

Cours IFT6266,

Introduction à la reconnaissance des formes

- Le problème de la **reconnaissance des formes** est d'apprendre des fonctions de $\mathcal{R}^n$ vers $\{1, 2, \dots, N_c\}$, où N_c est le nombre de classes. Il s'agit donc d'apprendre une partition étiquetée de $\mathcal{R}^n$, qui associe à chacune des régions dans la partition un symbole (représenté par un entier borné). Souvent on étudie plutôt le cas de 2 classes ($N_c = 2$) car en principe, un ensemble de classifieurs binaires peut représenter n'importe quel classifieur multi-classe. Voir le livre [Duda and Hart, 1973] pour une introduction classique à la reconnaissance des formes (du point de vue statistique). Voir [Le Cun and Bengio, 1995] pour un survol un peu plus récent du point de vue de l'application à la reconnaissance des formes des méthodes reliées aux réseaux de neurones. Pour une autre introduction récente, voir le livre de Bishop [Bishop, 1995].
- Exemples:
 - l'entrée est une image discrétisée, c'est à dire un ensemble de valeurs associées à chaque pixel ("**p**icture **e**lement") de l'image. Ces valeurs peuvent être
 - * binaires: c'est une image en noir et blanc,
 - * réelles (mais souvent discrétisées en 256 niveaux): c'est une image en tons de gris,
 - * des vecteurs réels: par exemple, pour une image couleur RGB, trois valeurs indiquent pour chaque pixel l'intensité des couleurs rouge, vert et bleu.

La sortie représente une classification de l'image: par exemple,

- * Y-a-t-il un visage ou non?
 - * De qui est-ce le visage? (parmi un nombre fini de possibilités).
 - * Est-ce le visage d'une personne particulière? (oui ou non).
 - * Est-ce un caractère d'un alphabet latin, cyrillique, hébreu, arabe, chinois?
 - * Quel est l'identité de ce caractère? (parmi un nombre fini de possibilités).
 - * Est-ce l'image d'une plaque d'immatriculation?
 - * Est-ce que l'objet, e.g., résultat d'un processus industriel quelconque, présent dans l'image, satisfait certaines normes ou est-ce qu'il devrait être rejeté?
 - * Est-ce l'image d'une main? Dans quel type de position? (parmi un nombre fini)?
 - * Est-ce que ces DEUX images représentent le même objet?
 - * etc...
- L'entrée est une "image" dont un axe représente le temps et l'autre représente la structure spectrale locale (dans chacun des intervalles temporels). La sortie désirée pourrait être:

- * l'identité du phonème prononcé (parmi un nombre fini),
 - * l'identité du locuteur qui a prononcé ces mots,
 - * le type d'objet sondé par un radar,
 - * le type de matériau analysé par une sonde (dont le signal a été analysé avec une décomposition temps/fréquence),
 - * etc...
- L'entrée pourrait être un ensemble de variables économiques transformées représentant de l'information disponible aujourd'hui, et la sortie pourrait représenter une décision financière (où mettre son argent, parmi un nombre fini de possibilités)?
 - L'entrée pourrait être un ensemble de symptômes ou de résultats de tests médicaux (certains pourraient avoir une valeur particulière "non-mesurée"), et la classification pourrait indiquer un diagnostic concernant un patient.
 - etc...
- Généralement, le but de l'apprentissage est de **minimiser le nombre d'erreurs de classification**, i.e., la fonction de coût à minimiser est

$$Q(f, z) = 0 \text{ si } f(x) = y \text{ et } 1 \text{ sinon}$$

où z est la paire entrée/sortie (x, y) , et f est une solution possible au problème.

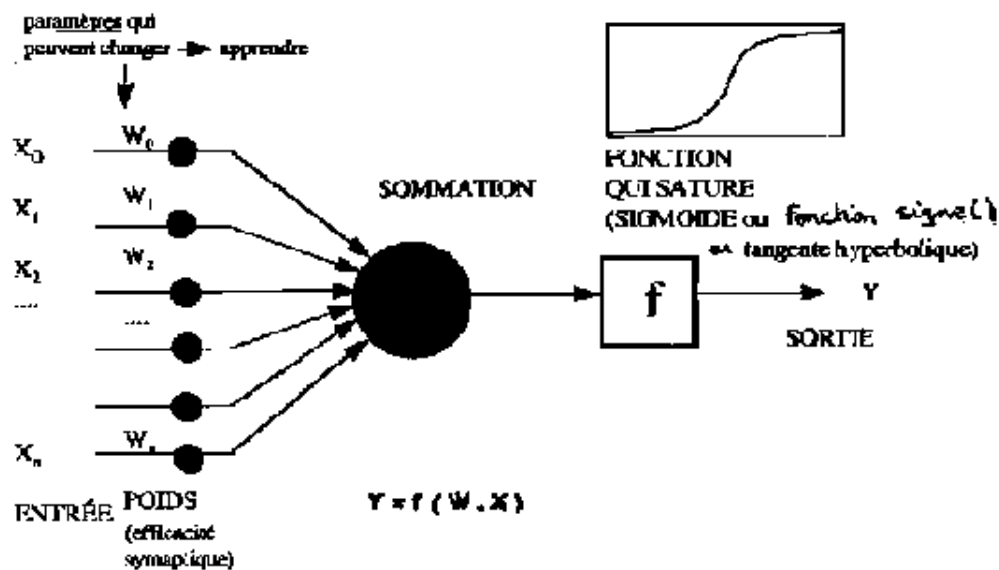
- Une généralisation de cette fonction coût permet de tenir compte du fait que certaines erreurs sont plus coûteuses que d'autres:

$$Q(f, z) = 0 \text{ si } f(x) = y \text{ et } C_{f(x),y} \text{ sinon}$$

où $C_{i,j}$ est une **matrice de coûts** (la pénalité quand la bonne réponse était j et qu'on avait répondu i).

- Il est à noter qu'un certain nombre de problèmes (malgré les apparences) ne rentrent pas tout à fait dans le cadre classique de la reconnaissance des formes, et en particulier nous étudierons ceux pour lesquelles il y a des **dépendances séquentielles** (les exemples ne sont pas tirés i.i.d., i.e., indépendamment d'une même distribution).

Neurone Formel



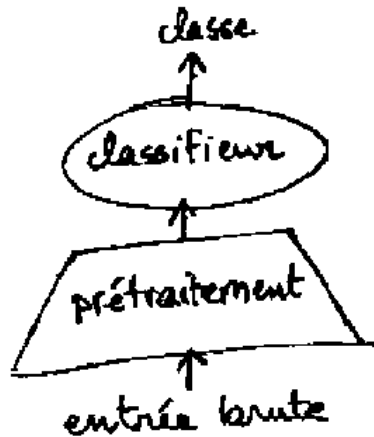
- Les classifieurs linéaires et le neurone formel. Soit $x \in \mathcal{R}^n$ l'entrée, les classifieurs linéaires ont la forme:

$$f(x) = \text{signe}(w \cdot x + b)$$

où la fonction $\text{signe}(s)$ donne 1 quand $s > 0$ et donne -1 quand $s \leq 0$, $w \in \mathcal{R}^n$ et $b \in \mathcal{R}$, et $w \cdot x = \sum_{i=1}^n w_i x_i$. Le neurone formel est un classifieur linéaire. Il correspond à une simplification (énorme) du fonctionnement du neurone.

- La capacité des classifieurs linéaires dans $\mathcal{R}^n$ est $n + 1$. Sauf pour des cas maladroits (exemples collinéaires ou superposés) on peut donc toujours classifier correctement $n + 1$ exemples en $\mathcal{R}^n$ avec un classifieur linéaire (ce qui ne veut pas dire que ce classifieur fonctionnera bien en généralisation).
- Notez que la famille des classifieurs linéaires est très importante parce qu'elle constitue la base sur laquelle sont construits beaucoup d'algorithmes d'apprentissage. Quand il existe w et b qui classifient un ensemble d'exemples D , on peut trouver un tel couple (w, b) de manière garantie et assez rapide (algorithme du *Perceptron* et *Support Vector Machines*).

- Division d'un système de reconnaissance des formes en deux parties:



1. **Prétraitement et extraction de traits** (*preprocessing* ou *feature extraction*). Beaucoup de connaissances à priori peuvent être utilisées dans le design de ce module.
2. **Classifieur entraînable** (algorithme d'apprentissage), avec des paramètres libres, idéalement choisis pour minimiser un critère d'erreur. L'entrée du classifieur est la sortie du module de prétraitement.

Cette décomposition est particulièrement importante pour les classifieurs linéaires, puisque leur capacité est très limitée. En extrayant un grand nombre de traits (selon un design décidé d'avance, "à la main"), on augmente d'autant la capacité du modèle.

- **Les classifieurs polynomiaux.** Un "prétraitement" très simple consiste à projeter le vecteur d'entrée $x \in \mathcal{R}^n$ vers un espace de plus haute dimension, en multipliant entre eux tous les éléments de x , par groupes de 1, 2, 3, etc.. jusqu'à d . Ensuite on peut appliquer un classifieur linéaire sur ces produits. Par exemple, si l'ordre du polynôme est $d = 2$ et $n = 3$, alors le vecteur d'entrée du classifieur linéaire est $(x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3)$. La taille du vecteur projeté croît de l'ordre de n^d (et donc le nombre de paramètres et la capacité de cette classe de fonctions croissent exponentiellement en d).
- **La malédiction de la dimensionalité.** Que se passe-t-il si on extrait un très grand nombre de traits avec le prétraitement? Il devient plus facile de séparer les exemples d'apprentissage (de les apprendre), mais cela ne se traduit pas toujours par une amélioration de la généralisation. Une raison est simplement la capacité accrue, mais une autre est la fameuse malédiction de la dimensionalité. Pour comprendre pourquoi beaucoup d'algorithmes d'apprentissage donnent de piètres résultats en haute dimension, considérons l'algorithme d'apprentissage suivant (une méthode d'histogramme). On divise l'espace d'entrée, $\mathcal{R}^n$, en un grand nombre de petits cubes, obtenus en divisant chacun des axes en M intervalles

disjoints. Ou bien on a des entrées qui sont n variables discrètes, prenant chacune M valeurs possibles. Dans les deux cas, cela donne M^n petits cubes qui correspondent à des combinaisons possibles des valeurs des variables d'entrée. Dans chacun des cubes, on compte le nombre de fois que chacune des classes est observée parmi les exemples d'apprentissage. Quand un nouvel exemple arrive, on regarde dans le cube qui contient le point d'entrée et on répond la classe majoritaire dans ce cube. Le problème, en haute dimension, c'est que la majorité des cubes sont vides (ou ne contiennent pas assez d'exemples pour permettre d'estimer correctement la classe correcte), parce que $l \ll M^n$ pour presque n'importe quel M quand n est assez grand et l est fini et fixe. Nous verrons beaucoup d'autres algorithmes plus sophistiqués, mais basés essentiellement sur ce même principe (on regarde les exemples dans un voisinage local). Nous avons vu aussi avec les classifieurs polynomiaux que pour pouvoir capturer des dépendances de haut niveau (i.e. entre d variables), ces classifieurs ont besoin d'un nombre de paramètres qui croît très vite (en n^d) quand la dimensionnalité n de l'entrée est grande. Pour essayer de comprendre ce qui ne fonctionne plus en haute dimension, on peut aussi étudier la géométrie de ces espaces: les notions usuelles de distance perdent leur sens; presque tous les points sont à la surface d'un nuage de points; presque tous les points d'un nuage de points sont équidistants les uns des autres (voir les exemples géométriques dans [Bishop, 1995]). Ainsi, quand l'entrée est en haute dimension, il faut privilégier l'utilisation d'un prétraitement qui **réduit la dimensionnalité** plutôt que de l'augmenter, et donc qui est approprié pour le problème considéré. Cela veut dire que soit

- ce prétraitement est en fait appris (c'est ce qu'on fait avec les réseaux de neurones multi-couche), et/ou
 - ce prétraitement utilise des connaissances a priori pertinentes pour le problème traité, i.e., il extrait des caractéristiques de l'entrée qui aident spécifiquement à séparer les classes désirées.
- **Classification ambiguë.** Dans un problème de reconnaissance des formes, il peut être parfois désirable que notre solution au problème ne soit pas simplement une partition de l'espace d'entrée, mais indique aussi l'ambiguïté de la classification. Quelles sont les sources de cette ambiguïté?
 - Informations manquantes (ou le “bruit”) = pas assez d'information dans X pour arriver à une décision sûre sur Y , même si la distribution $P(X, Y)$ des données étaient connue, car Y n'est pas une fonction de X .
 - Nombre limité d'exemples: on ne peut se permettre une surface de décision trop complexe, sinon on a de l'*overfitting*. Dans certaines régions de l'espace d'entrée, nous n'avons pas assez d'exemples pour pouvoir décider de manière sûre quelle est la classe appropriée.

Une manière très commune de représenter cette incertitude est basée sur l'utilisation de modèles probabilistes, par exemple de la probabilité de la classe étant donnée l'entrée.

- **Décision de Bayes**

Soit C la classe associée à une entrée X . Si on connaissait la distribution a posteriori des classes, $P(C|X)$, on pourrait l'utiliser pour prendre la décision optimale pour l'entrée x :

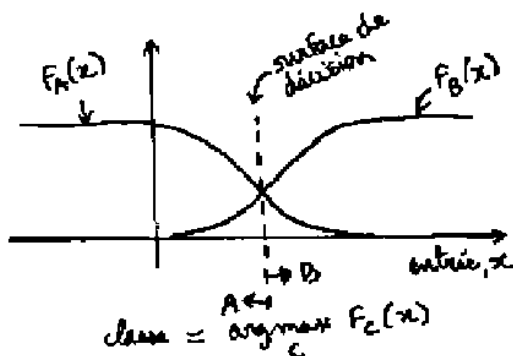
$$c^*(x) = \operatorname{argmax}_c P(C = c|X = x)$$

On peut montrer que cette décision minimise le nombre d'erreurs de classification. Beaucoup d'algorithmes d'apprentissage sont donc basés sur l'estimation de la **probabilité a posteriori des classes** à partir d'exemples.



- **Surfaces de décision et fonctions discriminantes.** Une fonction de décision $c(x)$ de $\mathcal{R}^n$ vers les classes définit les **surfaces de décision** qui sépare les différentes régions de l'espace d'entrée associées à chaque classe. Une représentation communément utilisée de ces surfaces de décisions est donnée par un ensemble de **fonctions discriminantes**, $g_i(x)$, pour chaque classe i , telles que la fonction de décision $c(x)$ est donnée par

$$c(x) = \operatorname{argmax}_i g_i(x)$$



La surface de décision entre la classe i et la classe j est donc définie par l'équation $g_i(x) = g_j(x)$. Les probabilités a posteriori des classes (ou un estimé) forment des fonctions discriminantes. Les fonctions discriminantes sont généralement plus faciles à apprendre que la fonction de décision car elles peuvent être lisses. Elles ont aussi l'avantage de pouvoir représenter de l'information sur l'ambiguïté de la décision (donc concernant la probabilité d'erreur).

- **Classification par l'estimation de densité conditionnelle des classes.**

Dans certains algorithmes d'apprentissage, on estime $P(C|X)$ indirectement, en estimant $P(X|C)$ et $P(C)$ (puis on utilise $P(C = c|X = x) = \frac{P(X=x|C=c)P(C=c)}{\sum_c P(X=x|C=c)P(C=c)}$). D'une part on peut facilement apprendre les **probabilités a priori** des classes, $P(C = c)$, à partir des fréquences des classes. D'autre part, il faut estimer la densité $P(X = x|C = c)$ (que l'on peut faire séparément pour chaque classe). Un avantage est que si la distribution des classes dans nos exemples d'apprentissage ne reflète pas la réalité (donc on connaît par ailleurs la distribution a priori des classes), on peut utiliser ces probabilités de classes a priori. Le système est aussi plus modulaire (apprentissage séparé d'un modèle par classe), mais malheureusement, l'estimation de densité est un problème fondamentalement plus difficile que la classification (quand la dimensionalité de la variable d'entrée est plus grande que celle de la variable de sortie). De plus, cette approche apprend plus de choses que strictement nécessaire (en effet on apprend implicitement la densité $P(X)$ et donc la distribution jointe $P(X, C)$, alors que seule $P(C|X)$ est nécessaire).

- **Rejet des cas trop ambigus.** Dans certaines applications, une décision particulière est possible: "je ne sais pas". Par exemple, dans ces cas là, un humain est appelé à trancher. Par exemple, pour la classification des codes postaux sur des enveloppes, on voudrait faire intervenir un humain seulement quand le coût du traitement par un humain est plus petit que le coût d'une erreur de classification fois la probabilité de cette erreur. Quand les fonctions discriminantes ne représentent pas fidèlement les probabilités a posteriori, d'autres critères de rejet sont parfois utilisés: un seuil sur la différence entre les valeurs des deux fonctions discriminantes ayant la plus grande valeur, ou un degré de consensus entre plusieurs classifieurs.

References

- [Bishop, 1995] Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press, London, UK.
- [Duda and Hart, 1973] Duda, R. and Hart, P. (1973). *Pattern Classification and Scene Analysis*. Wiley, New York.
- [Le Cun and Bengio, 1995] Le Cun, Y. and Bengio, Y. (1995). Pattern recognition and neural networks. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, pages 711–714. MIT Press.