Classification supervisée

N. TSOPZE

Département d'Informatique - Université de Yaoundé I

Plan

Plan

Introduction

- Séparation en n catégories; $n = 2$ classification binaire ou $n > 2$ multiclasse.
- Désigne classification supervisée (Classification en Anglais ou problème de décision en statistique) et classification non supervisée (Clustering en Anglais ou classification en statistique).
- Le nombre de classes est fini.
- donnée x_i est caractérisée par P attributs et par sa classe $yi \in Y$ (y_i n'existe pas en classification non supervisée)
- Exemple: classer les clients entre les 'bons' et les 'mauvais'.

Classification supervisée

problème

Le problème consiste alors, en s'appuyant sur l'ensemble d'exemples $X = \{(x_i, y_i) i \in \{1, ..., N\}\}$,à prédire la classe de toute nouvelle donnée $x \in D$.

Utilisation d'un ensemble d'exemples classés pour prédire la classe de nouvelles données ; "apprentissage à partir d'exemples", ou "apprentissage supervisé".

Classeur

procédure (un algorithme) qui, à partir d'un ensemble d'exemples, produit une prédiction de la classe de toute donnée.

Deux phases:

1. phase d'apprentissage ou de construction du modèle
2. phase de test.

Classification supervisée

- Repartir les données en catégories mutuellement exclusives et exhaustives (Classes).
- Chaque exemple DOIT appartenir à une et une seule classe.
- Deux étapes: apprentissage (+validation) + classement
- Quelques méthodes de résolution: kppv, réseaux bayesiens, réseaux de neurones, arbres de décisions, SVM,...

Classification supervisée

Validation d'un classifieur

1. Disponibilité:
   - ensemble d'apprentissage + ensemble de test
   - Mesure de convergence
2. Attendue:
   - Paramètres d'un classifieur
   - Mesures de performances

Classification supervisée

Validation

Calcul de la probabilité que la classe prédite pour une donnée quelconque soit correcte

E =erreur de classification = erreur en généralisation = taux d'échec

Erreur de classification

E = probabilité que ce classeur ne prédise pas correctement la classe d'une donnée.

Le taux de succès $= 1 - E$.

erreur apparente $= E_{app}$= erreur d'entraînement = erreur d'apprentissage

Classification supervisée

On sépare le jeu de données en deux:

1. le jeu d'exemples d'apprentissage (noté X_{app}) ou d'entraînement sert à la construction du modèle.
2. le jeu d'exemples de test (noté X_{test}) pour estimer les erreurs; on connaît leur classe. On obtient alors E_{test}

Mesures

1. VP : vrais positifs : exemples positifs prédits comme positive ;
2. VN : vrais négatifs : exemples négatifs prédits comme négative ;
3. FP : faux positifs : exemples négatifs prédits comme positive ;
4. FN :faux négatifs : exemples positifs prédits comme négative.

matrice de confusion

	positifs	négatifs
positifs	VP	FN
négatifs	FP	VN

Table: Matrice de confusion

1. précision
   - précision pour les positifs $= VP/(VP + FP)$
   - précision pour les négatifs $= VN/(VN + FN)$
2. rappel ;
   - rappel pour les positifs $= VP/(VP + FN)$
   - rappel pour les positifs $= VN/(VN + FP)$
3. Mesure F

$$F = \frac{2 rappel \times precision}{rappel + precision} = \frac{2VP}{2VP + FP + FN}$$

Généralités

1. Holdout : Diviser le jeu de données en 2, construire le modèle avec le jeu d'apprentissage et tester avec le jeu de test
2. Validation croisée d'ordre k : diviser le jeu de données en k sous ensembles, itérer k fois la construction et le test du modèle. Prendre la moyenne des taux d'erreur.
3. leave-one-out : validation croisée d'ordre n, n est le nombre d'exemples.

Méthodes de classification supervisée

Deux catégories:

1. Symboliques: Ensemble de règles
   - Arbres de décision
   - règles de classification
2. Statistiques
   - Réseaux de neurones
   - Réseaux bayésiens
   - Plus proches voisins

Plan

Arbres de décision

Représentation graphique du processus de classification

- chaque nœud correspond à un test sur la valeur d'un ou plusieurs attributs ;
- chaque branche partant d'un nœud correspond à une ou plusieurs valeurs de ce test ; a chaque feuille est associée une valeur de l'attribut cible Décision

L'arbre de décision peut être ensuite exploité de différentes manières :

- en *y* classant de nouvelles données ;
- en faisant de l'estimation d'attribut;
- en en extrayant un jeu de règles de classification concernant l'attribut cible;
- en interprétant la pertinence des attributs.

Construction de l'arbre

Principe général:

1. Recherche de la variable et du seuil qui sépare le mieux
2. Applique la séparation à la population
3. Obtention de nouveaux nœuds
4. Arrêt de l'approfondissement de l'arbre lorsque les conditions d'arrêts sont rencontrées
5. Éventuel élagage de l'arbre

Quelques algorithmes : ID3, CART, C4.5

Algorithme ID3

Entropie (binaire)

L'entropie $H(X)$ de X est : $H(X) = -p_+ log_2 p_+ - p_- log_2 p_-$.
$p_+ + p_- = 1$.

Remarques:

1. si $p_+ = 0$ ou $p_- = 0$, alors $H(X) = 0$.
2. si $p_+ = p_- = 0,5$, alors $H(X) = 1$.

Entropie générale

L'entropie de l'ensemble d'exemples X est : $H(X) = -\sum p_i log_2 p_i$

Gain d'information dû à l'attribut a_j

$$Gain(X, a) = H(X) - \sum_{v \in valeurs(a)} \frac{\|X_{a_j=v}\|}{\|X\|} H(X_{a_j=v})$$

Algorithme ID3

Algorithm 1 Algorithme ID3

1: Créer un nœud racine
2: Si tous les éléments de X sont d'une classe c alors racine.étiquette $\leftarrow c$ et retourner racine
3: Si $A = \{\}$ alors racine. étiquette $\leftarrow$ valeur la plus présente de la classe parmi les X et retourner racine
4: $a* \leftarrow argmax_{a \in A} Gain(X, a)$; racine. étiquette $\leftarrow a*$
5: **for** valeurs v_i de $a*$ **do**
6: ajouter une branche à racine correspondant à la valeur v_i
7: former $X_{a=v_i} \subset X$ dont l'attribut $a*$ vaut v_i
8: Si $X_{a*=vi} = \{\}$ alors à l'extrémité de cette branche, mettre une feuille étiquetée avec la valeur la plus présente de la classe parmi les X
9: Sinon à l'extrémité de cette branche, mettre $ID3(X_{a*} = v_i, A\{a*\})$
10: **end for**
11: retourner racine

Classement d'un exemple

Algorithm 2 Classification d'une donnée dans un arbre de décision

Require: arbre de décision AD, exemple x
Ensure: classe de x.

1: $nc \leftarrow racine(AD)$
2: **while** $nc \neq feuille$ **do**
3: en fonction de l'attribut testé dans nc et de sa valeur dans x, suivre l'une des branches de nc. Le nud atteint devient nc
4: **end while**
5: retourne nc

1. La différence entre les algorithmes résident au niveau de la méthode du choix de l'attribut et du type des attributs
2. ID3 est limité aux attributs symboliques contrairement aux autres

AD - Attributs numériques

1. Trier les attributs et associer le numéro de son exemple associé ainsi que la valeur de l'attribut cible
2. détermine le seuil s pour partitionner cet ensemble d'exemples. C4.5 utilise les règles suivantes :
   1. ne pas séparer deux exemples successifs ayant la même classe ;
   2. si on coupe entre deux valeurs v et w $(v < w)$ de l'attribut le seuil s est fixé à v (on aurait pu aussi utiliser $(v + w)/2$);
   3. choisir s de telle manière que le gain d'information soit maximal.

Algorithme C4.5 - Rapport de Gain

$$Rapport\ de\ gain(X, a) = \frac{Gain(X,a)}{SplitInfo(X,a)}$$

où $SplitInfo(X, a) = \sum_{x \in valeurs(a)} = \frac{|X_{a=v}|}{|X|} log_2 \frac{|X_{a=v}|}{|X|}$ Choisir la variable qui maximise le Rapport de gain

Algorithme CART - Indice de Gini

$IG = 1 - \sum_i^n f_i^2$ où f_i est la fréquence de la classe i

Choisir la variable qui maximise
$IG(avant\ sep.) - (IG(fils1) + IG(fils2))$

Extraction des règles à partir des AD

Règles des AD

Mettre sous forme d'un ensemble de règles, le processus de décision de l'arbre de décision

écrire sous forme de règles tous les chemins allant de la racine à chaque feuille de l'arbre. Chaque chemin décrit est alors une règle de classification avec pour conclusion la feuille (représentant une classe) et la reste du chemin forme la prémisse.

Règles de classification

$$si\ condition(x)\ alors\ classe(x) = \ldots$$

où : x est une donnée ; $condition(x)$ est une condition exprimée sur les attributs de la donnée x de la forme "attribut = valeur" est une valeur possible pour la classe: "vrai" ou "faux" si on est dans un problème de classification binaire par exemple
Mésures:

1. La couverture: pourcentage de tuples qui vérifient la règles
2. La précision: rapport entre le nombre de tuples bien classés et la couverture de la règle.

Détermination

1. L'utilisation d'une règle de classification est tout à fait intuitive: si la condition est vérifiée sur une certaine donnée, alors on en prédit sa classe.
2. Pour déterminer la classe d'une donnée, on utilise un ensemble de règles de classification. Ces règles sont ordonnées (numérotées):
   - pour classer une donnée, on regarde si la donnée vérifie la condition de la première règle ;
   - si elle est vérifiée, on applique la conclusion de la règle ;
   - sinon, on regarde si la donnée vérifie la condition de la deuxième règle ;
   - si elle est vérifiée, on applique sa conclusion ;
   - etc.

Plan

Neurone biologique

Neurone biologique

Cellule du système nerveux

Neurone biologique

Neurone biologique

Cellule du système nerveux

Parties:

- Dendrites
- Noyau
- Synapses
- ...

Ses états:

- Excité
- Inhibé

Neurone biologique

Neurone biologique

Cellule du système nerveux

Parties:

- Dendrites
- Noyau
- Synapses
- ...

Ses états:

- Excité
- Inhibé

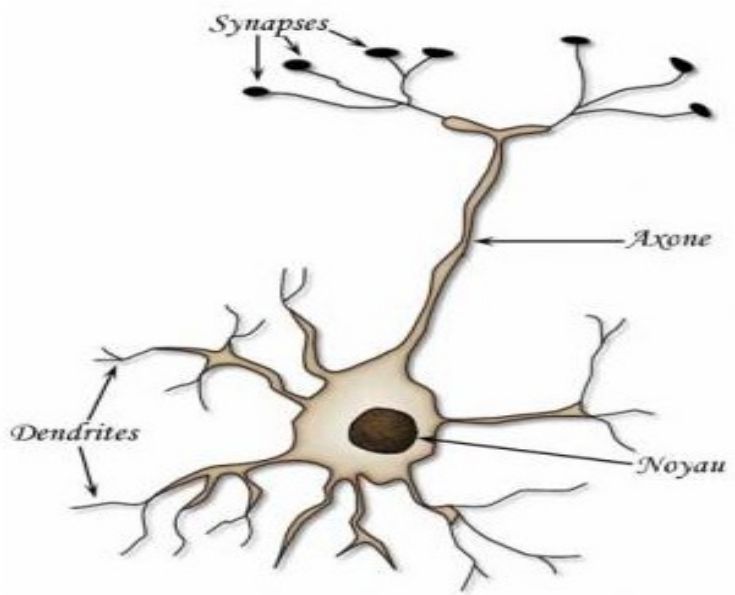


Schéma d'un neurone biologique.

Réseaux de Neurones Artificiels

Neurone (artificiel)

Modèle mathématique de la cellule du cerveau (neurone biologique)

Unité de traitement capable d'échanger l'information avec son environnement.

Ses composants sont:

- Etat interne x: inhibé ($x = 0(-1)$) ou excité ($x = 1$).
- Seuil d'activation: θ
- Fonction d'activation :
  - fonction sigmoide: $f(x) = \frac{1}{1+e^{-x}}$
  - fonction de Heavyside: $f(x) = 1$ *si* $x \geq x_0$ *sinon* 0
  - fonction tangent hyperbolique:$f(x) = tanh(x) = \frac{1-e^{-x}}{1+e^{-x}}$
- Ensemble d'entrées (généralement combinée à la rétine).
- Fonction de combinaison : combinaison linéaire, distance,...

Neurone formel

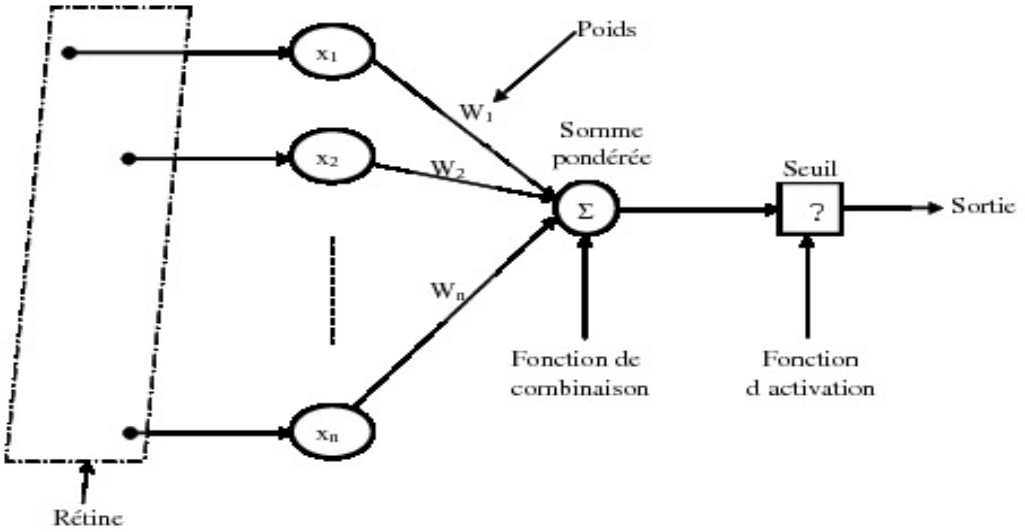
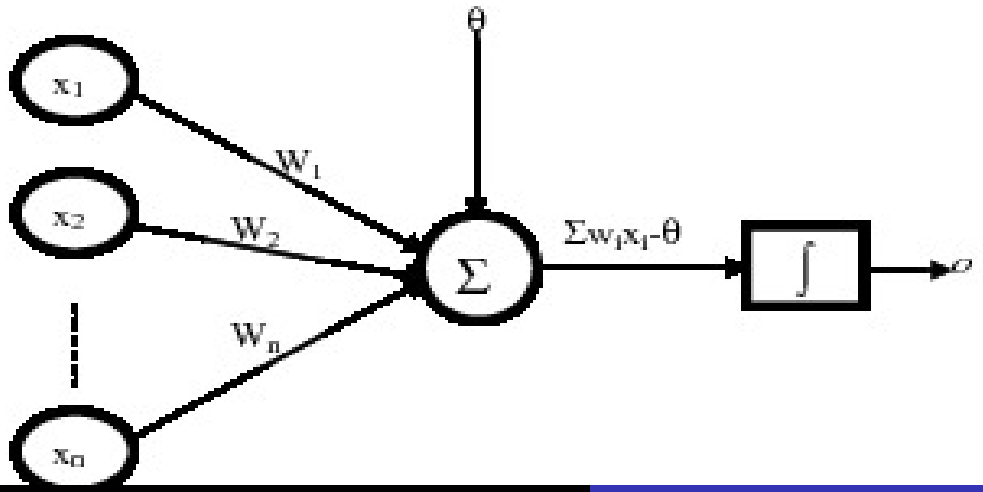
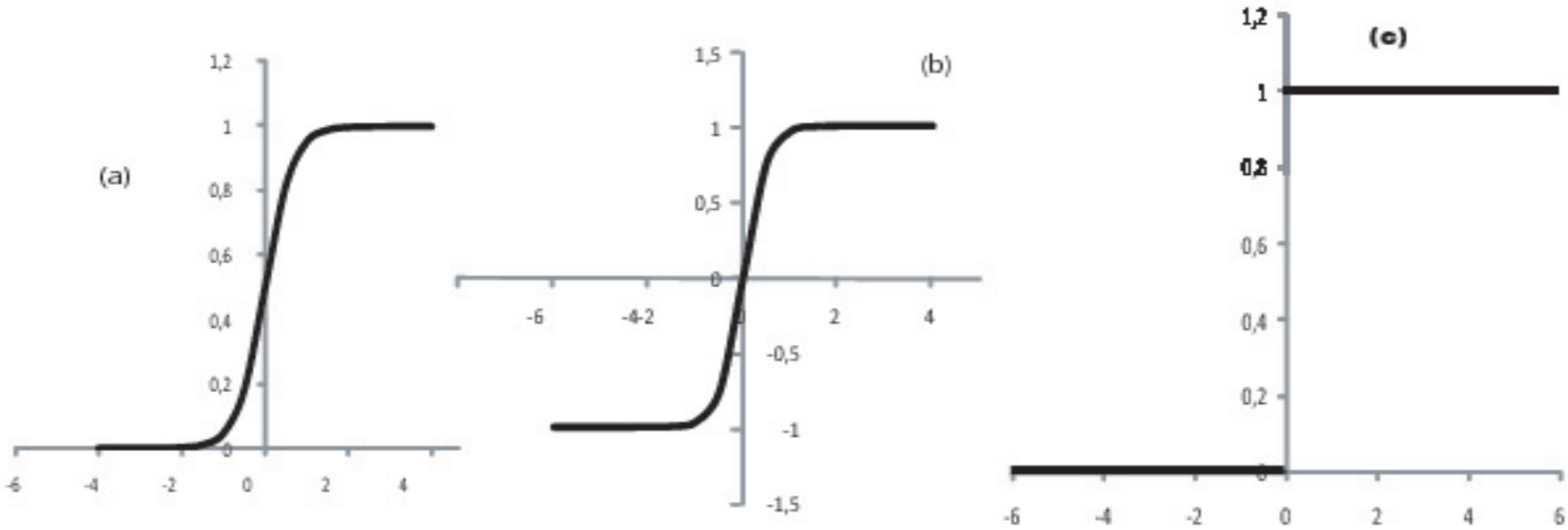


Schéma d'un neurone



Fonction d'activation



graphique de quelques fonctions d'activation

- Gauche (a): Fonction Sigmoïde.
- Milieu (b): Tangente hyperbolique.
- Droite (c): Fonction de Heavyside.

Apprentissage

Algorithm 3 Détermination des poids d'un perceptron

Require: taux d'apprentissage $\alpha \in]0,1[$, ϵ la mesure d'erreur

initialiser les w_i aléatoirement
repeat
  $E \leftarrow 0$ // Erreur courante
  Rendre aléatoire l'ordre des exemples
  for tous les exemples du jeu d'apprentissage X **do**
    $E_i \leftarrow y_i - f(\sum_{i=1}^{n} w_i x_i - \theta)$
    $E \leftarrow E + E_i$
    for tous les poids w_k; **do**
      $w_k \leftarrow w_k + \alpha E_i x_i$
    end for
  end for
until $E < \epsilon$

Perceptron

Limites:

- limité au problème de classification binaire
- ne converge pas lorsque les données ne sont pas linéaiement séparables

Résolution

Ajout des couches intermédiaires entre la couche d'entrée et la couche de sortie dans le but de reduire l'erreur en sortie

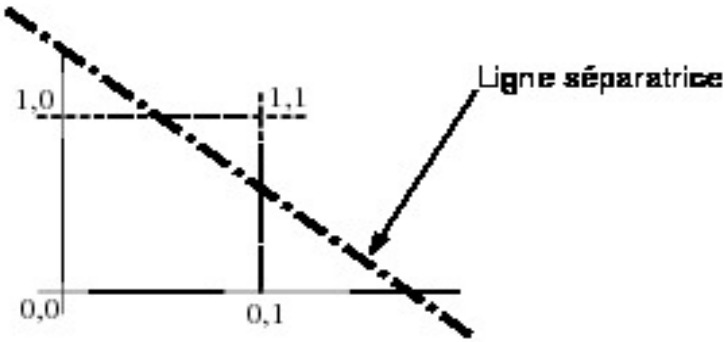
Exemples

1. Fonction booléenne ET(x,y)
2. Fonction booléenne OU(x,y)
3. Fonction booléenne XOR(x,y) - OU exclusif.

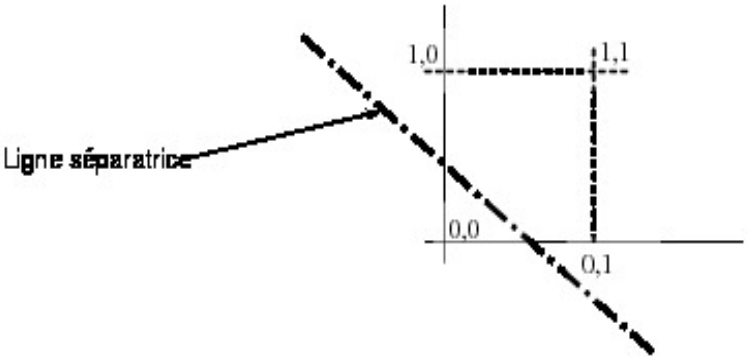
limites

convergence en présence des données linéairement séparables, pas de convergence dans le cas contraire

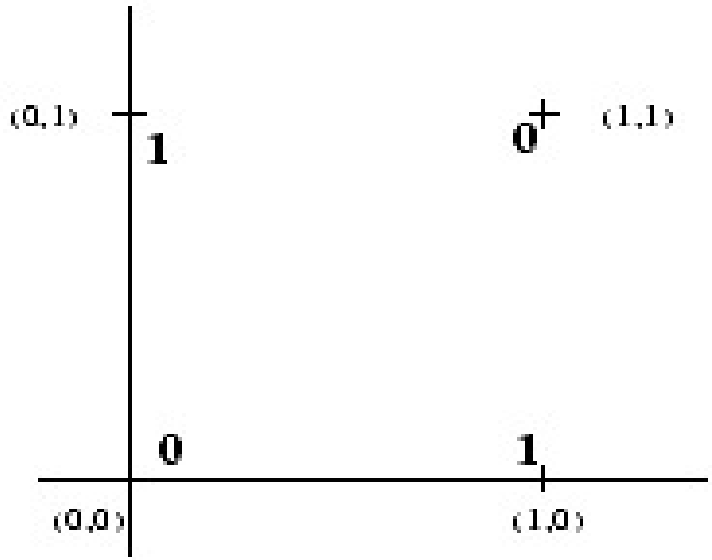
Exemples



La fonction logique ET (x,y)



La fonction logique OU (x,y)



La fonction logique OU exclusif.

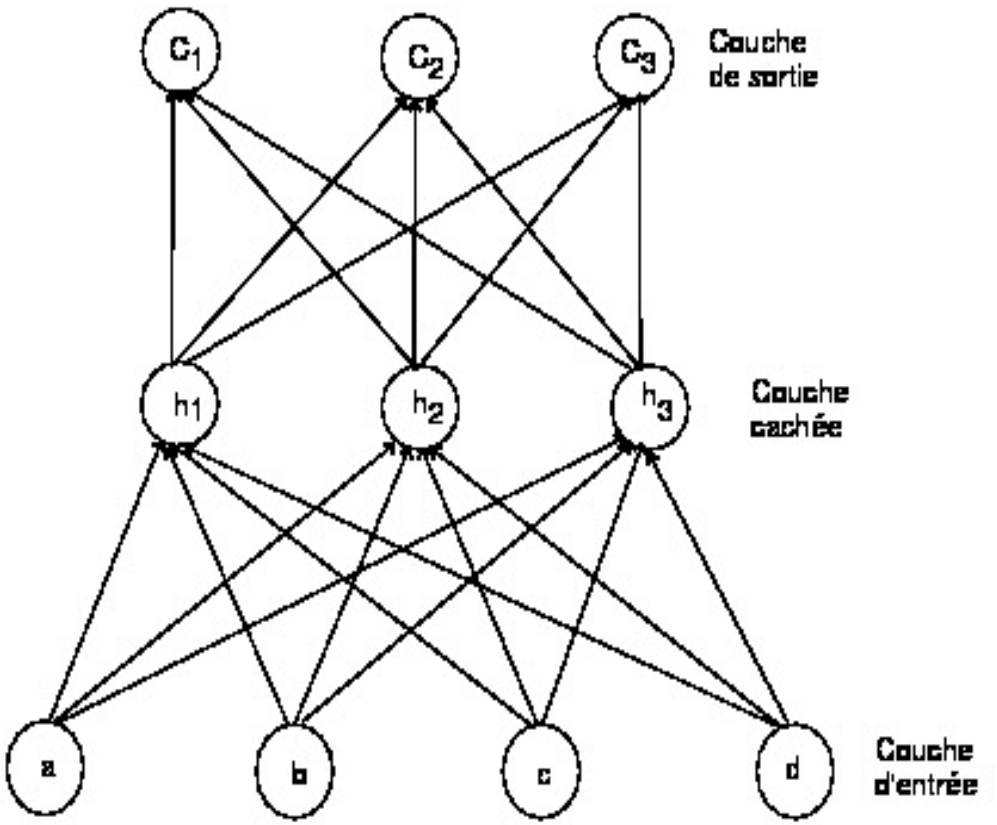
Réseau de Neurones Artificiels (RNA)

Ensemble de neurones inter connectés qui échangent les informations entre eux et aussi avec l'environnement extérieur.

Caractéristiques

- Connexions pondérées entre neurones: $w_{i,j}$ entre le neurone i et le neurone j.
- Etat interne des neurones : fonction de combinaison de ses entrées.
- Disposition éventuelle des neurones en couches:
  - Couche d'entrée: reçoit les signaux de l'extérieur
  - Couche de sortie: émet les décisions vers l'extérieur
  - Couche intermédiaire (cachée): fait des calculs internes.

Exemple de RNA



Un réseau de neurones artificiels.

Modèles de RNA

RNA non bouclés

L'information circule des entrées vers les sorties (sans retour)

RNA bouclés

L'information circule des entrées vers les sorties et peut aussi circuler des sorties vers les entrées

autres modèles

- Modèles de Hopfield
- Modèles de Kohonen (SOM)
- Mémoires associatives

Modèles de RNA

RNA non bouclés

L'information circule des entrées vers les sorties (sans retour)

RNA bouclés

L'information circule des entrées vers les sorties et peut aussi circuler des sorties vers les entrées

autres modèles

- Modèles de Hopfield
- Modèles de Kohonen (SOM)
- Mémoires associatives

Les RNA sont utilisés lorsque établir l'expression analytique d'une fonction f n'est pas facile, trouver (par apprentissage) une fonction f^* qui approxime au mieux la fonction f .

Apprentissage des RNA

Plusieurs algorithmes sont utilisés pour l'apprentissage des RNA; l'utilisation de ces algorithmes dépend de l'architecture des RNA et leur convergence de la séparabilité linéaire des données:

- l'algorithme d'apprentissage du perceptron $\rightarrow$ réseau monocouche;
- l'algorithme 'Pocket perceptron with rachet modification' $\rightarrow$ réseau monocouche;
- l'algorithme du rétropropagation du gradient de l'erreur $\rightarrow$ réseau multicouche.

Algorithme de rétropropagation du gradient

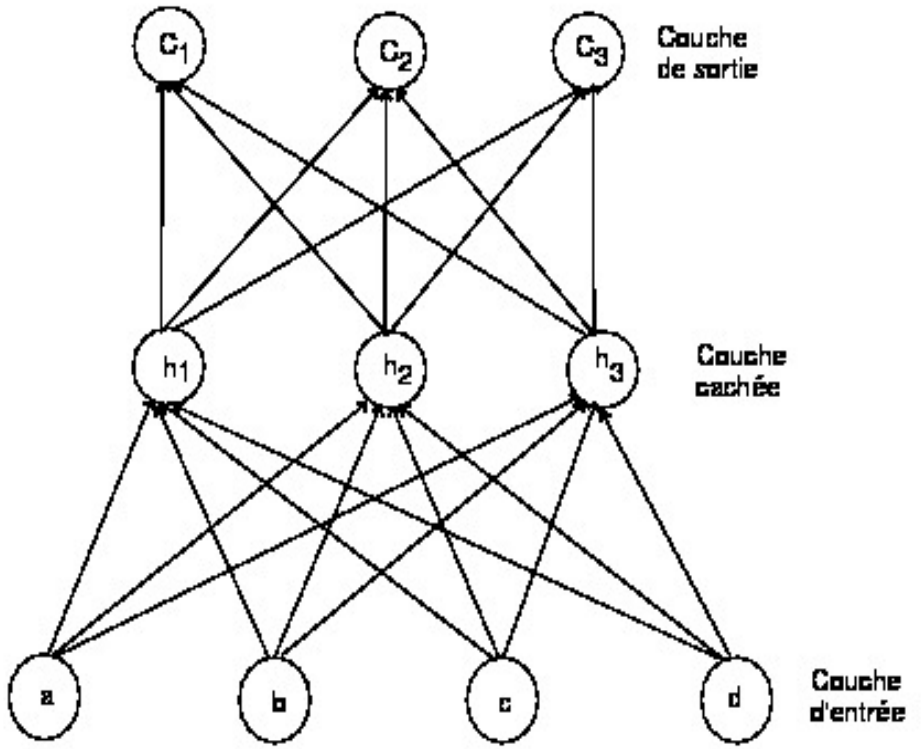


Figure: Exemple de RNA à 3 couches

Algorithme de rétropropagation du gradient

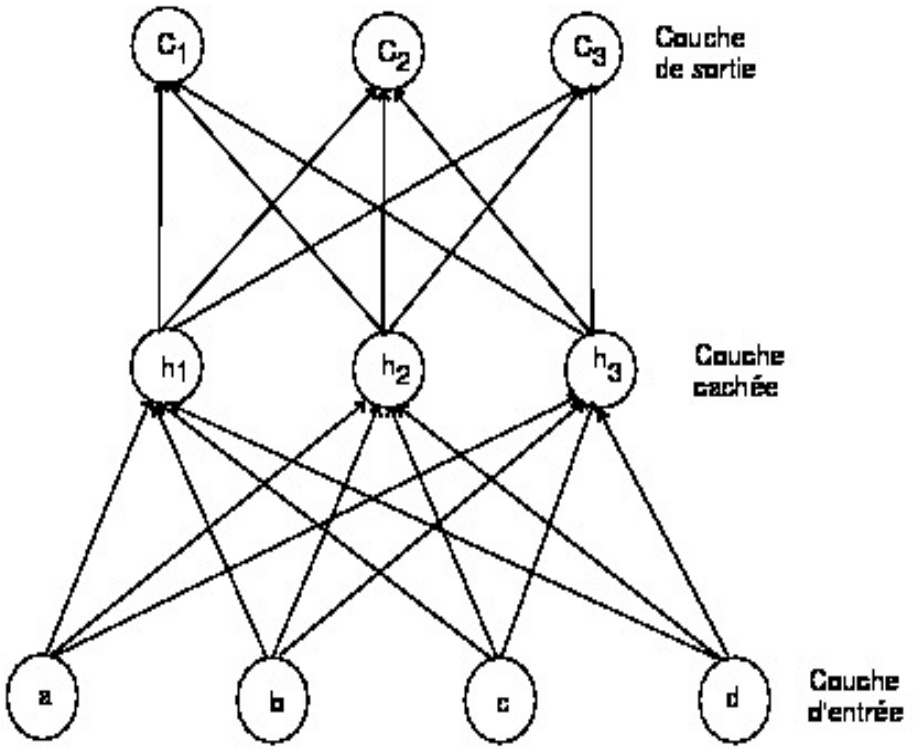


Figure: Exemple de RNA à 3 couches

Algorithme de rétropropagation du gradient

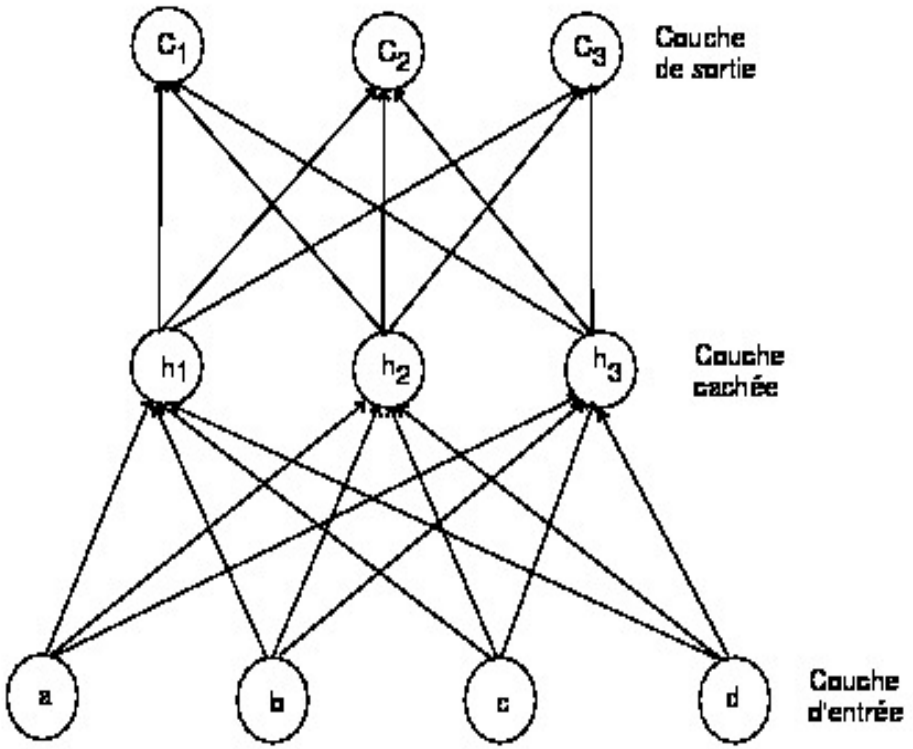


Figure: Exemple de RNA à 3 couches

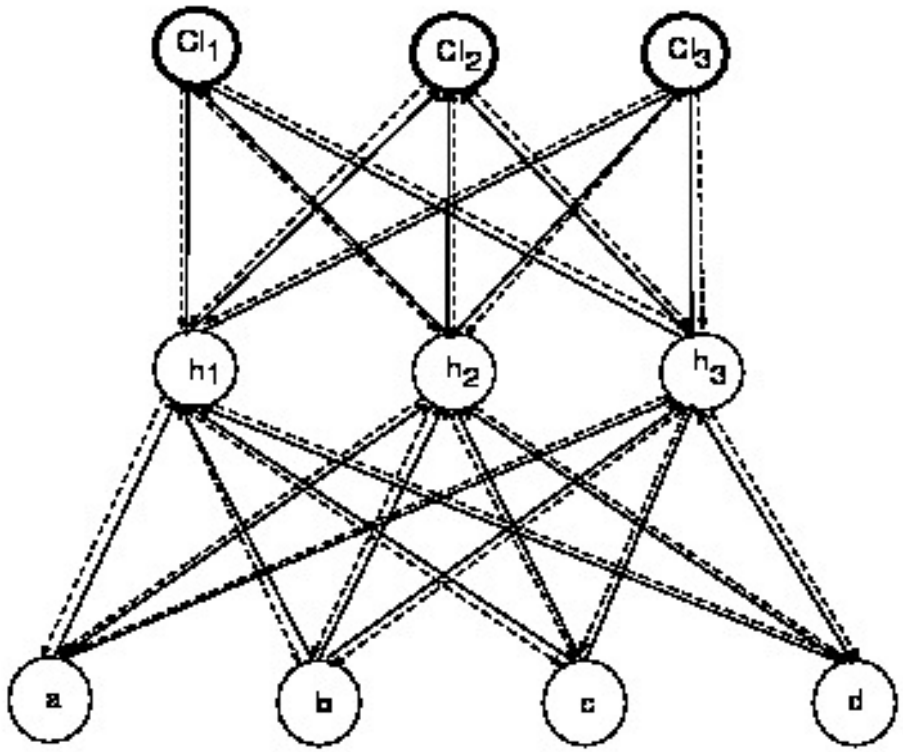


Figure: Rétropropagation de l'erreur

RNA multicouche MLP

but

Minimiser l'erreur en cas de non-séparabilité linéaire de l'ensemble d'apprentissage.

L'apprentissage se fait par rétro propagation du gradient de l'erreur.

Algorithme d'apprentissage des RNA multicouches

Algorithme de rétropropagation de l'erreur

1. Initialiser les poids de manière aléatoire
2. Repeter
   1. Pour chaque exemple faire
      1. Placer cet exemple en entrée du réseau
      2. Propager le signal dans le réseau jusqu'à la sortie
      3. Calculer l'erreur en sortie
      4. Calculer l'erreur dans les autres couches an allant de la sortie vers les entrées
      5. Modifier les poids de connexion
   2. Finpour
3. jusqu'à la convergence

L'algorithme de rétropropagation produit des bons résultats mais:

- Quel est le nombre approprié de couches?
- Quel est le nombre de neurones par couche?
- Comment connecter les neurones?
- Comment les initialiser?
- ...

Les réponses à ces questions constituent l'architecture du RNA.

Utilisation des RNA

Deux phases:

1. Construction du réseau
   - définition de nombre de couches
   - définition du nombre de neurones pas couches
   - initialisation des paramètres
2. Apprentissage et test du réseau
   - Recherche des 'bons' poids
   - évaluation du réseau sur les données 'connues'
   - mesures
3. Utilisation: proposition d'une classe une nouvelle donnée.

Réseaux bayésiens

- Classifieur bayésien naïf: variables sont indépendantes
- Réseaux d'inférence bayésien ou réseau causal
- Repose sur le théorème de Bayes sur le calcul de propabilités
- Possibilité d'intégrer la connaissance a priori

Réseaux bayésiens

Théorème de Bayes:

$$P(A/B) = \frac{P(B/A) \times P(A)}{P(B)}$$

- $P(A)$ (resp. $P(B)$: probabilité a priori d'obtenir l'évènement A (resp. B)
- $P(B/A)$: vraisemblance de l'évènement B si A est vérifié

En classification, on suppose les évènements:

- c_i : observer la classe i
- X ; observer la donnée X

$\Longrightarrow$ calculer la probabilité $P(c_i/X) = \frac{P(X/c_i)P(c_i)}{P(X)}$

Réseaux bayésiens

$$P(c_i/X) = \frac{P(X/c_i)P(c_i)}{P(X)}$$

- $P(c_i)$: probabilité à priori d'observer la classe c_i
- $P(X)$: probabilité à priori d'observer le vecteur X
- $P(X/c_i) = \prod_{i=1}^{N} P((a_i = v_i)/c_i)$

Face à la difficulté de calculer la probabilité $P(X)$, on utilise:

$$P(c_i/X) = \frac{P(X/c_i) \times P(c_i)}{\sum_{c_i} P(X/c_i) \times P(c_i)}$$

La classe de X est celle qui maximise la probabilité à postéori et donnée par :

$$C_{MAP} = argMax_{c_i} P(c_i/X) = argMax_{c_i} P(X/c_i) \times P(c_i)$$

Si on ne tient pas compte de $P(c_i)$, alors on parle de Maximum de Vraissemblance:

$$c_{ML} = argMax_{c_i} P(X/c_i)$$

Si les exemples sont équitablement répartis entre les classe alors $c_{MAP} = c_{ML}$

Attributs continus: On suppose que la distribution suit la loi normale:

$$P(a = x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}} dx$$

KPPV - KNN

- Encore appelée classification par des exemples représentatifs
- on stocke les exemples tels quels dans une table ;
- pour prédire la classe d'une donnée, on détermine les exemples qui en sont le plus proche ;
- de ces exemples, on déduit la classe du nouvel exemple

Deux phases:

1. Apprentissage = stockage des exemples
2. Classement = recherche des voisins + affectation d'une classe

KPPV - Classement

Recherche des voisins : distance d entre deux exemples

$$d : D \times D \longrightarrow R;$$
$$(x, y) \longmapsto d(x, y)$$

Propriétés:

- $d(x, y) = 0 \Longleftrightarrow x = y$
- $d(x, y) = d(y, x)$ (symétrie)
- $d(x, z) \leq d(x, y) + d(y, z)$

Quelques problèmes:

- dominance d'un attribut: la normaliser.
- attributs non numériques: $d(a, b) = 1$ si $a \neq b$ et 0 sinon
- trouver k : tester plusieurs valeurs.

KPPV - Classement

Classement:

1. Calculer la distance entre les exemples
2. sélectionner les k plus proches exemples
3. Choix de la classe:
   - Classe majoritaire
   - Utilisation des poids pour chaque classe p_i (p_i= proportion de i) et choisir la chasse k telle que le produit du nombre d'éléments (voisins) et de son poids soit le plus élevé.

Problème

$x_{i \in 1,...,N}$ le i^e individu appartenant à la classe y_i et $x_{i,j}$ la j^e composante. Ces attributs sont numériques, quantitatifs

probème

trouver un hyperplan qui sépare les exemples positifs des exemples négatifs.

Formellement, il faut trouver :

$$Argmax_{w,w_0} \ min \left\{ \|x - x_i\| : x \in R^d, (wx + w_0) = 0, i = 1, ..., m \right\}$$

Problème linéairement séparable

Chaque exemple peut être représenté par un point de l'espace. Les individus positifs sont séparables des individus négatifs par un hyperplan H. Notons H_+ l'hyperplan parallèle à H qui contient l'individu positif le plus proche de H, resp. H_- pour l'individu négatif.

A faire

Une MVS linéaire recherche l'hyperplan qui sépare les données de manière à ce que la distance (marge) entre H_+ et H_- soit la plus grande possible

Un hyperplan a pour équation $y = <\vec{w}\vec{x} + b$. Il faut alors trouver les w tels que:

$$y = <\vec{w}\vec{x} + b \geq 0 \text{ si } y = +1 \text{ et } y = <\vec{w}\vec{x} + b \leq 0 \text{ si } y = -1$$

hyperplan séparateur

hyperplan séparateur ou séparatrice linéaire = hyperplan qui sépare parfaitement les deux classes + et -, ie qui vérifie l'équation précédente, il sépare parfaitement leurs points d'apprentissage.

Il faut donc calculer la marge maximale:

- le vecteur $\vec{w}$ est perpendiculaire à l'hyperplan H et B un point de H_+ (sortie=1) et A le point le plus proche de B sur H_- (sortie =-1); $\vec{OB}=\vec{OA}+\vec{AB}$
- par définition des points A et B, $\vec{AB}$ est parallèle à $\vec{w}$, donc $\vec{AB}=\lambda\,\vec{w}$, $\vec{OB}=\vec{OA}+\lambda\,\vec{w}$
- On aimerait avoir $B \in H_+ \Rightarrow <\vec{w}\vec{OB}> +b = 1$ et $A \in H_- \Rightarrow <\vec{w}\vec{OB}> +b = -1$
- en résolvant, on obtient :

$$\lambda = \frac{2}{\|w\|}$$

La lagrangien

Rechercher l'hyperplan revient alors à résoudre la formulation primale:

$$\begin{cases} \text{Minimiser} & \frac{1}{2}\|w\|^2 \\ \text{sous la contrainte} & u_i(wx_i + w_0) \geq 1, \ i = 1, ..., m \end{cases}$$

Il faut donc règler $d + 1$ paramètres, où d est la dimension de l'espace des entrées X. Il faut donc utiliser les techniques de programmation quadratique, et comme l'équation 9 la fonction objectif et les contraintes sont convexes, on peut obtenir la forme duale (lagrangien) qui a les mêmes solutions que le problème de départ:

$$L(w, w_0, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha_i(u_i.((wx_i) + w_0) - 1$$

Résolution

Point-selle

point où la dérivée par rapport aux variables primaires du lagrangien s'annulle

$$\frac{\partial L}{\partial w}(w, w_0, \alpha) = 0;\ \frac{\partial L}{\partial w_0}(w, w_0, \alpha) = 0$$

d'où

$$\sum_{i=1}^{m} \alpha_i u_i \text{ et } w^* = \sum_{i=1}^{m} \alpha_i u_i x_i$$

$$\begin{cases} Max_\alpha \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j u_i u_j < x_i, x_j > \right] \\ \text{avec } \alpha_i \geq 0, i = 1, ..., m \text{ et } \sum_{i=1}^{m} \alpha_i u_i = 0 \end{cases}$$

L'hyperplan solution s'écrit:

$$h^*(x) = (w^* x) + w_0^* = \sum_{i=1}^{m} \alpha_i^* u_i . < x_i, x > + w_0^*$$

en prennant un exemple (x_c, u_c), on a:
$w_0 = u_c - w x_c = u_c - \sum_{i=1}^{m_c} u_i \alpha_i x_i x_c$

Problème non linéairement séparable

Utiliser les fonctions de noyau pour exploiter une corrélation entre les entrées et d'identifier les exemples critiques par rapport auxquels la frontière pourra être trouvée
Soit Φ une transformation non linéaire de l'espace X en un espace de redescription $\Phi(X)$, on a:

$$x = (x_1, ..., x_d) \longrightarrow \Phi(x) = (\phi_1(x), ..., \phi_d(x), ...)$$

Transformation

Le problème d'optimisation devient alors:

$$\begin{cases} Max_{\alpha} \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j u_i u_j < \Phi(x_i), \Phi(x_j) > \right] \\ \alpha_i \geq 0, i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i u_i = 0 \end{cases}$$

et l'équation du plan devient:

$$h(x) = \sum_{i=1}^{m} \alpha_i^* u_i < \Phi(x).\Phi(x_i) > + w_0^m$$

Fonction à noyau

On peut donc remplacer le problème de déterminer la transformation non linéaire Φ par celui du choix d'une fonction noyau $K(.,.)$ et le problème d'optimisation devient:

$$\begin{cases} Max_\alpha \left[\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j u_i u_j < K(x_i), K(x_j) > \right] \\ \alpha_i \geq 0, i = 1, ..., m \\ \sum_{i=1}^{m} \alpha_i u_i = 0 \end{cases}$$

et l'hyperplan:

$$h(x) = \sum_{i=1}^{m} \alpha_i^* u_i < \Phi(x).K(x_i) > + w_0^m$$

Quelques fonctions

- polynomiale:$K(x, y) = (1 + x.y)^s$
- sigmoïde: $K(x, y) = tanh(kx.y - \delta)$
- radial basis function: $K(x, y) = exp(-(x - y)^2/2\sigma^2$

Problèmes multi classes

Les SVM sont conçus paour les problèmes de classification binaire, donc pour un problème à N classes, il faut construire N SVM.
Il faut alors choisir le résultat du SVM qui a la plus grande marge.