Classification

Classification

- Elle permet de **prédire** si un élément est membre d'un groupe ou d 'une catégorie donné.
- Classes
  - Identification de groupes avec des profils particuliers
  - Possibilité de décider de l'appartenance d'une entité à une classe

- Caractéristiques
  - Apprentissage supervisé : classes connues à l'avance
  - Pb : qualité de la classification (taux d'erreur)
    - Ex : établir un diagnostic (si erreur !!!)

Processus à deux étapes

Etape 1 :

Construction du modèle à partir de l'ensemble d'apprentissage (training set)

Etape 2 :

Utilisation du modèle : tester la précision du modèle et l'utiliser dans la classification de nouvelles données

Construction du modèle

- **Chaque instance** est supposée appartenir à une classe prédéfinie
- La classe d'une instance est déterminée par l'attribut "classe"
- L'ensemble des instances d'apprentissage est utilisé dans la construction du modèle
- Le modèle est représenté par des règles de classification, arbres de décision, formules mathématiques, …

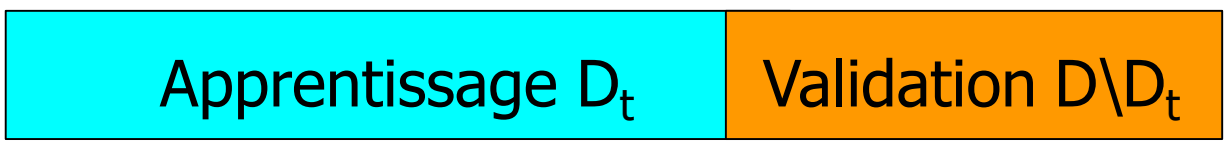
Utilisation du modèle

- **Classification de nouvelles instances ou instances inconnues**
- **Estimer le taux d'erreur du modèle**
  - la classe connue d'une instance test est comparée avec le résultat du modèle
  - Taux d'erreur = pourcentage de tests incorrectement classés par le modèle

Validation de la Classification (accuracy)

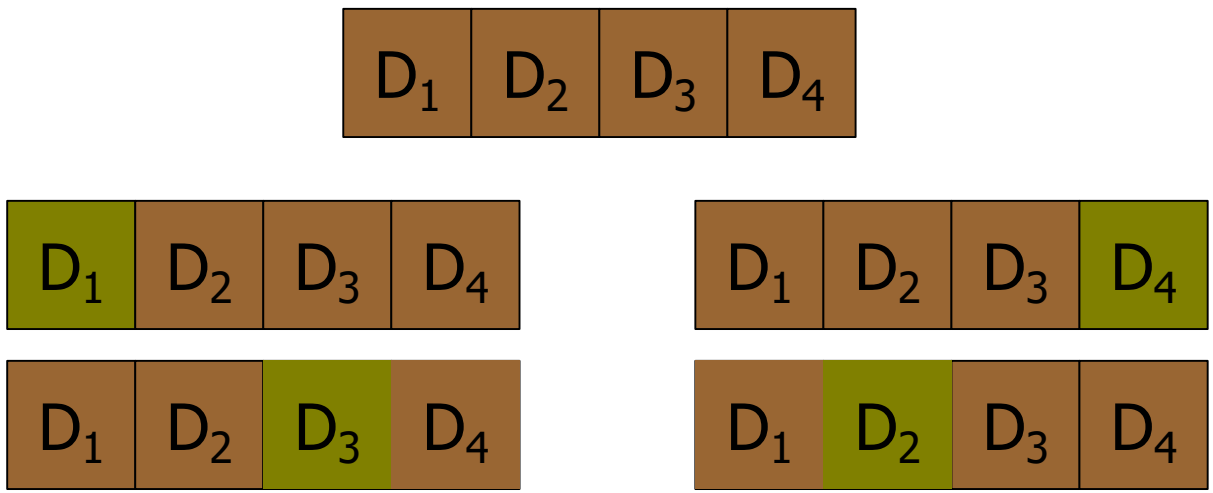
Estimation des taux d'erreurs :

- **Partitionnement** : apprentissage et test (ensemble de données important)
  - Utiliser 2 ensembles indépendents, e.g., ensemble d'apprentissage (2/3), ensemble test (1/3)



Validation de la Classification (accuracy)

- **Validation croisée** (ensemble de données modéré)
  - Diviser les données en *k* sous-ensembles
  - Utiliser *k-1* sous-ensembles comme données d'apprentissage et un sous-ensemble comme données test



- **Bootstrapping** : n instances test aléatoires (ensemble de données réduit)

Evaluation

- Taux d'erreur (Accuracy)
- Temps d'exécution (construction, utilisation)
- Robustesse (bruit, données manquantes,...)
- Extensibilité
- Interprétabilité
- Simplicité

Méthodes de Classification

- Méthode K-NN (plus proche voisin)
- Arbres de décision
- Réseaux de neurones
- Classification bayésienne

- Caractéristiques
  - Apprentissage supervisé (classes connues)

Méthode des plus proches voisins

- Méthode dédiée à la classification (k-NN : nearest neighbor).
- Méthode de raisonnement à partir de cas : prendre des décisions en recherchant un ou des cas similaires déjà résolus.
- Pas d'étape d 'apprentissage : construction d 'un modèle à partir d'un échantillon d 'apprentissage (réseaux de neurones, arbres de décision, …).
- Modèle = échantillon d'apprentissage + fonction de distance + fonction de choix de la classe en fonction des classes des voisins les plus proches.

kNN (K-nearest neighbors)

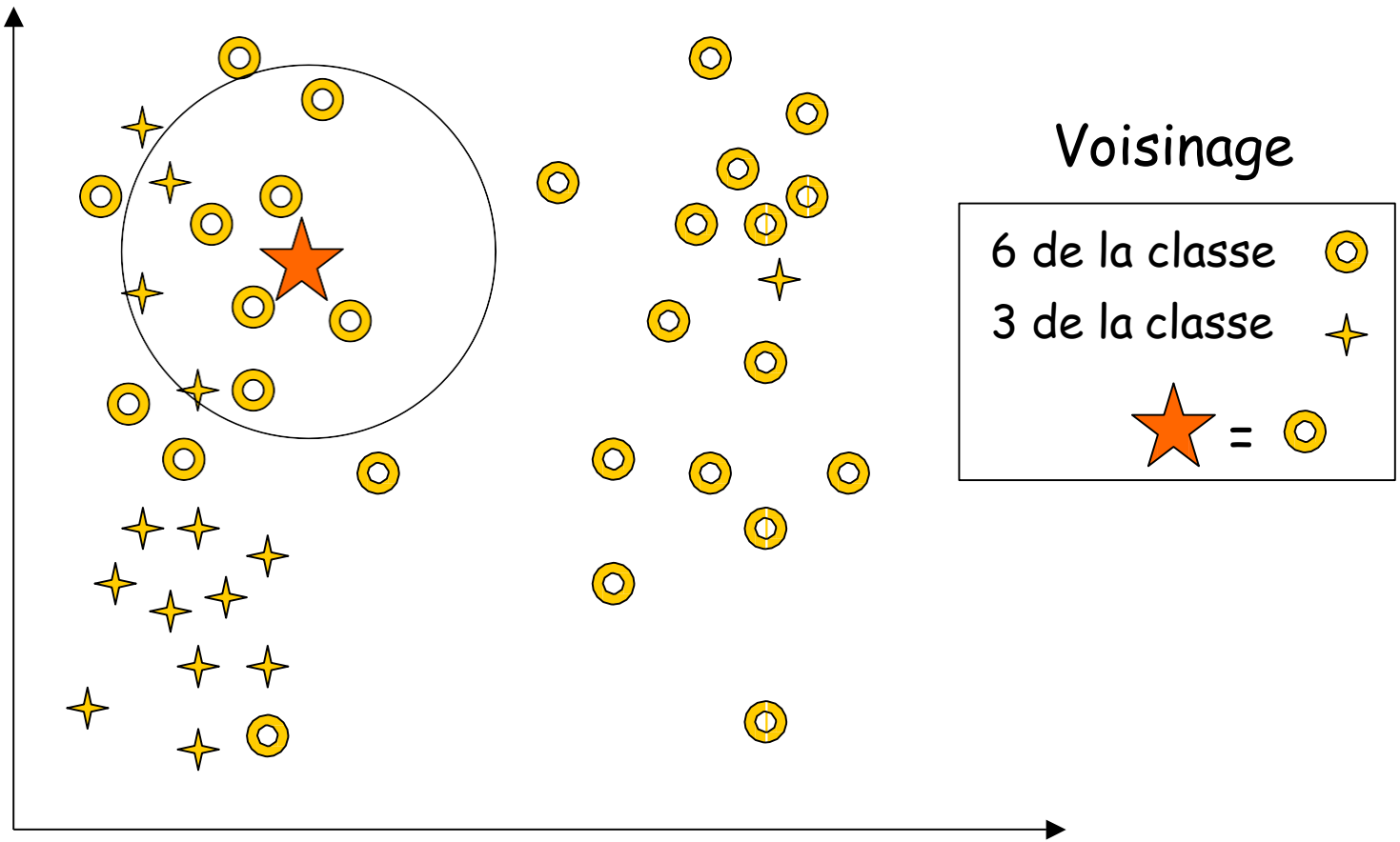
- Objectif : affecter une classe à une nouvelle instance
- donnée : un échantillon de m enregistrements classés (x, c(x))
- entrée : un enregistrement y
  - 1. Déterminer les k plus proches enregistrements de y
  - 2. combiner les classes de ces k exemples en une classe c
- sortie : la classe de y est c(y)=c

kNN : sélection de la classe

- Solution simple : rechercher le cas le plus proche et prendre la même décision (Méthode 1-NN).
- Combinaison des k classes :
  - Heuristique : k = nombre d 'attributs + 1
  - Vote majoritaire : prendre la classe majoritaire.
  - Vote majoritaire pondéré : chaque classe est pondérée. Le poids de c(xi) est inversement proportionnel à la distance d(y,xi).

- Confiance : Définir une confiance dans la classe attribuée = rapport entre les votes gagnants et le total des votes.

Illustration



Algorithme kNN : critique

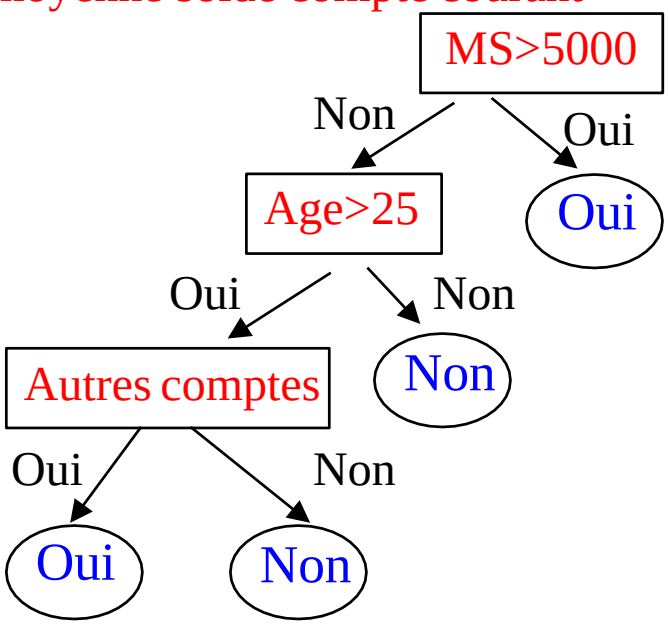
- Pas d'apprentissage : introduction de nouvelles données ne nécessite pas la reconstruction du modèle.
- Clarté des résultats
- Tout type de données
- Nombre d'attributs
- Temps de classification : -
- Stocker le modèle : -
- Distance et nombre de voisins : dépend de la distance, du nombre de voisins et du mode de combinaison.

Arbres de décision

- Génération d'arbres de décision à partir des données
- Arbre = Représentation graphique d'une procédure de classification

Accord d'un prêt bancaire

MS : moyenne solde compte courant



Un arbre de décision est un arbre où :

- **Noeud interne** = un attribut
- **Branche d'un noeud** = un test sur un attribut
- **Feuilles** = classe donnée

Génération de l'arbre de décision

Deux phases dans la génération de l'arbre :

- **Construction de l'arbre**
  - Arbre peut atteindre une taille élevée
- **Elaguer l'arbre (Pruning)**
  - Identifier et supprimer les branches qui représentent du "bruit" et Améliorer le taux d'erreur

Algorithmes de classification

- Construction de l'arbre
  - Au départ, toutes les instances d'apprentissage sont à la racine de l'arbre
  - Sélectionner un attribut et choisir un test de séparation (split) sur l'attribut, qui sépare le "mieux" les instances.

    La sélection des attributs est basée sur une heuristique ou une mesure statistique.
  - Partitionner les instances entre les noeuds fils suivant la satisfaction des tests logiques

Algorithmes de classification

- Traiter chaque noeud fils de façon récursive
- Répéter jusqu'à ce que tous les noeuds soient des terminaux.
- Un noeud courant est terminal si :
  - Il n'y a plus d'attributs disponibles
  - Le noeud est "pur", i.e. toutes les instances appartiennent à une seule classe,
  - Le noeud est "presque pur", i.e. la majorité des instances appartiennent à une seule classe (Ex : 95%)
  - Nombre minimun d'instances par branche (Ex : algorithme C5 évite la croissance de l'arbre, k=2 par défaut)
- Etiqueter le noeud terminal par la classe majoritaire

Algorithmes de classification

- Elaguer l'arbre obtenu (pruning)
  - Supprimer les sous-arbres qui n'améliorent pas l'erreur de la classification (accuracy) et l'arbre ayant un meilleur pouvoir de généralisation, même si on augmente l'erreur sur l'ensemble d'apprentissage
  - Eviter le problème de sur-spécialisation (over-fitting), i.e., on a appris "par coeur" l'ensemble d'apprentissage, mais on n'est pas capable de généraliser

Sur-spécialisation - arbre de décision

- **L'arbre *généré* peut sur- spécialiser l'ensemble d'apprentissage**
  - Plusieurs branches
  - Taux d'erreur important pour les instances inconnues
- **Raisons de la sur-spécialisation**
  - bruits et exceptions
  - Peu de donnée d'apprentissage
  - Maxima locaux dans la recherche gloutonne

Comment éviter l'overfitting ?

- **Deux approches :**
- **Pré-élagage** : Arrêter de façon prématurée la construction de l'arbre
- **Post-élagage** : Supprimer des branches de l'arbre complet ("fully grown")
- Convertir l'arbre en règles ; élaguer les règles de façon indépendante (C4.5)

Construction de l'arbre

- Evaluation des différents branchements pour tous les attributs
- Sélection du "meilleur" branchement "et de l'attribut "gagnant"
- Partitionner les données entre les fils
- Construction en largeur (C4.5) ou en profondeur (SPLIT)
- Questions critiques :
  - Formulation des tests de branchement
  - Mesure de sélection des attributes

Mesures de sélection d'attributs

- Gain d'Information (ID3, C4.5)
- Indice Gini (CART)
- Table de contingence statistique χ^2 (CHAID)
- G-statistic

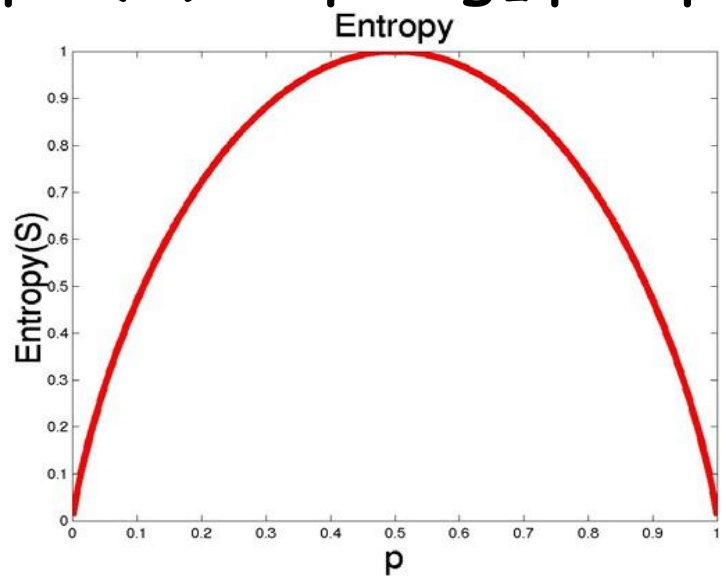
Gain d'information

- Sélectionner l'attribut avec le plus grand gain d'information
- Soient P et N deux classes et S un ensemble d'instances avec p éléments de P et n éléments de N
- L'information nécessaire pour déterminer si une instance prise au hasard fait partie de P ou N est (entropie) :

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Entropie

- S est l'ensemble d'apprentissage
- p_+ est la proportion d'exemples positifs (P)
- p_- est la proportion d'exemples négatifs (N)
- Entropie mesure l'impureté de S
  - Entropie(S) = $-p_+ \log_2 p_+ - p_- \log_2 p_-$



Gain d'information

- Soient les ensembles $\{S_1, S_2, \ldots, S_v\}$ formant une partition de l'ensemble S, en utilisant l'attribut A
- Toute partition S_i contient p_i instances de P et n_i instances de N
- L'entropie, ou l'information nécessaire pour classifier les instances dans les sous-arbres S_i est :

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- Le gain d'information par rapport au branchement sur A est

$$Gain(A) = I(p, n) - E(A)$$

- Choisir l'attribut qui maximise le gain → besoin d'information minimal

Indice Gini

- Utiliser l'indice Gini pour un partitionnement pur

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

$$Gini(S_1, S_2) = \frac{n_1}{n} Gini(S_1) + \frac{n_2}{n} Gini(S_2)$$

- pi est la fréquence relative de la classe c dans S
- Si S est pur (classe unique), Gini(S) = 0
- Gini(S1,S2) = Gini pour une partition de S en deux sous-ensembles S1 et S2 selon un test donné.
- Trouver le branchement (split-point) qui minimise l'indice Gini Nécessite seulement les distributions de classes

Méthodes à base d'arbres de décision

- **CART** (BFO'80 - Classification and regression trees, variables numériques, Gini, Elagage ascendant)
- **C5** (Quinlan'93 - dernière version ID3 et C4.5, attributs d'arité quelconque, entropie et gain d'information)
- **SLIQ** (EDBT'96 — Mehta et al. IBM)
- **SPRINT** (VLDB'96—J. Shafer et al. IBM)
- **PUBLIC** (VLDB'98 — Rastogi & Shim)
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
- **CHAID** (Chi-square Automation Interaction Detection – variables discrètes)

Avantages

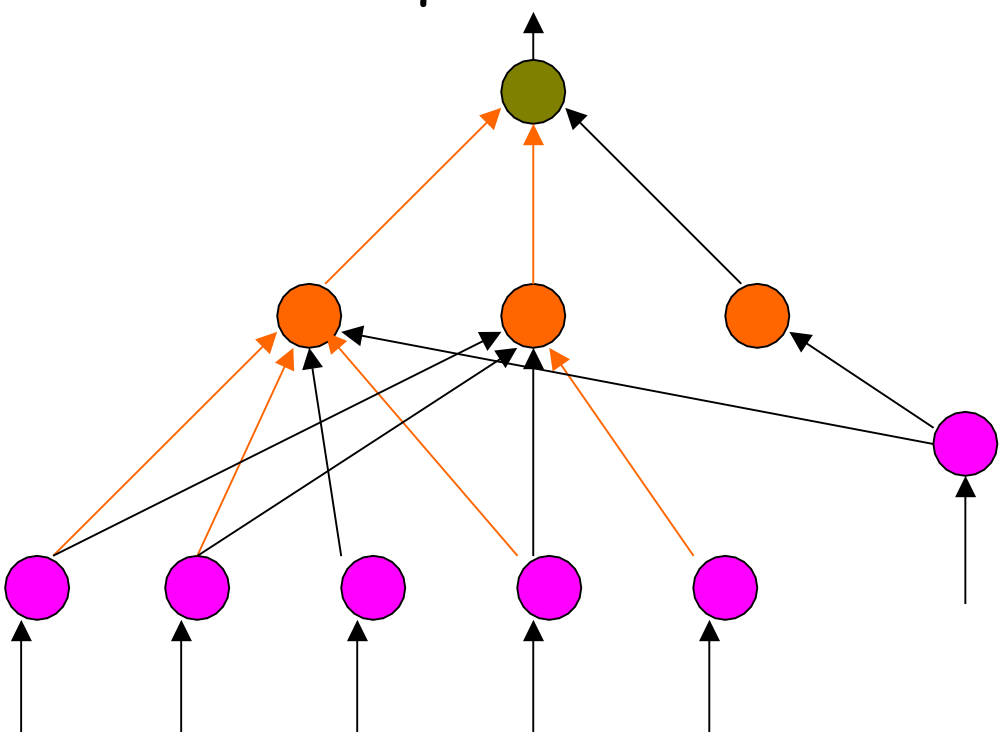
- Compréhensible pour tout utilisateur (lisibilité du résultat – règles - arbre)
- Justification de la classification d'une instance (racine → feuille)
- Tout type de données
- Robuste au bruit et aux valeurs manquantes
- Attributs apparaissent dans l'ordre de pertinence → tâche de pré-traitement (sélection d'attributs)
- Classification rapide (parcours d'un chemin dans un arbre)
- Outils disponibles dans la plupart des environnements de data mining

Inconvénients

- Sensibles au nombre de classes : performances se dégradent
- Evolutivité dans le temps : si les données évoluent dans le temps, il est nécessaire de relancer la phase d'apprentissage

Réseaux de neurones

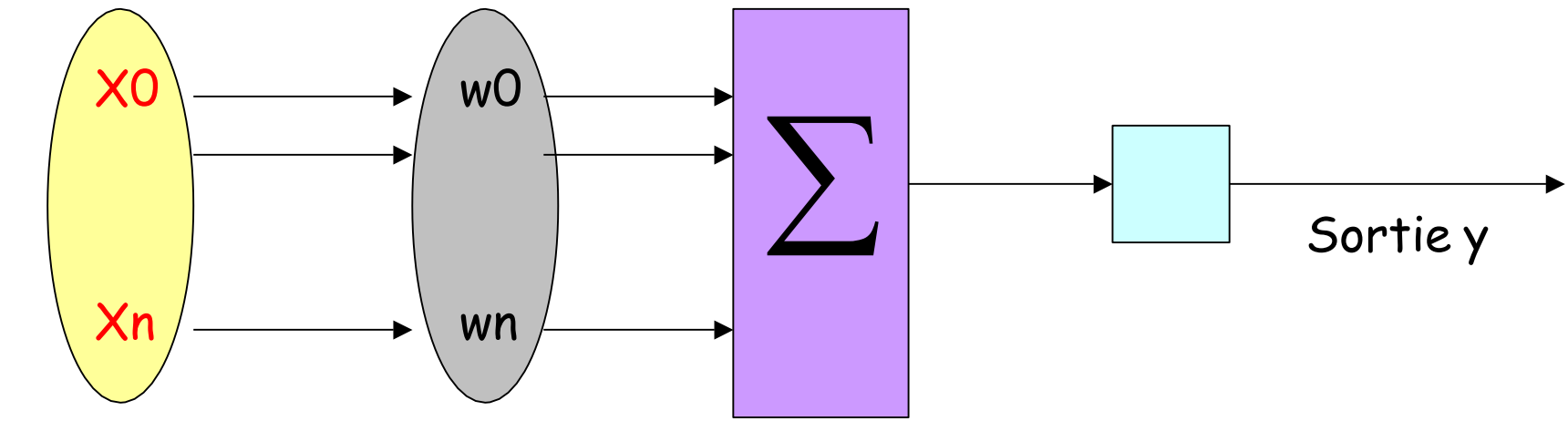
- Réseau neuronal : simule le système nerveux biologique
- Un réseau de neurones est composé de plusieurs neurones interconnectés. Un poids est associé à chaque arc. A chaque neurone on associe une valeur.



- Temps de "switch" d'un neurone > 10^{-3} secs
- Nombre de neurones (humain) ~10^{10}
- Connexions (synapses) par neurone : ~10^4–10^5

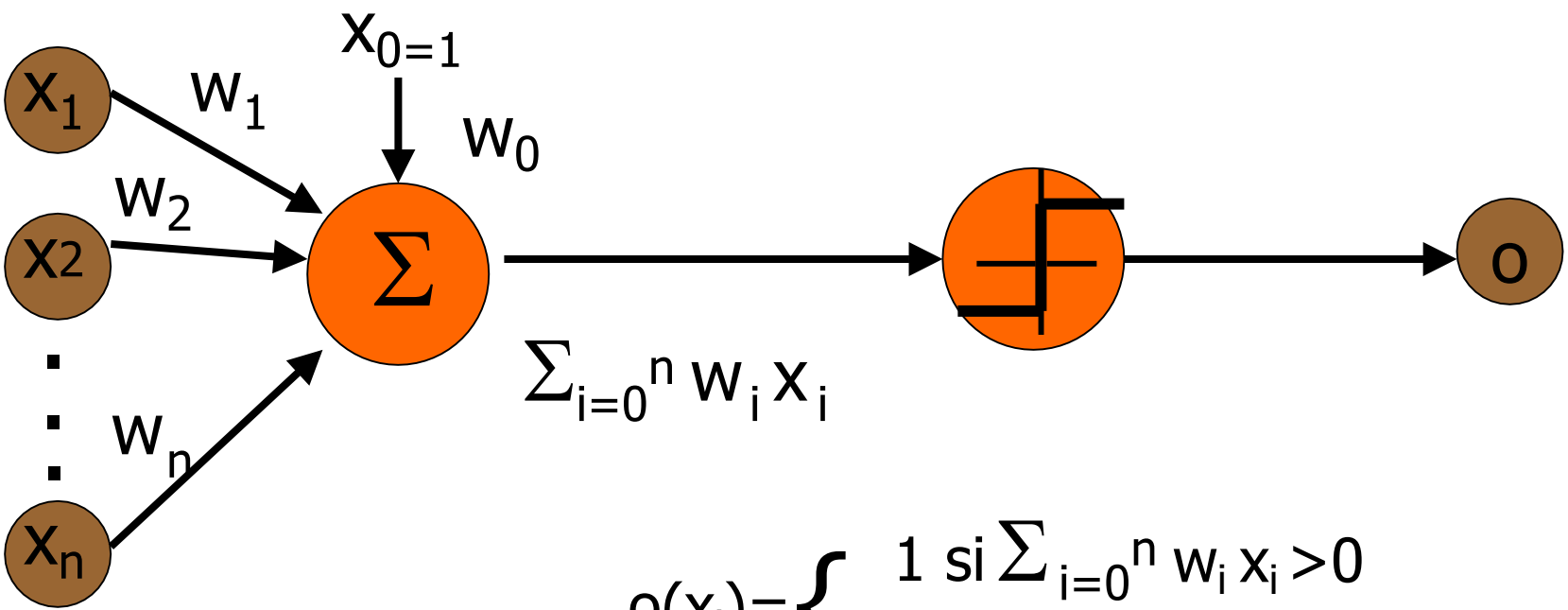
Neurone ou perceptron

- Neurone = Unité de calcul élémentaire
- Le vecteur d'entrée X est transformé en une variable de sortie y, par un produit scalaire et une fonction de transformation non linéaire



Neurone ou perceptron

Linear treshold unit (LTU)



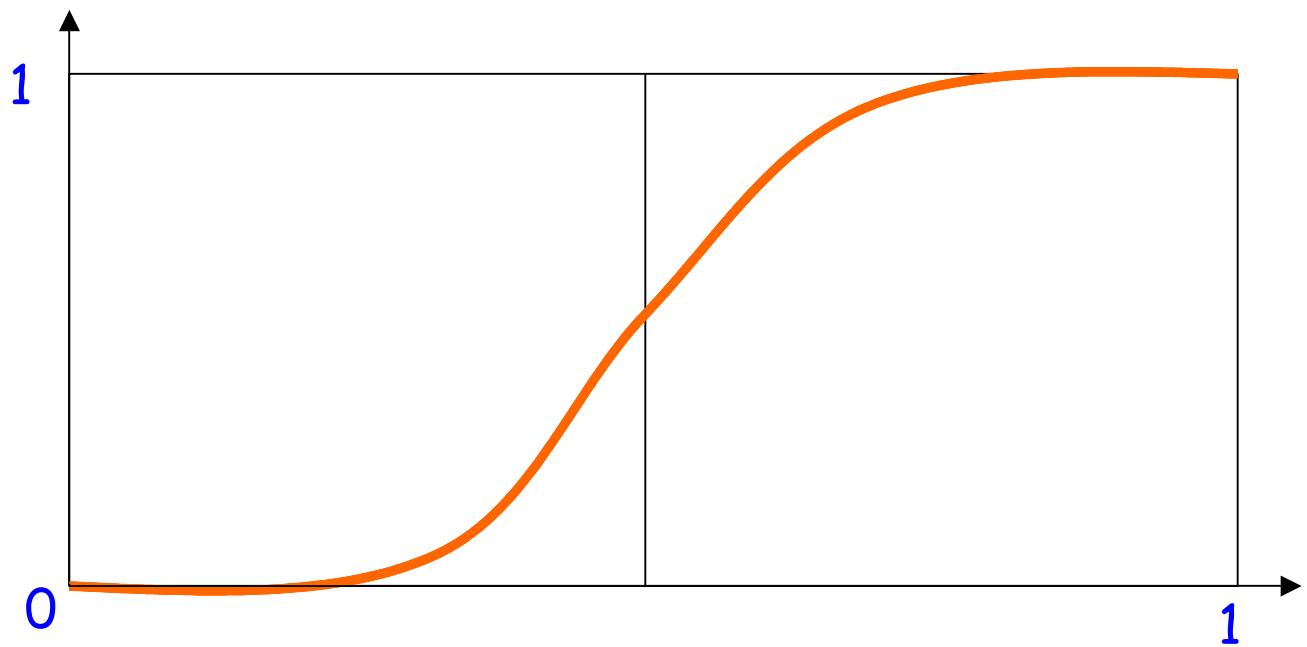
$$o(x_i) = \begin{cases} 1 \text{ si } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ sinon} \end{cases}$$

Neurone

- Fonction d'activation la plus utilisée est la fonction sigmoide

$$\sigma(x)=\frac{1}{1+e^{x}}$$

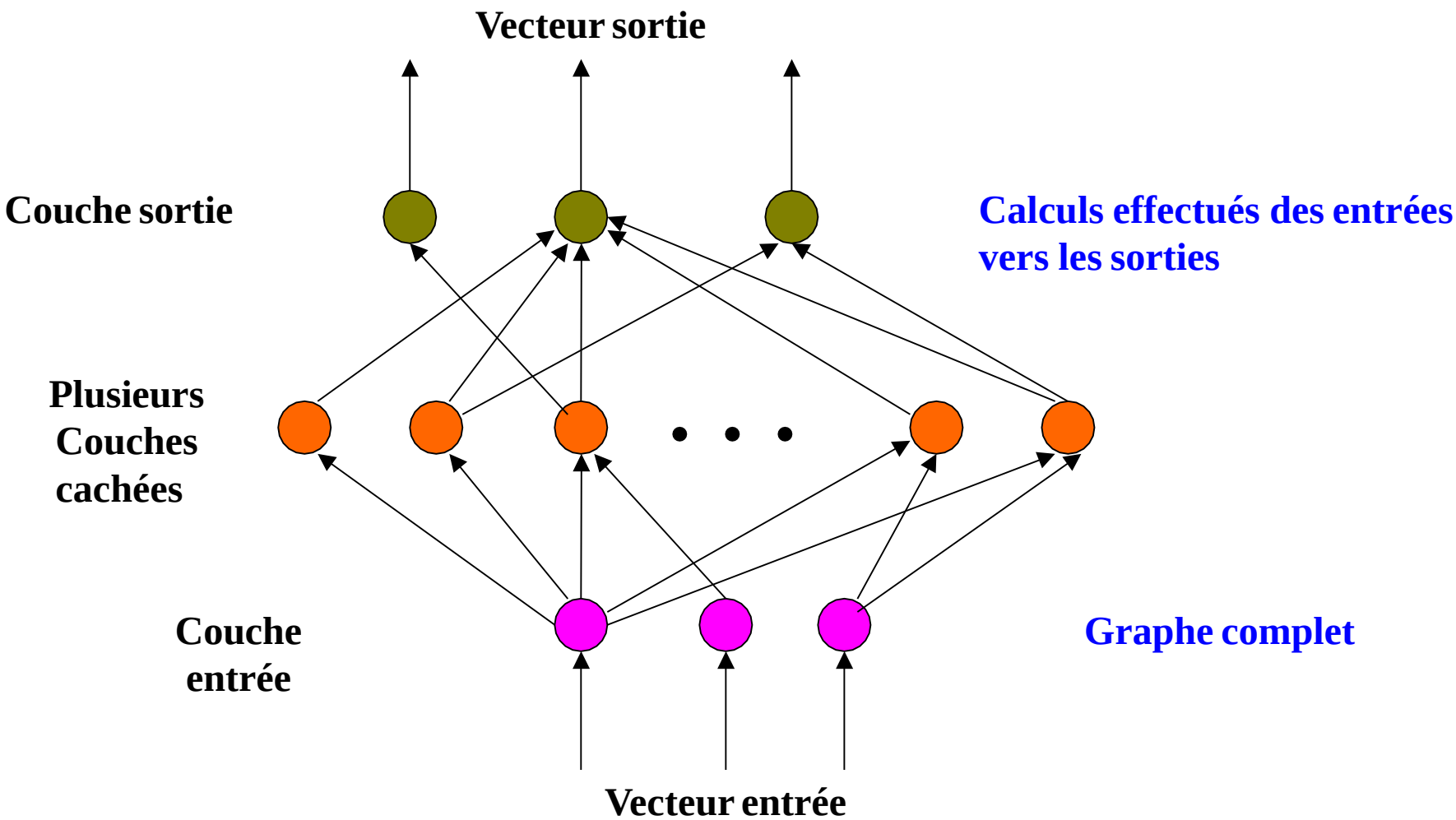
- Elle prend ses valeurs (entrée et sortie) dans l'intervalle [0,1]



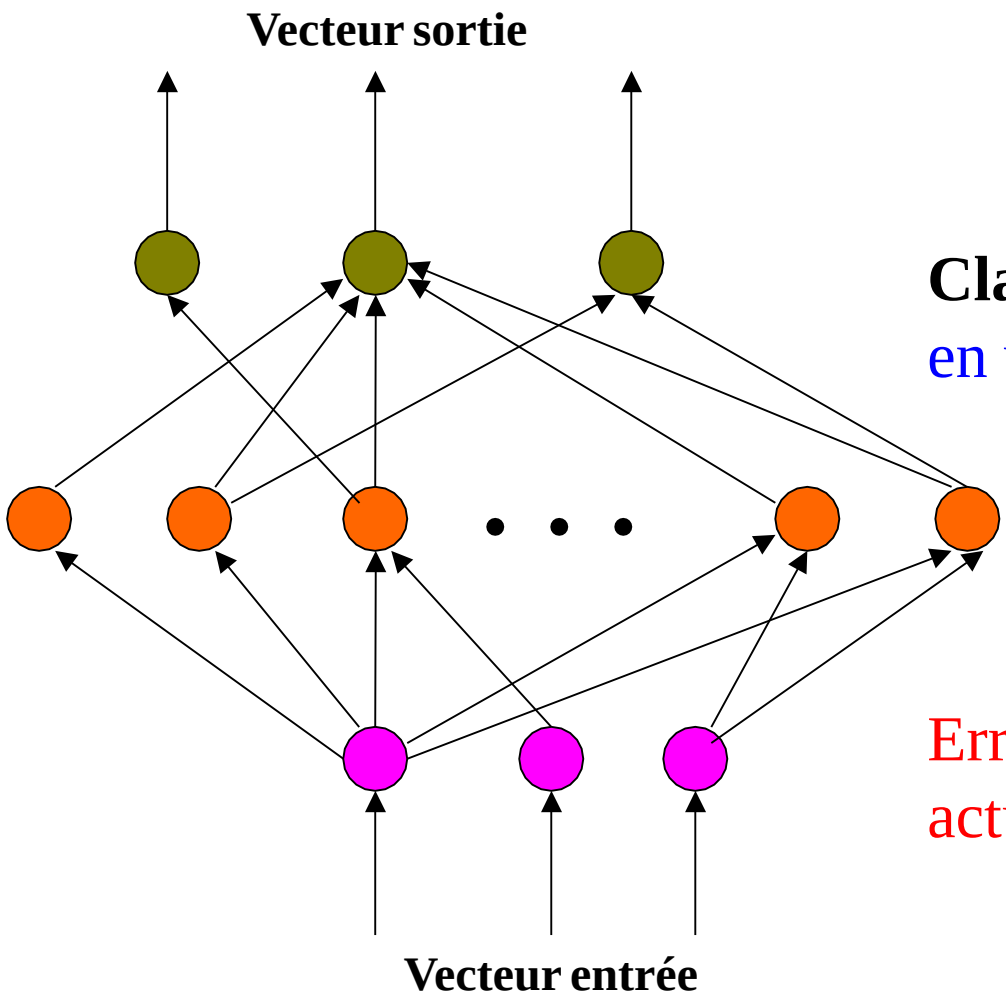
Réseaux de neurones

- Capacité d 'apprentissage : apprendre et changer son comportement en fonction de toute nouvelle expérience.
- Permettent de découvrir automatiquement des modèles complexes.
- Plusieurs modèles de réseaux de neurones : PMC (Perceptron Multi-Couches), RBF (Radial Basis Function), Kohonen, ...

Perceptron Multi Couches (PMC)



Paradigme d'apprentissage



Classification : Ajuster les poids en utilisant l'erreur

Erreur = Valeur désirée – Valeur actuelle

Algorithmes d'apprentissage

- Rétro-propagation du gradient (Back propagation)
- Kohonen
- RBF (Radial basis function)
- Réseaux de neurones probabilistes
- ART (Adaptive resonance theory)
- …

Rétro-propagation du gradient

Principales étapes

- Construction du réseau
  - Représentation des entrées
  - Nombre de couches, nombre de noeuds dans chaque couche
- Apprentissage du réseau utilisant les données disponibles
- Elagage du réseau
- Interprétation des résultats

Construction du réseau

- Nombre de noeuds en entrée : correspond à la dimension des données du problème (attributs ou leurs codages).

Normaliser dans l'intervalle [0,1].

Exemple énumératif : Attribut A prenant ses valeurs {1,2,3,4,5}

- 5 entrées à valeurs binaires ; 3 = 00100
- 3 bits ; 3 = 010
- 1 entrée réelle ; 0, 0.25, 0.5, 0.75, 1

Construction du réseau

- Nombre de couches cachées : Ajuster pendant l'apprentissage.
- Nombre de nœuds par couche : Le nombre de nœuds par couche est au moins égal à deux et au plus égal au nombre de nœuds en entrée
- Nombre de nœuds en sortie : fonction du nombre de classes associées à l'application.

- Réseau riche → pouvoir d'expression grand (Ex. 4-2-1 est moins puissant que 4-4-1)
- Attention : Choisir une architecture riche mais pas trop – Problème de sur-spécialisation

Apprentissage du réseau

- Objectif principal : obtenir un ensemble de poids qui font que la plupart des instances de l'ensemble d'apprentissage sont correctement classées.
- Etapes :
  - Poids initiaux sont générés aléatoirement
  - Les vecteurs en entrée sont traités en séquentiel par le réseau
  - Calcul des valeurs d'activation des nœuds cachés
  - Calcul du vecteur de sortie
  - Calcul de l'erreur (sortie désirée – sortie actuelle).

$$e(PMC)=\frac{1}{2}\sum_{x\in S}\left(d(x)-a(x)\right)^2$$

  - d(x) : sortie désirée, a(x) : sortie actuelle

Apprentissage du réseau

- Les poids sont mis à jour en utilisant l'erreur. Le nombre d'instances qui sont passés dans le réseau avant la mise à jour des poids est un paramètre (convergence rapide et minimum local et convergence lente).

- Rétro propagation à l'aide de la méthode de gradient. Le paramètre taux d'apprentissage [0,1] influe sur la modification des poids.

  Valeur grande : modification forte ; Valeur petite : modification minime

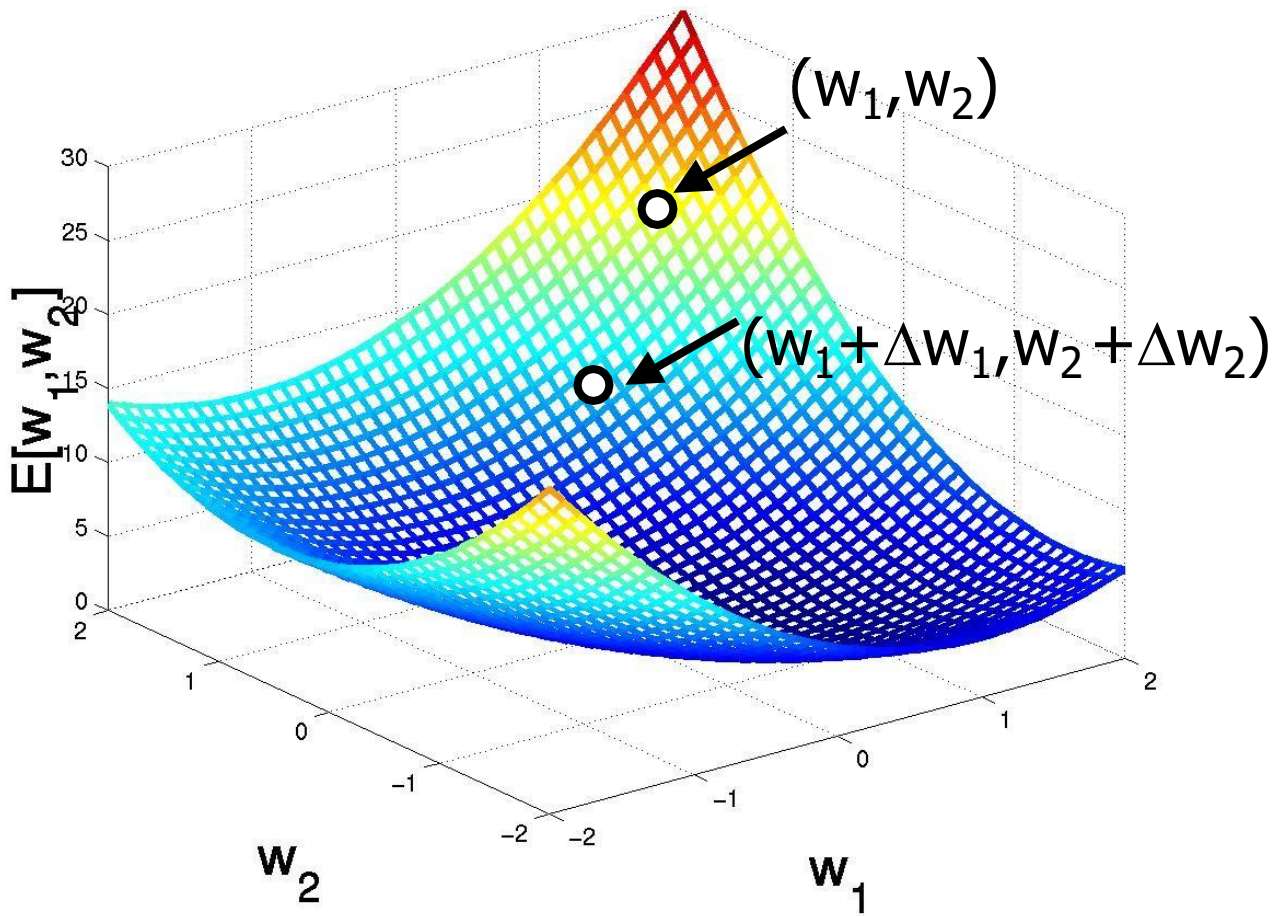
Apprentissage du réseau

$w_i = w_i + \Delta w_i$
$\Delta w_i = \eta\ (t - o)\ x_i$
t=c(x) est la valeur désirée
o est la sortie obtenue
η est le taux d'apprentissage (e.g 0.1)

- Critère d'arrêt : la tolérance définit l'erreur cible.

  et/ou Nombre d'instances bien classées (seuil)

Apprentissage du réseau



Elagage du réseau

- Réseau fortement connexe est difficile à articuler
- N nœuds en entrée, h couches cachées, et m nœuds en sortie→ h(m+n) arcs (poids)
- Elagage : Supprimer les arcs et les nœuds qui n'affectent pas le taux d'erreur du réseau. Eviter le problème de sur-spécialisation (over-fitting).

  Ceci permet de générer des règles concises et compréhensibles.

Réseaux de neurones - Avantages

- Taux d'erreur généralement bon
- Outil disponible dans les environnements de data mining
- Robustesse (bruit) – reconnaissance de formes (son, images sur une rétine, …)
- Classification rapide (réseau étant construit)
- Combinaison avec d'autres méthodes (ex : arbre de décision pour sélection d'attributs)

Inconvénients

- Apprentissage très long
- Plusieurs paramètres (architecture, coefficients synaptiques, …)
- Pouvoir explicatif faible (boite noire)
- Pas facile d'incorporer les connaissances du domaine.
- Traitent facilement les attributs numériques et binaires
- Evolutivité dans le temps (phase d'apprentissage)

Classification bayésienne : Pourquoi ?

- **Apprentissage probabiliste** :
  - calcul explicite de probabilités sur des hypothèses
  - Approche pratique pour certains types de problèmes d'apprentissage
- **Incrémental** :
  - Chaque instance d'apprentissage peut de façon incrémentale augmenter/diminuer la probabilité qu'une hypothèse est correcte
  - Des connaissances a priori peuvent être combinées avec les données observées.

Classification bayésienne : Pourquoi ?

- **Prédiction Probabiliste** :
  - Prédit des hypothèses multiples, pondérées par leurs probabilités.

- **Référence en terme d'évaluation** :
  - Même si les méthodes bayésiennes sont coûteuses en temps d'exécution, elles peuvent fournir des solutions optimales à partir desquelles les autres méthodes peuvent être évaluées.

Classification bayésienne : Pourquoi ?

- Le problème de classification peut être formulé en utilisant les **probabilités a-posteriori** :
  - **P(*C*/*X*)** = probabilité que le tuple (instance) $X=<x_1,\ldots,x_k>$ est dans la classe *C*

- **Idée :** affecter à une instance *X* la classe *C* telle que **P(*C*/*X*)** est maximale

Estimation des probabilités a- posteriori

- **Théorème de Bayes** :
  - $P(C/X) = P(X/C) \cdot P(C) / P(X)$
- $P(X)$ est une constante pour toutes les classes
- $P(C)$ = fréquence relative des instances de la classe C
- C tel que $P(C/X)$ est maximal = C tel que $P(X/C) \cdot P(C)$ est maximal
- **Problème** : calculer $P(X/C)$ est non faisable !

Classification bayésienne naive

- Hypothèse Naïve : indépendance des attributs
  - $P(x_1,\ldots,x_k/C) = P(x_1/C)\cdot\ldots\cdot P(x_k/C)$

  $P(x_i/C)$ est estimée comme la fréquence relative des instances possédant la valeur x_i (i-ème attribut) dans la classe C

- Non coûteux à calculer dans les deux cas

Classification bayésienne – l'hypothèse d'indépendance

- … fait que le calcul est possible
- … trouve un modèle de classification optimal si hypothèse satisfaite
- … mais est rarement satisfaite en pratique, étant donné que les attributs (variables) sont souvent corrélés.
- Pour éliminer cette limitation :
  - **Réseaux bayésiens**, qui combinent le raisonnement bayésien et la relation causale entre attributs
  - **Arbres de décision**, qui traitent un attribut à la fois, considérant les attributs les plus importants en premier

Autres méthodes de classification

- Réseaux bayésiens
- Algorithmes génétiques
- Case-based reasoning
- Ensembles flous
- Rough set
- Analyse discriminante (Discriminant linéaire de Fisher, Algorithme Closest Class Mean - CCM-)

Classification - Résumé

- La **classification** est un problème largement étudié
- La **classification**, avec ses nombreuses extensions, est probablement la technique la plus répandue

  - Modèles
    - Arbres de décision
    - Règles d'induction
    - Modèles de régression
    - Réseaux de neurones

Facile à comprendre

Difficile à comprendre

Classification - Résumé

- **L'extensibilité reste une issue importante pour les applications**

- **Directions de recherche : classification de données non relationnels, e.x., texte, spatiales et données multimédia**