TARCA ADI LAURENTIU

NEURAL NETWORKS IN MULTIPHASE REACTORS DATA MINING: FEATURE SELECTION, PRIOR KNOWLEDGE, AND MODEL DESIGN

Thèse présentée
à la Faculté des études supérieures de l'Université Laval
dans le cadre du programme de doctorat en génie chimique
pour l’obtention du grade de Philosophiae Doctor (Ph.D)

FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL
QUÉBEC

AVRIL 2004

Résumé

Les réseaux de neurones artificiels (RNA) suscitent toujours un vif intérêt dans la plupart des domaines d'ingénierie non seulement pour leur attirante « capacité d'apprentissage » mais aussi pour leur flexibilité et leur bonne performance, par rapport aux approches classiques. Les RNA sont capables «d'approximer» des relations complexes et non linéaires entre un vecteur de variables d'entrées $\mathbf{x}$ et une sortie y. Dans le contexte des réacteurs multiphasiques le potentiel des RNA est élevé car la modélisation via la résolution des équations d'écoulement est presque impossible pour les systèmes gaz-liquide-solide. L'utilisation des RNA dans les approches de régression et de classification rencontre cependant certaines difficultés. Un premier problème, général à tous les types de modélisation empirique, est celui de la sélection des variables explicatives qui consiste à décider quel sous-ensemble $\mathbf{x}_s \subset \mathbf{x}$ des variables indépendantes doit être retenu pour former les entrées du modèle. Les autres difficultés à surmonter, plus spécifiques aux RNA, sont : le sur-apprentissage, l'ambiguïté dans l'identification de l'architecture et des paramètres des RNA et le manque de compréhension phénoménologique du modèle résultant.

Ce travail se concentre principalement sur trois problématiques dans l'utilisation des RNA: i) la sélection des variables, ii) l'utilisation de la connaissance *apriori*, et iii) le design du modèle. La sélection des variables, dans le contexte de la régression avec des groupes adimensionnels, a été menée avec les algorithmes génétiques. Dans le contexte de la classification, cette sélection a été faite avec des méthodes séquentielles. Les types de connaissance *a priori* que nous avons insérés dans le processus de construction des RNA sont : i) la monotonie et la concavité pour la régression, ii) la connectivité des classes et des coûts non égaux associés aux différentes erreurs, pour la classification. Les méthodologies développées dans ce travail ont permis de construire plusieurs modèles neuronaux fiables pour les prédictions de la rétention liquide et de la perte de charge dans les colonnes garnies à contre-courant ainsi que pour la prédiction des régimes d'écoulement dans les colonnes garnies à co-courant.

Abstract

Artificial neural networks (ANN) have recently gained enormous popularity in many engineering fields, not only for their appealing "learning ability," but also for their versatility and superior performance with respect to classical approaches. Without supposing a particular equational form, ANNs mimic complex nonlinear relationships that might exist between an input feature vector $\mathbf{x}$ and a dependent (output) variable y. In the context of multiphase reactors the potential of neural networks is high as the modeling by resolution of first principle equations to forecast sought key hydrodynamics and transfer characteristics is intractable. The general-purpose applicability of neural networks in regression and classification, however, poses some subsidiary difficulties that can make their use inappropriate for certain modeling problems. Some of these problems are general to any empirical modeling technique, including the feature selection step, in which one has to decide which subset $\mathbf{x}_S \subset \mathbf{x}$ should constitute the inputs (regressors) of the model. Other weaknesses specific to the neural networks are overfitting, model design ambiguity (architecture and parameters identification), and the lack of interpretability of resulting models.

This work addresses three issues in the application of neural networks: i) feature selection ii) prior knowledge matching within the models (to answer to some extent the overfitting and interpretability issues), and iii) the model design. Feature selection was conducted with genetic algorithms (yet another companion from artificial intelligence area), which allowed identification of good combinations of dimensionless inputs to use in regression ANNs, or with sequential methods in a classification context. The type of *a priori* knowledge we wanted the resulting ANN models to match was the monotonicity and/or concavity in regression or class connectivity and different misclassification costs in classification. Even the purpose of the study was rather methodological; some resulting ANN models might be considered contributions *per se*. These models-- direct proofs for the underlying methodologies-- are useful for predicting liquid hold-up and pressure drop in counter-current packed beds and flow regime type in trickle beds.

Foreword

The core of each chapter in this dissertation is built from the results of one or more scientific articles, which at the time of the thesis submission were either published or in evaluation. The first author in all six articles is also the submitter of this PhD thesis. The introductory section in each article was edited to improve the continuity of the thesis. Certain chapters were further edited to avoid the redundancy typical of theses built from articles.

The first chapter contains the results of three articles dealing with dimensionless neural network correlations. The first article, focusing on the neural networks inputs optimization with genetic algorithms, was published in *Industrial and Engineering Chemistry Research*, 41(10), 2002, 2543-2551. The second, concerning primarily the match of prior knowledge within the neural networks was published in *Chemical Engineering and Processing,* 42(8-9), 2003, 653-662. The last article, studying the benefits of combining several neural network models, was published in *Industrial and Engineering Chemistry Research,* 42(8), 2003, 1707-1712.

In the second chapter, another alternative-- using dimensional features directly in the models-- is explored. In a fourth paper, which has been accepted by *Computers and Chemical Engineering* (2004)*,* we developed a procedure to obtain mathematically guaranteed monotonic neural networks that match concavity information. Model training was performed with a genetic algorithm - genetic hill climber optimizer.

The last chapter deals with classification issues. It is composed from two papers. While the first is still in evaluation, the second is already accepted at *Chemical Engineering Science* (2004). The former (fifth article) treats the issue of feature selection, with information theoretic or others subset goodness measure while sequential methods are used as combinatorial optimization schemes. A natural extension of this study is the sixth article, which uses the features selected in the previous study to explore a multitude of classification algorithms while keeping in mind the prior knowledge matching. As with most scientific work, many queries and attempts from this three-year research are not presented here, but were nonetheless valuable experientially.

To my parents Constantin and Afrodica for
their unlimited contribution to my education
and for their endless love.

Acknowledgments

I feel lucky to have had as thesis director Mr. Bernard P.A. Grandjean, and as co-director, Mr. Faïçal Larachi. The pragmatism of the first, combined with the enthusiasm of the second, were invaluable during this three-year experience. I appreciated their receptivity concerning all aspects related to my research work.

Beside my professors, the remaining members of the thesis jury, Mr. Jules Thibault, Mr. François Anctil, and Mr. Alain Garnier, are all acknowledged for their pertinent criticism that allowed me to improve the quality and readability of the thesis manuscript.

Table of contents

Index of tables

Index of figures

Introduction and objectives

The aim of the present work was to improve neural network modeling practice in the context of multiphase reactors data modeling. Here the neural networks served as universal function approximators, tackling the two general paradigms in statistical pattern recognition: regression and classification. While in regression an attempt is made to approximate a continuous (usually highly nonlinear) output variable given an known input feature vector, in classification the task is to assign particular realizations of the feature vector into a finite number of predefined classes (usually non-linearly separable and overlapping). In multiphase reactors, the function approximation was the typical application of neural networks thus far; i.e., formulation of some known problems in terms of classification are less likely to be encountered in the literature.

Typically, in *data mining*, one disposes of a collection of data and wants to extract as much information as possible without having the possibility to undertake other experimental observations. Our research group (Larachi F. and Grandjean B.P.A.) gathered almost all-nonproprietary information available for several types of multiphase reactors. In such situations, if one tries to use neural networks to model the complex input output relationships, the following problems are often encountered: i) feature selection (FS), ii) model design (MD) (architecture and parameters learning), and iii) qualitative prior knowledge (PK) matching. Most of the work stream in this thesis was directed toward these issues, which are by themselves active research areas dealing with neural networks and statistical pattern recognition. Besides attempting to update the multiphase reactors neural networks modeling practice, we focused on incorporating domain-specific prior knowledge into the neural network models. Examples of such prior knowledge information were *monotonicity* and *concavity* (for regression problems) and different misclassification costs and class connectivity (in classification).

The remaining part of the *Introduction* section presents a short description of the multiphase reactors general problematic, followed by the neural networks and the above-mentioned related issues. It closes with a schematic presentation of the research path followed in this work. A more detailed introduction and corresponding bibliographical review will be made in each corresponding subsequent section.

Multiphase reactors

The multiphase reactors (Gas-Liquid-Solid) are frequently used today in the chemical and petrochemical industry to conduct the hydrogenation of unsaturated oils, hydrodesulphurization of petroleum feedstocks, hydrodenitrogenation, hydrocracking, etc. (see Ramachandran and Chaudhari, 1983; Dudukovic *et al*., 2002). A function of the static of the bed, the G-L-S reactors can be delimited into two main categories: *fixed bed reactors* and *slurry reactors*. In the first category, three types of flow can be distinguished: concurrent down flow of gas and liquid (trickle beds), down flow of liquid and countercurrent up flow of gas, and concurrent up flow of both gas and liquid (packed-bed bubble column). In the second category, we usually find mechanically agitated slurry reactors (catalyst particles are kept in suspension by means of mechanical agitation), bubble column slurry reactors (the particles are suspended by means of gas-introduced agitation), and three-phase fluidized-bed reactors (particles suspended because of combined action bubble movement and concurrent liquid flow).

The design and efficiency of this equipment requires knowledge of hydrodynamics and transport characteristics: flow regimes, pressure drop, phase holdups, mass transfer coefficients, etc. The rigorous theoretical treatment derived from the first principles of the multiphase flow problem remains problematic; that is why the most of these characteristics fail to be accurately predicted using phenomenological models. A review of the multiphase reactors problematic would reveal that the preponderant knowledge of the most important aspects of three-phase reactors resides in the form of empirical correlations. Since the pioneering of the multiphase systems, data has been correlated to predict characteristics, but even today there are numerous restrictions regarding the validity of these correlations for different systems and/or values of the operating parameters. The lack of available methods to predict the key fluid dynamics parameters is emphasized also in Dudukovic *et al*. (2002). On the other hand, the current increase in experimental data availability and quality measured in the three-phase systems, together with the recent development of data mining tools such as artificial neural networks (ANN), backed by an increase of computation power, have inspired researchers to develop empirical correlations for the key characteristics of these systems.

Artificial neural networks

Actually there are many examples of successful applications of ANN computing to correlate heat, mass, and momentum transport in multiphase flow literature. ANN correlations have been derived for the pressure gradient in distillation columns (Whaley *et al.*, 1999; Pollock *et al.*, 2000) and textile fabrics applications (Brasquet *et al.*, 2000), for flooding inception and interfacial mass transfer in counter-current random-packing towers (Piché *et al.*, 2001a, 2001b), for mass transfer applications in stirred tanks (Yang *et al.*, 1999), trickle beds (Iliuta *et al.*, 1999a), and fast fluidized beds (Zamankhan *et al.*, 1997), for the displacement of water during infiltration in porous media of non-aqueous phase liquids (Morshed *et al.*, 2000), for holdups and wake parameters in gas-liquid-solid fluidization (Larachi *et al.*, 2001), for the prediction of bubble diameter in bubble columns (Jamialahmadi *et al.*, 2001), and for improving simulation of multiphase flow behavior in pipelines (Rey-Fabret *et al.*, 2001).

The most common artificial neural networks type used in such function approximation applications is the multi-layered feed-forward neural network, also known as the multi-layer perceptron (MLP) (Rumelhart *et al.* 1986). These "black-box" modeling tools have gained enormous popularity in many other engineering fields, perhaps due not only to their appealing "learning ability," but also because of versatility and performance with respect to classical statistical methods. Without supposing a particular equational form, MLPs are able to mimic complex nonlinear relationships between an input feature vector **x** and a dependent (output) variable y by consuming the information in a set of training samples of known input and output values. The parameterized neural network fitting function with a single output (approximating thus a scalar function) has the form:

$$f(\mathbf{x},\mathbf{w}) = \sigma^{(2)}\left(\sum_{j=1}^{J+1}\left(w_j \cdot \sigma_j^{(1)}\left(\sum_{i=1}^{I+1}(x_i \cdot w_{i,j})\right)\right)\right) \qquad \text{(I.1)}$$

where **w** are the free parameters called weights and I is the dimensionality of the input vector. The J activation functions in the first layer $\sigma_j^{(1)}$ and the single one in the output layer $\sigma^{(2)}$ are sigmoid functions, while the I+1 component of the feature vector, x_{I+1}, and the

J+1 activation function, $\sigma_{J+1}^{(1)}$, are set to a constant value of 1. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{I.2}$$

Such a neural network function is capable of universal function approximation, provided enough hidden neurons are available and σ is not polynomial (Cybenko, 1989; Hornik, 1990). Training of the neural network means determining the parameters **w** in such a way that the estimate produced by the neural network $\hat{y}(\mathbf{x}) = f(\mathbf{x}, \hat{\mathbf{w}})$ closely approaches the true value $y(\mathbf{x})$ on a set of training samples (the design set) $D = \{(\mathbf{x}_r, y(\mathbf{x}_r)), r = 1...n\}$. The training algorithms minimize the sum of squared prediction errors on the training samples using gradient-based techniques. See McLoone, (1997) for a review of the fast gradient-based techniques to optimize network weights.

Problems associated with neural network modeling

The general-purpose applicability of neural networks in regression (when *y* is continuous) and classification (when *y* is discrete) does pose some subsidiary difficulties that can reduce their appeal. Some of these problems are general to any modeling technique, while others are more specific to neural networks. This doctoral dissertation analyzed these problems and devised new methodologies to handle them. In the first category is the feature selection (FS) step, in which one has to decide which subset $\mathbf{x}_s \subset \mathbf{x}$ should constitute the inputs (regressors) in the model. Feature selection is a method of dimensionality reduction, which may lessen the number of samples required for model training and increase reliability of weight estimates and model performance (Jain *et al.*, 2000). Feature selection requires defining a measure of goodness of the potential subsets $\mathbf{x}_s$ as well as a combinatorial optimization algorithm to generate these solutions by maximizing the goodness measure. In the context of regression, one may use as set goodness measure the negative value of MSE (mean squared error) or AARE (average absolute relative error). In the context of classification, some class separability measures or the accuracy rate of the resulting classifier may be used to distinguish among subsets. An excellent review of features saliency measures may be found in the dissertation of Steppe (1994). Once a criterion for comparing the relevance of subsets is defined, a combinatorial optimization algorithm should be used to identify the subset maximizing the criterion. There are several kinds of such algorithms. The most straightforward is the enumerative technique, in which one has to test each possible feature combination to pick the best one. Others, like the sequential methods (see Pudil *et al.*, 1994), are stepwise; i.e., start with the single best feature (or with all features), and then add (or respectively remove) one or more features at a time. Other procedures are in the class of genetic algorithms (GA) (see Goldberg, 1989), where the relevance measure is maximized by evolving a population of possible solutions (subsets).

Other problems more specific to the neural networks include the overfitting phenomenon (also called overtraining), which appears when the neural network too closely approaches the training data points and is not able to generalize (interpolate) well in new situations. The literature proposes different approaches for preventing overfitting (see Tetko, 1997; Prechelt, 1998; Gencay, 2001), but rarely can this phenomenon be avoided, especially when data is noisy and sparse.

The non-transparency of the resulting models is perhaps the greatest deficiency of the neural networks from an engineering perspective. When one tries to interpret a resulting ANN model, the only information he may withdraw is the saliency of the input variables--i.e., a measure of their contribution at the final output, as proposed by Garson (1991) or Steppe (1994). Another interesting attempt to extract knowledge from a trained ANN model is that of Daniels and Kamp (1998), who inferred the signs of the derivatives of the function to learn y with respect to the input feature x_i, $\frac{\partial y}{\partial x_i}$, from $\frac{\partial f(\mathbf{x},\hat{\mathbf{w}})}{\partial x_i}$ where $f(\mathbf{x},\hat{\mathbf{w}})$ is the trained neural network approximating y. Here $\hat{\mathbf{w}}$ refers to the estimated (learned) weights. The possibility of reducing overfitting and consequently increasing confidence in the predictions of ANNs, while simultaneously giving more interpretability to the resulting ANN models, is to embedded prior knowledge about the characteristic to approximate, y. A common type of *a priori* knowledge encountered in multiphase reactors is the monotonicity and in some cases the concavity of y with respect to some dimensional variables describing the G-L-S system. This knowledge, sometimes referred to as the modeler's bias (Sill, 1998), is often found not only in chemical engineering (Kay *et al*., 2000) but in other fields as well, e.g., human cognition, reasoning, decision making, etc. (Abu-Mostafa, 1993; Wang, 1996). Consider that y is the pressure drop in counter current packed beds. Then one would expect $f(\mathbf{x}, \hat{\mathbf{w}})$ to predict higher output with increasing gas velocity. Of course, such qualitative prior knowledge should be supported by physical principles governing the system and be manifested within the data. ANN models matching the monotonicity prior knowledge have to be treated differently, whether the dimensional variables with respect expected monotonicity are directly fed into the network as inputs or if they are first combined into dimensionless Buckingham Π groups. In the former case, mathematically guaranteed neural networks can be obtained by constraining the signs of the neural network's weights during the training process; in the latter case they can only be checked and assessed (analytically or numerically) after training. In the context of classification, taking as an example the flow regime classification in trickle beds, there is also some prior knowledge that we may exploit to increase the transparency of the resulting ANN model and boost its expected performance. Such knowledge includes the different costs associated with various misclassifications, which means giving more penalties to more severe errors. Additionally, one would expect that the impact of an input variable on the simulated output of a classification model would exactly match the one observed in practice.

Instead of including prior knowledge in the ANN model, one can associate the ANN model with a phenomenological model, generating a hybrid predictor. Ideally this would combine the accuracy of ANN predictors with the robustness of the phenomenological model. For e.g., in a water treatment optimization problem, Côté *et al.* (1995) used a feed-forward ANN to model the errors between the simulated responses given by a mechanistic model and the corresponding experimental values. Iliuta *et al.* (1998) and (1999b), in a multiphase modeling issue, used neural networks to predict some parameters appearing in a phenomenological model. Acuna *et al.* (1999) studied several possibilities in combining neural network models with phenomenological models, resulting into so-called *grey-box* models that they applied when predicting kinetic rates for a fermentation process.

In conclusion, this work focuses on three issues in the application of neural networks to regression and classification problems: i) feature selection (FS), ii) model design (MD) (architecture and parameters learning), and iii) qualitative prior knowledge (PK) matching. All issues did not receive equal attention. For example, the model design (MD) was not treated as an issue *per se,* but this step is mandatory in any experimentation involving neural networks. However, in the second chapter we developed a customary procedure to train the network, so we added this issue to the list with the more documented ones. A synthesis of the workflow of this research is given in Figure I. 1.

Although the purpose of the study was rather methodological, some concrete finite ANN models were obtained as proof of the underlying methodologies. These models may be useful when predicting liquid hold-up and pressure drop in counter current packed beds and classifying flow regimes in trickle beds.

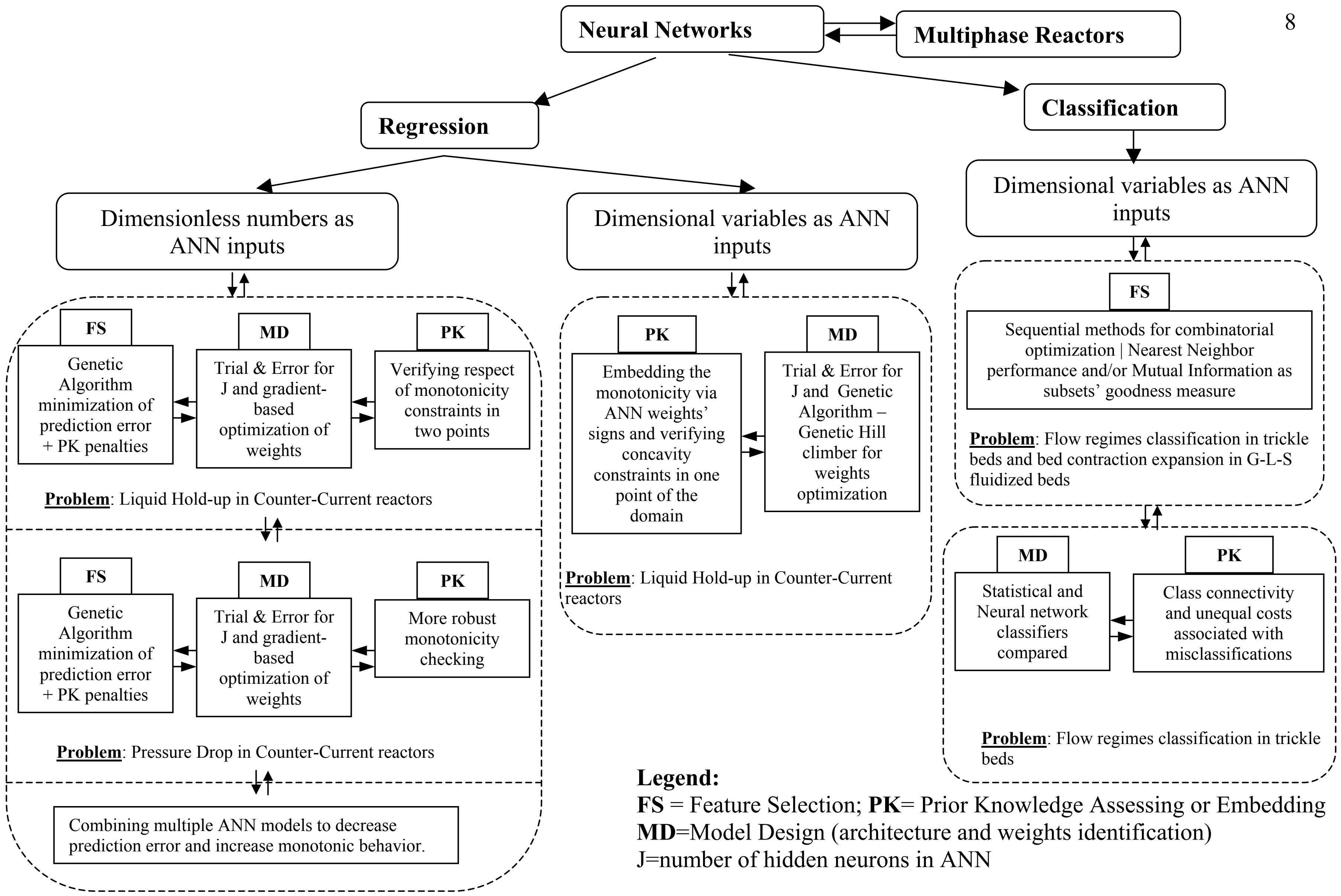

Neural Networks
Multiphase Reactors
Regression
Classification
Dimensionless numbers as ANN inputs
Dimensional variables as ANN inputs
Dimensional variables as ANN inputs
FS
Genetic Algorithm minimization of prediction error + PK penalties
MD
Trial & Error for J and gradient-based optimization of weights
PK
Verifying respect of monotonicity constraints in two points
Problem: Liquid Hold-up in Counter-Current reactors
FS
Genetic Algorithm minimization of prediction error + PK penalties
MD
Trial & Error for J and gradient-based optimization of weights
PK
More robust monotonicity checking
Problem: Pressure Drop in Counter-Current reactors
Combining multiple ANN models to decrease prediction error and increase monotonic behavior.
PK
Embedding the monotonicity via ANN weights' signs and verifying concavity constraints in one point of the domain
MD
Trial & Error for J and Genetic Algorithm – Genetic Hill climber for weights optimization
Problem: Liquid Hold-up in Counter-Current reactors
FS
Sequential methods for combinatorial optimization | Nearest Neighbor performance and/or Mutual Information as subsets' goodness measure
Problem: Flow regimes classification in trickle beds and bed contraction expansion in G-L-S fluidized beds
MD
Statistical and Neural network classifiers compared
PK
Class connectivity and unequal costs associated with misclassifications
Problem: Flow regimes classification in trickle beds
Legend:
FS = Feature Selection; PK= Prior Knowledge Assessing or Embedding
MD=Model Design (architecture and weights identification)
J=number of hidden neurons in ANN

Figure I. 1 Synthesis of research topics and methodology covered in this work. Dotted lines delineate the different investigations that constituted the object of a publishable paper.

1. Neural network dimensionless correlations for continuous multiphase reactors characteristics

1.1 Bibliographical review

1.1.1 Existing ANN dimensionless correlations

There are several types of G-L-S reactors. For most of them, researchers have tried to propose empirical correlations (data driven models) to predict their requisite characteristics. Although there are many such models, we limit our attention to neural network correlations, and, in this first chapter, only to those neural network models whose inputs are dimensionless Π Buckingham groups computed from the original variables describing the G-L-S. The more consistent databases, containing experimental observations in different types of reactors, were available to researchers when developing such predictive neural network models. The Laval University heritage is among the most comprehensive data sets in the world concerning the hydrodynamics and mass transfer characteristics of multiphase reactors. A bibliographical review of publications containing neural network modeling combined with dimensionless analysis will be given for different types of reactors and their modeled characteristics. As this review is mostly intended to inventory such correlations and not describe them in detail, please consult the Notation section of this chapter for details on some dimensionless groups appearing in the text. For the sake of simplicity, these modes are given only in a generic form, and sometimes may have identical variables as arguments. However, the models certainly differ by the values and the number of internal parameters (weights).

Concurrent down flow fixed bed reactor (trickle beds)

The first type of reactor we considered was the concurrent down flow fixed bed reactor, also known as the trickle bed. The pressure drop prediction in these systems is important, as the throughput and the mass-transfer coefficients depend on the energy supplied, which is a function of the pressure drop. Iliuta *et al*. (1999b) developed an ANN correlation for the dimensionless liquid pressure drop Ψ_L for both a high interaction regime (HIR) and low interaction regime (LIR). The equations of this model for LIR and HIR respectively are:

$$\Psi_L = f(\mathrm{Re}_L^*, \mathrm{Re}_G^*, We_L, Ga_L^*, X_L, S_b) \qquad (1.1)$$

and

$$\Psi_L = f(\mathrm{Re}_L^*, \mathrm{Re}_G^*, We_L, Ga_L^*, X_G, S_b) \qquad (1.2)$$

A phenomenological model to predict the pressure drop for the LIR regime was given by Holub *et al.* (1992) and extended by Iliuta *et al.* (1999b). Phase interaction factors included in these phenomenological models were evaluated by Iliuta *et al.* (1998, 1999b) using ANNs. This hybrid model combines the robustness of physical models with neural network accuracy, as shown by Dudukovic *et al*. (2002).

Another characteristic of interest in the concurrent packed beds is the liquid holdup (ε_l), which represents the fraction of the reactor space occupied by liquid phase. If the bed particles are porous, as will be the case of most trickle-bed reactors, the total liquid hold-up will be the sum of the internal (liquid held in pores of the catalyst) and external holdup. The external contribution can be divided into static, or residual, holdup (ε_{ls}) and dynamic, or free-draining holdup (ε_{ld}). Using the database of Laval University (F.L./B.G.), Iliuta *et al.* (1999b,c) developed an ANN correlation for the total liquid holdup for both low and high interaction regimes. The correlation for LIR has the form:

$$\varepsilon_{L,t} = f(\mathrm{Re}_L^*, \mathrm{Re}_G^*, We_L, Ga_L^*, X_G, S_b) \qquad (1.3)$$

The error of this model (measured as the average absolute relative error AARE) is lower than any of the other models in HIR, according to these authors. For the low interaction regime, it is however comparable with the error of other correlations. The overall gas-liquid mass transfer coefficient K_La can be related to the individual gas-side and liquid-side mass transfer coefficient as:

$$\frac{1}{K_L a} = \frac{1}{H_A k_g a} + \frac{1}{k_L a} \tag{1.4}$$

Same authors in Iliuta *et al.* (1999a) published several ANN models to predict the mass transfer parameters. The predicted variables in these models are: Sh_L^*, Sh_G^* (the modified Sherwood numbers of phases) and *a* (the interfacial area) function of several dimensionless numbers. The correlations are:

$$Sh_L = f(\mathrm{Re}_L, We_L, X_G, Mo_L, Sc_L, S_b) \tag{1.5}$$

$$Sh_G = f(\mathrm{Re}_L, St_L, We_G, X_G, Sc_G, S_b) \tag{1.6}$$

$$a = f(\mathrm{Re}_L, \mathrm{Re}_G, We_L, Fr_L, X_G, Eo_m, S_b) \tag{1.7}$$

The above relations are valid for both LIR and HIR.

Counter-current trickle-beds

The second type of reactor for which dimensionless neural network correlations are encountered is the counter-current trickle-bed. One basic design parameter here is the loading capacity. It designates the smallest superficial gas velocity, which, at a given superficial liquid velocity "causes a discernable build-up of liquid" (Leva, 1953). The first ANN correlation that predicts the loading capacity in these systems was presented by Piché *et al.* (2001c). The correlation is given for the Lockart –Martinelli parameter that embeds the gas superficial velocity at loading:

$$\chi = f(\phi, \mathrm{Re}_L, Ga_L, St_L, S_B) \tag{1.8}$$

A second important characteristic of counter-current systems is the flooding capacity, or the maximum amount of fluid the bed can hold without overflowing. There are many given definitions of this phenomenon, which were presented by Silvey and Keller (1966), but it is basically the operating point beyond which a tiny increase in gas velocity produces a substantially important change in the pressure drop and liquid hold-up in the column.

Recently, an ANN correlation for the gas superficial velocity at flooding ($U_{G,Fl}$) was proposed by Piché *et al.* (2001a). The correlation is given for the Lockart-Martinelli parameter that includes the gas superficial velocity at flooding point:

$$\chi = f(\phi, \mathrm{Re}_L, Ga_L, St_L, S_B) \tag{1.9}$$

Piché *et al.* (2001d) proposed a correlation for the dimensionless frictional pressure drop:

$$f_{LGG} = f\left(\mathrm{Re}_G, Ga_G, \mathrm{Re}_L, Ga_L, St_L, S_B, \chi\right) \tag{1.10}$$

and for the liquid hold-up (2001e) :

$$h_T = f\left(Fr_G, St_G, \mathrm{Re}_L, Fr_L, Oh_L\right) \tag{1.11}$$

The gas to liquid mass transfer coefficients were correlated using the same approach by Piché *et al.* (2001b) via the dimensionless gas (or liquid) film Sherwood number ($Sh_{L/G}$) as a function of six dimensionless groups: Reynolds (Re_L), Froude (Fr_L), Eotvös (Eo_L), the gas (or liquid) Schmidt number ($Sc_{L/G}$), the Lockhart-Martinelli parameter (χ), and a bed-characterizing number (K). Using the ANN correlation and the two-film theory, a reconciliation procedure was implemented, resulting in better predictions of the gas (or liquid) overall volumetric mass transfer coefficients.

Concurent up-flow packed beds

The last type of fixed bed reactors that we will discuss is the concurrent up-flow packed beds. The frictional pressure drop in these systems has been investigated by a number of researchers, including Turpin and Huntington (1967), who used the friction factor approach, and Colquhoun-Lee and Stepanek (1978), who suggested that the two-phase pressure drop data should be correlated with a single-phase energy dissipation of liquid. Larachi *et al.* (1998) proposed the following ANN correlation:

$$f_{LG} = f\left(X_L, Fr_L, St_L, \mathrm{Re}_{LG}, Mo_L\right) \tag{1.12}$$

between the gas-liquid frictional pressure drop and the physical properties of the phases embedded into the dimensionless numbers given in (1.12).

Gas and liquid holdup are also important design parameters of three-phase fixed-bed reactors with concurrent up-flow. If one variable is known, the other can be estimated from the equation:

$$\varepsilon_G + \varepsilon_L = \varepsilon_B \tag{1.13}$$

An ANN correlation for the external liquid holdup was determined by Bensettiti *et al.* (1997):

$$\varepsilon_L = f\left(X_L, Ca_L, S_b, \mathrm{Re}_{LG}, Eo_m, \mathrm{Re}_L / Fr_L\right) \tag{1.14}$$

The above bibliographic review contains most of the ANN correlations built for counter-current trickle-bed, concurrent trickle-beds, and fixed-beds with concurrent up-flow. Some correlations are compared with other empirical and / or phenomenological models in Iliuta *et al*. (1999d). The neural network models published by these authors have multi-layer perceptrons with logistic sigmoid transfer functions in the hidden and output nodes. Weights were determined using the computer software developed by Cloutier *et al.* (1996), which uses Broyden-Fletcher-Goldfarb-Shanno's method (Press *et al*., 1989) to minimize the sum of squared prediction errors on a training data set. One way to verify that the model retained the main tendencies in data and was not overtrained was based on monotonicity tests simulating the output of the model in some ranges of the dimensional variables. The authors also had to select the most suitable dimensionless numbers to use as networks input vector.

The chemical engineering literature presents same kind problems for other types of chemical reactors. Jamialahmadi *et al*. (2001) developed a dimensionless RBF (radial basis function) neural network correlation for the bubble diameter in bubble columns. These authors did not devise the resulting model to be used as a predictor for the respective characteristic (the bubble diameter), as was the case in previous research; instead, they used the ANN model to generate artificial data (input-output pairs) to fit an imposed form empirical correlation, which was less accurate, but still gave a low prediction error. In all of these studies, however, the ANN was checked to see if it would predict smooth monotonic outputs when some dimensional variables (here, the liquid viscosity and surface tension) composing the dimensionless inputs of the ANN (here Bond, Froude, and Galileo numbers) were increased.

1.1.2 Study target problematic and current procedures

Until recently, identifying the most relevant ANN model's inputs, generally dimensionless Buckingham Π groups, has been a laborious trial-and-error procedure. It consists of choosing an arbitrary combination of inputs and training on a learning data set several ANN models differing by the number of nodes in their hidden layer, J. The resulting models are further tested on a validation data set to evaluate their generalization performances. The ANN model to be retained among all the simulated ones is the one that yields the smallest relative error on both training and generalization data sets. Thence, the topology of the model is thoroughly tested for phenomenological consistency within the valid range of the working database to determine whether it exhibits the expected trends. Any inconsistent behavior disqualifies the choice.

Until now, this time-consuming approach was not automated because the human expertise regarding the phenomenological consistency was somehow difficult to formulate mathematically into an optimization criterion. Nothing ensures that this blind-search approach can successfully identify the most relevant set of dimensionless inputs. This is especially true in multiphase flow context, where the dimensionless groups abound and the combinatorial problem is explosive. Finding the best ANN model would become a matter of chance. Assuming agreement between an ANN model and the expected physical evidence can be assessed automatically using an *expert-system* of rules, the trial-and-error method would become suitable for a computer algorithm. However, the main problem would remain: how do we find the fittest input combination when the evaluation of all the combinations is CPU time-consuming?

Genetic algorithms (GA) have been successfully applied to such combinatorial problems where high quality solutions within reduced search times are needed. Based on the mechanisms of natural selection and natural genetics, GAs can extract the information from evaluated input combinations, i.e., parent specimens, while assuring good exploration of the search space (Goldberg, 1989). GAs have also been combined with ANNs in several different ways. GAs have been used to generate i) the ANN connectivity weights (Morshed *et al*., 1998), ii) the ANN architecture (Blanco *et al.* 2000), and iii) ANN architecture and weights simultaneously (Gao *et al*., 1999)

Problem statement

The dimensionless neural network models identification given the discussion so far could be summarized simply as the following:

> Having, as in Table 1.1, a sufficiently large database (N occurrences) in the form of dimensionless Buckingham Π groups, where M candidate Π groups, CI_1, CI_2 … CI_M (N>M >> 1) embed redundantly the physical and operational parameters stemming from a process, find an ANN model that uses, as inputs, only a subset of m pertinent Π groups among the M ones, in order to predict an output *y*, a key process characteristics. The three-layer ANN model to identify, in the case of a single output, is described by the following set of equations (using normalized data and sigmoid activation functions):

$$\hat{y} = \frac{1}{1+\exp\left(-\sum_{j=1}^{J+1} w_j H_j\right)} \tag{1.15}$$

with $1 \le j \le J$

$$H_j = \frac{1}{1+\exp\left(-\sum_{i=1}^{m+1} w_{ij} I_i\right)} \qquad 1 \le j \le J \tag{1.16}$$

where:

m is the number of inputs, I

for $1 \leq i \leq m$, $I_i = CI_{S(i)} \in \{CI_1, CI_2, \ldots CI_M\}$ with S, an m-input selector (sub-set).

J is the number of hidden neurons in the middle layer; $I_{m+1} = H_{J+1} = 1$ are the biases;

w_j and wij are the connectivity weights.

The models described by these equations should fulfill the following requirements:

i) Accuracy: The model must be very accurate, preferably to the level of experimental error with which the output is measured.

ii) Phenomenological consistency: The model to be built must preserve, at least within the database-documented domain, the expected trend of the output (monotonic increasing or decreasing) in accordance with all known aspects of the process physics.

iii) Low complexity: The ANN model must preferably involve a minimal number of inputs (m Π groups) and hidden neurons (J), resulting in a correspondingly minimal number of connectivity weights (the $m \cdot J + 2 \cdot J + 1$ neural fitting parameters).

Table 1.1 Typical structure of the database for applying the GA-ANN methodology

Candidate Inputs (independent variables)				**Output (dependent variable)**
CI_1	**CI_2**	…	**CI_M**	***y***
$CI_{1,1}$	$CI_{1,2}$	…	$CI_{1,M}$	y_1
$CI_{2,1}$	$CI_{2,2}$	…	$CI_{2,M}$	y_2
…	…	…	…	...
$CI_{N,1}$	$CI_{N,2}$	…	$CI_{N,M}$	y_N

1.2 Genetic algorithm-based procedure to develop dimensionless ANN correlations matching phenomenological prior knowledge

1.2.1 Methodology description

The present contribution is intended to provide an integrated GA-ANN methodology to facilitate the development of an ANN regression model (three-layer perceptron type) on a given problem. The GA implemented in this study was designed to identify the most relevant ANN input combination resulting in a neural model. This is done by minimizing a multi-objective criterion that includes ANN prediction errors on the learning and generalization data sets, and, most importantly, a penalty function that embeds the phenomenological rules accounting for ANN model likelihood. The integrated GA-ANN methodology is illustrated on a comprehensive liquid holdup database of counter-current randomly-dumped packed towers with the aim of finding the best liquid hold-up ANN correlation.

A typical approach in solving multi-objective problems consists in optimizing a primary response function while turning the other functions into constraints (Viennet *et al*., 1995). The genetic algorithm practice, on the other hand, consists of optimizing a composite objective function which sanctions violations of the restrictions by means of the penalty method (Goldberg, 1989). In our problem, we combined approaches. Table 1.2 reports, in a hierarchical order, the parameters to be identified and how their searches have been managed and integrated.

As the number of inputs, m, and the number of hidden nodes, J, must be low to minimize model complexity, they have been varied by discrete sweeps in selected ranges. m has been varied in the range [4;6] and J has been varied in the range [2m-1;2m+3], as suggested by Maren (Maren *et al*., 1990, pag. 240). The determination of the input selector **S** consists of the identification of m pertinent inputs among M ones. This proves to be a tedious task; the search space of combinations is large, and the solution S must meet both phenomenological consistency and accuracy of the resulting ANN model.

Table 1.2 Parameter identification strategy

Parameter to identify	Search method	Objective function
m	Trial and error in the range $[m_{min}, m_{max}]$	Expert decision
S	Genetic algorithm, using binary bit strings	Multi-objective Fitness Eq. (1.19)
J	Trial and error in the range $[J_{min}, J_{max}]$	Multi-objective Criterion Eq. (1.17)
$\mathbf{w_j}$; $\mathbf{w_{ij}}$	BFGS variable metric method	Least square on prediction errors $SSE = \sum_{k=1}^{N_T} (y_k - \hat{y}_k)^2$

m, number of inputs; **S**, input selection operator; **J**, number of hidden nodes; $\mathbf{w_j}$, $\mathbf{w_{ij}}$, connectivity weights; N_T number of training samples

There are few search techniques-- such as: enumerative technique, random walk, simulated annealing, and GA-- that find solutions over discrete domains using only the value of the function in different points of the domain. Because of its robustness (Goldberg, 1989) and natural appeal, the GA technique is employed in this work for searching the best-input selector **S**. The connectivity weights w_{ij} and w_j are adjusted by minimizing the sum of squares of the prediction errors on part of the data, referred to as the training data set, using the Broyden-Fletcher-Goldfarb-Shanno's variable metric method (Press *et al.*, 1989). The remaining data are used to evaluate the generalization capability of the ANN model. This step has been processed with a slave software, NNFit (Cloutier *et al.*, 1997). The integrated GA-ANN procedure used to handle the problem is presented in Figure 1.1 and will be detailed in the following sections.

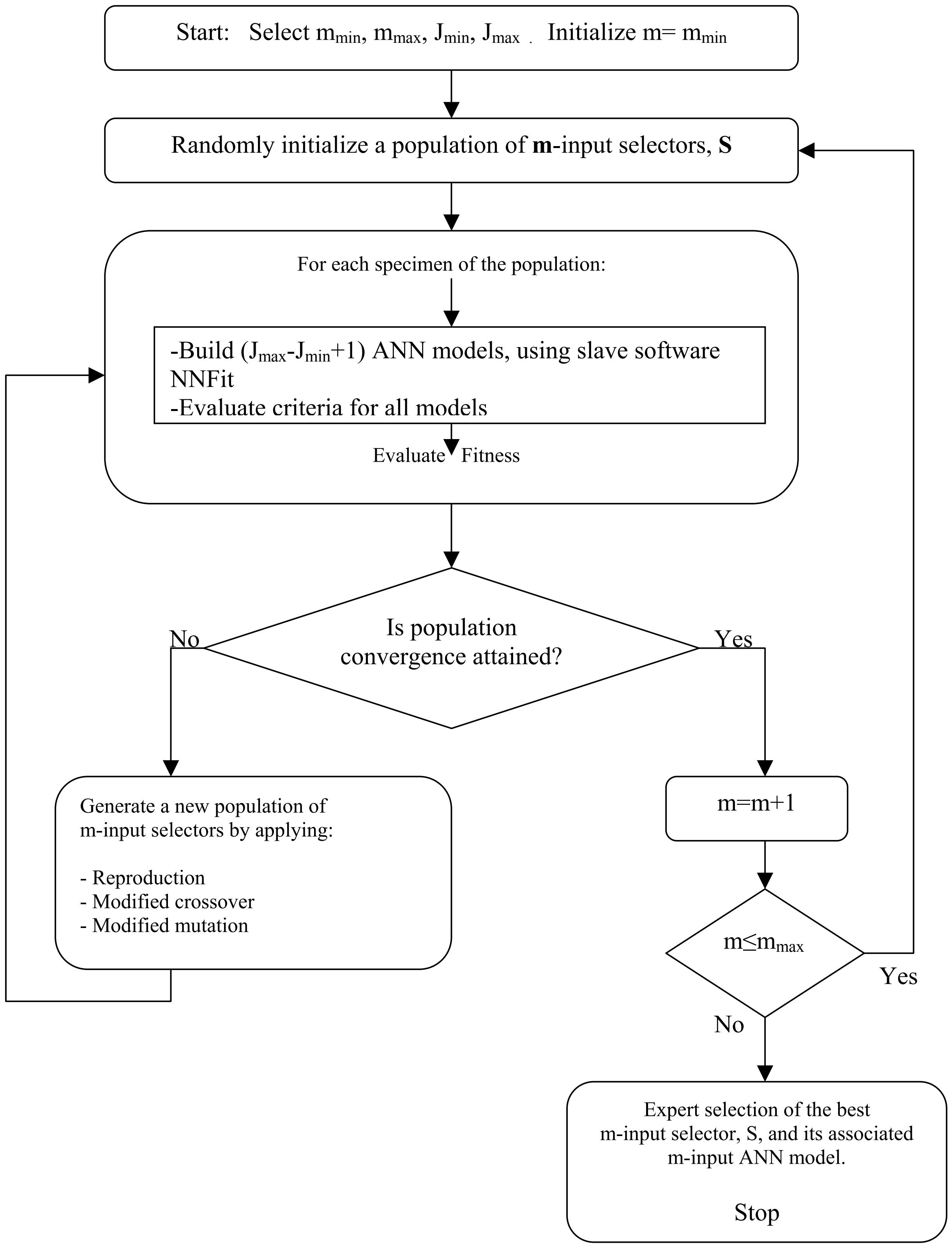

Start: Select m_{min}, m_{max}, J_{min}, J_{max} . Initialize m= m_{min}
Randomly initialize a population of **m**-input selectors, **S**
For each specimen of the population:
-Build (J_{max}-J_{min}+1) ANN models, using slave software NNFit
-Evaluate criteria for all models
Evaluate Fitness
Is population convergence attained?
No
Yes
Generate a new population of m-input selectors by applying:
- Reproduction
- Modified crossover
- Modified mutation
m=m+1
m≤m_{max}
Yes
No
Expert selection of the best m-input selector, S, and its associated m-input ANN model.
Stop

Figure 1.1 Logical flow diagram for the GA-ANN methodology.

1.2.1.1 GA Encoding solutions

The GA approach requires a string representation of the m-input selector, **S**. In our context, **S** is a selection of indices representing some of the candidate input variables of the database sketched in Table 1.1. The encoding modality chosen was M-sized bit strings, allowing only m "one bit" values per string (Figure 1.2). In this binary representation of solutions, M corresponds to the total number of candidate input variables (or input columns) in the database (Table 1.1). The "1" at a given rank of the string stands for a selected input occupying the same rank in the database. Conversely, the "0" stands for an input variable being discarded.

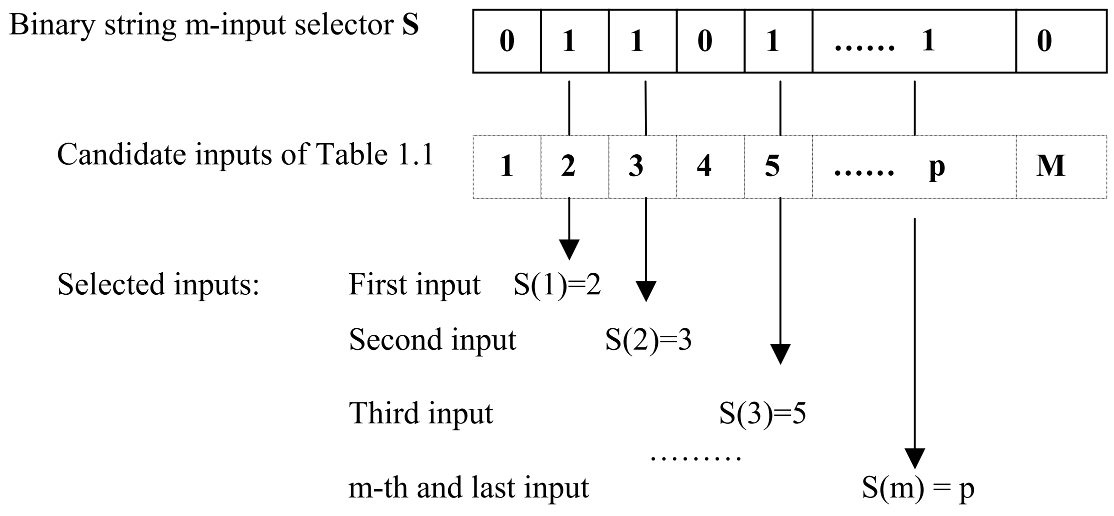


Figure 1.2 Bit string representation of the m-input selector, **S**.

1.2.1.2 Multi-objective criterion and fitness function

To identify the best input selector S and its related ANN model fulfilling the requirements i)-iii) of Section 1.1.2, the composite criterion Q was formulated:

$$Q(\mathbf{S}) = \min_{J_{\min} \le J \le J_{\max}} Q_J(\mathbf{S}) \qquad (1.17)$$

with

$$\mathrm{Q_J}(\mathbf{S}) = \alpha \cdot AARE[ANN_J(\mathbf{S})]_{\mathrm{T}} + \beta \cdot AARE[ANN_J(\mathbf{S})]_G + \gamma \cdot PPC[ANN_J(\mathbf{S})] \qquad (1.18)$$

In Eq. (1.18) AARE[ANN$_J$(**S**)]$_T$ is the average absolute relative error the ANN (having J hidden nodes) achieves on the *training* data set for a given input combination (or specimen) **S**. Equivalently, AARE[ANN$_J$(**S**)]$_G$ measures is the accuracy of the ANN model on the *generalization* data set remaining after optimizing the neural connectivity weights using the training set. The composite criterion of a *penalty* for *phenomenological consistency*, PPC[ANN$_J$(**S**)] ideally guarantees that the model will exhibit the behavior expected of the simulated output. By "expected behavior" we mean an ensemble of prescribed *behavioral rules* known to govern the phenomenon of interest, and which are embedded, as will be shown in §1.4.2, in the term PPC[ANN$_J$(**S**)]. Ideally, the penalty term is zero if the topological features of the ANN function meet all the rules. The role of the weighting multipliers α, β, and γ is to enable more flexibility in targeting models that fit better the training data set or finding models that generalize better while still satisfying, through the PPC term, the phenomenological consistency at various degrees. The choice of α, β, and γ values is described in the next sections. The stepwise construction of the criterion Q(**S**) is shown in Figure 1.3.

In the genetic algorithm practice, *fitness maximization* is preferred to the classical minimization problem. Hence, the better the solution, the greater is its fitness value. Since every minimization problem can be turned into a maximization problem, the composite criterion Q(**S**) can easily be switched into a fitness function using the simple linear transformation (Friese *et al.*, 1998):

$$Fitness(\mathbf{S}) = C \cdot Q_{\max} - Q(\mathbf{S}) \qquad (1.19)$$

where C is a conversion coefficient greater than 1 to ensure positive fitness function values, and $Q_{\max}$ is the maximum value of Q among the population having MAXPOP specimens, **S**.

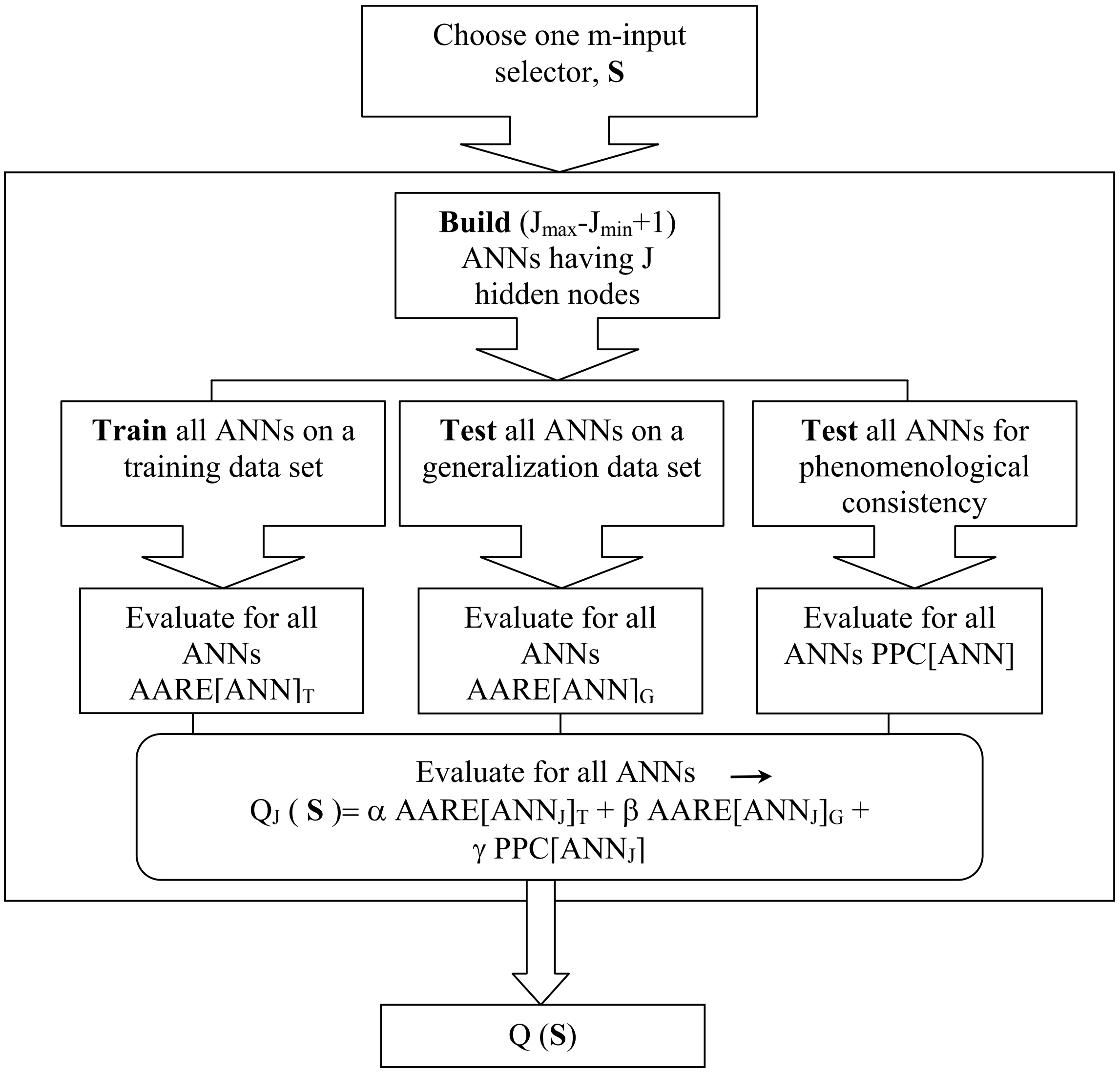

Choose one m-input selector, **S**
Build (J_{max}-J_{min}+1) ANNs having J hidden nodes
Train all ANNs on a training data set
Test all ANNs on a generalization data set
Test all ANNs for phenomenological consistency
Evaluate for all ANNs AARE[ANN]$_T$
Evaluate for all ANNs AARE[ANN]$_G$
Evaluate for all ANNs PPC[ANN]
Evaluate for all ANNs →
Q_J (**S**)= α AARE[ANN$_J$]$_T$ + β AARE[ANN$_J$]$_G$ + γ PPC[ANN$_J$]
Q (**S**)

Figure 1.3 Stepwise construction of the composite criterion, Q(**S**).

1.2.1.3 Building the generations

Starting with a null M-sized string (all bits are zero), each first-generation specimen was built by turning randomly and equally probable m zeroes among the M into ones. The operation was repeated MAXPOP times. A uniform random number generator based on the Knuth subtractive method (Press *et al.*, 1989) was used throughout this work. Once the initial population was available, it was allowed to evolve in order to better identify specimens that maximized the fitness function Eq. (1.19). The evolution process rested on the so-called reproduction, recombination (crossover), and mutation operators pioneered in the area of artificial systems by Holland (1975).

Reproduction operator

The purpose of this operator is to ensure that the fittest specimens perpetuate through off-springs and/or have greater chances to be found in the next generation. Numerous schemes are known which introduce various levels of determinism into the selection process. Among them, three have been tested in this work: the roulette wheel selection with elitism, the stochastic remainder selection without replacement, and the stochastic remainder selection without replacement with elitism (Goldberg, 1989; De Jong, 1976). This third method was the one retained for our genetic algorithm. With this method, best individuals are receiving a number of identical copies in the next generation. The number is a function of its fitness; however, even genetically inferior individuals could be duplicated in the next generation.

Modified Recombination Operator

No matter how perfect, reproduction does not create *new* better specimens; recombination and mutation do. The recombination operator combines useful features from two different specimens, yielding offspring. For instance, classical two-point crossover recombination – taking a random start point and length for the selected sub-string – would produce from two parent specimens (A nd B) two new children by simply interchanging a selected region in specimen A with that corresponding in specimen B. Though this recombination proves efficient for unconstrained GAs (Frantz, 1972; De Jong, 1976), it was unsuitable in our context because the compulsory m one-bit values in the specimens were not automatically preserved during parent-offspring transition. The crossover operator was modified to ensure *conservative* passage with fixed m one-bit values in the offspring. This was done by splitting the crossover operation into two distinct steps.

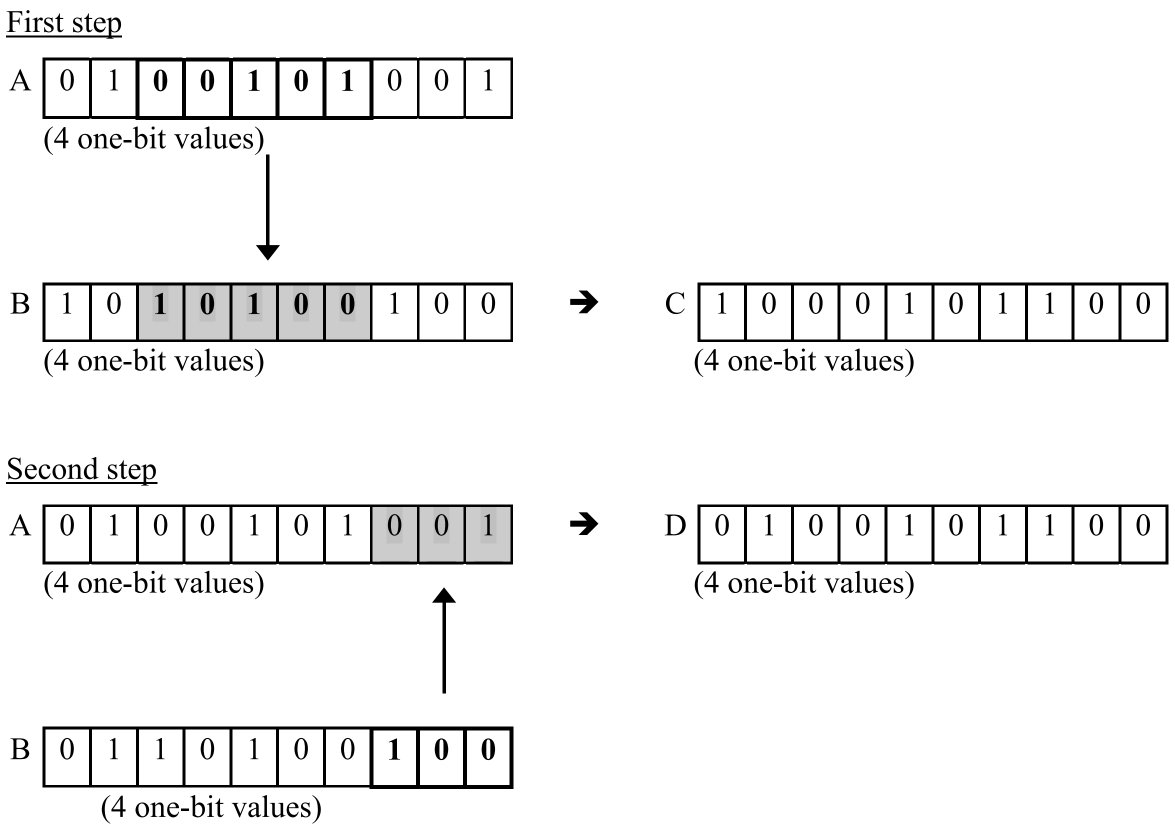


Figure 1.4 Successful trial to produce 4 one-bit valued specimens in the modified two-step two-point crossover

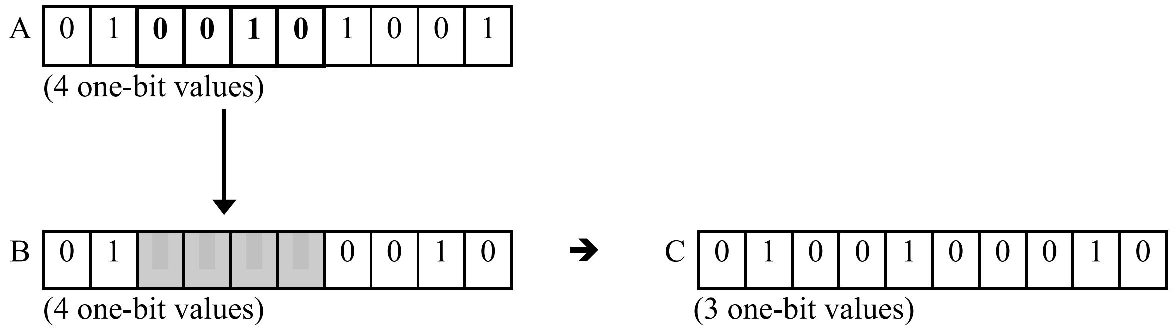


Figure 1.5 Unsuccessful trial to produce a 4 one-bit valued specimen in the first step of the modified two-point crossover

In the first step (Figure 1.4), a sub-string in specimen A (with random length and start point) was selected and transferred in specimen B at the same location. The resulting child, specimen C, was retained, provided it possessed m one-bit values like his parents (Figure 1.4) and it was distinct. If not (Figure 1.5), the first step was repeated until this condition was satisfied or the number of trials exceeded a given value. The second step was then resumed to create a second offspring, specimen D (Figure 1.4). This step was identical to the first one, with the exception that now a sub-string in specimen B was transferred into A. The recombination operator acting on the whole population of specimens issued from the reproduction:

- Randomly split in two equal sets the population obtained at the end of the reproduction step.
- Took all pairs of specimens having the same rank in each part and simulated a coin toss weighted with the crossover probability p_c.
- Applied modified crossover if the coin showed “true.”

Modified Mutation Operator

Mutation prevents permanent loss of useful information and maintains diversity within the population. A specimen is altered by mutation with a low probability p_m. Classical mutation consists in changing the value of *one single bit* at a randomly chosen position in the string. As in the case of crossover, classical mutation is not m one-bit conservative. Nevertheless, to allow new features to be introduced in the specimens, a two-step mutation, namely mutation and *repair–mutation*, was defined to maintain the m-one bit structure of the strings. The repair–mutation merely reverses mutation by acting on another randomly chosen opposite bit value in the same specimen to restore the constant amount of ones in the string. For example, if mutation is $0 \rightarrow 1$, then anti-mutation is $1 \rightarrow 0$ on a different randomly chosen 1-bit value in the specimen (Figure 1.6). Below is a summary of operations used when applying modified mutation:

- A coin toss weighted with a mutation probability p_m for each bit is simulated for all the MAXPOP specimens of the population.

- If the coin shows "true", mutation and anti-mutation are applied; we then skip to the next specimen, authorizing just one operation per specimen.

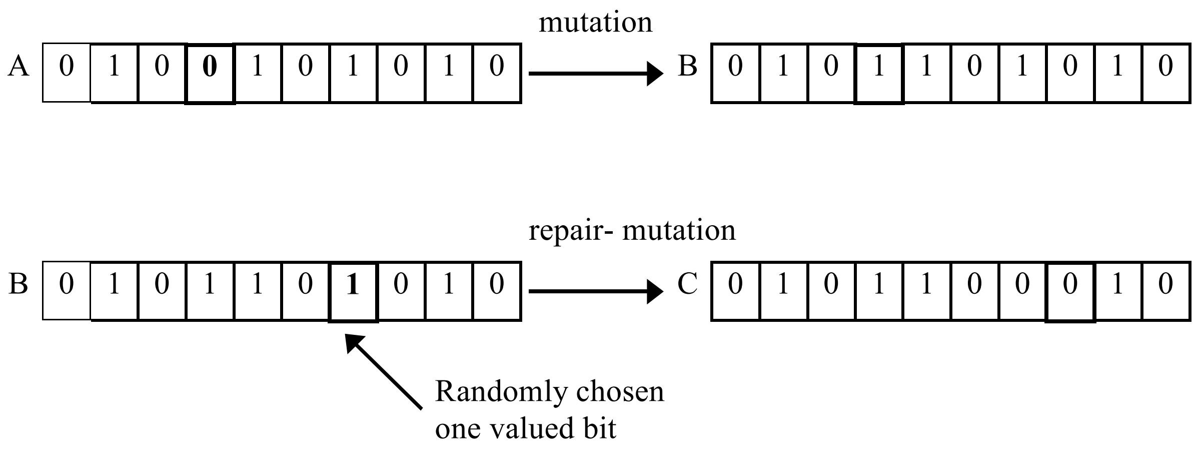


Figure 1.6 The m-conservative modified mutation: case of "0 → 1" mutation followed by repair–mutation

The mutation probability must be kept very low; excessive mutations could erase useful parts of the combinations, rendering the search directionless.

__**General remarks on the constrained GA**__

The parameters needed to run a GA are the population size, MAXPOP, and the crossover and the mutation probabilities, p_c and p_m. The choice of these parameters is important for the GA global efficiency. The parameter set MAXPOP=50, p_c=0.6, p_m=0.003 was used in this work; it was inspired by the general recommendations of De Jong (1976) and was adapted for the peculiarity of our *constrained* GA by trial and error procedures. As the number of genes per specimen is 27, there is a $27\cdot 0.003=8.1\%$ chance that an individual will undergo a mutation.

Regarding the issue of constraining the specimens to m non-null bit strings, one could have argued that penalty terms in the fitness function would prevent larger m-input combinations to dominate the population. However, from an efficiency standpoint, unconstrained GAs would have been less appropriate. One obvious reason is that searching among $\sum_{m_{min}}^{m_{max}} C_M^m$ solutions ($m_{min}>1$ and $m_{max}<M$) is far more efficient than searching among the whole $\sum_{m=1}^{M} C_M^m$ combinations. Moreover, inclusion of penalty terms in the objective function is desirable only if, within the *interrogation* space, the feasible regions are larger than the unfeasible ones (Lohl *et al.*, 1998). The fact that in our case $\sum_{m=1}^{M} C_M^m$ is superior to $\sum_{m_{min}}^{m_{max}} C_M^m$ means that unconstrained GAs would have spent most of the time evaluating unusefull solutions. Combinations exceedingly large or small are also unusable. Very low m values do not yield accurate ANN models, whereas if m is too large, the resulting ANN models are cumbersome. What is considered *low* and what is considered a *high* m value is problem-dependent; m has to be specifically tailored, as explained in Figure 1.1and Table 1.2 .

1.2.2 Methodology validation on liquid hold-up modeling

The proof-of-concept of the integrated GA-ANN methodology will be illustrated using a comprehensive database concerning the total liquid hold-up for counter-current gas-liquid flows in randomly packed towers. Recall that the goal behind this approach is to identify the liquid hold-up ANN model that best satisfies the three requirements summarized in Section 1.1.2. The *data mining* role of the genetic algorithm consists in interrogating a broad *reservoir* of M input vectors to enable the extraction of an elite of m inputs best mapping, through ANN, the (hold-up) output.

1.2.2.1 Brief overview of the liquid hold-up database

A large liquid hold-up database (1483 experimental points) set up in a recent study (Piché *et al.*, 2001e) was re-organized by converting all the physical properties and operating parameters relevant to the modeling of liquid hold-up into M = 27 dimensionless Buckingham Π groups (or candidate inputs) according to Table 1.1 format. These groups, listed below, cover all possible force ratios or external effects the liquid hold-up might experience in randomly packed beds:

liquid phase Reynolds (Re_L), Blake (Bl_L), Froude (Fr_L), Weber (We_L), Morton (Mo_L), Eotvos (Eo_L), modified Eotvos (Eo'_L), Galileo (Ga_L), modified Galileo (Ga'_L), Stokes (St_L), modified Stokes (St'_L), Capillary (Ca_L), and Ohnesorge (Oh_L).

gas phase Re_G, Bl_G, Fr_G, Ga_G, Ga'_G, St_G, and St'_G.

solid phase Wall factors K_1, K_2, and K_3, bed parameters B, and S_B.

two-phase Lockhart-Martinelli number (χ), Saberian number (Sa).

Details of group definitions are given in the Notation section of this chapter. For convenience, here are the forces ratios that the most popular groups represent: Reynolds ⇔ inertia-to-viscous; Froude ⇔ inertia-to-gravitational; Weber ⇔inertia-to-capillary; Morton ⇔ viscous-to-capillary, gravitational-to-capillary; Eotvos ⇔ gravitational-to-capillary; Galileo ⇔ gravitational-to-capillary, gravitational-to-viscous; Stokes⇔ inertia-to-gravitational, gravitational-to-viscous; Capillary ⇔ viscous-to-capillary; Ohnesorge ⇔ viscous-to-capillary.

The database has N = 1438 rows and M = 27 columns of candidate inputs. The best m-inputs selector, **S**, to be identified must contain a minimum number of elements, m. It has to be found among all possible combinations of M = 27 input columns. To demonstrate how computationally laborious this task can be, the combinatorial size for m = 5, *ca.* 81 000 combinations, would require 84 CPU days on a dual 800 MHz processor to identify the optimal ANN model using an enumerative technique.

1.2.2.2 Evaluation of the PPC term and choice of α, β, γ multipliers

To run the GA-ANN procedure, the penalty for phenomenological consistency appearing in the composite criterion Eq. (1.18) needs to be formulated. As mentioned earlier in Section1.2.1, this term must embed some prescribed behavioral rules, which verify the behavior of the ANN model. Such prescribed rules are inferred after tedious *expert-system* analyses that combine i) thorough inspection of the trends exhibited by the liquid hold-up in the database, ii) consentual observations from the literature, iii) any qualitative and quantitative information revealed from first-principle based phenomenological models in the field, such as the Billet and Schultes liquid hold-up models in the pre-loading and the loading regions (Billet *et al*., 1993; 1999). As a result, the following six monotonicity rules can be stated in the form of inequalities (for symbols, see the Notation section):

$$\frac{\partial \varepsilon_L}{\partial \rho_G} > 0 \qquad (1.20)$$

$$\frac{\partial \varepsilon_L}{\partial \rho_L} < 0 \qquad (1.21)$$

$$\frac{\partial \varepsilon_L}{\partial \sigma_L} > 0 \qquad (1.22)$$

$$\frac{\partial \varepsilon_L}{\partial u_L} > 0 \qquad (1.23)$$

$$\frac{\partial \varepsilon_L}{\partial \mu_L} > 0 \qquad (1.24)$$

$$\frac{\partial \varepsilon_L}{\partial u_G} > 0 \qquad (1.25)$$

The total liquid hold-up (the fraction of reactor volume occupied by the liquid phase) will always increase with the liquid throughput (u_L). The same kind of impact has the increase in the velocity of the gas stream (u_G) flowing upwards counter-current with the liquid, etc.

The match of the gradient information was verified at the corners of the $\hat{y}(u_G, v)$ surface, where $\hat{y}$ is the output of the trained neural network model, and v is one of the remaining variables (ρ_G, ρ_L, σ_L, u_L or μ_L). For the six physical variables ρ_G, ρ_L, σ_L, u_L, μ_L and u_G, the gradient conditions were considered fulfilled if they proved true simultaneously at the two points near the edges of the corresponding valid intervals. In order to better distinguish between models, the gradient conditions were equivalently recast into ten rules:

$$\left.\frac{\partial \hat{y}}{\partial \rho_G}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \rho_G}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_1 > 0 \tag{1.26}$$

$$\left.\frac{\partial \hat{y}}{\partial \rho_G}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \rho_G}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_2 > 0 \tag{1.27}$$

$$\left.\frac{\partial \hat{y}}{\partial \rho_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \rho_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_1 > 0 \tag{1.28}$$

$$\left.\frac{\partial \hat{y}}{\partial \rho_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \rho_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_2 > 0 \tag{1.29}$$

$$\left.\frac{\partial \hat{y}}{\partial \sigma_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \sigma_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_1 > 0 \tag{1.30}$$

$$\left.\frac{\partial \hat{y}}{\partial \sigma_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \sigma_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_2 > 0 \tag{1.31}$$

$$\left.\frac{\partial \hat{y}}{\partial u_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_1 > 0 \tag{1.32}$$

$$\left.\frac{\partial \hat{y}}{\partial u_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_2 > 0 \tag{1.33}$$

$$\left.\frac{\partial \hat{y}}{\partial \mu_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \mu_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_1 > 0 \tag{1.34}$$

$$\left.\frac{\partial \hat{y}}{\partial \mu_L}\right|_1 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial \mu_L}\right|_2 > 0 \quad \& \quad \left.\frac{\partial \hat{y}}{\partial u_G}\right|_2 > 0 \tag{1.35}$$

Each index indicates the points where the gradient was evaluated: 1 at the beginning and 2 at the end of the valid range of each physical variable, while "&" stands for the logical

AND. A scale from 0 to 10, measuring disagreement with the monotonicity tests, was then established to quantify the rules violated by an ANN model. The PPC term is expressed simply as the number of rules transgressed by an ANN having J hidden nodes and using input selector, **S**. If no rule was transgressed by the ANN model, $PPC_j[ANN(\mathbf{S})] = 0$, and the model had no penalty. Conversely, if the model violated all rules, the penalty was the maximum, i.e., equal 10.

The multipliers α and β, i.e., the weighting coefficients of the training and generalization AAREs in (1.18), were assigned the values 0.8 and 0.2, respectively. These values corresponded to the splitting of the initial database into training and generalization sets. Several values for the penalty coefficient γ were tested, and a value of 0.05 was then retained. Basically, as PPC lies in the interval [0,10] the value γ=0.05 gives to the PPC term an importance of 0.5 on a scale of 0 to 1.

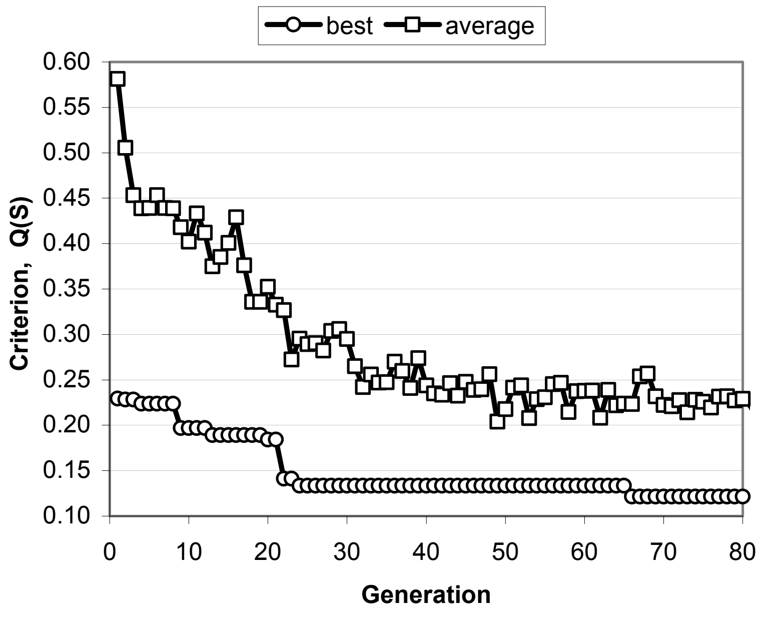


Figure 1.7 Best and population averaged criterion Q(S) in a typical GA run searching ANN models to predict liquid hold-up: for m = 5, J_{Min}=9, J_{max}= 13. (Computational system specification: dual CPU speed 1000 MHz, Operating system Linux, computation time 8 hrs.)

1.2.2.3 GA optimization through generations

As an example of the evolution of the performance of a population through successive generations, Figure 1.7 reports both the evolutions of the average and the minimum criteria of the population of individuals. The average criterion measures how well the population is doing, as well as how fast it is converging to the optimal solution. The minimum criterion indicates how well the GA has performed in finding a minimum-cost solution (Carroll, 1996). In Figure 1.7, the sharp decrease occurring at the 22nd generation is related to the first creation of a fully phenomenologically-sought ANN model (i.e., $PPC_j[ANN(\mathbf{S})] = 0$).

1.2.2.4 Results and discussion

The exposed methodology implies a systematic search with GA of ANN models for several values of m, i.e., number of nodes in the ANN input layer, with the objective of choosing a model with the least complexity, full phenomenological consistency, and the best accuracy. A search was conducted by launching GA runs for m = 4, 5, and 6. The evolution through generations of the best criterion is illustrated in Figure 1.8. The first occurrence of a full phenomenological consistency model occurred, for m=4 and 6, after 3 and 6 generations respectively. After 22 generations, the penalty term, PPC, became zero for the three cases. Then the criterion reduced to the averaged sum of ARRE on both training and generalization data sets. The ANN models presented lower criterion values with increasing m.

Since there was no significant improvement in the prediction performance between m=5 and m=6, we retained the less complex model, i.e., that for m=5. The best ANN model found with m = 5 involves J=12 hidden neurons and expressed as a function $\varepsilon_L = ANN(Bl_G, We_L, St'_L, K2, K3)$. It involves *significant* dimensionless numbers describing the liquid, gas, and solid phases, thus making the model appropriate for predicting liquid hold-up for different types of beds and fluids. The model AARE is 12.8% on the whole database; the standard deviation is 11.7 %. The model requires only 85 connectivity weights, and most importantly fulfills all imposed 10 rules given by Eqs. (1.26)-(1.35).

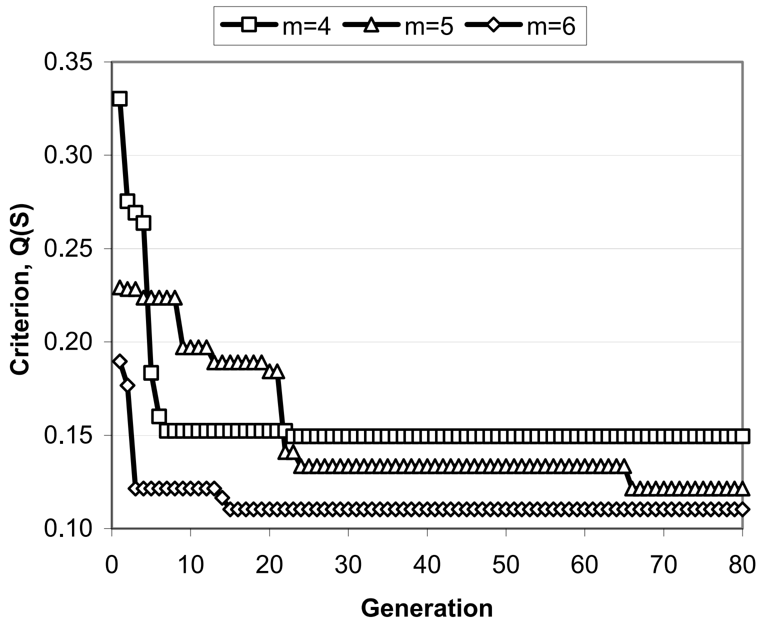


Figure 1.8 Evolution of best criterion for various numbers of ANN inputs, m.

The parity chart of the ANN model, shown in Figure 1.9, shows agreement between experimental and predicted liquid hold-up data, with also almost uniform data scatter distribution around the parity line. The performance of the model identified using the integrated GA-ANN procedure was slightly better than that reported by Piché *et al*. (2001e).

Due to the methodology design and the GA's population-based approach, the search process is inherently parallel. Implementation of the suggested integrated GA-ANN methodology on parallel processing computers renders the development of ANN models extremely fast, even using very large databases. This more efficient and automated procedure of dimensionless ANN identification allows, therefore, in the same time, feature selection (FS), as it optimizes the inputs of the network; model design (MD), as it finds a good architecture and learns the weights; and a check for prior knowledge respect (PK). Of course, in terms of model design, we use the classic well-known approaches such as trial and error for architecture and BFGS method for weights learning.

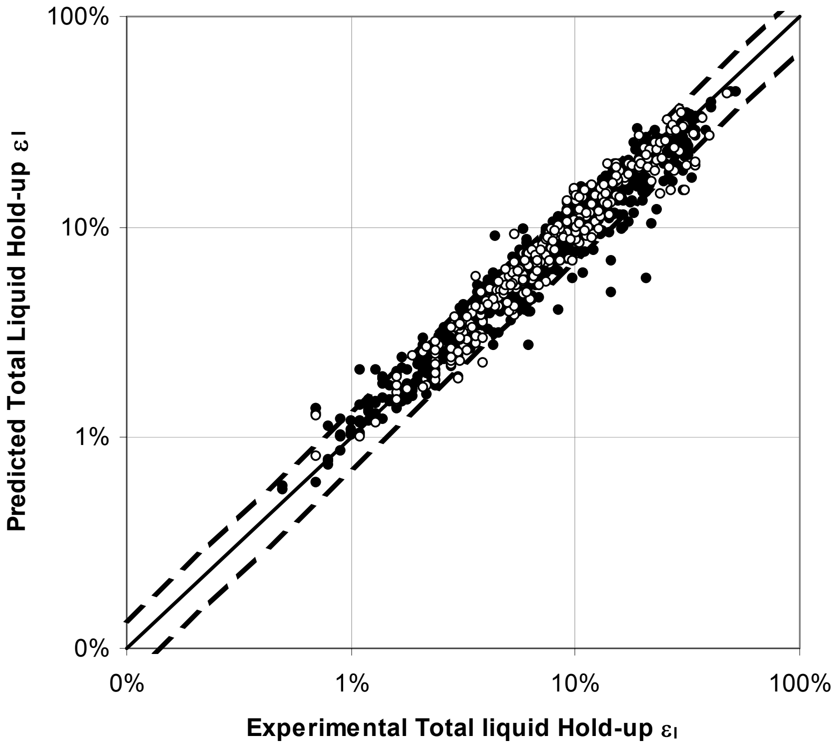


Figure 1.9 Parity chart of the ANN model ε_L = ANN(Bl_G, We_L, St'_L, K2, K3) for the learning (•) and the generalization (○) data sets. Dotted lines represent ± 30% envelopes.

Contribution here in terms of PK is the use of tests to verify the prior knowledge matching and to avoid overfitting. In terms of FS the originality also lay in the fact that the genetic algorithm uses modified genetic operators to keep constant the number of inputs per specimen. As the monotonicity property of the learned function was evaluated by numerical test at the edges of the definition domain for the involved dimensional variables, it would be reasonable to perform a study to determine representativeness of such tests of monotonicity likelihood for the neural network model.

1.2.3 Reinforcing the match of prior knowledge: Application to pressure drop modeling

In this section, the prior knowledge issue is more thoroughly addressed. Pitfalls in the standard procedure to guarantee adherence to PK of the ANN models are highlighted. A more robust procedure is developed and tested thoroughly to identify highly consistent ANN models. The material presented in this section will be organized as follows:

First, some basic phenomenological consistency requirements that an ANN model must fulfill are defined. For illustration purposes, the database of two-phase total pressure drops in counter-current packed beds (Piché *et al.*, 2001d) has been chosen. The deficiency of the standard procedure is evaluated on the pressure drop correlation developed therein. Secondly, an elaborate procedure assessing the PK match is proposed; a pseudo algorithm for its implementation is detailed in Appendix 1. Thirdly, the capability of the GA-ANN methodology described in (Tarca *et al.*, 2002) is upgraded by embedding the new PK match evaluation algorithm. This leads to high PK performance ANN models. Finally, a new ANN pressure drop correlation is presented and its performances discussed. Later in this chapter, a model that matches the prior knowledge will be alternatively called phenomenological consistent (PC).

1.2.3.1 Database and phenomenological consistency

The primary database used in this study is the one described in (Piché *et al.*, 2001d), containing N = 5005 pressure drop records in countercurrent randomly packed towers. The properties of the gas, liquid, and solid phases, and the measured pressure drops are compiled column-wise in the form of a 2-D matrix.

A set of 28 dimensionless groups was selected and computed for each row in the primary database. These dimensionless groups were those likely to contain the main variability within the database while expressing all the possible force ratios in play. The set of dimensionless numbers selected and presented in Table 1.3 is similar to the one listed in Section 1.2.2.1. For the sake of simplicity, they will be designated hereafter by N_i with $i = 1$

to 28. Afterwards, a working database was constructed in which appear, in a row-to-row correspondence, the calculated values of the dimensionless groups, together with the value of the pressure drop we wanted to model. Hence, the working database simply maps the primary database in a space with dimensionless variables. The working database was then split in two fractions, the first being referred to as the training set (N_T = 3503 records), and the second as the generalization set (N_G = 1502 records). However, with respect to the first study (Section 1.1) a larger fraction of data was used for generalization (30% instead of 20%).

The basic phenomenological consistency requirements of an ANN model that predicts the pressure drop were formulated after analyses that combined: (i) consentual observations from the literature and thorough inspection of the trends exhibited by the pressure drop in the primary database (Piché *et al.*, 2001d) and (ii) any qualitative or quantitative information revealed by the fundamental models (Billet *et al.*, 1999; Maćkowiak, 1991). The consentual PC requirements can be formulated as:

$$\frac{\partial(\Delta P/Z)}{\partial u_G} > 0 \tag{1.36}$$

$$\frac{\partial(\Delta P/Z)}{\partial u_L} > 0 \tag{1.37}$$

$$\frac{\partial(\Delta P/Z)}{\partial \rho_G} > 0 \tag{1.38}$$

$$\frac{\partial(\Delta \mathrm{P}/\mathrm{Z})}{\partial \mu_\mathrm{L}} > 0 \tag{1.39}$$

$$\frac{\partial(\Delta P/Z)}{\partial a_T} > 0 \tag{1.40}$$

Ideally, the simulated ANN pressure drop output is a monotonically increasing function of each one of the five testing dimensional variables. Practically, due to the problem of overfitting and local poor quality of some measured data, a model could fail to fulfill Eqs. (1.36)-(1.40) gradient conditions simultaneously in the vicinity of some data points.

Charts illustrating that the gradient conditions (or PC requirements) can be satisfied at least over some sub-domains in the database usually accompany several ANN correlations published in the field (Piché *et al*., 2001d; Larachi *et al*., 1999). Typically, the search utilized was conducted as follows: when a gradient condition is tested with respect to a particular dimensional variable, all the other variables are assigned values corresponding to a particular point in the primary database.

If all the gradient conditions are satisfied, the model is considered phenomenologically consistent around the test point. Such an analysis, in the absence of an automated procedure, is cumbersome and time-consuming. It must be repeated several times in order to find the ANN models that best obey the gradient conditions in the domain and fit the data points.

The question arising with this method is: will the ANN model exhibit the same type of trend when moving to a different test point?

A search was been carried out using the GA-ANN methodology described in the previous sections, and three distinct ANN models, M1, M2, M3, were identified. All verified the gradient conditions Eqs. (1.36)-(1.40) in the vicinity of the test point I (Table 1.4) belonging to the database. These models differed mainly by the dimensionless groups used as ANN inputs. They were trained on the same training database and exhibited good prediction performances, as shown in Table 1.5.

Table 1.3 Candidate dimensionless input variables for pressure drop modeling

i	1	2	3	4	5	6	7
Dimensionless Number N_i	Reynolds (Re_G)	Blake (Bl_G)	Froude (Fr_G)	Galileo (Ga_G)	Mod. Galileo (Ga_G^m)	Stokes (St_G)	Mod. Stokes (St_G^m)

8	9	10	11	12	13	14
Reynolds (Re_L)	Blake (Bl_L)	Froude (Fr_L)	Weber (We_L)	Morton (Mo_L)	Eotvos (Eo_L)	Mod. Eotvos (Eo_L^m)

15	16	17	18	19	20	21
Galileo (Ga_L)	Mod. Galileo (Ga_L^m)	Stokes (St_L)	Mod. Stokes (St_L^m)	Capillary (Ca_L)	Ohnesorge (Oh_L)	Wall factor (K1)

22	23	24	25	26	27	28
Wall factor (K2)	Wall factor (K3)	Correction number (S_B)	Correction number (S_B) (2)	Correction number (S_B) (3)	Lockart-Mart. (χ)	Saberian number (Sa)

Table 1.4 Ranges of dimensional variables and data points I and II

	u_G (m/s)	u_L (m/s)	ρ_G (kg/m^3)	μ_L (kg/m.s)	a_T (m^2/m^3)	D_C (m)	ρ_L (kg/m^3)	σ_L (N/m)	ϕ (-)	Z (m)	μ_G (kg/m.s)	ε (-)
Max	4.5E+0	9.0E-2	4.2E+1	4.5E-2	7.0E+2	9.1E-1	1.3E+3	7.4E-2	6.0E-1	3.0E+0	1.9E-5	9.9E-1
Average	1.1E+0	1.3E-2	2.1E+0	5.8E-3	2.2E+2	5.0E-1	1.0E+3	6.5E-2	3.1E-1	1.6E+0	1.8E-5	7.9E-1
Min	5.0E-2	4.9E-4	9.2E-1	7.8E-4	5.7E+1	5.1E-2	8.1E+2	2.6E-2	5.6E-2	3.0E-1	1.4E-5	5.5E-1
Point I	1.0E+0	2.0E-2	1.2E+0	1.0E-3	1.7E+2	2.0E-1	1.0E+3	3.5E-2	9.1E-2	1.0E+0	1.8E-5	9.7E-1
Point II	5.6E-1	1.8E-2	1.2E+0	9.4E-3	1.8E+2	6.1E-1	1.2E+3	7.2E-2	4.3E-1	2.4E+0	1.8E-5	6.9E-1

Table 1.5 Three ANN models with low AARE and phenomenological consistency in the vicinity of point I

ANN model	**Dimensionless numbers used as inputs(*)**	**$AARE_T$[%]**	**$AARE_G$[%]**	**$AARE_{T+G}$[%]**
M1	$N_4, N_8, N_{15}, N_{23}, N_{24}, N_{28}$	19.9	20.5	20.1
M2	$N_2, N_{13}, N_{18}, N_{25}, N_{27}, N_{28}$	21.2	21.6	21.3
M3	$N_4, N_8, N_{15}, N_{23}, N_{24}, N_{27}$	20.4	21.1	20.6
Piché *et al.* 2001d	$N_1, N_4, N_8, N_{15}, N_{17}, N_{24}, N_{27}$	19.6	21.1	20.0

* The significance of the input N_i is the same as in Table 1.3

Table 1.6 Three ANN models with low AARE and phenomenological consistency in vicinity of most training points

ANN model	**Dimensionless numbers used as inputs(*)**	**$AARE_T$[%]**	**$AARE_G$[%]**	**$AARE_{T+G}$[%]**
CP1	$N_{13}, N_{14}, N_{19}, N_{20}, N_{24}, N_{27}$	20.9	21.8	21.2
CP2	$N_{11}, N_{13}, N_{14}, N_{19}, N_{24}, N_{27}$	22.3	23.3	22.6
CP3	$N_2, N_{10}, N_{17}, N_{18}, N_{24}, N_{27}$	22.4	24.2	22.9

* The significance of the input N_i is the same as in Table 1.3

Table 1.7 The PCE values for the two series of ANNs found by using the classic and the new requirements

	Models found with the classic requirements (low AARE and phenomenological consistency around a customary point)				**Models found with the new requirements** (low AARE and phenomenological consistency around the majority of points)		
Model	M1	M2	M3	SP	CP1	CP2	CP3
PCE_G[%]	53.5	66.4	54.4	81.1	22.4	14.2	22.2
PCE_T[%]	53.6	66.8	52.8	80.5	23.4	14.5	21.9
PCE_{T+G}[%]	53.5	66.7	53.3	80.7	23.1	14.4	22.0

For comparison, the Piché *et al*. (2001d) correlation that fulfills all five conditions at point I, was also tested. Figure 1.10 shows the simulated effect of liquid viscosity on the pressure drop predicted by models M1, M2, and M3 and by the model of Piché *et al*., (2001d) (labeled as SP) in the vicinity of point I when the liquid viscosity varied around its initial value. Though the predictions given by the four models were very distinct in some regions, all the models showed the expected increasing trend of pressure drop when liquid viscosity increased.

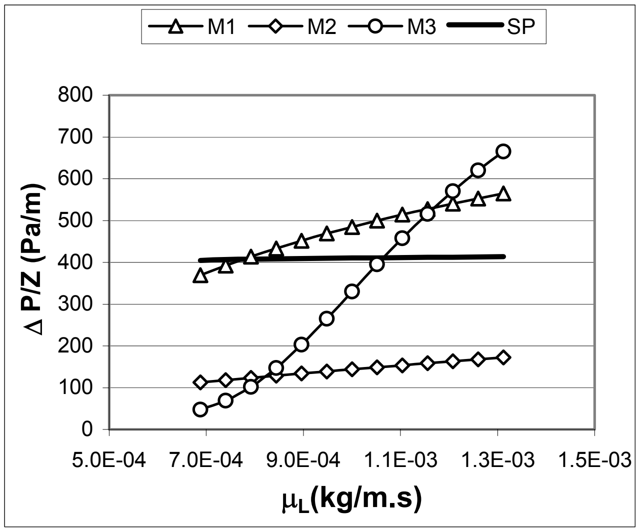


Figure 1.10 Phenomenological behavior of four ANN models around point I

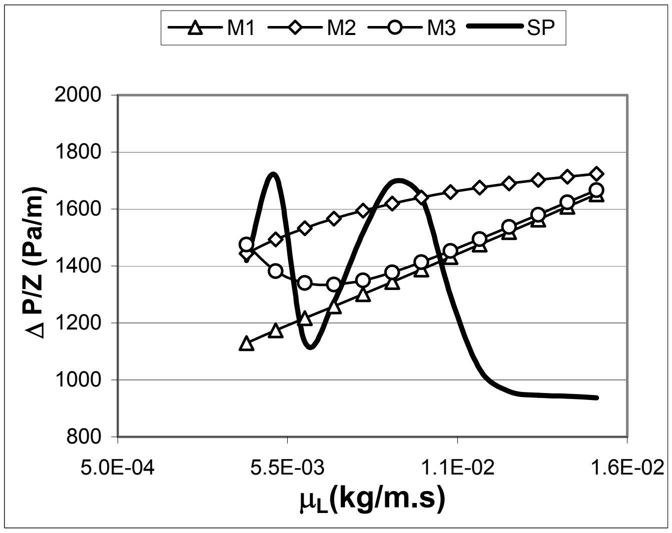


Figure 1.11 Phenomenological behavior of four ANN models around point II

A viscosity-pressure drop plot similar to that in Figure 1.10, is drawn to illustrate the behavior of the same models around point II (Figure 1.11). Two of them must be disqualified (M3 and SP) for misbehaving in terms of viscosity impact on pressure drop around point II because rule Eq. (1.39) is violated.

As the models were all phenomenologically consistent around point I, and not distinguishable by their AARE values on the training and generalization data sets (Table 1.5), any one of them could have been chosen as a predictor. However, the predicted pressure drop values would be very different from one model to another. Near point II, M3 and SP models presented abnormal gradient changes. Two main reasons could explain such a behavior: a) in the vicinity of point II, the database did not reveal the increasing pattern of pressure drop with viscosity b) overfitting of the data points occurred in that region.

Overfitting occurs when a network too closely approaches some training points and has not learned to generalize new inputs. It produces a relatively small error on the training set, but gives a much larger error when new instances are presented to the network. E*arly stopping* and *regularization* techniques are used to prevent overfitting. Early stopping uses two different data sets. The training set is used to update the weights, and the validation set is used to stop training when the network begins to overfit data. Regularization modifies the network's performance function, the measure of error that the training process minimizes. By changing it to include the size of the weights, training produces a network that not only performs well with the training data, but behaves predictably when exposed to new instances. For details and examples of these anti-overfitting techniques, consult Tetko (1997), Prechelt (1998), and Gencay (2001).

Models M1 – M3 and SP were built using early stopping. Table 1.5 shows that the difference in errors between learning and training data sets were marginal, suggesting that the training process was adequately stopped. The ANN models M3 and SP suggested the contrary, as they overfit the data (Figure 1.11). Clearly, the early stopping technique alone was not sufficient to prevent overfitting and violation of model phenomenological consistency.

1.2.3.2 New method for assessing phenomenological consistency of ANN models

If ANN models are phenomenologically consistent near some points in the database, they are not necessarily consistent over the whole database space. We therefore checked the fulfillment of Eqs. (1.36)-(1.40) in the vicinity of every point available for training and computed the percentage of data points for which not all gradient conditions are met. We then defined the phenomenological consistency error (PCE) as a statistical indicator for measuring the overall disagreement yielded by an ANN model.

Before explaining how PCE is quantified, recall that the ANN models are trained with some of the dimensionless groups N_i ($i = 1\ldots28$) taken from the working database and not directly with the physical properties of the primary database. Obviously, each dimensionless number spans a range that is bounded by some extreme values, i.e., $N_{i,min} \leq N_i \leq N_{i,max}$. The models are thus valid only when the dimensionless numbers evolve within these ranges.

Suppose now a trained ANN model uses as inputs the dimensionless numbers N_1 to N_m to predict the pressure drop. Consider then a training data point, $\mathbf{p_k}$, in the space of the primary database that has the form $\mathbf{p_k} = \{u_G, u_L, \rho_G, \mu_L, a_T, \varepsilon, \phi, Z, D_C, \rho_L, \sigma_L, \mu_G\}$ and for which the experimental value of the pressure drop $y^{(exp)}(\mathbf{p_k})$ is known. Consider a testing dimensional variable, v_j, from a list of five $\{u_G, u_L, \rho_G, \mu_L, a_T\}$ variables, which is nothing but the subset of dimensional variables used to coerce the ANN outputs via Eqs. (1.36)-(1.40). A maximum increment Δ is determined such that when added to, or subtracted from, the initial value of the variable v_j, will remain between N_1 to N_m, which are recalculated with the new values of the variable v_j. (Alternatively, the increment Δ can also take a small value equivalent to a constant percentage of the initial value of the variable v_j). Let us denote by $\mathbf{p_{k,j}}^{+\Delta}$ and $\mathbf{p_{k,j}}^{-\Delta}$ the points that result from respectively adding and subtracting an increment Δ from variable v_j in vector $\mathbf{p_k}$. (For example, if $j = 4$, $\mathbf{p_{k,4}}^{+\Delta} = \{u_G, u_L, \rho_G, \mu_L+\Delta, a_T, \varepsilon, \phi, Z, D_C, \rho_L, \sigma_L, \mu_G\}$). Accordingly, the outputs from the ANN model can be computed forwards, central, and backwards as $y^{(calc)}(\mathbf{p_{k,j}}^{+\Delta})$, $y^{(calc)}(\mathbf{p_k})$, $y^{(calc)}(\mathbf{p_{k,j}}^{-\Delta})$.

Provided the following order holds

$$\mathrm{y}^{(\mathrm{calc})}(\mathbf{p}_{\mathbf{k},j}^{-\Delta}) \le \mathrm{y}^{(\mathrm{calc})}(\mathbf{p}_{\mathbf{k},j}) \le \mathrm{y}^{(\mathrm{calc})}(\mathbf{p}_{\mathbf{k},j}^{+\Delta}) \qquad (1.41a)$$

the gradient (calculated forward and backward) of the ANN output is positive with respect to the variable j when evaluated in the vicinity of point $\mathbf{p_k}$. If Eq. (1.41a) is satisfied by all five testing variables $\{u_G, u_L, \rho_G, \mu_L, a_T\}$, the ANN model is phenomenologically consistent near $\mathbf{p_k}$. The above procedure is repeated for all the data points available for training (k = 1…N_T), and PCE is computed as the percentage of the data points around which the ANN model fails the phenomenological consistency test. The pseudo-algorithm for PCE evaluation is detailed in Appendix 1. A mathematical formula for PCE is

$$PCE = 1 - \frac{1}{N_T}\sum_{i=1}^{N_T} I\left(\mathrm{I}_{v_j} \left(\left.\frac{\partial y}{\partial v_j}\right|_{p_i} \ge 0 \right)\right) \qquad (1.41b)$$

in which I is the identity function (taking the value 1 if the condition passed as argument is true) and I stands for logical intersection. Ideally, a model that fulfills the gradient conditions Eqs. (1.36)-(1.40) near all the data points in the database would give PCE = 0%. Such a value is unlikely to occur, due to the inherent overfitting problem and/or the local poor quality of experimental data.

1.2.3.3 Finding ANNs with low PCE value to model pressure drop

In this section we shall present how we can obtain ANN models with low PCE values and remarkable accuracy. There are two methods that yield ANN models that do more than fitting the data points.

The first consists in modifying the ANN training procedure in such a way that the model learns to fit the data and simultaneously satisfy the phenomenological constraints. For example, the supplementary information about the function to be learned, also referred to as *hints*, can be added through *new* data points that contain that information (Abu-Mostafa, 1993; Sill and Abu-Mostafa, 1997). A second method adopted in this work proposes not to alter the training procedure or the data. Instead, the network's architecture, capable of retaining the supplementary information about the function to be learned, is searched. The supplementary information an ANN has to learn for predicting pressure drop ensures small

PCE values. By means of GA, the best input selector **S** (combination of dimensionless numbers as ANN inputs among the 28 candidates N_i) and the appropriate number of nodes in the hidden layer are determined (see section 1.1.2-1.1.4).

The best input selector **S** and the corresponding ANN model minimizes the prediction error and PCE. The following composite criterion, Q, is formulated:

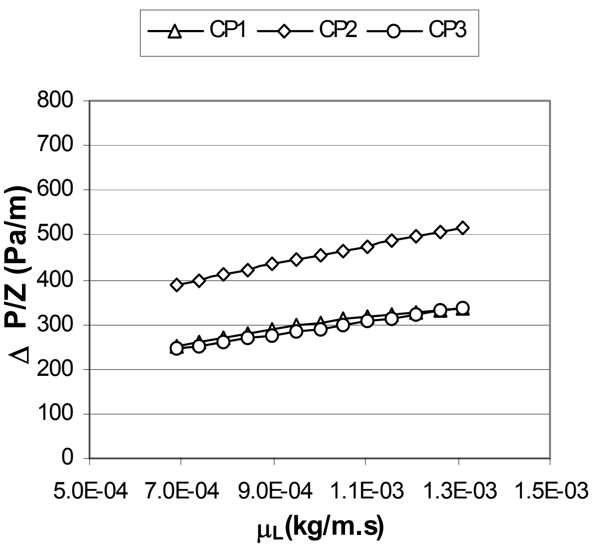
$$Q(\mathbf{S}) = \min_{J_{Min} \le J \le J_{Max}} Q_J(\mathbf{S}) \tag{1.42}$$

With

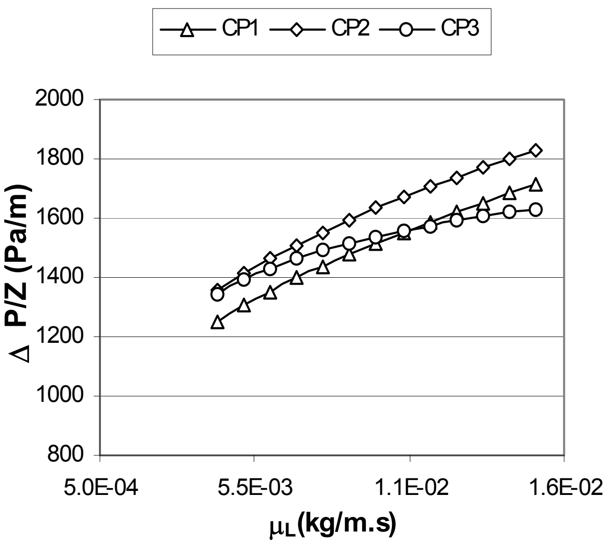
$$\mathrm{Q}_{\mathrm{J}}(\mathbf{S}) = AARE[ANN_J(\mathbf{S})]_{\mathrm{T}} + \alpha \cdot PCE[ANN_J(\mathbf{S})] \tag{1.43}$$

In Eq. (1.43) AARE[ANN$_J$(**S**)] is the average absolute relative error the ANN (having J hidden nodes) achieves on the *training* data set for a given input combination **S**. Inclusion in the composite criterion of a *penalty* for *phenomenological consistency*, PCE[ANN$_J$(**S**)], ideally guarantees that the model is not likely to display unexpected behavior. The multiplier α in Eq. (1.43) was set to 0.25 by trial and error targeting to obtain models whose AARE[ANN$_J$(**S**)] and PCE[ANN$_J$(**S**)] were similar. This may be justified by the fact that we give about the same importance to the training data points as we gave to prior knowledge matching. However, as the PCE values for different combinations **S** were in general higher than AARE values, a sub-unitary multiplier had to be assigned. To reduce the computation time, the criterion Q did not include AARE[ANN$_J$(**S**)]$_G$ on the generalization data set, as implemented in § 1.2-1.4, nor the phenomenological consistency error PCE[ANN$_J$(**S**)]$_G$.

Let the GA-ANN methodology search for the three best ANN models having six entries, as in the M1-M3 models presented in Table 1.5. These models, labeled CP1, CP2, CP3 (Table 1.6), show phenomenological consistency around the majority of data points in the training set, i.e., low PCE (<25%) and low AARE (~21%). These three models were tested around data points I and II (Figure 1.12).



a)



b)

Figure 1.12 Phenomenological behavior of CP1 to CP3 ANN models around point a) I and b) II

All three ANN models exhibited a monotonically increasing trend around points I and II (Figure 1.12). There was also a closer consensus in prediction by the three models, compared to the M1, M2, M3 and SP models (Figure 1.10 and Figure 1.11).

We have just exemplified the behavior of the models with respect to one test variable (μ_L) and around two custom points I and II. How well all these models behave with respect to all the testing variables (u_G, u_L, ρ_G, μ_L, a_T) simultaneously and around all 5005 points of the database is given by the PCE values in Table 1.7. The best model yielded CPE = 53% (M1 or M3) when the constraint was applied at the peculiar point I (models M1-M3 and SP). This means that among the 5005 data in the primary database, more than 2652 violated at least one of the gradient conditions Eqs. (1.36)-(1.40). Constraining systematically all the points in the database drastically reduced the PCE values; the best was CP2 model with PCE = 14%. Note that to allow worthy comparisons between models, all of them had 14 to 15 hidden nodes, the same number of entries (six), and identical training epochs. CP1-CP3 models outperformed M1-M3 and SP models because the data representation of the former models was less sensitive to noise than the latter.

1.2.3.4 An improved correlation for pressure drop prediction

The SP model (Piché *et al.*, 2001d) discussed above is a seven-entry model. We decided to search a better seven-entry ANN model using the new GA-ANN methodology and to compare its performance to that of the SP model. The best model (named CP4) found in the last generation explored by the GA was:

$$\frac{\Delta P / Z}{\rho_L g} = f\left(Bl_L, Fr_L, Eo_L, Eo_L^{'}, K_1, S_B, \chi\right) \tag{1.44}$$

Table 1.8 Compared performances of ANN models SP and CP

Statistics	**Piché *et al.* (2001d) (SP)**	**This work (CP4)**
$AARE_G$ [%]	21.1	19.9
$AARE_T$ [%]	19.6	19.2
$AARE_{T+G}$ [%]	20.0	19.4
σ_G [%]	20.7	18.8
σ_T [%]	19.3	19.4
σ_{T+G} [%]	19.8	19.2
PCE_G [%]	81.1	17.3
PCE_T [%]	80.5	16.7
PCE_{T+G} [%]	80.7	16.9
No. Weights	109	127

The comparative performances of CP4 and SP models are detailed in Table 1.8. CP4 model restored the expected gradient conditions on the simulated output with a success of 83% for the testing variables u_G, u_L, ρ_G, μ_L or a_T. Its PCE was four times lower than the SP model's. The equations for the CP4 model are given in Table 1.9. A parity chart showing CP4 model predictions versus measured pressure drops is depicted in Figure 1.13. A uniform distribution of data around the parity line is present.

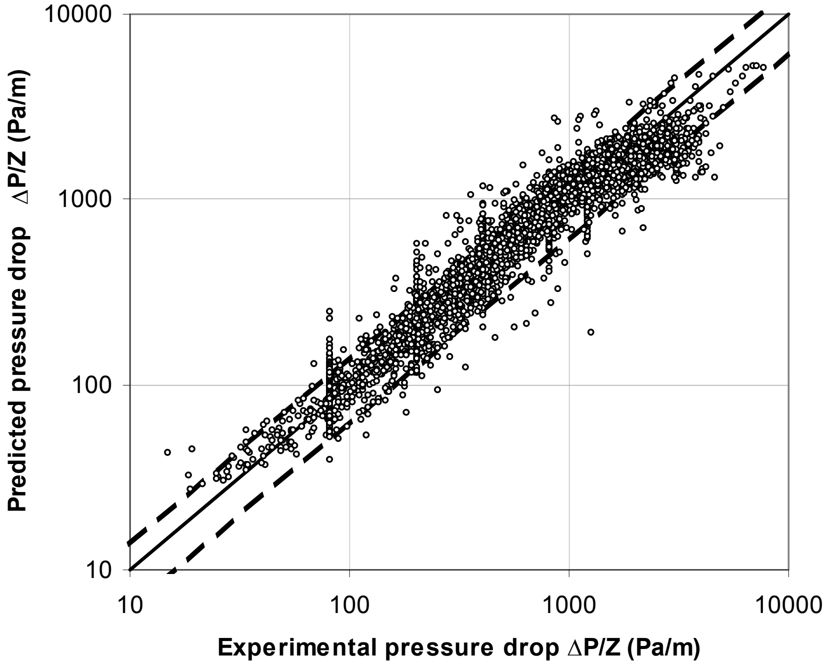


Figure 1.13 Parity chart of CP4 ANN model. The dotted lines represent the $\pm 2\sigma$ envelopes

1.2.3.5 Discussion

In Section 1.2.3 we showed that the simple tests of monotonicity of the ANN output with respect to some dimensional variables, performed at the extremities of their definition ranges, cannot guarantee representative monotonic behavior in the entire feature space. The PCE (phenomenological consistency error) is a better measure, as it evaluates the monotonicity in all the data points available for training. However, a 0% PCE was not attained; therefore, the next section discusses lowering the PCE and prediction error by combining several different ANN models.

Table 1.9 ANN normalized input and output functions and the corresponding weights (Ranges of applicability in brackets)*

$$S=\frac{1}{1+\exp\left(-\sum_{j=1}^{15}\omega_j H_j\right)}$$

$$H_j=\frac{1}{1+\exp\left(-\sum_{i=1}^{8}\omega_{ij}U_i\right)}$$

$$1\le j\le 14 \qquad H_{15}=1$$

$$S=\frac{\log\left(\dfrac{f_{LGG}}{1.52\times10^{-3}}\right)}{2.66}$$

$$f_{LGG}=\frac{\Delta P}{Z}\frac{1}{\rho_L g}$$

$$\begin{bmatrix} f_{LGG}\ge 1.52\times10^{-3} \\ f_{LGG}\le 6.94\times10^{-1}\end{bmatrix}$$

$$U_1=\frac{\log\left(\dfrac{Bl_L}{2.637\times10^{-2}}\right)}{4.627} \quad U_2=\frac{\log\left(\dfrac{Fr_L}{3.136\times10^{-6}}\right)}{5.353} \quad U_3=\frac{\log\left(\dfrac{Eo_L}{7.442\times10^{-2}}\right)}{2.801} \quad U_4=\frac{\log\left(\dfrac{Eo'_L}{2.735\times10^{-1}}\right)}{2.180} \quad U_5=\frac{\log\left(\dfrac{K_1}{3.928\times10^{-1}}\right)}{0.387}$$

$$U_6=\frac{\log\left(\dfrac{S_B}{7.976\times10^{0}}\right)}{2.27153} \quad U_7=\frac{\log\left(\dfrac{\chi}{2.362\times10^{-2}}\right)}{3.8768} \quad U_8=1$$

$$Bl_L=\frac{\rho_L U_L}{a_T\mu_L} \quad Fr_L=\frac{U_L^2}{g\cdot d_P} \quad Eo_L=\frac{\rho_L\cdot g\cdot d_P^2}{\sigma_L} \quad Eo'_L=\frac{\rho_L\cdot g}{\sigma_L\cdot a_T^2} \quad K_1=\left(1+\frac{2}{3(1-\varepsilon)}\left(\frac{d_{PV}}{D_C}\right)\right)^{-1} \quad S_B=\frac{a_S d_h}{(1-\varepsilon)}$$

$$\chi=\frac{U_G}{U_L}\sqrt{\frac{\rho_G}{\rho_L}}$$

$$\begin{bmatrix} Fr_L\ge 3.14\times10^{-6} \\ Fr_L\le 7.07\times10^{-1}\end{bmatrix} \begin{bmatrix} Eo_L\ge 7.44\times10^{-2} \\ Eo_L\le 4.71\times10^{1}\end{bmatrix} \begin{bmatrix} Eo'_L\ge 2.74\times10^{-1} \\ Eo'_L\le 4.14\times10^{1}\end{bmatrix}$$

$$\begin{bmatrix} K_1\ge 3.93\times10^{-1} \\ K_1\le 9.58\times10^{-1}\end{bmatrix} \begin{bmatrix} S_B\ge 7.98\times10^{0} \\ S_B\le 1.49\times10^{3}\end{bmatrix} \begin{bmatrix} \chi\ge 2.36\times10^{-2} \\ \chi\le 1.78\times10^{2}\end{bmatrix}$$

ω_{ij}	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	-15.625	-20.166	5.199	1.223	-0.390	-0.324	-24.203	-2.801	8.174	6.196	-13.786	0.020	1.323	-11.417
2	7.758	8.868	-0.208	-5.096	4.687	0.880	15.430	9.247	-5.896	-9.190	8.455	-6.875	6.026	10.259
3	5.833	23.015	-11.210	-1.560	21.665	-27.436	4.137	0.035	44.384	11.244	-15.656	19.684	-6.908	20.613
4	1.001	-2.220	6.854	-0.289	-15.279	28.561	-86.487	0.266	-51.511	4.074	-24.571	-17.069	4.074	-1.256
5	-27.213	16.980	12.332	-1.787	-30.549	15.451	-26.909	-1.132	-27.135	21.730	-7.706	5.382	-8.830	21.087
6	0.526	5.805	-0.622	-0.491	-100.040	-25.615	-11.960	-3.824	40.254	22.886	-29.674	-11.934	6.704	27.156
7	1.307	-1.929	5.582	-6.182	1.367	0.155	1.546	-0.087	0.899	-5.678	0.495	10.682	6.984	4.519
8	21.160	1.602	-5.781	9.120	39.939	-3.092	45.910	-3.144	6.234	-23.785	26.847	12.533	-16.151	-40.058

ω_j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
	-0.846	4.307	11.544	-14.378	0.906	-5.570	-5.292	6.172	-3.426	-2.806	5.611	-0.246	2.652	-1.779	2.447

*A "user-friendly" spreadsheet of the neural correlation is accessible at: http://www.gch.ulaval.ca/~grandjean or http://www.gch.ulaval.ca/~flarachi

1.3 ANN meta-models to enhance prediction and phenomenological consistency

1.3.1 Introduction to ANN combination schemes

In this section we study the possibility of combining several *good* ANNs in order to achieve better predictions. Although all the models are *individually* equally good *on average*, they are not on each individual point from the database. Some ANNs may be locally good, while others, because of different inputs' sets and architectures, may not be. Hence, combining ANNs could create a synergistic effect, especially in the database regions where the contrast in performance between individual ANNs is great.

This approach has been investigated in several research works (Alpaydin, 1993; Hashem *et al*., 1997; Benediktsson *et al*., 1993; Alpaydin, 1998; Ueda, 2000), and consists in feeding the predictions of several distinct networks, referred to here as *base-models* (level 0), into an upper level-model (level 1), referred to as *meta-model*, which is generally linear. Meta-model are more robust than the individual base-models because base models might be specialized on different regions in the input space. Hopefully, they will not all be wrong at the same point in the database space. The base models might be different by the data representation; i.e., the learner might use different representations of the same inputs, by the training scheme, the initial weight set, etc. Comprehensive classification of the possible differences between the base-models, the functions, and the combinatorial strategies of their output into a meta-model were discussed by Alpaydin (1998). An illustration of ANNs combination in a pulp and paper industry application is presented in Lanouette *et al*. (1999). In their work, the base ANN models differed mainly in the data samples with which they were trained.

Hashem (1997) investigated combining a number of trained networks by performing weighted sums of the outputs of the base (component) networks. An unconstrained MSE-OLC (mean squared error-optimal linear combination) of networks with constant term, which theoretically yields the smallest MSE, was proposed. Independently, Perrone (1993) developed the general ensemble method (GEM) for constructing improved regression estimates. The GEM is equivalent to the constrained MSE-OLC (Hashem, 1997). In these works, the weights in the meta-model were determined by minimizing the MSE of the meta-model on the same data on which the base-models were trained. This is simple to do, but if base-models are highly cross-correlated, i.e., base-models are all weak in the same regions of the input space, the meta-model will lack robustness (Breiman, 1992). Breiman (1992) extended the Wolpert (1992) approach to stacking regressions by estimating the meta-model regression coefficients (level 1 model) based on the performance of the base-models (level 0 model) on generalization data. The major drawback of Breiman's method is its computational heaviness, since the base-models need to be retrained on the cross-validation data (1992). In addition, the common feature to these studies on networks combination was their focus on improving the prediction accuracy without concerning with monotonicity constraints that the meta-model may need to match.

1.3.2 Base-models and meta-model

Consider again the pressure drop prediction problem in counter-current packed bed reactors in which the characteristic to approximate, y, is the dimensionless pressure drop, $\Delta P/\rho_L gZ$. Let us also assume that y is contributed by a deterministic function g, such that $y(\mathbf{p}) = g(\mathbf{p}) + \varepsilon$, where ε is a normally distributed, zero-mean, random variable. Estimators of the function g could be neural networks, e.g., multilayer perceptron, radial basis functions, trained with pairs $(p_k, y(p_k))$. The inputs of such neural networks could be dimensionless Buckingham Π groups "edited" from some of the dimensional variables contained in vectors $\mathbf{p}_k = \{u_G, u_L, \rho_G, \mu_L, a_T, \varepsilon, \phi, Z, D_C, \rho_L, \sigma_L, \mu_G\}$. The reservoir of dimensionless numbers considered is given in Table 1.3. Using dimensionless numbers enlarges the applicability ranges for the model but also creates ambiguity and uncertainty as to their selection. Depending on the pertinence of these inputs, the resulting networks may or may not be accurate and phenomenologically consistent, i.e., will not exhibit low AARE and PCE values. The GA-ANN methodology described in section 1.2.2 and reinforced in 1.2.3 enabled identification of several networks exhibiting low AARE and PCE. Instead of retaining the *best* among them and discarding the others, as we did in 1.2.3, we retained the three best networks (referred to as bm_r, r = 1,3) and built a meta-model.

These base-models differed by the quantity and type of dimensionless numbers they used as their inputs, as well as by the number of hidden nodes (Table 1.10). All the other related training parameters were the same: bm1-bm3 were trained on 70% of the available data ($N_T = 3503$), and the remaining 30% ($N_G = 1502$) were used to evaluate their generalization capabilities, as is standard in ANN modeling (Flexer, 1994). The following statistics were computed:

i) the average absolute relative error (AARE)

$$AARE = \frac{1}{N}\sum_{k=1}^{N}\left|\frac{y(\mathbf{p}_k) - y^{calc}(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right| \tag{1.45}$$

with $y(\mathbf{p}_k)$ the experimental value of y, and $y^{calc}(\mathbf{p}_k)$ the predicted value of y for the data point p_k.

ii) the standard deviation of the absolute relative error (STDEV)

$$STDEV = \sqrt{\sum_{i=1}^{N}\left[\left|\frac{y(\mathbf{p}_k) - y^{calc}(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right| - AARE\right]^2 \Big/ (N-1)} \tag{1.46}$$

iii) the maximum absolute relative error (MAXARE)

$$MAXARE = \max_k\left|\frac{y(\mathbf{p}_k) - y^{calc}(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right| \tag{1.47}$$

iv) the mean square error

$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left(y(\mathbf{p}_k) - y^{calc}(\mathbf{p}_k)\right)^2 \tag{1.48}$$

v) the phenomenological consistence error (PCE), computed as the percentage of data points $\mathbf{p_k}$ in whose vicinity at least one of Eqs. (1.36)-(1.40) monotonicity constraints was violated.

Table 1.10 The base models used to build the meta-model

Model	Inputs(**)	No. Inputs	No. Hidden Nodes	AARE [%]	STDEV [%]	MAXARE [%]	MSE*	PCE [%]
bm_1	N_{10}, N_{13}, N_{14}, N_{18}, N_{23}, N_{26}, N_{27}	7	15	19.49	18.02	164	1.22e-2	21.6
bm_2	N_9, N_{10}, N_{13}, N_{14}, N_{21}, N_{24}, N_{27}	7	14	19.95	18.77	195	1.29e-2	17.3
bm_3	N_{10}, N_{14}, N_{17}, N_{18}, N_{24}, N_{27}	6	14	21.84	20.35	194	1.56e-3	20.2

(*) MSE was computed on the log values of calculated and experimental *y*; (**) The significance of the input N-i is the same as in Table 1.3

Table 1.11 The values of the weighting coefficients in the meta-model

Model	β_1	β_2	β_3
AARE+STDEV – optimal meta-model	0.367	0.380	0.279
MSE – optimal meta-model	0.392	0.404	0.204

Table 1.12 The performances of the meta-model compared with the best and simple average models

Model	AARE [%]	STDEV [%]	MAXARE [%]	MSE*	PCE [%]
AARE+STDEV – optimal meta-model	17.28	15.00	133	1.17e-2	7.9
MSE – optimal meta-model	17.97	16.51	148	1.08e-2	9.1
Simple average	18.05	16.69	156	1.09e-2	8.0
Best base-model	19.49	18.02	164	1.22e-2	21.6

(*) MSE was computed on the log values of calculated and experimental *y*.

The most popular measure for assessing the accuracy of a model is the MSE on the generalization data; it does not, however, always adequately describe the quality of the model's fitting. More informative and relevant measures include AARE, STDEV, and MAXARE. Among the series bm$_1$ - bm$_3$ (Table 1.10), bm$_1$ was the best model, with the lowest AARE on generalization data, while its PCE approximated that of the other two models. The meta-model (Figure 1.14) using the n = 3 base-models bm$_r$, was a weighted summation of their outputs:

$$meta(\mathbf{p}) = \sum_{r=1}^{n} \beta_r \cdot bm_r(\mathbf{p}) \tag{1.49}$$

As y spans several decades, log values of the base-model outputs were taken as the linear regressors of the log value of y. However, when the statistics i-v were reported, actual y values, except for MSE, were employed.

The simplest way to estimate β is by minimization of MSE for the meta-model on the training data (Hashem, 1997). This does no, however, necessarily, lead to the lowest AARE and STDEV for the meta-model. To circumvent such a limitation, we defined in this work a different optimality criterion to determine the meta-model regression coefficients β. This AARE+STDEV criterion is simply the sum of AARE and STDEV the meta-model achieves on the N$_T$ training data:

$$C(\boldsymbol{\beta}) = \frac{1}{N_T}\sum_{k=1}^{N_T}\left|\frac{y(\mathbf{p}_k)-meta(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right| + \sqrt{\sum_{i=1}^{N_T}\left[\left|\frac{y(\mathbf{p}_k)-meta(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right| - \frac{1}{N_T}\sum_{k=1}^{N_T}\left|\frac{y(\mathbf{p}_k)-meta(\mathbf{p}_k)}{y(\mathbf{p}_k)}\right|\right]^2 \Big/ (N_T-1)} \tag{1.50}$$

Proper value for **β** is obtained by minimizing criterion C using Newton's method, with the derivatives computed numerically.

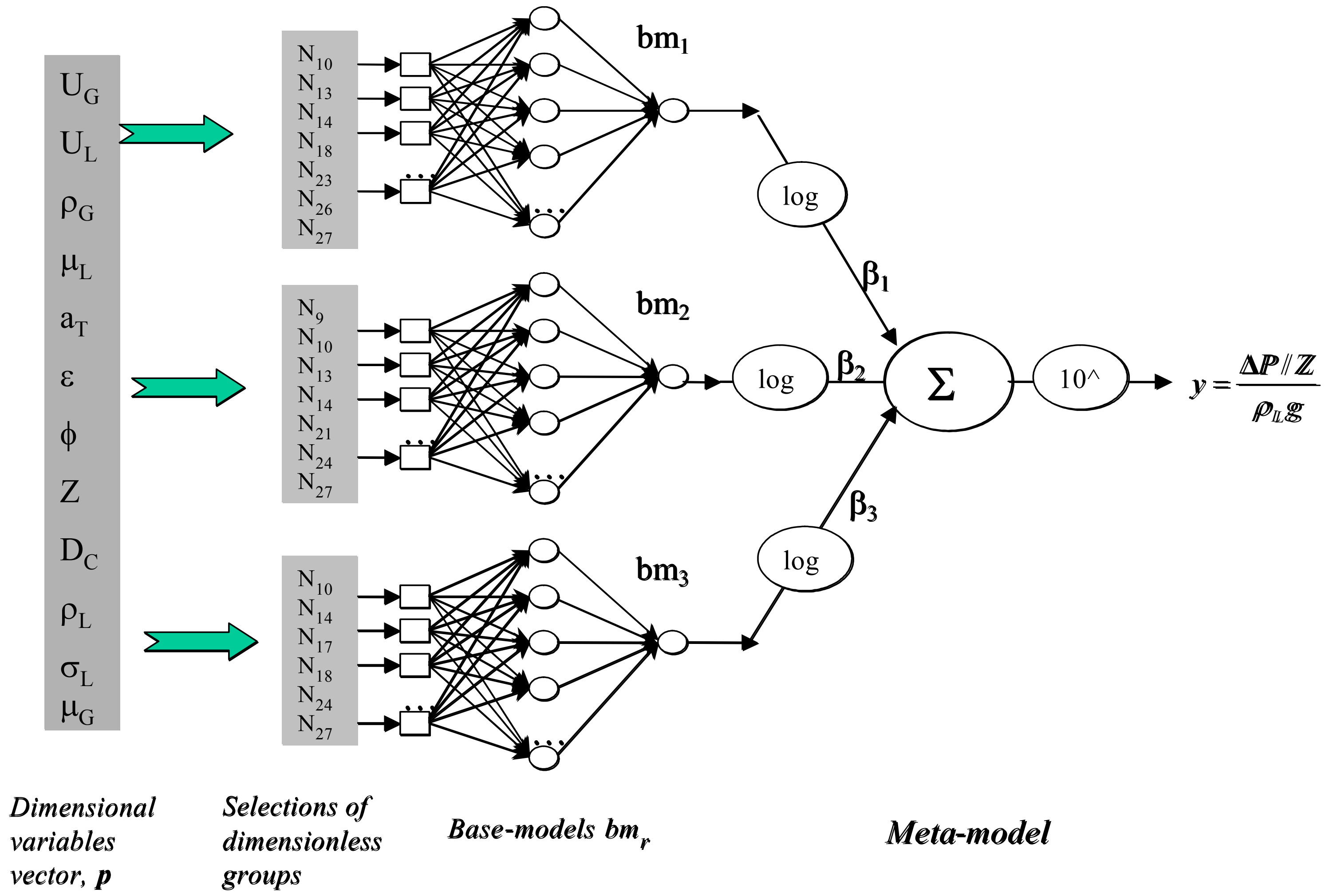

U_G
U_L
ρ_G
μ_L
a_T
ε
ϕ
Z
D_C
ρ_L
σ_L
μ_G
N_{10} N_{13} N_{14} N_{18} N_{23} N_{26} N_{27}
N_9 N_{10} N_{13} N_{14} N_{21} N_{24} N_{27}
N_{10} N_{14} N_{17} N_{18} N_{24} N_{27}
bm_1
bm_2
bm_3
log
log
log
β_1
β_2
β_3
Σ
10^
$y = \frac{\Delta P / Z}{\rho_L g}$
Dimensional variables vector, ***p***
Selections of dimensionless groups
Base-models bm_r
Meta-model

Figure 1.14 Meta-model construction: original variables converted in different dimensionless numbers become the inputs of the base models, whose outputs are fed in the meta-model, which predicts a dimensionless form of the pressure drop

1.3.3 Results and discussion

The β regression (or weighting) coefficients were determined according to Hashem (1997) (MSE) and Eq. (1.50) (AARE+STDEV) optimality criteria. The obtained weighting coefficients (Table 1.11) can be interpreted as the *certainty* of a network in its output (Alpaydin, 1993). All coefficients are significant and close to each other. However, this is not enough to guarantee robustness of the resulting meta-model; a potential problem that affects estimation of the β coefficients is colinearity among base-model outputs (Hashem, 1997; Breiman 1992). Colinearity has a chance to occur because all bm_r models are trained to approximate the same function.

Let us denote by $\mathbf{X}$ the matrix whose columns are the log values of the outputs of the bm_r models (which are linear regressors in the meta-model) for each point $\mathbf{p}_k$ in the training set. Eigenvectors (also called principal components) of the scaled and non-centered $\mathbf{X'X}$ matrix, as well as condition indices and variance-decomposition proportions, are computed for individual variables (regressors in the meta-model) using the SPSS software. According to Belsley (1991), a colinearity problem occurs when an eigenvector associated with a high condition index contributes strongly to the variance of two or more variables. This was not the case in our three base models bm_r.

To determine if the models completed each other and the resulting meta-model was robust, we compared its performance with the best base-model and the simple average model. The statistics i-v presented above were used in the comparisons; the results are summarized in Table 1.12. Both MSE and AARE+STDEV optimal meta-models outperformed the best and the simple average models. Even the simple average model was a improvement over the best base-model. Moreover, AARE, STDEV, MAXARE, and PCE were respectively reduced by 13%, 20%, 23%, and 173% if the AARE+STDEV optimal meta-model was used instead of the best base-model.

To illustrate how a meta-model achieves lower AARE and PCE than base-models, an experimental point $\mathbf{p}$ was chosen. The base-models and meta-model were use to simulate an output for a range of liquid velocity values, u_L, in the vicinity of $\mathbf{p}$ (Figure 1.15).

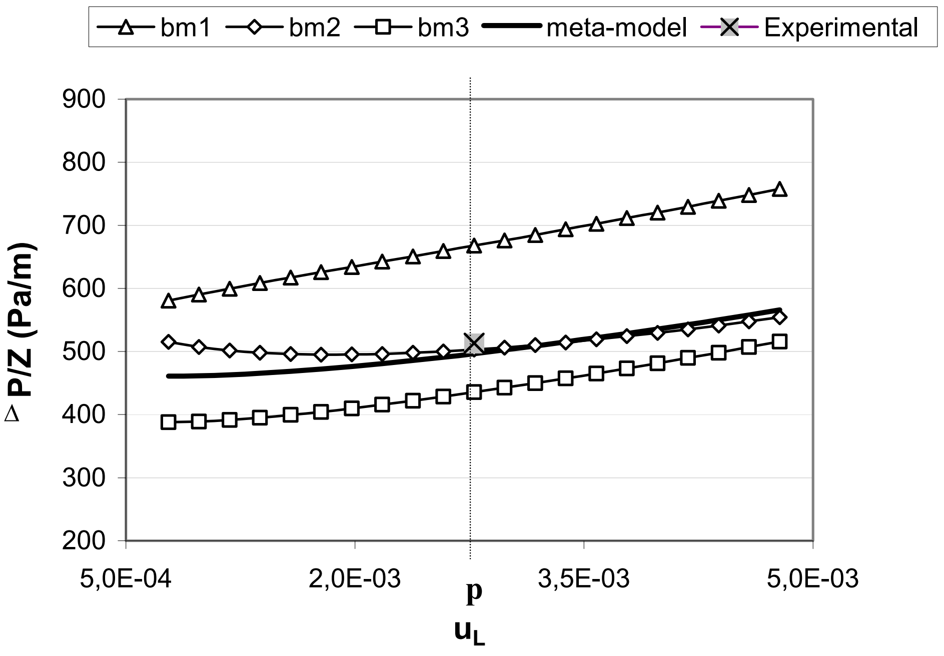


Figure 1.15 The meta-model showing monotony with respect to u_L and accuracy in prediction for data point **p**. The base-models show either imprecision or phenomenological inconsistence

This example shows that although the base-model bm_2 predicted the pressure drop value at point **p** very well, it failed one of Eqs. (1.36)-(1.40) monotonicity constraints. This would eventually lead to the incorrect prediction of a point with lower u_L. On the other hand, although the other two models, bm_1 and bm_3, exhibited monotonically increasing trends, their predictions were not as accurate, overestimating (bm_1) or underestimating (bm_3) the pressure drop at point **p**. Conversely, the meta-model was not only very accurate near **p,** but also adhered to the monotonicity constraint with respect to the liquid velocity.

Now, in the end of the dimensionless correlations chapter, we would like to give the reader some details pertinent to the neural networks experimentation performed herein. The estimate of the neural network accuracy may vary as a function of the partition of data in training and generalization sets, as well as on the weight initialization (Flexer, 1994; Prechelt, 1998). In sections 1.2.2 and 1.2.3, the error rates of models and a measure of disagreement with the prior knowledge in terms of monotonicity were combined to guide the search for good input combinations. While performing this search with the genetic algorithm, the partition of data and the weights initialization remained fixed in both cases (1.2.2 and 1.2.3). However, due to the inherent parallelism of the GA search, features which were relevant could not be eliminated by chance alone just because of a poor weight initialization. This is because the features were present in the population in a multitude of combinations which could not fail simultaneously. Of course, once a combination of inputs was selected (section 1.2.3), different partitionings of data and weight initializations were performed before proposing a final model.

1.4 Conclusions

In this first chapter, we treated the issue of regression with neural networks whose inputs are dimensionless groups computed from the dimensional variables: i.e., a collection of 14 physical properties and operating conditions, characterizing the three (G, L, S) phases. We devised a genetic algorithm-based procedure for building ANN models by identifying the most expressive dimensionless groups and their numbers, as well as appropriate network architecture. We directed the search toward models matching the monotonicity restrictions gathered from prior knowledge concerning the particular modeled characteristics. In a first step, the monotonicity was tested at the edges of definition ranges for the dimensional variables. Even though the automated GA-ANN procedure easily identified several passing models, there was no guarantee that such behavior was representative of the whole domain. A new measure of the monotonicity behavioral likelihood, termed Phenomenological Consistency Error (PCE), was devised as a more significant measure. Several models were identified matching the monotonicity rules in the vicinity of 80% of the points used for test purposes. These models differed mostly by their inputs and network architecture. The third part of this chapter treated the possibility of further reducing the prediction error and, more importantly, the PCE, by exploiting the diversity of the models. Positive results were obtained, as PCE and AARE on generalization sets were further diminished . However, 0% could not be achieved, as the monotonicity was not imposed in the neural model, but assessed via PCE after training. Imposing monotonicity via hard restrictions in the functional form of the neural network model will be treated in the next chapter. This is possible only if the dimensional variables are the inputs of the network and not dimesionless numbers computed from from them.

1.5 Notation

$ANN_j(S)$	ANN having j hidden nodes and using the m-input selector S
a	Interfacial area (m2)
a_S	External area of particle and wall per unit volume aS = aT + 4/DC (m2/m3)
a_T	Bed-specific surface area (m2/m3)
B	Bed number $B = \frac{6 \cdot (1-\varepsilon)}{a_T \cdot D_C \cdot \sqrt{\phi}}$
Bl_G	Gas Blake number $Bl_G = \frac{\rho_G \cdot u_G}{a_T \cdot (1-\varepsilon) \cdot \mu_G}$
Bl_L	Liquid Blake number $Bl_L = \frac{\rho_L \cdot u_L}{a_T \cdot (1-\varepsilon) \cdot \mu_L}$
bm_r	Individual neural network model
C	Scaling coefficient in linear conversion of criterion into fitness function
Ca_L	Liquid Capillary number $Ca_L = \frac{u_L \cdot \mu_L}{\sigma_L}$
$C(\boldsymbol{\beta})$	AARE+STDEV criterion depending on $\boldsymbol{\beta}$ coefficients
c_c	Convergence criterion in ANN learning step
D_C	Column diameter (m)
d_h	Krischer-Kast hydraulic diameter $d_h = d_{PV}\left(16\varepsilon^3/(9 \cdot \pi \cdot (1-\varepsilon)^2)\right)^{1/3}$ (m)

d_P	Sphere diameter equivalent to the particle specific area $d_P = 6(1-\varepsilon)/a_T$ (m)
d_{PV}	Sphere diameter equivalent to the particle volume $d_{PV} = 6(1-\varepsilon)/(\phi \cdot a_T)$ (m)
Eo_L	Liquid Eotvos number $Eo_L = \frac{\rho_L \cdot g \cdot d_P^2}{\sigma_L}$
Eo'_L	Modified liquid Eotvos number $Eo'_L = \frac{\rho_L \cdot g}{\sigma_L \cdot a_T^2}$
f_{LGG}	Friction factor
Fr_L	Liquid Froude number $Fr_L = \frac{u_L^2}{g \cdot d_P}$
Fr_G	Gas Froude number $Fr_G = \frac{u_G^2}{g \cdot d_P}$
g	Generation in a GA run; gravitational constant (m/s^2)
Ga_G	Gas Galileo number $Ga_G = \frac{\rho_G^2 \cdot g \cdot d_P^3}{\mu_G^2}$
Ga_L	Liquid Galileo number $Ga_L = \frac{\rho_L^2 \cdot g \cdot d_P^3}{\mu_L^2}$
Ga'_G	Modified Gas Galileo number $Ga_G = \frac{\rho_G^2 \cdot g}{\mu_G^2 \cdot a_T^3}$
Ga'_L	Modified Liquid Galileo number $Ga_L = \frac{\rho_L^2 \cdot g}{\mu_L^2 \cdot a_T^3}$
H	Number of hidden nodes
H_j	Activation function of the j neuron in hidden layer

H_A	Henry's law constant(-)
I_i	Input variable representing a column in the database
J	Number of nodes in hidden layer
J_{max}	Maximum number of nodes in hidden layer
K_1	Wall factor $K_1 = \left(1 + \frac{2}{3(1-\varepsilon)}\left(\frac{d_{PV}}{D_C}\right)\right)^{-1}$
K_2	Wall factor $K_2 = 2\left(\frac{d_{PV}}{D_C}\right)$
K_3	Wall factor $K_3 = \left(\frac{d_{PV}}{D_C}\right)\left(\frac{Z}{D_C}\right)$
K_L	Overall mass transfer coefficient (m/s)
k_L	Liquid-side mass transfer coefficient(m/s)
k_g	Gas-side mass transfer coefficient(m/s)
m	Number of ANN inputs selected by **S**
M	Number of input columns in the database
MAXARE	Maximum absolute relative error
MAXPOP	Size of population
*meta(**p**$_k$)*	Output of the meta-model for the input point ***p***$_k$
Mo_L	Liquid Morton number $Mo_L = \frac{\mu_L^4 \cdot g}{\rho_L \cdot \sigma_L^3}$
MSE-OLC	Mean square error optimal linear combination

N	Number of samples in a data set
N_i	Dimensionless group computed from the dimensional variables
N_P	Number of packings per unit bed volume (m^{-3})
N_w	Number of connectivity weights
O	Output variable of interest for a particular problem
Oh_L	Liquid Ohnesorge number $Oh_L = \sqrt{\frac{\mu_L^2}{\rho_L \cdot \sigma_L \cdot d_p}}$
P	Pressure (Pa)
$\boldsymbol{p}_k$,	Vector of the dimensional variables recorded at the position *k* in the data base, $\mathbf{p_k} = \{u_G, u_L, \rho_G, \mu_L, a_T, \varepsilon, \phi, Z, D_C, \rho_L, \sigma_L, \mu_G\}$
PPC	Number of phenomenological rules violated by an ANN model
Q(**S**)	Value of criterion for **S**
Re_G	Gas Reynolds number $\mathrm{Re}_G = \frac{u_G \cdot \rho_G \cdot d_P}{\mu_G}$
Re_L	Liquid Reynolds number $\mathrm{Re}_L = \frac{u_L \cdot \rho_L \cdot d_P}{\mu_L}$
S	Combination of dimensionless numbers or input selector
S	Normalized output variable
S_B	Bed correction factor $\mathrm{S_B} = (\mathrm{a_S} \cdot \mathrm{d_h})/(1-\varepsilon)$
S_{B2}	Bed correction factor $S_{B,2} = \left(16\varepsilon^3/(9 \cdot \pi \cdot (1-\varepsilon)^2)\right)^{1/3} \cdot \frac{a_T^2 / a_S}{1-\varepsilon}$

S_{B3}	Bed correction factor $S_{B,3} = \left(16\varepsilon^3/(9\cdot\pi\cdot(1-\varepsilon)^2)\right)^{1/3}\cdot\frac{a_S^2/a_T}{1-\varepsilon}$
St_L	Liquid Stokes number $St_L = \frac{\mu_L\cdot u_L}{\varepsilon\cdot\rho_L\cdot g\cdot d_p^2}$
St_G	Gas Stokes number $St_G = \frac{\mu_G\cdot u_G}{\varepsilon\cdot\rho_G\cdot g\cdot d_p^2}$
St'_L	Modified Liquid Stokes number $St'_L = \frac{\mu_L\cdot u_L\cdot a_T^2}{\varepsilon\cdot\rho_L\cdot g}$
St'_G	Modified Gas Stokes number $St'_G = \frac{\mu_G\cdot u_G\cdot a_T^2}{\varepsilon\cdot\rho_G\cdot g}$
STDEV	Standard deviation of ARE

$$\sigma = \sqrt{\sum_{i=1}^{N}\left[\left|\frac{y^{\exp}(P_k)-y^{calc}(P_k)}{y^{\exp}(P_k)}\right| - AARE\right]^2 \Big/ (N-1)}$$

u	Phase velocity (m/s)
U_i	Normalized input variable
v_j	Testing dimensional variable
We_L	Liquid Weber number $We_L = \frac{u_L^2\cdot\rho_L\cdot d_P}{\varepsilon^2\cdot\sigma_L}$
$w_{i,j}$, w_j	ANN connectivity weights
X	The matrix whose columns are the log values of the outputs of the bm_r models for each point $\mathbf{p}_k$ in the training set
$y^{(calc)}(\mathbf{p}_k)$	Dimensional pressure drop calculated from the output S for the input vector $\mathbf{p}_k$

$y^{(\exp)}(\mathbf{p_k})$, $y(\boldsymbol{p}_k)$	Dimensional pressure drop measured for the input vector $\mathbf{p_k}$
Z	Bed height (m)

Greek letters

χ	Lockhart-Martinelli parameter $\chi = \frac{U_G}{U_L}\sqrt{\frac{\rho_G}{\rho_L}}$
ρ	Phase density (kg/m^3)
μ	Phase viscosity (kg/m.s).
$\boldsymbol{\beta}$	Weighting coefficients vector in the meta-model
α, β	Weighting coefficients in the criterion
ε	Bed porosity; normally distributed, zero-mean, random variable
ϕ	Particle sphericity factor $\phi = (\pi \cdot N_P / a_T)(6(1-\varepsilon)/\pi \cdot N_P)^{2/3}$
ΔP	Irrigated pressure drop (Pa)
σ	STDEV; phase surface tension (N/m)

Abbreviations

AARE	Average absolute relative error; $\mathrm{AARE} = \frac{1}{N}\sum_{k=1}^{N}\left\lvert\frac{y^{\exp}(P_k) - y^{\mathrm{calc}}(P_k)}{y^{\exp}(P_k)}\right\rvert$
ANN	Artificial Neural Network
ARE	Absolute relative error

PC	Phenomenological consistency
PCE	Phenomenological consistency error

Subscripts and Superscripts

calc	Calculated
exp	Experimental
G	Gas, generalization
G+T	Generalization and training
GA	Genetic algorithm
L	Liquid
max	Maximum
min	Minimum
pred	Predicted

2. Neural Network Dimensional Correlations for Continuous Multiphase Reactors Characteristics

2.1 Bibliographical review and problematic

In this chapter we deal with the case when the inputs in a neural network model are the dimensional variables describing the three phases. These inputs should provide information about the output characteristic we wish to model. In this situation, the feature selection problem is less pronounced than in dimensionless modeling. This is because in multiphase systems, the raw variables are less numerous than the possible dimensionless groups from which they can be edited. Of course, some of these variables might be more useful than others in predicting the characteristic of interest; however, we shall consider that the set of features is already selected. The problem from the previous chapter then remains:, i.e., how do we select models that not only accurately predict the data, but which also match prior knowledge in terms of monotonicity behavior? There are not as many examples in multiphase reactor literature of dimensional ANN correlations as there are in the dimensionless category. Larachi *et al.* (1999) proposed ANN modeling of the liquid superficial velocity at transition between the trickle and pulse flow regimes as a function of the raw variables: liquid viscosity (μ_L), superficial gas velocity (v_{SG}), gas density (ρ_G), volume-equivalent particle diameter (d_p), and two bed parameters as:

$$v_{SL,tr} = f(\rho_L, \mu_L, \sigma_L, v_{SG}, \rho_G, d_p, \varepsilon) \tag{2.1}$$

However, the larger field of chemical engineering, as well as other fields of science, has attempted to use monotonicity information. Recent works have emphasized the importance of *a priori* information and domain-specific knowledge in neural network development. Abu-Mostafa (1993), referring to the supplementary information about the function to be learned as *hints*, developed a systematic method for incorporating this *a priori* knowledge in the usual learning-from-examples process. Typical hints that were considered included invariance, monotonicity, and approximation, which were presented to the learning process by examples. Monotonicity, however, has been studied in more recent works considering the subject of embedding *a priori* knowledge in the neural network development. Wang

(1996) proposed an algorithm to build monotonic concave back-propagation networks by choosing as training samples only those data points from the training set satisfying a mono-concave relationship explicitly. In a finance application, Sill (1998) proposed a modified feed-forward neural network trained with a gradient-based technique guaranteeing monotonicity by its functional form. In chemical engineering applications, monotonic neural networks have been introduced by Kay and Ungar (1993, 2000). They developed monotonic multilayer feed-forward neural networks by forcing the signs of the networks' weights. They also showed that monotonicity information contributes to squeeze down model confidence band, yielding more reliable models. Network training was performed using a sequential quadratic algorithm; no second-order information was used. Second order monotonicity information refers to the sign of the second order derivative of the modeled characteristic with respect to an input variable, coinciding with concavity.

In the next sections we present a new method for building neural networks that are able to achieve 100% monotonicity at the first and possibly the second order. The neural network training is performed by means of an evolutionary algorithm which combines genetic and hill climbing searches (GA-GHC). To date, there are no reports of using genetic algorithm based search techniques for training monotonic-concave neural networks. Furthermore, in contrast with past attempts to develop dimensionless multiphase flow correlations, we used raw dimensional operating variables, rather than the dimensionless groups, as network inputs. When mapping the dependent variable y (reactor transport parameter) to the G-L-S operating (independent) variables $\mathbf{v} = (v_1, v_2, ..., v_a)$, it is not necessarily better to reduce the dimensionality of the input vector by creating fewer dimensionless groups N_i=f($\boldsymbol{v}$) to be used as model inputs. The dimensionless groups $\mathbf{N} = (N_1, N_2, ..., N_b)$, $b<a$ may be cross-correlated (as they may contain some common dimensional variables v_i) which is not helpful for learning. Also, the number of network weights required for capturing the relationship $y(\mathbf{N})$ is not necessarily less than the number needed for learning the relationship $y(\mathbf{v})$. Finally, it is far more awkward to develop mathematically guaranteed monotonic networks with respect to the dimensional variables when the network's inputs are dimensionless groups N_i. The proof-of-concept of embedding monotonicity and concavity information in the training of NNs by means of genetic algorithms will be illustrated in correlating total liquid holdup in randomly packed bed containing counter-current gas-liquid towers.

2.2 Monotonic networks

The most common architecture used in function approximation is the multilayer feed-forward neural network, depicted in Figure 2.1. It consists of *I* nodes in the input layer, *J* hidden nodes, and a single-output node. Hidden and output layers are endowed with a nonlinear activation function, the logistic sigmoid:

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2.2}$$

The estimate produced by the network is computed as:

$$\hat{y} = \sigma\left(\sum_{j=1}^{J}\left(w_j \cdot \sigma\left(\sum_{i=1}^{I}(x_i \cdot w_{i,j}) + w_{I+1,j}\right)\right) + w_{J+1}\right) \tag{2.3}$$

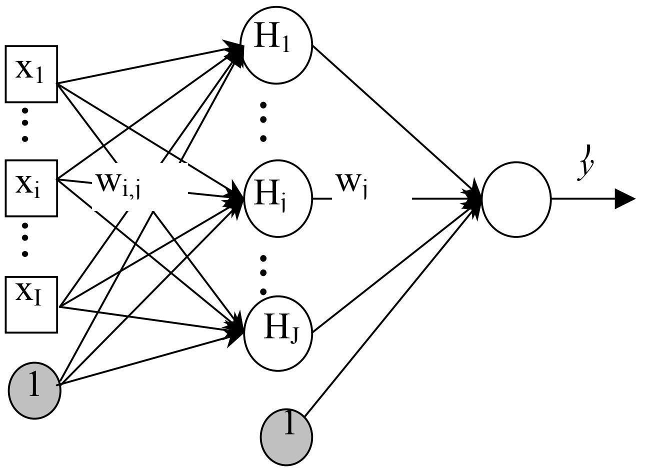


Figure 2.1 Typical feed-forward multilayer neural network used for function approximation

One classical approach for training feed-forward neural networks focuses on minimizing the sum of squared errors $\sum_{i=1}^{N_T}(y_i - \hat{y}_i)^2$ of the fitting function *f(**x**,**w**)* on a training data set, where y_i represents the true (experimental) value of y for the input pattern $\boldsymbol{x}_i$, $\hat{y}_i$ = $f(\mathbf{x_i},\mathbf{w})$, and N_T is the number of training samples. Very often, there are too many degrees of freedom in estimating the networks weights, ***w***, using such a simple sum of squared errors criterion, leading thus to networks exhibiting poor generalization ability. The training process optimizes the weights in such a way that the fitting function $f(\mathbf{x},\mathbf{w})$ too closely approaches some training points, which incidentally may be corrupted or noisy. This problem, known as overfitting, may be reduced at the cost of introducing bias with some regularization or early stopping techniques. The degrees of freedom in the training process may be drastically reduced, and hence overfitting diminished, if the course of possible functions is narrowed to include only monotonic functions (Kay and Ungar, 1993, 2000). This becomes possible provided the signs of the first derivative of the function $f(\mathbf{x},\mathbf{w})$, to be learned with respect to some particular inputs x_k, are known with *certainty*. If this is the case, $f(\mathbf{x},\mathbf{w})$ may be constrained to obey that expected behavior. For example, for *non-strict increasing* monotonicity:

$$\frac{\partial f}{\partial x_k} \geq 0 \tag{2.4}$$

The 1st derivative may be calculated as:

$$\frac{\partial f}{\partial x_k} = \frac{e^{-a}}{(1+e^{-a})^2}\left(\sum_{j=1}^{J} w_j \cdot \frac{e^{-b}}{(1+e^{-b})^2} \cdot w_{k,j}\right) \tag{2.5}$$

with

$$a = \sum_{j=1}^{J}\left(w_j \cdot \sigma(b)\right) + w_{J+1} \tag{2.6}$$

$$b = \sum_{i=1}^{I}\left(x_i \cdot w_{i,j}\right) + w_{I+1,j} \tag{2.7}$$

A sufficient condition for satisfying Eq. (2.4) is:

$$w_j \cdot w_{k,j} \geq 0, \quad \forall j, 1 \leq j \leq J \tag{2.8}$$

Training the above monotonically increasing neural networks with respect to input x_k is equivalent to solving the following optimization problem (Kay and Ungar, 1993, 2000):

$$\min_{w} \sum_{i=1}^{N_T} (y_i - \hat{y}_i)^2 \qquad \textit{subject to} \quad w_j \cdot w_{k,j} \geq 0, \quad \forall j, 1 \leq j \leq J \tag{2.9}$$

To enforce concavity as well as first-order monotonicity, supplementary constraint on the second-order derivative must be imposed: non-strict upward concavity or $\frac{\partial^2 f}{\partial {x_k}^2} \geq 0$, and non-strict downward concavity, or $\frac{\partial^2 f}{\partial {x_k}^2} \leq 0$, in as much as Eq. (2.4) is verified. Second derivative is computed by differentiating Eq. (2.5) r.h.s. Unlike monotonicity (Eq. (2.4)), it is difficult to infer network concavity information via the signs of weights (such as Eq. (2.8)). For practical reasons, function concavity may be judged only at particular learned sample points where monotonicity is fulfilled. However, the problem described by Eq. (2.8) with supplementary penalties forcing to desired concavity (upward or downward), cannot be solved with simple constrained nonlinear optimization techniques.

2.3 Reformulation of neural network training problem with monotonicity and concavity constraints

Let us denote by *m* the number of network inputs for which monotonicity information is available, and by *c* the number of inputs among the *m* ones for which concavity information is also given $(c \leq m)$. Monotonicity and concavity information for a particular problem can be given in a two-row and *m*-column matrix referred to as MCI matrix:

$$MCI = \begin{array}{c} \\ \\ \end{array}\begin{array}{cccc} 1 & 2 & \ldots\; k \ldots & m \\ \end{array}$$

$$MCI = \begin{bmatrix} -1 & 1 & \ldots & -1 \\ 1 & -1 & \ldots & 0 \end{bmatrix} \begin{array}{l} \textit{monotonicity} \\ \textit{concavity} \end{array}$$

A '1' value in 1st row and kth column of MCI matrix means $\frac{\partial f}{\partial x_k} \geq 0$; similarly, '-1' stands for $\frac{\partial f}{\partial x_k} \leq 0$. Equivalently, a "1" value in 2nd row and kth column means $\frac{\partial^2 f}{\partial x_k{}^2} \geq 0$, whereas "-1" stands for $\frac{\partial^2 f}{\partial x_k{}^2} \leq 0$, and "0" for no concavity restrictions when is not supplied. No zeroes occur in the first row of the MCI matrix because all the network's inputs, where monotony information is unavailable, are excluded from the list.

To obtain a neural network that closely agrees with the defined MCI matrix, we need to reformulate the problem of the network training as the minimization of a composite criterion, which is equivalent to a global error (*GErr*):

$$\mathrm{GErr}(\mathbf{w}) = \sum_{i=1}^{N_T} (y_i - \hat{y}_i)^2 + \alpha \cdot P_{mon} + \beta \cdot P_{conc} \tag{2.10}$$

where P_{mon} and P_{conc} are, respectively, penalty functions for monotonicity and concavity.

A natural form for the monotonicity penalty function is:

$$P_{mon} = \sum_{k=1}^{m} \sum_{j=1}^{J} mon(\mathbf{w}) \tag{2.11}$$

$$mon(\mathbf{w}) = \begin{cases} 1 \quad if \quad \mathrm{sgn}(w_j \cdot w_{k,j}) = -\mathrm{sgn}(MCI(1,k)) \\ 0 \quad otherwise \end{cases} \tag{2.12}$$

where *sgn* is the sign function. In this way, we measure the *degree* in which a particular weight vector ***w*** satisfies Eq. (2.10) for all k inputs. The penalty function that accounts for concavity agreement between model and *a priori* knowledge is defined as:

$$P_{conc} = \begin{cases} c & if \quad P_{mon} > 0 \\ \sum_{k=1}^{c} conc(\mathbf{w}) & otherwise \end{cases} \tag{2.13}$$

$$conc(\mathbf{w}) = \begin{cases} 1 & if \quad \mathrm{sgn}(\frac{\partial^2 f}{\partial x_k{}^2}) = -\mathrm{sgn}(MCI(2,k)) \\ 0 & otherwise \end{cases} \tag{2.14}$$

The role of the penalty terms is explained as follows: P_{mon} is evaluated through Eq. (2.11). $P_{mon} = 0$ implies that *f(**x**, **w**)* is monotonic with respect to all first-row inputs of the MCI matrix. It makes sense then to turn to $\frac{\partial^2 f}{\partial x_k{}^2}$ to compute the penalty term for concavity P_{conc} to discriminate, among networks, which fulfill or violate concavity. If $P_{mon} \neq 0$, P_{conc} is attributed maximum penalty value, and no second-order derivative is computed.

Setting the appropriate values for multipliers in Eq. (2.10) is important if monotonic networks with low SSE are to be created. These monotonic networks should also satisfy the concavity constraint. Guidance on how to set them is based on the following heuristics:

Consider the r.h.s of Eq. (2.8) divided by the number of training samples. It would read: $MSE + \frac{\alpha}{N_T} \cdot P_{mon} + \frac{\beta}{N_T} \cdot P_{conc}$, with MSE representing the mean square prediction error.

All of these terms together are to be minimized by a genetic algorithm, which was designed to be elitist in nature; i.e., it never drops the best solution found.

For a given problem, there always exists a multitude of weight sets **w** for which $P_{mon} = 0$, so the initial population of the optimization algorithm is formed by (randomly generated) weight sets whose signs are set in such a way that P_{mon} is null. The MSE term will always be lower than 1, as both the estimate

$\hat{y}_i$ and the true y_i belong to the interval [0,1]. The lowest degradation (i.e., increase with respect to zero) of the term P_{mon} is 1.

In order to ensure that the optimization scheme will never prefer a weight set with lower MSE and non-null P_{mon}, it is enough to set $\frac{\alpha}{N_T} \cdot 1$ greater than the maximum hypothetical decrease in the MSE term, i.e., 1 (from 1 to 0). In practice, we may choose $\frac{\alpha}{N_T}$ as equal with the MSE of the best weight set randomly generated in the first generation of the genetic algorithm (but having $P_{mon} = 0$). The value of $\frac{\beta}{N_T}$ should be set as a fraction of the value of $\frac{\alpha}{N_T}$ in order to generate solutions with considerably lower MSE, even if they exhibit non-null P_{conc}, rather than an ill-fitting solution with $P_{conc} = 0$. β value, such that $0 < \beta < \alpha$, is problem-dependent, and should be established as the fraction of the total potential decrease in MSE sacrificed for the sake of having $P_{conc} = 0$ verified. If the training data exhibits concavity in the sought direction, the MSE will be decreased by the genetic algorithm (by tuning the values of weights and/or their signs) up to the point when $P_{conc} = 0$ would be fulfilled.

The function to be optimized (Eq. (2.10)) is multimodal due to the sum of the squared-errors term and/or the penalty terms that also induce discontinuity of the function *GErr(**w**)*. There are several methods that can be employed to optimize a multivariable function for which gradient information may not be used. There are genetic algorithms, random search, stochastic hill climbing, particle swarm optimization, simulated annealing, or combinations thereof (Krink and Løvbjerg, 2002). Here, the network training being defined as a minimization problem of criterion *GErr* for vector ***w***, let us describe next the algorithm used to perform optimization.

2.4 Genetic algorithm - genetic hill climber optimizer

Gradient-based techniques are not suitable because of the *discontinuity* nature of the penalty functions added in the training criterion. Hence, since the function to be optimized is locally similar to a classical error surface of neural networks, it is more appropriate to use genetic algorithms (Schaffer *et al*., 1992; Branke, 1995; Whitley, 1991, 1995).

Genetic algorithms (GAs), first pioneered by Holland (1975), are now among the most general optimization methods used in science and engineering. As described by Goldberg (1989), classical genetic algorithms applied to multivariable function optimization require first encoding the variables $\boldsymbol{w}$ in a binary string that becomes an individual in a population that is evolved through several generations. A first population is randomly initialized, and then the function value is calculated for each individual among this population. The next generations are formed by means of three genetic operators: *reproduction* (selection), *recombination* (crossover), and *mutation*. The reproduction operator ensures that highly-fitted individuals will be propagated in the next generation and/or produce offspring by crossover and mutation. The recombination operator yields two new individuals from two parents by simply interchanging bit substrings. (The start point and length of interchanged substrings are randomly chosen). Mutation maintains diversity within the population by altering, with a low probability, the value of individual bits. All the implementation aspects in a computer algorithm of the evolutionary principles affect performance, especially when the optimization problem is a neural network training problem where the permutations problem (numerous equivalent symmetric solutions) makes the search difficult (Whitley, 1995).

Classical genetic algorithms have not proven effective in neural networks training, and several modifications have been suggested in the literature. To be distinguished from classical hyper-plane sampling genetic algorithms, such search algorithms have been named *genetic hill-climbers* (Whitley, 1995). The first modification concerned using real-valued encoding (Branke 1995; Whitley 1991; Whitley 1995) instead of binary encoding (Figure 2.2). This implies that recombination may occur only between weights. Second, the role of mutation has been switched from a mere background operator to a principal role during search. Finally, the population size has been lowered (e.g., 50 individuals).

$$\mathbf{w} = [\underbrace{[w_{1,1}, w_{1,2} \dots w_{1,J}],[w_{2,1}, w_{2,2} \dots w_{2,J}]\dots[w_{I+1,1}, w_{I+1,2} \dots w_{I+1,J}]}_{\mathbf{w}_{i,j}}, \underbrace{w_1, w_2 \dots w_{J+1}}_{\mathbf{w}_j}]$$

Figure 2.2 Real-valued string representation of a feed-forward neural network with single output node

The evolutionary algorithm designed here combines a kind of classic genetic search with genetic hill-climbing. The idea behind this approach is to take advantage of two fundamentally different improvement processes: gene interchange between individuals (specific to classical GAs) and hill-climbing in attractive regions specific to stochastic hill-climbers.

```
1. Initialize a MAXPOP population of real encoded individuals. g=1.
2. Do
          a) Apply reproduction (selection)
          b) If g%3≠0 (i.e., for 2 of 3 generations)
               •  Apply classic crossover                      } Classic genetic search
               •  Apply low probability mutation               }
             Else (i.e., for 1 of 3 generations)
               •  Apply modified arithmetic crossover          } Genetic hill climbing
               •  Apply high probability mutation              }
          c) g=g+1;
   While (solution not found or g ≤ MAXSESSIONS)
3. Stop
```

Figure 2.3 Pseudo-code for the genetic algorithm-genetic hill climber optimizer

The basic steps of the proposed algorithm are depicted in Figure 2.3. A population of MAXPOP individuals is first randomly initialized within a given range specific to the optimization problem. Then, the reproduction operator (to be described in detail later) promotes some of the best individuals in the next generation g+1. Within two of three generations, a classical GA search is performed using classical crossover (substring interchanges of weights between two parents to yield offspring) and low probability mutation. In one of three generations, genetic hill-climbing is performed using a high mutation rate and an arithmetic crossover that generate two offspring by linear combination of some parts of the parent strings.

2.4.1 Reproduction (Selection)

The purpose of this operator is to ensure that the fittest specimens perpetuate through offspring and/or have greater chances to be found in the next generation. Numerous schemes are known which introduce various levels of determinism into the selection process. The one we used was the stochastic remainder selection without replacement but with elitism (Goldberg, 1989). The principle here is that above average individuals are receiving at least one copy in the next generation, while even the inferior individuals retain a chance to be promoted. As the criterion *GErr(**w**)* was minimized, a linear transformation was applied to convert it into a *fitness* measure:

$$fitness_i^* = C \cdot Gerr_{\max} - GErr_i \tag{2.15}$$

A linear fitness scaling was performed as described elsewhere (Goldberg, 1989) to transform this raw fitness into functional fitness, $fitness_i$.

Then a survival probability was computed for each individual i of the *MAXPOP* population:

$$p_i = \frac{fitness_i}{\sum_{i=1}^{MAXPOP} fitness_i} \tag{2.16}$$

Using this probability, the expected number of copies of each solution i was computed:

$$E_i = MAXPOP \cdot p_i \tag{2.17}$$

Each solution was then copied into the next generation $Int(E_i)$ times, $Int(E_i)$ being the integer part of E_i. To complete the new population to *MAXPOP* individuals, the fractional remainder

$$R_i = E_i - Int(E_i) \tag{2.18}$$

of each solution was treated as the probability of further duplication with a weighted simulated coin toss until the new population was balanced. As this selection is also elitist, special attention was paid to the best *g*-generation individual stored and inserted in *g*+1 after crossover and mutation were performed.

2.4.2 Recombination (Crossover)

In binary encoded GAs, crossover exchanges substrings from two different parents, creating two children who inherit some of the parents' genetic material. The *classical crossover* used in our algorithms is a two-point crossover that treats the strings as a ring. It works basically the same way as in binary encoding except that, weights, not bits, are exchanged from homologous positions.

The *modified arithmetic crossover* employed here is also similar to classical crossover, except that the selected genes do not exchange places, but are linearly combined. Starting from two parents, $\boldsymbol{w}_{parent1}$ and $\boldsymbol{w}_{parent2}$, the children $\boldsymbol{w}_{child1}$ and $\boldsymbol{w}_{child2}$ are initialized as: $\boldsymbol{w}_{child1} = \boldsymbol{w}_{parent1}$ and $\boldsymbol{w}_{child2} = \boldsymbol{w}_{parent2}$. Then a substring $\boldsymbol{s}_{child1}$ selected with random start point and length from $\boldsymbol{w}_{child1}$ is replaced by the vector $t \cdot \mathbf{s}_{child1} + (1-t) \cdot \mathbf{s}_{child2}$ where $\boldsymbol{s}_{child2}$ is the homologous substring in $\boldsymbol{w}_{child2}$ and t is a random number between 0 and 1. Similarly, a substring $\boldsymbol{s}_{child2}$ selected with random start point and length from $\boldsymbol{w}_{child2}$ is replaced by the vector $(1-t) \cdot \mathbf{s}_{child1} + t \cdot \mathbf{s}_{child2}$ where $\boldsymbol{s}_{child1}$ is the homologous substring in the initial child1. Note that if $t = 0$, then the arithmetic crossover degenerates to classical crossover.

The proposed *modified arithmetic crossover* differs from the classical arithmetic crossover described by Krink and Løvbjerg (2002). The latter produces offspring as some linear combination of parents, while the former linearly combines only *fragments* from parents. This modified arithmetic crossover is less disruptive than the classical arithmetic crossover by letting unchanged potentially desirable parts in individuals to propagate *g* to *g*+1.

2.4.3 Mutation

Entry l of an individual w_l was mutated with a value Δw_l, a uniformly random-generated number within a range that reduces during the generational evolution.

Consider the monotonic decreasing function :

$$q(g) = \frac{\ln(MAXSESSIONS) - \ln(g)}{\ln(MAXSESSIONS)} \tag{2.19}$$

in which *g* represents the generation attained with GA-GHC, and *MAXSESSIONS* is a customary chosen maximum number of generations to explore. This function is a monotonically decreasing logarithmic function of *g* and takes its maximum value 1 at the first generation (*g=1*) and its minimum value of 0 at the last generation (g=MAXSESSIONS).

We used this function as an envelope for the amplitude of mutation increment Δw_l by calculating it as:

$$\Delta w_l = rnd\{wLow \cdot \gamma \cdot q(g); wHigh \cdot \gamma \cdot q(g)\} \tag{2.20}$$

where *rnd{a;b}* returns a uniformly random-generated number in the range (*a,b*), *wLow* and *wHigh* are, respectively, the lower and higher limits of the interval in which w_l is searched, and $\gamma = rnd(0,1)$. The γ coefficient was introduced to fluctuate the wideness of the interval from which Δw_l was sampled, i.e., to allow a certain number of small mutations. In the above heuristics, the interval {*wLow, wHigh*} was assumed symmetric with respect to 0 (as is the case for neural network weights identification) so that Δw_l would be equally likely to be a positive or negative value. This modality of computing mutational increment Δw_l allows function optimization in any real definition domain for the variable vector ***w*** because it embeds information about the domain bounds.

2.4.4 Benchmarking the GA-GHC optimizer

Before applying the algorithm to our real problem (liquid holdup neural network training), we decided to test it on some benchmark problems to assess its performance. The primary goal in adjusting the algorithm's parameters was to obtain enough search power to solve, with a single set of parameters, different problems, such as function optimization and neural network training. A summary of the characteristics and parameters of the GA-GHC optimizer is provided in Table 2.1.

Table 2.1 Parameters' set of the GA-GHC optimizer used in all benchmarks and the real problem

Representation	real
Selection	stochastic reminder selection
Fitness scaling	linear scaling
Elitist strategy	single copy of the best individual preserved

Genetic operators	for 2 of 3 generations: -two point crossover p_c=0.5
	- low probability mutation p_{lm} =0.03
	for 1 of 3 generations: -two point *modified arithmetic crossover* p_c=0.5
	- high probability mutation p_{hm} =0.5

Population size	MAXPOP = 50
Number of generations to explore	MAXSESSION: problem dependent
Search range	wLow, wHigh : problem dependent

Assessment of the GA-GHC optimizer was performed on three case studies:

- medium sized multimodal functions optimization (10-30 parameters)
- large multimodal functions optimization (100 parameters)
- neural network training (35 parameters)

The functions selected to test the solving power of the algorithm are very often encountered in optimization algorithm benchmarks. They are named after the authors that introduced them as benchmark functions (Table 2.2).

Table 2.2 Test functions

Function		Search Space
Rastrigin	$f_1(x) = \sum_{i=1}^{n} \left(x_i^2 - A \cdot \cos(2 \cdot \pi \cdot x_i) + A \right)$	$-5.12 \le x_i \le 5.12$
Griewank	$f_2(x) = \frac{1}{4000} \sum_{i=1}^{n} \left(x_i - 100 \right)^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i - 100}{\sqrt{i}} \right) + 1$	$-600.0 \le x_i \le 600.0$
Ackley	$f_3(x) = 20 + e - 20 \cdot \exp\left(-0.2 \cdot \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2 \cdot \pi \cdot x_i) \right)$	$-30.0 \le x_i \le 30.0$

The Rastrigin function, f_1, has many suboptimal peaks whose values increase as the distance from the global optimum increases. The product term in Griewank function, f_2, introduces interdependency between the variables; this is why this function disrupts the optimization techniques working on one variable at a time. The Ackley function, f_3, is also multimodal at low resolution. The optimum of these functions is located at (f (0,0,..0)=0).

Table 2.3 Results on test functions with medium dimensions number

Function	**Dimensions (n)**	**Range of initialization for all parameters**	**Minimum reached** Standard GA (1)	**Minimum reached** GA-GHC optimizer
Rastrigin (A=3)	20	$-5.12 \le x_i \le 5.12$	6	5.3
Griewank	10	$-600.0 \le x_i \le 600.0$	0.1	0.081
Ackley	30	$-30.0 \le x_i \le 30.0$	1	0.024

Table 2.3 shows the average over 50 runs of the minimum function value found (best individual) for a standard GA and for the GA-GHC optimizer after 100 000 function evaluations. The values for the standard GA(1) were extracted (with graph precision) from Potter and De Jong (1994). In this standard

GA$^{(1)}$, the representation of parameters is binary, so the crossover exchanges bit substrings between individuals in order to create offspring.

Other performances, referred to as standard GA$^{(2)}$, were recently reported (Krink and Løvbjerg, 2002) for real parameter representation and arithmetic crossover. This time, the number of dimensions is raised to 100 for all three functions. To make the search for the optimization algorithms more difficult, the initial population is asymmetrically initialized with respect to the global minimum. Note also that parameter *A* in the Rastrigin function is augmented from 3 to 10, rendering this function more difficult to optimize because of the stretching of the suboptimal peak's amplitude. Table 2.4 shows a comparison between the performances of the GA-GHC optimizer, a standard GA$^{(2)}$, and a stochastic hill-climber (SHC) Krink and Løvbjerg, 2002. The performance measure is, as in the previous benchmark, an average over 50 runs of the minimum function value found (best individual), but this time, the number of function evaluations is 2 500 000.

Table 2.4 Results on test functions with large dimensions number

Function	**Dimensions (n)**	**Initialization range**	**Mimimum reached** Standard GA$^{(2)}$	**Mimimum reached** SHC	**Minimum reached** GA-GHC
Rastrigin (A=10)	100	$2.56 \le x_i \le 5.12$	0.539	725.8	50.94
Griewank	100	$300.0 \le x_i \le 600.0$	175.2	269.9	0.022
Ackley	100	$16.38 \le x_i \le 32.76$	0.035	21.2	19.8

The GA-GHC optimizer performs better than the stochastic hill climber (SHC) on all three test functions. It exhibits better performance than the standard GA$^{(2)}$ for the Griewank function only, while it underperforms for the Rastrigin and Ackley functions. The standard GA$^{(2)}$ uses parameter sets (p_c and p_m) specifically tuned for each individual problem, whereas the GA-GHC optimizer uses the same parameter assortment for all benchmarks.

As the purpose of GA-GHC optimizer is training neural networks, a test was performed on a benchmark neural network-training problem having neither monotony nor concavity restrictions. This problem, known as the *addition problem* (Rumelhart, 1986), involves adding two 2-bit numbers. The tested version had two 2-bit inputs, 4 hidden nodes, and 3 output nodes with fully connected layers. For this problem, Whitley *et al*. (1989) compared the performances of their GENITOR algorithm to GENESSIS (a classical GA$^{(1)}$ algorithm). GENITOR differs in two major ways from GENESSIS. The first is the explicit use of ranking, an improved reproduction operator. Secondly, GENITOR abandons the generational approach and reproduces new genotypes on an individual basis. The performance measure in this case was the sum of squared errors on all three outputs of the network for the 16 patterns presented. The number of function evaluations was 100 000 in all cases. GENESSIS only reduced error to 5.8 (averaged over 13 runs). GA-GHC optimizer performed noticeably better than GENESSIS, reducing the error to 2.95 (averaged over 5 runs), which approaches the performance of GENITOR, which achieved 2.48 (averaged over 5 runs).

In summary, the GA-GHC optimizer has proven to be more effective than standard GA$^{(1)}$ with binary encoding. It also outperforms stochastic hill-climber, but performs moderately comparably with other improved GAs (standard GAs$^{(2)}$ and GENITOR).

2.5 Methodology validation on liquid holdup modeling

After devising the GA-GHC optimization algorithm and providing the tools for designing neural network models that agree with *a priori* monotonic-concave information, we tackled the real problem of liquid holdup ANN correlation in counter-current packed beds. Recently, Piché *et al*. (2001e), using a wide database (1483 measurements) of total liquid hold-up in counter-current packed beds, developed a dimensionless neural network correlation. This correlation outperforms, in terms of AARE (average absolute relative error) and ARE's standard deviation (STDEV), almost all previous empirical correlations. It is based on a feed-forward neural network, whose inputs are some dimensionless numbers built from the dimensional variables characterizing the G-L-S system, i.e., u_G, u_L, ρ_G, μ_L, a_T, ε, ϕ, Z, D_C, ρ_L, σ_L, μ_G. This approach's difficulty lies in deciding which dimensionless numbers to use as correlation's inputs. Although the inputs selection can be done automatically with GAs (see Chapter 1), the procedure is time-consuming and complex. More importantly, it is unable to guarantee full monotonicity of the networks for the *dimensional* variables $v_s \in$ [$u_G, u_L, \rho_G, \mu_L, a_T, \varepsilon, \phi, Z, D_C, \rho_L, \sigma_L, \mu_G$] because the network inputs, x_i, are functions (dimensionless numbers, N_i) of the dimensional variables v_s.(See proof in Appendix 2).

The same monotonicity rules, Eqs. (1.20)-(1.25), are used, to which we added the following two rules regarding the concavity information:

$$\frac{\partial^2 \varepsilon_L}{\partial u_G{}^2} \geq 0 \tag{2.21}$$

$$\frac{\partial^2 \varepsilon_L}{\partial u_L{}^2} \leq 0 \tag{2.22}$$

All the dimensional variables that were recorded in the database and which were likely to affect liquid holdup were considered model inputs, except for gas density, because the majority of data points recorded in the database were at or near atmospheric pressure. For this problem, the monotonicity-concavity-information matrix, MCI, equivalent to Eqs. (1.20)-(1.25) and (2.21)-(2.22) writes as:

$$MCI = \begin{array}{c} \begin{array}{cccccc} U_G & U_L & \mu_L & \sigma_L & \rho_L & a_T \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & -1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

To determine the best set of weights for the model $\varepsilon_l = f(u_G, u_L, \mu_L, \sigma_L, \rho_L, a_T, \varepsilon, \phi, Z, D_C, \mu_G)$, the global error *GErr(w)* (Eq. (2.10)) was minimized. Ideally, at the end of the optimization, the penalty terms in this criterion would reach zero. The number of hidden nodes was determined by trial and error. It was set to *J=5* yielding 66 weight parameters. The first population of the GA-GHC optimizer was initialized in the range [-1,1], and the signs of the weights were tuned such that the monotony penalty was null. All individuals of the first population obeyed the monotonic condition according to the MCI matrix. A typical run of the GA-GHC optimizer on this problem is shown in Figure 2.4.

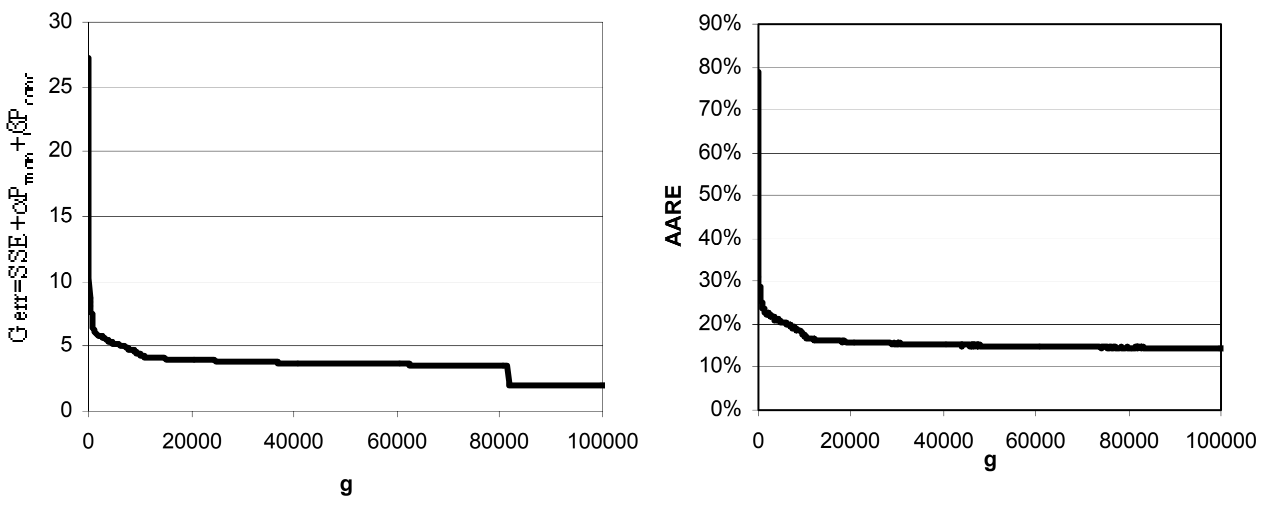


Figure 2.4 Monotonic concave NN training with GA-GHC optimizer (α=2.5,β=1.5)

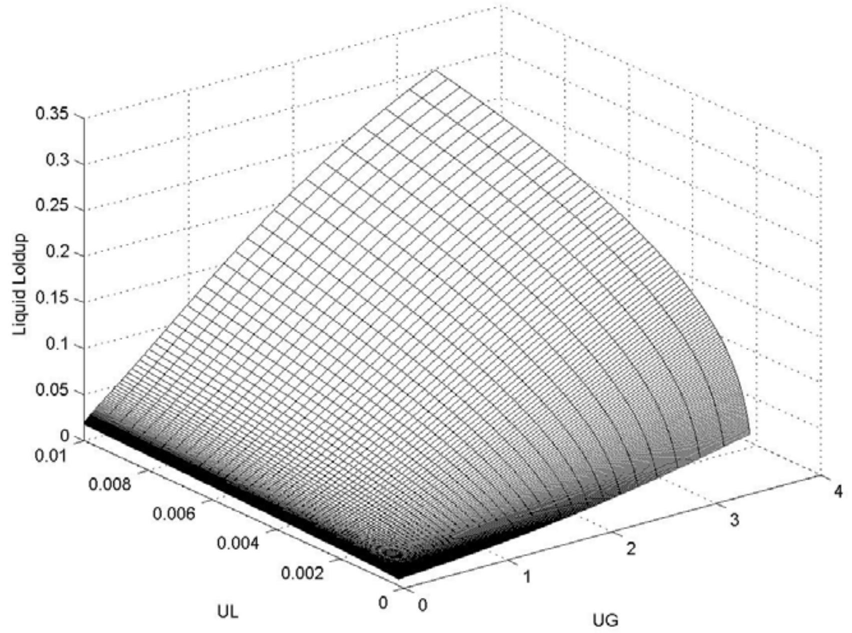
The variables monitored during the generational evolution were *GErr* and *AARE* of the best model in generation *g*. The training was done on 1046 training patterns (the training set, T) which represented 70% of the whole data, while the reminder set (G) was kept to assess generalization of the trained models. After *ca.* 2000 generations, the model *AARE* attained 15%; however, more than 80 000 generations were needed before the penalty term for the second derivative of the model with respect to gas velocity became 0. Restarting the algorithm several times yielded similar results. The best NN model found in the last generation explored by this approach will be referred as the *ε_l monotonic model*.

Classical NN training algorithms may not prevent over-fitting because some data are corrupted or noisy, the data in the input space is sparse, or the phenomenon to be modeled is complex. This is illustrated in Figure 2.5, which compares the ε_l monotonic model output against the output from an NN model trained with BFGS algorithm and early stopping. The two models used the same training data and network configuration.

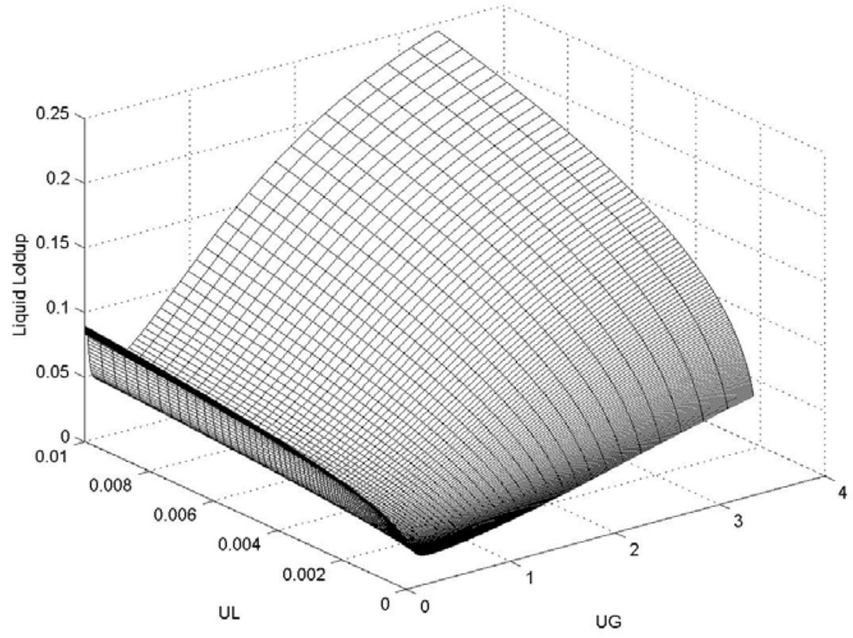
Table 2.5 Comparison between two correlations for liquid hold-up

Model performance or characteristics	**Piché *et al.* (2001e)**	**This work** ***ε_l monotonic model***
AARE $_{T+G}$	13.8%	14.2%
STDEV of ARE $_{T+G}$	14.6%	12.1%
AARE $_{G}$	14.0%	14.2%
STDEV of ARE $_{G}$	16.7%	12.2%
No weights	92	66
Max weight	39.2	9.3
Min weight	-25.15	-12.5

Table 2.5 compares the performances and characteristics of the ε_l monotonic model to the Piché *et al.* (2001e) model built on the same database. Monotonicity was 100% mathematically guaranteed over the whole database space in the ε_l monotonic model, but not in the Piché *et al.* (2001e) model. The *ε_l monotonic model* captured the main tendencies and avoided over-fitting the training samples by not following misleading noisy training instances. Both AARE and STDEV (Table 2.5) were virtually the same on generalization set (G) and all database (T+G). The fact that the *ε_l monotonic model*, obeying Eqs. (1.20)-(1.25) and trained with the GA-GHC optimizer, performed equally well in terms of prediction error on all data means that (i) monotonicity-concavity information imposed during training is truly manifested within the data and is thus *a posteriori* proof of its correctness (ii) the GA-GHC optimizer was robust enough to identify a suitable model.



a)



b)

(μ_L) - (kg/m.s)	σ_L - (N/m)	ρ_L - (kg/m^3)	ε - (-)	a_T - (m^{-1})	ϕ - (-)	Z - (m)	D_C - (m)	(μ_G) - (kg/m.s)
1.82E-03	3.64E-02	9.59E+02	9.77E-01	2.13E+02	7.84E-02	3.00E-01	1.55E-01	1.77E-05

Figure 2.5 Surface implemented by two ANN models a) ε_l monotonic model b) classically trained model. (In both cases, the same training data and network configuration are used. The G-L-S properties for which the outputs were simulated are given in the accompanying table.

Table 2.6 ε_l monotonic model: normalized input and output functions and the corresponding weights (Ranges of applicability in brackets)*

$$H_j = \frac{1}{1+\exp\left(-\sum_{i=1}^{12} w_{ij} U_i\right)}$$

$$1 \le j \le 6 \qquad H_6 = 1$$

$$S = \frac{1}{1+\exp\left(-\sum_{j=1}^{6} w_j H_j\right)}$$

$$U_1 = \frac{\log\left(\frac{u_G}{1.716\times10^{-3}}\right)}{3.293} \quad U_2 = \frac{\log\left(\frac{u_L}{1.859\times10^{-4}}\right)}{2.612} \quad U_3 = \frac{\mu_L - 8.720\times10^{-4}}{4.913\times10^{-2}} \quad U_4 = \frac{\sigma_L - 2.410\times10^{-2}}{4.629\times10^{-1}} \quad U_5 = \frac{\rho_L - 8.154\times10^{2}}{9.555\times10^{3}}$$

$$U_6 = \frac{\varepsilon - 5.130\times10^{-1}}{4.670\times10^{-1}} \quad U_7 = \frac{a_T - 7.500\times10^{1}}{8.600\times10^{2}} \quad U_8 = \frac{\phi - 6.700\times10^{-2}}{5.730\times10^{-1}} \quad U_9 = \frac{Z - 3.000\times10^{-1}}{2.900} \quad U_{10} = \frac{D_C - 4.400\times10^{-2}}{9.560\times10^{-1}}$$

$$U_{11} = \frac{\mu_G - 1.680\times10^{-5}}{1.617\times10^{-5}} \quad U_{12} = 1$$

$$\boxed{\varepsilon_L = 5.354\times10^{-3}\cdot10^{S\cdot1.989}}$$

$$\begin{bmatrix} \varepsilon_L \ge 5.354\times10^{-3} \\ \varepsilon_L \le 5.219\times10^{-1} \end{bmatrix}$$

$$\begin{bmatrix} u_G \ge 1.716\times10^{-3} \\ u_G \le 3.365\times10^{0} \end{bmatrix} \begin{bmatrix} u_L \ge 1.859\times10^{-4} \\ u_L \le 7.616\times10^{-2} \end{bmatrix} \begin{bmatrix} \mu_L \ge 8.720\times10^{-4} \\ \mu_L \le 5.000\times10^{-2} \end{bmatrix} \begin{bmatrix} \sigma_L \ge 2.410\times10^{-2} \\ \sigma_L \le 4.870\times10^{-1} \end{bmatrix} \begin{bmatrix} \rho_L \ge 8.154\times10^{2} \\ \rho_L \le 1.037\times10^{4} \end{bmatrix} \begin{bmatrix} \varepsilon \ge 5.130\times10^{-1} \\ \varepsilon \le 9.800\times10^{-1} \end{bmatrix}$$

$$\begin{bmatrix} a_T \ge 7.500\times10^{1} \\ a_T \le 9.350\times10^{2} \end{bmatrix} \begin{bmatrix} \phi \ge 6.700\times10^{-2} \\ \phi \le 6.400\times10^{-1} \end{bmatrix} \begin{bmatrix} Z \ge 3.000\times10^{-1} \\ Z \le 3.200\times10^{0} \end{bmatrix} \begin{bmatrix} D_C \ge 4.400\times10^{-2} \\ D_C \le 1.000\times10^{0} \end{bmatrix} \begin{bmatrix} \mu_G \ge 1.680\times10^{-5} \\ \mu_G \le 3.297\times10^{-5} \end{bmatrix}$$

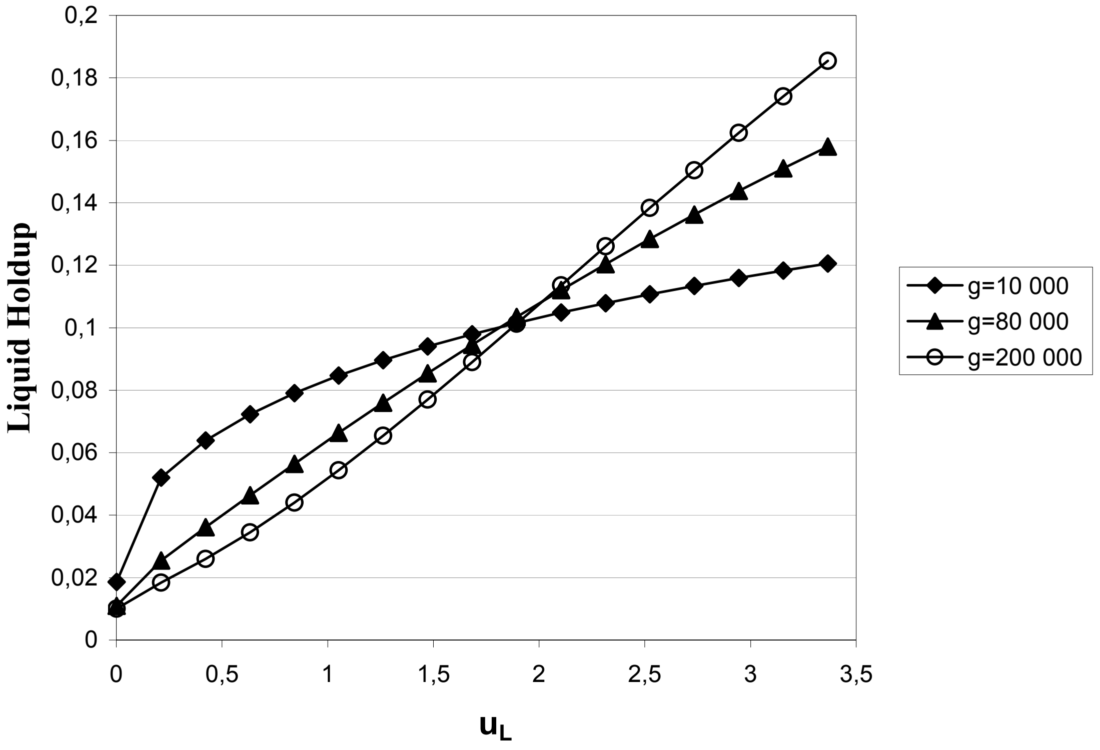
wij	**1**	**2**	**3**	**4**	**5**
1	4.394E+00	1.445E-01	2.353E-06	-1.638E-04	-4.893E-01
2	4.104E-07	1.608E+00	1.462E-04	-3.667E+00	-5.698E+00
3	1.443E-02	1.031E+00	1.594E-04	-3.653E-01	-4.760E+00
4	1.482E-05	1.885E-04	5.377E-01	-2.250E+00	-3.813E-03
5	-8.727E+00	-2.128E+00	-5.440E-03	3.941E-04	3.382E+00
6	-6.724E-01	6.752E-01	-3.026E+00	2.786E-01	9.740E-01
7	9.128E-07	1.237E+00	7.317E+00	-1.061E+00	-9.538E+00
8	-3.481E+00	2.539E+00	4.182E+00	1.948E+00	3.483E+00
9	-4.268E+00	-2.602E+00	9.403E+00	-3.319E+00	1.622E-01
10	2.750E+00	-3.817E-01	7.484E+00	1.677E+00	-4.596E-01
11	-5.130E+00	6.033E+00	-1.743E+00	1.085E+00	-1.249E+01
12	-4.356E+00	-1.768E+00	-2.901E+00	3.331E+00	-7.038E-01

w_j	**1**	**2**	**3**	**4**	**5**	**6**
	9.581E+00	2.980E+00	1.394E+00	-3.117E+00	-5.637E+00	-5.618E-01

*A "user-friendly" spreadsheet of the neural correlation is accessible at: http://www.gch.ulaval.ca/~grandjean or http://www.gch.ulaval.ca/~flarachi

In terms of AARE, the models performed equally (~14%), whereas in terms of ARE dispersion, the ε_l *monotonic model* performed better. The improvement in ARE dispersion was much more important on the generalization data set on which the real performance of the model should be judged (Flexer, 1994).

There were fewer weights in the ε_l *monotonic model:* 66 vs. 92 (Table 2.5). The absolute values of maximum and minimum weights were good indicators of the model's capacity to produce smooth trends. Smooth output is assured by small weights.



U_G - (m/s)	U_L - (m/s)	(μ_L) - (kg/m.s)	σ_L - (N/m)	ρ_L - (kg/m^3)
1.32E+00	2.00E-03	1.82E-03	3.64E-02	9.59E+02

ε - (-)	a_T - (m^{-1})	ϕ - (-)	Z - (m)	D_C - (m)	(μ_G) - (kg/m.s)
9.77E-01	2.13E+02	7.84E-02	3.00E-01	1.55E-01	1.77E-05

Figure 2.6 Evolution of model's concavity with respect to gas velocity during training. The G-L-S properties for which the outputs were simulated are presented in the accompanying table.

Figure 2.6 shows how model concavity with respect to gas superficial velocity evolved during the training cycles. Though an acceptable prediction error for the model was achieved in the early stages, a positive second order derivative of the liquid holdup model with respect to gas velocity was obtained only after 80000 generations. In fact, the sign of this derivative constitutes not only a phenomenological consistency proof, but is also an under-fitting detector.

Table 2.6 contains the ANN normalized input and output functions and the corresponding weights and ranges of applicability of the liquid holdup correlation. A downloadable simulator is available at the following web addresses: http://www.gch.ulaval.ca/~grandjean or http://www.gch.ulaval.ca/~flarachi. A standalone Java application implementing the algorithm is also available. Details on the software we developed are given in Appendix 3.

2. 6 Conclusion

A methodology to build neural network models which fulfill *a priori* information on monotonicity and concavity of the function to be learned was developed and presented in this chapter. The relevance of this approach was threefold:

(i) Accurate ANN models may be obtained by directly using the dimensional variables as network's inputs, suppressing the complication with the selection of the fittest dimensionless numbers.

ii) The network training was performed with a genetic algorithm-genetic hill-climber optimizer, designed to direct the search toward monotonic and concave ANN models in agreement with prescribed knowledge.

(iii) A robust correlation having phenomenological consistency in the entire database was issued for the prediction of liquid hold-up in counter current packed beds.

This chapter presented several original elements. Firstly, second order information (concavity) was added in addition to the monotonicity information. Furthermore, the training of the network was performed with an evolutionary algorithm with two new design elements. One of these elements was the modified arithmetic crossover; the second one was generational switch between a genetic like search and a hill climbing type search. A contribution per-se was the result obtained: the mono-concave liquid holdup prediction model.

2.7 Notation

a_T	Bed specific surface area (m^2/m^3)
c	Number of inputs for which concavity information is available
$conc(\mathbf{w})$	Concavity function for the network represented by **w**
C	Scaling coefficient in linear conversion of *GErr* into fitness
D_C	Column diameter
E_i	Expected number of copies of the specimen *i*
$f(\mathbf{x},\mathbf{w})$	Neural network function
$fitness_i$	Functional fitness value for the individual *i*
$fitness^*_i$	Raw fitness value for the individual *i*
g	Generation in a GA-GHC run
$Gerr(w)$	Global error function minimized by the GA-GHC
H_j	Activation function of the *j'*s neuron in hidden layer
I	Number of inputs in the network
J	Number of nodes in the hidden layer
m	Number of inputs for which monotonicity information is supplied
$mon(\mathbf{w})$	Monotonicity function for the network represented by **w**

MAXPOP	Size of population
MAXSESSIONS	Number of generations to explore
$\boldsymbol{N}$	Vector whose elements are dimensionless groups
N_i	Dimensionless group computed from the dimensional variables $\mathbf{v}$
N_T	Number of pairs $(\mathbf{x}_i, y_i)$ in the training set
p_{Si}	Probability of a particular solution *i*'s survival
p_c	Crossover probability
p_{lm}, p_{hm}	Low, respectively high, probability mutation rates
P_{mon}, P_{conc}	Penalty functions for monotonicity and concavity respectively
R_i	Remainder of the expected number of copies of an individual i
t	Uniform random number between 0 and 1
u	Phase velocity (m/s)
v_i	Dimensional variable describing the properties of the G-L-S system
$w_{i,j}$	Connectivity weight between the input *i* and hidden unit *j*
w_j	Connectivity weight between the hidden unit *j* and the output unit
$\boldsymbol{w}$	String representation of all connectivity weights $w_{i,j}$, w_j
wLow, wHigh	Minimum and maximum values for weights initialization
$\boldsymbol{x}_i$	ith input vector
x_i	ith component of a particular input vector
$\hat{y}(\boldsymbol{x}_i)$	Estimate produced by the network for input sample $\mathbf{x}_i$
y_i	True experimental value for input sample $\mathbf{x}_i$
Z	Bed height

Greek letters

α, β	Weighting coefficients in global error function
Δw_l	Increment to be added to the lth element of the weight string $\mathbf{w}$
ε	Bed porosity
μ	Phase viscosity (kg/m.s)
ρ	Phase density (kg/m^3)
σ	Phase surface tension (N/m)
$\sigma(z)$	Logistic sigmoid function

Abbreviations

ARE	Absolute relative error
AARE	Average absolute relative error
NN	Neural network
G	Gas, generalization
GA	Genetic algorithm
GHC	Genetic hill climber
G+T	Generalization and training
L	Liquid
MCI	Monotonicity-concavity information
PC	Phenomenological Consistence ⇔ matching prior knowledge
PK	Prior Knowledge
S	Solid
STDEV	Standard deviation of ARE
T	Training

3. Data classification in multiphase reactors

The first two chapters dealt with the prediction of continuous characteristics. In such cases, the output of the ANN directly approximated (a normalized form of) the characteristic we wanted to predict based on knowledge of the input variables. In this chapter the focus is on situations in which the characteristic to model is categorical; i.e., it might take a finite number of discrete values. In this case the neural network is designed to approximate the probability of each occurring class, given a particular observed realization of the input vector (posterior probability).

As in the problems of regression, the methodological study was made on some concrete multiphase reactors data sets that we had at our disposal. More specifically, the case studies here are the flow regime classification in trickle bed reactors and the bed initial expansion or contraction in three-phase fluidized beds. In addition to these data sets, other publicly available real data sets or simulated data sets were analyzed. The three primary problematics remain: feature selection, inclusion of prior knowledge, and model design. However, there are some differences in the way these key aspects of modeling are addressed. First, some statistical approaches are considered along with the neural networks. Their use is twofold: i) to help identify relevant features for the classification problem, as they may be lighter in terms of computation required, ii) to give a comparison basis for the neural network approaches. Second, we consider only the dimensional approach; i.e., the features used in the models are the original, untransformed, but scaled variables. Even though these features are less numerous than the possible dimensionless groups from which they can be built, we are still concerned with the feature selection problematic. This problematic is viewed from the statistical pattern recognition point of view, rather than the neural networks perspective. Third, the feature selection step is treated separately from the remaining two problematics, i.e., model design with prior knowledge incorporation. We first determine the most relevant features for predicting the class variable using different relevance criteria, and then search for an accurate, robust classification model.

The material in this chapter is divided into two sections. The relevant bibliographical review will be made for each section respectively.

3.1 Feature selection methods for multiphase reactors data classification

3.1.1 Bibliographical review

Chemical reactor engineering faces the challenging task of extracting knowledge from data, which increasingly becomes more available and accurate. The goal for researchers and engineers is to anticipate and predict the behavior of complex systems such as the multiphase reactors, which still challenge the current physically-based first-principles approaches (Dudukovic *et al.*, 2002). One problem often encountered is accurately identifying the state of a particular system, given prior information in the form of measurable observations. When the state to be predicted takes the form of a *categorical* variable, the problem is known as a *classification* problem.

In spite of several decades of research in the area of pattern recognition dealing with the general classification issue, a *general purpose machine pattern recognizer* remains undesigned. Provided enough data samples are available, this process splits into two steps: *feature selection,* followed by *classifier design*. The first refers to the task of selecting, from among all the candidates, a limited number of features (or variables) which are the most relevant to the classification task. The second refers to the choice and design of a particular inference engine (classifier) than can learn from data and make reliable predictions in new situations. Feature selection is critical in reducing classifier complexity and cost and improving model accuracy, visualization, and comprehensibility of induced concepts.

Suppose that n examples (or instances) ω_k, $k = 1 \dots n$, represented by the input vector $\mathbf{x}_p(\omega_k) = (x_{k,1}, x_{k,2}, \dots, x_{k,p})$ and the label of class $y(\omega_k)$, are available. (Here $y = 1, 2 \dots N_c$, are particular values of the generic class variable denoted Y, N_c being the number of classes). Using this dataset, we may want to design a classifier able to assign the correct class label y for a new instance ω'. Prior to design, a feature selection step is required, as we do not know *a priori* which among the p available features are important in the classification. Selecting only a reduced number d of features among all p, $d<p$, is attractive because the classifier performance depends on the interrelationship between the sample size n used for learning, the number of features $d,$ and the classifier complexity (Jain *et al.*, 2000). As

discussed by Bishop (1995), the number of learning samples needs to be an exponential function of the feature dimension, so there is an obvious interest in keeping d as low as possible.

Selection of a good feature *subset* may be of little interest when the number of training samples is sufficiently large and representative (or equivalently, when the class-conditional probability density functions are fully known). In this case, the probability of misclassification does not increase as the number of features increases (Sebban *et al.*, 2002). In practice, however, added features can degrade the classifier performance when n is relatively small with respect to the number of features. This behavior, known as *peaking*, occurs when for a given sample size (n), supplementary features increase the number of classifier parameters, thereby reducing classifier reliability (Jain *et al*., 2000). In these instances, low-dimensional pattern representations are more advantageous in terms of cost and accuracy. Notwithstanding, excessive reductions in the number of features can alter classifier discrimination power and inflate inaccuracy.

Feature extraction and *feature selection* are the two main methodologies used in dimensionality reduction. The former refers to algorithms generating a reduced number of new features based on transformations of original features, *e.g.*, principal component analysis. The resulting variables usually lack physical sense and are thus not easily interpretable. Feature selection algorithms, on the contrary, target the best d-subset among the available p features without features alteration. For this reason, this is this category of algorithms explored in this multiphase reactor classification problems study.

There exist two generic approaches for feature selection, termed by John *et al*. (1994) as *filter* and *wrapper* techniques. Filter model, through statistical techniques, are indicative of the accuracy of potentially induced classifiers. They "filter out" irrelevant features before the induction process, and are usually fast (absence of training). A common quality criterion in filter models is the Shannon's *mutual information*, $I(Y|\boldsymbol{X}_s)$, which measures the information provided by $\boldsymbol{X}_s$ on the class variable Y (see Shannon and Weaver, 1949; Ash, 1990). In wrapper model, good subsets $\boldsymbol{X}_s$ are searched using the induction algorithm itself where the accuracy rate (*AR)*, estimated by holdout, cross-validation, or bootstrap, is to be maximized. Here, more CPU time is needed, and the solution depends on the particular classifier.

Once an appropriate relevance criterion J is defined, the problem of feature selection can be formulated as follows: given a set of p features, select a subset of size d that maximizes J.

The simplest approach is to examine all C_p^d possible combinations and choose that with the largest J value. But even for moderate p and d values, such an exhaustive search may become impractical. Most currently used methods evaluate only a fraction of combinations, providing speed, but not guaranteeing optimality of solution. A second simple method would be to select the best d individual features as an approximate solution to the feature selection problem. Using the mutual information criterion, Batitti (1994) selected the inputs for a neural network classifier. Inter-feature mutual information was considered for selecting features both informative about the class Y and relatively independent of each other.

Most current feature selection algorithms are sequential methods. The *sequential forward selection* (SFS) is an example. With SFS, the best single feature is first selected. Then, an extra feature is added which, combined with the already selected features, maximizes criterion J. SFS, in conjunction with the mutual information criterion, was implemented by Sridhar *et al*. (1998) to select inputs for neural networks in functions approximation. The main distinction here with respect to the work of Batitti (1994) was the possibility of identifying jointly important features, albeit at the expense of a supplementary computation overhead.

The heuristic basis of most sequential feature selection algorithms is the assumption that the criterion J is monotonic; *i.e.*, any change in feature set size (and therefore feature set information content) is positively correlated with the change in J. If this is true when J is the mutual information, it is not always the case with other criteria, such as the accuracy rate AR (or its complement, the prediction error) (Pudil *et al*., 1994). In this case, sequential selection methods with backtracking such as the "*plus-l-take-away-r*" (or (l,r) search) method (Stearns, 1976) or its improved version, the *sequential floating forward selection* (SFFS) (Pudil *et al*., 1994), are more suitable. These methods first enlarge the feature subset by l features using SFS, and then delete r of them one by one. The feature withdrawn at each step is that which causes the smallest decrease in J. Though computationally more demanding than SFS (because more combinations are being evaluated), such methods are more efficient in conjunction with non-monotonic criteria (Pudil *et al*., 1994).

The single "optimal" technique (optimal only if J is monotonic) is based on the Branch and Bound algorithm (Narendra and Fukunaga, 1977), which avoids an exhaustive search by using intermediate results for obtaining bounds on the final criterion value. Because this algorithm involves evaluating a quantity of possibilities that is still an exponential function of the number of variables (*p*), and because the monotonicity criterion does not hold here, the Branch and Bound was not considered in this study. Other feature search methods, such as those based on genetic algorithms (GA), have been proposed for classification (see Siedlecki and Sklansky, 1988).

An alternative method for determining the most important features for a classification task is the Garson method (1991) of interpreting the weights of neural networks. In this method, a feed-forward neural network is trained to learn the mapping $Y(\boldsymbol{X}_s)$. Then a saliency index for each input of the network is computed. This index is calculated by assessing the share of weights associated with each. This method was experimentally evaluated by Nath *et al.* (1997), who concluded that the method had potential merit.

3.1.2 Study objective and organization

The objective of section 3.1 is to examine the extension of feature selection algorithms in two classification problems relevant to the field of multiphase reactor engineering: flow regime assignment in trickle-bed reactors (LIR, TR, and HIR) and identification of bed initial expansion/contraction (IBE, IBC) in three-phase fluidized-bed reactors. The ability of these methods to provide good solutions (elite subsets $\boldsymbol{Xs}$) that agree with each other was investigated on two benchmark problems: a synthetic problem and the Anderson's iris data classification problem. For these two cases, *a priori* knowledge of the relevance of the features was available. Furthermore, a new feature selection algorithm which mixes filter and wrapper algorithms was devised.

Table 3.1 summarizes the four methods (M-I through M-IV) tested in this work, in order to identify the subset $\boldsymbol{X}_s$ (of size d), instead of the whole feature subset $\boldsymbol{X}_{p,}$ to be used in classification. The method of Garson (1991), referred to as M-V method, is not shown because it shares nothing in common with this classification of methods M-I to M-IV, except perhaps the fact that it needs training of the classifier (common to wrappers).

Table 3.1 Feature selection strategies

Criterion to maximize (J) / **Selection method**	**Mutual information (information theory)** $J=I(Y\|\boldsymbol{X}_s)$	**Accuracy rate of a 1-NN classifier** $AR(1\text{-}NN)$	$I(Y\|\boldsymbol{X}_s)$ **and** $AR(1\text{-}NN)$
Sequential Forward Selection (SFS)	**Yes** (M-I) (filter method)	**Yes** (M-II) (wrapper method)	**Yes** (M-IV) SFS with $I(Y\|\boldsymbol{X}_s)$ continued by (l,r) search with $AR(1\text{-}NN)$ (filter-wrapper method)
plus-*l*-take-away-*r*	**No** (Not necessarily justified, as J is monotonic)	**Yes** (M-III) (wrapper method)	

A "Yes" entry in Table 3.1 means that the selection algorithm specified by the row header in conjunction with the criterion specified by the column header is tested.

The classifier used to assess the importance of sets in methods M-II to M-IV is the one-nearest neighbor (1-NN) classifier. Its performance is evaluated by five-fold cross validation. Inherent details unfold in the following sections. A further extension of this work would be to build simple, reliable, and interpretable classifiers using as input variables the solutions $\boldsymbol{X}_s$ found within this study. We could provide a new brand of design tools as an alternative to the existing first-principle based models that are still unsatisfactory. This constitutes the scope of the section 3.2.

3.1.3 Relevance assessment

A feature selection algorithm decides to retain or drop features based on their *relevance*. There are several definitions of relevance, each addressing the question "relevant to what?" (e.g. John et *al.*, 1994). Here, the relevance of a feature subset $\boldsymbol{X}_s$ in predicting the class variable Y is judged using three measures:

a) Mutual information $I(Y|\boldsymbol{X}_s)$ between the feature vector $\boldsymbol{X}_s$ and the class variable to be predicted Y

b) Accuracy rate of a 1-NN classifier *AR(1-NN)* that uses $\boldsymbol{X}_s$ as discriminatory features

c) The saliency index of Garson (1991).

Note that the a) and b) relevance measures, in conjunction with a selection algorithm such as SFS or (l,r) search, identify the *d*-subset $\boldsymbol{X}_s$ to use for predicting class membership without sacrificing the discrimination power involving all *p* features. The Garson's saliency index, on the other hand, judges only the relevancy of each feature if the whole $\boldsymbol{X}_p$ set is used as a neural net input. This mostly provides a relevance order for all features in $\boldsymbol{X}_p$, *i.e.*, features-ranking rather than indications on which $\boldsymbol{X}_s$ subset is the most pertinent for classification.

3.1.3.1 Mutual information

Recall first that the records ω_k have an input vector, from which is selected a features subset $\boldsymbol{x}_s(\omega_k)=(x_{k,1},x_{k,2},\ldots,x_{k,d})$ of cardinality *d*, and a (class membership) label $y(\omega_k)$. A classifier that uses $\boldsymbol{X}_s$ to predict the class *Y* decreases its initial *uncertainty* by using the *information* in the features of $\boldsymbol{X}_s$. Due to insufficient input information or sub-optimal operation of the classifier, the uncertainty about the class cannot be decreased to zero. Shannon's information theory (Shannon and Weaver, 1949; Ash, 1990) gives a suitable formula for quantifying these concepts.

If the probability that the class variable *Y* takes a particular value *y* is denoted with *P(y)*, $y=1,2\ldots N_c$, the initial uncertainty in the output class variable is given by the *entropy*:

$$H(Y) = -\sum_{y=1}^{Nc} P(y)\cdot \log P(y) \tag{3.1}$$

Practically, the probability *P(y)* can be estimated using the occurrence frequency of *y*:

$$P(y) = \frac{n_y}{n} \tag{3.2}$$

where n_y is the number of occurrences of *y*, and *n* is the total number of samples.

The entropy of the features vector $\boldsymbol{X}_s$ can be similarly estimated. Since features are continuous variables, the sum is replaced by an integral:

$$H(\mathbf{X}_s) = -\int P(\mathbf{X}_s) \cdot \log P(\mathbf{X}_s) d\mathbf{X}_s \tag{3.3}$$

The simplest way to estimate *P(X$_s$)* is by using histograms. First, each feature *X$_s$*, among all *d* constituting ***X$_s$***, is discretized into a large number of intervals, *nbx.* For simplicity, the same number of intervals is used for each feature. The hypercubes with the volume $dV = dX_{s,1} \times dX_{s,2} ... \times dX_{s,d}$ are called bins. Bins construction is exemplified in Figure 3.1 for a 2-D set of features.

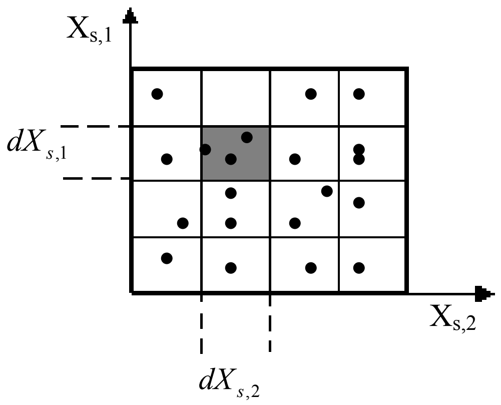


Figure 3.1 Bins construction: Suppose a 2-feature vector *X$_s$*. Here a bin is the volume $dV = dX_{s,1} \times dX_{s,2}$.

Consider now each of the *nbx^d* bins, and count how many samples, among all *n*, fall into each bin. For all bins, *b=1... nbx^d* probabilities $P(\mathbf{X}_{s \subset b}) = \frac{n_b}{n}$ (n_b = number of samples falling in bin *b*) of ***X$_s$*** occurring in a particular bin *b* are evaluated. The entropy *H(**X$_s$**)* is computed using a discretized form of Eq. (3.3):

$$H(\mathbf{X}_s) = -\sum_{b=1}^{nbx^d} P(\mathbf{X}_{s \subset b}) \cdot \log P(\mathbf{X}_{s \subset b}) \tag{3.4}$$

The average uncertainty on *Y* after knowing the feature vector ***X$_s$*** with *d* components is the *conditional* entropy $H(Y \mid \mathbf{X}_s)$:

$$H(Y \mid \mathbf{X}_s) = H(\mathbf{X}_s, Y) - H(\mathbf{X}_s) \tag{3.5}$$

where $H(\mathbf{X}_s, Y)$ is the *joint entropy* estimated using a similar box counting procedure:

$$H(\mathbf{X}_s, Y) = -\sum_{b=1}^{nbx^d} \sum_{y=1}^{N_c} P(\mathbf{X}_{s \subset b}, y) \cdot \log P(\mathbf{x}_{s \subset b}, y) \tag{3.6}$$

in which $P(\mathbf{X}_{s \subset b}, y)$ is the joint probability that $\boldsymbol{X_s}$ belongs to bin b and Y takes the value y.

By definition, the amount by which the uncertainty is decreased is the mutual information $I(Y|\boldsymbol{X_s})$ between variables Y and $\boldsymbol{X_s}$ (Batitti, 1994):

$$I(Y \mid \mathbf{X}_s) = H(Y) - H(Y \mid \mathbf{X}_s) \tag{3.7}$$

This function, symmetric with respect to Y and $\boldsymbol{X_s}$, can be reduced to:

$$I(Y \mid \mathbf{X}_s) = I(\mathbf{X}_s \mid Y) = \sum_{y=1}^{N_c} \int P(Y, \mathbf{X}_s) \log \frac{P(Y, \mathbf{X}_s)}{P(Y) \cdot P(\mathbf{X}_s)} d\mathbf{X}_s \tag{3.8}$$

The uncertainty $H(Y,X_s)$ in the combined events (Y,X_s) is usually less than the sum of the individual uncertainties $H(Y)$ and $H(X_s)$. Using Eqs. (3.7) and (3.5), one obtains a symmetric function:

$$I(Y \mid \mathbf{X}_s) = H(Y) + H(\mathbf{X}_s) - H(Y, \mathbf{X}_s) \tag{3.9}$$

In function approximations, Sridhar *et al.* (1998) derived an *asymmetric dependency coefficient* by dividing Eq. (3.9) r.h.s. by *H(Y)*. We adopted this normalization here. In these circumstances $I(Y \mid \mathbf{X}_s) = 0$ means that $\boldsymbol{X_s}$ contains no useful information about the class Y, whereas $I(Y \mid \mathbf{X}_s) = 1$ means that Y is completely predictable if $\boldsymbol{X_s}$ is known. In practice, however, the value of $I(Y \mid \mathbf{X}_s)$ also depends on grid coarseness (or number of bins). Coarser grids probably inflate inaccuracy of $I(Y \mid \mathbf{X}_s)$ because important functional variations might be overlooked, while finer grids often overestimate $I(Y \mid \mathbf{X}_s)$ by counting noise as meaningful functional variation.

3.1.3.2 1-NN classifier accuracy rate

The nearest neighbor classifier is one of the simplest methods used to perform non-parametric general-purpose classification. Proven to give accurate results on many pattern recognition problems (Jain *et al*., 2000), it can be represented by the following decision rule: *assign a new pattern to the class of its nearest example in the training set as measured by a metric* (usually Euclidian) *distance*. The Euclidian distance between two points **a** and **b** is simply:

$$d_E(\mathbf{a},\mathbf{b}) = \|\mathbf{a}-\mathbf{b}\| = \sqrt{\sum_{i=1}^{d}(a_i - b_i)^2} \tag{3.10}$$

As it requires no training, the nearest neighbor classifier was used in this study to assess the capability of a subset $\boldsymbol{X}_s$ drawn from the larger $\boldsymbol{X}_p$ set to predict the class Y.

One method used to estimate the accuracy rate AR of a classifier is to compute its confusion matrix on several z-fold cross-validation sets (Kohavi, 1995 for cross-validation issues). Consider a set A of records $\omega_k,\ k=1...n$, available for the classification task. Each record ω_k is represented by the feature vector $\boldsymbol{x}_{s(}\omega_{k)}$ and label $y(\omega_k)$. Let set A be partitioned in say z = 5 folds. Each fold, once per time, is set aside as a *test* set, while the ensemble of remaining z-1 sets are taken as *training* set. For each test set, the confusion matrix of size $(N_c \times N_c)$ synthesizes the predictions of the classifier in the form:

$$\overset{\Pr edicted}{\underset{Actual}{}}\begin{pmatrix} 80 & 10 & 10 \\ 10 & 75 & 15 \\ 5 & 5 & 90 \end{pmatrix}.$$

For illustration here, a three-class classification problem in which there were 100 samples in each class in the test set was considered. Each cell of the confusion matrix indicated how many of the 100 samples were assigned to class y, labeled column-wise. The actual (true) class index is indicated by row. Note that the sum over each row is equal to 100. For convenience, each element in a row of the confusion matrix can be normalized by the number of samples in the class indicated by the row's index. As there are z test sets, we computed the mean confusion matrix, as well as the standard deviation, over the z sets. The elements lying along the main diagonal of the confusion matrix provided the per-class accuracy rate. Their averaged value is the global accuracy rate AR of the classifier.

3.1.3.3 Garson's saliency index

The third criterion to be implemented was the saliency index S_{ind} (Garson, 1991), which determines the importance of the candidate features by interpreting the weights of feed-forward neural networks (Figure 3.2) trained with back-propagation algorithms (Rumelhart *et al.*, 1986). The Garson method determines which input nodes (and thus features) are responsible for most of the output changes.

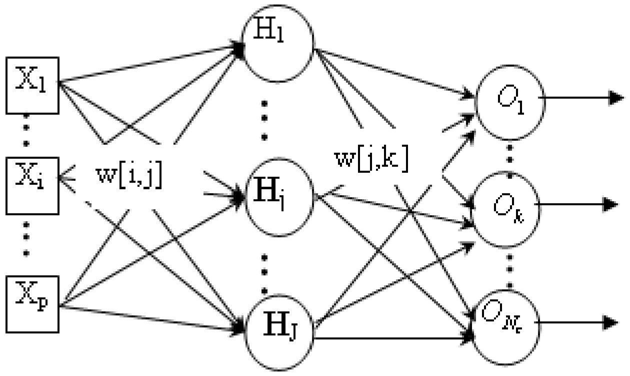


Figure 3.2 Feed-forward neural network used for classification involving N_c classes

All the features in the set X_p are fed into the network by the input nodes, which preprocess the data by normalizing it to the [0,1] interval. The J hidden units perform a nonlinear projection of the feature space, and the output layer decides into which class a particular input point is assigned. The number of output nodes equals the number of classes. For a 2-class problem, a single network output suffices. The parameters w[i,j] and w[j,k] on the inter-layer connections are the network weights. Network training is meant to optimize such weights in a way that when a particular record ω, belonging to the class k, is presented as input, the network output vector $[O_1,O_2,\ldots O_{Nc}]$ shows (nearly) zero elements except for the k^{th} component which will be 1 (or near to).

Saliency $S_{ind}(i,k)$ of input *i* with respect to output *k* is estimated as follows: First, each hidden-to output weight *w[j,k]* is incorporated into the input-to hidden weights *w[i,j]*. Then, for each input variable, a summation is made over all hidden units and converted into a percentage of the total for all input nodes:

$$\mathrm{S}_{ind}(i,k) = \frac{\sum_{j=1}^{J}\left(\frac{|w[i,j]|}{\sum_{i=1}^{p}|w[i,j]|}\times|w[j,k]|\right)}{\sum_{i=1}^{p}\sum_{j=1}^{J}\left(\frac{|w[i,j]|}{\sum_{i=1}^{p}|w[i,j]|}\times|w[j,k]|\right)} \tag{3.11}$$

where *i=1,2,..., p* input variables; *j=1, 2,...J* sweeps the hidden nodes and |.| means absolute value.

In Garson's work, however, situations in which there are more than two classes (and thus more than one output node) were not treated. We proposed, therefore, to generalize the Eq. (3.11) to cases where there were N_c classes. We computed therefore, an overall saliency $S_{ind}(i)$ for each input node and multiple-output neural networks ($k=1, 2, ..., N_c$) as:

$$S_{ind}(i) = 1/N_c \cdot \sum_{k=1}^{N_c} S_{ind}(i,k) \tag{3.12}$$

The logic behind Eq. (3.12) is that the importance of an input variable should be judged by its impact on the whole output layer.

3.1.4 Feature selection methods

Let us now present the methods for solving the following feature selection problem:

Given a set $\mathbf{X}_p$ of p features, select subset $\mathbf{X}_s$ of size d, $d \le p$, that maximizes the relevance criterion J.

Of techniques introduced in the beginning of section 3.1.1, we shall focus only on the class of sequential methods: sequential forward selection (SFS), sequential backward selection (SBS), and (*l,r*) method that were effectively implemented in this work.

SFS and SBS are step-optimal because only the best feature is always added. A limitation inherent to these two methods is their inability with the nested feature combinations to correct decisions in later steps, creating therefore sub-optimal subsets with lower *J*. Stearns (1976) combined SFS and SBS to avoid the nesting effect, bringing thus a net improvement to the sequential methods. This combined method, referred to as (*l,r*) method, repeatedly applies SFS *l* times, followed by *r* SBS steps, until the required number of features is reached. The (*l,r*)-search algorithm, or its particular cases (1,0)-search (or SFS) and (0,1)-search (or SBS), as described by Pudil *et al.* (1994), is detailed in Appendix 4. The termination criterion (Appendix 4) is based on *a priori* knowledge of *d*, *i.e.*, size of best subset. As in our problems, *p* is not large, so termination is set for $d = p$. The best $\mathbf{X}_{s,d}$ subset is retained according to either modality: a) mutual information based criterion *J*: *d* is chosen which yields $1 - J(\mathbf{X}_{s,d}) \leq \zeta$, with ζ a very close to zero threshold value; b) *AR (1-NN)* based criterion *J*: *d* is chosen as $d = \arg\max_i J(\mathbf{X}_{s,i})$.

The (*l,r*) method was further improved by automatically tuning *l* and *r* values by Pudil *et al.* (1994). Their so-called sequential floating forward selection (SFFS) consists in applying a number of backward steps after each forward step, provided the resulting subsets are better than the previously evaluated ones for the same size *d*. Using the mutual information as relevance criterion, SFS promises $I(Y \mid \mathbf{X}_s)$ will always increase through adding supplementary features. This relevance criterion, combined with SFS, has been used to identify neural net inputs (Sridhar *et al*., 1998), albeit not in classification problems. Earlier work sketching the use of $I(Y \mid \mathbf{X}_s)$ to select inputs for neural network classifiers used mutual information between the individual features $X_{s,i}$ and the class *Y*, to assess the relevance of the whole subset $\boldsymbol{X}_s$ (Battiti, 1994). However, that approach did not promote jointly relevant variables as we have here.

Conversely, if *AR(1-NN),* which is a non-monotonic criterion, is used as a relevance measure, the (l,r) search is more appropriate, since it is able to reconsider previous decisions. The use of $I(Y \mid \mathbf{X}_s)$ enabled us to find the subset $\boldsymbol{X}_s$ that yielded about the same information about the class *Y* as the entire set $\boldsymbol{X}_p$. But $I(Y \mid \mathbf{X}_s)$ is a statistical measure of the *general dependency* between $\boldsymbol{X}_s$ and *Y*, and not the classifier accuracy itself. Therefore, it does not guarantee that the resulting $\boldsymbol{X}_s$ will actually be paralleled by the best accuracy rate of the1-NN classifier. However, the mutual information filter criterion is faster in execution than a classifier training session because it does not require iterative computation on the dataset.

As $I(Y\,|\,\mathbf{X}_s)$ evaluates the intrinsic properties of the data, rather than interactions with a particular classifier, the result should exhibit more generality; *i.e.*, the solution will be "good" for several types of classifiers. On the contrary, using *AR(1-NN)* relevance criterion is more computationally intensive and rests on the capability of this particular classifier to exploit those features in the classification task. However, the wrapper criteria possess a mechanism to avoid overfitting; they also generally achieve better recognition rates than do filters, since they are tuned to the specific interactions between the classifier and the dataset.

Hence, M-IV method (Table 3.1) was devised as an alternative in this work. It first selects an initial set of predictive features $\boldsymbol{X}_{s,initial}$ identified by SFS in conjunction with $I(Y\,|\,\mathbf{X}_s)$. The M-IV method then "grows up" and "prunes down" this set with a (*l,r*) search in conjunction with *AR(1-NN)*. First This method is faster than launching an (*l,r*) search in conjunction with *AR(1-NN)* starting with an empty set. This is because $I(Y\,|\,\mathbf{X}_s)$ is easier to compute than *AR(1-NN)* and because SFS is more rapid than an (*l,r*) search. The further adjustments of the resulting selection $\boldsymbol{X}_{s,initial}$ allow deletion or addition of some features which appear to be respectively less or more important for the nearest neighbor classification rule.

3.1.5 Problems and datasets description

The four problems tested are presented next to i) compare the feature selection methods (M-I to M-IV) and judge their efficiency to identify the relevant features ii) identify the most predictive variables in a couple of actual multiphase reactor problems, namely flow regime classification in trickle beds and initial bed expansion/contraction classification in three-phase fluidized beds.

3.1.5.1 Synthetic problem

A synthetic domain problem, in which the correct answer is known *a priori,* was built. The setup is: Generate three sets of 100 10-dimensional data points. In each set, the ten variables are random normally distributed. The mean for each feature in each class (c_1 to c_3) was set as shown in Table 3.2

Table 3.2 Candidate features in a synthetic problem

Var No.	**Mean for class c_1**	**Mean for class c_2**	**Mean for class c_3**	**Relevance rank**
1	3	0	-3	1
2	4	0	0	2
3	0	0	4	2
4	1	0	0	3
5	0	0	1	3
6	1	0	1	3
7	0	0	0	4 –irrelevant
8	0	0	0	4 –irrelevant
9	0	0	0	4 –irrelevant
10	0	0	0	4 –irrelevant

The single most relevant variable is 1, as the average interclass distance between the central points of the normal distributions was the largest for this variable. The two most important features are 1 and 3 (according to the same class separability measure), and the best three features are {1,3,2}. The variables 4, 5, 6 were equally important to the classification task but are far less important than the previous variables. Features 7-10 are irrelevant to the classification task, as they are normally distributed random variables with the same 0 central value for each class. This problem was considered only for benchmarking purposes. We expect that the features {1,3,2} will successfully be identified by the feature selection methods M-I to M-V. A similar setup was used by Nath *et al.* (1997) to evaluate the efficacy of Garson's method when ranking features in a classification problem. However, the setup we chose here is more realistic, as we introduced redundant features in the set, and the overlap between classes was higher.

3.1.5.2 Anderson's iris data

Anderson (1935) famous iris data is a collection of 150 4-dimensional data points, each falling into one of the three classes: Setossa (Se), Versicolor (Ve), or Virginica (Vi). In each class there are 50 data points. The independent variables considered as candidates are shown in Table 3.3

Table 3.3 Candidate features for the iris data classification

No	Variable Name	Symbol	max	min
1	Sepal Length	SL	7.9	4.3
2	Sepal Width	SW	4.4	2
3	Petal Length	PL	6.9	1
4	Petal Width	PW	2.5	0.1

The class Se is easily separable from the other two, which are overlapping classes. Features 3 and 4 can jointly play the roles of features 1 and 2, as shown by Li *et al*. (2002). Using a neuro-fuzzy classifier, they were able to classify data and simultaneously reveal its important features. For this classifier, only the features {3,4} brought the same prediction accuracy as variables 1 to 4. This does not imply that this is incontestably the best unique subset of features for all other types of classifiers as well. Batitti (1994), for example, showed that subset {1, 3} (or {1, 4}) is the best in terms of information content and for a multilayer feed-forward neural network classifier. It is expected, thus, that the feature selection methods tested will point to one of these answers: {3,4} {1,3} or {1,4}.

3.1.5.3 Three-class flow regimes classification in trickle beds

The first real classification problem in the domain of multiphase reactors concerns a trickle-bed reactor in which the gas (G) and liquid (L) are flowing concurrently downwards throughout a bed of catalytic solid (S) particles. The efficiency of such a device is highly dependent on the prevailing flow regime in the reactor for a given set of operational conditions (Dudukovic *et al*., 2002). Depending on the level of interaction between fluids, one may generally distinguish three flow regimes: low interaction regime (LIR), transitional regime (TR), and high interaction regime (HIR). The flow regime, therefore, is the class variable Y which takes particular values $y = 1, 2 \ldots N_c$. Here, $y = 1$ for LIR, $y = 2$ for TR, and $y = 3$ for HIR, while $N_c = 3$. The type of flow regime is often determined by the available p features that characterize the three phases (G-L-S), *e.g.*, porosity, sphericity, gas density, liquid viscosity, fluids' superficial velocity, etc., which are contained in $\boldsymbol{X}_p = (X_1, X_2, \ldots, X_p)$. The assignment of class label (LIR, TR, and HIR) to a particular operating point was based on the visual observation of the experimenter.

2809 flow regime data points (945 LIR, 945 TR., and 919 HIR) were extracted from the Laval University comprehensive trickle bed reactor database (Larachi *et al.*, 1999). The independent variables considered as candidates are summarized in Table 3.4 along with the span over which the measurements were available.

Table 3.4 Candidate features for flow regime class prediction in trickle beds

No	Variable Name	Symbol	Max	Min
1	Liquid superficial velocity (m/s)	u_L	1.74E-01	4.36E-04
2	Gas superficial velocity (m/s)	u_G	3.74E+00	4.98E-04
3	Foaming property (-)	Foam.*	1	0
4	Column diameter (m)	D_c	5.10E-01	2.30E-02
5	Bed porosity (-)	ε	7.40E-01	2.63E-01
6	Grain specific area (m^{-1})	a_G	5.16E+03	4.67E+02
7	Bed specific area (m^{-1})	a_T	3.81E+03	2.78E+02
8	Effective particle diameter (m)	d_p	1.28E-02	1.16E-03
9	Sphericity (-)	ϕ	1.00E+00	3.36E-01
10	Liquid density (kg/m^3)	ρ_L	1.18E+03	6.50E+02
11	Liquid viscosity (Pa.s)	μ_L	6.63E-02	3.10E-04
12	Surface tension (N/m)	σ_L	7.62E-02	1.90E-02
13	Gas density (kg/m^3)	ρ_G	1.16E+02	1.60E-01
14	Gas viscosity (Pa.s)	μ_G	1.97E-05	1.45E-05

(*) Foaming property is a categorical variable. 0=coalescing, 1=foaming

In the working database, the variables were normalized to fall between 0 and 1. For variables covering more than 2 decades, log values were used. The classes to be predicted were low interaction regime (LIR), transition regime (TR), and high interaction regime (HIR).

3.1.5.4 Two-class bed expansion/contraction in three phase fluidized beds

The second problem refers to the initial bed expansion (IBE) or contraction (IBC) in a gas-liquid-solid fluidized bed. This phenomenon occurs upon introduction of a tiny gas flow rate in a liquid fluidized bed and is an important indicator of the bubble wake activity and bubble size. Large bubbles are associated with large wakes that suck liquid into their structures, thereby inducing liquid starvation in the emulsion phase and making the bed contract. On the contrary, small bubbles are associated with small (or no) wakes that barely affect the emulsion liquid; the bed thus expands smoothly with further increasing gas throughputs. Similarly, the class variable Y referring to IBE ($y = 1$) or IBC ($y = 2$) may depend on the p features characterizing the three phases (G-L-S) grouped in input vectors $\boldsymbol{X_p}$ for which we have measurements.

Our goal here is to identify the features that help determine whether *initial* expansion or contraction of the bed will occur. The porosity dataset was extracted from the Laval university three-phase fluidization database (Larachi *et al.*, 2001) after analyzing the behavior of several porosity series at constant liquid velocities u_L and increasing gas velocities u_G. From each series, the observation corresponding to the smallest u_G was retained. The class was considered expansion (IBE) if bed porosity increased with respect to u_G in the initial liquid-fluidized state, or, conversely, contraction (IBC) if bed porosity decreased. As expansion data points were about thrice the contraction ones, two replicates of the contraction points were made to keep about the same number of samples for each class.

Table 3.5 Candidate features for the bed contraction-expansion in fluidized beds

No	Variable Name	Symbol	Max	Min
1	Liquid velocity (m/s)	u_L	2.60E-01	1.09E-03
2	Liquid density (kg/m^3)	ρ_L	1.62E+03	7.78E+02
3	Liquid viscosity (Pa.s)	μ_L	7.19E-02	7.16E-04
4	Surface tension (N/m)	σ_L	7.59E-02	2.48E-02
5	Solid density (kg/m^3)	ρ_s	2.90E+03	1.07E+03
6	Effective particle diameter (m)	d_p	1.54E-02	6.50E-04
7	Terminal velocity (m/s)	u_t	7.84E-01	4.32E-02
8	Column diameter (m)	D_c	2.18E-01	4.00E-02
9	Bed height at rest (m)	H_0	6.00E+00	5.08E-02
10	Foaming property (-)	Foam.*	1	0

(*) Foaming property is a categorical variable. 0=coalescing, 1=foaming

The dataset counted 401 contraction points and 414 expansion points. The candidate features are summarized in Table 3.5.

3.1.6 Results

In this section are the results obtained by applying feature selection methods M-I through M-IV and feature ranking method M-V to the four problems described above. For each problem, we have provided solutions found using the different approaches: 1) which $\boldsymbol{X}_s$ subset produces about the same discrimination power as the whole $\boldsymbol{X}_p$ set (for M-I to M-IV), 2) what is the rank of variables importance (M-V).

3.1.6.1 Synthetic problem

As explained in § 3.1.5.1, we expected the features subset {1,3,2} to be identified. First, SFS with mutual information as a relevance criterion (method M-I) was used. Figure 3.3 shows the result obtained by applying this filter method on the synthetic problem. The number of divisions in the domain for each feature was *nbx=10*. The "+"sign before feature label indicates that the feature was added to the combination at the corresponding epoch.

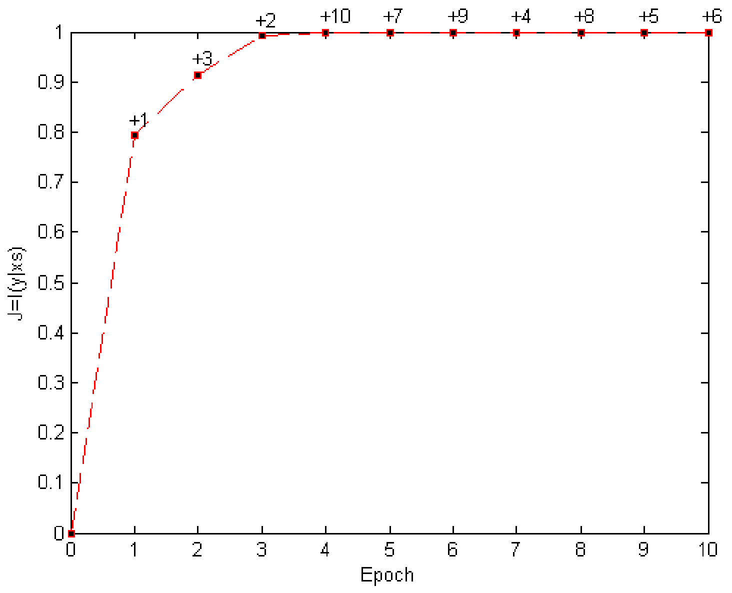


Figure 3.3 Sequential forward selection with mutual information as relevance criterion (M-I). "+" means that the corresponding feature was added to the current subset.

As expected the first selected features were {1,3,2}, which contained almost all information available in the set of features $\boldsymbol{X}_p$={1,2,…,10}. Once feature 2 was added into the combination, further enlargement of the feature set yielded no significant increase in the relevance criterion *J*. Applying SFS with the alternate relevance criterion *AR(1-NN)* (method M-II) induced the same order of preference for the first three variables, features set {1,3,2}. See Figure 3.4. After adding the third variable, the accuracy rate did not increase significantly; on the contrary, it started decreasing after the sixth epoch.

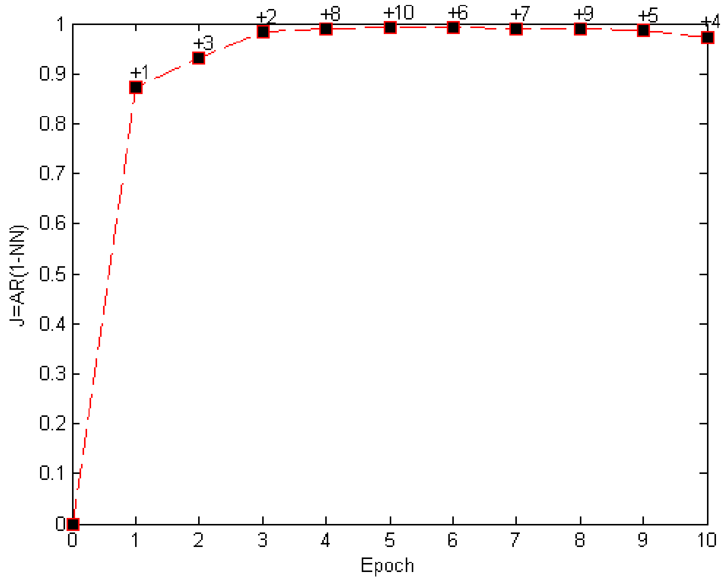


Figure 3.4 Sequential forward selection with accuracy rate as relevance criterion (M-II). "+" means that the corresponding feature was added to current subset.

Method M-III consisting of (*l,r*) search with *AR(1-NN)* as relevance criterion was also tested. In this work, *l=2* and *r=1* were chosen, as they required the minimum computational effort. At each step, two features were added and one was removed. The suggested selection subset $\boldsymbol{X}_s$ was also {1,3,2}. This example was therefore too simple to show any difference in the searching power between SFS and (l,r) search. The filter/wrapper approach (method M-IV) yielded the same solution as M-III, but with less computation effort.

The saliency index values calculated with Eq. (3.12) (method M-V) are shown in Figure 3.5.

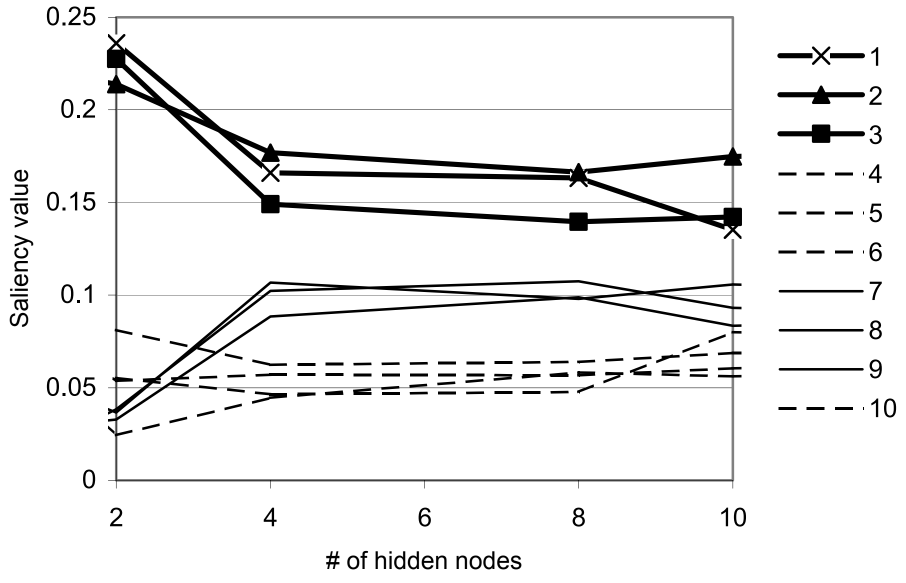


Figure 3.5 Saliency values for the synthetic problem (M-V)

Here the mean saliency values over 10 distinct ANN training sessions for all 1 to 10 features and different numbers of hidden nodes were computed. At each training session, 4000 iterations of back-propagation with adaptive learning rate and momentum were used for a feed-forward neural network with sigmoid transfer functions in hidden and output neurons using 75% of the 300 available points. For a number of hidden nodes between 2 and 8, the accuracy rate *AR(ANN)* of the neural network classifier evaluated on the generalization set (remaining data 25%) approached 98%. It may therefore be concluded that the network was not overfitting the training samples when the number of hidden nodes was less than 8, nor was it underfitting when the number of hidden nodes was equal or more than 2.

As seen in Figure 3.5, the saliency values for the first three features clearly outperformed the others, denoting their importance in the classification process. However, saliency index calculation does not help to confidently rank them. Furthermore, the other somehow significant features, 4 to 6, appeared less relevant than the group of completely irrelevant features 7 to 9. We may thus conclude that the Garson method can indicate the rank only if differences in the importance of variables is large.

The synthesis results of this problem are sketched in Table 3.6. To conclude, all methods produced the same result.

Table 3.6 Summary of methods M-I to M-V for the synthetic problem

Selection strategy	**#of var.**	**Order**	**AR (1-NN) (%)**
None (all available features considered)	10	NA	97.33
M-I: Forward selection with *I(Y|$\boldsymbol{X}_s$)*. (*nbx=10*)	3	**1 3 2** (4 5 6 7 8 9 10)*	98.33
M-II: Forward selection with AR(1-NN).	3	**1 3 2** (4 5 6 7 8 9 10)	98.33
M-III: (l,r) search with AR(1-NN).	3	**1 3 2** (4 5 6 7 8 9 10)	98.33
M-IV: Forward with *I(Y|$\boldsymbol{X}_s$)* continued with (l,r) search with AR(1-NN).	3	**1 3 2** (4 5 6 7 8 9 10)	98.33
M-V: Garson's saliency values through ANN	NA	(1 2 3) (4 5 6 7 8 9 10)	NA

()* brackets here denote that the features inside cannot be confidently ranked. NA means not available.

3.1.6.2 Anderson's iris data

The filter method M-I was first applied for this problem. The number of divisions in the domain for each feature was *nbx=20*. The selected features were {4,3} in agreement with the conclusions of Li *et al*. (2002). Methods M-II through M-IV produced exactly the same results, which is even better performance than if all features were used. The Garson method identified feature 3 as most important and feature 1 as the least relevant, while the importance of features 2 and 4 remained undetermined due to their inconclusive saliency values. For 4 to 10 hidden neurons, the generalization accuracy rate of the network was almost constant and approached 95%, denoting well-trained networks.

Table 3.7 Methods M-I to M-V compared on the iris data classification problem

Method	**# of var.**	**Order**	**AR (1-NN) (%)**
None (all available features considered)	4	NA	93.52
M-I: Forward selection with $I(Y\|\boldsymbol{X}_s)$. (*nbx=20*)	2	**4 3** (1 2)	94.12
M-II: Forward selection with AR(1-NN).	2	**4 3** (1 2)	94.12
M-III: (l,r) search with AR(1-NN).	2	**4 3** (1 2)	94.12
M-IV: Forward with $I(Y\|\boldsymbol{X}_s)$ continued with (l,r) search with AR(1-NN).	2	**4 3** (1 2)	94.12
M-V: Garson's saliency values through ANN	NA	3 (4 2) 1	NA

()* brackets here denote that the features inside cannot be confidently ranked with the respective feature selection method. NA means not available.

3.1.6.3 Three-class flow regimes classification in trickle beds

For this problem, we wanted to determine which variables among those listed in Table 3.4 were most likely to be predictive for the flow regimes: LIR, TR, and HIR. The summary of the analysis of the different methods is given in Table 3.8. For methods M-I to M-IV, the solution subset $\boldsymbol{X}_s$ consisted of all variables added (and not removed) until the epoch when the relevance criterion reached a maximum value.

Method M-III (Figure 3.6) yielded better results than did method M-II, since (l,r) search allows backtracking and removes variable 7 at the end of step 4 (epoch 12). This variable, bed specific area, a_T, was removed, even if shown to be the best at epoch 2 in conjunction with variable 1 (liquid velocity, u_L). As this problem involved the largest number available features, p, it was a perfect opportunity to show the usefulness of method M-IV. This method starts with the first 6 most relevant features found with M-I and continues to grow and prune this pre-selection (Figure 3.7). As seen in Figure 3.7, the (l,r) search starting with initial pre-selection {5,1,2,11,14,4} continued to improve the J value.

The Garson's method only provided meaningful ranking for the first 3 variables. It can underline only that the liquid velocity (feature 1), the gas density (feature 13), and the gas velocity (feature 2) are important; the other features could not be confidently ranked.

The subset of variables (identified by methods M-III and M-IV): $\boldsymbol{X}_s = \{u_L,\ \mu_G,\ u_G,\ \sigma_L,\ D_c,\ \phi,\ \mu_L,\ \rho_G,\ \varepsilon,\ \rho_L,\ Foam.\}$ was most likely sufficient for predicting the flow regime classes.

There are several tools presented in recent literature that allow identification of flow regimes in the form of flow charts, empirical or fully conceptual correlations, for the liquid velocity ($u_{L,tr}$) that demarcates the transition between the LIR and HIR (Dudukovic *et al*., 2002). Most of these methods, which generally lack robustness when tested thoroughly (Larachi *et al*., 1999), use only a few variables (features) to indicate the transition between LIR and HIR. Flow charts like those of Turpin *et al*. (1967) or Sato *et al*. (1973) use only the gas and liquid mass flow rates (involving only the variables u_L, u_G, ρ_L, ρ_G), being thus restrictive and applying mainly to water-like and air-like fluids.

Table 3.8 Methods M-I to M-V compared on flow regime classification

Method	**#of var.**	**Order**	**AR (1-NN)(%)**
None (all available features considered)	14	NA	91.86
M-I: Forward selection with $I(Y\|\boldsymbol{X}_s)$. (*nbx=50*)	8	**5 1 2 11 14 4 12 13** (3 6 7 8 9 10)	92.79
M-II: Forward selection with AR(1-NN).	11	**1 7 14 2 12 11 9 4 13 10 3** (5 6 8)	93.04
M-III: (l,r) search with AR(1-NN).	11	**1 14 2 12 4 9 11 13 5 10 3** (6 7 8)	93.18
M-IV: Forward with $I(Y\|\boldsymbol{X}_s)$ continued with (l,r) search with AR(1-NN).	11	**1 14 2 12 4 9 11 13 5 10 3** (6 7 8)	93.18
M-V: Garson's saliency values through ANN	NA	1 13 2 (4 5 6 7 8 9 10 11 12 3)	NA

()* brackets here denote that the features inside cannot be confidently ranked. NA means not available. Meaning of the variables:

1	2	3	4	5	6	7	8	9	10	11	12	13	14
u_L	u_G	Foam	D_c	ε	a_G	a_T	dp	ϕ	ρ_L	μ_L	σ_L	ρ_G	μ_G

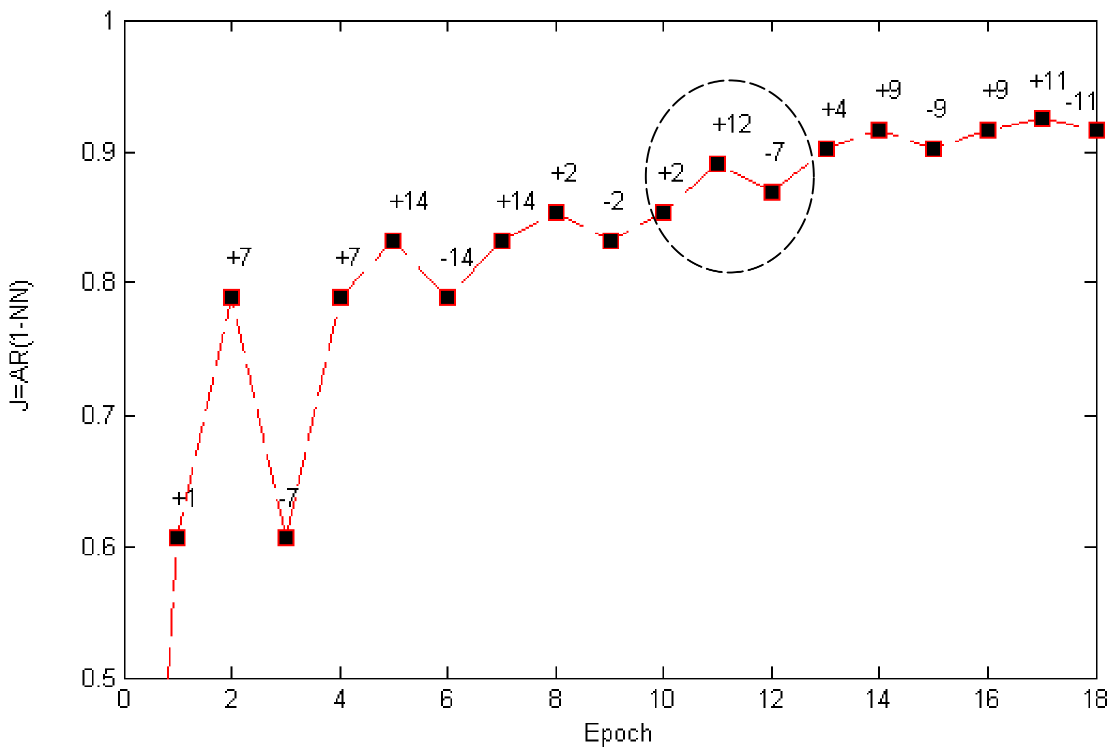


Figure 3.6 (l,r) search with accuracy rate as relevance criterion (M-III). Only the first 6 steps are shown for clarity. Dotted circle shows the 4th step. "+" / "-" means that the corresponding feature was added to /deleted from the current subset.

More recent correlations (Dudukovikc and Mills, 1986; Wang *et al*., 1994) have tried to predict $u_{L,tr}$ by taking into account also σ_L, μ_L and eventually ε. A comprehensive correlation for the liquid transition velocity was also recently proposed (Larachi *et al*., 1999) by taking into consideration the variables ρ_L, μ_L, σ_L, u_G, ρ_G, u_L, μ_G, ε and d_p.

Hence, these variables deemed in the literature to be important in flow regime identification were included directly or indirectly within the subset using the present selection feature algorithms. Note that the particle diameter was somehow involved through embedding in the sphericity and the bed porosity defined in $d_P = 6(1-\varepsilon)/a_T$ and $\phi = \pi\left(\dfrac{6(1-\varepsilon)}{\pi N_P}\right)^{2/3} \times \dfrac{N_P}{a_T}$ (N_p: number of particles per unit bed volume).

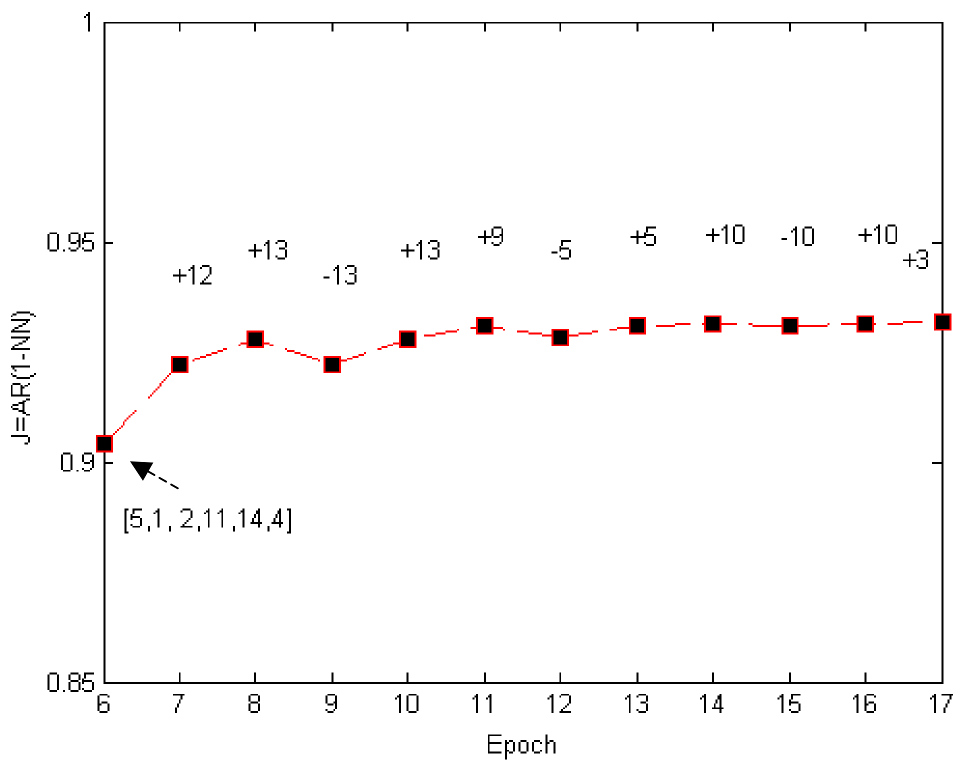


Figure 3.7 Method M-IV applied on flow regime problem. The method starts with the first six variables found with M-I and continues until no more increase in the accuracy rate of a 1-NN classifier is observed. "+" / "-" means that the corresponding feature was added to /deleted from the current subset.

3.1.6.4 Two-class bed expansion/contraction in three phase fluidized beds

In this problem, there were 10 features that might indicate bed contraction/expansion upon introduction of a tiny gas flow rate in the initially liquid-fluidized bed. Using all the features, we obtained AR(1-NN) = 96.62 %. The solution provided by methods M-I to M-IV was $\boldsymbol{X}_s$={5, 4, 1, 7, 9}. Using only these 5 variables, the accuracy rate decreased slightly to AR(1-NN)=96.26%. The first method, M-I, based on mutual information criterion, induced the following order of preference:

5	4	1	7	9
ρ_s	σ_L	u_L	u_t	H_0

while methods M-II to M-IV suggested the following order:

7	4	1	9	5
u_t	σ_L	u_L	H_0	ρ_s

Figure 3.8 and Figure 3.9 show the difference between the importance of the same selected variables by the two different relevance criteria: respectively, mutual information (M-I) and accuracy of a nearest neighbor classifier (method M-II).

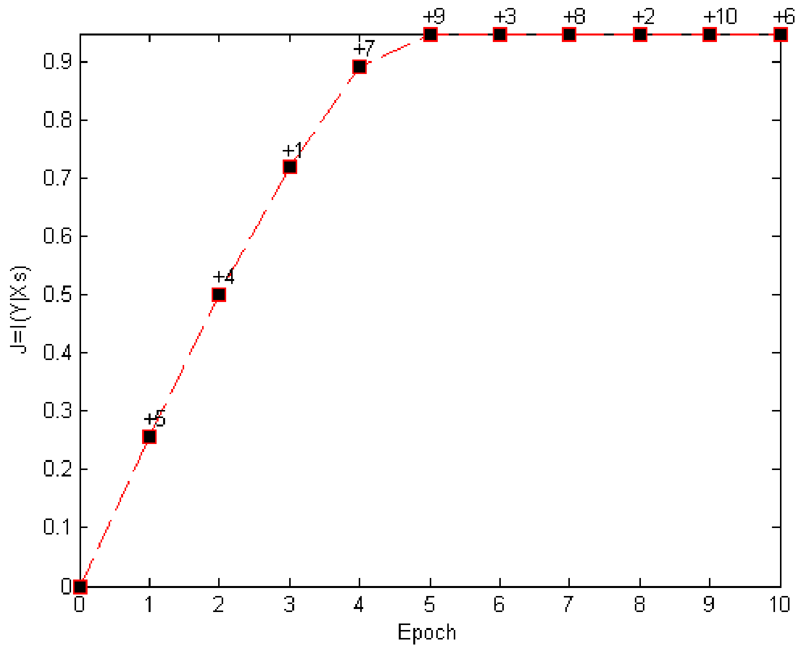


Figure 3.8 Sequential forward selection with mutual information as relevance criterion (M-I) on the bed expansion/contraction problem. "+" means that the corresponding feature was added to the current subset.

As the same search technique (sequential forward selection) was used in both cases, the difference lay only in the relevance criterion. Mutual information (M-I) gave equal importance to the solid density, liquid surface tension, particles terminal velocity, and liquid superficial velocity, and less importance to the bed height at rest. Method M-II revealed that the terminal velocity is more important than all other features in the selection, and the other features make comparable contributions to class predictability. The method M-IV did not provide a sound ranking of the features, as the saliency values were not significantly different for the 10 variables.

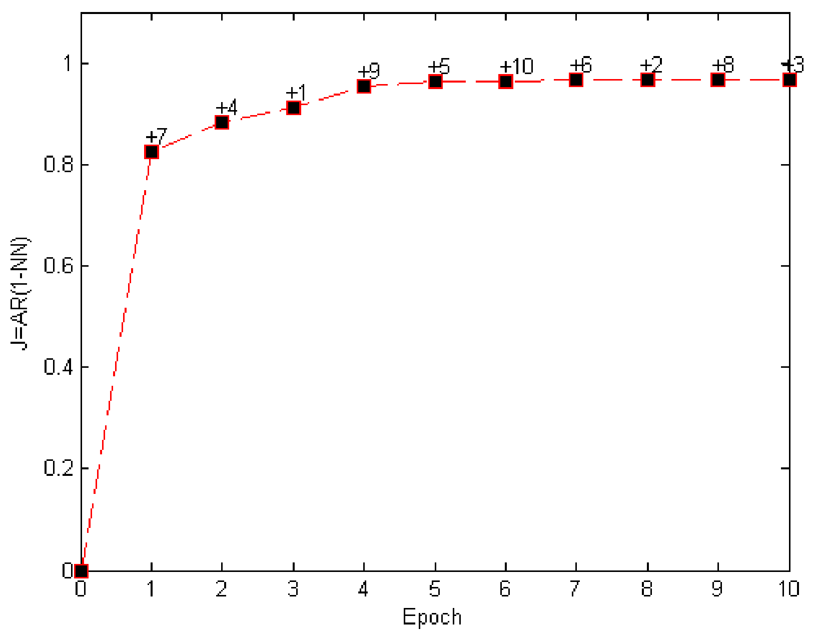


Figure 3.9 Sequential forward selection with accuracy rate as relevance criterion (M-II) on the bed expansion/contraction problem. "+" means that the corresponding feature was added to the current subset.

The elite subset of variables (identified by methods MI to M-IV): $\boldsymbol{X_s}$ = {u_t, σ_L, u_L, H_0, ρ_s} are needed to accurately predict the initial behavior of the fluidized bed. This conclusion (as in the case of trickle bed flow regimes) is based on the available data and is supported by the following physical grounds.

For a liquid fluidized bed containing small solid particles, there should be an increase in the bed porosity $\varepsilon = 1 - \varepsilon_s$ when superimposing a gas stream to a liquid-fluidized bed (ε_s = solid hold-up). In some instances, however, the bed may initially contract until it reaches a critical point, beyond which the bed height will resume its increase with an increase in gas flow rate. This happens because bubbles entrain the liquid and particles into their wakes, thereby reducing the amount of liquid in the bed used to fluidize the remaining particles (Jiang *et al*., 1997).

Bed contraction/expansion phenomena are conceptualized by the generalized wake model (Bathia and Epstein, 1974; Jean and Fan, 1986) which suggests that the liquid velocity, the particle terminal velocity (both present in $\boldsymbol{X_s}$) and the k and x bubble wake parameters control the initial bed state. The bubble wake parameters are influenced (Larachi *et al*., 2001), among other parameters, by σ_L and ρ_s, both belonging to $\boldsymbol{X}_s$. Jiang *et al.* 1997 also acknowledged that bed contraction is closely linked to the presence of large bubbles, whose number is somehow affected by the liquid superficial tension, σ_L. The Soung (1978) empirical correlation for initial bed state stated that the importance of u_t is directly correlated to u_L/u_G ratios. Feature selection algorithms reveal that bed height at rest (H_0) has an influence which is relatively marginal with respect to the other variables in set $\boldsymbol{X}_s$ (see Figure 3.8 and Figure 3.9). This is supported by literature studies that have also shown that initial bed heights could affect bed expansion or contraction.

3.1.7. Conclusions

In the first part of Chapter 3, we presented different feature selection algorithms, which helped identify the most relevant variables in two multiphase reactors classification problems. Relevance here was assessed in terms of:

i) mutual information measuring the dependence among the features and the class variable,

ii) accuracy rate of a one-nearest neighbor classifier known to work well on most problems.

The first criterion belonged to the class of filters and was a statistical measure of the filtering capabilities of variables. The second belonged to the wrappers category and used a particular classifier (here 1-NN) to test relevance. A third relevance criterion, based on the Garson’s saliency index, interpreted the size of the weights of trained neural network classifiers, which we extended to work with multiple outputs neural networks. The selection algorithms targeting maximization of the relevance criteria (mutual information and accuracy rate) in charge of the combinatorial search were sequential forward selection and the (l,r) search method.

We devised here a hybrid filter-wrapper approach, which in the first step used the mutual information criterion and SFS to obtain a set of features describing the class to be predicted. This set was further updated using the (l,r) search to maximize the performance rate of a 1-NN classifier. This last method was faster than a mere (l,r) search starting with an empty set and provided the same results on the test problems.

The different selection schemes were applied to four problems. The first two problems were benchmarks (synthetic and real) to test whether the schemes captured the proper solutions. The last two problems were borrowed from the area of multiphase reactors: i) flow regime identification in trickle beds and ii) bed initial contraction/expansion in three-phase fluidized bed reactors. For both of these problems, the ULaval multiphase reactors databases were used to identify the most relevant variables (features) for classification. The feature reduction induced, in all cases, an increase in classification performance, except for the bed expansion/contraction, where reducing the number of variables from 10 to 5 slightly decreased the accuracy. In the case of flow regime classification, the following variables were found to be important in trickle beds: u_L, μ_G, u_G, σ_L, D_c, ϕ, μ_L, ρ_G, ε, ρ_L, Foam. For bed expansion/contraction in three-phase fluidized beds, the most relevant features were ρ_s, σ_L, u_L, u_t and H_0. The next part of Chapter 3 addresses the classification issue using as discriminant features those that appeared relevant in this first step.

3.2 Data classification: application to flow regime classification in trickle beds

3.2.1 Bibliographical review and problematic

In this second half of Chapter Three, the focus is on methods used to perform classification when pertinent discriminant features are available. The case study detailed herein is the flow regime classification in trickle bed reactors (TBR). We wanted to obtain a model able to assign the correct flow regime to any particular realization of the input feature vector.

Many gas-liquid-solid (G-L-S) catalytic reactions and hydro-treating processes of petroleum refining are conducted in trickle bed reactors (Dudukovic *et al*., 2002; Holub, 1993; Ramachandran and Chaudhari, 1983). Their friendly design, which consists of a porous static granular bed traversed concurrently downward by gas and liquid streams, hides a monumental complexity associated with the randomness and chaos of fluid and solid partitions and mutual interactions. The mathematical rationalization of the flow patterns arising from this complexity has long been fueled with intuitive empirical approaches of limited success, as witnessed by the rapidly increasing number of published correlations (for an exhaustive survey, see Al-Dahhan et al., 1997; Larachi et al., 1999; Dudukovic et al., 2002).

The various flow regimes that manifest in trickle beds range from gas-continuous to liquid-continuous patterns, *e.g.*, trickle, pulse, spray, bubbly, dispersed bubble, foaming, foaming pulsing flow regimes, and so forth, depending on operating conditions and fluid and packing characteristics. A useful simplification presented in the literature consists in categorizing the flow patterns as low interaction regime (LIR), high interaction regime (HIR), and their neighboring transition regime (TR) (Charpentier and Favier, 1975). However, since the changeover from trickle flow to pulse flow regimes has been the most widely investigated over the past several decades, LIR and HIR are often confounded with trickle flow and pulse flow regimes, respectively.

Graphical representation of the data in the form of flow regime maps (Charpentier and Favier, 1975; Gianetto *et al*., 1978; Fukushima and Kusaka, 1978; Holub *et al*., 1993; Larachi *et al*., 1993) has its own pragmatic merits, considering the lack of generally accepted theories regarding the mechanisms at the origin of the various transitions. Nonetheless, compression into two or three flow chart coordinates of the relatively high number of variables having an impact on flow regime transition is not always the best choice (Larachi et al., 1999). Using a comprehensive knowledge-referenced database, Larachi et al. (1999) developed a neural-network based correlation for the superficial liquid velocity at transition $u_{L,tr}$ between trickle flow (LIR) and pulse flow (HIR) regimes. This correlation, like many other empirical ones, views the transition (TR) as a *sharp* separation between LIR and HIR, when in reality it should represent a progressive transition region in-between.

Recently, Tarca et al. (2003c) implemented a series of feature selection algorithms on the same database to determine which among the several process variables were most relevant in a subsequent flow pattern recognition classifier predicting flow regime class. The relevance measures considered were the mutual information (Batitti, 1994) and the accuracy rate of a nearest neighbor classifier. Following this analysis, the variables predicting the flow regime were the superficial liquid velocity (u_L), gas viscosity (μ_G), superficial gas velocity (u_G), surface tension (σ_L), column diameter (D_c), sphericity factor (ϕ), liquid viscosity (μ_L), gas density (ρ_G), bed porosity (ε), liquid density (ρ_L), and foam index (*Foam*). Particle size did not appear in the optimal features set. The most informative features set established, the next step was to design the data-driven inference engine (classifier).

The present study developed appropriate classification models that would predict the flow regime class using the above-mentioned features. Instead of approximating the transition superficial liquid velocity, $u_{L,tr}$, as done in the past, we attempted to model the probability that a given system state is affixed to one of the three LIR, TR, or HIR classes. This allowed us to view class TR as a gradual in-between band demarcating the transition between LIR and HIR classes, rather than as a sharp change. Flow regime classification can thus be approached like in statistical pattern recognition field. This is done by considering classes LIR, TR, and HIR, and by searching for discriminant functions predicting the class function of the system particularities and operating conditions.

There are numerous statistical and neural network paradigms used to perform supervised classification, using as a basis a data set of examples with known flow regime membership. Those tested in this work included: Gaussian (quadratic discrimination rule), linear (normal based), nearest mean class, nearest neighbor, *k*-nearest neighbor, binary decision tree, radial basis functions, and multilayer perceptron neural networks. Multilayer perceptrons are by far the most popular neural network classifiers, as they generally exhibit superior performance with respect to other classification algorithms (Lowe and Webb, 1990). However, using neural networks in the classical way can be unsuitable. Therefore, specificities of flow regime classification problems must be dealt with as:

- The misclassification cost inequality for non-adjacent classes. Predicting a system in LIR, while it actually belongs to HIR, is a much more serious misclassification than predicting it in class TR.

- The adherence of the model to some *a priori* knowledge in the form of monotonicity constraints, assuring shift from one class to another. These qualitative rules guarantee model phenomenological consistency (Tarca *et al*., 2003a).

This contribution is organized as follows. First, the knowledge-referenced database is briefly described. It is followed by an introduction to classification, the common classifiers, and the measures for their performance assessment. A comparison is then made between these classifiers, using as criteria their cross-validated error, a loss measure pertinent to flow regimes classification, complexity, and interpretability. Different possibilities of incorporating prior knowledge as class connectivity and different misclassification costs are presented. Finally, an improved neural network model is devised for flow regime classification in trickle beds, and its capabilities and limitations are discussed.

3.2.2 Description of flow regime database

The comprehensive knowledge-referenced database of flow regime observations (Larachi *et al*., 1999) was considered. A database query was used to retrieve only the sources of records where the authors observed all three regime classes with no missing values for u_L, μ_G, u_G, σ_L, D_c, ϕ, μ_L, ρ_G, ε, ρ_L, *Foam* within each class. The resulting database contained 5,061 points distributing as: 1,937 LIR, 958 TR, and 2,166 HIR. Note the uneven distribution of instances among classes with TR class having *ca*. twice fewer records. Data normalization was performed as follows:

$$x_{r,j} = \begin{cases} \dfrac{x'_{r,j} - \min(x'_j)}{\max(x'_j) - \min(x'_j)} & \text{if} \quad \dfrac{\max(x'_j)}{\min(x'_j)} < 100 \\ \dfrac{\log\left(x'_{r,j} / \min(x'_j)\right)}{\log\left(\max(x'_j) / \min(x'_j)\right)} & \text{otherwise} \end{cases} \tag{3.13}$$

In which x'_j is the value of feature j in the feature vector $\{u_L, \mu_G, u_G, \sigma_L, D_c, \phi, \mu_L, \rho_G, \varepsilon, \rho_L\}$ in the original measurement units, and r is the index of the record. The ranges of the feature values in the original measurement units are summarized in Table 3.9.

Table 3.9 Ranges of input variables in the flow regime classification

Variable	Symbol	Max	Min
Liquid superficial velocity (m/s)	u_L	1.74E-01	9.03E-06
Gas viscosity (Pa.s)	μ_G	1.97E-05	1.45E-05
Gas superficial velocity (m/s)	u_G	4.08E+00	4.98E-04
Surface tension (N/m)	σ_L	7.62E-02	1.90E-02
Column diameter (m)	D_c	5.10E-01	2.30E-02
Sphericity (-)	ϕ	1.00E+00	3.20E-01
Liquid viscosity (Pa.s)	μ_L	6.63E-02	3.10E-04
Gas density (kg/m^3)	ρ_G	1.16E+02	1.60E-01
Bed porosity (-)	ε	7.50E-01	2.63E-01
Liquid density (kg/m^3)	ρ_L	1.18E+03	6.50E+02
Foaming property (0=coleasing; 1=foaming)	FOAM	1	0

3.2.3 Supervised classification, classifiers, and performance evaluation

3.2.3.1 Supervised classification

Let us consider the classification issue. Suppose there are n objects (or patterns) $\mathbf{x}_r$, $r = 1 \ldots n$, $\mathbf{x}_r = (x_{r,1}, x_{r,2}, \ldots x_{r,j}, \ldots x_{r,p})^T$, each with a class label y_i, $i = 1, 2 \ldots C$, where C is the number of classes (C=3). These samples constitute the design set $D = \{(\mathbf{x}_r, y_i(\mathbf{x}_r)), r = 1 \ldots n\}$ to be used for building a classifier (decision rule or inference engine) able to generalize for any new observation $\mathbf{x} \in [0,1]^p$, i.e., for any value each of the p features will take in the interval [0,1].

Consider now the *C* classes $y_i, \mathrm{i} = 1...\mathrm{C}$, (LIR, TR, HIR) with *a priori* probabilities (the probabilities of each class occurring) *p(y_i)* assumed known. In order to minimize the probability of making an error in the classifier operation, and with no information other than the prior probabilities *p(y_i),* we assign an object to class y_i if

$$p(y_i) > p(y_k), \quad \mathrm{k} = 1, 2 \ldots \mathrm{C}\,; \mathrm{k} \neq i \tag{3.14}$$

which classifies all objects as belonging to the majority class. (For classes with equal probabilities, patterns are assigned arbitrarily among those classes). Knowing the values of the observation vector ***x*** and its associated conditional probability, ***x*** must be assigned to class y_i if *the probability of class $y_{i,}$ given the observation **x***, i.e., $p(y_i \,|\, \boldsymbol{x})$, is the largest over all classes. That is, *assign **x** to class y_i if*:

$$p(y_i \mid \mathbf{x}) > p(y_k \mid \mathbf{x}), \quad \mathrm{k} = 1, 2 \ldots \mathrm{C}\,; \mathrm{k} \neq i \tag{3.15}$$

Using the Bayes' rule, such *a posteriori* probabilities $p(y_i \mid \mathbf{x})$ can be expressed in terms of the a priori probabilities $p(y_i)$ and the *class-conditional density functions* $p(\mathbf{x} \mid y_i)$:

$$p(y_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid y_i) \cdot p(y_i)}{p(\mathbf{x})} \tag{3.16}$$

Eq. (3.15) may therefore be rewritten as:

*assign **x** to class y_i if*:

$$p(\mathbf{x} \mid y_i) \cdot p(y_i) > p(\mathbf{x} \mid y_k) \cdot p(y_k), \quad \mathrm{k} = 1, 2 \ldots \mathrm{C}\,; \mathrm{k} \neq i \tag{3.17}$$

which is known as the Bayes' rule for minimum error.

In practice, one may set a priori probabilities as equal ($p(y_i)=1/C$) when a new sample point ***x*** drawn from the samples space R^p is expected to fall in either class y_i with equal probability. Alternatively, class distribution in the design set *D* can be considered representative of the sample space. In this case, the priors were computed simply as $p(y_i)=n_i/n$, where n_i was the number of class *i* occurrences in the design set *D*.

Assuming that $p(y_i)$ is known, we need to estimate only the class conditional density $p(\mathbf{x} \mid y_i)$, as $p(\mathbf{x})$ is independent of the class.

The statistical pattern recognition considers two basic approaches to density estimation: parametric and nonparametric. In the parametric approach, we assume that $p(\mathbf{x} \mid y_i)$ is of a known form, but has an unknown set of parameters. In the nonparametric approach, the density is estimated without making any functional assumption. In both cases, the design set D of observations with known class is used to approximate the class-conditional probabilities.

3.2.3.2 Classifiers

In multi-class classification problems like flow regime assignment in trickle beds, a classifier may be viewed as *C* discriminant functions $g_i(\boldsymbol{x})$ such that:

$$g_i(\mathbf{x}) > g_k(\mathbf{x}) \Rightarrow \mathbf{x} \in y_i , \quad \mathrm{k} = 1, 2 \ldots \mathrm{C} \, ; \mathrm{k} \neq i \tag{3.18}$$

meaning that a pattern is assigned to the class having the largest discriminant function which, according to the Bayes decision rule Eq. (3.17), is written as:

$$g_i(\mathbf{x}) = p(\mathbf{x} \mid y_i) \cdot p(y_i) \tag{3.19}$$

Literature is replete with discriminant functions varying in complexity from linear (in which *g* is a linear combination of x_j) to multiparameter nonlinear functions such as multilayer perceptron neural networks. Below is a brief description of some classifiers tested in this work, with some details drawn from Webb (2002).

A) Gaussian classifier (quadratic discrimination rule)

This is a classifier based on the normality assumption of the class-conditional probability function, i.e.:

$$p(\mathbf{x} \mid y_i) = \frac{1}{(2\pi)^{\frac{p}{2}} \left|\boldsymbol{\Sigma}_i\right|^{1/2}} \exp\left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right\} \tag{3.20}$$

where μ_i and $\mathbf{\Sigma}_i$ are respectively the center and the covariance matrix of the normal distributions and p the dimensionality of the input vector $\boldsymbol{x}$. The normal based quadratic discriminant function (McLachlan, 1992) may be obtained from taking log value of Eq. (3.19) r.h.s. with $p(\mathbf{x} \mid y_i)$, as indicated in (3.20), and removing the terms which remain constant for all classes:

$$g_i(\mathbf{x}) = \log(p(y_i)) - \frac{1}{2}\log\left(\left|\hat{\mathbf{\Sigma}}_i\right|\right) - \frac{1}{2}(\mathbf{x} - \mathbf{m}_i)^T \hat{\mathbf{\Sigma}}_i^{-1}(\mathbf{x} - \mathbf{m}_i) \tag{3.21}$$

In Eq. (3.20), the true mean μ_i and covariance matrix $\mathbf{\Sigma}_i$ were replaced by their maximum likelihood estimates

$$\mathbf{m}_i = \frac{1}{n_i}\sum_{r=1}^{n_i}\mathbf{x}_r \tag{3.22}$$

and

$$\hat{\mathbf{\Sigma}}_i = \frac{1}{n_i}\sum_{r=1}^{n_i}(\mathbf{x}_r - \mathbf{m}_i)(\mathbf{x}_r - \mathbf{m}_i)^T \tag{3.23}$$

B) Normal based linear

A simplification of the gaussian classifier assumes that the class covariance matrices $\mathbf{\Sigma}_i$ are all the same, in which case the discriminant functions g_i become:

$$g_i(\mathbf{x}) = \log(p(y_i)) - \frac{1}{2}\mathbf{m}_i^T \mathbf{S}_w^{-1}\mathbf{m}_i + \mathbf{x}^T \mathbf{S}_w^{-1}\mathbf{m}_i \tag{3.24}$$

The unbiased estimate of the pooled within-group sample covariance matrix $\mathbf{S}_{\mathrm{w}}$ is given by

$$\mathbf{S}_w = \frac{n}{n - C}\sum_{i=1}^{C}\frac{n_i}{n}\hat{\mathbf{\Sigma}}_i \tag{3.25}$$

O'Neill (1992) showed that the linear discriminant rule (Eq. (3.24)) is quite distinct from the equal covariances matrix assumptions and may perform better than the optimum quadratic discriminant rule for normally distributed classes when the true covariance matrices are unknown and the sample sizes are small.

C) Nearest class mean

This is a quite simple classification approach. The mean vectors of samples in each class $\mathbf{m}_i$ are computed with Eq. (3.22). Any new pattern $\boldsymbol{x}$ is assigned to the class whose mean, evaluated with the Euclidian metric distance, is the nearest. The squared Euclidian distance is

$$|\,\mathbf{x}-\mathbf{m}_i\,|^2 = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{m}_i + \mathbf{m}_i{}^T\mathbf{m}_i \qquad (3.26)$$

The discrimination function implemented by the nearest mean class is:

$$g_i(\mathbf{x}) = \mathbf{w}_i{}^T\mathbf{x} + w_{0i} \qquad (3.27)$$

with $\mathbf{w}_i = \mathbf{m}_i$ and $w_{0i} = -\frac{1}{2}|\,\mathbf{m}_i\,|^2$

The maximum g_i will then correspond to the minimum distance between $\boldsymbol{x}$ and the class mean.

D) Nearest Neighbor

The discriminant function implemented by the nearest neighbor principle is very similar to that of the nearest class mean classifier. In the nearest neighbor principle, however, distances from the point $\boldsymbol{x}$ are computed with respect to all patterns in all classes, and the class of the nearest neighbor is assigned to $\boldsymbol{x}$.

Consider n_i patterns in class y_i, $\mathbf{p}_i^1,\ldots\mathbf{p}_i^{n_i}$, $i = 1, 2\ldots C$. The nearest neighbor discriminant function for class y_i is

$$g_i(\mathbf{x}) = \max_{k=1,\ldots,n_i} g_i{}^k(\mathbf{x}) \qquad (3.28)$$

with

$$g_i^k(\mathbf{x}) = \mathbf{x}^T \mathbf{p}_i^k - \frac{1}{2} \mathbf{p}_i^{k^T} \mathbf{p}_i^k, \; k = 1, 2 \ldots n_i \; ; \; i = 1, 2 \ldots C \tag{3.29}$$

A pattern $\boldsymbol{x}$ is assigned to the class for which $g_i(\mathbf{x})$ is largest, that is, to the class of the nearest prototype vector. This discriminant function generates a piecewise decision boundary.

E) k-Nearest-Neighbor

This is a natural extension of the nearest neighbor rule. Once the subsidiary discriminant functions $g_i^k(\boldsymbol{x})$ are computed as described earlier, they are sorted decreasingly to correspond to an increasing order of distances from $\boldsymbol{x}$ to all the patterns. Suppose that in the first k samples, there are k_m in class y_m, $\sum_{m=1}^{C} k_m = k$. The k-nearest neighbor rule reads: *assign* $\boldsymbol{x}$ *to class* y_m *if* $k_m \geq k_i$, $i=1,\ldots,C$.

For situations where two or more classes receive an equal number of votes, we break the tie by assigning $\boldsymbol{x}$ to the class, out of the classes with tying values of k_i, that has the nearest mean vector (calculated over the k_i samples) to $\boldsymbol{x}$.

F) Multilayer perceptron neural network

The multilayer perceptron (MLP) is a feed forward neural network that has recently been the subject of much discussion. Used for function approximation (in regression context) and classification, MLP is robust compared with other available statistical tools. First introduced by Rumelhart *et al.* (1986), an MLP used in a C –class classification problem can be seen as a set of C discriminant functions g_i.

$$g_i(\mathbf{x}) = \sigma\left(\sum_{j=1}^{J} \left(w_{j,i} \cdot \phi_j \left(\sum_{k=1}^{p} \left(x_k \cdot \alpha_{k,j} \right) + \alpha_{p+1,j} \right) \right) + w_{J+1,i} \right) \qquad i=1,\ldots,C \tag{3.30}$$

with

$$\phi_j(z) = \frac{1}{1+e^{-z}} \tag{3.31}$$

which is the logistic sigmoid.

The MLP computing the output i is a single hidden layer, fully connected feed-forward neural network. (See Eq. (3.30)). There are J neurons (nonlinear function) denoted with ϕ_j. First, the input vector $\boldsymbol{x}$ is projected onto each of the J directions described by the vectors α_j; transforming the projected data (offset by the bias $\alpha_{p+1,j}$) by the nonlinear functions $\phi_j(z)$; then, a linear combination is done using the weights $\boldsymbol{w}_i$. If σ is taken as the identity function $\sigma(z) = z$ (obtaining thus a MLP with linear output units), this is the final result. If σ is taken as the logistic sigmoid Eq. (3.31) (obtaining thus a MLP with nonlinear output units), a second nonlinear transformation is done. We considered the latter option in this study.

The free parameters (weights) of the discriminant functions $\alpha_{k,j}$, k=1,..., p+1, j=1,...J; and $w_{j,i}$ j=1,...J+1, I=1,...C were estimated using a training procedure. Classically, in one-of-C target coding, the error of the neural network (constituted by the C discriminant functions $g_i(\boldsymbol{x})$ considered all together) is computed as

$$E = \sum_{r=1}^{n} \sum_{i=1}^{C} \left(t_{r,i} - g_i(x_r) \right)^2 \tag{3.32}$$

where $t_{r,i}$=1 if $\boldsymbol{x}_r$ belongs to the class i , and $t_{r,i}$=0 otherwise.

The training algorithm we used to estimate the weights of the neural network was the Levenberg-Marquardt algorithm implemented in Matlab® Neural Networks Toolbox.

G) Radial basis functions

Radial basis functions were originally proposed for function approximation. They were first used for discrimination by Broomhead and Lowe (1988). They are very closely related to both kernel methods for density estimation and regression and to normal mixture models. Mathematically they can be described as a linear combination of radially symmetric nonlinear basis functions. They transform a pattern $\mathbf{x} \in R^p$ to a C-dimensional output space:

$$g_i(\mathbf{x}) = \sum_{j=1}^{J}\left(w_{i,j} \cdot \phi_j\left(\frac{|\,\mathbf{x}-\boldsymbol{\mu}_j\,|}{\beta}\right)\right) + w_{i,J+1} \qquad i=1,\ldots,C \tag{3.33}$$

The transfer functions here are gaussian, i.e.,

$$\phi_j(z) = \exp(-z^2) \tag{3. 1}$$

The weights $\boldsymbol{w}_i$ are determined with a Least Squares method. The centers $\boldsymbol{\mu}_i$ are selected from the training samples, and the spread of basis functions is set by trial and error. We used the orthogonal least squares algorithm of Chen *et al*. (1991) implemented in Matlab® Neural Networks Toolbox.

H) Binary classification trees

A special type of classifier is the decision tree, which is trained by an iterative selection of individual features that are the most salient at each node in the tree. The criteria for feature selection and tree generation include node purity or Fisher's criterion. The most commonly used decision tree classifiers are binary and use a single feature at each node, resulting in decision boundaries that are parallel to the feature axes. They are, therefore, intrinsically suboptimal, but are able to interpret the decision rule in terms of individual features.

There are several heuristic methods for constructing decision-tree classifiers. They are usually constructed top-down, beginning at the root node and successively partitioning the feature space. The construction involves three main steps:

1. Selecting a splitting rule for each internal node, i.e., determining the feature together with a threshold that will be used to partition the data set at each node.

2. Determining which nodes are terminal nodes. This means that, for each node, we must decide whether to continue splitting or to make the node terminal and assign it a class label.

3. Assigning class labels to terminal nodes. This is straightforward; labels can be assigned by minimizing the estimated misclassification rate. The binary tree we used is based on the Quinlan's C 4.5 algorithm (Quinlan, 1993).

3.2.3.3 Performance evaluation

3.2.3.3.1 Misclassification rate

One method used to estimate the misclassification rate involves computing the classifier confusion matrix on several *v-fold cross-validation sets* (Kohavi, 1995 for cross-validation issues). Consider the design set $D = \{(\boldsymbol{x}_r, y_i(\boldsymbol{x}_r)),\ r = 1,..,n\}$ for a given classification problem. Let set D be partitioned in v disjoint subsets (or folds) A_k such that $A_k \subset D$, $\forall\ k=1,.., v$ and $A_i \cap A_j = \{\varnothing\}\ \forall\ i \neq j$. In each fold A_k the percentage of samples belonging to class y_i, $i=1,\ldots,C$ is about the same as in set D.

Let $\eta(z;D–A_k)$ denote the class label predicted by the classifier trained on the $D–A_k$ data set when input z, with true class $y(z)$, is presented. The loss function usually adopted in classification error estimation is:

$$Q\big(\mathrm{y}(\mathbf{z}),\eta(\mathbf{z};D-A_k)\big)=\begin{cases}0 & \mathit{if}\ \mathrm{y}(\mathbf{z})=\eta(\mathbf{z};D-A_k)\\ 1 & \quad\text{otherwise}\end{cases} \tag{3.35}$$

The misclassification rate may be defined as:

$$Err=\frac{1}{n}\sum_{k=1}^{v}\sum_{m=1}^{n_k^{'}}Q\big(\mathrm{y}(\mathbf{z}_{\mathrm{m}}),\eta(\mathbf{z}_m;D-A_k)\big) \tag{3.36}$$

where $n_k^{'}$ represents the number of samples in the set $\mathrm{A_k}$ and n is the number of samples in the whole set D. Thus, the misclassification rate expresses the fraction of points in D that are misclassified.

The meaning of the misclassifications partition can be clarified by using the global confusion matrix, which is obtained by summing up the v confusion matrices obtained by testing the classifiers on the A_k sets while being trained on the complementary $D\text{-}A_k$ sets.

The size of the confusion matrix CM is $(C\times C)$. Each of its cells indicates how many samples are assigned to class y_i (i is the column index) when the actual (true) class index is y_j (j is the row index):

$$CM = \begin{array}{lll|ccc|} & & & \multicolumn{3}{c}{predicted\quad class} \\ & & & i=1 & i=2 & i=3 \\ & & & (LIR) & (TR) & (HIR) \\ & true & j=1(LIR) & 80 & 15 & 5 \\ & class & j=2(TR) & 10 & 75 & 15 \\ & & j=3(HIR) & 5 & 5 & 90 \end{array} \tag{3.37}$$

Therefore, the misclassification rate *Err* is the sum of the off-diagonal elements in the confusion matrix divided by the sum of all elements in the matrix. In the example given in Eq. (3.37), the design set has 300 data points; 55 of those were misclassified. The *Err* is then 55/300.

Note that the classifier error estimate *Err* obtained using the above procedure yields an upper bound limit of the error rate, since at each fold a data fraction $1/v$ is not used in training. However, if the classifier is trained on all data, the obtained *Err* value would be lower.

3.2.3.3.2 Loss value using class connectivity information

Let us define a flow regime classification loss function L by proposing a heuristic cost matrix for misclassification. A reasonable cost matrix could be:

$$Cost = \begin{array}{cc} & predicted\,class \\ \begin{array}{c} true \\ class \end{array} & \begin{pmatrix} 0 & 1 & 3 \\ 1 & 0 & 1 \\ 3 & 1 & 0 \end{pmatrix} \end{array} \tag{3.38}$$

Each element in this matrix denotes the cost assigned to predicting a pattern in class j (given by column index) while the true class is i (indicated by row index). Misclassifying a point increases the cost by 1, whereas confusing non-adjacent HIR and LIR classes is penalized trice.

Finally, the loss L is obtained through element-wise multiplication of the confusion matrix with the cost matrix, taking the sum over all elements:

$$L = \sum_{i=1}^{C} \sum_{j=1}^{C} Cost(i,j) \cdot CM(i,j) \tag{3.39}$$

3.2.4 Results

3.2.4.1 Results with common classifiers

The performance measures, *i.e.*, misclassification rate (*Err*) and loss value *L* for the classifiers presented in §3.2 were tested on the 3-class flow regime database (LIR, TR, HIR, 5 061 records). The results on flow regime classification with the various statistical and neural network classifiers are summarized in Table 3.10. For each classifier, the number of parameters, the misclassification rate, the loss, and the confusion matrix are given.

The smallest prediction error was achieved with the nearest neighbor classifier, which also gave the minimal loss. However, this classifier needed all the data set (features plus class membership) as parameters in order to make flow regime predictions. The second best classifier was the MLP neural network, which exhibited a reasonably low number of free parameters: 153 instead of 60 732 for the nearest neighbor classifier. The error of the Gaussian classifier was about twice as high as the MLP, suggesting that the sample distributions in each of the three classes deviated strongly from the normality assumption used in the quadratic discrimination rule. Also, deviation from normality was confirmed by the poor performance of the nearest class mean classifier, which could have yielded better predictions, if the data within each class had been clustered around their class means.
Though the normal-based linear classifier lead to poorer classification than the Gaussian quadratic classifier, it did, however, largely prevent confounding the HIR with LIR and vice-versa, as revealed by the lower loss value. The radial basis functions performed rather poorly compared with the MLP neural network, despite the fact that RBFs use twice as many free parameters.

The binary classification tree, which is the most interpretable classifier among all listed in Table 3.10, was too imprecise to be considered further.

The MLP neural network exhibited the best trade-off between accuracy (ranked third for misclassification rate) and complexity (ranked third with lowest number of parameters) as shown in Table 3.10. It will be presented in the next section, which focuses on alternative methods of embedding prior knowledge in this type of classifier.

Table 3.10 Results on flow regime classification problem with statistical and neural network classifiers

Classifier	**# Parameters**	**Misclassifi - cation rate (%)**	**Loss**	**Confusion Matrix**
A) Gaussian (quadratic discriminant) $p(y_i) = 1/C$, i =1…C	$C \cdot (p \cdot p + p) = 396$	23.3	1899	$\begin{pmatrix} 1559 & 257 & 121 \\ 219 & 655 & 84 \\ 239 & 259 & 1668 \end{pmatrix}$
B) Normal based linear discriminant	$C \cdot p + p \cdot p = 154$	28.0	1671	$\begin{pmatrix} 1313 & 541 & 83 \\ 72 & 779 & 107 \\ 43 & 573 & 1550 \end{pmatrix}$
C) Nearest class mean	$C \cdot p = 33$	60.1	4431	$\begin{pmatrix} 240 & 1044 & 653 \\ 77 & 680 & 201 \\ 41 & 1027 & 1098 \end{pmatrix}$
D) Nearest Neighbor	$n \cdot (p+1) = 60732$	9.0	840	$\begin{pmatrix} 1783 & 53 & 101 \\ 61 & 807 & 90 \\ 92 & 57 & 2017 \end{pmatrix}$
E) k- Nearest Neighbor (k=5)	$n \cdot (p+1) + 1 = 60733$	10.9	1011	$\begin{pmatrix} 1760 & 57 & 120 \\ 86 & 737 & 135 \\ 109 & 46 & 2011 \end{pmatrix}$
F) Multilayer perceptron J=10 hidden neurons, 200 ite. (LM) (Standard MLP).	$(p+1) \cdot J + (J+1) \cdot C$ $= 153$	11.5	1005	$\begin{pmatrix} 1778 & 63 & 96 \\ 118 & 725 & 114 \\ 116 & 74 & 1976 \end{pmatrix}$
G) Radial basis functions J=20 hidden neurons	$p \cdot J + (J+1) \cdot C + 1$ $= 284$	21.5	1854	$\begin{pmatrix} 1760 & 74 & 216 \\ 251 & 353 & 354 \\ 167 & 26 & 1973 \end{pmatrix}$
H) Binary classification tree Nodes=20, Leafs=11	$3 \cdot Nodes + Leafs +$ $+ 3 \cdot (Nodes - Leafs) =$ $= 98$	29.8	2842	$\begin{pmatrix} 1439 & 19 & 479 \\ 338 & 141 & 479 \\ 188 & 5 & 1973 \end{pmatrix}$

3.2.4.2 Knowledge augmented MLP classifiers

There are at least two types of qualitative prior information available about the flow regime classification that could improve MLP performances.

MLP-A) Using costs directly in MLP training

Multi-class problems often use a 1-of C coding scheme for MLP training. Unfortunately, this does not take into account the different costs associated with misclassifications.

A possible way to use the cost information without altering the learning power of MLP training algorithms is to use the rows of the Cost matrix as target vectors for the neural network (Lowe and Webb, 1990).

The cost matrix (Eq. (3.37)) normalized by division with its maximum element (in order to obtain values in the interval [0 1]) was therefore used for network training:

Class	Classic target vectors	Cost coded target vectors
1	[1 0 1]	[0 1/3 1]
2	[0 1 0]	[1/3 0 1/3]
3	[0 0 1]	[1 1/3 0]

The class to which a pattern is assigned by this trained network corresponds to the coded target vector with respect to which the networks' response vector is closest in terms of the Euclidian distance. The confusion matrix, the loss value, and misclassification rates obtained by this neural network are given in Table 3.11.

MLP-B) Continuous output MLP classifier

Because the classes are somehow ordered-- i.e., LIR, then TR, followed by HIR-- we may change the classification into a regression. For this, a single output MLP is used. As the output is endowed with sigmoid transfer function, the following coding is used: 0 for LIR, 0.5 for TR, and 1 for HIR.

Once trained, the network will produce an output $\hat{y}(\mathbf{x})$ for each new input point $\boldsymbol{x}$. This will be assigned to class LIR if $0 \le \hat{y}(\mathbf{x}) < 0.25$, to TR if $0.25 \le \hat{y}(\mathbf{x}) < 0.75$, or to HIR if $0.75 \le \hat{y}(\mathbf{x}) < 1$.

The number of hidden nodes chosen by trial and error was 12, which gave us 157 parameters. This number was almost the same as in the standard MLP, (see the Table 3.10) as well as in MLP-A discussed previously. The performance in terms of loss is presented in Table 3.11. These results are far better than those obtained with MLP-A.

MLP-C) Continuous output MLP classifier with monotonicity constraints

Another aspect of *a priori* class connectivity knowledge on flow regime classification concerns the shift of the class memberships in a hierarchical manner subject to monotonic variations in some of the process variables. As an example, from this prior knowledge, the influence of liquid and gas superficial velocities and gas density on class connectivity is given as:

$u_L \uparrow$: class moves in the order LIR → TR → HIR

$u_G \uparrow$: class moves in the order LIR → TR → HIR

$\rho_G \uparrow$: class moves in the order HIR→ TR→ LIR

As the classification problem turns into a regression one, these rules may be also viewed as:

$$\frac{\partial \hat{y}(\mathbf{x})}{u_L} \geq 0 \tag{3.40}$$

$$\frac{\partial \hat{y}(\mathbf{x})}{u_G} \geq 0 \tag{3.41}$$

$$\frac{\partial \hat{y}(\mathbf{x})}{\rho_G} \leq 0 \tag{3.42}$$

The same network configuration as in MLP-B is then trained in such a way to guarantee monotonicity behaviour. The training in our study was done as described in a previous work (Tarca *et al.*, 2004a) using a genetic algorithm-genetic hill climber optimizer. The model provided by this methodology was fine-tuned using a gradient-based constrained technique implemented in Matlab ®. Both optimization schemes were performing the following constrained minimization problem:

$$\min_{w} \sum_{i=1}^{N_T} (y_i - \hat{y}_i)^2 \qquad (3.43)$$

subject to:

$$w_j \cdot w_{k,j} \geq 0 \,,\ \forall j, 1 \leq j \leq J \text{ and k=\{1,3\}} \qquad (3.44)$$

$$w_j \cdot w_{k,j} \leq 0 \,,\ \forall j, 1 \leq j \leq J \text{ and k=\{8\}} \qquad (3.45)$$

In Eq. (3.43) y_i represents the coded desired output (y=0 for LIR, y=0.5 for TR, and y=1 for HIR) while $\hat{y}_i$ is the value predicted by the model for the training sample $\mathbf{x}_i$. (See Table 3.12 for the full set of equations).

If all the weights $w_{k,j}$ between the input k and the hidden node j have the same sign as the weights w_j from the hidden node j to the output node, then the neural network function $\hat{y}$ will be monotonically increasing with respect to the input k in the entire definition domain. Conversely, if the signs are all opposite, then decreasing monotonicity will be achieved. The inputs {1,3} referred to in eq. (3.44) correspond to u_L and u_G, while {8} correspond to ρ_G. A five-fold cross-validation was carried out to estimate an upper bound limit of the misclassification rate for all classifiers. The 16.2% misclassification rate for MLP-C was slightly higher than for MLP-B (Table 3.11) which is natural as long as MLP-C was trained respecting the monotonicity constraints. Using all the training data (5061) for building an MLP-C classification model yielded, as expected, a slightly lower misclassification rate of 15.5% with respect to the cross-validation procedure. Moreover, this MLP-C model required only 118 parameters, as opposed to. 157 with MLP-B. Table 3.12 gives the full set of equations for the MLP-C model. An Excel spreadsheet implementing the model is also available at http://www.gch.ulaval.ca/bgrandjean or http://www.gch.ulaval.ca/flarachi. Figure 3.10 illustrates the flow regime boundaries obtained with MLP-C for a particular trickle bed system with the following simulated gas-liquid-solid properties:

μ_G	σ_L	D_c	ϕ	μ_L	ρ_G	ε	ρ_L	*Foam*
1.74E-05	7.00E-02	5.00E-02	1.00E+00	1.02E-03	1.20E+00	3.80E-01	1.00E+03	0

Figure 3.10 also shows the 31 experimental data points that both fulfill the above constraints and fall in the transition regime class. Note that the MLP-C classifier predicts a progressive changeover from LIR to HIR through a relatively broad band TR, particularly in the low gas load region. Obviously the level of regime interaction shifts in the right direction with increasing u_L, u_G, and ρ_G, as illustrated in Figure 3.10 and Figure 3.11. Note how a low-pressure HIR operation (point in asterisk) is shifted to the transition regime class at high pressure (Figure 3.11). A further increase in gas density would result in full operation in LIR.

Table 3.11 Knowledge augmented MLP classifiers

Classifier	Prior knowledge considered	Misclassifi - cation rate (%)	Loss	Confusion Matrix
Standard Multilayer perceptron *J=10 hidden neurons, 3 output nodes*	None	11.5	1005	$\begin{pmatrix} 1778 & 63 & 96 \\ 118 & 725 & 114 \\ 116 & 74 & 1976 \end{pmatrix}$
Cost trained Multilayer perceptron (MLP-A) *J=10 hidden neurons, 3 output nodes*	Cost encoded in targets	15.1	892	$\begin{pmatrix} 1696 & 210 & 31 \\ 189 & 617 & 152 \\ 33 & 149 & 1984 \end{pmatrix}$
Continuous output MLP classifier (MLP- B above) *J=12 hidden neurons, 1 output node*	Considers a natural ranking of the classes, so LIR shares no border with HIR.	14.8%	871	$\begin{pmatrix} 1731 & 177 & 29 \\ 164 & 632 & 162 \\ 31 & 188 & 1947 \end{pmatrix}$
Continuous output **monotonic** MLP classifier (MLP-C) *J=12 hidden neurons, 1 output node*	Considers a natural ranking of the classes and monotonicity: $u_L\uparrow_{\text{Class}}\uparrow$; $u_G\uparrow_{\text{Class}}\uparrow$ $\rho_G\uparrow_{\text{Class}}\downarrow$	16.22 %	935	$\begin{pmatrix} 1704 & 205 & 28 \\ 192 & 597 & 169 \\ 29 & 198 & 1939 \end{pmatrix}$

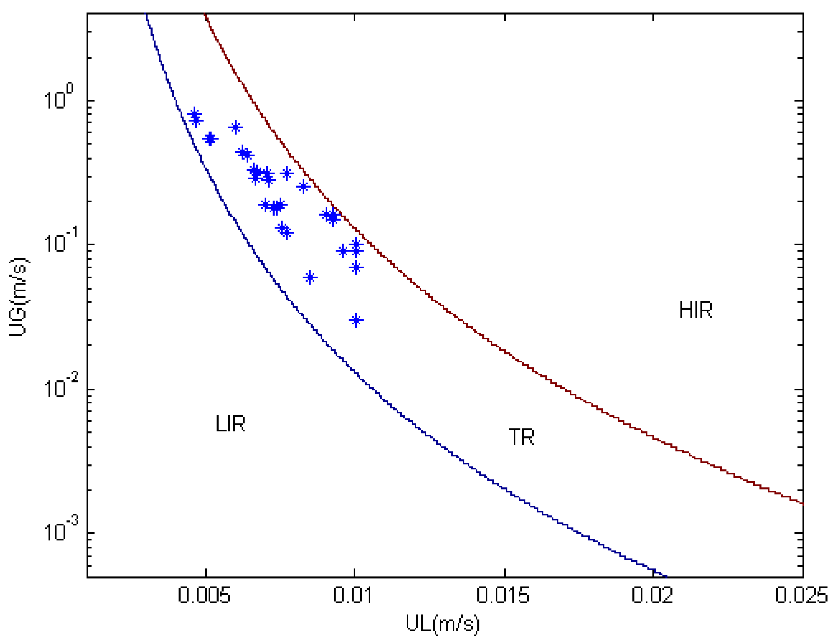


Figure 3.10 Decision boundaries delineating LIR, TR, and HIR classes obtained from continuous output monotonic MLP-C model. Experimental data points in the chart are known to belong to TR class.

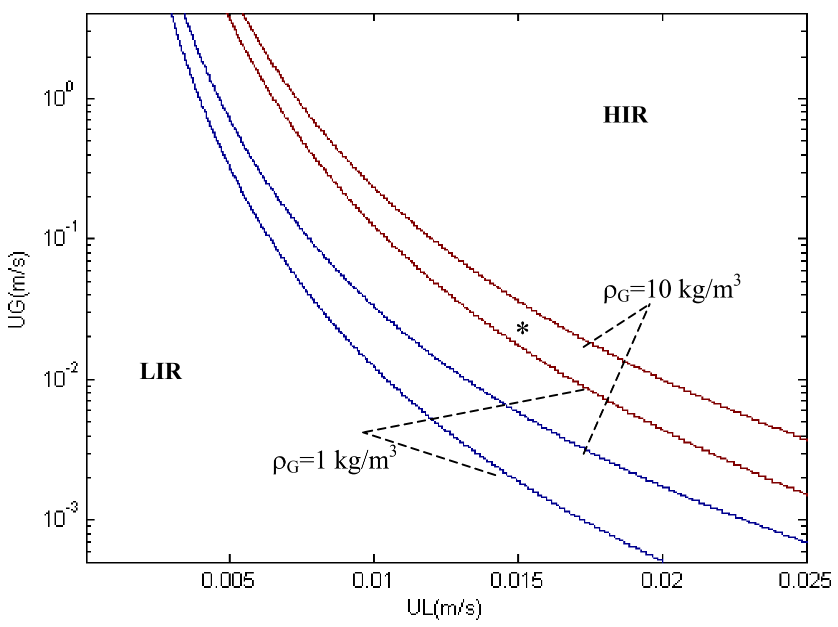


Figure 3.11 Incidence of gas density on regime classification decision boundaries obtained from continuous output monotonic MLP-C. The operating point highlighted with an asterisk (*) belongs to HIR class at $\rho_G = 1$ (located right of the TR/HIR border). At $\rho_G = 10$, it falls within TR band. The system's properties, with the exception of gas density, are the same as in Figure 3.10 above.

We close this chapter by making the reader aware of the fact that all the performance measures in Table 3.11 were obtained by 5-fold cross validation. At each fold, new random initialization of weights was done, in agreement with current recommendations for neural networks practice (Flexer, 1994; Prechelt, 1998).

3.2.5 Conclusions

Most of the studies done thus far on flow regime classification in trickle beds have focused on correlating the liquid superficial velocity demarcating the transition between low (LIR) and high (HIR) interaction regimes.

In this work, a conceptually different approach was taken, in which we modeled the class-conditional probabilities for the three flow patterns: LIR, TR, and HIR. Instead of formulating an unlikely sharp transition, a more physical TR band marking the gradual changeover between LIR and HIR was proposed. Instead of classical classification MLP neural networks, which use as many output neurons as classes, in this work we exploited class connectivity as *a priori* knowledge and encoded the output as a real variable, taking 0 at LIR, 0.5 at TR, and 1 at HIR. In doing so, misclassification between non-adjacent classes LIR and HIR was reduced significantly. Furthermore, monotonicity with respect to some variables, for which a priori knowledge was available, was mathematically guaranteed through inclusion of constraints. This was achieved by forcing the signs of the weights in the MLP neural network model, and training it with a genetic algorithm-genetic hill-climber optimizer and a constrained optimization algorithm. This facilitated identification of an MLP model having a misclassification error rate of 16.2%. The proposed classification MLP model, which incorporated prior knowledge of actual system behavior, is more interpretive than classical black-box neural correlations.

Table 3.12 Neural network flow regime classifier equations

$$H_j = \frac{1}{1+\exp\left(-\sum_{i=1}^{12} w_{ij}U_i\right)}$$

$$1 \le j \le 10 \qquad H_{10} = 1$$

$$\hat{y} = \frac{1}{1+\exp\left(-\sum_{j=1}^{10} w_j H_j\right)}$$

$$0 \le \hat{y} < 0.25 \Rightarrow LIR$$
$$0.25 \le \hat{y} < 0.75 \Rightarrow TR$$
$$0.75 \le \hat{y} \le 1 \Rightarrow HIR$$

$$U_1 = \frac{\log\left(\frac{u_L}{9.03\times10^{-6}}\right)}{4.275} \quad U_2 = \frac{\mu_G - 1.45\times10^{-5}}{5.2\times10^{-6}} \quad U_3 = \frac{\log\left(\frac{u_G}{4.98\times10^{-4}}\right)}{3.913} \quad U_4 = \frac{\sigma_L - 1.90\times10^{-2}}{6.0\times10^{-2}}$$

$$U_5 = \frac{D_C - 2.30\times10^{-2}}{4.90\times10^{-1}}$$

$$U_6 = \frac{\phi - 3.20\times10^{-1}}{6.80\times10^{-1}} \quad U_7 = \frac{\log\left(\frac{\mu_L}{3.10\times10^{-4}}\right)}{2.287} \quad U_8 = \frac{\log\left(\frac{\rho_G}{1.60\times10^{-1}}\right)}{2.862} \quad U_9 = \frac{\varepsilon - 2.63\times10^{-1}}{4.9\times10^{-1}} \quad U_{10} = \frac{\rho_L - 6.50\times10^{2}}{5.28\times10^{2}}$$

$$U_{11} = Foam \qquad U_{12} = 1$$

$$\begin{bmatrix} u_L \ge 9.03\times10^{-6} \\ u_L \le 1.74\times10^{-1} \end{bmatrix} \begin{bmatrix} \mu_G \ge 1.45\times10^{-5} \\ \mu_G \le 1.97\times10^{-5} \end{bmatrix} \begin{bmatrix} u_G \ge 4.98\times10^{-4} \\ u_G \le 4.08\times10^{0} \end{bmatrix} \begin{bmatrix} \sigma_L \ge 1.90\times10^{-2} \\ \sigma_L \le 7.62\times10^{-2} \end{bmatrix} \begin{bmatrix} D_C \ge 2.3\times10^{-2} \\ D_C \le 5.1\times10^{-1} \end{bmatrix} \begin{bmatrix} \phi \ge 3.2\times10^{-1} \\ \phi \le 1\times10^{0} \end{bmatrix}$$

$$\begin{bmatrix} \mu_L \ge 3.1\times10^{-4} \\ \mu_L \le 6.63\times10^{-2} \end{bmatrix} \begin{bmatrix} \rho_G \ge 1.60\times10^{-1} \\ \rho_L \le 1.16\times10^{2} \end{bmatrix} \begin{bmatrix} \varepsilon \ge 2.63\times10^{-1} \\ \varepsilon \le 7.50\times10^{-1} \end{bmatrix} \begin{bmatrix} \rho_L \ge 6.50\times10^{2} \\ \rho_L \le 1.18\times10^{3} \end{bmatrix} \begin{bmatrix} Foam = 0 & for\, coleascing \\ Foam = 1 & for\, foamig \end{bmatrix}$$

w_{ij}	**j→1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	1.986E+1	7.017E+0	8.264E+0	2.436E+1	2.784E+1	-9.777E+0	-2.778E+1	-3.131E+1	1.146E+1
2	-1.199E+1	-1.043E+1	-2.281E+1	3.409E+0	-6.482E-1	-7.566E-1	1.253E+1	-2.336E+1	-2.966E+1
3	1.218E+1	1.141E+1	1.829E-1	2.109E+1	5.134E+0	-3.507E+0	-2.182E+0	-2.622E+1	2.618E+1
4	-1.203E+1	-1.053E+1	-2.478E+1	-5.899E+0	-1.750E+0	1.642E+0	1.828E+1	-2.317E+1	-3.303E+1
5	-3.540E+0	-1.487E+0	-1.041E+1	-2.095E+0	-2.347E-1	3.805E-1	4.389E+0	-4.976E+0	-1.343E+1
6	-2.013E+1	-1.409E+1	-3.009E+1	7.866E+0	-4.172E-1	5.059E+0	2.690E+1	-2.999E+1	-3.844E+1
7	-6.814E+0	-5.277E+0	-1.588E+1	5.081E-1	3.128E+0	5.048E+0	2.257E+1	-1.193E+1	-2.168E+1
8	-1.617E+1	-9.679E+0	-2.385E+1	-3.235E+0	-5.575E-1	4.170E+0	4.126E+0	1.511E+1	-5.351E+0
9	-8.867E+0	-5.328E+0	-1.556E+1	1.758E+0	-3.030E+0	4.196E+0	1.343E+0	-1.105E+1	-2.171E+1
10	-9.832E+0	-8.898E+0	-2.098E+1	2.572E-2	-2.376E+0	-9.044E+0	1.752E+1	-1.995E+1	-2.550E+1
11	-5.335E+0	-4.379E+0	-9.935E+0	1.213E+1	2.067E-1	-6.252E+0	-2.585E+1	-1.240E+1	-1.833E+1
12	-2.607E+1	-1.578E+1	-3.874E+1	1.631E+1	-1.837E+1	-5.814E-2	1.474E+1	-3.864E+1	-4.358E+1

w_j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	1.943E+1	1.509E-1	5.710E-3	4.295E+1	6.556E+0	-2.789E+1	-2.725E+1	-7.936E+0	2.466E+1	-1.824E+1

*:A “user-friendly” spreadsheet of the neural correlation is accessible at (http://www.gch.ulaval.ca/bgrandjean)

3.3 Notation

a_G	Grain specific area
AR	Accuracy rate of a classifier *Ar=1-Err*
a_T	Bed specific area
C	Number of classes
CM	Confusion matrix having the size C×C
Cost	Cost matrix having the size C×C
d	Dimension of a subset of the set $\boldsymbol{X}_p$, d≤p
D_c	Column diameter
d_E	Euclidian distance
d_p	Particle diameter $d_P = 6(1-\varepsilon)/a_T$
Err	Misclassification rate, defined as the fraction of samples misclassified by a particular classifier.
Foam.	Foaming property: 0=coalescing, 1=foaming
$g_i(\mathbf{x})$	Discriminant function of the class y_i
H	Entropy function
H_0	Bed height at rest
$I(Y\|\boldsymbol{X}_s)$	Mutual information (information content) given by $\boldsymbol{X}_s$ on Y
J	Relevance criterion ($I(Y\|\boldsymbol{X}_s)$ or AR(1-NN)); Number of hidden nodes in an ANN
L	Loss function

n	Number of training instances ω_k; number of samples in the design set $D=\{(\mathbf{x}_r, y_i(\mathbf{x}_r)), r=1...n\}$
n_i	Number of class i occurrences in the design set D
nbx	Number of divisions in the space of each variable when assessing probabilities using bins
N_c	Number of classes
N_p	Number of particles per unit bed volume
p	Number of features available for a classification problem; i.e., size of $\boldsymbol{X}_p$
$p(y_i)$	Prior probability of y_i
$p(y_i \mid \mathbf{x})$	Posterior probability (probability of class y_i occurring, given $\mathbf{x}$)
$p(\mathbf{x} \mid y_i)$	Class-conditional density functions
$S_{ind}(i,k)$	Saliency value for the input i with respect to the output k in a ANN
u	Phase velocity
u_t	Particle terminal velocity in a three phase fluidized bed
$w[i,j]$	Input to hidden layer weight in the ANN
$w[j,k]$	Hidden to output layer weight in the ANN
$w_{i,j}$, w_j	ANN connectivity weights
$\boldsymbol{x}_p$	Particular realization of $\boldsymbol{X}_p$
$\mathbf{X}_p$	Set of all available features
$\boldsymbol{X}_s$	Subset of features from $\boldsymbol{X}_p$. $\boldsymbol{X}_s \subseteq \boldsymbol{X}_p$
$\boldsymbol{x}$	Point in the normalized input feature space $\mathbf{x}=\{u_L, \mu_G, u_G, \sigma_L, D_c, \phi, \mu_L, \rho_G, \varepsilon, \rho_L, \text{Foam}\}$

y	Particular value of the generic class variable *Y;* class variable taking the discrete values y_i, $i = 1, 2 \dots C$, $y_i \in \{LIR, TR, HIR\}$; continuous variable taking the values 0 for LIR, 0.5 for TR, and 1 for HIR
$\hat{y}$	Output of the neural network model
ε	Bed porosity
ϕ	Particle sphericity $\phi = \pi\left(\frac{6(1-\varepsilon)}{\pi N_P}\right)^{2/3} \times \frac{N_P}{a_T}$
μ	Phase viscosity
ρ	Phase density
σ	Phase superficial tension
ω_k	Training instance $\omega=\{\boldsymbol{x}_p, y\}$

Abbreviations

ANN	Artificial neural network (here, this term designates a multi-layer perceptron neural network)
FS	Feature selection
G	Gas
HIR	High interaction regime
IBC	Initial bed contraction
IBE	Initial bed expansion
L	Liquid

LIR	Low interaction regime
PK	Prior knowledge
S	Solid
SFFS	Sequential floating forward selection
SFS	Sequential forward selection
TR	Transition flow regime

Conclusion

In this work we addressed three key issues in neural networks modeling (regression and classification) of multiphase reactors data: i) feature selection (FS), ii) model design (MD) (architecture and parameters learning), and iii) qualitative prior knowledge matching or embedding (PK). We made methodological recommendations, which we validated by the resulting state-of-the-art neural network models. The Decision Makers' Direct (Issue No: 01/03/1) says that:

"The science of modeling involves converting domain reality- quantitative, and qualitative (like ethics, preferences, experience) to mathematical abstraction, using quantitative tools, and providing solutions as abstracted reality. The ultimate objective is to give quantitative expression to the decision maker's expertise."

In this light, quantitative domain reality consisted of the pairs of known input-output we had for training the model, while the qualitative domain reality was what we referred to as prior knowledge.

Now, a synthesis of the main results:

For the liquid hold-up in counter-current packed beds, a study was conducted to obtain dimensionless correlations that, in addition to giving a low estimate of the AARE (average absolute relative error), reveal monotonic trends with respect to six dimensional variables influencing liquid holdup: gas and liquid velocities and densities, as well as liquid viscosity and superficial tension. The subsidiary problematic we addressed here was how to select the appropriate dimensionless numbers to be used as network inputs. Many feature selection criteria exist, but we decided to evaluate the usefulness of features based on the error of the resulting model and the extent to which it matched prior knowledge in terms of monotonicity. Monotonicity was evaluated near the edges of the dimensional variables definition domain. A global error was defined by combining AARE on training and test sets with the number of monotonicity tests the model failed. This global error was minimized using a genetic algorithm whose operators were customized to search only among combinations with an imposed number of features. We maintained the model design (MD) typically used in the field: networks architecture determined by trial and error and weights learning with BFGS method. The conclusion of this first study was that such an automated procedure was efficient. The genetic algorithm was able to deal with the minimization of the nonmonotonic subset goodness criterion, identifying an elite of 5 dimensionless numbers (Bl_G, We_L, St'_L, K2, K3) which give an AARE of less than 13% on all data while matching all the necessary (but not sufficient) monotonicity conditions.

Using the dimensionless neural network modeling of pressure drop in randomly packed beds as a case study, we evaluated the effectiveness of the simple monotonicity tests we were performing near the edges of original variables ranges to ensure an overall monotonic behavior. We observed that a simple trends inspection in some points of the feature space was not necessarily representative of the model's monotonicity behavior likelihood in the entire domain. Therefore, we proposed gradient conditions checking in the vicinity of all the points available for training. Using the same methodology based on the genetic algorithm, but with reinforced gradient conditions, we identified a neural model useful for predicting the pressure drop in counter-current packed beds as $\frac{\Delta P / Z}{\rho_L g} = f\left(Bl_L, Fr_L, Eo_L, Eo'_L, K_1, S_B, \chi\right)$. The overall AARE of the model was approximately 20% with 127 parameters. Compared with the model of Piché *et al.*, (2001c) our model was superior, restoring the expected monotonic trends to almost 79% of the data points, compared with less than 20% for the former model.

As we were unable to decrease to zero the number of data points around which at least one of the monotonicity tests failed, we tried to combine several ANN models issued by the GA-ANN methodology. The point was to exploit the fact that there were different good features sets and architectures that lead to similar error rates. It was hoped that these models, being different, were not all bad in the same region of the original variables' input space. The outputs of each constituent model were weighted and converted into a new prediction via a linear meta-model. The resulting predictor not only showed an improved error rate, but also passed the monotonicity tests in more than 92% of the data points, compared with 79% for the single best individual model used.

A new aspect of this study was its dimensional approach. We tried to correlate the reactor's characteristics of interest directly to the original variables, rather than to dimensionless groups derived from them. As case study, the same liquid holdup problem was considered. The inputs were fixed to eleven original variables: u_G, u_L, ρ_G, μ_L, a_T, ε, ϕ,.Z, D_C, ρ_L, and σ_L. The recent work of Kay and Ungar (1993, 2000) has shown that is possible to guarantee monotonicity of the neural model if the variables with respect to which monotonicity is expected are directly fed into the neural network as inputs. Our contribution to their approach resides in the fact that we used concavity information (i.e., the signs of the second order derivatives with respect to some inputs). For concavity information matching, we used some necessary conditions, as the sufficient ones were unable to achieve via the weights' signs. The learning of the weights under monotonicity and concavity penalties was performed with a genetic algorithm-genetic hill climber optimizer, which proved superior to classic binary GAs. This evolutionary optimization algorithm combines classic genetic search and hill climbing in attractive regions of space. The approach efficiency was demonstrated by obtaining a neural model that guaranteed increasing liquid-holdup with increasing u_G, u_L, μ_L, $\sigma_{L,}$ and a_T or decreasing ρ_L, while the slope increased with u_G and decreased with u_L in some regions of the validity ranges. This model can be considered a contribution because it *is* less complex and more accurate than other neural and empirical correlations while matched all the priori knowledge considered in terms of mono-concavity.

A second group of applications considered in this dissertation was the supervised classification. As with the regression problems, feature selection was the first step to consider. We studied different feature selection algorithms, which allowed us to identify the most relevant variables in two multiphase reactors classification problems. Relevance here was assessed in terms of i) mutual information, which is a measure of the statistical dependence between feature subsets and the class variable, and ii) accuracy rate of a reliable one-nearest neighbor classifier that was simpler to operate than an ANN. A third relevance criterion, based on Garson's saliency index, interpreted the weights' size of trained neural network classifiers. We extended it to work with multiple outputs neural networks. The selection algorithms that targeted maximization of the relevance criteria (mutual information and accuracy rate) in charge of the combinatorial search were sequential forward selection and the (l,r) search method. We devised here a hybrid filter-wrapper approach, which in the first step used the mutual information criterion and SFS to obtain a set of class-informative features. This set was updated by using the (l,r) search to maximize the performance rate of a 1-NN classifier. This method is faster that a mere (l,r) search starting with an empty set, and gives the same results on the test problems. The different selection schemes were applied to four problems. The first two problems were benchmarks (synthetic and real) to test whether the schemes were able to find the proper solutions. The last two problems were real multiphase reactors: a) flow regime identification in trickle beds, and b) bed initial contraction/expansion in three-phase fluidized bed reactors. For both of these problems, the ULaval multiphase reactors databases were used to identify the variables (features) most relevant for classification. In all cases, the feature reduction induced an increase in classification performance, except for the bed expansion/contraction, where reducing the number of variables from 10 to 5 slightly decreased the accuracy. In the case of flow regime classification, the variables u_L, μ_G, u_G, σ_L, D_c, ϕ, μ_L, ρ_G, ε, ρ_L, and Foam were selected as informative about the flow regime. For bed expansion/contraction in three-phase fluidized beds, the most relevant features were ρ_s, σ_L, u_L, u_t, and H_0.

The final section of Chapter Three dealt with the classification issue itself, using as discriminant features those identified as relevant in this first step. We considered here only the flow regime classification in trickle bed reactors. Most work thus far in terms of empirical correlations for flow regime classification has focused on the correlation of the liquid superficial velocity at transition between low (LIR) and high (HIR) interaction regimes and the physical properties of the three phases involved. We used a conceptually different approach by using a neural network model to approximate the probability of each class occurring as a function of the phases' properties. Instead of obtaining a simple transition curve between classes LIR and HIR, we obtained a band, a physical manifestation of the gradual changeover in reality. The neural network model had an overall misclassification rate of about 16%, assessed by 5-fold cross-validation on 5061 data samples. While classic use of MLP neural network in classification assumes as many output neurons as classes, we used the class connectivity information as apriori knowledge and built a single output neural network classifier. Actually, we encoded the output as a real variable, taking 0 value at LIR, 0.5 at TR, and 1 at HIR. By so doing, the misclassifications between non-adjacent classes LIR and HIR were significantly reduced. Furthermore, the monotonicity of the interaction level between gas and liquid, with respect to the gas and liquid superficial velocities and gas density, was mathematically guaranteed. This was achieved by constraining the signs of the neural network model. The weights optimization was conducted with the genetic algorithm-genetic hill-climber optimizer designed previously. The resulting model was fine-tuned with a constrained optimization algorithm.

Following are some recommendations when building neural networks correlations for continuous (regression) or discontinuous (classification) reactors characteristics, assuming sufficient data records are available, and the important independent variables describing the three phases are available. This hypothesis was considered to hold in the cases studies we treated in his work. The importance of having his hypothesis true may be illustrated with the following popular quote in computer modeling:

"A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant." Eigen, Manfred (1927-), The Physicist's Conception of Nature

It is, however, not enough to have good data to obtain trustful models; we must also use any available qualitative information and select only the most relevant features. Recommendations follow:

- **Use dimensional variables as network inputs, rather than dimensionless groups formed by their nonlinear combinations**. Reasons for this include following: I) the performance of the resulting model is not significantly affected by input representation (dimensional or dimensionless); II) it is more difficult to identify a good set of dimensionless groups than to identify the dimensional variables relevant to the modeling task; III) when monotonicity of the model's output is expected with respect to some dimensional variables, it is impossible to guarantee it if nonlinear combinations of them (dimensionaless numbers) are used as inputs. Of course, we do not underestimate here the power of the dimensionless analysis, which allows extrapolation of the model's applicability outside the ranges of the raw variables within the training data base.

- **Determine a set of relevant features** using mutual information between sets and output variables or the accuracy of a custom model, while using as combinatorial search algorithms either sequential methods or genetic algorithms. Genetic algorithms are most suitable when the search space is large and a good guess of the number of variables to search for is available.

- **Use monotonicity and concavity prior knowledge, if available.** In regression problems, this squeezes the confidence band of the model (Kay and Ungar 1993, 2000) and reduces overfitting. Moreover, a guaranteed monotonic model is more interpretable. The interpretability increases because one will know in advance how the model's output will behave when the inputs vary. In classification problems, using information on connectivity of classes, as well as different misclassification costs, may reduce the chance of some particular types of misclassifications.

- **Compare the crossvalidated performance measure of the neural network classifiers** with classic statistical discriminants or decision trees to ensure that there is no simpler or more interpretable model which might perform as well as the neural network model.

References

Abu-Mostafa, Y. S. (1993). A method for learning from hints. *Advances in Neural Information Processing Systems*, 5, 73-80.

Acuna, G., Cubillos, F., Thibault, J. and Latrille E. (1999). Comparison of methods for training grey-box neural network models. *Computers and Chemical Engineering Supplement*, S561-S564.

Al-Dahhan, M., Larachi, F., Dudukovic, M.P., Laurent, A. (1997). High pressure trickle bed reactors: A review. *Industrial and Engineering Chemistry Research*, 36, 3292-3314.

Alpaydin, E. (1993). Multiple networks for function learning. In *Proc. IEEE Int. Conf. Neural Networks*, 1, 9-14, IEEE Press.

Alpaydin, E. (1998). Techniques for combining multiple learners. In *Proc. Eng. Intelligent Systems*, 2, 6-12, ICSC Press.

Anderson, E. (1935), The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, 59, 2–5.

Ash, R.B. (1990). *Information Theory*. Dover Publications, New York.

Batitti, R. (1994). Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4), 537-550.

Belsley, D.A. (1991). *Conditioning diagnostics: Colinearity and weak data in regression*. John Wiley & Sons, New York.

Benediktsson, J.A., Seveinsson, J.R., Ersoy, O.K., Swain, P.H. (1993). Parallel consentual neural networks. In *Proc. IEEE Int. Conf. Neural Networks*, 1, 27-32, IEEE Press.

Bensetiti, Z., Larachi, F., Grandjean, B. P. A., Wild, G. (1997). Liquid saturation in concurrent upflow fixed-bed reactors: a state-of-the-art correlation. *Chem. Eng. Sci.,* 52, 4239-4247.

Bhatia, V. K. and Epstein, N., (1974). Three phase fluidization: a generalized wake model, in *Fluidization and Its Applications*, in Angelino, et al., eds., Cepadues-Editions, Toulouse, 380-392.

Billet, R., Schultes, M. (1993). A Physical Model for the Prediction of Liquid Hold-up in Two-phase Countercurrent Columns. *Chem. Eng. Technol*., 16, 370-375.

Billet, R., Schultes, M. (1999). Prediction of Mass Transfer in Columns with Dumped and Arranged Packings. *Trans. IchemE.*, 77, 498-504.

Bishop, C.M. (1995). *Neural Networks for pattern Recognition*. Oxford: Clarendon Press.

Blanco, A., Delgado, M., Pegalajar, M.C. (2000). A Genetic Algorithm to obtain the Optimal Recurrent Neural Network. *Int. J. Approximate Reasoning*, 23, 67-83.

Branke, J. (1995). Evolutionary algorithms for neural network design and training. In *Proceedings 1st Nordic Workshop on Genetic Algorithms and its Applications*, Vaasa, Finland.

Brasquet, C., Lecloirec, P. (2000). Pressure Drop Through Textile Fabrics – Experimental Data Modelling Using Classical Models and Neural Networks. *Chem. Eng. Sci.*, 55, 2767-2778.

Breiman, L. (1992). *Stacked regressions*. Technical report 367, Department of Statistics, University of California, Berkeley, California 94720, USA.

Broomhead, D.S. and Lowe, D. (1988) Multi-variable functional interpolation and adaptive networks, Complex Systems, 2(3):269-303.

Carroll, D.L. (1996). Chemical Laser Modeling with Genetic Algorithms. *AIAA J.* 34, 338-346.

Charpentier, J. C. and Favier, M. (1975). Some liquid holdup experimental data in trickle bed reactors with foaming and nonfoaming hydrocarbons. *AIChE J.* 21, 1213.

Chen, S., Cowan, C.F.N.; Grant, P.M. (1991) Orthogonal least squares learning algorithm for radial basis function networks, *IEEE Transactions on neural networks* 2(2)**:** 302-309.

Cloutier, P., Tibirna, C., Grandjean, B. P. A., Thibault, J. (1996). NNfit, logiciel de régression utilisant les réseaux à couches, http://www.ulaval.ca/~nnfit.

Colquhoun-Lee, I., Stepanek, J. B. (1978). *Trans. Inst. Chem. Eng.*, 56, 136.

Côté, M., Grandjean, B.P.A., Lessard P. and Thibault J. (1995). Dynamic modeling of the activated sludge process: Improving prediction using neural networks. *Wat. Res*. 29(4), 995-1004.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 3, 303-314.

Daniels H. and Kamp B., (1998). Application of MLP Networks to Bond Rating and House Pricing. *Neural Comput. & Applic*., 8, 226-234.

De Jong, K.A. (1976) An Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD Thesis, University of Michigan, USA. Cited in Goldberg, (1989).

Dudukovic, M.P. and Mills P.L. (1986). Contacting and hydrodynamics in trickle-bed reactors. In *Encyclopedia of fluid mechanics*, Chereminisoff, Ed., Huston Gulf Publishing, 969-1017.

Dudukovic, M.P., Larachi, F., Mills P.L. (2002). Multiphase catalytic reactors: A perspective on current knowledge and future trends. *Catal. Rev. Sci. & Eng.*, 44, 123-246.

Flexer, A. (1994) Statistical evaluation of neural network experiments: Minimum requirements and current practice. Technical report, The Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010, Vienna, Austria.

Frantz, D. R. (1972). Non-linearities in Genetic Adaptive Search. Doctoral dissertation, University of Michigan, *Dissertation Abstracts International*, 33(11), 5240B-5241B. Cited in Goldberg, (1989).

Friese, T., Ulbig, P., Schultz, S. (1998). Use of Evolutionary Algorithms for the Calculation of Group Contribution Parameters in order to Predict Thermodynamic Properties. *Computers Chem. Eng.* 22, 1559-1572.

Fukushima, S., and Kusaka K., (1978) Boundary of hydrodynamic flow region and gas phase mass transfer coefficient in packed column with concurrent downward flow, J. of Chem. Eng. Japan, 11, 241

Gao, F., Li, M., Wang, F., Wang, B., Yue, P.L. (1999). Genetic Algorithms and Evolutionary Programming Hybrid Strategy for Structure and Weight Learning for Multi-layer Feed-forward Neural Networks. *Ind. Eng. Chem. Res.* 38, 4330-4336.

Garson, G. D. (1991). Interpreting Neural Network Connection Weights. *AI Expert*, 6(7), 47-51.

Gencay, R., Qi M. (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging, *IEEE Transactions on Neural Networks*, 12(4) (2001), 726-734.

Gianetto, A., Baldi, G., Specchia, V., Sicardi, S. (1978). Hydrodynamics and solid-liquid contacting effectiveness in trickle-bed reactors. *AIChE J.* 24, 1087.

Goldberg, D.E. (1989). Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley, Reading, MA.

Hashem, S. (1997). Optimal linear combinations of neural networks. *Neural Networks*, 10, 599.

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: The University of Michigan Press. Cited in Goldberg, (1989).

Holub, R. A., Dudukovic, M. P., Ramachandran, P. A. (1992). A phenomenological model for pressure drop, liquid holdup, and flow regime transition in gas-liquid trickle flow, *Chem. Eng. Sci.* 47(9-11), 2343-2348.

Holub, R.A., Dudukovic, M.P., Ramachandran, P.A. (1993) Pressure drop, liquid holdup, and flow regime transition in trickle flow AIChE Journal, 39(2), 302-321.

Hornik, K. (1990). Approximation capabilities of multilayer feedforward neural networks, *Neural Networks*, 4, 251-257.

Iliuta, I., Larachi, F., Grandjean, B. P. A. (1998). Pressure Drop and Liquid Holdup in Trickle Flow Reactors: Improved Ergun Constants and Slip Correlations for the Slit Model. *Ind. Eng. Chem. Res.* 37, 4542-4550.

Iliuta, I., Larachi, F., Grandjean, B.P.A., Wild, G. (1999a). Gas-liquid Interfacial Mass Transfer in Trickle-bed Reactors: State-of-art Correlations. *Chem. Eng. Sci.* 54(23), 5633-5645.

Iliuta, I., Ortiz, A., Larachi, F., Grandjean, B. P. A., Wild, G., (1999b), Hydrodynamics and mass transfer in trickle-bed reactor: an overview, *Chem. Eng. Sci.*, 54(21), 5329-5337.

Iliuta, I., Larachi, F., Grandjean, B. P. A., Wild, G. (1999c). ECCE2, Montpelier, France.

Iliuta, I., Muntean, O., Iliuta, M. C., Larachi, F. (1999d). *Reactoare Multifazice Vol (I-II),* Printech Bucharest.

Jain, A. K., Duin P.W., and Mao. J. (2000). Statistical pattern recognition: A Review. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1), 4-37.

Jamialahmadi, M., Zehtaban, MR., Muller-Steinhagen, H., Sarrafi, A., Smith, J.M. (2001). Study of Bubble Formation under Constant Flow Conditions. *Trans. IchemE*, 79, 523-532.

Jean, R. and Fan L.S. (1986). A simple correlation for solids holdup in a gas-liquid-solid fluidized bed, *Chem. Eng. Sci*. 41(11), 2823-2828.

Jiang P., Luo X., Lin T., Fan L. (1997). High temperature and high pressure three-phase fluidization-bed expansion phenomena. *Powder Technology,* 90, 103-113.

John G.H., Kohavi R., Pfleger K. (1994). Irrelevant features and subset selection problem, *Proceedings of the 11th Int. Conf. On Mach. Learning*, 121-129.

Kay, H. and Ungar L.H. (1993). Deriving Monotonic Function Envelopes from Observations. *Working Papers from the Seventh International Workshop on Qualitative Reasoning about Physical Systems*, Orcas Island, Washington, 117.

Kay, H., Ungar, L.H. (2000). Estimating monotonic functions and their bounds. *AIChE J. 46*, 2426.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection, *Proceedings of the 15th Int. Joint Conf. On Artif. Intell.*, 1137-1143.

Krink, T. and Løvbjerg, M. (2002). The LifeCycle model: Combining Particle Swarm Optimization, Genetic Algorithms and HillClimbers. *Proceedings of Parallel Problem Solving from Nature VII (PPSN-2002),* 621.

Larachi, F., Laurent, A., Wild, G. and Midoux, N. (1993). *Can. J. Chem. Eng*. 71, 319.

Larachi, F., Bensetiti, Z., Grandjean, B. P. A., Wild, G. (1998). Two-phase frictional pressure drop in flooded-bed reactors: A state-of-the-art correlation. *Chem. Eng. Technol*., 21(11), 887-893.

Larachi, F., Iliuta, I., Chen, M., Grandjean, B.P.A. (1999). Onset of pulsing in trickle beds: evaluation of current tools and state-of-the-art correlation, *Can. J. Chem. Eng.*, 77, 751-758.

Larachi, F., Belfares, L., Iliuta, I., Grandjean, B.P.A. (2001). Three-phase Fluidization Macroscopic Hydrodynamics Revisited. *Ind. Eng. Chem. Res.* 40, 993-1008.

Lanouette R., Thibault J., Valade J.L. (1999). Process modeling with neural networks using small experimental datasets, *Computers and Chemical Engineering*, 23, 1167-1176.

Leva, M. (1953). *Tower packings and packed tower design*, 2nd Edition, United States Stoneware Company, Akron.

Li, R., Mukaidono, M., Turksen, I. B. (2002). A fuzzy neural network for pattern classification and feature extraction. *Fuzzy Sets and Systems*, 130, 101-108.

Lohl, T., Schultz, C., Engell, S. (1998). Sequencing of Batch Operations for Highly Coupled Production Process: Genetic Algorithms Versus Mathematical Programming. *Computers Chem. Eng.* 22, S579-S585.

Lowe D. and Webb A.R. (1990) "Exploiting knowledge in network optimization: An illustration from medical prognosis", Network: Computation in Neural Systems, 1, 299-323.

Maćkowiak, J. (1991). Pressure drop in irrigated packed columns. *Chem. Eng. Process.*, 29, 93-105.

Maren, A.J., Harston, C.T. and Pap, R.M. (1990). *Neural Computing Applications*. Academic Press, Inc., San Diego.

McLachlan, G.J. (1992) Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York.

McLoone, S., Irwin, G. W. (1997). Fast parallel off-line training of multilayer perceptrons, *IEEE Transactions on Neural Networks*, 8(3), 646-653.

Morshed, J., Kaluarachchi, J. (1998) Application of Neural Network and Genetic Algorithm in Flow and Transport Simulations. *Adv. Water Resour.* 22, 145-158.

Morshed, J., Powers, S.E. (2000). Regression and Dimensional Analysis for Modeling Two-Phase Flow. *Transport Porous Med*. 38, 205-221.

Narendra P.M. and Fukunaga K. (1977). A branch and bound algorithm for feature subset selection. *IEEE Transactions on Computers*, C-26(9), 917-922.

Nath R., Rajagopalan B., and Ryker R. (1997). Determining the saliency of input variables in neural network classifiers. *Computers and Operations Research*, 24(8), 767-773.

O'Neil, T.J. (1992) Error rates of non-Bayes classification rules and robustness of Fisher's linear discriminant function. *Biometrika*, 79(1):177-184.

Perrone, M. P. (1993) *Improving regression estimation: Averaging methods for variance reduction with extensions to general convex measure optimization*. PhD Thesis, Department of Physics, Brown University.

Piché, S., Larachi, F., Grandjean, B.P.A. (2001a). Flooding Capacity in Packed Towers: Database, Correlations and Analysis. *Ind. Eng. Chem. Res.* 40, 476-487.

Piché, S.; Grandjean B.P.A., Iliuta I., Larachi F. (2001b). Interfacial Mass Transfer in Randomly Packed Towers: A Confident Correlation for Environmental Applications. *Environ. Sci. Technol.*, *35*, 4817-4822.

Piché, S., Larachi F., Grandjean, B.P.A. (2001c). Loading capacity in packed towers-Database, correlations and analysis. *Chem. Eng. Technol*., 24, 373-380.

Piché, S., Larachi F., Grandjean, B.P.A. (2001d). Improving the prediction of irrigated pressure drop in packed absorption towers, *CJChE*, 79(4), 584.

Piché, S., Larachi F., Grandjean, B.P.A. (2001e), Improved liquid hold-up correlation for randomly packed towers, *Trans IChemE.*, 79, 71.

Pollock, G.S., Eldridge, R.B. (2000). Neural Network Modeling of Structured Packing Height Equivalent to a Theoretical Plate. *Ind. Eng. Chem. Res.* 39, 1520-1525.

Potter, M.A., De Jong, K.A. (1994). A Cooperative Coevolutionary Approach to Function Optimization. In Y. Davidor, H.-P. Schwefel, and R. Manner (Eds.), *Parallel Problem Solving from Nature (PPSN III)*, 249.

Prechelt, L. (1998). Automatic early stopping using cross validation: quantifying the criteria, Neural Networks, 11(4) 761-767.

Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T. (1989). *Numerical recipes: The art of scientific computing*. Cambridge Univ. Press, Cambridge.

Pudil, P., Ferri, F.J., Novovicova, J., Kittler, J. (1994). Floating Search Methods for Feature Selection with Nonmonotonic Criterion Functions, *Proceedings - International Conference on Pattern Recognition*, 2, 279-283.

Quinlan J.R. (1993) *C4.5: Programs for Machine Learning*. San Mateo, Calif.: Morgan Kaufmann.

Ramachandran, P. A. and Chaudhari, R. V. (1983). *Three-Phase Catalytic Reactors*, Gordon and Breach Science Publisher, USA.

Rey-Fabret, I., Sankar, R., Duret, E., Heintze, E., Henriot, V. (2001). Neural Network Tools for Improving Tacite Hydrodynamic Simulation of Multiphase Flow Behavior in Pipelines. *Oil & Gas Science and Technology-Revue de l'Institut Français du Pétrole*, 56, 471-478.

Rumelhart, D.E., Hinton, G. and Williams, R. (1986). Learning internal representation by error propagation, *Parallel Distributed Processing*, 1, MIT Press, 318-364.

Sato, Y., T. Hirose, F. Takahashi, Toda, M. and Hashiguchi, Y. (1973). Flow pattern and pulsation properties of concurrent gas-liquid downflow in packed beds. *J. Chem. Eng. Japan,* 6, 313-319.

Sai, P. S. T. and Varma, Y. B. G., (1988), *Can. J. Chem. Eng.* 66, 353.

Schaffer J. D., Whitley D. and Eshelman, L. J. (1992) Combinations of genetic algorithms and neural networks: A survey of the state of the art. In L. D. Whitley and J. D. Schaffer, editors, *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, 1.

Sebban M., Nock R. (2002). A hybrid filter/wrapper approach of feature selection using information theory, *Pattern Recognition*, 35, 835-846.

Shannon, C. E. and Weaver, W. (1949). *The mathematical Theory of communication*. University of Illinois Press, Urbana.

Siedlecki, W. and Sklansky, J. (1988). On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, 2(2), 197-220.

Sill, J. (1998). Monotonic networks. *Adv. Neural Inf. Process.*, 10, 661.

Sill, J. and Abu-Mostafa, Y. S. (1997). Monotonicity Hints. *Advances in Neural Information Processing Systems*, 9, 634-640.

Silvey, F. C. and Keller, G. J. (1966). *Chem. Eng. Progr.* 62(1), 68.

Soung, W. Y. (1978). Bed Expansion in three-phase fluidization, *Ind. Eng. Chem. Process Des. Dev.* 17(33).

Sridhar, D.V., Bartlett, E.B. and Seagrave, R.C. (1998). Information theoretic subset selection for neural network models. *Computers Chem. Engng,* 22(4-5), 613-626.

Stearns, S.D. (1976). On selecting features for pattern classifiers. *In Third Int. Conf. On Pattern recognition*, 71-75, Colorado, CA.

Steppe, J.M. (1994). *Feature and model selection in feedforward neural networks*, PhD Dissertation at Air Force Institute of Technology, Ohio.

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2002). Integrated genetic algorithm - artificial neural network strategy for modeling important multiphase-flow characteristics, *Industrial and Engineering Chemistry Research*, 41(10), 2543-2551.

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2003a). Reinforcing the phenomenological consistency in artificial neural network modeling of multiphase reactors, *Chemical Engineering and Processing*, 42, (8-9), 653-662.

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2003b). Artificial Neural Network Meta Models To Enhance the Prediction and Consistency of Multiphase Reactor Correlations, *Industrial and Engineering Chemistry Research*, 42(8), 1707-1712.

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2003c). Feature selection for multiphase reactors data classification. (*Chemical Engineering Science*, submitted).

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2004a). Embedding monotonicity and concavity information in the training of multiphase flow neural network correlations by means of genetic algorithms, *Computers and Chemical Engineering*, in press.

Tarca, L. A., Grandjean, B. P. A., Larachi, F. (2004b). Designing supervised classifiers for multiphase flow data classification. *Chemical Engineering Science*, Accepted.

Tetko, I.V., Villa A.E.P. (1997). An enhancement of generalization ability in cascade correlation algorithm by avoidance of overfitting/overtraining problem. *Neural Processing Letters*, 6(1-2) 43-50.

Thibault, J. and Grandjean, B.P.A. (1990). A neural network methodology for heat transfer data analysis. *Int. J. Heat Mass Transf.*, 34, 2036-2070.

Turpin, J. L., Huntington, R. L. (1967). *AIChE J.* 13, 1196.

Turpin, J.L., Huntington R.L. (1967). Prediction of pressure drop two-phase, two component concurrent flow in packed beds, *AIChE J.*, 13, 1196-1202.

Ueda, N. (2000). Optimal linear combination of neural networks for improving classification performance. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 22, 207.

Viennet, R., Fonteix, C. and Marc, I. (1995). New Multicriteria Optimization Method Based on the Use of a Diploid Genetic Algorithm: Example of an Industrial Problem. *Lecture Notes in Computer Science* Artificial Evolution, Alliot, J.-M., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (Eds) Springer, 1063*,* 120-127.

Wang, R., Mao Z. S., and Chen J. Y. (1994). A study of trickle-to-pulse flow transition in trickle-bed reactors (TBR), *Chem. Eng. Commun*. 127, 109-124.

Wang, S. (1996). Learning monotonic-concave interval concepts using the back-propagation neural networks. *Computational Intelligence*, 12, 260.

Webb, A., (2002) *Statistical Pattern Recognition*, 2nd edition, John Wiley and Sons Ltd.

Whaley, A.K., Bode, C.A., Ghosh, J.G., Eldridge, R.B. (1999). HETP and Pressure Drop Prediction for Structured Packing Distillation Columns Using a Neural Network. *Ind. Eng. Chem. Res.*, 38, 1736-1739.

Whitley, D. (1989). The GENITOR Algorithm and Selective Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In *Proc. 3th International Conf. on Genetic Algorithms.* 116, D. Schaffer, ed., Morgan Kaufmann.

Whitley, D. (1995). Genetic Algorithms and Neural Networks. *Genetic Algorithms in Engineering and Computer Science.* G. Winter, J. Periaux, M. Galan and P. Cuesta, eds. 203-216, John Wiley.

Whitley, D., Dominic, S. and Das, R. (1991). *Genetic reinforcement learning with multilayer neural networks*. In Belew, R. K., and Booker, L. B., eds, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 562, Morgan Kaufmann.

Wolpert, D.H. (1992). Stacked generalization. *Neural Networks*, 5, 241.

Yang, H., Fang, B.S., Reuss, M. (1999). k_La Correlation Established on the Basis of a Neural Network Model. *Can. J. Chem. Eng.*, 77, 838-843.

Zamankhan, P., Malinen, P., Lepomaki, H. (1997). Application of Neural Networks to Mass Transfer Predictions in a Fast Fluidized Bed of Fine Solids. *AIChE J.* 43, 1684-1690.

Zhou, J. (1998). *Using genetic algorithms and artificial neural networks for multisource geospatial data modeling and classification*, PhD Dissertation University of Connecticut.

Appendix 1

Pseudo algorithm for PCE computation of a trained ANN

```
1. indexPoint=0; (counts the points around which the model fulfills all the gradient conditions eqs. 1-5)
2. For each point $\mathbf{p_k}$ of the training data set, k = (1..$N_T$), $\mathbf{p_k} = \{u_G,u_L,\rho_G,\mu_L,a_T,\varepsilon,\phi,Z,D_C,\rho_L,\sigma_L,\mu_G\}$:
   • indexVar=0. (counts how many among eqs. 1-5 will be satisfied around point k)
   • For each testing variable $v_j \in \{U_G,U_L,\rho_G,\mu_L,a_T\}$, i.e., each j, j=1..5:
       o Calculate the maximum increment $\Delta$ which, when added to or subtracted from $p_{k,j}$, still yields all the numbers $N_1$ to $N_m$ in the validity ranges. (determine a validity range for the variable j)
       o Calculate $N_1$ to $N_m$ for $\mathbf{p_k}$, $\mathbf{p_{k,j}}^{+\Delta}$ , $\mathbf{p_{k,j}}^{-\Delta}$ (the inputs of the ANN in three points)
       o If $y^{(calc)}(\mathbf{p_{k,j}}^{-\Delta}) \le y^{(calc)}(\mathbf{p_k}) \le y^{(calc)}(\mathbf{p_{k,j}}^{+\Delta})$ (ANN model presents monotonically increasing trend with respect to $v_j$)
           ▪ indexVar=indexVar+1 (ANN model has passed the test for the variable $v_j$ in point $\mathbf{p_k}$)
   • If indexVar=5 (ANN model is phenomenologically consistent in the vicinity of point $\mathbf{p_k}$)
       o indexPoint= idenxPoint+1
3. Return the phenomenology consistence error PCE as (1-indexPoint/$N_T$)*100.
```

Appendix 2

Proof that monotonic neural networks with respect to dimensional variable v_s, may not be guaranteed if the network's inputs are functions of the variables v_s.

For illustration, let us assume that the liquid holdup ε_l is a function of two dimensionless numbers constituting the neural model inputs:

$x_1 = N_1 = \dfrac{\rho_L \cdot u_L}{a_T \cdot \mu_L}$ and $x_2 = N_2 = \dfrac{\mu_L \cdot u_L \cdot a_T^{\;2}}{\rho_L \cdot g}$, i.e., liquid Reynolds and liquid Stokes numbers, respectively.

Let us assume that the network function *f(w,x)* exhibits decreasing non-strict monotonicity with respect to variable $v_S=\rho_L$: $\dfrac{\partial f}{\partial \rho_L} \le 0$.

Computing the neural network function in this particular case gives:

$$f(x,w) = \sigma\left(\left(\sum_{j=1}^{J} w_j \sigma\left(\frac{\rho_L \cdot u_L}{a_T \cdot \mu_L} \cdot w_{1,j} + \frac{\mu_L \cdot u_L \cdot a_T^{\;2}}{\rho_L \cdot g} \cdot w_{2,j} + w_{I+1,j}\right)\right) + w_{J+1}\right)$$

and

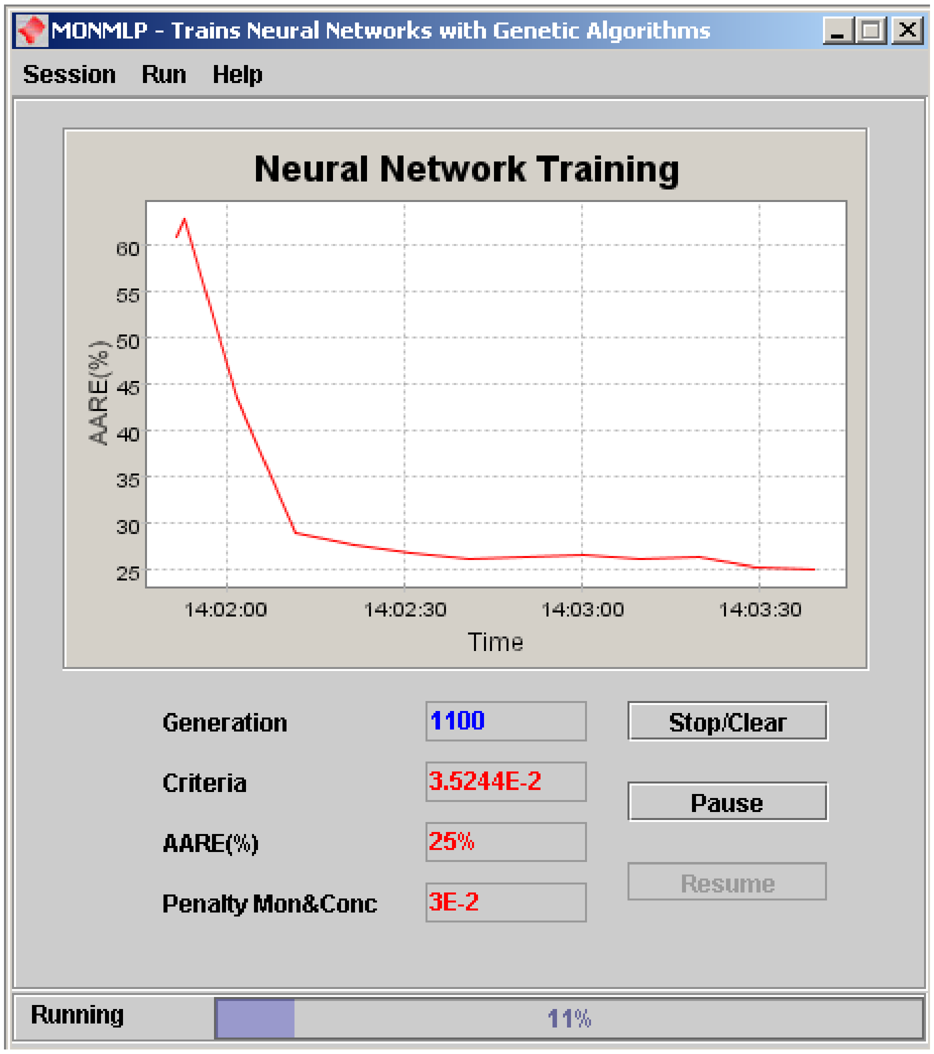
$$\frac{\partial f}{\partial \rho_L} = D(\sigma)\left(\left(\sum_{j=1}^{J} w_j \cdot \sigma\left(\frac{\rho_L^{\;2} u_L g w_{1,j} + \mu_L^{\;2} u_L a_T^{\;3} w_{2,j}}{a_T \mu_L \rho_L g} + w_{I+1,j}\right)\right) + w_{J+1}\right) \cdot u_L \cdot$$

$$\left(\sum_{j=1}^{J} w_j \cdot D(\sigma)\left(\frac{\rho_L^{\;2} u_L g w_{1,j} + \mu_L^{\;2} u_L a_T^{\;3} w_{2,j}}{a_T \mu_L \rho_L g} + w_{I+1,j}\right) \cdot \left(\rho_L^{\;2} g w_{1,j} - \mu_L^{\;2} a_T^{\;3} w_{2,j}\right)\right) / \left(\mu_L a_T \rho_L^{\;2} g\right)$$

The sign of this expression is a function of the data point **x** where it is evaluated (the properties of the liquid phase and bed); and it may be judged on factors other than weight signs.

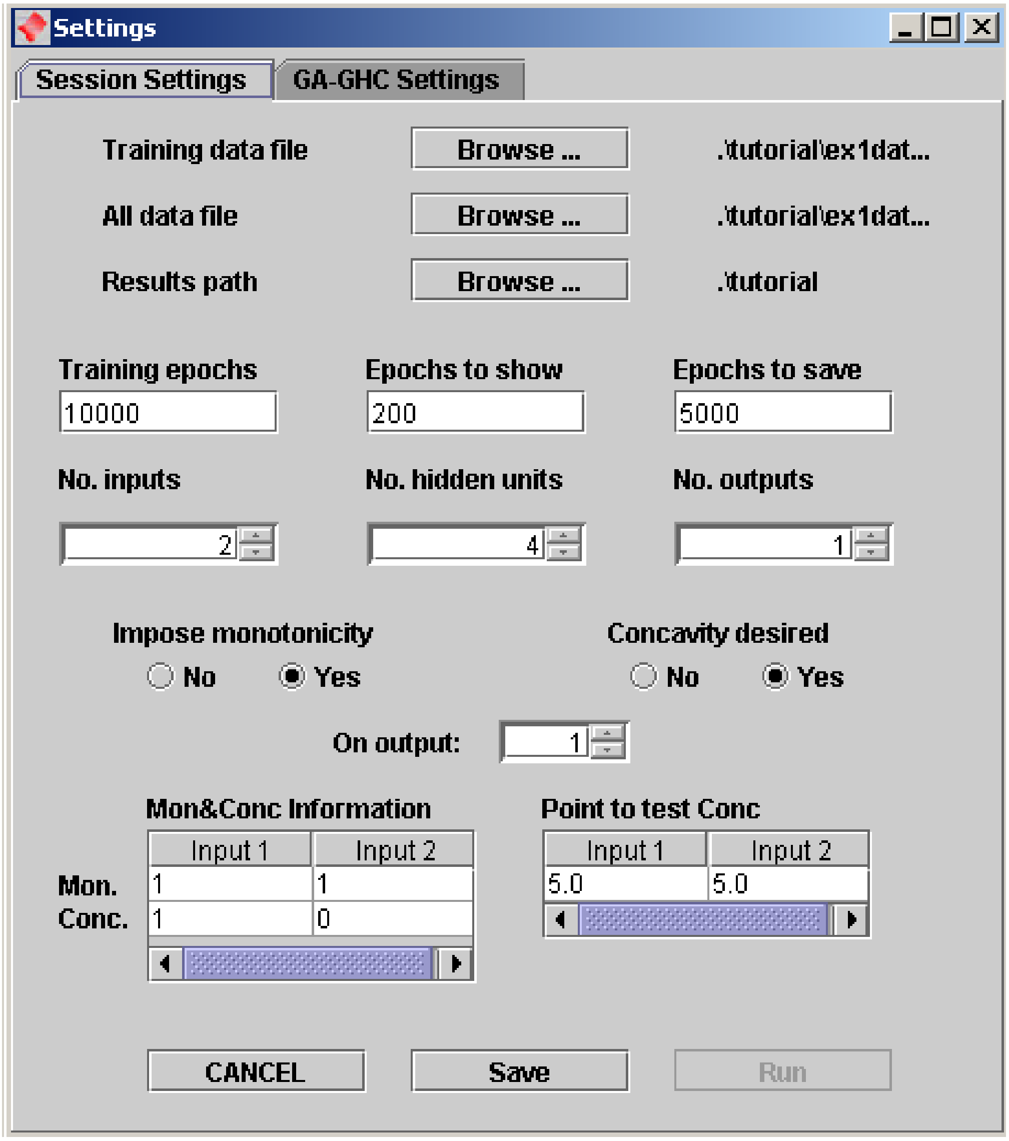
Appendix 3

The user interface for the MONMLP software developed in the JAVA language to carry out the training of monotonic ANNs with genetic algorithms. This software may be downloaded from http://www.gch.ulaval.ca/~grandjean.



Generation 1100
Criteria 3.5244E-2
AARE(%) 25%
Penalty Mon&Conc 3E-2

Main window of MONMLP, plotting the evolution of the AARE of the model in time. The generation, the composite criterion, and the penalty terms for monotonicity and concavity are also displayed.



Settings

Session Settings | GA-GHC Settings

Training data file	Browse ...	.\tutorial\ex1dat...
All data file	Browse ...	.\tutorial\ex1dat...
Results path	Browse ...	.\tutorial

Training epochs	Epochs to show	Epochs to save
10000	200	5000

No. inputs	No. hidden units	No. outputs
2	4	1

Impose monotonicity: No [✓ Yes]

Concavity desired: No [✓ Yes]

On output: 1

Mon&Conc Information

	Input 1	Input 2
Mon.	1	1
Conc.	1	0

Point to test Conc

Input 1	Input 2
5.0	5.0

The Session Settings window, where the user chooses the data set, network architecture, and monotonicity and concavity expected in the data, as well as some tuning parameters of the GA optimizer.

Appendix 4

The (*l,r*)-search algorithm, or its particular cases (1,0)-search (or SFS) and (0,1)-search (or SBS), as described by Pudil et. al (1994, and adapted for the current work notations.

```
Input:
$\mathbf{X}_p=\{X_{p,i} \mid i=1,\ldots, p\}$ //the set of all features//
Output:
$\mathbf{X}_{s,d}=\{X_{s,i} \mid i=1,\ldots, d, X_{s,i} \in \mathbf{X}_p\}$ , $d=0,1,\ldots, p$; //the selected subset of size $d$//
Initialization:
if $l>r$ then $d:=0$; $\mathbf{X}_{s,d} = \phi$; go to Step 1
        else $d:=p$; $\mathbf{X}_{s,d} = \mathbf{X}_p$; go to Step 2
Termination:
        Stop when $d$ equals the number of features required
Step 1 (Inclusion)
Repeat l times
    $X^+ := \arg \max_{X \in \mathbf{X}_p - \mathbf{X}_{s,d}} J(\mathbf{X}_{s,d} + X)$ //the most significant feature with respect to $\mathbf{X}_{s,d}$ //
    $\mathbf{X}_{s,d+1} := \mathbf{X}_{s,d} + X^+$; $k:=k+1$;
Step 2 (Exclusion)
repeat r times
    $X^- := \arg \max_{X \in \mathbf{X}_{s,d}} J(\mathbf{X}_{s,d} - X)$ //the least significant feature in $\mathbf{X}_{s,d}$ //
    $\mathbf{X}_{s,d-1} := \mathbf{X}_{s,d} - X^-$; $k:=k-1$
go to step 1
```