

Chapitre 3

Rappels sur la Conception Optimisée InTelligente

3.1 Introduction

Les entreprises disposent souvent de nombreuses informations stockées dans différentes bases de données sur des structures qu'elles ont construites ou simplement conçues. Ces bases de données sont souvent incompatibles entre elles. Elles constituent le savoir faire et elles peuvent perdre très rapidement leur transparence et leur usage (les informations sont probablement là, mais personne ne sait où elles sont ni comment les interpréter). Leur traitement par les outils classiques d'analyse statistique des données (analyse discriminante, analyse de corrélation non-linéaire, nuées dynamiques) ne peut être réalisé. De plus, les progrès des machines comme sur les connaissances fondamentales des procédés, des événements, des incidents sont aussi souvent cachés aux utilisateurs de ces données qui n'y ont accès que grâce à une interface ou un règlement sans en connaître les significations ou les raisons (l'expert a quitté la société depuis quelques années).

Comme nous l'avons souligné dans le rappel sur l'analyse à la fatigue des structures, de nombreux points ne sont pas clairs et compris par les meilleurs experts de ce domaine malgré toutes les recherches qui ont été lancées : "on sait que l'on ne sait pas tout!". Cette situation est en fait générale à de nombreux autres problèmes dans des industries très diverses.

Il y a quelques dizaines d'années, on avait pensé combler ce manque de compréhension grâce à l'Intelligence Artificielle et en particulier les générateurs de systèmes experts : une interface (fournie par le cogniticien) récupère le savoir des experts et génère le système expert qui est ensuite mis à la disposition des utilisateurs pour les aider à traiter les nouveaux cas et leur expliquer les raisons des choix et des réponses (moteur d'inférences). Cette voie s'est avérée être une impasse car en réalité, les experts, même s'ils donnaient une réponse, ne disposaient pas de toutes les règles pour résoudre le problème et ne comprenaient pas ce qu'ils faisaient.

Plus récemment, d'autres outils ont été bâtis comme par exemple le Data Mining (semblable à un virus qui se propage dans les bases de données complexes et qui "découvre" les tendances, les corrélations entre ces données!). D'autres outils, plus

rattachés à l'Intelligence Artificielle en techniques d'apprentissage automatique comme la logique symbolique, la logique floue, la logique possibiliste, les réseaux neuronaux ont aussi apporté de grands progrès actuellement.

Dans de nombreux aspects physiques (modélisation du comportement des matériaux, changements de phase, endommagement, fatigue, frottement, usure,...) mais aussi dans de nombreux aspects numériques (validité de la solution obtenue associée à une discrétisation spatiale-temporelle ou encore dans des aspects expérimentaux (représentativité des essais, mesures), le concepteur se trouve devant des verrous bloquants.

Cela a été le point de départ d'une nouvelle approche proposée par J. Zarka et ses collaborateurs dans les années 1986 : la Conception Optimisée InTelligente des Systèmes.

La partie la plus délicate et la plus importante dans cette l'approche concerne la bonne description des exemples comme nous le montrerons dans la suite de ce travail. Elle exige en effet beaucoup de temps et la participation des meilleurs experts du problème considéré.

3.2 Apprentissage automatique

Une voie importante de l'intelligence artificielle concerne l'apprentissage automatique.

Un logiciel SEA (diffusé dans l'industrie depuis quelques années) a ainsi été développé au Laboratoire de Mécanique des Solides par Sebag, Terrien, Schoenauer et Navidi mais il existe bien d'autres techniques fondées sur les réseaux neuronaux (logiciel NEUROHELL de Wards Systems), la logique floue ...

Tous ces outils sont capables de donner le réseau de règles avec la fiabilité la plus grande à partir d'une base d'exemples définie sous la responsabilité des experts. De tels systèmes SEA opèrent en trois étapes : la première consiste en la génération de règles à partir de la base d'exemples tandis que la deuxième exploite les règles générées sur des nouveaux cas inconnus ; enfin une troisième étape exploite les règles générées dans l'optimisation ou la résolution du problème inverse.

Ces systèmes disposent pour cela en général de six fonctions principales :

i) PREPARE effectue la préparation des données initiales et la récupération d'un fichier de données de type EXCEL. PREPARE considère comme descripteurs d'entrée, les nombres entiers et réels, les booléens, les alphanumériques, mais aussi les fichiers base de données (matériaux, process...), les fichiers courbes (traction, fatigue...), les fichiers signaux provenant de divers types de capteurs ou les fichiers images. Il doit aussi tenir compte que l'apprentissage automatique peut porter sur plusieurs conclusions. Il décompose principalement pour chacune des conclusions, le fichier base de données initiale en une base d'apprentissage et une base de test.

ii) APPREND extrait une base de règles à partir de la base d'apprentissage ; la base d'apprentissage peut contenir des exemples dont la description est incomplète ; elle peut également contenir des règles, qui représentent les

connaissances initiales disponibles et indiscutables des experts. Il y'a en général de nombreux paramètres empiriques à ajuster afin d'obtenir le meilleur apprentissage.

iii) ENCLAIR permet de visualiser les règles obtenues et les descripteurs retenus.

iv) TESTE permet d'évaluer la qualité des règles apprises sur les exemples mis par PREPARE dans la base de test ; on compare le diagnostic original et celui donné par la base de règles.

v) CONCLUT correspond au mode diagnostic classique d'un système expert. Il effectue le diagnostic d'un cas inconnu en utilisant la base de règles.

vi) OPTIMISE permet de résoudre le problème inverse, c'est à dire, quand des conditions sur les diverses conclusions et sur certains descripteurs sont imposées. Il propose une ou des solutions et éventuellement quand une fonction objectif est donnée, construit une solution optimale.

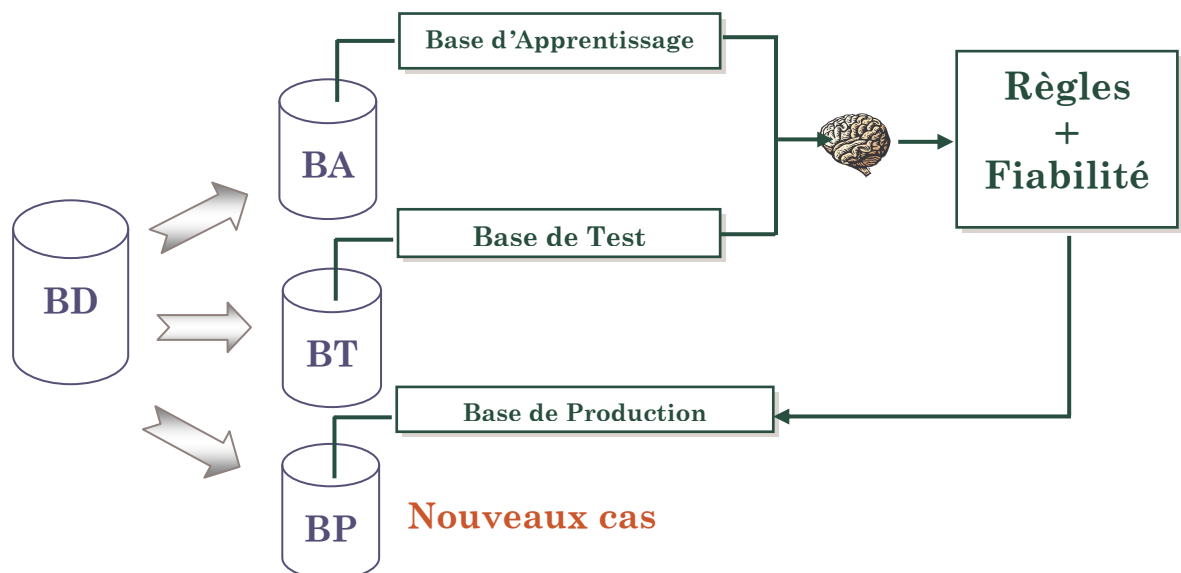


fig. 3.1 - Principes de l'apprentissage automatique

3.3 Conception Optimisée Intelligente

Il s'agit d'essayer de compenser nos nombreuses incertitudes actuelles sur le dimensionnement des structures, sur les modélisations des matériaux dans les domaines non-linéaires avec plasticité, fluage, fatigue, endommagement, rupture, sur les analyses numériques des structures, sur les états initiaux des structures, sur les chargements réels sur la structure.

Cependant, si dans ces domaines, il n'y a pas de solution précise, complète car les experts ne comprennent pas tout, de nombreux progrès ont été réalisés. Un couplage particulier de leurs connaissances et le traitement des retours d'expériences peut être effectué grâce aux techniques d'apprentissage automatique.

Le choix d'une bonne description est primordial pour toute tentative d'apprentissage automatique. De la qualité de ces descripteurs et des exemples disponibles dépend la qualité des règles construites et surtout de leur fiabilité et donc de leur généralisation.

La description d'un problème quelconque doit être vue à deux niveaux :

i) on retient d'abord les caractéristiques globales du problème : les **DESCRIPTEURS PRIMITIFS**, x où “chaque cas peut avoir un nombre différent et des natures différentes de descripteurs” ;

ii) on cherche ensuite à représenter ces caractéristiques par des **DESCRIPTEURS EVOLUES**, XX , où toute la connaissance des experts est exprimée mais où surtout, “tous les cas sont décrits par le même nombre et avec le même type de descripteurs” (il est essentiel de faire la fusion des données provenant des divers exemples). Chaque problème doit être pris séparément et être analysé par les meilleurs experts qui imposent la structure de la description intelligente spécifique à un problème. Il ne s'agit donc pas de Data Mining. Les descripteurs intelligents XX ne sont pas indépendants entre eux en général. La constitution de la base d'exemples dans sa description intelligente représente plus de 99 % du travail à réaliser pour résoudre un problème particulier.

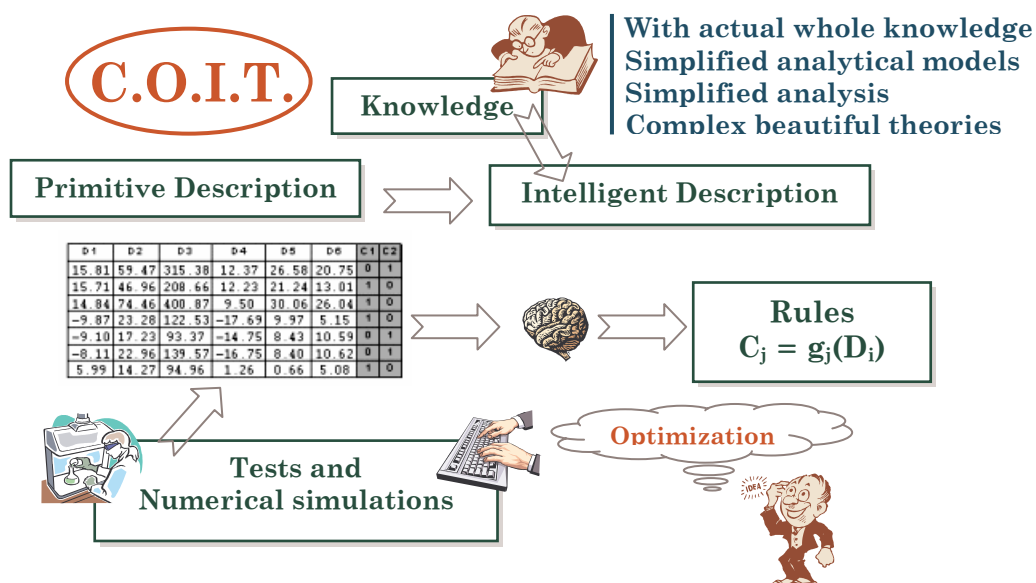


fig. 3.2 - Principes de la C.O.I.T.

Après l'apprentissage automatique sur la base d'exemples avec n'importe quel outil, il est possible de donner la solution instantanément à n'importe quel nouveau problème sans avoir à faire de nouvelles expériences ou nouvelles simulations

Il est alors aussi possible de réaliser deux types de stratégie d'optimisation (comme les temps de résolution de chaque cas sont très courts) :

i) les descripteurs évolués sont considérés comme indépendants, cela permet de découvrir éventuellement une solution qui n'a pas jamais été vue mais pour laquelle, une difficile étape doit être franchie, il faut comprendre comment réaliser réellement cette solution (semblable à l'optimisation de forme topologique),

ii) ces descripteurs proviennent de calculs ou d'essais d'un nombre de variables de conception, qui elles sont indépendantes et qui assurent que la réalisation de la solution est alors immédiate (semblable à l'optimisation structurale ou paramétrée).

J.M. Hablot (1990), dans sa thèse, a indiqué les propriétés que doivent vérifier les descripteurs d'un objet :

Un système de descripteurs doit d'abord être fixe : tous les objets sont décrits dans le même système de descripteurs. Rien ne peut être appris si chaque exemple possède son propre système de description. Cette description permet de faire la fusion des cas.

Un tel système est suffisant si on vérifie :

Conclusion (E1) \neq Conclusion (E2) \Rightarrow Description (E1) \neq Description (E2)

pour tout couple d'exemple (E1) et (E2) non contradictoires

Deux exemples sont contradictoires si leurs descriptions sont identiques et leurs conclusions incompatibles. Cela se produit dans deux cas de figure :

- l'un au moins des deux exemples est erroné ; la base est dite bruitée
- des descripteurs discriminants sont manquants, la description ne peut donc différencier ces exemples ; la base est dite inconsistante.

Les méthodes d'apprentissage doivent être suffisamment robustes pour supporter le bruit est l'inconsistance de la base d'exemples.

La description d'un objet n'est généralement ni injective ni surjective. Cela implique :

- qu'à partir de la description intelligente, on ne peut généralement pas retrouver l'objet
- que deux exemples différents peuvent être décrits de façon identique.

Le coût de la description d'un exemple doit être raisonnable. De plus, les descripteurs ne doivent pas être trop nombreux. Il faut trouver un compromis raisonnable entre la suffisance des descripteurs, le coût de leur obtention et le coût de leur traitement.

Les descripteurs doivent être indépendants de tout ce qui n'est pas significatif. En calcul des structures, les caractéristiques non significatives sont notamment les unités de calcul, les systèmes d'axe de coordonnées...

Deux méthodes permettent la description d'un paramètre dimensionné :

- un système d'unité est imposé et le paramètre est exprimé dans ce système, pour tous les exemples de la base.

- Le paramètre est rendu adimensionnel en le divisant par une quantité "de référence"

Cette approche peut renvoyer certaines informations aux experts et les aider à mieux comprendre leur problème. Mais surtout, il faut noter que cette approche conduit à une solution pratique du problème et avec en plus un accès instantané à une possible solution optimale associée à chaque nouveau cahier de charges, et cela sans avoir à faire de nouveaux essais ou nouvelles études.

Nous allons montrer dans la suite de notre travail, son application particulière à la prévision de la durée de vie d'une structure de forme générale et soumise à un chargement complexe général.

3.4 Description générale des problèmes mécaniques

Dans ce court rappel, nous ne donnons que la liste des descripteurs intelligents qui ont été proposés sans décrire les outils mécaniques qui ont été mis au point pour arriver à transformer les descriptions globales primitives des problèmes en des descriptions évoluées, intelligentes, physiquement correctes et susceptibles de s'adapter aux méthodes actuelles d'apprentissage automatique et d'optimisation en particulier par J.M. Hablot (1990) dans sa thèse.

3.4.1 Descripteurs matériaux

La plupart des termes classiques en référence avec les coefficients élastiques, les propriétés plastique, de rupture et de fatigue sont obtenus sur l'éprouvette lisse.

Il est facile de réaliser les caractérisations suivantes :

- D'abord, les courbes cycliques à $\sim \pm 1\%$ de déformation sont obtenues (ce choix est le meilleur puisque la première courbe de traction est une fonction de l'état initial du matériau, mais cela implique que le matériau est cycliquement stable) éventuellement à différentes températures. Généralement, ces courbes sont représentées par :

$$E = E^e + E^p \Leftrightarrow E = \frac{\Sigma}{Y_G} + \left(\frac{\Sigma}{K'} \right)^{1/n'}, \text{ où } K' \text{ et } n' \text{ sont des constantes.}$$

- Puis, les courbes de Wöhler sont construites à partir de chargements uniaxiaux, associés à différentes contraintes moyennes radiales et probabilités de rupture :

$$\Sigma_{max} = \Sigma_a + \Sigma_m, \text{ est la contrainte maximum,}$$

Σ_a est la contrainte purement alternée, et Σ_m est la contrainte moyenne

Comme proposé par les experts, ces courbes sont représentées par exemple par une équation de type stress-life (contrainte/nombre de cycles à la rupture ou à l'initiation) :

$$\Sigma_a = \sigma'_f(2N_a)^b - \sigma_D(N), \quad \sigma_D(N) \text{ étant la limite d'endurance à } N \text{ cycles}$$

ou de façon plus pratique, comme dans la représentation de Morrow, par une équation de type strain-life (déformation/nombre de cycles à la rupture ou à l'initiation) :

$$\frac{\Delta E}{2} = \frac{\Delta E^e}{2} + \frac{\Delta E^p}{2} = \frac{\sigma'_f}{Y_G} (2N_a)^b + \varepsilon'(2N_a)^c$$

Nous retenons seulement ces 11 coefficients dont 7 qui sont classiques en fatigue :

Module d'Young	Y_G
Coefficient de Poisson	ν
Limite élastique en traction macroscopique	σ_v
Module d'érouissage en traction macroscopique	H
Coefficient d'érouissage cyclique	n'
Coefficient de résistance à la déformation cyclique	K'
Contrainte ultime	
Limite élastique cyclique	σ'_v
Coefficient de résistance à la fatigue	σ'_f
Exposant de Basquin	b
Exposant de ductilité en fatigue	c

3.4.2 Descripteurs champ variable dans le temps

A chaque instant, n'importe quel champ $M(\underline{x})$ variant entre M_{min} et M_{max} peut être discrétisé par :

- i) l'intervalle $[M_{min} ; M_{max}]$ qui est divisé en n segments disjoints $[M_i ; M_{i+1}]$
 - ii) la proportion relative de surface (S_i/S_0) dans la région Ω pour qui le scalaire M appartient au $i^{\text{ème}}$ intervalle $[M_i ; M_{i+1}]$
- où S_0 est la surface totale de la région actuelle Ω .

Il est parfois nécessaire de décrire l'évolution temporelle des champs. Hablot (1990) a introduit les champs suivants :

- $\phi_1(\underline{x}) = \text{Sup}_t \left[\frac{M(\underline{x}, t)}{\text{Sup}_{\Omega} M(\underline{x}, t)} \right]$ qui est l'enveloppe au cours du temps, des équivalents

relatifs (compris entre 0 et 1). Ces équivalents repèrent à chaque instant les endroits les plus chargés, indépendamment de l'amplitude du chargement au temps étudié. Les tranches pourront être à altitude constante $\frac{1}{4} ; \frac{1}{2} ; \frac{3}{4} ; 1$.

Un chargement radial aura tendance à conserver les endroits les plus chargés. Inversement, un chargement complexe pourra à la limite conduire à un champ $\phi_1(\underline{x})$ équivalent à 1 sur tout le domaine. Dans ce cas, en tout point, à au moins un instant de l'évolution du chargement, le champ $M(\underline{x})$ obtenu a été maximal sur Ω .

- $\phi_2(\underline{x}) = Sup_t [M(\underline{x}, t)]$ qui est l'enveloppe au cours du temps des équivalents **absolus**, ce qui revient au plus fort chargement. Les altitudes des tranches ont une signification physique et sont choisies correspondant à des valeurs du champ égales à $M_0, 2M_0, 3M_0, 5M_0, 10M_0, 25M_0$, où M_0 est une valeur caractéristique du champ $M(\underline{x})$.

- $\phi_3(\underline{x}) = r_r Sup_t \left[\left\| grad \left(\frac{M(\underline{x})}{Sup_{\Omega} M(\underline{x})} \right) \right\| \right]$ qui est l'enveloppe des normes des gradients

rendus adimensionnels des équivalents **relatifs** (ceux calculés pour $\phi_1(\underline{x})$). Il ne s'agit ni du gradient du sup, ni du sup sur le gradient ramené à 1. Ce champ exprime au cours du temps à quels endroits de la structure se trouvent les gradients maximaux, quelle que soit à ce moment l'importance de la valeur du champ absolue.

- $\phi_4(\underline{x}) = r_r Sup_t [grad(M(\underline{x}))]$ qui est l'enveloppe des normes des gradients absolus, qui expriment les plus forts gradients atteints au cours du chargement. Il ne faut pas confondre cette quantité avec le gradient du sup, qui n'a aucun sens.

Ici r_r est une dimension caractéristique. Plusieurs autres quantités ont aussi été définies.

3.5 Principes généraux des systèmes experts par apprentissage automatique

3.5.1 Analyse de données

C'est une vieille technique qui a été développée pour expliquer, prendre des décisions et pour extrapoler de nouveaux cas. Récemment, avec le Data Mining, elle est redevenue à la mode.

Ces outils sont dédiés aux bases de données dans lesquelles chaque exemple est décrit par un nombre limité de paramètres ou descripteurs et quand un grand nombre d'exemples est disponible.

Ils ont été étendus récemment au cas à plusieurs paramètres ou descripteurs (avec du flou et/ou un manque sur les données) sont nécessaires pour décrire un exemple et seulement quelques exemples sont disponibles.

3.5.1.1 Analyse linéaire discriminante : classement

Supposons qu'il y ait g classes et N exemples chacun étant défini par m variables. L'idée principale est de trouver une représentation spéciale des données, donc un changement de variables tel que les exemples de chaque classe sont concentrés autour de leur centre d'inertie et loin de celles des classes voisines.

Une solution est obtenue en utilisant la matrice de covariance. Représentons les variables dans une base par X_{jI} .

I étant l'index de l'exemple Y_I ($I = 1 ; N$)
 j étant l'index du composant ($j = 1 ; m$)
 et α l'index de la classe ($\alpha = 1 ; g$)

Chaque classe n'a pas nécessairement le même nombre d'exemples.

La matrice de covariance totale $\underline{\underline{T}}$ est définie par :

$$T_{jk} = \frac{1}{N} \sum_{i=1}^N (X_{jI} - \bar{X}_j) (X_{kI} - \bar{X}_k)$$

où \bar{X}_j est la valeur moyenne de la j ième variable sur tous les N exemples. Elle vaut :

$$\bar{X}_j = \frac{1}{N} \sum_{i=1}^N X_{ji}$$

L'inverse de cette matrice symétrique permet de définir une norme entre les exemples (distance de Mahalanobis).

Les valeurs propres et les vecteurs propres de $\underline{\underline{T}}$ donnent un sous-espace discriminant.

- les termes diagonaux T_{jj} sont les covariances totales des j -ièmes variables, et
- les termes extra-diagonaux T_{jk} représentent les inter-corrélations entre les j -ième et les k -ièmes variables.

La matrice d'inter-corrélation $\underline{\underline{B}}$ est définie par :

$$B_{jk} = \frac{1}{g} \sum_{p=1}^g n_p \sum_{i=1}^{n_p} (X_{ji}^p - \bar{X}_j^p) (X_{ki}^p - \bar{X}_k^p)$$

où $\bar{X}_j^p = \frac{1}{n_p} \sum_{i=1}^{n_p} X_{ji}$ ($j \in$ classe p et $p \in 1 \dots g$) est la valeur moyenne de la j -ième variable de la classe p qui contient n_p termes.

Cette matrice mesure la distance entre les classes représentées par leur centre de gravité. Pour la variable j , le facteur discriminant est défini par :

$$d = \frac{B_{jj}}{T_{jj}} \quad (0 < d < 1)$$

La matrice interne $\underline{\underline{W}}$ est définie par :

$$\underline{\underline{T}} \equiv \underline{\underline{B}} + \underline{\underline{W}}$$

Elle donne une mesure de la dispersion dans chaque classe.

3.5.1.2 Régression linéaire

Cette méthode est utilisée pour exprimer une conclusion numérique C_I comme une fonction linéaire des variables X_{jI} .

$$C = b_0 + \sum_{j=1}^m b_j X_j$$

Les b_j sont appelés les coefficients de régression.

Ces coefficients sont déterminés par une méthode des moindres carrés afin de minimiser la somme des carrés des erreurs sur chaque exemple. Dans le cas d'une base de N exemples, chacun défini par une variable :

Observation	C	X
1	C_1	X_{11}
⋮	⋮	⋮
⋮	⋮	⋮
N	C_N	X_{1N}

On a :

$$C = b_0 + b_1 X_1$$

avec l'erreur :

$$err = \sum_{j=1}^N (C_j - b_0 - b_1 X_{1j})^2$$

On trouve alors les coefficients b_0 et b_1 :

$$b_1 = \frac{\sum (C_j - \bar{C})(X_{1j} - \bar{X}_1)}{\sum (X_{1j} - \bar{X}_1)^2}$$

$$b_0 = \bar{C} - b_1 \bar{X}_1$$

avec $\bar{C} = \frac{1}{N} \sum C_j$ et $\bar{X}_1 = \frac{1}{N} \sum X_{1j}$

Ces méthodes de classement et de régression ont été étendues aux cas non linéaires. M. Terrien au LMS les a introduits dans les programmes SEA version numérique.

3.5.2 Réseaux neuronaux

Parmi les métaphores biologiques employées de nos jours pour la résolution de problèmes, les réseaux neuronaux artificiels constituent certainement l'approche la plus utilisée. Ils consistent en des modèles plus ou moins inspirés du fonctionnement cérébral de l'être humain. Ils utilisent une base de connaissances (similairement à nos

expériences antérieures) afin de construire un système de neurones qui permettrait de prendre de nouvelles décisions, faire des classifications et prévisions face à de nouvelles situations/problèmes.

Les réseaux neuronaux cherchent des cas/exemples dans la base d'apprentissage, apprennent dessus et développent l'aptitude de classifier correctement de nouveaux cas ou de réaliser des prévisions et prédictions. Ils se prêtent très bien aux problèmes où la base de connaissances n'est pas obligatoirement précise.

Il existe deux grands types de réseaux neuronaux : supervisés et non-supervisés

Réseaux supervisés (classement ou régression non-linéaire)

Ces réseaux permettent de construire des modèles capables de classifier des cas, de faire des prédictions et prévisions en se référant à d'autres cas préalablement "appris". Dans un réseau supervisé, on montre au réseau comment faire la classification ou les prédictions en lui indiquant un grand nombre de classification/prédictions correctes. Les réseaux backpropagation, GRNN et PNN sont supervisés.

Réseaux non-supervisés (classification)

Ces réseaux peuvent classifier un ensemble d'exemples de la base d'apprentissage en un nombre spécifié par l'utilisateur de catégories sans savoir par avance comment les organiser. Cette tâche est réalisée en adoptant la technique du clustering (groupement). Les exemples sont regroupés en fonction de leur "proximité" dans un espace à N dimensions où N désigne le nombre de descripteurs d'entrées. Les réseaux Kohonen font partie de cette famille.

Remarque

Aucun de ces types de réseaux ne garantit de donner toujours la solution "correcte", surtout quand la base de connaissance est incomplète ou conflictuelle. Les résultats doivent être évalués en termes de pourcentage de bonnes réponses.

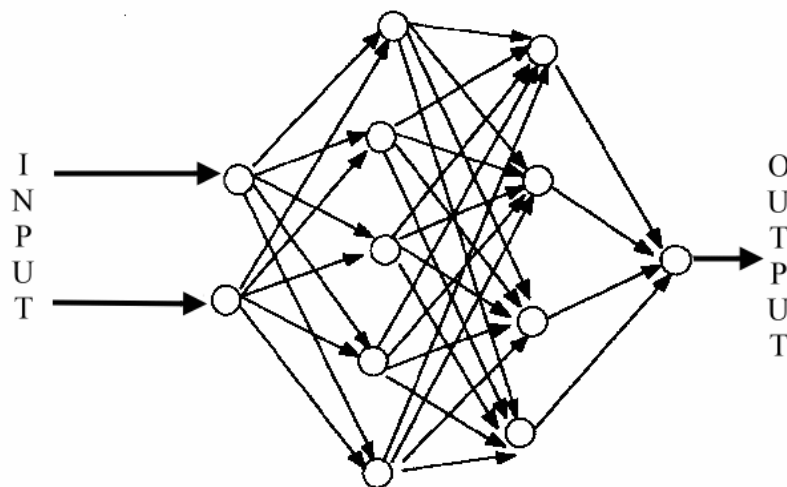


fig. 3.3 - Architecture d'un réseau neuronal à 3 couches

Suivant la méthode adoptée pour la minimisation de la fonction d'erreur, on obtient différentes règles d'apprentissage, dont la plus célèbre est certainement l'algorithme de rétropropagation (backpropagation) de l'erreur qui repose sur une minimisation par descente de gradient d'un critère d'erreur de type moindres-carrés ; il est néanmoins concevable d'utiliser d'autres méthodes de correction des paramètres, par exemple celle du recuit simulé. .

La topologie du réseau est fixée en général. Ce qui peut être un inconvénient puisqu'une autre architecture du réseau pourrait éventuellement donner un résultat meilleur et la seule façon de le savoir c'est de l'essayer à son tour.

Les réseaux sont plus efficaces s'ils disposent de plus de neurones et de couches cachées, mais le temps de recherche des coefficients augmente avec le carré du nombre de connexions.

3.5.2.1 Apprentissage

Le réseau "apprend" en ajustant les interconnexions des poids entre les couches. Les réponses produites par le réseau sont périodiquement comparées avec les réponses correctes, et à chaque instant les poids sont ajustés dans la direction des réponses correctes. Eventuellement, si le problème peut être appris, un jeu de poids stable va évoluer en s'adaptant à toutes les bonnes réponses des prédictions.

Considérons le cas simple suivant :

Nous avons n_i neurones qui correspondent au nombre de descripteurs de chaque exemple de la première couche (éventuellement avec un neurone biais) ; nous avons quelques n_j neurones cachés dans la couche cachée et n_l neurones dans la dernière troisième couche qui correspondent au nombre de conclusions.

Sur la première couche l'entrée est égale à la sortie de chaque neurone. L'entrée de chaque neurone dans la seconde couche est égale à

$$I_j = \sum_i w_{ji} O_i$$

qui donnera à la sortie O_j (en général une fonction sigmoïdale non linéaire)

$$O_j = \frac{1}{1 + \exp(-I_j)}$$

Puis, dans la dernière couche nous aurons

$$I_l = \sum_i w_{lj} O_i$$

Avec la sortie $O_l = \frac{1}{1 + \exp(-I_l)}$

L'erreur sur chaque exemple est définie par:

$$E_p = 0.5 \sum (t_{pl} - O_{pl})^2 \quad \text{où } t_{pl} \text{ est la valeur réelle.}$$

Le réseau est défini quand les poids w_{ji} et w_{ij} sont déterminés et tel que l'erreur est minimale pour chaque exemple.

Nous avons ensuite à chercher le minimum d'une fonction scalaire de ces poids. Une méthode du gradient peut être utilisée.

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial O_{pl}} \frac{\partial O_{pl}}{\partial w_{ij}} = - (t_{pl} - O_{pl}) \frac{\partial O_{pl}}{\partial w_{ij}}$$

$$\frac{\partial O_{pl}}{\partial w_{ij}} = \frac{\partial O_{pl}}{\partial I_l} \frac{\partial I_l}{\partial w_{ij}}$$

Puis, en posant $\delta_l = (t_l - O_l) O_l (1 - O_l)$ nous définissons les nouveaux poids par

$$w_{ij}^{k+1} = w_{ij}^k + \eta \delta_l O_l + \alpha (w_{ij}^k - w_{ij}^{k-1})$$

où η est le coefficient d'apprentissage et α , le facteur moment.

Il a été montré que l'erreur pouvait être prise comme :

$$\delta_h = (t_h - O_h) O_l \sum_{l=0}^{l=n_l} w_{lh} \delta_l$$

et les nouveaux poids sont calculés de la même manière :

$$w_{ij}^{k+1} = w_{ij}^k + \eta \delta_l O_l + \alpha (w_{ij}^k - w_{ij}^{k-1})$$

Quand tous les exemples ont été traités, une "période" est alors effectuée. Le point le plus difficile est de savoir quand arrêter l'apprentissage. Généralement, la meilleure solution consiste à optimiser le réseau en appliquant le réseau courant à une base d'exemples indépendante dite de test. Cette façon de procéder est un moyen d'estimer la capacité du réseau à pouvoir se généraliser et à donner de bons résultats sur de nouvelles données.

Des réseaux plus sophistiqués peuvent être construits. Par exemple, dans le programme Neuroshell de Ward Systems, plusieurs types de réseaux sont disponibles.

3.5.2.2 Architecture du réseau Probabilistic Neural Networks (PNN)

Les réseaux PNN sont capables d'apprendre sur une base d'exemples bruitée en séparant les données en un nombre spécifique de classes de sortie. Ils font partie des réseaux à 3 couches. La couche d'entrée reçoit les exemples servant à l'apprentissage et la couche de sortie dispose d'un neurone pour chaque classe possible. Il doit y avoir autant de neurones dans la couche cachée que d'exemples d'apprentissages.

Le réseau produit des activations dans la couche de sortie correspondant à la fonction de densité de probabilité à estimer pour cette classe. La sortie la plus grande représente la classe la plus probable.

Le nombre de neurones dans la couche d'entrée est égal au nombre de descripteurs d'entrées du problème et le nombre de neurones dans la couche de sortie correspond au nombre de classes.

Comme les réseaux PNN doivent classer les exemples, deux ou plusieurs descripteurs de sorties (classes) sont nécessaires. Toutes les valeurs des classes de sortie doivent être égales à 0 ou à 1.

Les réseaux PNN travaillent en comparant les exemples par rapport à la distance les séparant. NeuroShell 2 offre deux types de calculs de cette distance :

- La "*Vanilla Euclidean distance metric*", recommandée pour la plupart des réseaux, parce qu'elle donne les meilleurs résultats
- La "*City Block distance metric*", plus rapide à calculer mais qui est moins précise en général. Elle correspond à la somme des valeurs absolues des différences entre l'exemple et le vecteur poids de chaque neurone .

Lorsqu'une base de test est utilisée, la technique du calibrage ("*Calibration*") permet de trouver le meilleur facteur de lissage global associé au problème. La réussite d'un réseau PNN dépend de ce facteur, qui dépend lui-même de la qualité des exemples de la base de test.

Plusieurs techniques de recherche du facteur de lissage global sont possibles. La recherche par algorithme génétique a été retenue : les facteurs de lissage locaux associés à chaque descripteur d'entrée sont d'abord calculés puis le facteur de lissage global est déduit. Ce facteur local est en quelques sortes un ajustement utilisé pour modifier le facteur de lissage global afin de fournir une nouvelle valeur pour chaque entrée.

Cette technique génère des réseaux qui s'adaptent à la base de test mais le temps d'apprentissage est plus long. Si la base de test n'est pas suffisamment représentative, les réponses prédites sur de nouveaux exemples seront probablement mauvaises.

3.5.2.3 Architecture du réseau GMDH

La technique appelée Group Method of Data Handling (GMDH) a été inventée par le Russe A.G. Ivakhnenko puis améliorée par beaucoup d'autres chercheurs, comme A.R. Barron. Elle est aussi connue sous le nom de réseaux polynomiales ("*polynomial nets*").

Les réseaux GMDH opèrent en construisant successivement des couches. Ces couches sont reliées entre elles par des connexions complexes représentant les termes élémentaires du polynôme qui sont créés à partir de régressions linéaires et non-linéaires.

Dans chaque couche, un polynôme de degré 3 à 3 variables est construit dans le cas général:

$$P(X_1, X_2, X_3) = C_1 + \sum C_m X_i + \sum C_m X_i X_j + \sum C_m X_i X_j X_k$$

où C_1, C_2, C_3, \dots sont les coefficients du polynôme, et X_1, X_2 et X_3 sont les variables d'entrées.

Dans la première couche, les neurones sont les variables d'entrées du problème et les neurones de la seconde couche sont des polynômes comme ceux ci-dessus. Les couches ultérieures sont construites soit à partir des variables d'entrées, soit à partir des polynômes de la couche précédente ou soit à partir des deux. Ce sont essentiellement des polynômes de polynômes qui sont calculés. Seuls les meilleurs sont retenus par l'algorithme. Ils sont appelés survivants. Ce processus continue jusqu'à obtenir le meilleur réseau (selon un critère de sélection prédéfini).

Le résultat du réseau est représenté par un polynôme complexe. Les variables d'entrées les plus significatives sont indiquées par le système. L'avantage d'un tel réseau repose sur la construction de modèles très complexes en évitant le sur-apprentissage (overtraining).

Plusieurs paramètres permettent de contrôler l'architecture et la convergence du réseau :

- Nombre maximal de variables d'entrées à considérer dans chaque polynôme (≤ 3)
- Type de produits croisés ($X_1 X_2$ ou $X_1 X_2 X_3$)
- Degré maximal du polynôme (≤ 3)
- Type de critère de sélection : PSE (Prediction Squared Error), Régulation, ...

Le critère de sélection détermine quels candidats de survivants vont réellement survivre, pour pouvoir les passer à la prochaine couche. C'est le paramètre de contrôle le plus important des réseaux GRNN. De plus, une fonction objectif détermine à tout moment l'acceptabilité du modèle. Le point le plus difficile reste l'arrêt de l'apprentissage sans sur-apprentissage.

Le critère de sélection par Régulation a été utilisé lors de nos apprentissages. Il est applicable lorsqu'une base de test est définie et correspond à la moyenne du carré des erreurs du modèle sur cette base. Il est similaire au Calibrage dans les réseaux PNN dans le sens que le réseau utilise la meilleure architecture construite, sur la base de test.

3.6 Logiciels industriels utilisés

3.6.1 NEUROSHELL

NeuroShell 2 est un logiciel qui mime l'aptitude du cerveau humain à classer des données, à faire des prédictions ou à établir des décisions en se référant à l'expérience passée.

NeuroShell 2 permet de bâtir des problèmes sophistiqués, résoudre des applications sans avoir recours à la programmation. En indiquant au réseau ce qu'il faut prédire ou classer, **NeuroShell 2** est capable d'"apprendre" sur une base d'exemples avec

conclusions connues puis de réaliser ses propres classifications, prédictions ou décisions quand de nouvelles données sont présentées.

Comme le cerveau, les réseaux neuraux ne garantissent pas toujours une réponse “correcte”, surtout si certaines données sont manquantes ou bruitées. Les résultats doivent être évalués en terme de pourcentage de bonnes réponses.

C'est un générateur de systèmes experts par apprentissage automatique

3.6.2 SEA

SEA (générateur de Systèmes Experts par Apprentissage) a été élaboré au Laboratoire des Mécaniques des Solides à l'École Polytechnique. Ce logiciel a vu sa conception débiter en 1986 et est devenu opérationnel en 1988.

Le but initial recherché par ses concepteurs était de réaliser des systèmes experts en mécanique alors que les problèmes à traiter restaient encore mal connus. Il a été utilisé pour de multiples applications scientifiques et techniques dans des domaines très divers (prédiction d'erreurs lors de calculs en élastoplasticité, contrôle de fabrication, surveillance de machines, détection de pannes, diagnostics médicaux ...).

3.6.3 GENEHUNTER

GeneHunter est un logiciel d'optimisation par Algorithme Génétique de la firme Ward Systems. Il est fourni sous forme de dll Windows ou de macro complémentaire à intégrer dans Excel. L'appel de ces dll dans un programme Visual-Basic, C ou Delphi permet une utilisation personnalisée et générale du logiciel. Nous avons utilisé ce concept dans le programme FatSimPro que nous avons développé pour les calculs des fluctuations.

Bref rappel sur les Algorithmes Génétiques

Les algorithmes génétiques ont leur utilisation dans les problèmes d'optimisation où la fonction coût n'est pas dérivable et/ou quand certaines des variables de conception sont discrètes ou ne peuvent prendre que des valeurs particulières.

Ces algorithmes impliquent toujours un grand nombre d'itérations. Toutefois, ils sont aidés par le fait qu'une population d'individus d'une certaine espèce s'adaptera, après plusieurs générations, à son environnement. En d'autres termes, les membres de cette population tendront à disparaître s'ils ne peuvent pas s'ajuster, mais vivront pour se reproduire s'ils s'ajustent relativement bien ; les bons parents donneront les meilleurs enfants ; et quelques caractères génétiques de ces parents se retrouveront dans les chromosomes des enfants (évolution de Darwin).

Le principe des algorithmes génétiques est expliqué simplement de la façon suivante :

D'abord, chaque point ou individu x est codé, i.e., représenté par une chaîne ou une liste de chromosomes (ce schéma de codage est un art et dépend du problème). Chaque chromosome comporte une séquence bien définie de gènes. Ce codage doit permettre la représentation continue ou discrète des variables comme un objet dans une liste ou un objet exclusif dans une liste.

Un chromosome *continu*, est représenté selon la précision souhaitée par 8, 16 ou 32 gènes, et qui sont implémentés dans un ordinateur comme des octets binaires. Les valeurs distinctes d'un gène sont 0 et 1, et sont appelés allèles. Les chromosomes *discrets* sont employés dans les problèmes combinatoires.

Ainsi, chaque point possède une représentation interne dans l'algorithme, mais correspond finalement à un point dans le monde réel (décodage).

Généralement, les itérations sont basées sur :

- la création d'une population d'individus représentés par leurs individus
- via une fonction "fitness", l'évaluation de la performance de chaque individu
- la génération d'une nouvelle population grâce aux opérations de :
 - *sélection*, proportionnelle à leur performance
 - *croisement* et *mutation* sur les individus
- la substitution par une nouvelle population et boucle jusqu'à atteindre un certain critère.

Chacune des ces boucles est appelée *génération*.

Au départ, la population est générée de façon aléatoire ; puis quelques opérateurs génétiques améliorent la nouvelle population :

- **croisement**: 2 chromosomes ou parents, sont coupés à la même position (choisis de façon aléatoire) pour échanger ensuite leur partie pour générer 2 nouveaux chromosomes ou enfants.
- **mutation**: une petite perturbation est introduite de façon aléatoire dans le chromosome
- **reproduction**: de nouveaux individus sont créés après le croisement et la mutation
- **sélection**: il est nécessaire de favoriser les individus avec la meilleure performance

Points essentiels à retenir

Dans de nombreux problèmes industriels réels comme en conception des structures, il n'y a pas de solution précise et les experts ne comprennent pas tout, même si de nombreux progrès ont été faits.

L'approche de la Conception Optimisée InTelligente peut permettre aux experts eux mêmes de mieux comprendre leur problème. Mais surtout conduit à une solution pratique du problème et avec en plus un accès instantané à une solution optimale pour chaque nouveau cahier de charges.

Le choix d'une bonne description est primordial pour toute tentative d'apprentissage automatique. De la qualité de ces descripteurs et des exemples disponibles dépend la qualité des règles construites. Ainsi, une partie des descripteurs peuvent être inutile ou non pertinents ou alors manquante pour certains exemples de la base.

La première étape de cette nouvelle approche repose sur la construction d'une base de données d'exemples avec conclusions connues. Cette base de données est créée d'une part à partir de la récupération du savoir et d'autres parts à partir d'expériences et de simulations numériques.

La description de chaque exemple est d'abord faite par des descripteurs primitifs qui (généralement extraits des données expérimentales disponibles).

Ensuite en suivant les experts qui ont indiqué toutes les variables et descripteurs qui sont essentiels au problème, il faut transformer la description primitive en une description évoluée avec l'introduction des descripteurs intelligents et quasi-physiques (utilisation des connaissances et théories existantes). Ces descripteurs peuvent être en plus grand nombre mais surtout, ils servent à décrire tous les exemples avec le même format (même type et même nombre de descripteurs) et donc de faire la fusion des données disponibles.

Cette description assure la fusion des cas : il n'y a aucune différence entre la description d'une éprouvette lisse, entaillée et une structure complexe réelle. Ces descripteurs d'entrées peuvent être des nombres, des booléens, des chaînes de caractères, des noms de fichiers qui donnent accès à des bases de données, des courbes, des images. Les conclusions peuvent être des classes (bon/mauvais ...) ou des nombres (durée de vie, coût...)

La seconde étape repose sur la génération des règles à partir de la base d'exemples en utilisant un logiciel d'apprentissage automatique. Un niveau de confiance est associé à l'ensemble des règles obtenues.

Dans la troisième étape, si ce niveau est considéré comme suffisant, il est possible de les appliquer à des nouveaux cas.

Dans la quatrième étape, si cela est nécessaire, il est possible de traiter le problème de l'optimisation ou problème inverse, en donnant des bornes (cahier de charges) sur les conclusions et les descripteurs de conception ainsi qu'une fonction objectif (prix, durée de vie...)

Une telle démarche va être appliquée à présent au problème de la fiabilité/fatigue des structures.