Survol de l'apprentissage machine

Sébastien Gambs
Université de Rennes 1 - INRIA

sgambs@irisa.fr

31 janvier 2011

Apprentissage machine

Définition

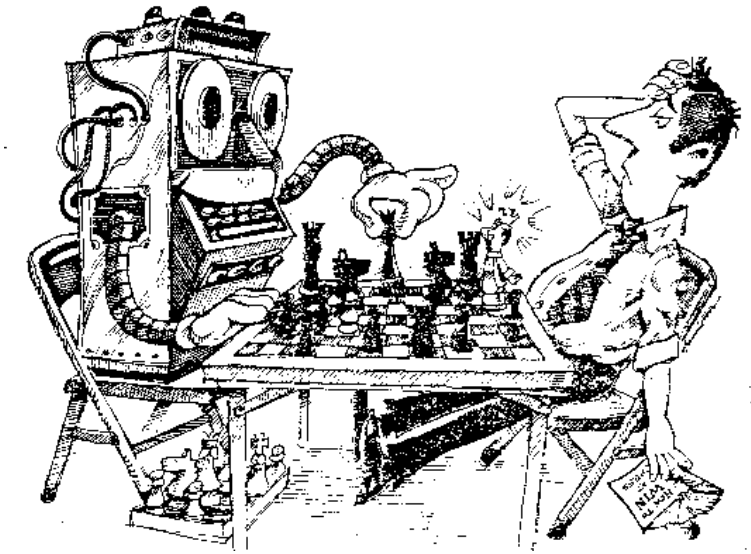
- L'apprentissage machine étudie les techniques permettant de *donner à la machine la capacité d'apprendre à partir d'expériences passées*.
- Il s'agit d'un des domaines formant l'intelligence artificielle.
- *Apprendre* est une capacité innée chez un être humain, mais difficile à formaliser pour la transmettre à un ordinateur.
- Important : mémoriser par coeur (comme le font très bien les ordinateurs), ne veut pas dire apprendre.

Généralisation

- En informatique "traditionnelle", on résout un problème en donnant explicitement à la machine les instructions qu'elle doit exécuter.
- En apprentissage machine, on cherche plutôt à donner à la machine la capacité d'apprendre à résoudre ce problème à partir d'exemples d'entrées/sorties formant l'ensemble d'entraînement.
- On espère qu'ensuite la machine puisse généraliser ce qu'elle a appris à des nouveaux cas non rencontrés auparavant.
- Remarque : l'apprentissage machine est particulièrement bien adapté pour des domaines où la connaissance est difficile à formaliser ou pour lesquels il y a peu ou pas d'experts.

Exemple de l'ordinateur qui joue aux échecs

- Important : Méditer sur la différence entre un programme jouant aux échecs en utilisant sa puissance de calcul pour rechercher l'espace des combinaisons de jeu possibles et un ordinateur qui aurait appris à jouer en observant des parties jouées entre grands maîtres.



- Même résultat mais deux approches fondamentalement différentes.

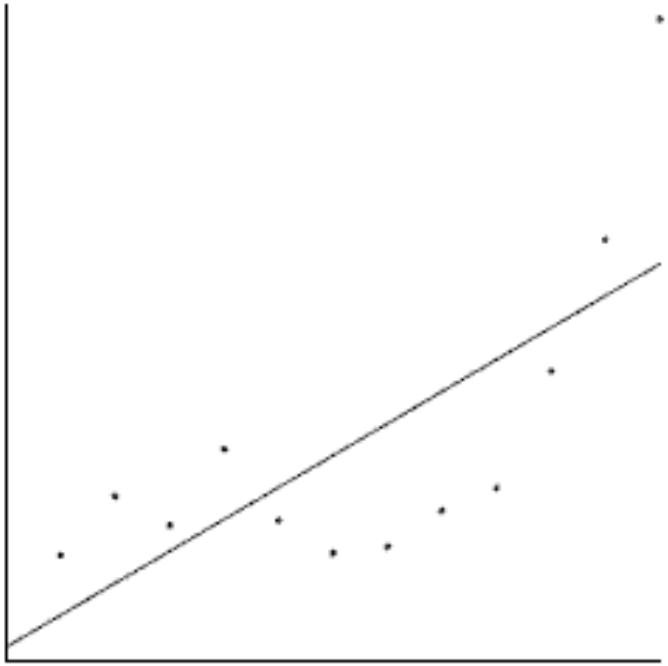
Quelques applications

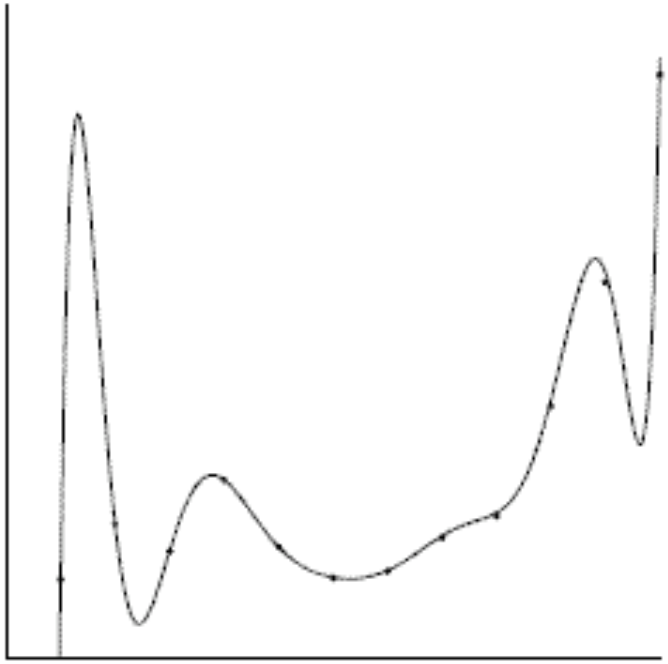
- Détection automatique de pourriels.
- Reconnaissance d'une forme ou d'un visage dans une image.
- Classification du genre musical d'une chanson.
- Prédiction du risque de crédit.
- Fouille de données en sociologie, bio-informatique, astronomie ou encore issue du Web.
- Compression de données.
- Apprendre à jouer comme un grand maître de Go.
- ...

Sur-apprentissage et sous-apprentissage

- Le sur-apprentissage (appelé *overfitting* en anglais) survient lorsque qu'on cherche à trop "coller" aux données d'entraînement (comme dans le cas d'un apprentissage par coeur).
- Au contraire si la classe de fonctions considérée par l'algorithme d'apprentissage n'est pas assez "riche" pour pouvoir décrire la diversité présente dans les données, on peut se retrouver en situation de sous-apprentissage.
- Dans les deux cas, les prédictions pour des nouveaux cas sera de basse qualité (*mauvaise généralisation*).

Exemple de l'interpolation de courbes





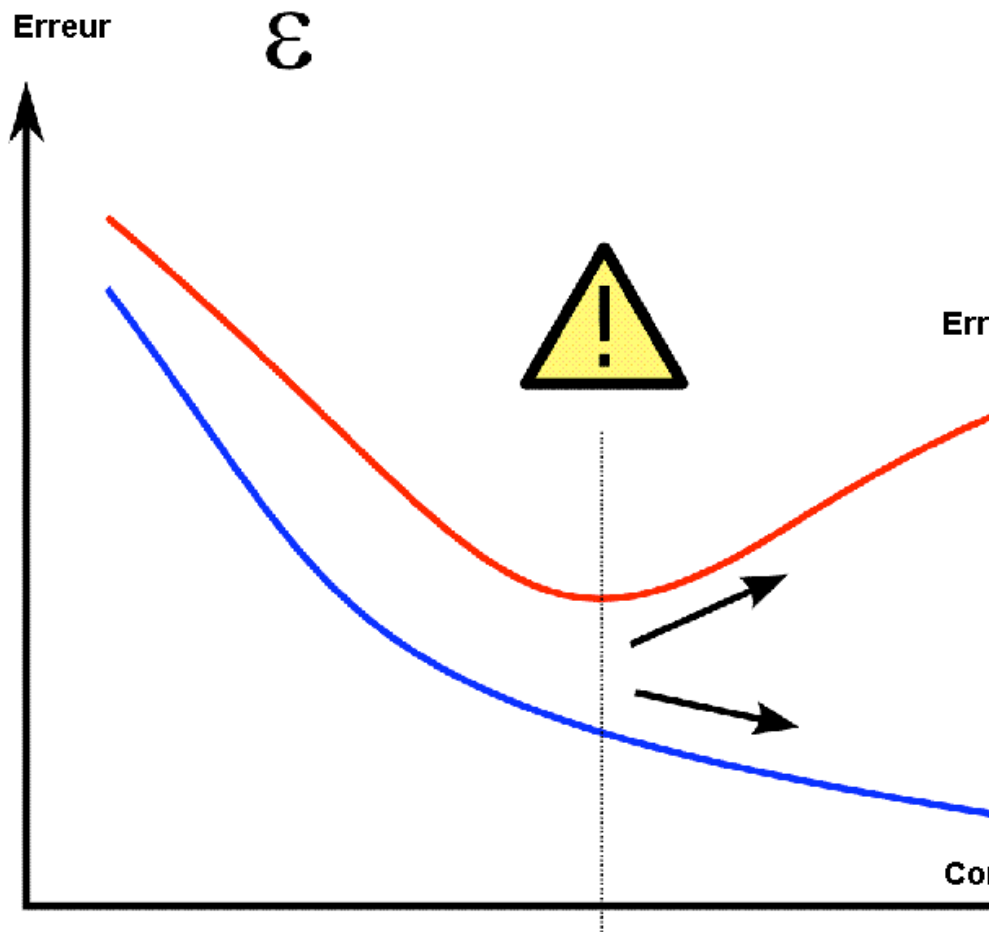
Ensemble d'entraînement

- Un ensemble d'entraînement comprenant n points de données peut être décrit comme $D_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ où:
  - $\mathbf{x}_i$ est un vecteur d'*observations* sur les *caractéristiques* du i^{eme} objet (ou point de données),
  - et y_i est la *classe* associée à cet objet.
- Exemple typique : si l'objet peut être décrit par d attributs réels, alors $\mathbf{x}_i \in \mathbb{R}^d$ et $y_i \in \{-1, +1\}$ pour la classification binaire.
- Exemple concret : le client d'une banque qui serait décrit par des attributs comme son âge, son salaire ou sa situation familiale, et pour lequel sa classe correspondante serait "être capable ou non de rembourser un prêt".

Phase de test

- Evaluation des performances d'un algorithme d'apprentissage sur un ensemble de données appelé ensemble de test.
- L'erreur de test constitue un estimé de la vraie erreur de généralisation. Plus on dispose de données, plus l'estimé sera precis et fiable.
- Important : pour que l'estimé mesuré soit non-biaisé, il est important que l'ensemble de test soit disjoint de l'ensemble d'entraînement.
- Exemple d'utilisation typique : utiliser $\frac{4}{5}$ des données disponibles pour l'entraînement et $\frac{1}{5}$ pour le test.

Erreur d'entraînement et erreur de test



Autres méthodes d'évaluation

- Validation croisée : on répète plusieurs fois l'étape d'entraînement/test mais en mélangeant aléatoirement les données à chaque fois.
- Validation par blocs : on coupe l'ensemble des données en k blocs distincts. Chaque bloc va servir une seule fois pour le test et se retrouvera dans l'ensemble d'entraînement le reste du temps.
- Validation jackknife (aussi appelé *leave-one-out*): on entraîne sur tous les points sauf un sur lequel on mesure l'erreur de test. On fait la même chose pour tous les n points de données (ce qui revient à faire une validation à n blocs).

Formes d'apprentissage-machine

- Apprentissage supervisé : on connaît des paires d'entrées/sorties $(\mathbf{x}, y)$ et on souhaite apprendre une fonction f qui permet d'associer une valeur y à un vecteur d'observations $\mathbf{x}$.
- Apprentissage non-supervisé : on connaît seulement les vecteurs d'observations $\mathbf{x}$ et on cherche à *découvrir la structure cachée se trouvant à l'intérieur des données*.
- Apprentissage par renforcement : situation d'apprentissage d'un agent dirigé par un but et interagissant avec un environnement incertain.
  Exemple : un robot explorant la surface de Mars.

Apprentissage supervisé

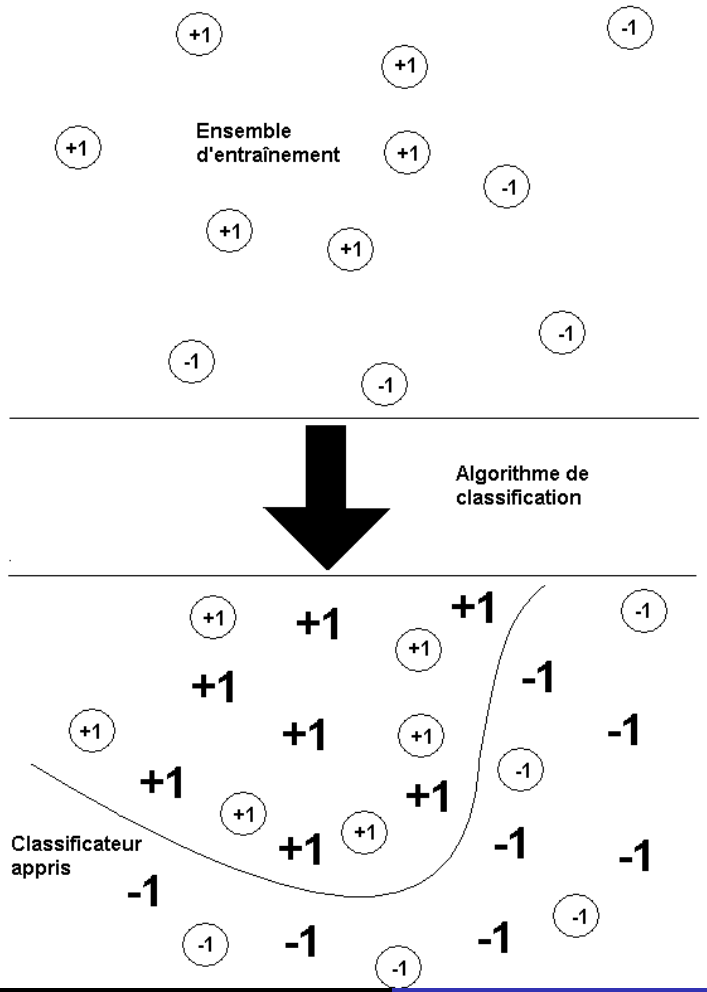
Tâches d'apprentissage supervisé

- Classification : on cherche à apprendre un classificateur qui va pouvoir déterminer la classe $y_?$ d'un objet inconnu $x_?$ à partir d'observations sur cet objet.
  Exemple : apprendre un classificateur qui pourra évaluer si un client pourra ou non rembourser un prêt (*classification binaire*).
- Régression : prédiction d'une caractéristique inconnue de $x_?$.
  Exemple : prédire *un risque* de non-remboursement (au lieu de simplement risque ou non-risque pour la classification).
- Ordonnancement: générer un ordre partiel sur les points de données à partir d'une requête.
  Exemple : pages retournées par un moteur de recherche comme Google, produits suggérés par un système de recommandation.

Classification

- La classification cherche à apprendre un classificateur *qui pourra être ensuite utilisée pour prédire la classe d'un objet inconnu à partir d'un vecteur d'observations portant sur cet objet*.
- But : apprendre une fonction f, appelée *classificateur*, qui chaque vecteur d'observations $\mathbf{x}$ va pouvoir associer sa classe correspondante y.
- Quelques exemples :
  - reconnaître les empreintes digitales ou le visage d'une personne,
  - classifier une nouvelle qui vient d'arriver comme appartenant à la section "sports" ou "culture",
  - détection des cas de fraudes.

Exemple de la tâche de classification



Pas de déjeuner gratuit mais un sac d'outils

- Théorème de l'impossibilité du déjeuner gratuit (ou *no-free lunch theorem* en anglais) (Wolpert 95) : il n'existe pas un algorithme d'apprentissage qui est supérieur à tous les autres pour toutes les distributions possibles des données.

- En pratique, cela veut dire qu'il ne faut pas chercher un "l'algorithme ultime" mais plutôt développer un sac d'outils composé de différents types d'algorithmes ayant chacun leurs forces et faiblesses.

Notions de distance et similarité

- Une distance sert à mesurer à quel point deux objets sont *proches* ou *éloignés*.
- Propriétés des distances :
  - $dist(\mathbf{a}, \mathbf{b}) \geq 0$
  - $dist(\mathbf{a}, \mathbf{a}) \leq dist(\mathbf{a}, \mathbf{b})$
  - $dist(\mathbf{a}, \mathbf{b}) = dist(\mathbf{b}, \mathbf{a})$
- Exemples de distance :
  - Distance de Manhattan : $d(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d} \parallel \mathbf{a}_i - \mathbf{b}_i \parallel)$
  - Distance euclidienne : $d(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d} \parallel \mathbf{a}_i - \mathbf{b}_i \parallel^2)^{\frac{1}{2}}$
  - Distance de Minkowski : $d(\mathbf{a}, \mathbf{b}) = (\sum_{i=1}^{d} \parallel \mathbf{a}_i - \mathbf{b}_i \parallel^p)^{\frac{1}{p}}$
- Une mesure de similarité peut aussi être utilisée pour comparer à quel point deux objets sont *similaires* ou *dissemblables*.

Méthodes à base de voisinage

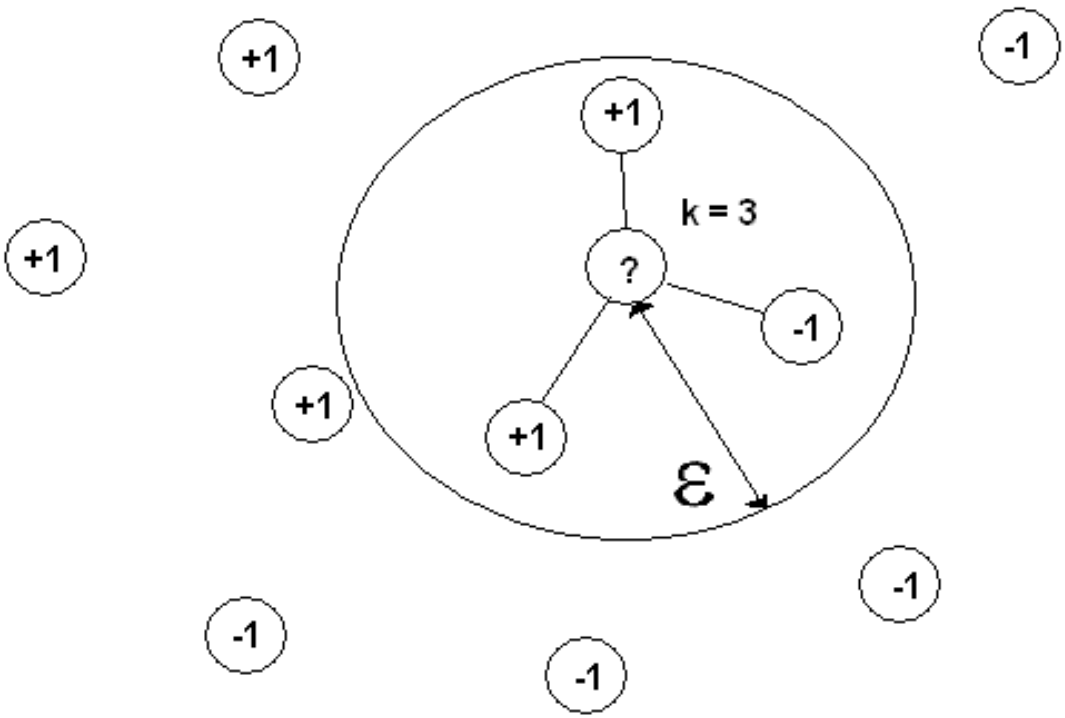
Une des manières les plus simples de classifier un nouveau point de donnée $\mathbf{x}_?$:

1. Trouver les points d'entraînement proches de $\mathbf{x}_?$.
2. Faire un vote de majorité par rapport aux classes des voisins.

Exemples d'algorithmes :

- k plus proches voisins : on considère seulement les k points les plus proches de $\mathbf{x}_?$.
- Fenêtres de Parzen : on considère tous les points dans un voisinage ϵ fixe.

Méthodes à base de voisinage : illustration



Avantages et inconvénients

Avantages :

- Très simple conceptuellement et facile à implémenter.
- Robuste par rapport au bruit.
- Ne requiert aucune phase d'entraînement (tout le travail se fait au moment de la classification).

Inconvénients :

- Le paramètre de l'algorithme, tel que le nombre de voisins k ou la taille du voisinage ϵ, doit être choisi avec discernement (souvent par validation croisée).
- Importance de la pertinence de la distance utilisée.
- Calculer la distance du point $\mathbf{x}_?$ par rapport à tous les points de D_n peut-être coûteux en temps de calcul.

Classifieurs Bayésiens

- Formule de Bayes : $P(y = i|\mathbf{x}) = \frac{P(y=i) \times P(\mathbf{x}|y=i)}{\sum_{j=1}^{k} P(y=j) \times P(\mathbf{x}|y=j)}$
- Hypothèse de simplification : tous les attributs sont indépendants les uns des autres.
- Dans ce cas: $f(\mathbf{x}) = argmax_i P(\mathbf{x}|y = i) \times P(y = i)$
- En pratique cela veut dire pour un nouveau point de donnée $\mathbf{x}_?$, on va prédire sa classe en regardant quelle est *la plus probable* par rapport aux données dont on dispose dans l'ensemble d'entraînement.

Avantages et inconvénients

Avantages :

- Très simple conceptuellement.
- Facile à implémenter.
- Prouvé comme minimisant l'erreur de Bayes.

Inconvénients :

- L'hypothèse d'indépendance des attributs est souvent non fondée en pratique.
- Minimiser l'erreur de Bayes ne guarantit pas de pouvoir généraliser correctement.

Arbres de décision

Dans un arbre de décision :

- Chaque noeud représente *un test sur un ou plusieurs attributs* dont le résultat va déterminer où on descend ensuite dans l'arbre.
- Chaque feuille est en général *etiquettée par une classe*.

Pour classifier un nouvel objet, on parcourt l'arbre en partant de la racine jusqu'à atteindre une feuille.

ID3 (Quinlan 86)

Algorithme itératif prenant un ensemble de données D et une liste d'attributs possibles en entrée :

1. Si tous les points de D ont la même classe y ou si ensemble d'attributs possibles est vide, créer une feuille étiquettée par la majorité de la classe des points de D et retourner.
2. Sinon déterminer l'attribut i qui maximise le gain d'information (minimise l'*entropie*).
3. Créer un noeud qui contient un test sur cet attribut.
4. Séparer la population des points de données en deux sous-populations D_A et D_B en appliquant le test.
5. Appeler ID3 sur D_A et D_B en enlevant l'attribut i des attributs possibles.

Avantages et inconvénients

Avantages :

- Compréhensibilité du modèle.

Inconvénients :

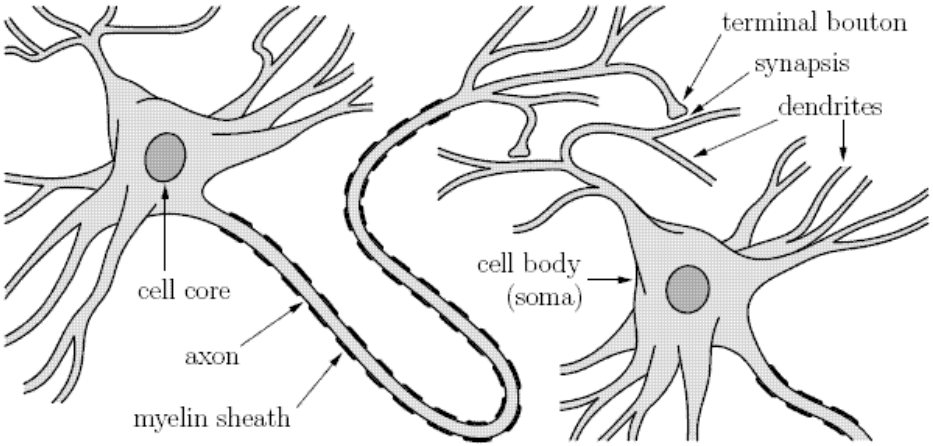
- Instabilité de l'algorithme, voulant dire suite à une petite perturbation des données, l'arbre produit peut-être très différent.
- Problème de sur-apprentissage si l'arbre de décision est trop profond.

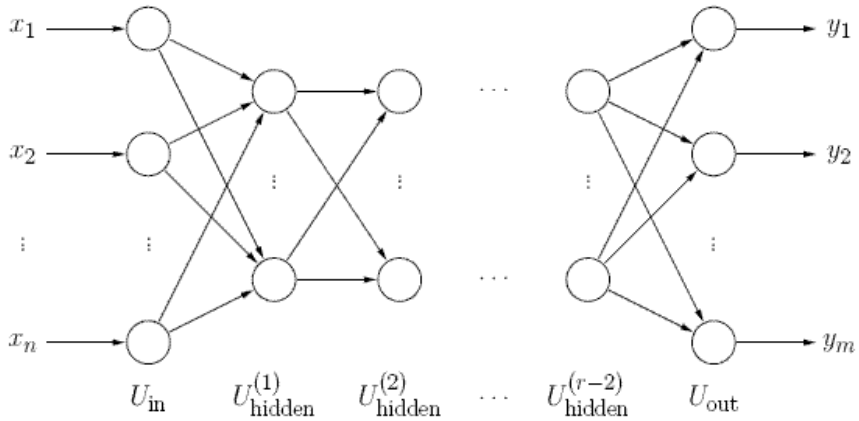
Exemple d'algorithme incrémental palliant à ces inconvénients et qui utilise des techniques d'élagage pour éviter que l'arbre de décision ne grandisse trop : C4.5 (Quinlan 93).

Réseaux de neurones

- Méthode s'inspirant du fonctionnement des réseaux de neurones dans le cerveau humain.
- Un réseau de neurones se compose de plusieurs couches de neurones.
- Les neurones sont connectés entre eux par des synapses, c'est à dire des *liens comportants des poids*.
- Chaque neurone est une petite unité de calcul :
  - qui prend en entrée les valeurs renvoyées par la couche précédente.
  - produit en sortie une valeur résultant de l'application d'une fonction.
  - Exemple de fonction : *sigmoïde* ou *tanh*.

Comparaison réseaux de neurones biologiques/artificiels





Avantages et inconvénients

Avantages:

- Bonne performance en pratique.
- Théorème de l'approximation universelle (Funahashi 89): il est possible d'approximer n'importe quelle fonction sur les réels en utilisant un réseau de neurones avec une seule *couche cachée* (c'est-à-dire une seule couche au milieu en plus de celle d'entrée et de sortie).

Inconvénients:

- Incompréhensibilité du modèle (le réseau de neurones fonctionne comme une *boîte noire*).
- Beaucoup de paramètres à optimiser (comme le nombre de neurones dans la couche cachée ou le choix des fonctions calculées par les neurones).

Méthodes d'ensemble

Illustration du proverbe "L'union fait la force".

Principe : On entraîne plusieurs classifieurs puis on les combine en un seul classifieur efficace (généralement par un mécanisme de vote).

AdaBoost (Freund et Schapire 97)

- Algorithme itératif qui s'arrête après un nombre T d'itérations.
- Chaque point de donnée à un poids qui change au fil des itérations.
- Ce poids reflète *combien un point est difficile à classifier*.
- Détail d'une itération :
  1. Trouver le classifieur faible qui minimise l'erreur pondérée sur les points de données.
  2. Calculer le coefficient de ce classifieur faible à partir de l'erreur pondérée.
  3. Repondérer les points de données.
     Pour chaque point, si celui-ci est correctement classifié par le classifieur faible de l'itération actuelle on diminue son poids, dans le cas contraire on l'augmente.

AdaBoost : illustration (tiré de Meir et Rätsch 03)

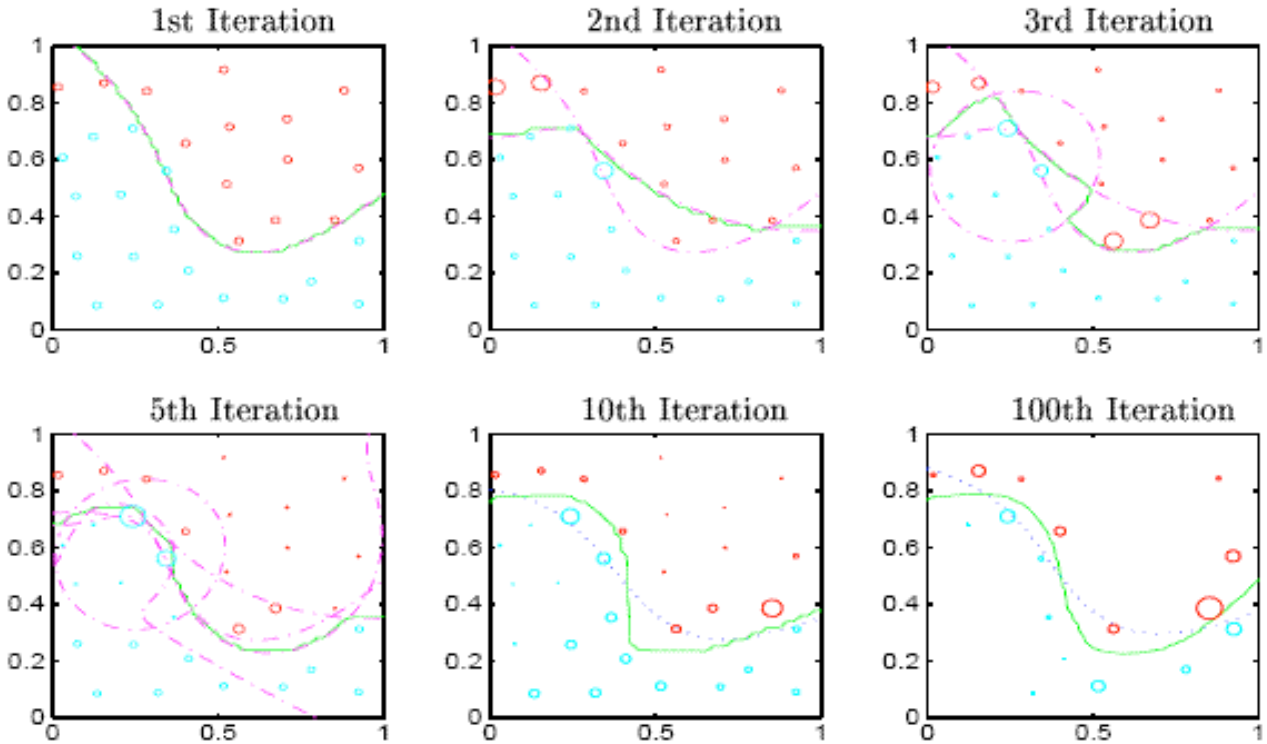


Fig. 1. Illustration of AdaBoost on a 2D toy data set: The color indicates the label and the diameter is proportional to the weight of the examples in the first, second, third, 5th, 10th and 100th iteration. The dashed lines show the decision boundaries of the single classifiers (up to the 5th iteration). The solid line shows the decision line of the combined classifier. In the last two plots the decision line of Bagging is plotted for a comparison. (Figure taken from [153].)

AdaBoost : avantages et inconvénients

Avantages :

- Très bonne performance en pratique.
- Un des rares algorithmes pour lequel faire du sur-apprentissage ne va pas de paire avec une mauvaise généralisation.

Inconvénients :

- Parfois particulièrement sensible au bruit présent dans les données.
- Importance du choix des classifieurs faibles, en particulier il ne faut pas que chaque classifieur faible pris individuellement soit trop "bon".
- Sans cela il y des risques que le classifieur final combiné risque de faire pire que les classifieurs faibles individuels.

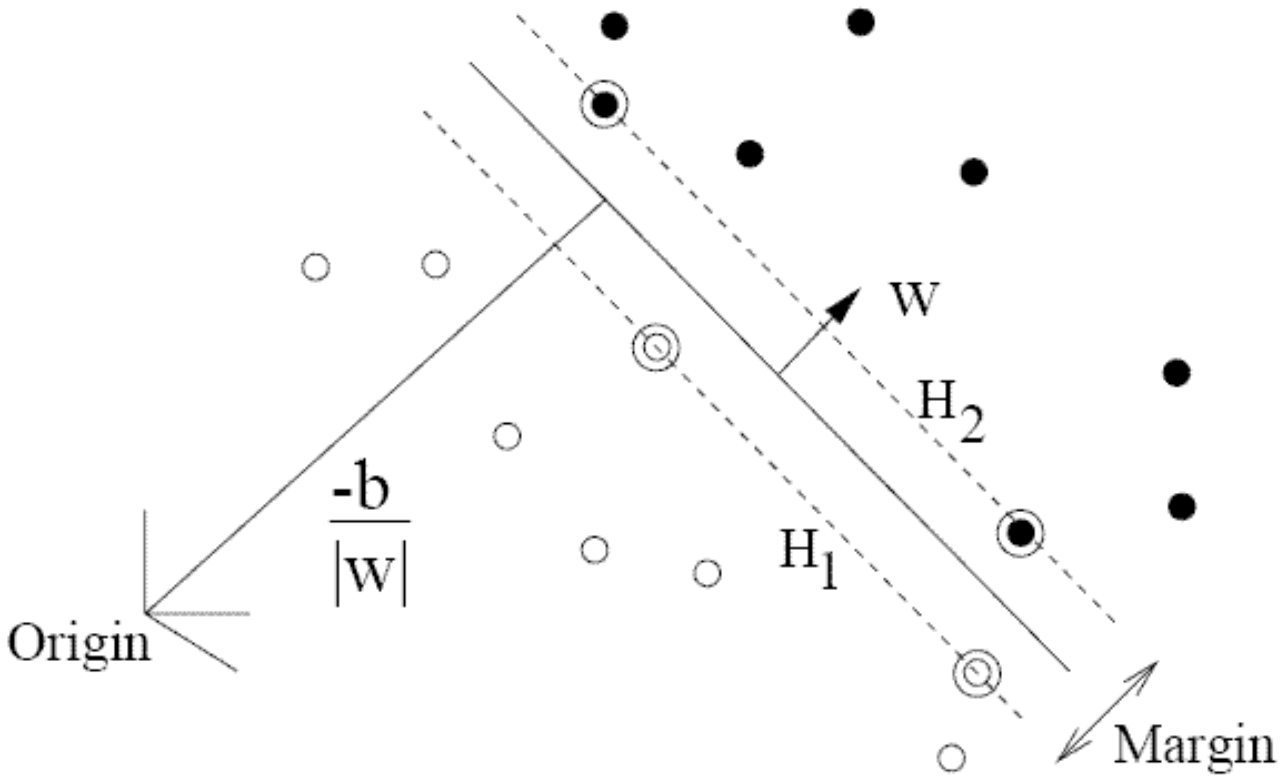
Autres méthodes d'ensemble

- Stacking (Wolpert 92) : combinaison itérative de classifieurs de types différents. A chaque itération on entraîne plusieurs modèles de classifieurs puis on garde le meilleur sur les données actuelles.
- Bagging (Breiman 96) : on pioche aléatoirement n points de données avec remplacement qu'on entraîne avec un classifieur, et on repète cette opération de nombreuses fois.
- Forêts aléatoires (Breiman 01).

Machines à vecteurs de support (Vapnik et Cortes 95)

- Idée : se focaliser seulement sur les points de données proches de la frontière de décision (appelés vecteurs de support) plutôt que sur l'ensemble des points de données.
- La machine à vecteurs de support va ensuite chercher à maximiser la marge, qui est la distance entre les vecteurs de support et la frontière de décision.
- Il a été observé/prouvé que la maximisation de la marge conduit souvent à une bonne généralisation.

Vecteurs de support et marge : illustration



Avantages et inconvénients

Avantages :

- Très bonne performance en pratique.

Inconvénients :

- Gourmand en calcul pour l'apprentissage (quadratique dans le nombre de points de données n).

Apprentissage non-supervisé

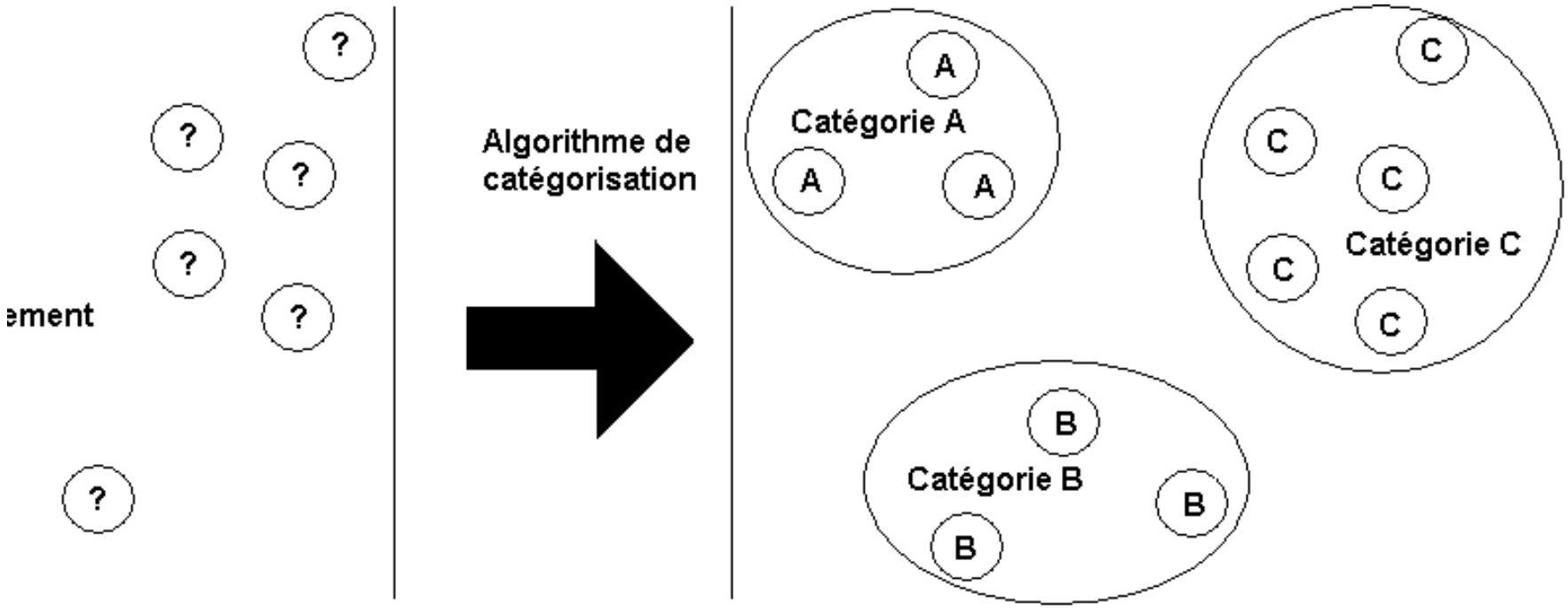
Tâches principales d'apprentissage non-supervisé

- Catégorisation (ou *clustering* en anglais) : découverte des classes naturelles présentes à l'intérieur des données.
  Exemple : révéler les groupes sociologiques se trouvant à l'intérieur d'une population à partir de données démographiques.
- Réduction de dimensionnalité : trouver une représentation en faible dimension des données de l'ensemble d'entraînement qui eux sont en haute dimension.
  Exemple : compression d'images ou de sons.
- Estimation de densité : apprendre explicitement une fonction de probabilité (appelée fonction de densité) qui représente la vraie distribution des données.
  Exemple : apprendre la structure musicale des morceaux d'un compositeur particulier.

Catégorisation

- La catégorisation (*clustering*) cherche à *découvrir les catégories naturelles présentes à l'intérieur des données*.
- Soit $D_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.
- But : associer à chaque $\mathbf{x}$ une catégorie $y \in \{1, \ldots, k\}$ de manière à ce que les objets similaires soient regroupés dans une même catégorie (*intra-similarité*) et que les objets dissemblables se retrouvent dans des catégories différentes (*inter-dissimilarité*).
- Exemple d'applications : trouver des catégories typiques existants parmi les clients d'un supermarché, regrouper automatiquement des chansons par thème.

Exemple de tâche de catégorisation

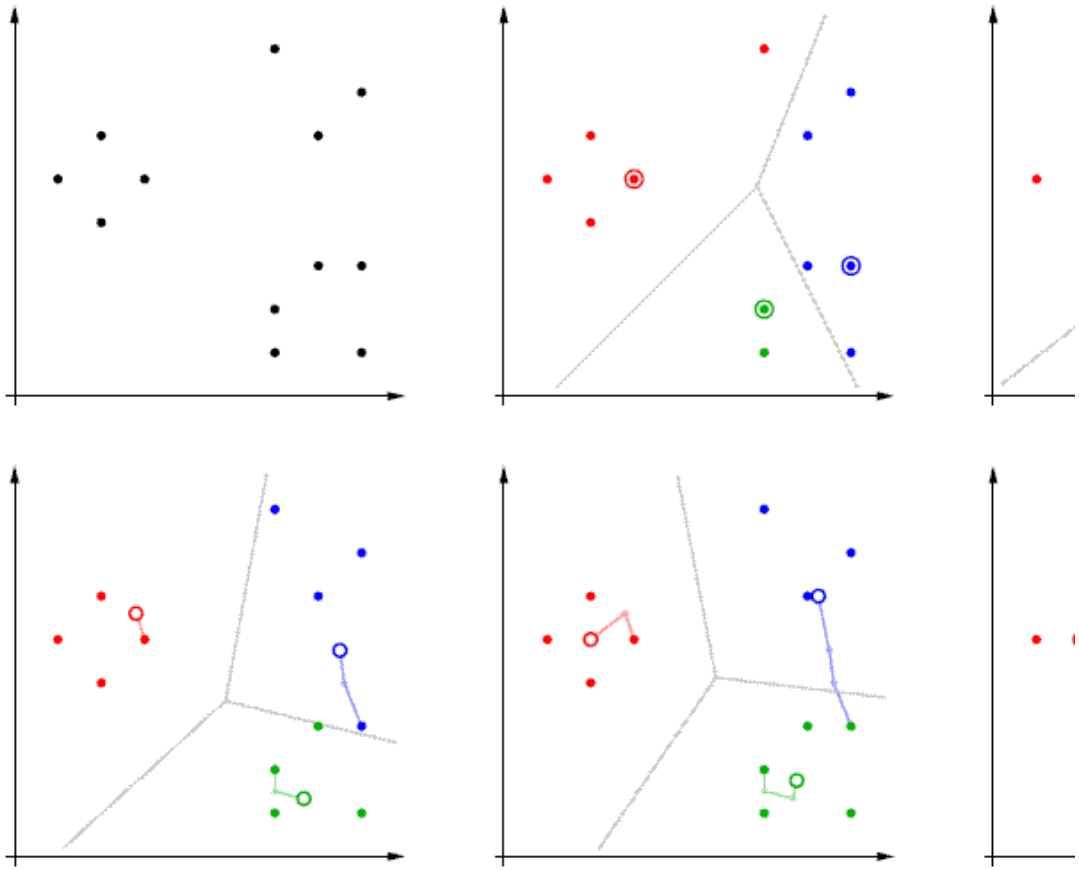


K-moyennes

1. Choisir aléatoirement les k points de données qui seront les centroïdes initiaux des catégories.
2. **Faire**
   - Pour chacun des $\mathbf{x}_i$ le rattacher au centroïde le plus proche.
   - Recalculer chacun des centroïdes en faisant la moyenne des points associés.

   Tant que changement parmi les centroïdes
3. Retourner les centroïdes calculés.

K-moyennes : illustration



De nombreuses approches possibles

Quelques approches :

- Partitionnement (e.g. k-moyennes).
- Hiérarchique :
  - Divisif.
  - Agglomératif.
- Graphe (e.g CHAMELEON).
- Densité (e.g. DBSCAN, DENCLUE).
- Conceptuel (e.g. COBWEB, CLARISSE).
- ...

Evaluation de la qualité des catégories

- Mesurer la qualité des catégories retournées par un algorithme de clustering est *difficile* et souvent *subjectif*.
- Première approche : définir une métrique qui va tenir compte à la fois de combien *les points à l'intérieur d'une catégorie sont proches* et combien *les points de deux catégories différentes sont éloignés*.
- Deuxième approche : prendre un ensemble de données avec étiquettes connues, passer les données à l'algorithme de catégorisation en *enlevant les étiquettes*, comparer les catégories retournées avec les vrais étiquettes.

Réduction de dimensionnalité

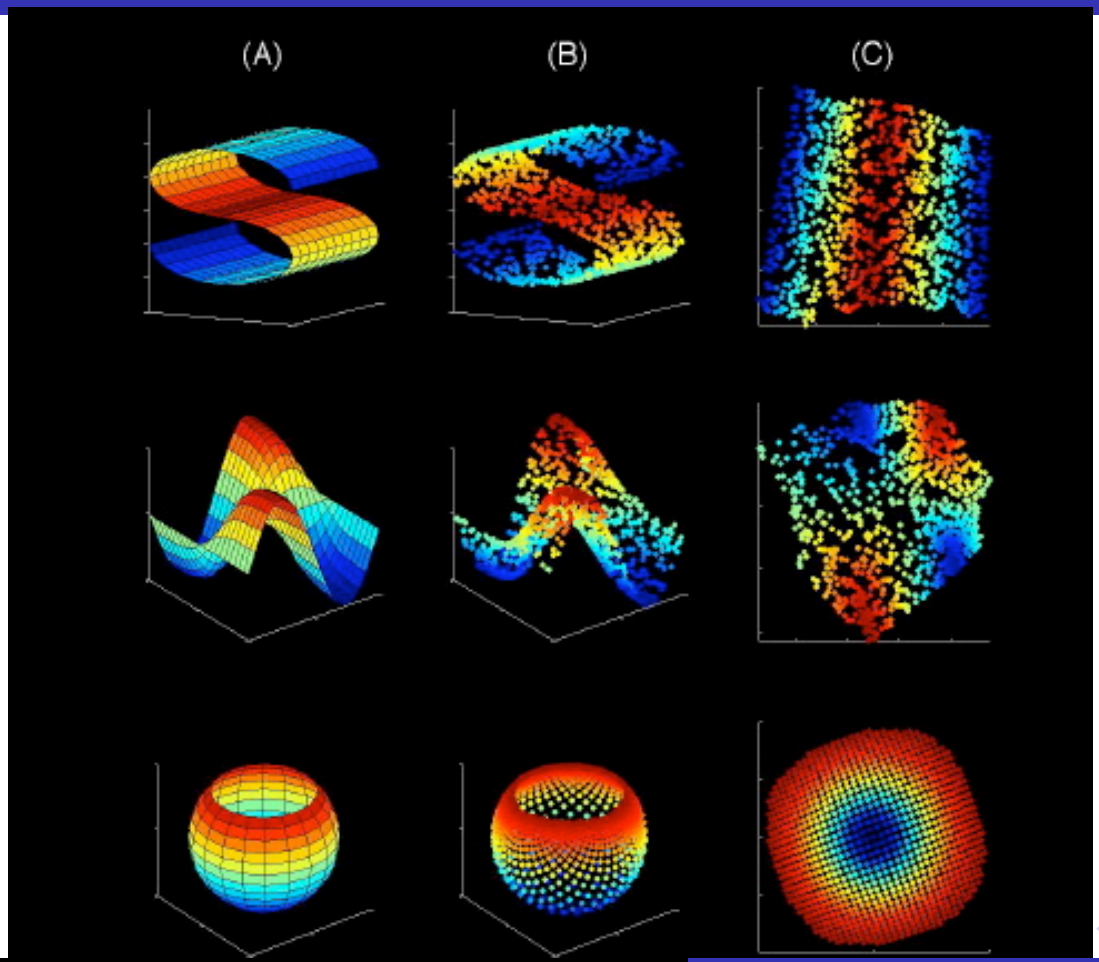
La réduction de dimensionnalité essaye de *trouver une représentation en faible dimension des données*.

Soit $D_n = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ tel que $\mathbf{x}_i \in \mathbb{R}^d$.

But : découvrir une transformation $\mathbf{x}_i \mapsto \mathbf{x}'_i$, tel que $\mathbf{x}'_i \in \mathbb{R}^k$ pour $k \ll d$, qui préserve au maximum l'information.

Exemple d'application : compression de données.

Réduction de dimensionnalité : exemples de sous-espace courbes (tiré de Roweis et Saul 00)



Quelques algorithmes

Sous-espace linéaire :

- Analyse en composantes principales.
- Échelonnement multidimensionnel.

Sous-espace non-linéaire :

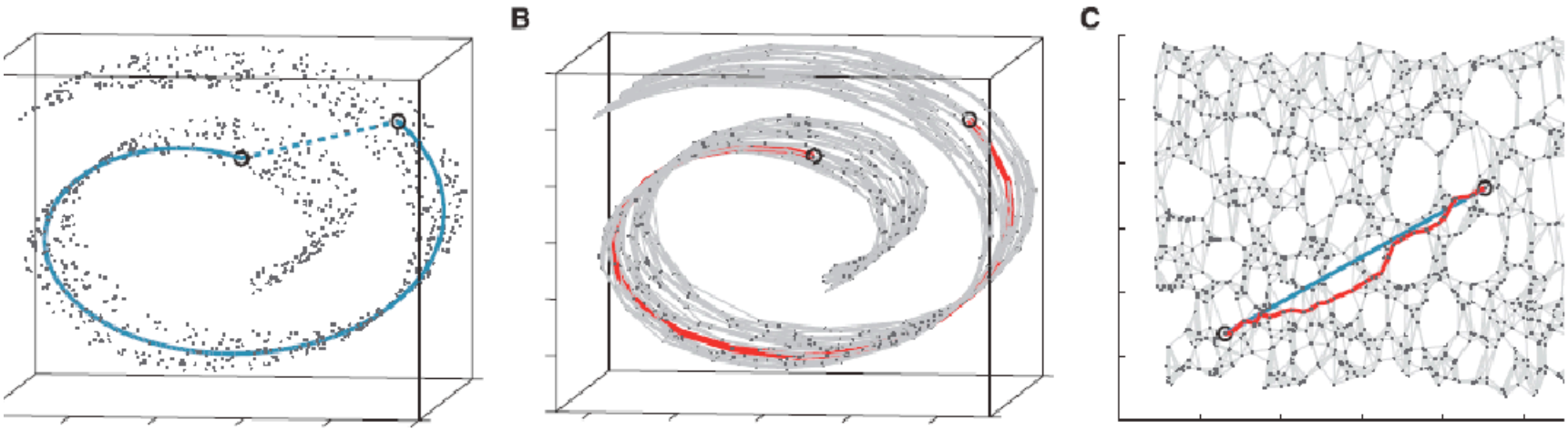
- Isomap (Tenenbaum, de Silva et Langford 00).
- LLE (Local Linear Embedding) (Roweis et Saul 00).

Catégorisation spectrale (Ng, Jordan et Weiss 01): combinaison de la réduction de dimensionnalité avec un algorithme de catégorisation.

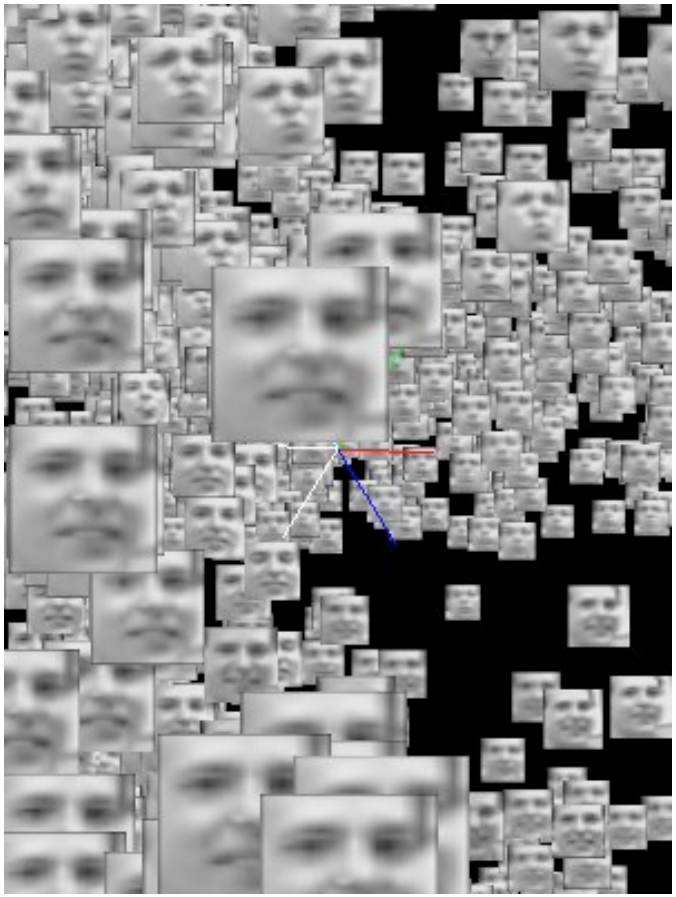
Isomap (Tenenbaum, de Silva et Langford 00)

1. Calculer la distance entre tous les points situés dans un voisinage fixe (tel qu'un ϵ-voisinage).
   Représenter ces distances sous la forme d'un graphe G où chaque arête entre deux points a un poids proportionnel à la distance entre les deux points.
2. Trouver la distance entre chaque paire de points en utilisant l'algorithme du plus court chemin sur le graphe G.
3. Appliquer l'échelonnement multidimensionnel sur la matrice de distance générée à l'étape précédente.

Exemple du *rouleau suisse* (tiré de Tenenbaum, de Silva et Langford 00)



Exemple de l'espace des visages (tiré de Roweis et Saul 00)



Estimation de densité

L'estimation de densité cherche à *modéliser la distribution sous-jacente aux données* sous la forme d'une fonction de probabilité (appelée fonction de densité).

Exemples de modèles :

- Mélange de Gaussiennes.
- Les chaînes de Markov cachées pour les données séquentielles.
- Les modèles graphiques comme les réseaux bayésiens pour modéliser les relations de cause à effet entre variables.

Conclusion

Conclusion

- Du point de vue théorique, l'étude de l'apprentissage-machine permet de mieux comprendre le fonctionnement, les mécanismes ainsi que les limites de l'apprentissage chez la machine.
- Du point de vue pratique, elle offre de nombreuses applications dans des domaines aussi différents que la vision, la musique, la prédiction financière, l'analyse du climat, l'astronomie ou encore la bio-informatique.
- L'apprentissage *supervisé* est adapté à dans le cas où on connait déjà les entrées/sorties alors que l'apprentissage *non-supervisé* permet de révéler la structure cachée à l'intérieur des données.

Conclusion (suite)

- Ces deux formes d'apprentissage sont cependant fortement *interconnectées*.
- Exemple : il arrive souvent qu'on utilise un algorithme de réduction de dimensionnalité durant une étape de pré-traitement avant de lancer un algorithme de classification afin d'accélérer celui-ci.
- Il est important aussi de se souvenir que l'apprentissage machine n'offre pas un algorithme d'apprentissage "ultime" mais plutôt un sac d'outils composé de différents algorithmes ayant chacun leurs forces et faiblesses dans lequel on peut piocher.

Quelques perspectives

Quelques perspectives non-abordées pendant la présentation :

- Apprentissage distribué et/ou préservant la confidentialité des données.
- Apprentissage de séquence temporelle.
- Exemple : prédire le temps qu'il fera demain ou l'évolution du cours de la bourse à partir des données des derniers jours
- Apprentissage en ligne : comment apprendre lorsqu'on reçoit de nouvelles données régulièrement.
- Réduction parmi les tâches d'apprentissage : comment réutiliser un algorithme qui résout une certaine tâche d'apprentissage pour en résoudre une autre.

Pour en savoir plus (cours et conférences)

Cours offerts au DIRO :

- IFT6390, Fondements de l'apprentissage-machine, Pascal Vincent.
- IFT6266, Algorithmes d'apprentissage, Yoshua Bengio.
- IFT6010, Traitement Statistique des Langues Naturelles, Philippe Langlais.

Conférences principales du domaine :

- ICML (International Conference of Machine Learning).
- NIPS (Neural Information Processing Systems).
- ECML (European Conference of Machine Learning).

Pour en savoir plus (livres et blog)

Quelques livres de référence :

- Bishop, C., *Neural networks for pattern recognition*, Oxford University Press, 1995.
- Duda, Hart and Stork, *Pattern Classification*, Wiley-Interscience, 2001.
- Tom Mitchell, *Machine Learning*, McGraw Hill, 1997.
- Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, 1999.

Blog de John Langford :

- http://hunch.net

C'est la fin!!!

Merci pour votre attention.
Questions?