CHAPITRE 3 Apprentissage

Remonstrez à vostre roy les erreurs que congnoistrez, et jamais ne le conseillez ayant esgard à vostre profit particulier, car avesque le commun est aussi le propre perdu. (Remontrez à votre roi les erreurs que vous reconnaissez, et ne le conseillez jamais en fonction de votre intérêt personnel, car le bien de chacun se perd dans le désastre commun)

FRANÇOIS RABELAIS*

UNE bonne partie de l'excitation que l'on trouve autour des réseaux de neurones provient de l'espoir qu'on y a mis d'éviter enfin des procédures longues et ennuyeuses, difficiles et coûteuses qui sont nécessaires dès lors que l'on veut assembler des heuristiques et des règles afin de construire des machines, des algorithmes, réalisant des tâches perceptuelles ou cognitives non triviales comme celles que réalisent le système visuel ou les systèmes experts.

Explicitons : le mot « algorithme » sert à désigner toute méthode de calcul permettant de résoudre un problème et qui se présente sous la forme d'une suite d'opérations élémentaires obéissant à un enchaînement déterminé. La façon de poser la multiplication de deux nombres à plusiers chiffres, ou la division, est un algorithme. Le mot sonne comme s'il dérivait du grec, à l'instar d'« arithmétique », mais ce n'est en fait que le résultat d'un déguisement. Le « th » a été introduit par analogie avec celui d'« arithmétique », quoiqu'il n'ait en réalité rien à faire ici)[1].

Nous préfèrons naturellement "apprendre" des exemples à nos machines en espérant que le bon algorithme "trouvera" les diverses règles et heuristiques nécessaires à la résolution du problème. Nous verrons dans la suite de ce chapitre que cet algorithme devra être optimal quel que soit la tâche à résoudre. Un tel système sera appelé « non biaisé » est ne sera pas *a priori* dédié à un problème ou à une classe de problèmes donnés.

L'apprentissage et la généralisation sont les deux sources auxquelles ceux qui manipulent les RN et autres systèmes adaptatifs doivent s'abreuver sous peine d'un voyage difficile dans la galaxie

** Gargantua, Chapitre XLIIII (1534).*

[1] Une orthographe reflétant mieux l'éthymologie serait « algorisme ». En effet, l'origine du mot est issue du nom du mathématicien persan (IV siècle après J.-C.) qui introduisit la notion de zéro dans la culture occidentale : Muhammad ibn Musa abu Abdallah al-Khoreami al-Madjusi al- Qutrubulli, dont le nom usuel al Khorezmi fut latinisé au moyen-Âge en « Algorismus ». On pouvait craindre pire. Son surnom indique que sa famille venait de la province de Khorezm, au sud de la mer d'Aral, maintenant partie de la nouvelle république indépendante d'Ouzbékistan. Il est l'auteur d'un traité mathématique dont le titre contient l'expression arabe « *al jabr* » signifiant « la permutation », qui nous a donné le mot « algèbre ». À l'origine, le mot « algorisme » désignait le système décimal de notation, dont on pense qu'il a en grande partie passé de l'Inde en Europe par le biais de la traduction en latin de « l'algèbre » d'al-Khorezmzi.

connexionniste.

Mais qu'est-ce que l'**apprentissage** ? Je n'ai pas trouvé de réponse autre que circulaire que je vous livre :

- "L'apprentissage, c'est à dire la capacité d'acquérir de nouveaux comportements ou d'en modifier d'anciens par expérience, est une propriété de presque tous les animaux, y compris ceux que l'on peut difficilement qualifier d'intelligents" Eric Kandel[2] : Cellular basis of behaviour.
- "On appelle apprentissage toute transformation d'un système cognitif ayant pour effet de conférer à ce système une nouvelle disposition". Daniel Andler Paris vII intellectica.

Mais qu'est-ce que la **généralisation** ?

- C'est la capacité à étendre une compétence à des exemples non appris.
- C'est la transformation d'une présentation partielle en une représentation complète ...
  ... c'est un processus d'inférence déductive. (D. Andler)

Cette dernière notion est très importante, les techniques de résolution de problèmes par apprentissage par l'exemple n'ont d'intérêt que si elles sont capables de généraliser.

Ces définitions, lorsqu'elles ne sont pas auto-référencées sont plutôt floues. Heureusement, on peut en trouver de plus précises pour des machines ou tout du moins pour des systèmes artificiels.

Apprentissage par l'exemple : soit un système à apprentissage quelconque et soit x une variable aléatoire dont la distribution de probabilité P_x est fixe mais inconnue et supposons qu'il existe une fonction $\phi(x) : X \rightarrow Y$. Par exemple en reconnaissance de caractères $x \in X$ peut être une matrice de pixels et $y \in Y$ le caractère alphanumérique associé. Le but de l'apprentissage est de construire une fonction paramétrique $\hat{\phi}(x, \theta) : X \rightarrow Y$ qui minimise :

$$\int_{x \in X} J(\hat{\phi}(x, \theta), \phi(x))\, dP_x$$

en modifiant comme il faut les θ. J est une fonction de coût scalaire (par exemple le carré de la différence entre ϕ et $\hat{\phi}$). Pour y parvenir avec un système d'apprentissage par l'exemple (ou apprentissage supervisé) on dispose généralement de N couples $\{x_i, \phi(x_i)\}$ tirés de (X, Y).

La performance d'un tel système se mesure par la somme des erreurs aux N points choisis :

$$\sum_{i=1}^{N} J(\hat{\phi}(x_i, \theta), \phi(x_i))$$

Le type d'apprentissage est déterminé par la façon dont les paramètres sont modifiés au cours du temps. On écrira $\theta^{t+1} = \theta^t + \Delta\theta^t$

On parlera alors d'algorithme d'apprentissage.

Il y a une grande quantité d'algorithmes qui diffèrent les uns des autres par la façon dont on calcule Δw. Ils dépendent du modèle de l'environnement dans lequel le RN est plongé (paradigme d'apprentissage).

Processus d'apprentissage							
algorithmes(règles)					paradigmes		
Boltzman	Hebb	Thorndike	compétitif	correction d'erreur	supervisé	renforcement	non supervisé

Généralisation : comment se comporte $\hat{\phi}$ sur des exemples autres que les N connues ? L'essentiel de ce chapitre est dévolu à essayer de répondre à cette question.

[2] de l'université Columbia, Prix Nobel de physiologie et de médecine en 2000.

3.1 Processus d'apprentissage

3.1.1 Apprentissage par correction d'erreur

Ou par essais et erreurs ou encore la carotte et le bâton.

Soit $d_i(n)$ la réponse désirée du neurone i au temps n, $y_i(n)$ la réponse effective lorsque le réseau reçoit $x(n)$ sur ses entrées (le stimulus). Le couple $\{x(n), y(n)\}$ est un des exemples à apprendre. On définira un signal d'erreur par :

$$e_i(n) = d_i(n) - y_i(n) \tag{3.1}$$

Le but final de l'apprentissage par correction d'erreur est de minimiser une fonction de coût J construite à partir de $e(n)$. Une fois qu'on a décidé de la fonction de coût, l'apprentissage devient un problème d'optimisation pour lequel il existe quantité de techniques.

Une fonction de coût fréquemment rencontrée est le *critère des moindres carrés* :

$$J = E[\frac{1}{2}\sum_i e_i^2] \tag{3.2}$$

où E est l'espérance statistique (la moyenne). Le terme $1/2$ est là pour simplifier les calculs futurs.

On suppose que le processus qui génère les entrées est stationnaire. Une méthode connue de minimisation est la descente de gradient, mais elle nécessite de connaître la distribution stationnaire des données qui, généralement, est inconnue. On cherchera alors une solution approchée en minimisant le coût instantané suivant :

$$\mathcal{E}(n) = \frac{1}{2}\sum_i e_i^2 \tag{3.3}$$

On optimise le réseau en minimisant $\mathcal{E}(n)$ par rapport aux poids. On trouve (Window Hoff 1960 – règle Delta).

$$\Delta w_{ij}(n) = \eta e_i(n) x_j(n) \tag{3.4}$$

η est une constante ≥ 0 appellée pas d'apprentissage. On trouve donc que la variation du poids est proportionnelle au produit du signal d'erreur par le signal qu'il y a sur la synapse en question.

Il faut faire attention à la valeur numérique de η qui non seulement fixe la vitesse du calcul mais aussi (et surtout) la stabilité du processus d'apprentissage par correction d'erreur, et donc son résultat.

Il faut théoriquement prendre un pas tel que :

$$0 < \eta < 1\frac{2}{\lambda_{max}[H(w)]} \tag{3.5}$$

où λ_{max} est la plus grande valeur propre de $H(w)$ et où $H(w)$ et le Hessien de la fonction de coût J évalué au point w. C'est une matrice symétrique dont les éléments sont les dérivées secondes partielles de la fonction de coût par rapport aux poids :

$$H(w) = \frac{\partial^2 J}{\partial w_{ij} \partial w_{kl}} \tag{3.6}$$

J(w) représente la surface d'erreur. Si les unités sont linéaires on a affaire à une boule possèdant 1 seul minimum ; sinon il existe un minimum global (éventuellement multiple) et des minimas locaux. Dans ce dernier cas, comme la descente de gradient procède pas à pas on peut rester piégé dans un minimum local.

3.1.2 Apprentissage de Hebb

Cette forme d'apprentissage est en fait un postulat tiré de la biologie et est la plus connue et la plus vieille des règles d'apprentissage (1949). Donald Hebb était un psychologue canadien.

Formulé dans un **contexte neurobiologique** il dit précisement :
"When an axon of cell A is near enough to excite a cell B repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased".

Dans un **contexte RN** on transforme cette proposition en

1. si les 2 neurones situés de part et d'autre de la synapse sont actifs simultanément (synchrones) alors on augmente le poids,
2. si les 2 neurones ne sont pas synchrones, la synapse décroît ou est éliminée.

Une telle synapse est dite Hebbienne, elle dépend du temps, les interactions sont fortement locales. Les variations du poids se font grâce à un mécanisme de conjonction si on regarde les signaux, de corrélation si on parle en termes statistiques. (On dira aussi qu'une synapse est anti-hebbienne lorsque sa force (le poids) baisse pour des activités post – et pré – synaptiques corrélées et augmente sa force lorsque ces activités sont non corrélées.

Dans un **contexte mathématique** on pourrait poser (voir figure 3.1) :

$$\Delta w_{ij}(n) = \eta y_i(n) x_j(n) \tag{3.7}$$

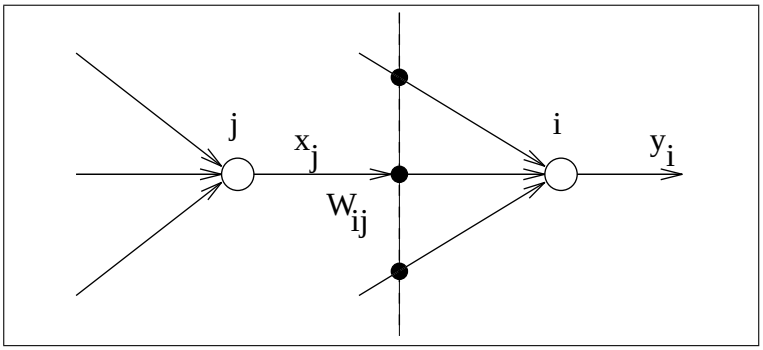

FIG. 3.1 – Interconnexion de deux neurones i et j par la synapse w_{ij}.

mais il y aurait variation exponentielle de w_{ij}. On ajoute alors un terme d'oubli :

$$\begin{aligned} \Delta w_{ij}(n) &= \eta y_i(n) x_j(n) - \alpha y_i(n) w_{ij}(n) && (3.8)\\ &= \alpha y_i(n)(c x_j(n) - w_{ij}(n)) && (3.9) \end{aligned}$$

ce qui permet d'éviter les problèmes de convergence.

Dans un **contexte statistique** on pourrait poser :

$$\begin{aligned} \Delta w_{ij} &= \eta cov(y_i(n), x_j(n)) && (3.10)\\ &= \eta E[(y_i(n) - \overline{y}_i)(x_j(n) - \overline{x}_j)] && (3.11)\\ &= \eta\{E[y_i(n)x_j(n)] - \overline{y}_i \overline{x}_j\} && (3.12) \end{aligned}$$

Le premier terme ressemble à la première proposition, le second terme $\overline{y}_i\overline{x}_j$ est un seuil égal au produit des activités moyennes post et pré synaptiques.

Retour à un **contexte biologique** : il se pourrait qu'il y ait une plausibilité physiologique (modèle de l'hyppocampe).

3.1.3 Apprentissage compétitif

Les neurones de sortie du réseau « décident » lequel d'entre eux sera actif. Il n'existe donc qu'un seul neurone de sortie actif à chaque pas de temps. Les premières idées viennent de 1973 (Van der Malsburg : self organisation et orientation des cellules dans le cortex strié visuel), Fukushima 75 (neocognition attention ! !), Grossberg 72 76, classification adaptation).

Les neurones deviennent des *détecteurs de caractéristiques* car ils s'adaptent à une caractéristique particulière.

La compétition se fait au travers de connexions latérales inhibitrices et l'algorithme d'apprentissage s'écrit de la forme :

$$\Delta w_{ij} = \begin{cases} \eta(x_j - w_j) & \text{si le neurone} \quad i \quad \text{gagne la compétition} \\ 0 & \text{sinon} \end{cases}$$

On contraint les neurones à avoir la même norme euclidienne :

$$\sum_j w_{ij} = 1 \qquad \text{pour tout} \quad i$$

3.1.4 Apprentissage de Boltzmann

Il s'agit d'un algorithme stochastique issu de la thermodynamique et de la théorie de l'information (car on va chercher un maximum de vraisemblance).

Le réseau a une structure récurrente (mais les neurones ne font pas de connexion sur eux-mêmes) ; les neurones ont 2 états ± 1 (qu'on appelle souvent *on* et *off*). On caractérise la machine par une fonction énergie E définie par :

$$E = -\frac{1}{2}\sum_{i \neq j} w_{ij} s_i s_j$$

L'algorithme est le suivant :

- On choisit un neurone j au hasard, on le flippe (change d'état) de l'état s_j à l'état $-s_j$ avec la probabilité suivante dépendant de la température :

$$P[w(s_j \rightarrow -s_j)] = \frac{1}{1 + exp(-\frac{\Delta E}{kT})}$$

  où T est l'analogue d'une température.
- On recommence en tirant un autre j.

Si on opère ainsi, au bout d'un certain temps la machine atteint l'équilibre thermique.

Les neurones sont classés en 2 groupes fonctionnels : *visible* et *cachés*. Les visibles fournissent une interface avec l'environnement.

Il y a 2 modes de fonctionnement :

- clampé : les visibles sont fixés à des états fixés par l'environnement,
- libre : tous les neurones sont libres.

On pose ρ_{ji}^{+} la corrélation entre les états de neurones i et j sachant que le réseau est clampé, ρ_{ji}^{-} quand le réseau est libre.

En moyenne ces 2 corrélations sur tous les états possibles de la machine lorsqu'elle est à l'équilibre thermodynamique sont les suivantes :

$$\begin{aligned} \rho_{ji}^{+} &= \sum_{\alpha}\sum_{\beta} P_{\alpha\beta}^{+} s_{j|\alpha\beta} s_{i|\alpha\beta} \\ \rho_{ji}^{-} &= \sum_{\alpha}\sum_{\beta} P_{\alpha\beta}^{-} s_{j|\alpha\beta} s_{i|\alpha\beta} \end{aligned}$$

où $s_{i|\alpha\beta}$ est l'état du neurone i sachant que les visibles sont dans l'état α et les cachés dans l'état β.

$P_{\alpha\beta}^{+}$ est la probabilité conditionnelle d'avoir les neurones visibles dansl'état α quand les cachés sont dans l'état β et la machine dans l'état clampé.

La règle est la suivante (Hinton et Sejnovski 86) :

$$\Delta w_{ij} = \eta(\rho_{ij}^{+} - \rho_{ij}^{-}) \qquad i \neq j$$

3.1.5 Le "crédit asignment problem"

Autrement dit : le problème du bouc émissaire.

Le « credit asignment problem » (CAP) peut être structurel si, dans un système à apprentissage à plusieurs composantes interconnectées, on veut savoir quel neurone doit voir son comportement modifié par la règle d'apprentissage, et de combien, afin d'augmenter les performances globales du réseau à l'itération suivante.

Le CAP peut être de nature temporelle quand le système a réalisé plusieurs actions et qu'on ne sait pas laquelle est prépondérante dans les résultats obtenus.

3.1.6 Apprentissage supervisé

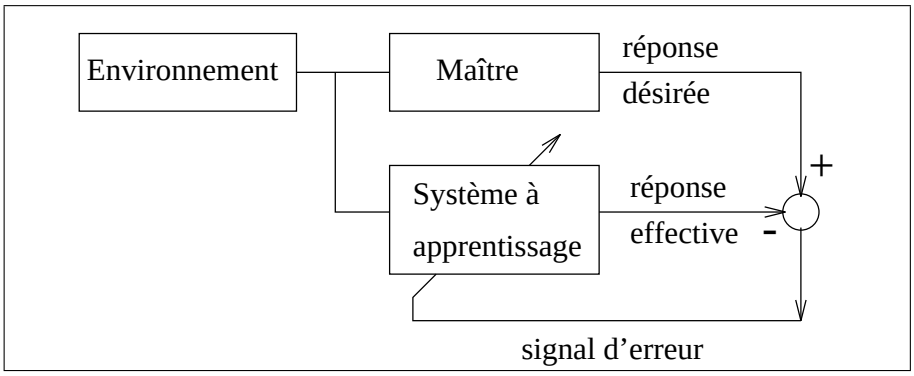

FIG. 3.2 – Schéma synoptique de l'apprentissage supervisé à correction d'erreur.

Le maître peut donner une réponse cible dit réponse *désirée* au système à appprentissage (voir figure 3.2). Le signal d'erreur (réponse désirée − réponse effective) sert à modifier la réponse désirée. Le système émule le maître. On le présume optimum dans un certain sens statistique. Un fois l'apprentissage terminé on peut faire fonctionner le système tout seul.

On utilise des algorithmes de correction d'erreur ; le plus simple est celui de Windrow et Hoff pour un simple neurone ou la *rétropopagation du gradient d'erreur* pour un réseau structuré en couches (voir le chapitre correspondant). L'apprentissage peut être statique ou en temps réel. Ce dernier cas est plus complexe.

Le **pouvoir d'expression** d'un RN artificiel est une mesure du nombre de fonctions différentes que celui-ci peut approximer (indépendamment des données ou de l'algorithme d'apprentissage). On montre qu'ils sont des approximateurs universels : ils peuvent approximer avec une précision arbitraire n'importe quelle transformation continue d'un espace à dimension finie vers un autre espace à dimension finie. A quelques cas pathologiques près, une seule couche cachée est suffisante.

Malheureusement il y a un problème d'**apprenabilité** ou de **complexité**. En effet Judd a montré que, étant donné un réseau de neurones artificiels arbitraire R et une tâche arbitraire T devant être résolue par R, le problème consistant à décider si dans l'espace de tous les paramètres de R (poids, structures) il existe une solution qui résoud adéquatement T, est équivalent au problème de satisfaisabilité, et est donc NP complet. Ainsi la recherche d'une telle solution (ie l'apprentissage) est-elle NP ardue.

Un désavantage de l'apprentissage supervisé est que, sans maître, un RN ne peut apprendre de nouvelles stratégies pour des situations particulières non couvertes par les exemples d'apprentissage. Cette limitation peut être levée par par un apprentissage adaptatif (dépendant du temps) ou encore par un apprentissage par renforcement.

3.1.7 Apprentissage par renforcement

À compléter plus tard.

3.1.8 Apprentissage non supervisé

Dans l'apprentissage non supervisé il n'y a pas de maître pour contrôler de l'extérieur le déroulement de l'apprentissage ; c'est pourquoi on l'appelle quelquefois auto-apprentissage. Le réseau va essayer de s'adapter aux régularités statistiques des données d'entrée. Il va donc automatiquement coder des classes dans sa représentation interne.

L'apprentissage non supervisé est réalisé, par exemple, par la règle de l'apprentissage compétitif vue plus haut. Il s'agit alors d'une stratégie appelée **winner-take-all**.

Si l'on compare l'apprentissage non supervisé à celui supervisé dont l'algorithme de rétropopagation du gradient est le représentant le plus largement connu et utilisé, on remarquera que les lois d'échelle sont défavorables aux algorithmes supervisés. Lorsqu'on augmente la taille d'un réseau possèdant plusieurs couches de neurones, le nombre de connexions croît de façon exponentielle alors qu'en apprentissage non supervisé cette variation est plutôt linéaire. *(Voir Jacobs et Jodan 91, Nowlan et Hinton 91, deSa et Ballard 92, Becker 91).*

3.1.9 Différentes tâches d'apprentissage

Le choix d'une procédure d'apprentissage particulière est conditionné très fortement par la tâche que le RN doit réaliser.

1. **approximation** : supposons que nous ayons une application non linéaire entrée $\rightarrow$ sortie de la forme :

$$d = g(\mathbf{x})$$

ou le vecteur $\mathbf{x}$ est l'entrée et le scalaire d la sortie. La fonction $g(\cdot)$ est inconnue. On veut construire un réseau qui approxime la fonction g non linéaire grâce à un ensemble de couples entrée-sortie $(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2)...(\mathbf{x}_N, d_N)$. À l'évidence on utilisera un apprentissage supervisé.

2. **association** : cette tâche peut prendre 2 formes : l'auto association et l'hétéro association. Dans l'auto association on veut stocker dans le réseau un certain nombre de motifs, de façon à ce que lorsqu'on lui présente un de ces motifs, bruité ou incomplet, le réseau retrouve (rappelle) le motif initial. On peut (doit) utiliser un apprentissage non supervisé. Dans l'hétéro association où le motif de sortie est différent de celui stocké on sera conduit à une procédure supervisée.
3. **classification** : Soit un certain nombre de catégories (classes), le problème consiste à ranger des stimuli (activations) donnés dans ces classes. La solution consiste à utiliser un apprentissage supervisé. L'avantage des RN est qu'ils sont capable de construire de façon non paramétrique des frontières de décision séparant ces classes, ils offrent donc la possiblité de résoudre des problèmes de classification extrêmement complexes.
4. **Prédiction**
   On possède des échantillons d'un signal :
   $$x(n-1), x(n-2), \cdots, x(n-M)$$
   et on veut prévoire $x(n)$. On le fera par correction d'erreur de manière supervisée en imposant $x(n)$ comme sortie désirée. En toute rigueur cependant, on pourrait dire qu'il s'agit en fait d'un apprentissage non supervisé puisque $x(n)$ appartient à l'ensemble des données d'entrée.
5. **Contrôle** : Werbos (1992) inventa le terme *neurocontrôle* pour désigner les processus de contrôle d'un système dynamique non linéaire qui utilisent les RN.
6. L'**Alignement de faisceaux** désigne les procédés qui, par exemple, permettent de sélectionner l'attention. Une image souvent utilisée est celle d'une « cocktail party » où, malgré une foule nombreuse et bruyante, nous sommes capables de focaliser notre attention sur ce que dit une personne éloignée. Une autre image est le système d'écho-location de la chauve-souris. Un problème de ce type se pose lorsque, avec une antenne acoustique composée de plusieurs récepteurs, on veut suivre un locuteur et focaliser sur lui le lobe principal de l'antenne *(voir Haykin 91, Widraw et Stears 85).*

Toutes ces tâches consistent fondamentalement à construire une application à partir d'un certain nombre d'exemples de cette application. Si on ne dispose pas de connaissances *a priori*, alors chacune de ces tâches est en fait un problème mal posé car il existe généralement plusieurs solutions pour réaliser l'application en question ; on verra plus tard qu'en utilisant la théorie de la régularisation on peut obtenir de bonnes solutions.

3.1.10 Adaptation et apprentissage

L'*espace* est une dimension importante du processus d'apprentissage, mais le *temps* l'est aussi. Lorsqu'un RN opère dans un environnement stationnaire, c'est-à-dire dont les statistiques ne changent pas avec le temps, il peut en théorie *apprendre* les caractéristiques essentielles de ces statistiques.

Fréquemment néanmoins l'environnement n'est pas stationnaire, les différents paramètres statistiques caractérisant les signaux évoluent avec le temps. Dans de telles situations, les méthodes

usuelles d'apprentissage ne sont d'aucune utilité car il devient nécessaire que le réseau suive ces variations et *adapte* continuellement et en temps réel ses paramètres internes.

Un système adaptatif doit très bien apprendre sinon il ne peut suivre les variations. Un réseau de neurones peut être vu comme un système non linéaire adaptatif qui apprend des structures temporelles. On sait depuis Kolmogoroff (1942) et Wiener (1949) qu'une solution mathématique formelle à ce problème est impossible à trouver. Gabor démontra néanmoins (1960) qu'on pouvait construire un filtre optimisant sa réponse par apprentissage. La sortie de ce filtre a la forme :

$$y(n) = \sum_{n=0}^{N} w_n x(n) + \sum_{n=0}^{N} \sum_{m=0}^{N} w_{nm} x(n) x(m) + \cdots$$

où les $x(1), \cdots, x(N)$ sont les exemples donnés en entrée. De nos jours on appelle ce polynome le polynome de Gabor-Kolmogoroff ou encore la série de Volterra. Les coefficients des termes linéaires w_n, des termes non linéaires diadiques w_{nm} sont appris par des méthodes de descente de gradient.

3.2 Apprentissage et statistique

3.2.1 L'apprentissage dans les RN est un processus de régression linéaire

Considérons un ensemble de N mesures ou observations $\mathbf{x}$, notées $x_1, x_2, \cdots, x_N$ où chaque x_i prend ses valeurs sur $\boldsymbol{R}^n$, ainsi qu'un deuxième ensemble d'observations d, sorties d'un système S. (On a ici posé d comme scalaire pour simplifier, mais cela n'ôte rien à la généralité des démonstrations futures).

Supposons que d dépende aussi d'une autre variable aléatoire $\mathbf{z} = \{z_1, z_2, \ldots\}$ inconnue [FS97]. Le système S réalise alors l'application (voir FIG. 3.3) :

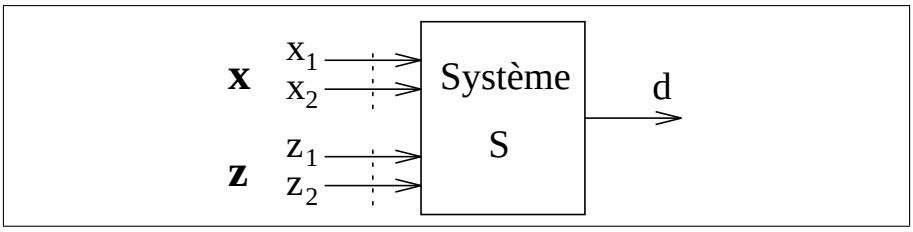

FIG. 3.3 – Les données sont issues d'une source connue $\mathbf{x}$ et d'une source inconnue notée $\mathbf{z}$.

$$d = f(x_1, x_2, \ldots, z_1, z_2, \ldots) \tag{3.13}$$

Mais comme $\mathbf{x}$ seulement est observable, nous modéliserons le système S par le modèle statistique :

$$d = g(\mathbf{x}) + \epsilon \tag{3.14}$$

où $g(\cdot)$ est une certaine fonction qui représente toute la connaissance que nous ayons sur la relation qui lie $\mathbf{x}$ à d, et ϵ un résidu, un « bruit » de moyenne nulle, dont la distribution de probabilité P_ϵ est inconnue mais indépendante de $\mathbf{x}$; il représente notre ignorance de la relation liant $\mathbf{x}$ à d.

Ce modèle statistique (3.14) est appelé *modèle régressif*, l'estimateur optimal de x est $E[d|\mathbf{x}]$, c'est-à-dire la fonction (déterministe) qui donne la valeur moyenne de d sachant $\mathbf{x}$. C'est l'espérance conditionnelle de la sortie sachant les entrées :

$$g(\mathbf{x}) = E[d|\mathbf{x}] \tag{3.15}$$

C'est la valeur de d réalisée *en moyenne* pour une réalisation particulière de l'évènement $\mathbf{x}$. Si d suit une loi de Bernoulli, alors $E[d|\mathbf{x}] = P(d|\mathbf{x})$.

S'il y a une seule valeur de d pour chaque x comme par exemple dans le problème de parité, alors dans ce cas $E[d|\mathbf{x}]$ n'est pas une moyenne mais une vraie valeur. En revanche dans le problème de classification de phonèmes en voisé/non voisé, il y a ambiguïté et $E[d|\mathbf{x}]$ est une vraie moyenne.

Posons F(**x**,**w**) la sortie d'un RN utilisé pour approximer d. La procédure consistera à ajuster les poids $\mathbf{w}$ de façon à minimiser l'erreur $J(\mathbf{w})$ définie par :

$$J(\mathbf{w}) = E[(d - F(\mathbf{x}, \mathbf{w}))^2] \tag{3.16}$$

Écrivons $J(\mathbf{w})$ sous une autre forme :

$$\begin{aligned} J(\mathbf{w}) &= E[(d - g(\mathbf{x}) + g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2] \\ &= E[(d - g(\mathbf{x}))^2 + (g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2 + 2(d - g(\mathbf{x}))(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))] \\ &= E[(d - g(\mathbf{x}))^2] + E[(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2] \\ &\quad +2E[(d - g(\mathbf{x}))(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))] \end{aligned} \tag{3.17}$$

Or on a :

$$\begin{aligned} E[(d - g(\mathbf{x}))(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))] &= E[\epsilon(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))] \\ &= E[\epsilon g(\mathbf{x})] - E[\epsilon F(\mathbf{x}, \mathbf{w})] \end{aligned} \tag{3.18}$$

dont le premier terme est nul car l'erreur n'est pas corrélée à $g(\mathbf{x})$; c'est le principe d'orthogonalité qui dit simplement que toute l'information disponible sur $\mathbf{x}$ est codée dans $g(\mathbf{x})$.

D'autre part, l'erreur espérée ϵ dans le modèle régressif de (3.14) est aussi décorrélée de $F(\mathbf{x}, \mathbf{w})$; le terme croisé de (3.18) est donc lui aussi nul et la fonction de coût (3.17) s'écrira alors :

$$J(\mathbf{w}) = E[(d - g(\mathbf{x}))^2] + E[(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2] \tag{3.19}$$

Le deuxième terme représente l'erreur du modèle. Il mesure l'écart de la fonction trouvée au modèle régressif (la partie prédictible contenue dans les données).

Notons que le premier terme de (3.19) ne dépend pas de $\mathbf{w}$. Donc, si un vecteur particulier $\mathbf{w_0}$ minimise $J(\mathbf{w})$, il minimise aussi le deuxième terme que l'on peut réécrire :

$$E[(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2] = \int_{\mathcal{R}^n} p(\mathbf{x})(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2 \, d\mathbf{x} \tag{3.20}$$

où $\mathbf{x} \in \mathcal{R}^n$ et $p(\mathbf{x})$ est la densité de probabilité de $\mathbf{x}$. En d'autres termes, le vecteur poids $\mathbf{w_0}$ est tel que $F(\mathbf{x}, \mathbf{w_0})$ est l'approximation de la probabilité conditionnelle $g(\mathbf{x}) = E[d|\mathbf{x}]$.

On notera que l'on a toujours :

$$J(\mathbf{w}) \geq E[(d - E[d|\mathbf{x}])^2]$$

ce qui signifie que le modèle régressif (3.15) est la meilleure approximation de la sortie désirée comme nous l'affirmions plus haut. Minimiser l'erreur quadratique entre $F(\mathbf{x}, \mathbf{w})$ et d (3.16) revient à minimiser l'espérance de l'erreur entre $F(\mathbf{x}, \mathbf{w})$ et $g(\mathbf{x})$ (3.20). C'est pourquoi on dit que l'apprentissage peut être vu comme un problème de régression non linéaire.

Remarquons qu'à la place de la fonction de coût quadratique, on aurait pu prendre l'entropie croisée :

$$F(\mathbf{x}, \mathbf{w}) \log(F(\mathbf{x}, \mathbf{w})) + (1 - F(\mathbf{x}, \mathbf{w})) \log(1 - F(\mathbf{x}, \mathbf{w}))$$

3.2.2 Dilemme biais/variance pour la généralisation

L'article fondateur du dilemme bias-variance est le fait de Geman, Bienenstock et Doursat [GBD92].

D'après ce que nous venons de voir, une mesure naturelle de la performance de l'estimateur $F(\mathbf{x}, \mathbf{w})$ est sa distance à la fonction régressive :

$$(g(\mathbf{x}) - F(\mathbf{x}, \mathbf{w}))^2 = (E[d|\mathbf{x}] - F(\mathbf{x}, \mathbf{w}))^2 \tag{3.21}$$

Que peut-on dire de cette distance lorsqu'on part d'une autre base d'apprentissage (mais qui a toujours la même taille) ce qui donnera donc des $F(\mathbf{x}, D)$ différents ?

Les informations contenues dans l'ensemble d'apprentissage :

$$D = \{(\mathbf{x}_1, d_1), (\mathbf{x}_2, d_2), \cdots, (\mathbf{x}_N, d_N)\}$$

sont implicitement présentes dans les poids $\mathbf{w}$; formellement on a donc le droit d'écrire la moyenne de l'erreur sur tous les échantillons possibles de taille N, sous la forme :

$$\begin{aligned} E_D[(E[d|\mathbf{x}] - F(\mathbf{x}, D))^2] &= E_D\left[(E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)] + E_D[F(\mathbf{x}, D)] - F(\mathbf{x}, D))^2\right] \\ &= E_D\left[(E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)])^2\right] + E_D\left[(E_D[F(\mathbf{x}, D)] - F(\mathbf{x}, d))^2\right] \\ &\quad + 2E_D\Big[(E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)]) \cdot (E_D[F(\mathbf{x}|D)] - F(\mathbf{x}|D))\Big] \end{aligned} \tag{3.22}$$

Le premier terme peut s'écrire :

$$E_D\left[(E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)])^2\right] = (E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)])^2$$

car l'espérance de $E[d|\mathbf{x}]$ est constante sur l'ensemble d'apprentissage D. Le terme croisé est nul car on peut utiliser cette même relation pour en simplifier la première partie et que :

$$E_D\left[E_D[F(\mathbf{x}, D)] - F(\mathbf{x}, D)\right] = E_D[F(\mathbf{x}, D)] - E_D[F(\mathbf{x}, D)] = 0$$

Ce qui donne finalement pour la moyenne de la distance à la fonction régressive :

$$E_D\left[(E[d|\mathbf{x}] - F(\mathbf{x}, D))^2\right] = (E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)])^2 + E_D\left[(F(\mathbf{x}, D) - E_D[F(\mathbf{x}, D)])^2\right] \tag{3.23}$$

Le premier terme de cette équation mesure le **biais** de l'erreur, le second représente la **variance**. Si, en moyenne, $F(\mathbf{x}, D)$ est différent de $E[d|\mathbf{x}]$, on dit que $F(\mathbf{x}, D)$ est un estimateur biaisé de $E[d|\mathbf{x}]$. Sur la figure 3.4, inspirée de [NG96], nous avons dessiné une représentation schématique de l'équation 3.23.

Soient $\mathcal{H}_\lambda$ l'ensemble des fonctions réalisables par un réseau de neurone de complexité λ (structure et nombre de poids par exemple) et $\mathcal{F}$ l'ensemble des fonctions ayant certaines propriétés (par exemple l'ensemble des fonctions m fois différentiables). On a $\mathcal{H}_\lambda \subset \mathcal{F}$.

Le biais est la distance qui sépare la fonction à rechercher $g(x) = E[d|\mathbf{x}]$ de la meilleure fonction prise dans $\mathcal{H}_\lambda$: $(E[d|\mathbf{x}] - E_D[F(\mathbf{x}, D)])^2$. C'est une erreur d'approximation, elle est indépendante des données et ne dépend que de la puissance d'approximation de la classe $\mathcal{H}_\lambda$.

Si $\mathcal{H}$ est dense dans $\mathcal{F}$, plus le nombre de paramètres (λ) augmente, plus le volume de $\mathcal{H}$ augmente et meilleure est l'approximation de la fonction régressive.

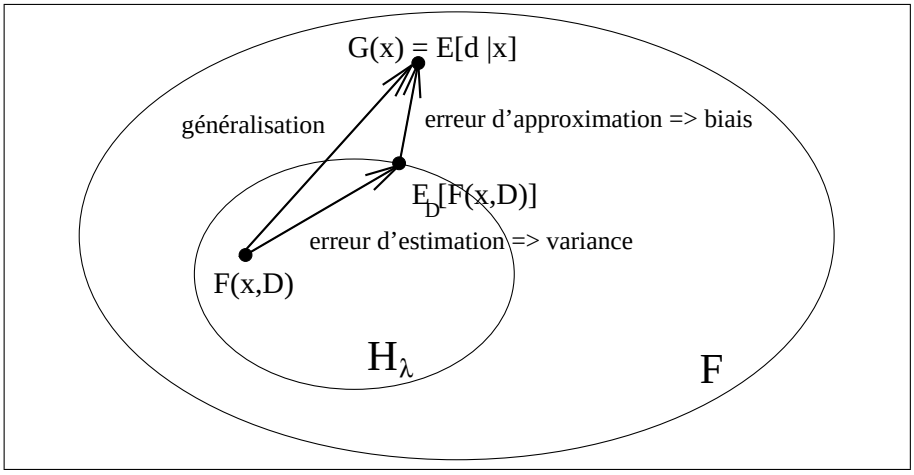

FIG. 3.4 – dilemme biais-variance.

La variance est une mesure de la distance séparant la fonction F trouvée de la meilleure fonction que l'on peut trouver $E_D[F]$, c'est donc une erreur d'estimation. C'est aussi une fonction croîssante de λ car, à nombre de données fixé, l'augmentation du nombre de paramètres conduira, en moyenne, à une augmentation de l'erreur d'estimation.

Les deux objectifs de biais faible (bonne précision) et variance faible (bonne stabilité) sont contradictoires. Quand on augmente λ on diminue le biais car le volume de $\mathcal{H}_\lambda$ augmente mais on augmente alors la variance (voir FIG. 3.5 inspirée de [FS97]).

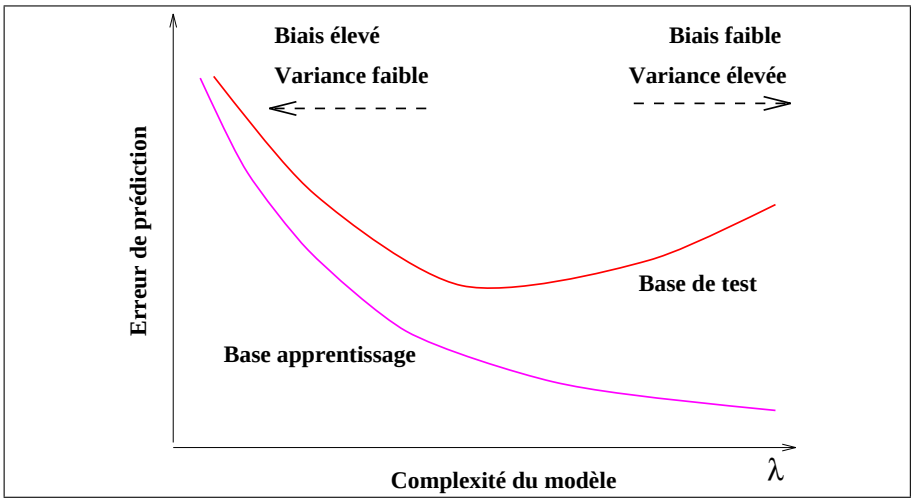

FIG. 3.5 – Comment trouver le meilleur estimateur.

Ce qui veut dire que dans le cas d'un réseau de neurones apprenant par l'exemple (ensemble d'apprentissage de taille finie), le prix à payer pour avoir un petit biais est une grande variance. C'est là une espèce de « relation d'incertitude ».

Lorsque le biais est faible, les frontières de décision sont complexes, donc le système est sensible au bruit et la variance est alors élevée. Lorsque le biais est élevé, par exemple si on filtre les données, la variance est faible et le système est moins sensible au bruit.

Pour le voir, prenons un exemple où la variable x est à une dimension et ou $g(x)$ est une fonction inconnue et bruitée.

D'une part, on pourrait définir $F(x, D)$ comme étant la fonction qui passe par tous les points. À l'évidence l'estimation n'est pas biaisée pour tous les x_i, $1 \leq i \leq N$ puisque :

$$E_D[F(x_i, D)] = E[g(x_i) + \epsilon] = g(x_i) = E[y|x_i]$$

De plus si $g(x)$ est continue, on peut espérer qu'il y a aussi très peu de biais au voisinage des points d'observation.

En revanche, si la variance de ϵ est élevée alors l'erreur quadratique moyenne sera élevée puisque la variance de F peut s'écrire :

$$E_D[(F(x_i, D) - E_D[F(x_i, D)])^2] = E_D[(g(x_i) + \epsilon - g(x_i))^2) = E[\epsilon^2]$$

qui est la variance de ϵ (ϵ ayant une moyenne nulle).

D'autre part, à l'autre extrême, on pourrait choisir une fonction ad-hoc qui ne dépendrait pas de D. À l'évidence la variance serait nulle mais le biais serait élevé puisque cet estimateur ne prendrait pas en compte les données !

Le meilleur choix sera quelque part entre ces deux extrêmes. On pourrait lisser les données par exemple ou utiliser un réseau de neurones entraîné par BP. On peut introduire volontairement du biais en choisissant certaines fonctions particulières $F(\mathbf{x}, \mathbf{w})$, ce qui diminuera la variance, mais ce ne sera valable que pour un type de problème donné. En pratique, on réalisera cela avec une architecture contrainte par, par exemple, des poids partagés, des champs récepteurs locaux, etc.

On verra plus loin qu'à un biais faible correspond une forte capacité et qu'à une capacité faible correspond une variance faible.

3.3 Théorie de l'apprentissage

3.3.1 Position du problème

Il n'existe pas vraiment de théorie mathématique décrivant les processus sous-jacents à l'apprentissage. Néanmoins on peut poser quelques principes pour l'apprentissage supervisé. Tout d'abord, le modèle sera le suivant (voir la figure 3.6) :

- l'environnement fournit le vecteur $\mathbf{x}$ dont la distribution de probabilité $P(\mathbf{x})$ est inconnue,
- le maître fournit en plus la sortie désirée pour chaque vecteur d'entrée $\mathbf{x}$, avec une distribution de probabilité $P(d|\mathbf{x})$ fixée mais inconnue. $\mathbf{x}$ et $\mathbf{d}$ sont reliés par une fonction inconnue $\mathbf{d} = \mathbf{g}(\mathbf{x})$,
- la machine est capable de réaliser une série d'applications $\mathbf{y} = F(\mathbf{x}, \mathbf{w})$.

Le problème de l'apprentissage est de choisir la fonction $\mathbf{F}(\mathbf{x}, \mathbf{w})$ qui approxime la réponse désirée (au mieux suivant un certain sens statistique) à partir d'un ensemble donné de couples {entrée, sortie désirée}. Mais une solution existe-t-elle ? Est-ce que la quantité d'informations qui se trouve dans les N données est suffisante pour construire la machine ? (C'est ce qu'on appelle le problème d'*apprenabilité*).

Soit $L(\mathbf{d}; \mathbf{F}(\mathbf{x}, \mathbf{w}))$ la mesure du désaccord existant entre la sortie désirée et la réponse effective $\mathbf{F}(\mathbf{x}, \mathbf{w})$ (par exemple la distance euclidienne $L(\mathbf{d}; \mathbf{F}(\mathbf{x}, \mathbf{w})) = \|\mathbf{d} - \mathbf{F}(\mathbf{x}, \mathbf{w})\|^2$). Sous sa forme générale on définira le risque[3] fonctionnel par :

$$R(\mathbf{w}) = \int L(\mathbf{d}; \mathbf{F}(\mathbf{x}, \mathbf{w}))\, dP(\mathbf{x}, \mathbf{d})$$

[3] *risque* est l'héritier de *risicare, doubler un promontoire* en italien du sud. La métaphore marine porte le couple aléa-action : derrière le promontoire, il y a le danger ou le paradis, pour passer le promontoire il faut naviguer. Le mot apparaît en France au XVI^{e} siècle avec le sens de hasard qui peut causer une perte.

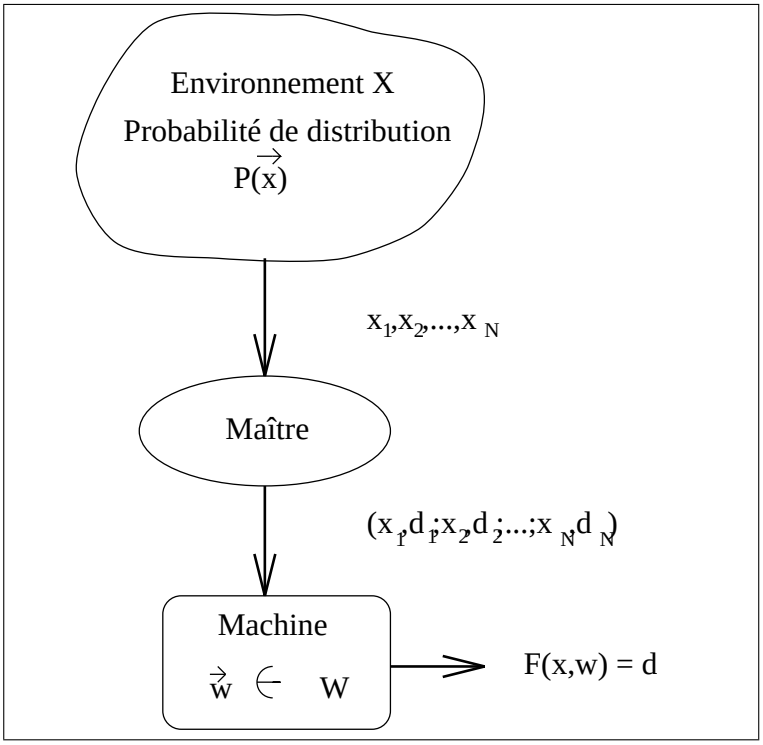

FIG. 3.6 – Synoptique du modèle d'apprentissage utilisé

où $P(\mathbf{x}, \mathbf{d})$ est la probabilité jointe de $\mathbf{x}$ et $\mathbf{d}$ (l'intégrale est de Riemann–Stieljes).
Le but de l'apprentissage est de minimiser R sur la classe des fonctions $\mathbf{F}(\mathbf{x}, \mathbf{w})$. C'est un problème difficile car la probabilité $P(\mathbf{x}, \mathbf{d}) = P(\mathbf{x})P(\mathbf{d}|\mathbf{x})$ est inconnue. Pour avancer tout de même on va utiliser un principe dit de « minimisation du risque empirique » que l'on va voir dans le paragraphe suivant.

3.3.2 Minimisation du risque empirique

La minimisation de l'erreur empirique est la traduction de l'expression anglaise consacrée : **empirical risk minimization** .
Ce risque empirique est construit sur la base d'apprentissage :

$$R_{emp}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} L(\mathbf{d}_i; \mathbf{F}(\mathbf{x}_i, \mathbf{w}))$$

qui ne dépend pas de la distribution de probabilité $P(\mathbf{x}, \mathbf{d})$ et peut donc être minimisée en théorie. Soit alors $\mathbf{w}_{emp}$ ce vecteur. Soit aussi $\mathbf{w}_0$ le vecteur de poids qui minimise l'erreur réelle. On va chercher une mesure de la différence entre ces deux erreurs.
En vertu de la loi faible des grands nombres, la moyenne arithmétique $R_{emp}(\mathbf{w})$ convergera en probabilité vers $R(\mathbf{w})$ si le nombre d'exemples $N \to \infty$. On peut donc écrire (voir la figure 3.7) :

$$\begin{aligned} \Pr\left\{ \sup_w |R(\mathbf{w}) - R_{emp}(\mathbf{w})| > \epsilon \right\} &\to 0 \qquad \text{quand} \qquad N \to \infty \\ \Pr\left\{ \sup_w |R(\mathbf{w}) - R_{emp}(\mathbf{w})| > \epsilon \right\} &< \alpha \end{aligned}$$

ce qui signifie que l'on a simultanément :

$$R(\mathbf{w}_{emp}) - R_{emp}(\mathbf{w}_{emp}) < \epsilon \tag{3.24}$$

$$R_{emp}(\mathbf{w}_0) - R(\mathbf{w}_0) < \epsilon \tag{3.25}$$

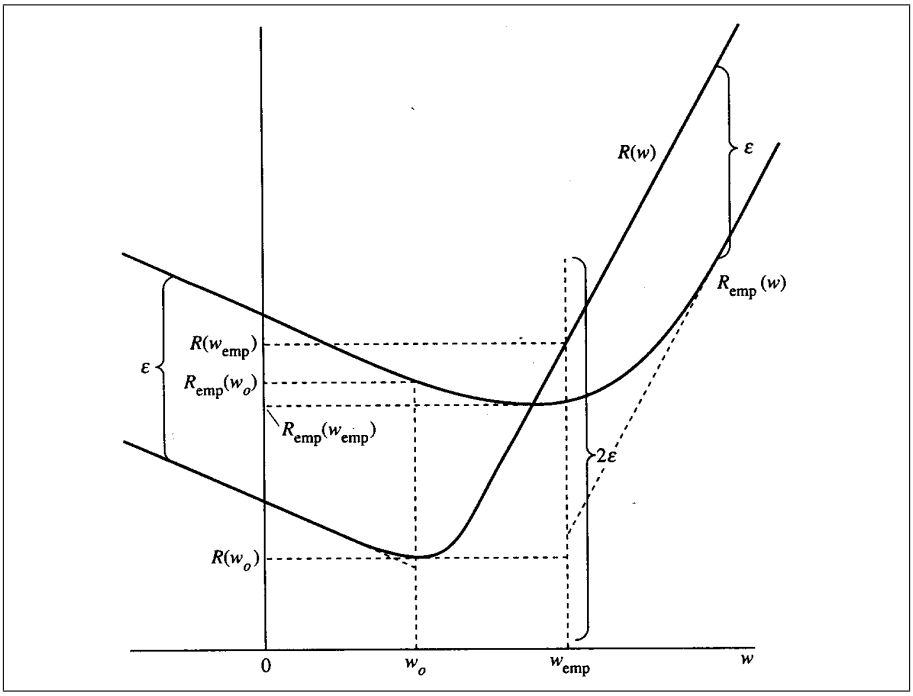

FIG. 3.7 – Illustration des inégalités relatives à la minimisation de l'erreur empirique

Effectuons la somme membre à membre des deux inégalités de 3.24, ce qui donne :

$$R(\mathbf{w}_{emp}) + [R_{emp}(\mathbf{w}_0) - R_{emp}(\mathbf{w}_{emp})] - R(\mathbf{w}_0) < 2\epsilon$$

le terme entre crochets est > 0, on peut donc écrire que l'on a, avec la probabilité α :

$$R(\mathbf{w}_{emp}) - R(\mathbf{w}_0) < 2\epsilon$$

que l'on peut transformer en :

$$\Pr\{R(\mathbf{w}_{emp}) - R(\mathbf{w}_0) > 2\epsilon\} < \alpha$$

Cette équation signifie que le risque fonctionnel converge vers le risque empirique pour chaque estimateur $\mathbf{F}$, mais elle ne garantit pas que ce soit le cas pour tous les $\mathbf{F}$ simultanément. Donc le fait que le risque empirique converge en probabilité vers le risque fonctionnel quand le nombre de données tend vers l'infini, ne garantit pas que le minimum du risque empirique converge vers le minimum du risque fonctionnel (la fonction de régression). Vapnik et Pollard ont dû introduire la notion de convergence uniforme que nous verrons plus loin.

En résumé, le principe de minimisation du risque empirique peut s'énoncer comme suit :

À la place du risque fonctionnel $R(\mathbf{w})$, construire le risque empirique sur la base des exemples d'apprentissage $(\mathbf{x}_i, \mathbf{d}_i)$:

$$R_{emp}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{d}_i; \mathbf{F}(\mathbf{x}_i, \mathbf{w}))$$

Si $\mathbf{w}_{emp}$ est le vecteur de poids qui minimise $R_{emp}(\mathbf{w})$, alors $R(\mathbf{w}_{emp})$ converge en probabilité vers la valeur minimale de l'erreur réelle $R(\mathbf{w})$ quand la taille de l'ensemble d'apprentissage tend vers l'infini pourvu que l'erreur empirique $R_{emp}(\mathbf{w})$ converge uniformément vers l'erreur fonctionnelle $R(\mathbf{w})$. La convergence uniforme étant définie par :

$$\Pr\left\{\sup_w |R(\mathbf{w}) - R_{emp}(\mathbf{w})| > \epsilon\right\} \quad \rightarrow 0 \qquad \textit{quand} \quad N \rightarrow \infty$$

3.3.3 VCdim

La VCdim d'une famille de fonctions $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in W\}$ est le nombre d'exemples qui peuvent être appris à coup sûr par la machine. Plus précisement, c'est le nombre de vecteurs $x_1, x_2, \cdots, x_n$ tels que, quelle que soit la partition de ces vecteurs, $\{F(\mathbf{x}, \mathbf{w}); \mathbf{w} \in W\}$ réalise cette partition.

C'est un concept combinatoire qui n'a pas de relation avec la notion géométrique de dimension. Dans la plupart des applications pratiques elle est difficile à évaluer analytiquement.

C'est une limite qui apparait dans les théories modernes de l'apprentissage car on peut démontrer, comme on le verra au paragraphe suivant, qu'une valeur finie de la VCdim entraîne la convergence uniforme.

3.3.4 Taux de convergence uniforme

Nous nous placerons dans le cas de fonctions de sortie binaires $d \in \{0, 1\}$, car les démonstrations y sont plus simples que dans le cas réel. Nous poserons :

$$L(d; F(\mathbf{x}, \mathbf{w})) = \begin{cases} 0 & \text{si} \quad F(\mathbf{x}, \mathbf{w}) = d \\ 1 & \text{sinon} \end{cases}$$

Dans ces conditions on notera $P(\mathbf{w})$ la probabilité moyenne de l'erreur de classification $R(\mathbf{w})$ et $\nu(\mathbf{w})$ le risque empirique $R_{emp}(\mathbf{w})$ (la fréquence des erreurs faites sur l'ensemble d'apprentissage).

Alors, d'après Vapnik [Vap82, Vap92] et la loi des grands nombres, la fréquence empirique d'un événement converge vers la probabilité de cet événement quand le nombre d'essais augmente. Donc ; quel que soit $\mathbf{w}$ et pour un ϵ donné, on a :

$$\Pr\{|P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon\} \quad \rightarrow \quad 0 \quad \text{quand} \quad N \quad \rightarrow \quad \infty \tag{3.26}$$

Ce qui n'implique pas que le vecteur de poids $\mathbf{w}$ qui minimise $\nu(\mathbf{w})$ minimise aussi la probabilité de l'erreur de classification.

Pour un ensemble d'apprentissage de taille N suffisamment grande, Vapnik a pu montrer que l'on a :

$$\Pr\left\{\sup_w |P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon\right\} \quad \rightarrow \quad 0 \quad \text{quand} \quad N \quad \rightarrow \quad \infty \tag{3.27}$$

On a convergence uniforme de la fréquence des erreurs d'apprentissage vers leurs probabilités moyennes.

On peut préciser la relation précédente en utilisant la VCdim h du système de classification :

$$\Pr\left\{\sup_w |P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon\right\} \quad < \quad \left(\frac{2eN}{h}\right)^h exp\,(-\epsilon^2 N) \tag{3.28}$$

On démontre ainsi que si h est fini, le deuxième membre tend vers 0 quand N augmente. Il a été démontré par ailleurs, voir Sontag 93 par exemple, que les réseaux à couches ayant des fonctions sigmoïdes ou exponentielles ont une VCdim finie et donc qu'ils sont apprenables.

Posons maintenant α la probabilité d'occurence de l'événement $\sup_w |P(\mathbf{w}) - \nu(\mathbf{w})| > \epsilon$, alors nous aurons, avec la probabilité $1 - \alpha$ et pour tout vecteur de poids $\mathbf{w}$:

$$P(\mathbf{w}) - \nu(\mathbf{w}) \quad < \quad \epsilon \qquad \Longrightarrow P(\mathbf{w}) \quad < \quad \nu(\mathbf{w}) + \epsilon$$

En utilisant 3.28 nous obtenons la borne suivante :

$$\alpha = \left(\frac{2eN}{h}\right)^h exp\,(-\epsilon^2 N) \tag{3.29}$$

Soit maintenant $\epsilon_0(N, h, \alpha)$ la valeur particulière de ϵ qui satisfait 3.29, alors nous obtenons l'intervalle de confiance suivant :

$$\epsilon_0(N, h, \alpha) = \sqrt{\frac{h}{N}\left[\ln\left(\frac{2N}{h} + 1\right)\right] - \frac{1}{N}\ln\alpha} \tag{3.30}$$

La limite décrite dans 3.28 avec $\epsilon = \epsilon_0(N, h, \alpha)$ est atteinte pour $P(\mathbf{w}) = 1/2$, le pire cas. Pour les petites valeurs de $P(\mathbf{w})$ Vapnik propose la relation suivante :

$$\Pr\left\{\sup_w \frac{|P(\mathbf{w}) - \nu(\mathbf{w})|}{\sqrt{P(\mathbf{w})}} > \epsilon\right\} \quad < \quad \left(\frac{2eN}{h}\right)^h exp\,(-\frac{\epsilon^2 N}{4}) \tag{3.31}$$

Il est alors facile de montrer qu'avec la probabilité $1 - \alpha$ on a :

$$P(\mathbf{w}) < \nu(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu)$$

avec :

$$\epsilon_1(N, h, \alpha, \nu) = 2\epsilon_0^2(N, h, \alpha)\left(1 - \sqrt{1 + \frac{\nu(\mathbf{w})}{\epsilon_0^2(N, h, \alpha)}}\right) \tag{3.32}$$

Ce nouvel intervalle de confiance dépend de l'erreur d'apprentissage $\nu(\mathbf{w})$. Pour $\nu(\mathbf{w}) = 0$, il se réduit à :

$$\epsilon_1(N, h, \alpha, 0) = \epsilon_0^2(N, h, \alpha)$$

Résumons ce qui précède :

- le taux de convergence uniforme vaut en général :

$$P(\mathbf{w}) < \nu(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu(\mathbf{w}))$$

- pour de faibles erreurs d'apprentissage :

$$P(\mathbf{w}) < \nu(\mathbf{w}) + 4\epsilon_0^2(N, h, \alpha)$$

- pour de fortes erreurs :

$$P(\mathbf{w}) < \nu(\mathbf{w}) + \epsilon_0(N, h, \alpha)$$

3.3.5 A faire, peut être à placer après

Erreur d'approx

erreur d'estimation

3.3.6 Minimisation de l'erreur structurelle

En Anglais : **structural risk minimization**. Cette minimisation permet de contrôler la VCdim.

On a vu que l'erreur d'apprentissage est la moyenne des erreurs commises par la machine pendant l'apprentissage et que l'erreur de généralisation est la moyenne des erreurs réalisées sur des exemples qui n'ont jamais été vus lors de l'apprentissage.

On peut poser que, avec la probabilité $1-\alpha$, pour un nombre d'exemples $N > h$ et pour toute fonction de classification $F(\mathbf{x}, \mathbf{w})$, l'erreur de généralisation est plus petite que le risque garanti défini comme étant la somme de deux termes antagonistes (Vapnik [Vap92], Guyon et al [GVB$^+$92]) :

$$\nu_{garanti}(\mathbf{w}) = \nu_{app}(\mathbf{w}) + \epsilon_1(N, h, \alpha, \nu_{app})$$

où l'intervalle de confiance ϵ_1 est celui défini par la formule 3.32.

Pour un nombre fixé d'exemple d'apprentissage, l'erreur d'apprentissage décroît lorsque h augmente alors que l'intervalle de confiance augmente de manière monotone. Ces variations sont représentées sur la figure 3.8.

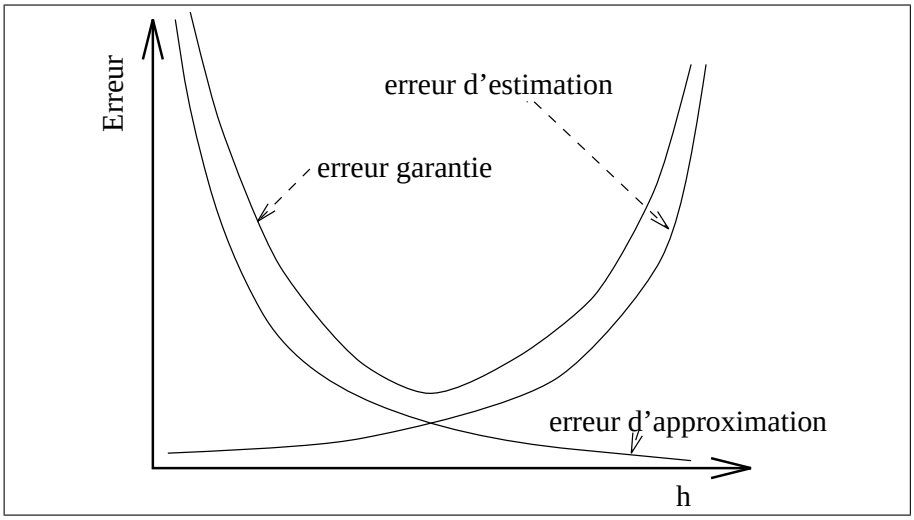

FIG. 3.8 – Illustration du dilemme biais-variance : principe de la minimisation du risque structurel lorsque la complexité du modèle varie (pour un nombre de données fixé).

Avant que le minimum ne soit atteint, le réseau est sur-déterminé : sa capacité est trop faible pour apprendre les détails. Après ce minimum la machine est sous-déterminée pour la quantité de données existante et elle commence à apprendre les idiosyncrasies de la base de données.

Le principe de mimimisation du risque structurel peut donc se résumer comme suit :

- on prend différents réseaux ayant $h_1 < h_2 < \cdots < h_n$,
- on minimise le risque empirique pour chacun et on choisit le réseau possédant le plus petit minimum,
- pour ce réseau on choisi $\nu_{garanti}$ minimum, le meilleur compromis entre les deux termes antagonistes de l'erreur garantie.

3.3.7 Contrôle de la capacité

Le but est de choisir une structure de réseau ayant une VCdim la plus faible possible et une erreur d'apprentissage la plus faible possible. Au moins trois méthodes permettent d'y arriver (Vapnik, Guyon) :

1. faire varier le nombre de neurones cachés,
2. utiliser « weight decay ». L'architecture étant fixée, le contrôle de la capacité se fait en faisant varier la norme euclidienne de $\mathbf{w}$. La minimisation de l'erreur empirique est effectuée en minimisant la fonction de coût suivante :

$$R(\mathbf{w}, \lambda_k) = \frac{1}{N} \sum L(d_i, \mathbf{F}(\mathbf{x}, \mathbf{w})) + \lambda_k \|\mathbf{w}\|^2$$

   où λ_k joue le rôle de paramètre de régularisation,
3. réduire la dimension de l'espace des entrées. Par exemple en utilisant une analyse en composantes principales.

Il existe beaucoup d'autres procédures. Pour l'instant il suffit de savoir que pour adapter la capacité d'un réseau de neurones aux données, la méthode de minimisation du risque structurel est très puissante et permet de construire le RN.

3.4 Discussion

Dans ce chapitre nous avons vu 4 règles d'apprentissage qui sont utilisées avec succès dans des RN supervisés ou non : correction d'erreurs, de Hebb, apprentissage compétitif, machine de Boltzmann.

Ce chapitre serait incomplet si on ne mentionnait pas les modèles de sélection darwinienne (Reeke 90, Edelman 87) qui supposent, entre autre, l'existence de beaucoup de sous-réseaux dont seulement certains, ceux qui sont proches de la réponse désirée, seront soumis à la procédure d'apprentissage. Il faut aussi souligner la puissance de notions comme le principe de conservation d'un maximum d'information, qui relèvent de la théorie de l'information (ce sera l'objet d'un prochain chapitre).

Nous avons introduit la VCdim. Celle-ci est reliée à un autre modèle de l'apprentissage, le PAC (probably approximatively correct) introduit par Valiant en 1984. Nous verrons cela plus tard.

3.5 Retour à la biologie

On avance souvent que l'existence du cerveau est la preuve qu'il existe des méthodes presque optimales pour apprendre avec peu d'exemples. C'est vrai que quelquefois notre apprentissage est très rapide mais cependant fiable. Il suffit de se souvenir comment un enfant est capable de mémoriser de nouveaux mots après ne les avoir entendu qu'une ou deux fois.

C'est en contradiction apparente avec les processus d'inférences stochastiques et tout ce que nous avons vu plus haut. Mais si nous supposons qu'à la construction un biais est introduit dans le cerveau alors la condradiction disparaît. Un précâblage de la représentation des objets autoriserait de bonnes performances dans les tâches de perception par exemple, ou même d'autres tâches cognitives. Le paradigme de l'apprentissage à partir d'une *tabula rasa* (i.e. apprentissage non biaisé) peut ne pas jouer un grand rôle en biologie.

On est alors en droit de se poser la question de savoir si les RN artificiels contiennent vraiment les principes de base de l'anatomie et de la physiologie des cerveaux biologiques. L'excitation dont je parlais au tout début de ce chapitre serait alors sans objet.

Beaucoup de tâches, si elles peuvent *théoriquement* être apprises, sont en fait *pratiquement* impossibles. Des propriétés importantes doivent être pré-cablées, quitte à être éventuellement affinées par la suite grâce à un apprentissage.

Le problème de l'apprentissage neuromimétique serait donc plutôt de trouver comment « conditionner » le réseau de façon à utiliser certains mécanismes, par exemple de représentation. Identifier ces mécanismes est sans doute plus fondamental que de comprendre l'apprentissage *per se*.

Bibliographie

[FS97] Françoise Fogelman-Soulié. *Statistique et méthodes neuronales*, chapter 1, pages 1–19. Thiria Sylvie, Lechevallier Yves, Gascuel Olivier, Canu Stéphane, Dunod edition, 1997.

[GBD92] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias-variance dilemma. *Neural Computation*, 4(1) :1–58, 1992.

[GVB+92] I Guyon, Vladimir N. Vapnik, B. Boser, L. Bottou, and S. Solla. Structural risk minimization for character recognition. In R.P. Lippmann J.E. Moody, S.J. Hanson, editor, *Advances in Neural Information Processing Systems*, volume 4, pages 471–479. Morgan Kaufmann, San Mateo, CA, 1992.

[NG96] Partha Niyogi and Federico Girosi. On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions. *Neural Computation*, 8 :819–842, 1996.

[Vap82] Vladimir N. Vapnik. *Estimation of dependences based on Empirical Data*. Springer-Verlag, 1982.

[Vap92] Vladimir N. Vapnik. Principles of risk minimization for learning theory. In R.P. Lippmann J.E. Moody, S.J. Hanson, editor, *Advances in Neural Information Processing Systems*, volume 4, pages 831–838. Morgan Kaufmann, San Mateo, 1992.