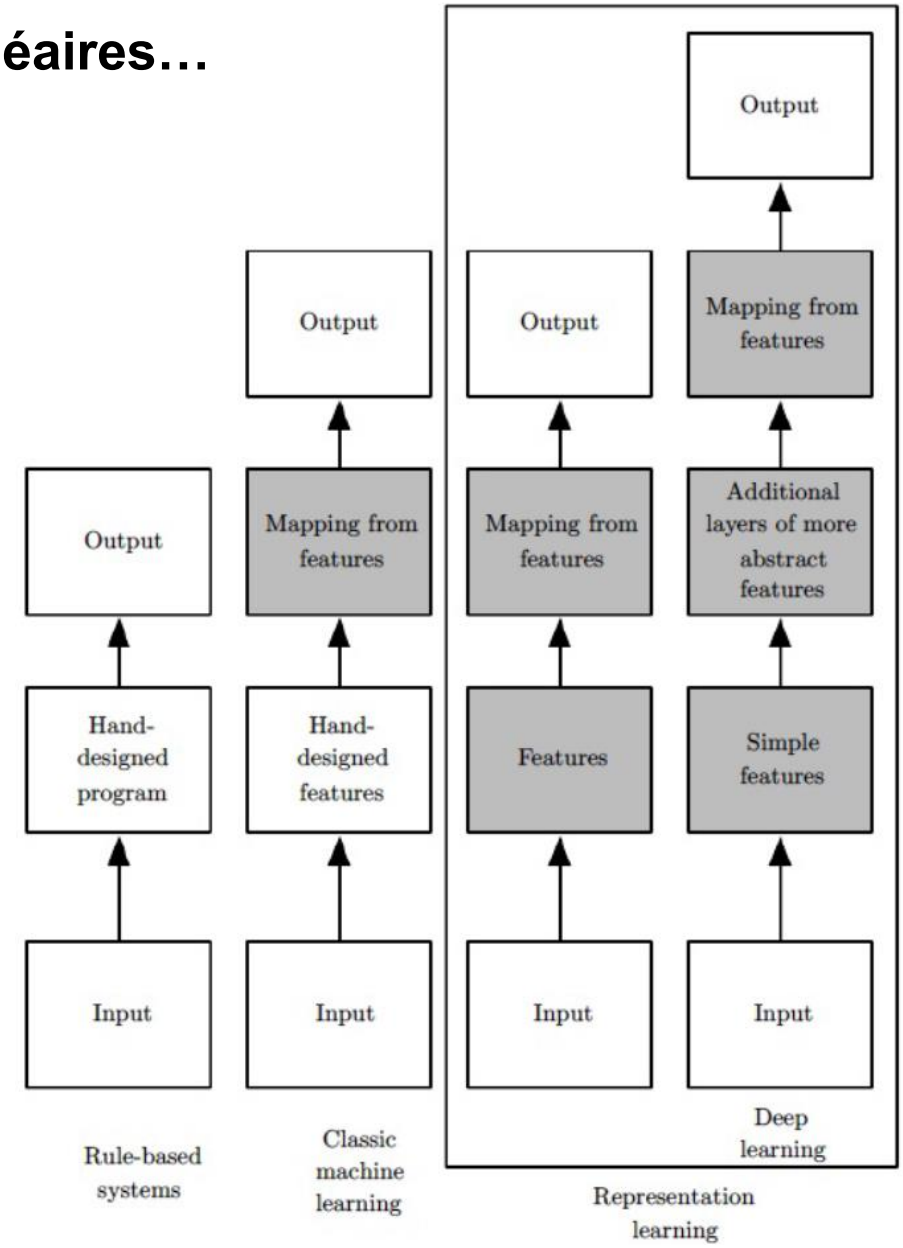
- Changement de representation



- Classifieur linéaire généralisé



- Architectures plus profondes

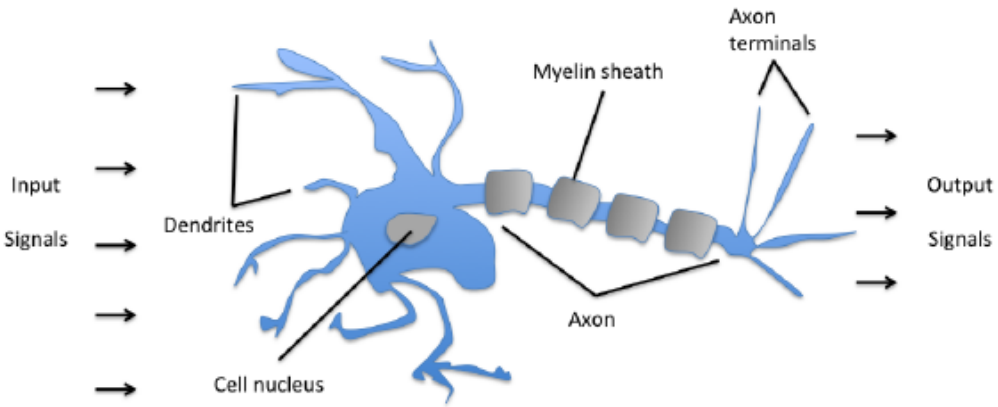


Approche par apprentissage

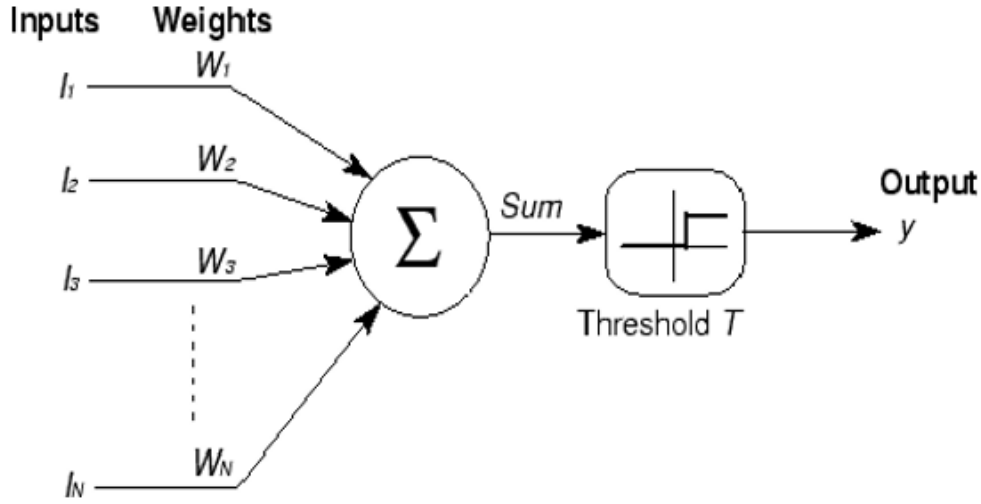
De la régression logistique aux réseaux de neurones

Le modèle de McCulloch et Pitts [McCulloch1943]

- Idée : neurones biologiques vus comme des portes logiques effectuant des opérations de la logique booléenne



Schematic of a biological neuron.

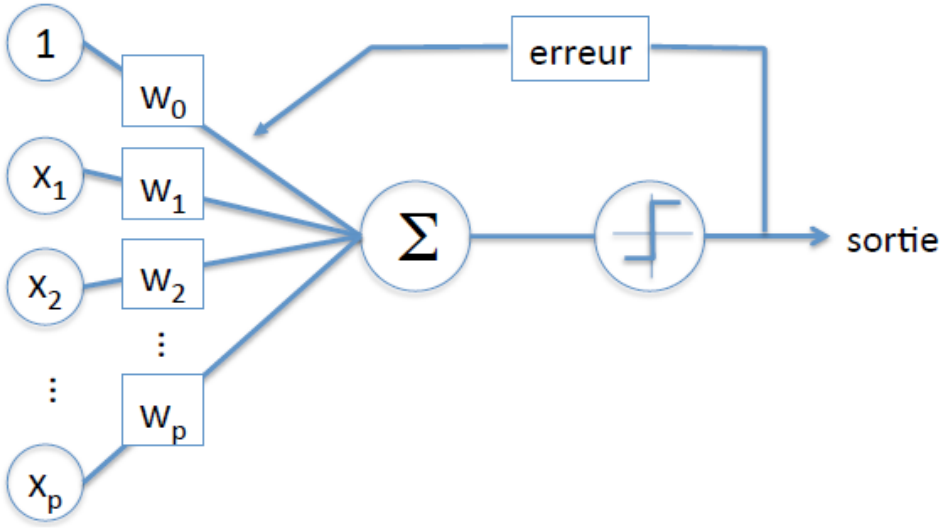


Approche par apprentissage

De la régression logistique aux réseaux de neurones

Le Perceptron [Rosenfeld1957]

- Idée : une algorithme qui apprend les poids pour résoudre des problèmes de classification binaire.



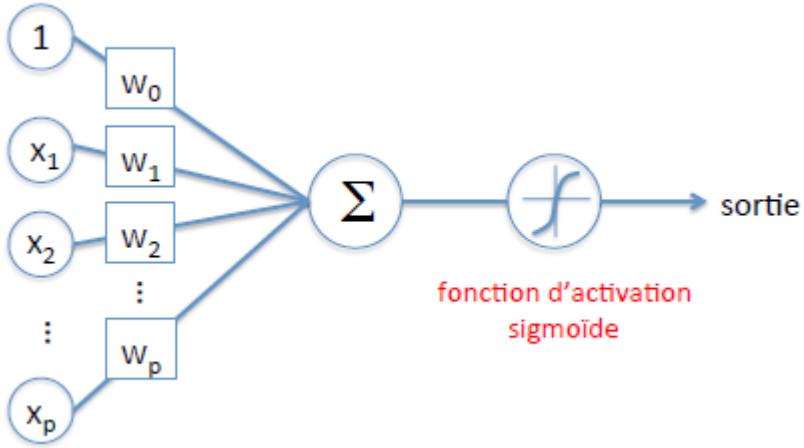
Limites :

- L'algorithme ne converge que si les deux classes sont bien séparées
- Difficilement généralisable à plus de deux classes

Approche par apprentissage

De la régression logistique aux réseaux de neurones

Version moderne du perceptron



Sortie :

$$g(\mathbf{x}) = \frac{1}{1 + \exp\left[-(w_0 + w_1 x_1 + \ldots w_p x_p)\right]}$$

Apprentissage des poids par minimisation de l'entropie croisée

C'est exactement le modèle de la régression logistique !

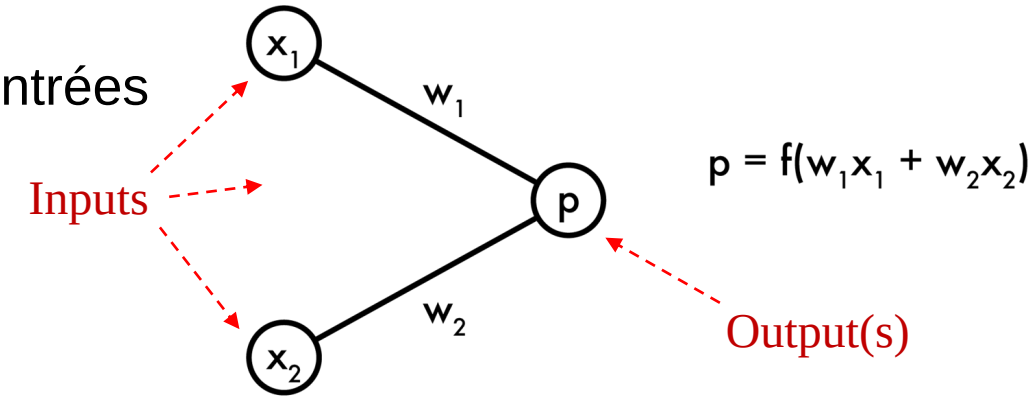
Outrepasser le problème de "feature engineering”

Le choix des caracteriistiques → de l’espace de représentation

- Il s’agit ici de la limitation principale du machine learning (surtout en pratique)
- ML fonctionne bien si il existe une relation claire entre les entrées du système (espace de représentation: x_i) et les sorties désirées
- La relation *p* est la fonction, le modéle que l’on cherche à apprendre

Avec les modèles linéaires

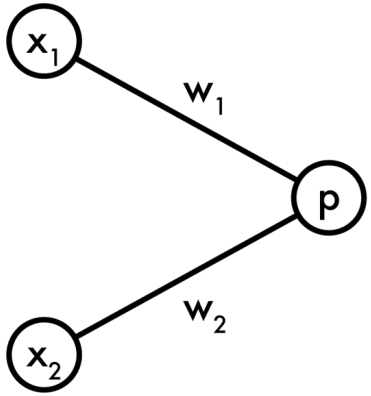
- Calcul d’une probabilité d’appartenance pour chaque classe à partir des valeurs prises par les feature choisies
- Sortie = combinaison linéaire des entrées
- Apprentissage des poids



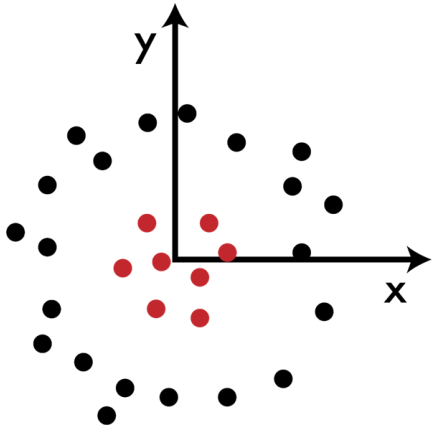
Outrepasser le problème de "feature engineering”

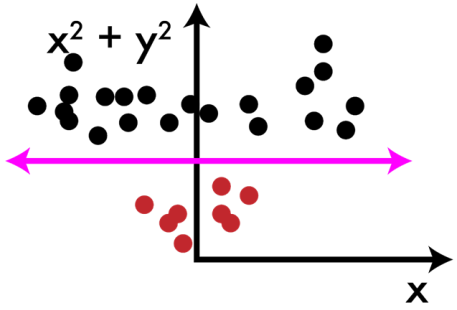
Avec d’autres modèles

- Même avec des modèles plus sophistiqués, on ne fait que proposer des combinaisons des caractéristiques choisies
- Impossible d'apprendre les transformations des caractéristiques
- L'homme doit choisir, créer de bonnes caractéristiques



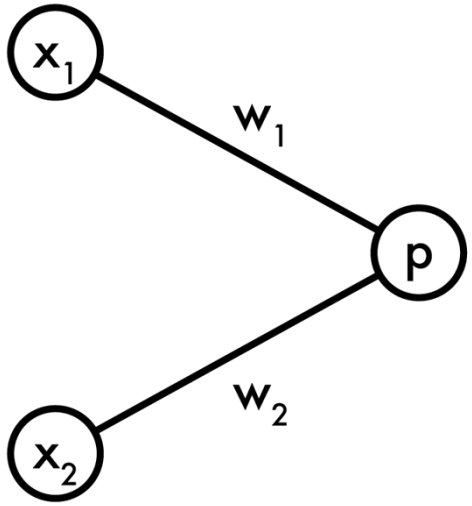
$$p = f(w_1x_1 + w_2x_2)$$





Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

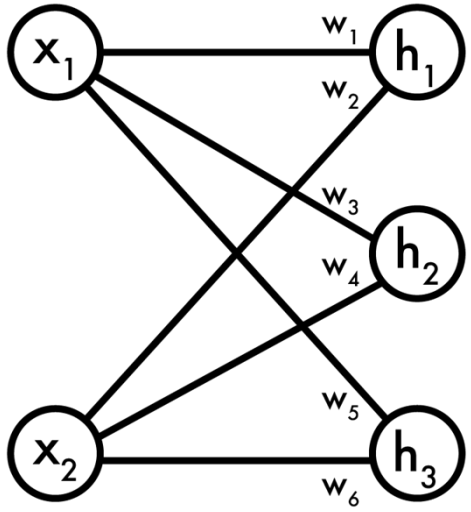


$$p = f(w_1x_1 + w_2x_2)$$

Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

- Créer de “nouvelles” caractéristiques à partir des précédentes
- Ajout d’une couche supplémentaire (cachée) nommée **H**

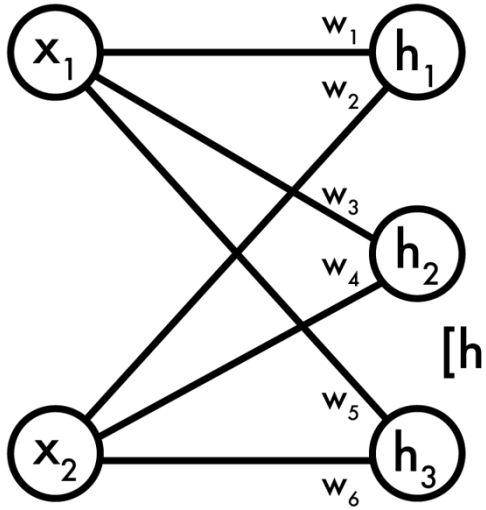


$h_1 = \varphi(w_1 x_1 + w_2 x_2)$
$h_2 = \varphi(w_3 x_1 + w_4 x_2)$
$h_3 = \varphi(w_5 x_1 + w_6 x_2)$

Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

- Comme le modele lineaire, **H** peut etre exprimé sous forme matricielle



$$h_1 = \varphi(w_1 x_1 + w_2 x_2)$$
$$h_2 = \varphi(w_3 x_1 + w_4 x_2)$$
$$h_3 = \varphi(w_5 x_1 + w_6 x_2)$$

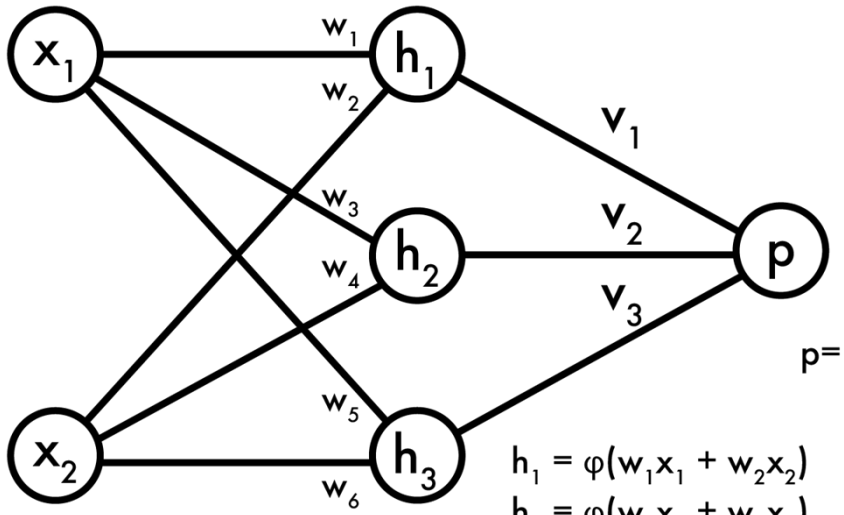
$$[h_1\ h_2\ h_3] = \varphi([x_1\ x_2]\begin{bmatrix} w_1 & w_3 & w_6 \\ w_2 & w_4 & w_5 \end{bmatrix})$$

$$\mathbf{H} = \varphi(\mathbf{Xw})$$

Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

- Maintenant, les prédictions p sont fonction de la couche cachée



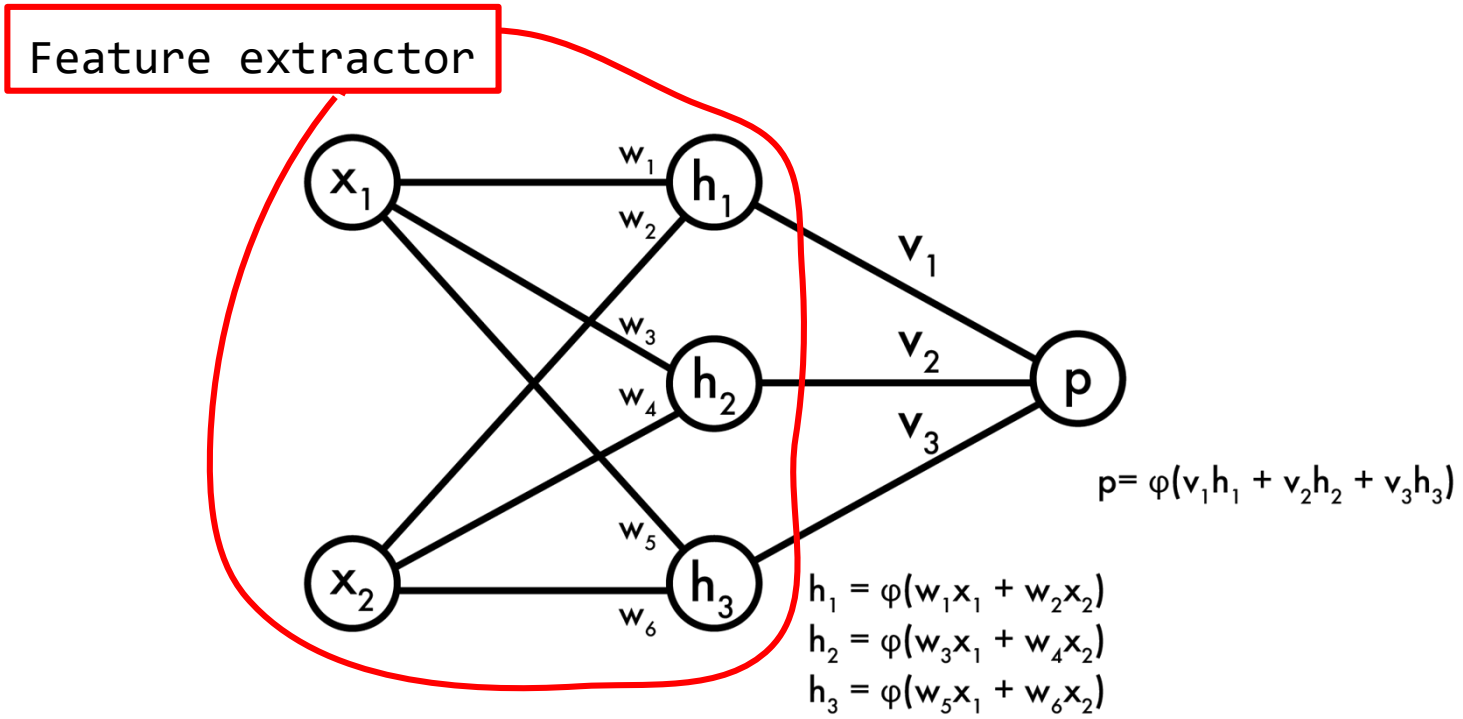
$p = \varphi(v_1 h_1 + v_2 h_2 + v_3 h_3)$

$h_1 = \varphi(w_1 x_1 + w_2 x_2)$
$h_2 = \varphi(w_3 x_1 + w_4 x_2)$
$h_3 = \varphi(w_5 x_1 + w_6 x_2)$

Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

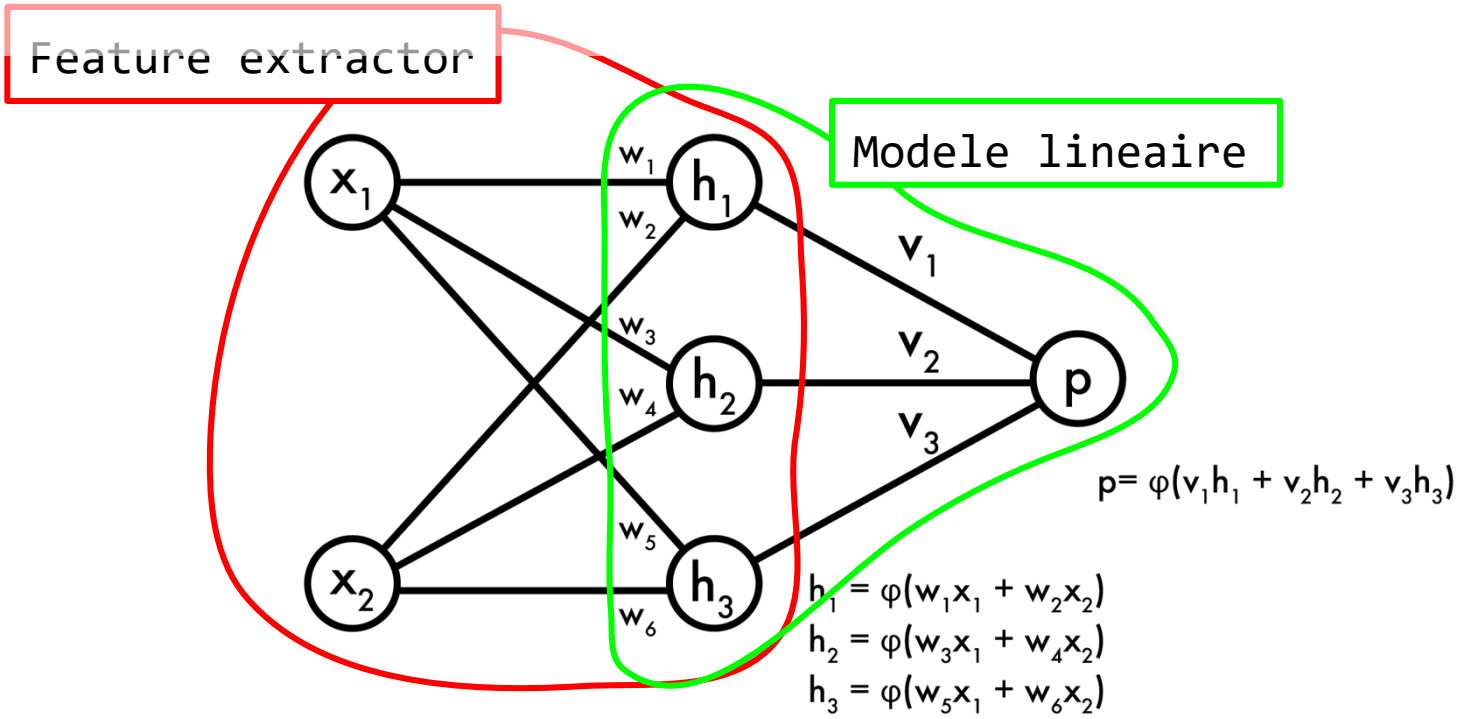
- Maintenant, les prédictions p sont fonction de la couche cachée



Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

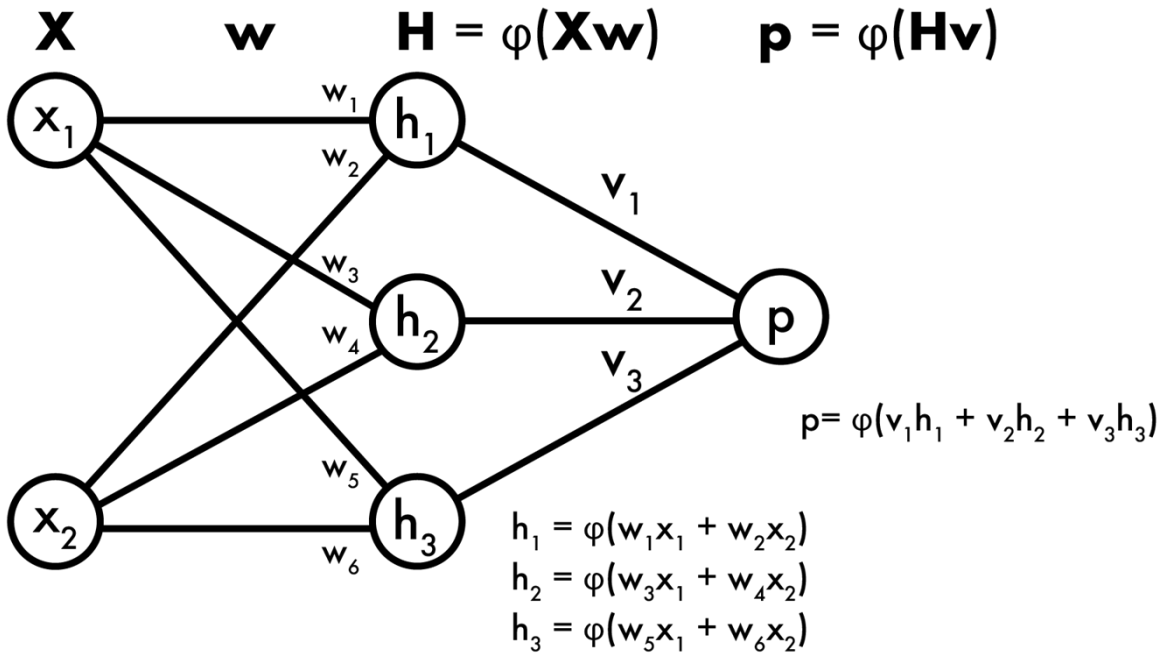
- Maintenant, les prédictions p sont fonction de la couche cachée



Outrepasser le problème de "feature engineering”

Et si on ajoutait des transformations?

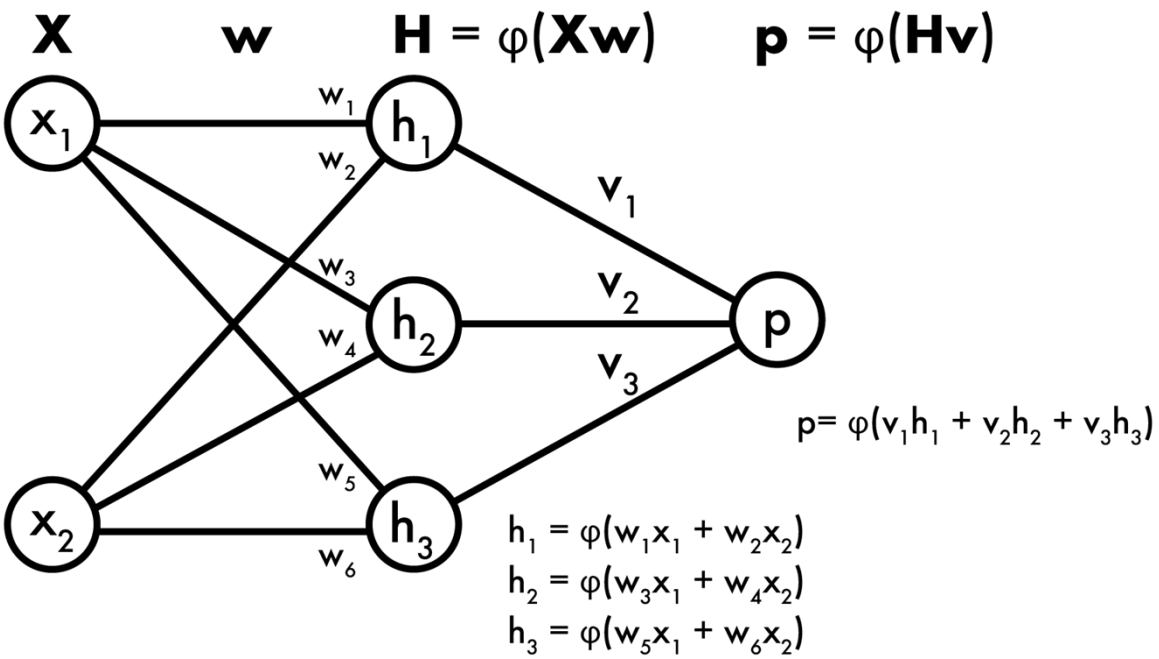
- L’ensemble du processus peut s’exprimer sous forme matricielle
- Très important pour des questions de temps de calcul



Outrepasser le problème de "feature engineering”

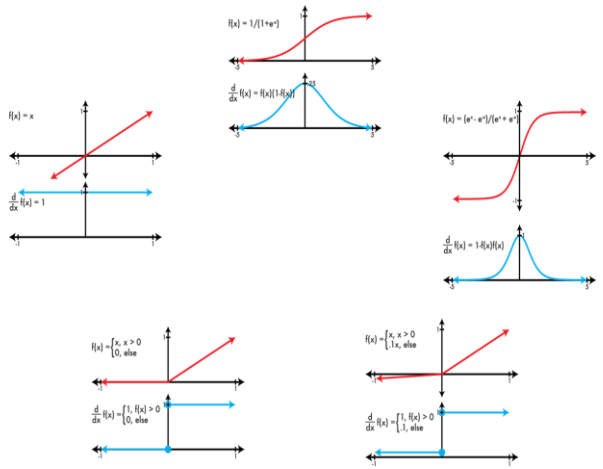
Ici commence à apparaitre la notion / question du Deep Learning

- Il est possible de multiplier le nombre de couches cachées
- Chaque couche correspondant à des fonctions φ appliquées aux combinaisons linéaires de la couche précédente



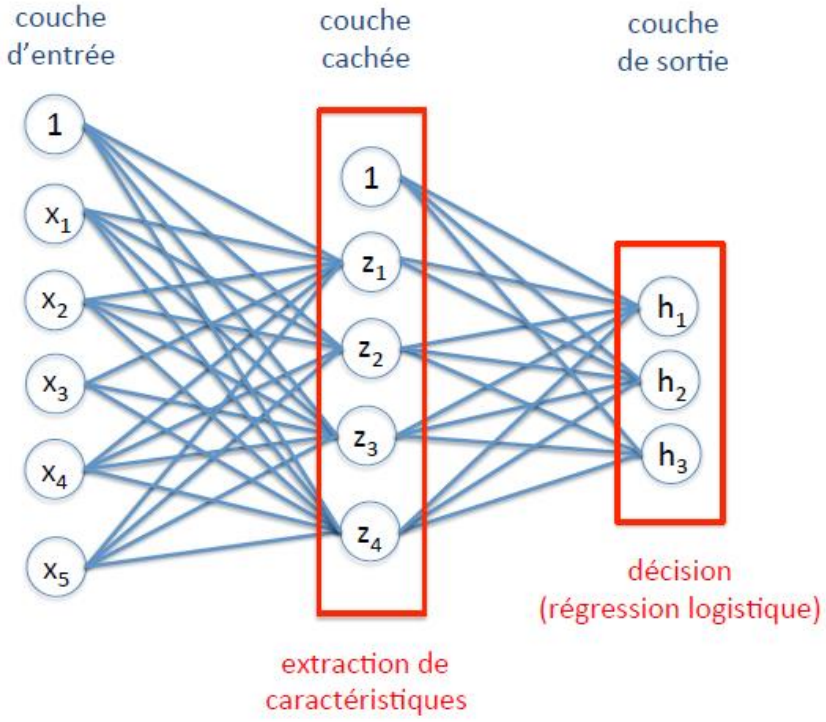
What about activation functions φ ?

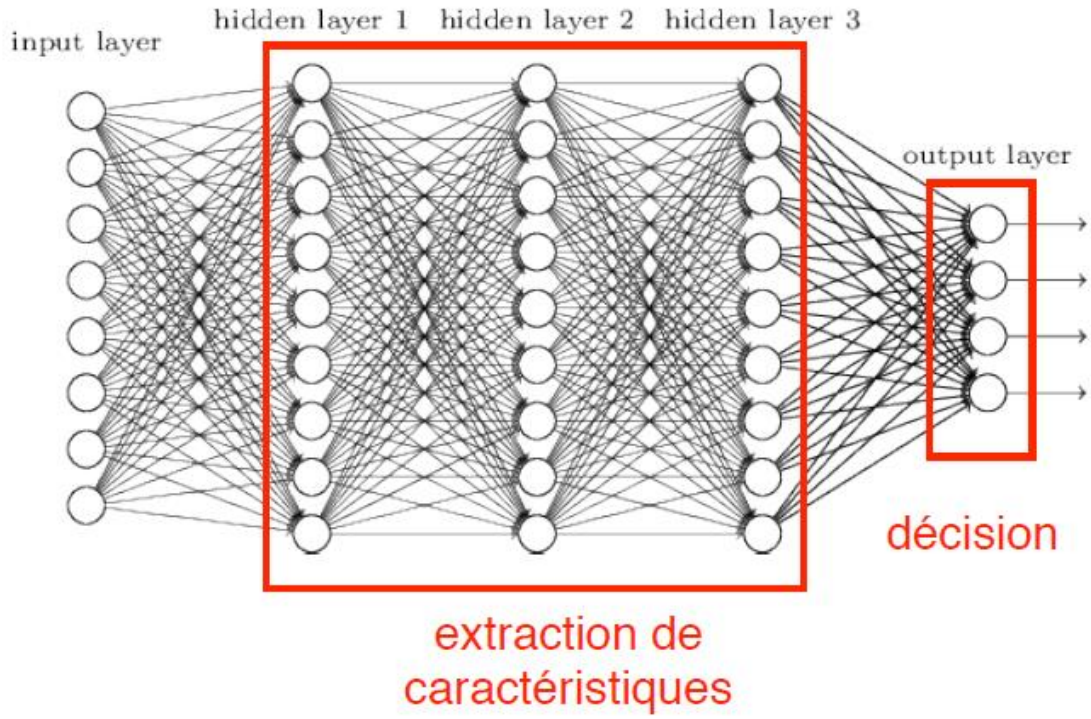
- So many possible options
- want them to be easy to take derivative



Outrepasser le problème de "feature engineering”

Vers les réseaux profonds





Comme pour la régression logistique, l’apprentissage des poids se fait en minimisant une fonction d’erreur telle que l’entropie-croisée.

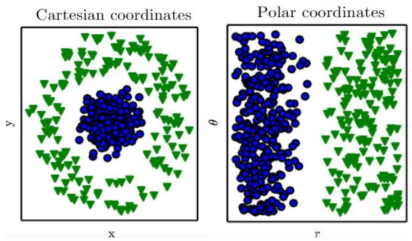
Problèmes: Très grand nombre de variables (poids du réseau) - Fonction d’erreur non-convexe - beaucoup de minima locaux

➔ Nécessité de grosses capacités de calcul

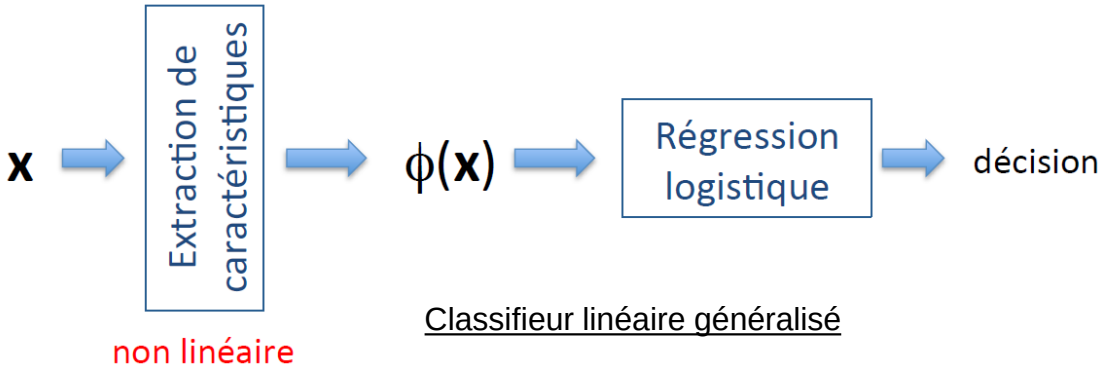
Outrepasser le problème de "feature engineering”

➔ Apprentissage de représentations

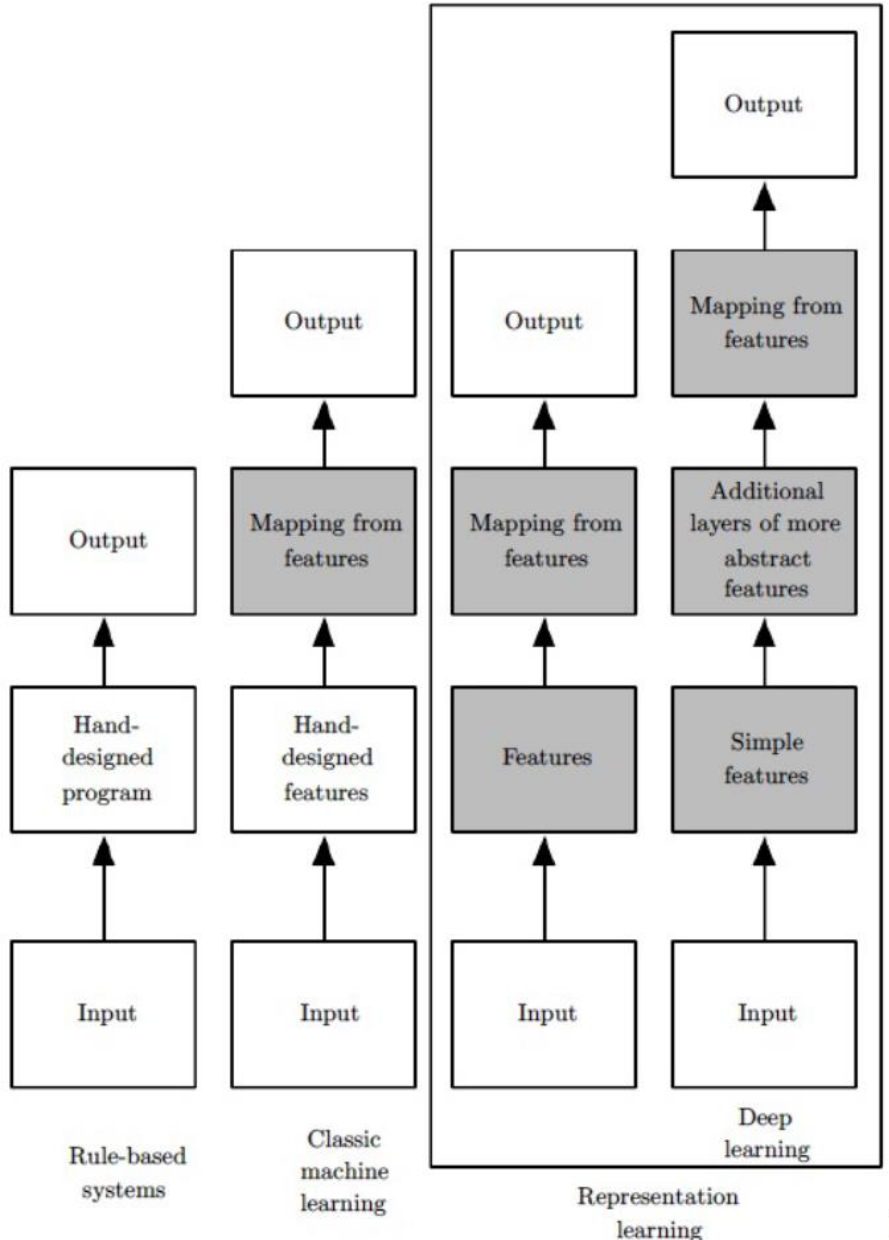
- Changement d’espace de représentation (en restant dans le domaine numérique)
  - Approche par noyau (pas d’apprentissage)



  - Approche Deep learning (apprentissage)



- Changement d’espace de représentation (en quittant le domaine numérique → graphes)



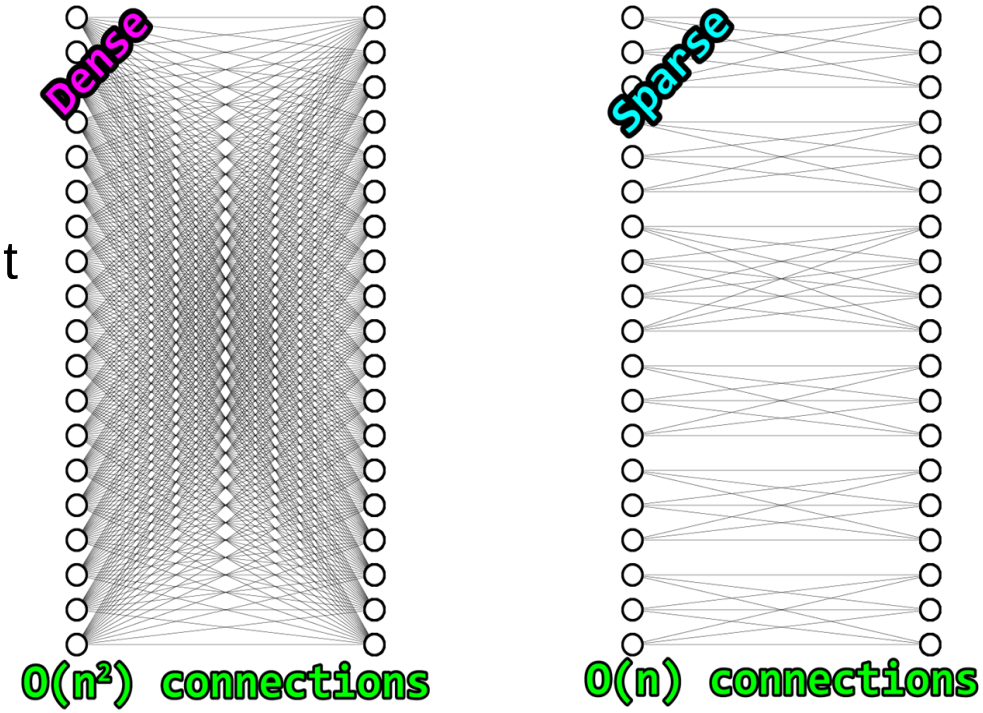
Outrepasser le problème de "feature engineering”

Le problème a été déplacé…

- **“Architecture engineering” remplace “feature engineering”**

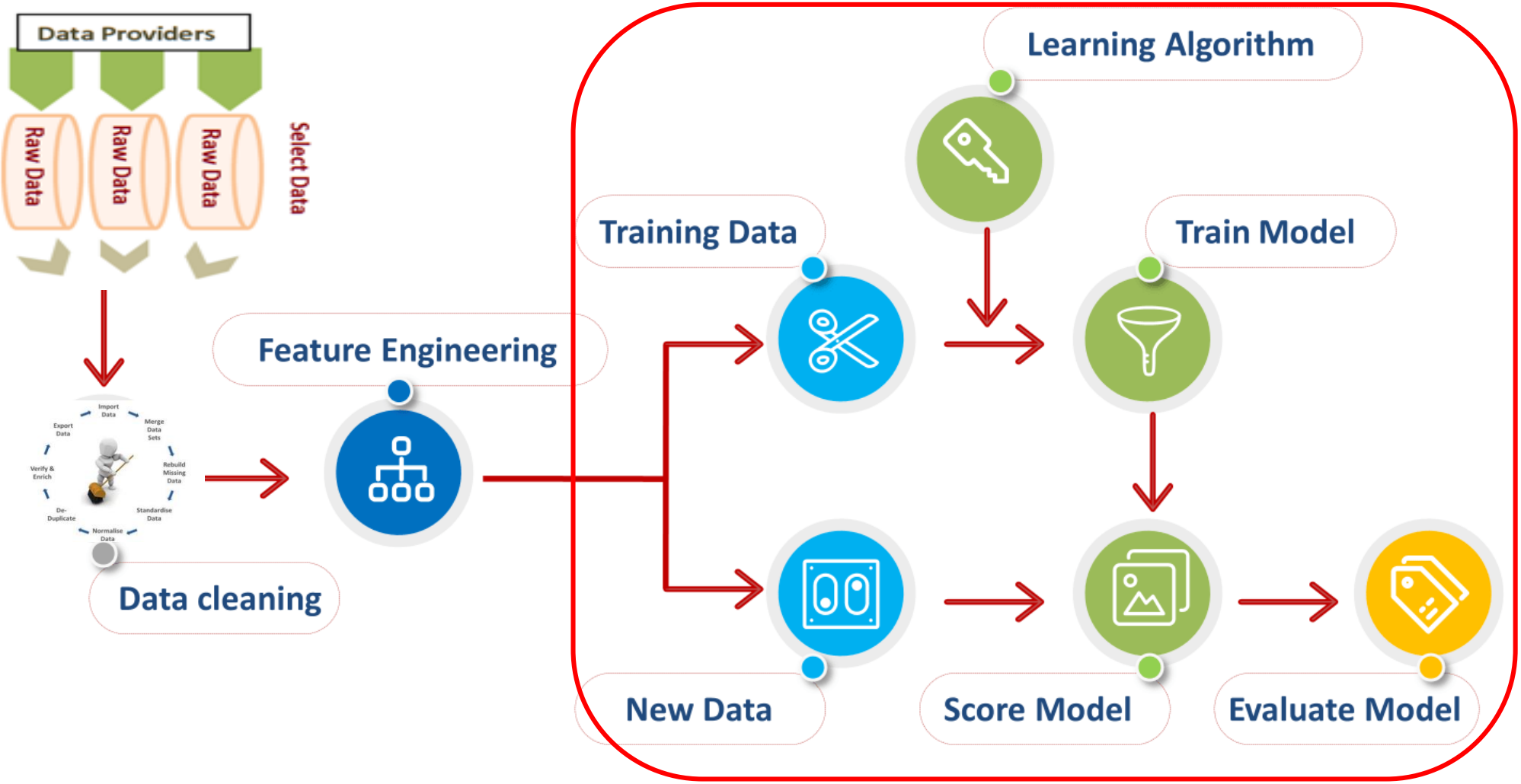
- Trop de poids ➔ poids partagés
  - Notion de voisinage → fenêtre d’attention
  - Convolutions : Juste des sommes pondérées de petites zones voisines dans les données (images)

- Trop de poids ➔ boite noire produisant des résultats non interprétables par l’humain



Modélisation possible d'un système d'IA

Interaction ? Ou est l'HUMAIN dans tous ca ?



Adapted from [Meduim2019]

Evaluation des performances

Utilisation / génération de différentes bases

- Pour estimer la probabilité d'erreur du classifieur, il faut disposer d'un voire **plusieurs** ensembles indépendants (**bases de validation et test**).
  - Leave one out
  - N-folding (création de N sous-bases d'apprentissage par tirage aléatoire avec souvent une taille = 80%)
  - …
- Optimisation des paramètres
  - Base de validation
  - Grid-search

- On construit alors les **matrices de confusion** pour ces ensembles
- On fait des moyennes

		Vraie classe	
		Positif	Négatif
Prediction	Positif	14	10
	Négatif	21	55

Vers une typologie des systèmes d'IA (ML / RdF)

Quelques pistes de réflexions [Ramel2019] **…**

- Comment est intégrée l'expertise humaine dans les systèmes d'IA ?
- Existe-t-il des systèmes faisant coopérer Intelligence Artificielle et Intelligence Humaine → systèmes d'IA interactifs ?
  (Boites blanches, transparence, explainabilité…)
- Existe-t-il des systèmes d'IA évolutifs, capable d'apprendre en continu (au fil de l'eau) ?
- Existe-t-il des systèmes d'IA évolutifs et interactifs ?

Vers une typologie des systèmes d'IA (ML / RdF)

Tentative de représentation globale

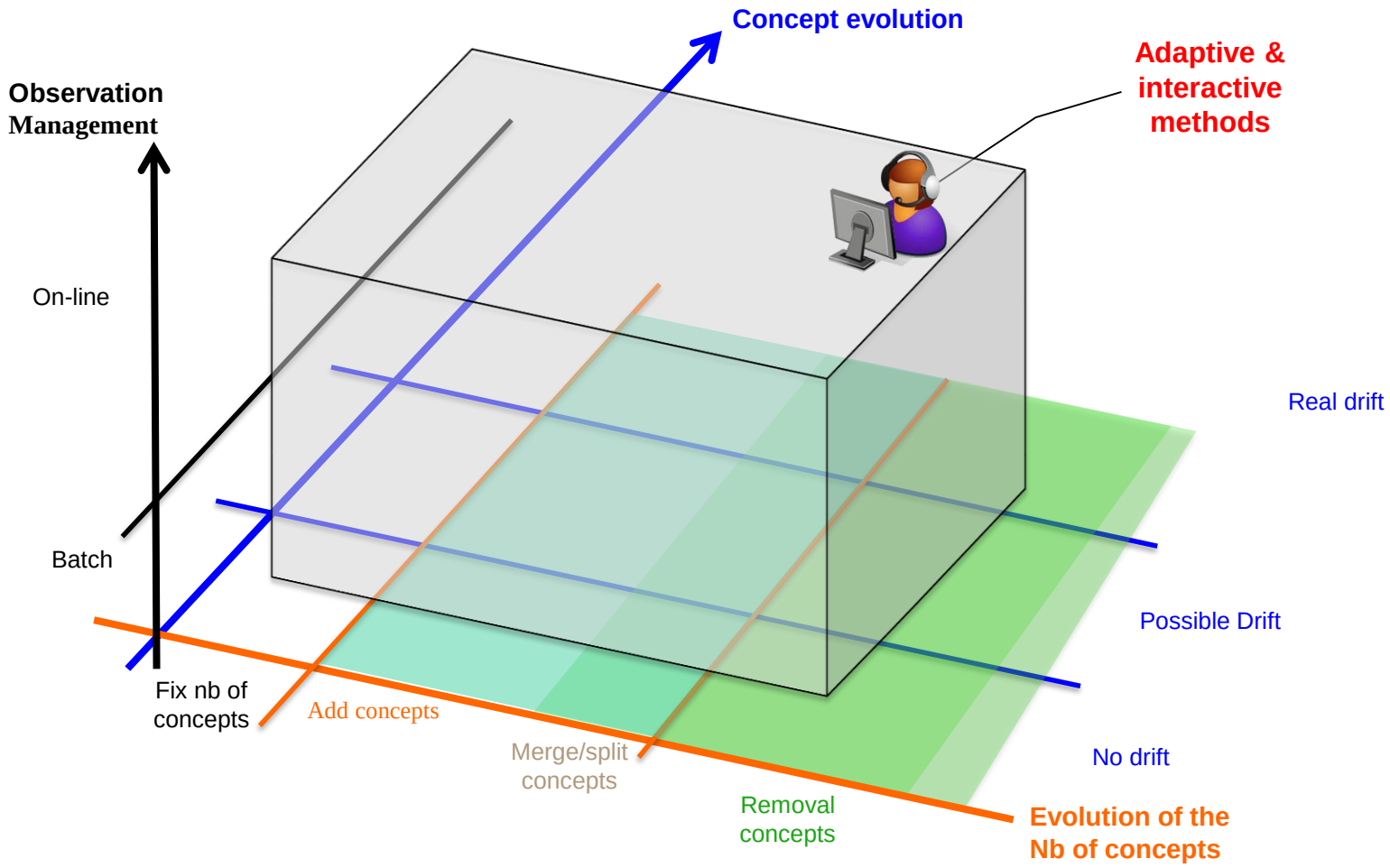
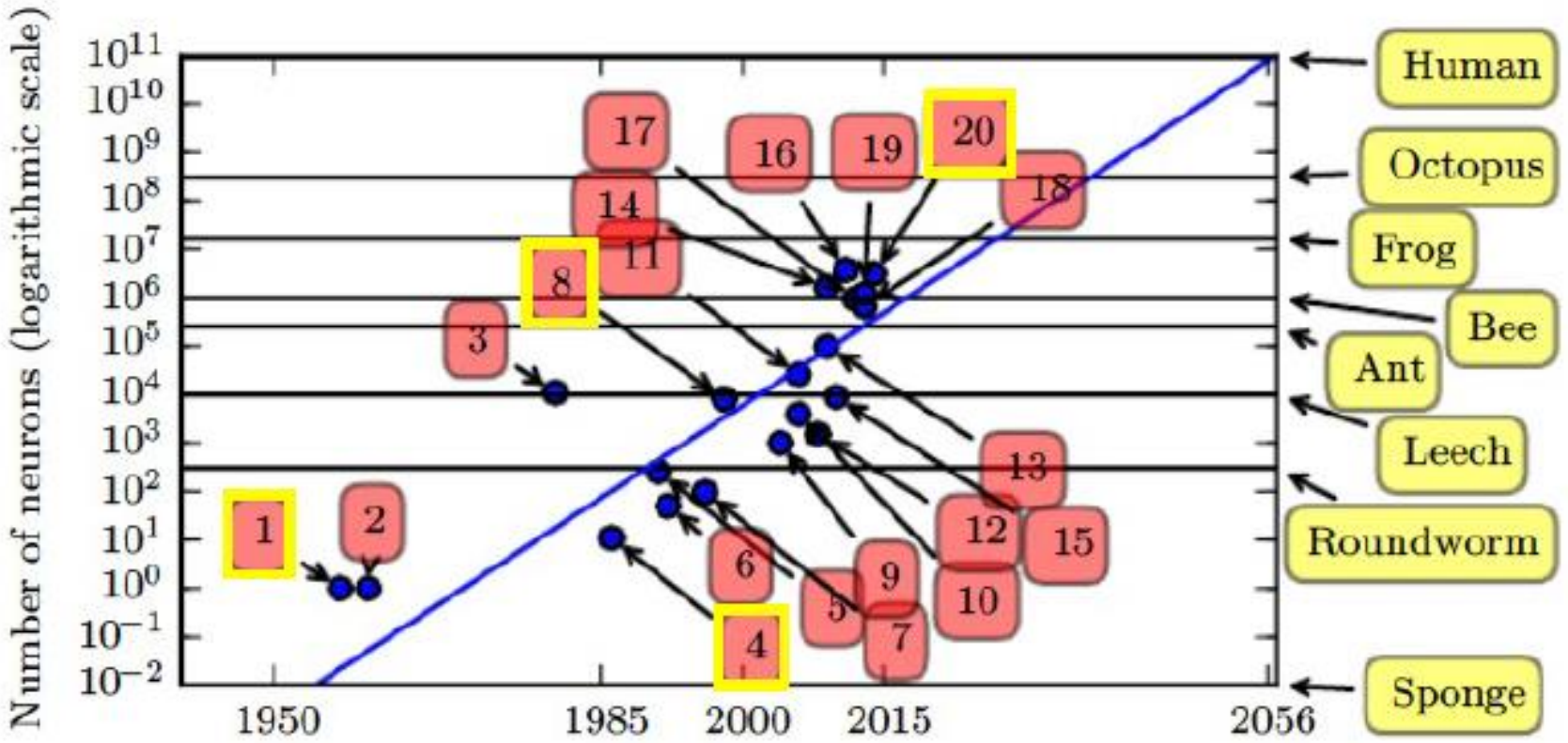


Schéma adapté de [Ragot2017]

Bilan : Réseaux de neurones

Vers les reseaux profonds

Depuis leur introduction dans les années 1950, la taille des réseaux de neurones a doublé en moyenne toutes les 2:4 années.

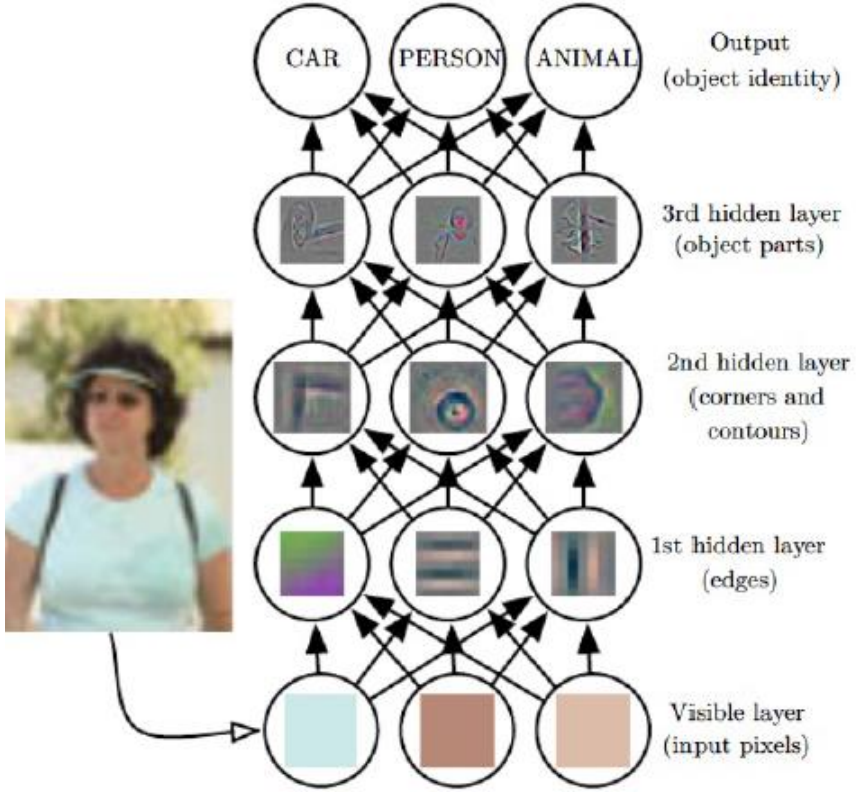


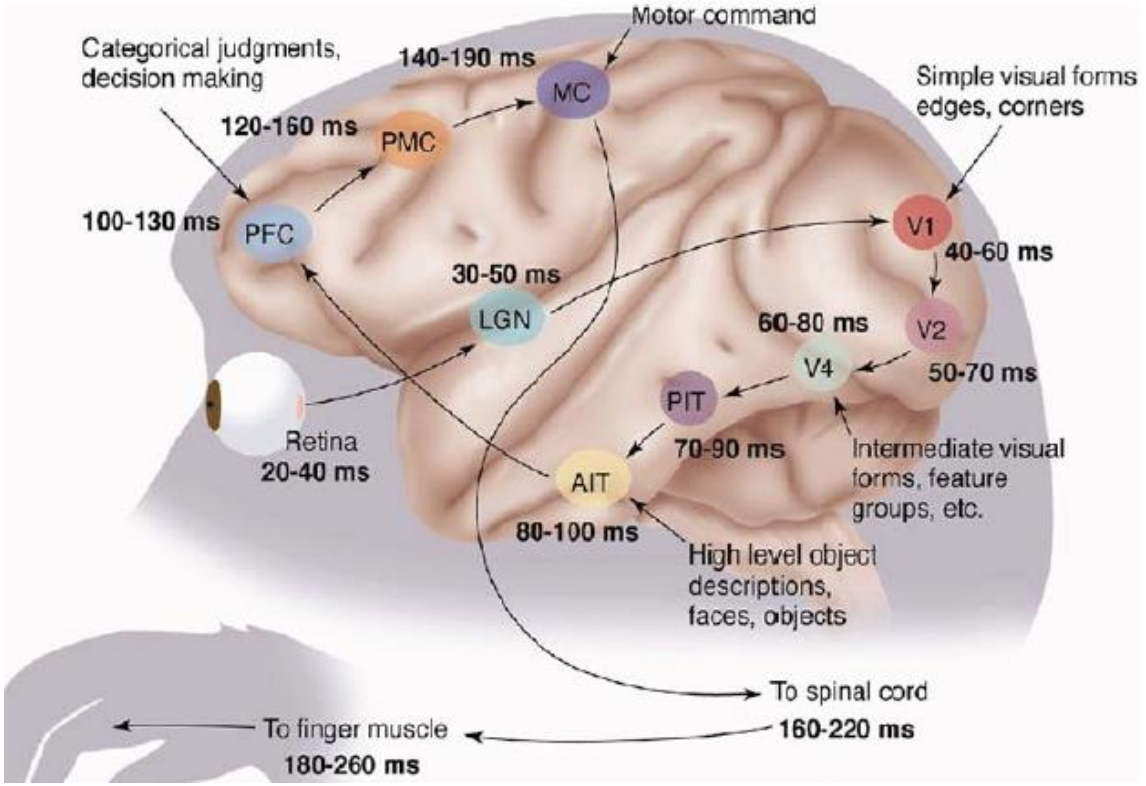
1 : Perceptron ; 4 : Premiers perceptrons multi-couches ; 8 : LeNet-5 (LeCun et al., 1998b) ; 20 : GoogLeNet

Bilan : Réseaux de neurones

Vers les réseaux profonds

Mais on est très loin de répliquer la complexité du cerveau humain !!!





Bilan final

L'IA : un des grands défis scientifiques et technologiques de notre temps

- Questions scientifiques :
  - Intégration de techniques symboliques (raisonnement, gestion des connaissances) et connexionnistes
  - Explication des raisonnements et des décisions
  - Quantification des incertitudes
  - Connaissances a priori en apprentissage (apprendre à partir de peu d'exemples)
  - Etc.
- Questions éthiques (protection de la vie privée, robots autonomes, etc.)
- Questions économiques (automatisation : impacts négatifs et positifs sur l'emploi, transformation de certains secteurs économiques – transports, services, etc.)

Références (i)

- Thierry Denoeux. Introduction à l'apprentissage automatique. Cours de l'Université de Technologie de Compiègne. Dép. Génie Informatique. Heudiasyc (UMR CNRS 7253). 2018.
- W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4) :115-133, 1943.
- F. Rosenblatt. The perceptron, a perceiving and recognizing automaton (Project PARA). Cornell Aeronautical Laboratory, 1957.
- Burr Settles, Active Learning Literature Survey. : A Computer Sciences Technical Report, University of Wisconsin–Madison, 2009
- JY Ramel, N Vincent. Semantic and interaction: when Document Image Analysis meets Computer Vision and Machine Learning ICDAR 2019. Sydney, Australia.
- Ramel JY, Vincent N., Emptoz H., Interprétation de documents techniques par "cycles perceptifs" à partir d'une perception globale du document. Revue Traitement du Signal. Vol. 15 n°2 - 1998. p1-20.

Références (ii)

- R. Polikar, L. Udpa, S. Udpa, V. Honavar. Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics. Rowan University USA, 2001.
- Kun Deng, Yaling Zheng, Chris Bourke. New algorithms for budgeted learning May 2013 Machine Learning 90(1)
- K Trapeznikov, V Saligrama. Supervised sequential classification under budget constraints - Artificial Intelligence and Statistics, 2013
- Zhixiang Xu. Supervised Machine Learning Under Test-Time Resource Constraints: A Trade-off Between Accuracy and Cost. Washington University in St. ETDs Thesis. Louis 2014
- Xiaodan Liang Hongfei Zhou Eric Xing. Dynamic-structured Semantic Propagation Network arXiv:1803.06067v1 [cs.CV] 16 Mar 2018
- N. Ragot. Contributions à la reconnaissance de formes et applications à l'analyse de l'écrit et des documents. HDR Université de Tours. 2017.

Licence

- Cette présentation est distribuée sous licence Creative Commons
- Attribution-ShareAlike 4.0 International