University of Coimbra
Faculty of Sciences and Technology
Department of Electrical and Computer Engineering

Francisco Alexandre Andrade de Souza

Computational Intelligence Methodologies for Soft Sensors Development in Industrial Processes

Tese de Doutoramento em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação e Robótica, orientada pelo Prof. Dr. Rui Alexandre de Matos Araújo e apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

February 2014

Universidade de Coimbra

University of Coimbra
Faculty of Sciences and Technology
Department of Electrical and Computer Engineering

Computational Intelligence Methodologies for Soft Sensors Development in Industrial Processes

by

Francisco Alexandre Andrade de Souza

Tese de Doutoramento em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação e Robótica, orientada pelo Prof. Dr. Rui Alexandre de Matos Araújo e apresentada ao Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

February 2014

Dedico esta tese à minha família, e a Dona Mazé, minha avó de coração.

"O louco que reconhece sua loucura possui algo de prudente; porém, o louco que se presume sábio esse está realmente louco."
Sakyamuni

Acknowledgements

Gostaria de agradecer às varias pessoas que, de diferentes formas, contribuíram para a realização desta tese. O meu reconhecimento e gratidão ao Professor Dr. Rui Araújo pelo suporte, orientação e conhecimentos transmitidos no decorrer destes 5 anos de doutoramento. O meu profundo agradecimento ao Mestre Carlos Alberto Coelho Belchior, meu amigo desde o tempo de graduação, que me incentivou à fazer o doutoramento na Universidade de Coimbra. O meu profundo agradecimento a equipe da Acontrol, em especial ao Mestre Pedro Sousa. E um agradecimento ao Instituto de Sistemas e Robótica (ISR-UC) por permtir e apoiar realização da minha tese.

Agradeço à Fundação para a Ciência e a Tecnologia (FCT) que suportou esta tese através de uma bolsa de doutoramento com a referência SFRH/BD/63454/2009. Este trabalho foi também suportado pelo Projecto SInCACI "Sistemas Inteligentes de Controlo, Aquisição e Comunicação Industrial" (referência: SInCACI/3120/2009), e pelo Projecto SCIAD "Sistemas de Controlo Industrial Auto-Aprendizes através de Dados do Processo" (referência: SCIAD/2011/21531), ambos co-financiados pelo QREN, no âmbito do "Mais Centro - Programa Operacional Regional do Centro", e pela União Europeia através do Fundo Europeu de Desenvolvimento Regional (FEDER), bem como pela Agência de Inovação (AdI) no caso do projecto SInCACI. Agradeço a estes projectos.

Aos meus pais, Sônia Maria e Francisco Filho, e meu irmão, Francisco Emmanoel, que me ajudaram de todas as formas no decorrer do meu curso de graduação e durante o doutoramento. Em especial a minha mãe, que sempre foi uma fonte de inspiração, ao sempre mostrar que nunca se deve desistir, mesmo quando existem todos os motivos para isto. Ao meu pai e irmão, que sempre se mostraram disponíveis para me ajudar. Um agradecimento especial aos amigos da família que contribuiram indiretamente ao desenvolvimento desta tese, em especial ao Aristoteles e Daniel

Boris.

Aos meus amigos pelos excelentes momentos passados durante o meu percurso académico em Coimbra. Aos meus colegas e amigos do ISR, pela união e amizade, em especial ao Cristiano Premebida, Jérôme Mendes, Ricardo Maia, Carlos Belchior, Sérgio Sousa, Teresa Sousa, Omar Mahjoubi, Dulce Gabriel, Luís Maricato, Jorge Ribeiro, Tiago Matias, Rita Joana, Simone Rodrigues, Symone Soares, Ana Paula, Diana Guardado, Saeid Rastergar, Oswaldo Ludwig e Diego Faria. Aos meus amigos do IPMMPA, que geraram momentos excepcionais e inesquecíveis.

Aos meus amigos do Brasil e de Coimbra, que contribuiram de forma indireta ao desenvolvimento desta tese, em especial ao João Chamusqueiro, Luis Cruz e José Barata, aos Talesianos (D.M., F.E., D.S., A.C., F.A.), Diego Mendes e Daniel Soares, e aos amigos da UFC, em especial ao Dalton, Antônio Barbosa e Henrique Innecco. Ao professores da UFC, Ricardo Thé, Laurinda, Otacílio, René Bascopé pelo exemplo de profissionalismo e amor à Engenharia Elétrica.

Um obrigado especial, à minha namorada Sara Ribeiro pelo seu apoio durante o desenvolvimento desta tese.

Abstract

Data-driven soft sensors are inferential models that use on-line available sensors (e.g. temperature, pressure, flow rate, etc) to predict quality variables which cannot be automatically measured at all, or can only be measured at high cost, sporadically, or with high delays (e.g. laboratory analysis). Soft sensors are built using historical data of the process, usually provided from the supervisory control and data acquisition (SCADA) system or obtained from the laboratory annotations/measurements. In the soft sensor development, there are many issues to deal with. The main issues are the treatment of missing data, outliers detection, selection of input variables, model training, validation, and soft sensor maintenance. This thesis focuses on three of these issues, namely the selection of input variables, model training, and soft sensor maintenance. Novel methodologies are proposed in each of these areas. The selection of input variables is based on the multilayer perceptron (MLP) neural network model (the most popular non-linear regression model in soft sensors applications). The second issue, the model training, is addressed in the context of multiple operating modes. Examples of multiple operating modes are diurnal load variation of a power plant, summer-winter operation of a refinery, etc. In this thesis, to train a model in the context of multiple modes context, the partial least squares regression (PLS), a well know method in the chemometrics literature and one of the mostly used methods in industry, is inserted into the mixture of experts (ME) framework, deriving so the mixture of partial least square (Mix-PLS) regression. The Mix-PLS is able to characterize multiple operating modes. The third problem is related to soft sensor maintenance. In soft sensor maintenance, the model is updated using recent samples of the process. The most common way to do so is by the exponentially recursive learning of parameters, using the incoming samples of the process. In exponentially recursive learning, a forgetting factor is used to give exponentially less weight to older samples. In many applications, small values of the forgetting factor

can lead to better predictive performance. However, the forgetting factor is directly related to the "effective" number of samples, and low values of forgetting factor can bring the same problem faced when modeling static systems, such as overfitting, poor prediction performance, etc. To solve this problem, a new model, based on the mixture of univariate (thus low dimensional) linear regression models is proposed (MULRM), allowing the use of small values of forgetting factor. All the methods proposed in this thesis are evaluated in soft sensors data sets coming from real-world processes. Each of the proposed methods is compared with the corresponding state of the art methods, thus validating the proposed approaches.

Resumo

Sensores virtuais são modelos inferenciais que utilizam sensores disponíveis online (e.g. temperatura, pressão, vazão, etc) para prever variáveis relacionadas com a qualidade do processo, que não podem ser medidas de forma automática, ou só podem ser medidas com um custo elevado, de forma esporádica, ou com longos atrasos (e.g. análises laboratóriais). Sensores virtuais são construídos usando os dados históricos de processo, geralmente fornecidos pelo sistema de controlo de supervisão e aquisição de dados (SCADA) e pelas anotações das medições de laboratório. No desenvolvimento dos sensores virtuais, há muitas questões para lidar. As principais questões são o tratamento de dados em falta, a detecção de outliers, a seleção das variáveis de entrada, o treino do modelo, a validação, e a manutenção do sensor virtual. Esta tese centra-se em três destas questões, nomeadamente, a seleção de variáveis de entrada, o treino do modelo e a manutenção do sensor virtual. Novas metodologias são propostas em cada uma destas áreas. A selecção das variáveis de entrada é baseada na rede neuronal *multilayer perceptron* (o modelo de regressão não linear mais popular em aplicações de sensores virtuais). A segunda questão, o treino do modelo, é tratado no contexto de múltiplos modos de operação. Exemplos de multiplos modos de operação são a variação da carga diurna de uma central de produção energia, a operação verão-inverno de uma refinaria, etc. Nesta tese, para treinar um modelo no contexto dos múltiplos modos de operação, o modelo de regressão por mínimos quadrados parciais (PLS), um método muito difundido na literatura de quimiometria e um dos métodos mais utilizados na indústria, é inserido no método de mistura de especialistas (ME), derivando assim o método de mistura de modelos mínimos quadrados parciais (Mix-PLS) especialistas. O Mix-PLS permite caracterizar múltiplos modos de operação. O terceiro problema está relacionado com a manutenção do sensor virtual. Na manutenção do sensor virtual, o modelo é actualizado utilizando amostras recentes do processo. A maneira mais

comum de fazê-lo é através de aprendizagem exponencial-recursiva dos parametros do modelo. Na aprendizagem exponencial-recursiva, é utilizado um factor de esquecimento para dar exponencialmente menos pesos para as amostras mais antigas. Em muitas aplicações, valores pequenos do factor de esquecimento levam a um melhor desempenho de predição. Contudo, o factor de esquecimento está directamente relacionado com o número "efectivo" de amostras, e valores baixos do factor de esquecimento podem proporcionar os mesmos problemas enfrentados na modelação de sistemas estáticos, tais como *overffiting*, mau desempenho de predição, etc. Para resolver este problema, um novo modelo, baseado na mistura de modelos de regressão linear univariados (portanto de baixa dimensionalidade) (MULRM), é proposto, permitindo a utilização de baixos valores de factor de esquecimento. Todos os métodos propostos nesta tese são testados em conjuntos de dados obtidos de processos reais. Cada método proposto é comparado com os respectivos métodos do estado da arte, validando assim as abordagens propostas.

Symbols and Abbreviations

General Abbreviations

ACT	Accuracy-Complexity Trade-off
AIC	Akaike Information Criterion
AKL	Adaptive Kernel Learning
ANN	Artificial Neural Network
B&B	Branch and Bound
BIC	Bayesian Information Criterion
CC	Correlation Coefficient
CGPCNE	Conjugate Gradient Precondition Normal Equation
DOF	Degrees of Freedom
EFS	Evolving Fuzzy System
EM	Expectation Maximization
EN	Elastic Net
FCM	Fuzzy C-Means
FIR	Finite Impulse Response
FMGM	Finite Mixture of Gaussian Models
FS	Fuzzy Systems
GA	Genetic Algorithm
iff	if and only if
i.i.d.	Independent and Identically Distributed
ILLSA	Incremental Local Learning Soft Sensing Algorithm
IANN	Iteratively Adjusted Neural Network
IRLS	Iterative Reweighted Least Squares

KNN	k-nearest neighbors algorithm
LASSO	Least Absolute Shrinkage and Selection Operator
LM	Linear Model
LOOCV	Leave One Out Cross Validation
LS	Least Squares
ME	Mixture of Experts
MI	Mutual Information
Mix-PLS	Mixture of Partial Least Squares (PLS) Experts
MLRE	Mixture of Linear Regression Experts
ML	Maximum Likelihood
MLP	Multilayer Perceptron
MM	Mixture of Models
MSE	Mean Square Error
MULRM	Mixture Univariate Linear Regression Models
mRMR	minimum Redundancy Maximum Relevance [principle]
NFS	Neuro-Fuzzy System
NIPLS	Nonlinear Iterative Partial Least Squares
NIPALS	Nonlinear Iterative Partial Least Squares
NIR	Near Infrared
NMIFS	Normalized Mutual Information Feature Selection
NN	Neural Network
NRMSE	Normalized Root Mean Square Error
OS-ELM	Online Sequential Extreme Learning Machine
PCA	Principal Component Analysis
pdf	probability density function
PLS	Partial Least Squares
Prop.	Proposed
RF	Receptive Field
RLS	Recursive Least Squares
RPLS	Recursive Partial Least Squares

r.h.s	right hand side
RR	Ridge Regression
RSS	Residual Sum of Squares
SA	Simulated Annealing
SBS	Sequential Backward Search
SCADA	Supervisory Control And Data Acquisition
SFS	Sequential Forward Search
SFFS	Sequential Forward Float Search
SLNN	Single Layer Neural Network
SVM	Support Vector Machine
SVR	Support Vector Regression
TS	Takagi Sugeno
VIF	Variance Inflate Factor
WLL	Weighted Log-Likelihood
WLS	Weighted Least Squares
WTP	Water Treatment Plant

General Symbols

σ	Standard deviation of approximation error ξ
ω	Variance of approximation error ξ
Φ	Set of k data exemplars (samples)
Φ_{-r}	Set of k data exemplars (samples) without variable r
$\xi(i)$	ith sample of the approximation error of deterministic function
$\boldsymbol{\theta}$	Parameter vector of deterministic function $f(\cdot)$
$D(\theta)$	Number of parameters of the deterministic function $f(\cdot)$
$\mathcal{O}(\cdot)$	Complexity
D	Number of input variables
$\mathbb{E}_{\mathbf{a}}(\mathbf{b})$	Expectation of $\mathbf{b}$ with respect to distribution of $\mathbf{a}$
$f(\cdot)$	Deterministic function
$\mathbf{I}$	Identity Matrix
k	Number total of samples

$\mathcal{N}(a(i)\|\nu,\omega)$	Gaussian/Normal distribution of variable a with mean ν and variance ω
$p(\mathbf{a})$	Pdf of variable a
$p(\mathbf{a}\|\mathbf{b})$	Conditional pdf of variable a given b
$\mathbf{X}$	Input variables matrix
$\mathbf{X}_{-r}$	Input variables matrix without variable r
X	Set of input variables
X_{-r}	Set of input variables without variable r
$\mathbf{x}$	Input variables vector
$\mathbf{x}(i)$	ith sample of input variables vector
$\mathbf{x}_{-r}$	Input variables vector, without variable r
$\mathbf{x}_{-r}(i)$	ith sample of input variables vector, without variable r
$\mathcal{X}$	Space of input variables
x_j	jth input variable
$\mathbf{y}$	Output vector
$y(i)$	ith sample of output variable
y	Output variable
$\mathcal{Y}$	Space of output variables

Variable Selection Using the MLP Model

$\mathbf{\Lambda}$	Set of weight and bias parameters of MLP neural network
$\psi(\cdot)$	Activation functions of the nodes of the output layer of MLP neural network
Δw_{ij}	Gain of gradient-based delta-rule for weight w_{ij}
Δb_{ij}	Gain of gradient-based delta-rule for bias b_{ij}
η	Learning rate of backpropagation algorithm
ϵ	Difference between E^{all} and E^{rel}
$\hat{\epsilon}(i)$	Difference between the errors of models (with and without variable r) for each sample i
$\hat{\mathbf{\Delta}}(i)$	Vector representation of $\hat{\epsilon}(i)$ into the first layer of the MLP model
δ_{ij}	Adjusting factor for input weight w_{ij}
$\mathbf{b}_I$	Hidden layer biases of MLP neural network
$\mathbf{b}_{I,-r}$	Hidden layer biases of MLP neural network without the variable r

$\mathbf{b}_O$	Output bias of MLP neural network
$\mathbf{b}_{O,-r}$	Output bias of MLP neural network without the variable r
e	Number of training (epochs) iterations
$E(f(\mathbf{X};\mathbf{\Lambda}),\mathbf{y})$	Error of MLP model with parameters $\mathbf{\Lambda}$
E^{all}	Error of MLP model trained with all variables
E^{rel}	Error of MLP model trained with a subset of variables
$\mathcal{E}_{-r}$	Difference between the error of a MLP model trained with and without variable r
$\mathcal{E}^*_{-r}$	Difference between the error of a MLP model adjusted with and without variable r
$g(\cdot)$	Activation functions of the nodes of the hidden layer of MLP neural network
h	Number of hidden layer nodes of MLP neural network
L_{-r}	Set of indices of the elements in X_{-r}
l	Element of L_{-r}
$\mathcal{MLP}(\Phi)$	MLP model trained with dataset Φ
O	Output node of MLP neural network
S	Input variables ranked (ordered in a selection rank) in order of importance by the proposed variable selection algorithm
$\mathbf{W}_I$	Matrix of the input weights of MLP neural network
$\mathbf{W}_{I,-r}$	Matrix of the input weights of MLP neural network without the variable r
$\mathbf{W}^*_{I,-r}$	Matrix of input weights updated/adjusted according to the δ_{ij} values
$\mathbf{w}_O$	Vector of the output weights of MLP neural network
$\mathbf{w}_{O,-r}$	Vector of the output weights of MLP neural network without the variable r
$\hat{y}(i)$	ith sample of predicted output
$\hat{y}_{-r}(i)$	ith sample of predicted output of MLP model without variable r
$\hat{y}^*_{-r}(i)$	ith sample of predicted output of adjusted MLP model without variable r

Soft Sensors in Multiple Operating Modes with Mix-PLS

$\boldsymbol{\Omega}$	Additional pdf parameters
$\gamma_p(i)$	Responsibility of model p; probability of model p generating the data sample i
ω_p	Variance of the model of expert p
$\boldsymbol{\Gamma}_p$	Diagonal matrix of the responsibilities of expert p
$\boldsymbol{\theta}_p$	Parameters of linear model of expert p
$\boldsymbol{\Theta}$	Matrix of model weights parameters
$\boldsymbol{\theta}$	Weight parameter of linear model
$\boldsymbol{\vartheta}$	Set of parameters of the mixture of experts
$\upsilon_p\left(\mathbf{x}(i), \mathbf{V}\right)$	Output of gate function of the model p with parameters $\mathbf{V}$
$\mathbf{B}$	diagonal matrix of the regression weights b_m
$\mathbf{B}_{(\boldsymbol{\Gamma},p)}$	PLS latent matrix of the weighted output $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$
$\mathbf{B}_{(\mathbf{R},p)}$	PLS latent matrix of the gate model $\mathbf{y}_{(\mathbf{R},p)}$
$\mathbf{b}_m$	mth regression weight
$\mathrm{dof}(m, \mathbf{X}, \mathbf{y}, \mathbf{T})$	Degree of freedom of the PLS model with m latent variables, input $\mathbf{X}$, output $\mathbf{y}$ and orthogonal latent matrix $\mathbf{T}$
$\mathbf{E}$	Matrix of input data residuals
$\boldsymbol{\mathcal{E}}$	Set of parameters of model expertts
$\mathbf{F}$	Matrix of output data residuals
$f_p(\mathbf{x}(i), \boldsymbol{\theta}_p)$	Output of the linear model of the expert p with parameters $\boldsymbol{\theta}_p$
$\ln \mathcal{L}_w$	Weighted log-likelihood
$\mathcal{M}$	Set of the possible/eligible number of latent variables for PLS model
M	Number of latent variables of the PLS model
Me_p	Number of latent variables of the PLS model expert p
Mg_p	Number of latent variables of the PLS model of gate p
P	Total number of experts
$\mathbf{P}$	Loading matrix of input
$\mathbf{P}_{(\boldsymbol{\Gamma},p)}$	PLS latent matrix of the weighted input $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$
$\mathbf{P}_{(\mathbf{R},p)}$	PLS latent matrix of the gate model $\mathbf{X}_{(\mathbf{R},p)}$
$\mathbf{p}_m$	mth loading vector of input
$Q_{\mathrm{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}})$	Expectation of the complete-data (output and hidden variables) log likelihood

Q_g	Contributions of the gate for the expectation of complete-data log likelihood
Q_e	Contributions of the experts for the expectation of complete-data log likelihood
$Q_{e,p}(\cdot)$	Contribution of expert p for the expectation of complete-data log likelihood
$\mathbf{Q}$	Loading matrix of output
$\mathbf{Q}_{(\mathbf{\Gamma},p)}$	PLS loading matrix of the weighted output $\mathbf{y}_{(\mathbf{\Gamma},p)}$
$\mathbf{Q}_{(\mathbf{R},p)}$	PLS loading matrix of the gate model $\mathbf{y}_{(\mathbf{R},p)}$
$\mathbf{q}_m$	mth loading vector of output
$\mathbf{T}$	Orthogonal latent matrix of input
$\mathbf{T}_{(\mathbf{\Gamma},p)}$	PLS loading matrix of the weighted input $\mathbf{X}_{(\mathbf{\Gamma},p)}$
$\mathbf{T}_{(\mathbf{R},p)}$	PLS loading matrix of the gate model $\mathbf{X}_{(\mathbf{R},p)}$
$\mathbf{t}_m$	mth orthogonal latent vector of input
$\mathbf{U}$	Orthogonal latent matrix of output
$\mathbf{u}_m$	mth orthogonal latent vector of output
$\mathbf{V}$	Set of parameters of gates
$\mathbf{v}_p$	Parameter of gate p
$\mathbf{X}_{(\mathbf{R},p)}$	Weighted inputs matrix of gate model p with weight matrix $\mathbf{R}_p$
$\mathbf{X}_{(\mathbf{\Gamma},p)}$	Weighted inputs matrix of model expert p with weight matrix $\mathbf{\Gamma}_p$
$\mathbf{y}_{(\mathbf{\Gamma},p)}$	Weighted output matrix of model expert p with weight matrix $\mathbf{\Gamma}_p$
$\mathbf{Z}$	Set of hidden variables
$\mathbf{z}(i)$	Vector of hidden variables for a sample i
$z_p(i)$	Value of hidden variable for expert p and a sample i
$\mathbf{z}_{(\mathbf{R},p)}$	Weighted output matrix of gate model p with weight matrix $\mathbf{R}_p$

Mixture of Linear Regression Models for Adaptive Soft Sensors

$\boldsymbol{\theta}$	Weight parameter of linear model
$\boldsymbol{\theta}_p$	Weight parameters of univariate linear regression model of variable x_p
$\mathbf{\Theta}$	Set of all weight parameters
$\boldsymbol{\vartheta}$	Set of parameters of the mixture of models
$\mathbf{\Upsilon}$	Set of mixing coefficient parameters

υ_p	Mixing coefficient of model p
Φ''_m	Complete historical data set with m samples
Φ_k	Historical data set with k samples
$\boldsymbol{\Omega}$	Additional pdf parameters
λ	Forgetting factor
$\phi_j(x_p)$	Nonlinear transformation of variable x_p
$\boldsymbol{\phi}(\mathbf{x})$	Vector of known functions
$\gamma_p(i)$	Responsibility of model p; probability of model p generating the data sample i
$\boldsymbol{\Gamma}_p$	Diagonal matrix of responsibilities of model p
$\mathbf{A}$	Design matrix
$\mathbf{A}_p$	Design matrix of univariate model p
d	Effective number of observations/samples
$E_{\mathbf{W}}(\boldsymbol{\theta})$	Weighted squared error
$\boldsymbol{\mathcal{E}}$	Set of model parameters in mixture of models
$f(\mathbf{x}, \boldsymbol{\theta})$	Output of linear model with parameters $\boldsymbol{\theta}$
$f_p(x_p, \boldsymbol{\theta}_p)$	Univariate linear regression model of variable x_p and parameter $\boldsymbol{\theta}_p$
$\mathcal{L}$	Expected squared loss
$\mathbf{P}$	Inverse of covariance matrix
$\mathbf{P}_p$	Inverse of covariance of MULRM model p
$\mathcal{M}_k$	Model learned using all, or only a selection of samples from Φ_k
NaN	Unknown value
$Q_{\text{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$	Expectation of complete data log-likelihood of mixture of models
Q_{MULRM}	Expectation of complete data log-likelihood of MULRM
$\mathbf{W}$	Diagonal matrix of weights
$\mathbf{W}_p$	Diagonal matrix of weights of univariate model p
$\mathbf{Z}$	Set of hidden variables
$\mathbf{z}$	Vector of hidden variables

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Contents

Industrial processes are well equipped with a variety of sensors, such as temperature, flow rate and pressure sensors, designed for online supervision, monitoring and control, and to maintain consistent product quality. Some variables, which may be quality variables for example, cannot be automatically measured online, due to the lack of sensors, or due to the high cost of the sensor, thus leading to the lack of enough information about the system state in real-time. Usually, laboratory tests of product samples are conducted to measure off-line the product quality on a specified interval base. In order to measure the quality variables in real-time, one can use computational intelligence methodologies to build intelligent/computational sensors to infer the value or the quality target variables from other on-line measured process variables. The basis for building such intelligent sensors is that the values of target variables, or the product quality, have a functional relationship with other process variables that can be measured on-line. Such kind of intelligent sensors are usually referred as soft sensors. They are important tools for many industrial processes, such as pulp and paper mills, wastewater treatment systems, cement kilns, refineries, and polymerization processes, just to give a few examples. In general terms,

soft sensors can be defined as inferential models that use online available sensor measurements (easy to measure variables) for on-line estimation of quality variables (hard to measure variable) which cannot be automatically measured at all, or can only be measured at high cost, sporadically, or with high delays (e.g. laboratory analysis). Their development encompasses the same design cycle of many pattern recognition/identification systems: data acquisition and filtering, input variable selection, model choice, model training, model validation, and model maintenance.

1.1 Objectives and Approach

The objective of this thesis is to propose new methodologies for soft sensors development. As mentioned above, soft sensor development encompasses many topics (e.g. data acquisition and filtering, input variable selection, model choice and training, model validation and model maintenance). Each of these topics may involve many techniques and theories, so it is quite impossible to cover all of these aspects during a PhD program. Then, the issues addressed during the development of this thesis were focused on the topics of selection of input variables, model choice and model training, and model maintenance.

Soft sensors applications usually have to deal with a large number of input variables during model design, and the number of input variables increases if there is the necessity to select the dynamics of each variable (i.e. involving the selection of not only the variables but also their time-lags) [Souza *et al.*, 2010b]. For linear models there are many established methods to select the most important inputs (e.g. correlation coefficient, residual regression, etc). However, for the cases where there is a non-linear relationship between inputs and output, the selection of input variables becomes a challenge. The most common way to select the input variables is to choose the ones that give the most accurate soft sensor model. This can be achieved by training the model with all inputs and then sequentially and iteratively removing the less relevant/important input variables. The most accurate form of accessing the relevance of an input variable is by training the model with and without such input. If the model performance increases in the absence of the input, it can be considered irrelevant and it can be discarded from the model. However, despite of the good results of such approach, its implementation is very time demanding, since it is necessary to retrain the model every time that an input is going to be evaluated,

preventing the implementation of such method when there is a large number of input variables. The method proposed in this thesis is a fast implementation of such input selection strategy (sequential removal of irrelevant inputs based on the performance of the prediction model) using the multilayer perceptron (MLP) neural network as the soft sensor model (MLPs are one of the most popular regression models in soft sensor development).

The second topic investigated in this thesis regards to the model training. The usual approach for development of soft sensors in industry is learning a single model based on all the training samples. However, such approach does not take into consideration some intrinsic characteristics of industrial processes, such as multiple operating modes. Multiple operating modes may result from a variety of causes, such as external disturbances, as for example the change in feedstock or product grade, or even changes such as the diurnal load variation of a power plant or the summer/winter operation of a refinery. The modeling in such kinds of scenarios is very challenging, because of the incomplete knowledge about the operating points and when they happen. An appropriate approach should detect the operating modes only based on the available data. In this thesis the mixture of experts (ME) framework is employed to deal with such problem. The partial least squares (PLS) model, the most popular approach in soft sensor modeling, is integrated into the ME framework, thus deriving the Mix-PLS, a novel algorithm for learning of prediction models in scenarios with multiple operating modes.

The third problem studied in this thesis is related to soft sensor maintenance. Industrial processes are rather dynamic environments and it is very difficult for a soft sensor constructed using limited information from historical data to react to changes in the operating environment. The maintenance capability in soft sensors applications corresponds to the problem of maintaining a good soft sensor response even in the presence of process variations, or some data change. A common strategy is to update the soft sensor model periodically using the incoming samples, and using an exponentially recursive learning of parameters to forget old samples and to adapt to the current trend of the process. In exponentially recursive learning, a forgetting factor is used to give exponentially less weight to older error samples. In many applications, small values of the forgetting factor can lead to better predictive performance. However, the forgetting factor is directly related with "effective" number of samples, and small values of forgetting factor can result in problems similar

to the ones faced when modeling static systems, such as overfitting, poor prediction performance, etc. To solve this problem, a new model, based on the mixture of univariate (thus low dimensional) linear regression models (MULRM) is proposed, allowing the use of small values of forgetting factor.

All the proposed methods were evaluated in real soft sensors data sets, coming from real-world processes. The processes where the methodologies were applied ranged from polymerization, water treatment plant, and distillation column. Each of the proposed methods was compared with the state of art methods in its respective area.

1.2 Key Contributions

The main contributions of this thesis encompass three distinct parts of soft sensors development: selection of input variables, model training, and soft sensor maintenance. The contributions are at a methodological level and they have been compared with the current state of art in their respective topics. The methodologies proposed were published in ISI international journals, as well in international conferences proceedings.

The specific contributions of this thesis are given as follows:

1. To deal with the problem of variable selection, this thesis proposes a new method of variable selection for prediction in regression scenarios using MLP models [Souza *et al.*, 2013]. The approach proposed in [Souza *et al.*, 2013] enables the selection of variables in non-linear scenarios and is based on a MLP model, with the compromise of being quite fast while providing good results of prediction. Additionally, in one of the case studies where the proposed variable selection algorithm was evaluated, the selected variables (only five variables, or six variable-delay pairs, despite of the eleven available variables) have shown to correspond to a meaningful physical interpretation, while attaining good performance in the prediction of the target variable of interest. The proposed method was also compared with four state-of-the-art methods, having the advantage of selecting an equal or a smallest number of variables when compared to the other four methods, while attaining good performance in the prediction of the target variable;

2. In this thesis, the learning of models in multiple operating modes was investigated. The motivation for the use of specific models in such scenarios is given and a new learning method for PLS regression models in the context of multiple operating modes is presented. The learning of models in the context of multiple operating modes was done as follows. The partial least squares regression (PLS), a well know method in the chemometrics literature and one of the mostly used methods for soft sensors in industry, is integrated into the mixture of experts framework, thus deriving the new mixture of partial least squares (Mix-PLS) regression, a nonlinear regression model which deals efficiently with multiple operating modes [Souza and Araújo, 2014b]. In the experimental part, the proposed method was evaluated in two real-world soft sensor problems, where multiple operating modes are present. The results suggests that the Mix-PLS algorithm, can attain better results when compared to state-of-the-art linear and non-linear regression models, such as PLS, MLP and SVR models, in such scenarios. The proposed method was also evaluated in a problem where multiple operating modes are not present, and where Mix-PLS has also shown to have superior performance.

3. Adaptive models is a recent subject of study in the soft sensors context. The use of such approach in industry is of great importance as an alternative for the offline-designed static soft sensors and their drawback of performance deterioration when the time passes. Then, a new method for learning in time-varying scenarios is proposed using an online mixture of univariate linear regression models [Souza and Araújo, 2014a]. The proposed method was evaluated in two real-world soft sensor data sets with time-varying characteristics, and it has been compared with four state-of-the-art regression methods for time-varying scenarios. The proposed method has shown to perform better in almost all experiments.

The list of contributed publications is given as follows:

International journals:

1. [Souza *et al.*, 2013]. Francisco Souza, Rui Araújo, Tiago Matias, Jérôme Mendes. A multilayer-perceptron based method for variable selection in soft

sensor design. *Journal of Process Control*, Vol. 23, pp. 1371-1378, November, 2013.

2. [Souza and Araújo, 2014b]. Francisco Souza, Rui Araújo. Mixture of partial least squares experts and application in prediction settings with multiple operating modes, *Chemometrics and Intelligent Laboratory Systems*, Vol. 130, pp. 192-202, January, 2014.

3. [Souza and Araújo, 2014a]. Francisco Souza, Rui Araújo. Mixture of univariate linear regression models for adaptive soft sensors, *IEEE Transactions on Industrial Informatics*, (In Press).

Conference proceedings:

1. [Souza and Araújo, 2012]. Francisco Souza, Rui Araújo. An online variable selection method using recursive least squares. In: 2012 17th IEEE *Conference on Emerging Technologies and & Factory Automation* (ETFA 2012), Krakow, Poland.

2. [Souza *et al.*, 2011]. Francisco Souza, Tiago Matias, Rui Araújo. Co-evolutionary genetic multilayer perceptron for feature selection and model design. In: 2011 16th IEEE *Conference on Emerging Technologies & Factory Automation* (ETFA 2011), Toulouse, France.

3. [Souza and Araújo, 2011]. Francisco Souza, Rui Araújo. Variable and time-lag selection using empirical data. In: 2011 16th IEEE *Conference on Emerging Technologies & Factory Automation* (ETFA 2011), Toulouse, France.

4. [Souza *et al.*, 2010a]. Francisco Souza, Rui Araújo. Variable selection based on mutual information for soft sensors applications. In: 9th *Portuguese Conference on Automatic Control*, Controlo 2010, Coimbra.

5. [Souza *et al.*, 2010b]. Francisco Souza, Pedro Santos, Rui Araújo. Variable and delay selection using neural networks and mutual information for data-driven soft sensors. In: 2010 15th IEEE *Conference on Emerging Technologies & Factory Automation* (ETFA 2010), Bilbao, Spain.

The main content of this thesis is derived from the works described above and published in international journals [Souza *et al.*, 2013; Souza and Araújo, 2014b,a]. However, during the PhD program, several works have been published in international conferences. These works are used as auxiliary content during the development of this thesis, by serving mainly as reference works. All of them are methodologies devoted to the variable selection step. In [Souza *et al.*, 2010b] a method for variables and delay selection was presented. In [Souza *et al.*, 2010b] the delays of each variable were selected using the mutual information (MI) criterion and then the variables were selected based on the MLP model. In [Souza *et al.*, 2010a; Souza and Araújo, 2011] a method for variable selection based on MI was presented. In [Souza *et al.*, 2011] an evolutionary framework was developed to derive the best MLP model and its respective input variables. In [Souza and Araújo, 2012] a simple approach was proposed to select the variables in adaptive scenarios.

During the PhD program, the interaction with other researchers and PhD students of the group at the "Institute of Systems and Robotics - University of Coimbra" (ISR-UC), allowed the collaboration in the development of the following other new works that were published on ISI journals [Matias *et al.*, 2014], [Mendes *et al.*, 2013], [Mendes *et al.*, 2012b], and at international conferences [Matias *et al.*, 2013], [Mendes *et al.*, 2012a], [Soares *et al.*, 2011], [Mendes *et al.*, 2010], but are outside the scope of this PhD work and thesis.

1.3 Thesis Outline

This thesis is organized as follows. Chapter 2 discusses the literature review on the soft sensor design. It includes a literature review on data collection and filtering, selection of input variables, model choice and training, model validation, and model maintenance.

In Chapter 3, a method for variable selection based on an MLP model is presented. It is based on the work [Souza *et al.*, 2013]. The motivation and derivation of the method are given. The chapter also includes experimental results on real-world soft sensors problems, in artificial and benchmark data sets. A comparison with other state of art methods is performed. A discussion on the results of all methods is presented.

Chapter 4 presents the derivation of the Mix-PLS, a method suitable for learning

in settings with multiple operating modes. It is based on the work [Souza and Araújo, 2014b]. In this chapter, the motivation and derivation of Mix-PLS is presented. A review on ME and the PLS algorithms is also introduced. The chapter also presents experimental results on artificial and real-world soft sensors problems, the last ones having multiple operating modes conditions. A comparison with other state of art methods is performed. A discussion on the results of all methods is presented.

Chapter 5 presents the mixture of univariate linear regression models. It is based on the work [Souza *et al.*, 2013]. In this chapter, the problem of online learning in regression scenarios, as well the most commonly used approaches in such context are presented. Then, the motivation and derivation of MULRM are presented. A review on the mixture of models (MM) approach is also introduced in the chapter. The chapter also presents experimental results on real soft sensors problems. A comparison with other state of art methods of online learning is performed. A discussion on the results of all methods is presented.

Chapter 6 gives a general conclusion on the content presented in this thesis and also gives a guide for future works.

Chapter 2

State of the Art

Contents

A soft sensor is a regression model which uses easy-to-measure variables to predict a hard-to-measure variable. It is subject of research in many areas. Originally, soft sensors were studied as part of chemometrics, which stands for statistical methods for extracting information from data sets that often consist of many measured variables [Wold, 1995]. According to Wold [1995]: "Chemometrics, is heavily dependent on the use of different kinds of mathematical models (high information models, ad hoc models, and analogy models). This task demands knowledge of statistics, numerical analysis, operation analysis, etc., and in all, applied mathematics.", i.e. chemometrics is not an isolated/sole research area. From the chemometrics literature it is possible to see the use of different approaches including machine learning and pattern recognition [Bishop, 2006], artificial intelligence [Haykin, 1999], system identification [Ljung, 1999], and statistical learning theory [Hastie *et al.*, 2001]. Despite the fact that the objectives and emphasis on all these areas are different, they are intrinsically connected by the necessity to learn models from data. This point of view is further justified in the work done in [Ljung, 2010], where the author revises the problem of system identification.

Then, the state of the art discussed here will not be limited the chemometrics literature, it will also discuss the main and recent contributions from the other areas.

Soft sensor development encompasses the same design cycle of classical regression systems [Duda *et al.*, 2000; Ljung, 1999]. However, it has its own peculiarities. Soft sensor development has the following main steps [Fortuna *et al.*, 2006; Kadlec *et al.*, 2009]: (I) data collection and filtering, (II) selection of input variables, (III) model choice and training, (IV) model validation, and (V) model maintenance. In the first stage the data is collected, and the goals of this stage include the handling of missing data and outliers. The goals of the second stage are the selection of most relevant inputs, and possibly also the respective time lags. The model choice and training requires the correct selection and learning of the model. The model validation step is necessary to judge if the learned model reproduces the target variables within acceptable quality or performance levels. The last step is soft sensor maintenance, where the goal is to maintain a good soft sensor response under the presence of process variations or some data change. In this chapter the literature review on each of these topics will be performed, reviewing the most important works in these areas.

2.1 Data Collection and Pre-Processing

Industries are usually required to store their data from the processes. This is the basis for the subsequent use of such data for system optimization, or other related data driven methods. Unfortunately, data collection in real industrial applications comes with well know problems to deal with, such as problems with sampling time, missing data, outliers, working conditions, accuracy, and so on.

2.1.1 Sampling Time

In industrial systems some variables are acquired at different time rates. This is most evident when analyzing the sample rates of easy-to-measure and hard-to-measure variables. In the majority of problems the acquisition frequency of easy-to-measure variables is much higher than the acquisition frequency of hard-to-measure variables. In such cases there is the necessity to synchronize the variables. This problem is usually refereed in literature as multirate character, or multiple-rate phenomenon [Wu and Luo, 2010]. In practice the following two approaches are most commonly adopted:

1. Down-sample of the easy-to-measure data samples, in accordance with the slow sampling rate of the hard-to-measure variables, by excluding the samples of the easy-to-measure variables that do not have a corresponding hard-to-measure (target) value [Kadlec and Gabrys, 2011; Lu *et al.*, 2004];

2. Instead of excluding the samples that do not have the respective target, a finite impulse response (FIR) model is estimated and applied on the samples in order to estimate the hard-to-measure, low sampling rate, variables. The big concern in this approach is the selection of weighting values and length of the FIR filter, in [Wu and Luo, 2010] a heuristic approach was adopted, while in [Xie *et al.*, 2013] an approach based on the expectation maximization (EM) was proposed.

Although down-sampling by excluding is straightforward to implement in practice, it has a critical drawback of information loss and may lead to inaccurate models, mainly if the hard-to-measure variable is sampled scarcely and/or with uncertain delays [Xie *et al.*, 2013]. A better approach is to model the data by using the FIR

filter. However, the weights and length of the FIR filter should be designed or estimated carefully.

2.1.2 Missing Data

It is quite common to have observations with missing values for one or more variables. The problem of missing data occurs when no value is stored for a variable in an observation. There are two common approaches to deal with missing data. The first one is the removal of samples containing missing data, an approach also known as listwise deletion. The second approach is to fill-in the missing values using some imputing method. The first approach can be used if the number of missing values is small, but otherwise it should be avoided [Hastie *et al.*, 2001]. In the second case, the simplest strategy is to impute the missing value with a mean or median of non missing values for that variable. Another approach is the hot-deck imputation, where a missing value is imputed from a randomly selected value of the input for similar target values [Andridge and Little, 2010]. These methods of mean/median imputation, and hot-deck imputation, are usually referred as multiple imputation.

Two other methods which are often employed for handling missing data are the maximum likelihood (ML) method and the EM method. The ML method models the missing variable/s based on the available data. Essentially, the ML assumes some model for the data distribution of the missing variable, and then the parameters of the model are estimated using ML. In [DeSarbo *et al.*, 1986] the authors assumed linear relationships, while in [Jerez *et al.*, 2010] several nonlinear models were used to model the relationship among the non-missing variables and the variable with missing values. In both cases, the authors reported significant improvement when compared to multiple imputation methods (hot-deck, and mean/median imputations). The EM approach to handle missing data is reported in [Enders, 2001], it works similarly to the ML procedure, although it is an iterative procedure. First it estimates the missing data using the observed data and the first estimates of the model parameters. In the second step, the estimated missing data are used together with observed data to estimate the parameters. This iterative process repeats until there are no significant changes in parameters estimates. In [Richman *et al.*, 2009] it is made an extensive review on methods for missing data imputation.

2.1.3 Outliers

Outliers are observation values that deviate significantly from the typical, meaningful range of values. Observations take inconsistent values when compared to the majority of recorded data, and this can greatly affect the performance of the soft sensor design [Kadlec *et al.*, 2009]. Outliers can be caused, for example, by sensor malfunction, communication errors, or sensor degradation. To alleviate the effects of outliers it is necessary first to detect them, and then to treat them. However, when applying outlier detection methods, usually the results have to be validated manually by the model developer and/or process expert. The goal of the manual inspection is to detect any possible outlier maskings (i.e. false negative detections - not detected outliers) and outlier swamping (i.e. false positive detections - correct values labeled as outliers).

Typical outlier detection methods are based on statistical techniques. The most simple approach is the 3σ-rule [Pearson, 2002], which is based on an univariate distribution of variables. The 3σ-rule works as follows: assuming that a variable is drawn from a Gaussian distribution with mean μ and standard deviation σ, the samples of that variable which are outside the bounds $[\mu - 3\sigma, \mu + 3\sigma]$ are considered outliers. A robust version of 3σ-rule is the Hampfel identifier [Davies and Gather, 1993], which considers the absolute mean and absolute mean deviation. The Hampfel identifier is suitable in the cases where the data is severely affected by outliers, and it has shown to be practically effective in real applications [Liu *et al.*, 2004; Pearson, 2002]. The above approaches are considered as univariate outlier detection methods, since they are applied on each variable separately. However, in many cases outliers cannot be detected by considering the variables individually. Then, multivariate techniques should be adopted. Outlier detection based on multivariate techniques takes into consideration the interaction among variables, and it can deliver most accurate results, as demonstrated by [Fortuna *et al.*, 2006; Bella *et al.*, 2007]. It often works by using distance measures to indicate those samples which are far from the center of data distribution. A common distance measure adopted is the Mahalanobis distance, where the samples considered outliers are the ones with a large value of Mahalanobis distance [Ben-Gal, 2005]. Other multivariate approach commonly used in the soft sensors context is based on data projection/dimensionality reduction techniques, such as principal component analysis (PCA) or partial least squares

(PLS), together with the Jolliffe parameters [Jolliffe, 2002; Warne *et al.*, 2004]. It works by decomposing the original data using PCA or PLS, and then using the decomposed data to compute the Jolliffe parameters [Jolliffe, 2002]. The Jolliffe parameters help to identify the samples that do not conform with the correlation structure of data and the ones that inflate the data variance. In [Warne *et al.*, 2004; Fortuna *et al.*, 2006] outlier detection based on PCA, PLS, and Jolliffe parameters was studied and has been shown to be a powerful alternative for outlier detection in soft sensors applications.

In [Penny and Jolliffe, 2001] several outlier detection methods were compared (six in total), and the authors concluded that the efficacy of the proposed methods depends strongly on the problem domain. In particular, the efficacy depends on whether the data is multivariate normal, on the dimension of data set, on the type of outliers, and on the amount of outliers in the data set. The authors recommend a battery of multivariate outlier detection tests to detect outliers. In the soft sensor context, [Bella *et al.*, 2007] compared several outlier detection methods in the modeling of a sulfur recovery unit. The use of outlier detection improved considerably the soft sensor accuracy in the case-study, and PCA-based outlier detection achieved the best results.

The book of [Fortuna *et al.*, 2009] provides several discussions regarding pre-processing techniques and their application in the soft sensor context. Real-world examples as well comparison of techniques are also presented. In [Kadlec *et al.*, 2009; Pani and Mohanta, 2011] general overviews on pre-processing techniques are also presented.

2.2 Variable Selection

In soft sensor applications there is frequently a large amount of candidates for input variables coming from the supervision structure of the process. The number of candidates can range to thousands [Souza and Araújo, 2011]. The use of black-box models already suggests that the soft sensor designer has few knowledge about the system to be modeled, and consequently about the variables which affect the target variable. However, this not true in all the cases, since in most of soft sensors applications the selection of a set of most relevant variables is made by system experts. Nonetheless, for physically large and highly integrated processes, enumeration and

selection of candidate variables based on process insight may not be feasible [Warne *et al.*, 2004]. Moreover, most of the works in the literature indicate that frequently only few variables are necessary to compose the soft sensor model. A reduced number of variables has several advantages, such as the reduction of model development time, possibility of aggregation of the information about the physical interpretation of the process, or the improvement of the model performance. Moreover, a reduction of the number of variables implies a lower number of required real sensors, decreasing costs, and increasing or enabling feasibility of applications.

The following are possible approaches concerning variable selection that may be adopted during soft sensor design [Nelles, 2001]:

Use of all inputs: This approach leads to extremely high dimensional approximation problems. The problems associated with learning of a model with many input variables suffer from large computational demand, large probability of occurring overfitting, and poor performance of the regression model. Overfitting means that the model is very accurate on training data, but it has poor accuracy on previously unseen test data. A large number of input variables and a limited number of samples causes a curse of dimensionality phenomena [Bellman, 1961], which refers to some, normally problematic, phenomenon that occurs in high-dimensional spaces but does not occur in low-dimensional spaces. In the case of a variable selection setting, one curse of dimensionality problem that occurs is that the number of samples required to represent an input space increases exponentially with the number of variables. Another problem that occurs is the increase of computational costs in algorithms dealing with high-dimensional spaces. Variable selection is one way to prevent overfitting, increase the model performance, and also to avoid the curse of dimensionality phenomena;

Unsupervised variable selection: The typical approach for unsupervised variable selection is based on principal component analysis (PCA) [Jolliffe, 2002]. It works by projecting the input space into a latent space, where the first latent variable (also called principal component) has the largest possible variance (i.e. it accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e. uncorrelated with) the preceding

components. Then, few components obtained by PCA are used to learn the model. The selection of the number of latent variables is crucial to attain satisfactory results. In a recent paper [Eshghi, 2014] discusses the ways to select the number of components to retain in a PCA. Applications of PCA as a basis for unsupervising variable selection are vast in soft sensors literature [Choi and Park, 2001; Zamprogna *et al.*, 2005; Lin *et al.*, 2007];

Supervised variable selection: In this approach the selection of input variables is directly guided by the goal of attaining the highest possible model accuracy; the relation between the model accuracy and a subset of inputs can be accessed independently or dependently of the model. Any procedure for input variable selection must be based on two main components [Bishop, 1995b]. First, a criterion to measure the quality of a subset must be defined, to judge whether one subset is better than another (this is usually refereed as cost/fitness function). Second, a search procedure must be defined to search through candidate subsets of variables. The selection criteria can be classified into three different classes: filter methods, wrapper methods, and embedded methods [Kohavi and John, 1997; Guyon, 2003]. Filter methods use statistical measures (e.g. correlation coefficient (CC), mutual information (MI)) to quantify the quality of a subset, and are independent of the model used. On the other hand, wrapper criteria use the performance of the model as the criterion, using for example the mean square error (MSE), the Akaike information criterion (AIC), or the Cp statistics (all these methods will be later explained in Section 2.3). In the third class, the embedded methods use a specific caracteristic about the model itself or the process of model learning to define the criterion (e.g. pruning methods, regularization). For all the three classes of methods, to achieve the optimal solution, the search procedure can consist of an exhaustive search of all possible subsets of variables. However, exhaustive search is highly computationally/time expensive, even for a moderate number of input variables. Then, in practical applications, simplified search methods such as sequential search, or stochastic search are usually employed in order to limit the computational complexity of the search procedure. Appendix B gives an overview on search procedures.

2.2.1 Filter Variable Selection

The use CC is the most popular method employed for input variable selection in soft sensors. In such CC variable selection method, the linear strength between each input and the target is computed using the Pearson correlation coefficient, and the variables are ranked according to their strength [Fortuna *et al.*, 2006; Delgado *et al.*, 2009; Gonzaga *et al.*, 2009]. For nonlinear regression settings, the Pearson correlation is usually replaced by the univariate mutual information (MI) [Cover and Thomas, 1991], and similarly to CC-based methods the variables are ranked according with their importance (see ranking search in Appendix B). The variable ranking algorithms based on the correlation coefficient and/or univariate MI can be used as the principal selection mechanism or as an auxiliary selection mechanism [Guyon, 2003]. As a principal selection mechanism, the selected inputs are used in the learning of the regression model. As an auxiliary mechanism, the variable ranking is used as a kind of screening step, removing only irrelevant variables, and then the remaining variables are passed to another variable selection algorithm to finally select the variables.

The multivariate MI approach for variable selection is a extension of the univariate MI approach, and it measures the dependency of a set of input variables on the target. In [Frénay *et al.*, 2013] it was demonstrated that the multivariate MI is an adequate criterion for variable selection in regression settings. However, the estimation of multidimensional probability density functions (pdfs) in the multivariate MI approach is not an easy task: sparcity of data, and high computational demand are some problems associated with this task.

In soft sensors/regression applications, the nonparametric k-nearest neighbors algorithm (KNN) [Kraskov *et al.*, 2004] and the histogram based estimators are the most commonly employed methods for pdf estimation in the multivariate MI approach [Beirlant *et al.*, 1997; Walters-Williams and Li, 2009]. The KNN approach tends to be used because of the good results reported in the literature [Rossi *et al.*, 2006; François *et al.*, 2007], and the histogram method is used because of its easy implementation and good results when working with a small number of variables [Ludwig *et al.*, 2009].

However, when dealing with a large number of input variables, the use of multivariate MI as a quality criterion for evaluating subsets of variables is not adequate.

The problems associated with pdf estimation are highly aggravated with the increase in problem dimensionality. In [Battiti, 1994], instead of estimating the multivariate MI, the authors approximate it by using the univariate MI. In the work of [Peng *et al.*, 2005], inspired in the work of [Battiti, 1994], the authors developed an algorithm called as the "minimum redundancy maximum relevance" (mRMR) principle for variable selection based on univariate MI. It is a well accepted method for variable selection (with more than 1890 citations since 2005). Furthermore, in [Balagani and Phoha, 2010] it was demonstrated that the algorithms of [Battiti, 1994; Peng *et al.*, 2005] are equivalent to maximization of the multivariate MI between inputs and the target. Another variant of [Battiti, 1994; Peng *et al.*, 2005] was proposed in [Estévez *et al.*, 2009] and is called normalized mutual information feature selection (NMIFS). The NMIFS criterion changes the form of how the mRMR criterion is defined, to reduce its bias and improve the quality of the selection of variables.

Several applications of MI in soft sensors and related areas have been developed. In [Ludwig *et al.*, 2009] a combination of genetic algorithms (GAs) and the mRMR principle was used to select the dynamics (i.e. time lags) of input variables of a MLP model. In [Souza *et al.*, 2010b; Souza and Araújo, 2011], the discrete mutual information was used to select the variables and corresponding time-lags in different soft sensors and regression problems. In [Souza and Araújo, 2011], it has been demonstrated that the KNN estimator of multivariate MI, together with the sequential forward search (SFS) procedure (see Appendix B), has a superior performance when compared with the CC variable selection method in two soft sensors problems. In [Souza and Araújo, 2011], the selected variables were employed in a support vector regression (SVR) model to predict the targets. In [Xing and Hu, 2009; Grbić *et al.*, 2013], the KNN estimator of multivariate MI, together with the SFS procedure was successfully employed as a variable selection tool in several real-world case-studies, and the model utilized was the MLP model. Another recent filter method for input variable selection was based on the nearest correlation spectral clustering [Fujiwara *et al.*, 2012]. The PLS model was learned with the selected inputs and then used for estimating the ethane concentration in an ethylene fractionator.

2.2.2 Wrapper Variable Selection

Another approach for selecting input variables is by assessing the performance of the learning model (wrapper approach). Usually this approach achieves more accurate prediction results when compared with filter methods, because it takes into account the approximation model. However, in the wrapper approach it is necessary to learn a regression model every time a subset of variables is going to be evaluated, which is therefore computationally expensive. Applications of wrapper methods in soft sensors/regression applications are given below.

In [Chu *et al.*, 2004], to overcome the problem associated with a limited number of samples and a large number of inputs, a bootstrapping resampling on data was applied. Then, a sequential forward float search (SFFS) (an improved version of SFS; see sequential search in Appendix B for an explanation on the SFFS procedure) together with a linear model (LM) with its parameters estimated by the least squares (LS) estimator, was used to select the relevant variables. The error of the LS model was used as the cost function. The selected variables were used in a PLS method to predict the vinyl chloride in a polimerization process. The reason for the use of LS instead of PLS, in selecting the variables, lies in the fact that LM has low computational cost when compared to PLS model.

A genetic algorithm (GA) (see stochastic search in Appendix B) together with the PLS model was applied in [Wang *et al.*, 2010] to select the input variables. Another method based on GA and PLS to select the variables and the dynamics of the system (i.e. the time lags) was proposed in [Kaneko and Funatsu, 2012]. In both these two works, the error of the PLS model was used as cost function.

In [Chatterjee and Bhattacherjee, 2011] a vision-based model was developed for the prediction of ore quality at the mine level. Due to the large number of available variables, a GA combined with a MLP network was applied to select the most relevant variables. The MLP error was used as the cost function.

To select the variables and the dynamics of the system, a SVR model together with a variant of GA encoding [Arakawa *et al.*, 2011] was used in [Kaneko and Funatsu, 2013]. The SVR error was used as the cost function. In [Liu *et al.*, 2010] the variables and the parameters of a SVR model were determined using a hybrid genetic simulated annealing search. To select the models with a complexity as small as possible, the fitness function was based on the AIC.

In [Macias-Hernandez *et al.*, 2007] the input variables were selected based on their individual prediction performance, based on the error of a Takagi Sugeno (TS)-fuzzy model. The authors compared selection performed by the expert with the automatic selection of the inputs, and it was concluded that both approaches are competitive, but in the presented case of study, better results were achieved with the automatic method.

In [Romero and Sopena, 2008a] variable selection based on MLP model and sequential backward search (SBS) (see sequential search in Appendix B) was studied. Discussion about the stopping criterion, accuracy, and computational time was performed. The authors concluded that the MLP together with SBS provides good results, but the main problem regarding this approach is its demanding computational time.

2.2.3 Embedded Variable Selection

Embedded algorithms form a class of variable selection algorithms where the selection of variables is embedded within the model or the model learning. They share similar characteristics with the wrapper algorithms, so it may be difficult or confusing to distinguish between embedded and wrapper approaches in some cases [May *et al.*, 2011]. However, the main difference between them is that an embedded method which is based on a specific model cannot be used/employed in combination/integration with another model.

Regularization methods are a class of embedded variable selection approaches. Such methods work by adding a penalty to the error function. This penalization shrinks the freedom of the model parameters during learning. For linear models they are used as an alternative to the LS solution, and in cases of poorly conditioned or ill-conditioned problems. From the statistical theory, the most well know regularization methods are the least absolute shrinkage and selection operator (LASSO) [Hastie *et al.*, 2001], ridge regression (RR) [Hoerl and Kennard, 1970], and elastic net (EN) [Zou and Hastie, 2005]. Another regularization method, widely employed in the chemometrics theory, is the PLS. In [Frank and Friedman, 1993] the authors give the statistical point of view on the PLS, and concluded that PLS plays a role similar to the RR.

The regularization approach can also be expanded to application in neural net-

works (NN), by adding a penalty function in the error function. A penalization method which penalizes both useless input variables and hidden nodes was proposed by [Similä and Tikka, 2009]. It was shown that the method outperforms the traditional regularization methods for weight decay penalization [Bishop, 1995b] and input decay [Chapados and Bengio, 2001].

In predictions settings based on NN models, variable selection can be based on sensitivity analysis approaches, also referred as pruning methods [Gevrey *et al.*, 2003; Yeh and Cheng, 2010]. In sensitivity analysis, the importance of an input is measured by computing the variation of the output when the input is perturbed. Usually, all inputs are used to train the network, and then irrelevant inputs are removed sequentially if they are considered irrelevant from the sensitivity metric point of view. After the removal of irrelevant variables, the model is retrained and the sensitivity analysis can be performed again. This procedure continues until the results get satisfactory. This is the same procedure as the SBS search (see Appendix B). Garson [1991] proposed a metric of importance based on the weights of the NN input layer. Several other proposed methods evaluate the relevance of a certain variable by computing the partial derivatives of the output with respect to that variable [Dimopoulos *et al.*, 1995, 1999]. In [Lemaire and Féraud, 2006] the importance is measured by varying the values of one variable while keeping all the others untouched, and the input variable whose changes mostly affect the output is the one that has the most relative influence. In [Castellano and Fanelli, 2000] a NN is trained with all variables, and then useless variables are sequentially removed according to an exclusion criterion based on the sensitivity metric proposed in [Garson, 1991]. However, in contrast with [Garson, 1991], when a variable is removed the existing NN model is adjusted with a lower computational cost when compared to performing again a complete retraining of the network.

Embedded methods for proposed for support vector machine (SVM) models are in their majority targeted for classification tasks, but some methods can be easily extended from classification to regression [Yang and Ong, 2011]. Despite their applicability, their use on soft sensor applications has not been tested yet, but they are worth to be mentioned here. Input selection based on SVM models proceeds in the same way as in MLP input selection based on sensitivity analysis, i.e. the selection process is usually performed as follows: train a SVM with all variables, select and remove the least relevant variables according to the sensitivity metric, re-train the

SVM model and proceed in the same manner until satisfactory results are obtained. In [Guyon *et al.*, 2002] the input weights of the SVM model were used as the sensitivity metric. The approach was applied in a cancer classification problem where the number of inputs is larger than 7000 and only few samples were available. A different approach to define the sensitivity metric was adopted by [Rakotomamonjy, 2007], where the sensitivity metric was based on the upper bound of the leave one out cross validation (LOOCV) error of the SVM model.

The embedded variable selection method based on the SVR model which is proposed in [Yang and Ong, 2011] is primarily devoted to regression. It exploits the characteristic that the SVR output can be interpreted as the conditional density function of the target, given the input variables, under the assumption that the output error is characterized by a Laplace or a Gaussian probability distribution (such interpretation that the output error is characterized by the Laplace or the Gaussian probability distributions is demonstrated in [Lin and Weng, 2004]). Thus, the proposed sensitivity metric measures the difference over the input variable space of the conditional density functions of the SVR prediction with and without the feature.

2.2.4 Hybrid Approaches

Several soft sensors applications combine several methods to promote the selection of input variables.

In [Bhartiya and Whiteley, 2001; Fortuna *et al.*, 2006] a combination of three variable selection methods was used to select the variables. The methods used were the correlation coefficient/scatter plots, partial correlation, and the Mallows Cp statistics [Mallows, 1973]. The scatter plots and correlation coefficient were used as pre-filtering, to form a preliminary subset. Then, the Cp statistics and the partial correlation were used to aid in the selection of the best subset.

In [Warne *et al.*, 2004], PCA pre-processing was applied on the variables as an unsupervised variable selection. It provided better results when compared with the variable selection methodology used in [Bhartiya and Whiteley, 2001; Fortuna *et al.*, 2006] (discussed in the previous paragraph). In [Qin, 1996], it is demonstrated that collinearity increases the variance of the MLP model, and then it is proposed to use the PLS as a pre-processing step for a MLP model, since PLS eliminates the

collinearity in the input space. The PLS together with a MLP model provided good results when compared to a single MLP.

In [Delgado *et al.*, 2009] the input variables of a fuzzy model are pre-selected from the variables of the dynamical process by means of correlation coefficients, Kohonen maps and Lipschitz quotients.

2.3 Model Choice and Training

There are two distinct model approaches applied for soft sensors development. The first is based on white-box models, obtained through a physical knowledge of the process, and the second class is based on black-box or data-driven models, based exclusively in constructing a model from empirical data of the process. Modeling by the white-box approach requires strong knowledge about the process and demands a long time of modeling work to build the models [Zahedi *et al.*, 2005]. It usually focuses on the description of the ideal steady-states, not being able to describe the real process conditions [Kadlec *et al.*, 2009]. For complex systems, the white-box modeling approach may be virtually infeasible. Black-box or data-driven models are based on empirical observations of the process (the methods themselves are empirical predictive methods). Black-box modeling is able to describe real conditions of the process, and it requires few knowledge about the system to be modeled. Nevertheless, it requires intensive work on process data. Some difficulties with these types of approaches are related to the difficulty of choosing the correct model type and structure, the functions to be used, and the quantity of function terms necessary for the development. The focus of this thesis is solely on black-box/data-driven modeling because it has shown to provide satisfactory results in soft sensors applications, with reasonable computational and design time efforts [Kadlec *et al.*, 2009; Fortuna *et al.*, 2006].

In black-box modeling, the first aspect to decide about is which kind of model is going to be used. There are always two choices: a linear model or a non-linear model. According to many authors, a linear model should always be considered before a nonlinear model. If the linear model does not provide satisfactory results, one possible explanation, besides many other possibilities, is that the system possesses a non-linear behavior, then a non-linear model should be the best choice [Nelles, 2001]. Good overviews of black-box structures for regression ranging from linear

models (e.g. PLS, LASSO, RR), to nonlinear models (e.g. NN, SVR, Fuzzy Systems (FS)) are reported in the classical books [Ljung, 1999; Haykin, 1999; Nelles, 2001; Hastie *et al.*, 2001; Bishop, 2006].

The most popular data-driven models used in soft sensors applications are the linear models with LS or PLS estimation methods [Jang *et al.*, 1997; Ma *et al.*, 2009], PCA [Jolliffe, 2002] in combination with a prediction model, NNs (mainly the MLP structure), SVRs, FS, and Neuro-Fuzzy Systems (NFS) [Shoorehdeli *et al.*, 2009; Mendes *et al.*, 2012b]. The PLS solution is the preferred and mostly applied solution in combination with linear models when comparing to LS, since it can handle data-collinearity, which is a common characteristic in industrial applications.

Soft sensors are not always composed of a single regression model. A combination of a collection of models is often employed. This is denominated an ensemble approach, which forms an ensemble of models. Ensemble methods play an important role in soft sensors applications, mainly when the number of samples for modeling is small [Soares *et al.*, 2011]. The ensemble of NN models was detailed and discussed in [Zhou *et al.*, 2002], where the authors proposed a method for building an ensemble of NN models based on GA. A related approach was used in [Soares *et al.*, 2013] where a framework to optimize the structure of an ensemble of MLP models was presented. Several MLP models with different structures were trained using the bootstrap resampling. Then, GA and simulated annealing (SA) were used to perform the optimization of the model architecture. In [Liu *et al.*, 2000], an evolutionary ensemble learning using NN and based on negative correlation learning was presented. However, [Liu *et al.*, 2000] has some shortcomings such as not considering the possibility of linear combination among models, and using pre-defined models' architectures.

Fuzzy models are knowledge-based models. In some complex applications it is difficult to tune such models. Some approaches try to overcome this difficulty by optimizing the fuzzy model using evolutionary algorithms. In [Delgado *et al.*, 2009] the TS-fuzzy model is tuned using a GA-based approach. In [Mendes *et al.*, 2012b] the work of [Delgado *et al.*, 2009] was expanded to learn the TS-fuzzy TS structure together with the selection of input variables and delays.

In almost all soft sensor applications, a single model is tuned using all available training samples, without distinguishing the operating modes of the process during the training phase. However, the existence of multiple operating modes in

a process is an inherent characteristic of most industrial applications. Sometimes multiple operating modes result from external disturbances, as for example a change in feedstock or product grade or even changes such as the diurnal load variation of a power plant or the summer-winter operation of a refinery [Matzopoulos, 2010; Wang *et al.*, 2012]. In these situations, it would be beneficial for the prediction accuracy and reasonability, to consistently train a different model for each operating mode of the process [Yu, 2012], or train a different model for each set of correlated operating modes [Facco *et al.*, 2009]; And during online operation, when a new sample is made available, the model which is the most adequate for the new sample is identified and then used to make the prediction. The identification of which model will be used is a key issue in the development [Facco *et al.*, 2009; Camacho and Picó, 2006; Lu and Gao, 2005], which can be done using expert knowledge [Facco *et al.*, 2009] or using automatic tools, such as finite mixture of Gaussian models (FMGM) [Yu, 2012].

2.4 Model Validation

The objective of the model validation step is to evaluate the capability/ability of the trained model to perform generalization to new samples. Generalization accuracy can also be used as an estimator for model ranking in a variable selection approach (e.g. in wrapper variable selection) [Duda *et al.*, 2000]. For a large data set, usually the model is learned using only a part of the data set and then the model performance is measured on the remaining data, usually called validation data set, using some performance metric, usually the MSE (e.g. lower values of MSE indicate better models) or the normalized root mean square error (NRMSE). The NRMSE is a normalized version of MSE, often expressed in percentage, which gives a more intuitive analysis on the performance of the model. For small data sets, a cross-validation technique is usually employed to evaluate the performance of the model. The common cross validation techniques are the K-fold cross validation and the leave-one-out cross validation (LOOCV). In K-fold cross validation, the training data set is randomly split into K folds, and then the model learning is performed using the samples from $(K-1)$ folds, and the resulting model is evaluated on the remaining fold, using some performance metric. This process is repeated for all K folds, and the performance of the model is the average of the performance metric on the K folds. The LOOCV is usually employed when the number of samples is

very small, and it is equivalent to the K-fold cross validation when the number of folds K is equal to the number of samples. Other approaches measure the quality of a model in terms of its accuracy-complexity trade-off (ACT), using criteria such as the AIC [Akaike, 1974], the Bayesian Information Criterion (BIC) [Schwarz, 1978], or the Cp statistics [Mallows, 1973].

For dynamic linear systems, the autocorrelation function of the residuals and the cross-correlation functions between the residuals and the input over a set of unseen data [Soderstrom and Stoica, 1989] are usually employed to evaluate the capability of the trained linear dynamic model. For non-linear dynamic systems, the work of [Billings *et al.*, 1992] has provided several metrics to evaluate non-linear dynamic models based on NN.

2.5 Soft Sensor Maintenance

During soft sensor design the historical data of the process is used to learn the soft sensor model. However, the historical data contains limited information, corresponding to a limited period of time, and possibly also focusing on a limited set of operation areas of the state space. When dealing with new events, not described in the historical data, the soft sensor tends to decrease its performance. In this context, and to overcome such performance deterioration, the objective of soft sensor maintenance is to maintain a good soft sensor response even in the presence of process variations, or some data change. Generally, this is done by updating the soft sensor model online/recursively, in batch or sample wise mode, using the incoming samples of the process (in this context the soft sensors are called "adaptive soft sensors" [Kadlec *et al.*, 2011]). From the machine learning perspective, the area of adaptive soft sensors is related to the problem of concept drift. Concept drift means that the statistical properties of the target variable changes over the time, the term concept means the object/target to be predicted [Zliobaite, 2010].

There are three types approaches commonly employed in dealing with concept drift: (1) sample selection, (2) sample weighting, and (3) ensemble learning (or learning with multiple concept descriptors) [Tsymbal, 2004]. Moreover, as already discussed before, the mostly used models in soft sensor applications are based on multivariate statistical methods (LS, PLS, PCA) or artificial intelligence techniques (NNs (mainly the MLP structure), SVRs, FS, and NFS). In adaptive soft sensors

such models can also be employed, but there is the concern regarding the learning/adaptation of parameters. The model(s) can be applied as a single model, in the sample weighting or sample selection approaches, or several models can be applied together in the ensemble approach.

2.5.1 Sample Selection

In sample selection, the idea is to select relevant samples related to the current concept. The next step is to use such samples to update or retrain the existing model. Normally, this selection is done using window-based approaches, where the samples which are inside of a window are used to retrain/update the model, while samples outside of the window are discarded. The issues of selecting the size of the window and deciding when to retrain/update the model are crucial for a successful implementation. If the selection of the window size is poorly handled, there is a danger that the soft sensor adapts to noise (if the window size is too short) or, in the case of a too long window, it can lead to limited adaptation capability [Kuncheva and Žliobaitė, 2009]. Some adaptive methods based on ANN models in the sample selection strategy were proposed in the literature. In [Lee *et al.*, 2005; Liukkonen *et al.*, 2013], a moving window was adopted to retrain the ANN model. When a new batch of samples is available the old data is dropped out of the window and the neural model is retrained adapting to the concept of the new data. In [Liukkonen *et al.*, 2013] the most relevant features were selected offline using the first part of the training data by using a forward search procedure in combination with a MLP network.

Adaptive learning methods for NFS and SVR have been proposed in the literature, and they are usually based on sample selection or ensemble learning. NFS are widely applied for prediction [Kadlec *et al.*, 2009; Mendes *et al.*, 2012b], but their parameters are usually learned offline. Online tuning of NFS can be done by Evolving Fuzzy Systems (EFS) [Angelov and Kordon, 2010]. However, the implementation of such methods is very complex and time demanding. A step-wise online learning algorithm for SVR training was proposed by [Cauwenberghs and Poggio, 2000], where the update can be done by removing or adding new support vectors. In [Wang *et al.*, 2006] it is proposed the Adaptive Kernel Learning (AKL) framework for prediction and monitoring tasks. In this case, the SVR optimization problem

was solved by the least squares approach [Suykens *et al.*, 2002]. In [Yang *et al.*, 2009], an adaptive kernel learning method was used. The examples were selected, and the exclusion of redundant examples was performed to reduce the complexity of training. It was shown to be superior to RPLS in the presented case of study.

2.5.2 Sample Weighting

In the sample weighting strategy, the samples are weighted according to their age (the importance of the samples decreases over time). The learning/adaptation of parameters is usually done using adaptive learning by means of exponentially recursive learning. The adaptive learning has relation to the recursive or online learning where each sample is presented once and only once to learn/adapt the parameters, but in adaptive learning there is the ability to forget old examples by exponentially assigning low weights to old samples, usually by setting a forgetting factor $0 < \lambda < 1$, such that the model could capture the information of the recent data [Tsymbal, 2004; Kadlec *et al.*, 2011]. Using such sample weighting approaches, there is no need to use memory to store the samples.

In the sample weighting approach, the following learning strategies have been used in the literature for the LS, PLS, and artificial neural networks (ANN) models. For the LS solution, there is the recursive LS (RLS) method, which is a well known example of recursive learning, where the coefficients of a linear model that minimize the linear least squares cost function are recursively computed. The PLS is implemented with its recursive/adaptive form, the recursive PLS (RPLS) [Dayal and MacGregor, 1997]. It is the most popular method in adaptive soft sensors [Helland *et al.*, 1992; Komulainen *et al.*, 2004; Li *et al.*, 2005; Mu *et al.*, 2006; Haavisto and Hyötyniemi, 2009; Facco *et al.*, 2010; Wang *et al.*, 2010; Kadlec and Gabrys, 2011; Muradore and Fiorini, 2012]. For the other state of the art methods, there are some adaptive learning strategies in the literature. For single layer feedforward ANN, a fast learning algorithm with offline and online solutions, called online sequential extreme learning machine (OS-ELM) was proposed in [Nan-Ying *et al.*, 2006]. All these methods are able to forget old samples by setting a forgetting factor.

2.5.3 Ensemble Learning

In the ensemble learning strategies, the goal is to construct a model for each concept in the data distribution. When a new input arrives, the final prediction value is a combination of the results of all the models built previously for all the concepts, such as a weighted average of such results. Moreover, in the ensemble method, there are two possible areas that may be subject to adaptation: at the level of the model combination, or at the level of the models. The ensemble method is less attractive because of its computational demand, necessary to process and store several models and/or samples.

Ensemble learning methods find different concepts in the historical data and learn a model for each of these concepts. In [Kadlec and Gabrys, 2011] a PLS model was constructed for each different concept found (an approach based on the PLS model error was used to determine the different concepts). The final prediction is a combination of the set of the available models, where the combination takes into account a probability of each model being responsible for the data to be predicted. The adaptation is performed at the level of model combination and at the level of recursive adaptation of the models. The authors termed this method the incremental local learning soft sensing algorithm (ILLSA). [Fu *et al.*, 2008] developed a soft sensor method using an ensemble learning strategy where a clustering method, based on the fuzzy C-means clustering (FCM) algorithm, was used to find different concepts, and then a SVR model was learned to predict in each concept. During online operation, when a new sample arrives, the FCM algorithm sets the corresponding adequate SVR model to be used to predict the output.

Chapter 3

Variable Selection Using the MLP Model

Contents

This chapter introduces a new variable selection algorithm to be utilized when working with MLP neural network models for soft sensor applications.

According to the recent review on soft sensors made by [Kadlec *et al.*, 2009], the MLP model is the most popular nonlinear regression model utilized in soft sensor applications. According with this review the MLP model covers approximately 18%

of soft sensors applications, the mostly used among nonlinear models. The most popular methods are the PLS and PCA linear models, covering approximately 38% of soft sensor applications. However, in most of the regression settings where the MLP is utilized (including in soft sensor applications), there is not a concern about the use, or not, of a variable selection algorithm, possibly because it is not given enough consideration to the impact that the choice of the input variables has on model complexity, learning difficulty, and performance of the subsequently trained MLP model [May *et al.*, 2011]. It is worth to emphasize that in the case of soft sensor applications, there is a preference in having a lower number of input variables, since it is a positive factor for decreasing implementation costs, or even making the soft sensor feasible at all. Moreover, in soft sensors applications where the expert knowledge is not sufficient to enumerate the most relevant variables it is inevitably necessary to use a variable selection algorithm. Following these considerations, in this chapter a variable selection algorithm based on the MLP model is presented. The main characteristic of the presented method is regarding its performance and its low computational time, since the network is trained only a single time, maintaining the low computational cost.

Currently, the variable selection methods used to select variables for the MLP models are based mainly in three distinct approaches: filter, wrapper, and embedded methods, as discussed in Chapter 2. The use of filter methods based on the MI criterion is a common approach adopted for variable selection in regression settings using the MLP model [Ludwig *et al.*, 2009; Xing and Hu, 2009; Souza *et al.*, 2010b; Grbić *et al.*, 2013]. As discussed in Chapter 2, the MI variable selection methods have the advantage of being fast and able to select the variables independently of the MLP model used, avoiding the drawback related to MLP model training and tuning. On the other hand, the use of the MLP model as basis for variable selection (in a wrapper approach), has the advantage of attaining better results when comparing to the MI based approaches, under the drawback of being a very time demanding approach (the demanded time is related to the tasks of training and tuning the MLP model), mainly in the presence of many input variables [Romero and Sopena, 2008a; Chatterjee and Bhattacherjee, 2011]. To reduce the computational demand needed by the wrapper approach, and still select the variables based on the MLP model, one can use embedded approaches, based on sensitivity analysis [Garson, 1991; Dimopoulos *et al.*, 1995, 1999; Gevrey *et al.*, 2003; Yeh and Cheng, 2010;

Castellano and Fanelli, 2000; Lemaire and Féraud, 2006] or regularization methods [Bishop, 1995b; Chapados and Bengio, 2001; Similä and Tikka, 2009]. According with [Romero and Sopena, 2008a] the most popular form of variable selection based on the MLP model is the one that utilizes the MLP prediction error, or other criterion based on sensitivity analysis of MLP model, as the fitness function, together with the SBS procedure (Appendix B). The use of a criterion based on sensitivity analysis is most common due its low computational cost when compared with the approach that uses the prediction error, because in the later approach it is necessary to retrain the model several times. However, the approach that uses the prediction error can provide better results when compared to the one based on sensitivity analysis.

The approach proposed here is a hybrid of wrapper and embedded methods, which tries to approximate the response/results of variable selection based on the MLP prediction error and the SBS search procedure, called here as SBS-MLP, but with much less computational effort. In the rest of the chapter, during the derivation of the proposed variable selection method, some notations and notions about regression models and neural network learning will be given. After the derivation of the proposed variable selection method, in the experimental part, the proposed method is applied in one artificial data set and three real-world datasets. For comparison purposes, the following variable selection algorithms were implemented, two filter methods based on MI criterion proposed in [Peng *et al.*, 2005] and [Estévez *et al.*, 2009], the embedded method proposed in [Castellano and Fanelli, 2000], and the wrapper method based on SBS-MLP [Marill and Green, 1963; Romero and Sopena, 2008a]. Moreover, it has been shown that the proposed method has similar prediction performance when compared to the traditional SBS-MLP algorithm, and has the advantage of having lower computation cost. The proposed method presents similar or better approximation performance when compared to the other four methods.

3.1 Notation

The notation used in this chapter is defined as follows. Considering a collection of input variables $x_1, \ldots, x_D$, such a collection can be collectively represented or organized as set of variables $X = \{x_1, \ldots, x_D\}$, and the vector of variables $\mathbf{x} = [x_1, \ldots, x_D]^T$ can also be defined. Let $\mathbf{x} = [x_1, \ldots, x_D]^T$ and y be defined as an

input and output sample, respectivelly. The values of these variables at time instant i are given by $\mathbf{x}(i) = [x_1(i), \ldots, x_D(i)]^T$ and $y(i)$. Matrix $\mathbf{X} = [X_{ij}] = [\mathbf{x}_1, \ldots, \mathbf{x}_D] \in \mathbb{R}^{k \times D}$, with elements $X_{ij} = x_j(i)$, and vector $\mathbf{y} = [y_{i,1}] = [y(1), \ldots, y(k)]^T \in \mathbb{R}^{k \times 1}$, with elements $y_{i,1} = y(i)$ are, respectively, the input matrix and output vector containing all the k exemplars (samples). $\mathbf{x}_j = [x_j(1), \ldots, x_j(k)]^T \in \mathbb{R}^{k \times 1}$, for $j = 1, \ldots, D$. Define the set $X_{-r} = X \setminus \{x_r\}$, and define $\mathbf{x}_{-r}^T$ as the input vector without variable r. Matrix $\mathbf{X}_{-r} = [\mathbf{x}_1, \ldots, \mathbf{x}_{r-1}, \mathbf{x}_{r+1}, \ldots, \mathbf{x}_D]$ is the matrix $\mathbf{X}$ with the variable r removed, or for a sample i, $\mathbf{x}_{-r}(i) = [x_1(i), \ldots, x_{r-1}(i), x_{r+1}(i), \ldots, x_D(i)]^T$. Moreover, $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_D$, and $\mathcal{Y}$, denote the space of input variables values and the space of output values, respectively, where $\mathcal{X} \subset \mathbb{R}^D$ and $\mathcal{Y} \subset \mathbb{R}$. A set of k data exemplars is denoted as $\Phi = \{(\mathbf{x}(i), y(i)); i = 1, \ldots, k\}$, where k is a number of time instants.

3.2 Regression Models

In the single output regression problem, the objective is to model the relationship from input variables that are the components of $\mathbf{x}$, to a target variable y, given a set of examples $\Phi = \{(\mathbf{x}(i), y(i)); i = 1, \ldots, k\}$. Often, in regression literature (e.g. [Bishop, 1995b; Hastie *et al.*, 2001]), y is assumed to be approximated by a deterministic function $f(\mathbf{x}, \boldsymbol{\theta})$, governed by a parameter vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{D(\theta)}]^T$:

$$y(i) = f(\mathbf{x}(i), \boldsymbol{\theta}) + \xi(i), \tag{3.1}$$

where $D(\theta)$ is the number of parameters of the model, $\xi(i) = y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})$, is defined as the approximation error, and ξ is assumed to be modeled by a zero mean random distribution with standard deviation σ and variance $\omega = \sigma^2$. Assuming ξ has a stationary zero-mean Gaussian distribution, then its pdf is given by:

$$p(\xi(i)) = \mathcal{N}(\xi(i)|0, \omega) = \frac{1}{\sqrt{2\pi\omega}} \exp\left(-\frac{\xi^2(i)}{2\omega}\right), \tag{3.2}$$

where σ does not depend on $\mathbf{x}$ or on i. By combining (3.1) and (3.2), the conditional probability of y given the input $\mathbf{x}$ can be represented by the normal distribution:

$$p(y(i)|\mathbf{x}(i)) = \mathcal{N}(y(i)|f(\mathbf{x}(i), \boldsymbol{\theta}), \omega) = \frac{1}{\sqrt{2\pi\omega}} \exp\left(-\frac{(y(i) - f(\mathbf{x}(i), \boldsymbol{\theta}))^2}{2\omega}\right). \tag{3.3}$$

Then, under the assumption that the examples in Φ are i.i.d., the maximum likelihood (ML) estimator can be used to estimate the parameters $\boldsymbol{\theta}$ and ω of (3.3), as follows:

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta}^*} \prod_{i=1}^{k} \mathcal{N}(y(i)|f(\mathbf{x}(i), \boldsymbol{\theta}^*), \omega) = \arg\min_{\boldsymbol{\theta}^*} \sum_{i=1}^{k} (y(i) - f(\mathbf{x}(i), \boldsymbol{\theta}^*))^2, \quad (3.4)$$

$$\omega = \arg\max_{\boldsymbol{\omega}^*} \prod_{i=1}^{k} \mathcal{N}(y(i)|f(\mathbf{x}(i), \boldsymbol{\theta}), \omega^*) = \frac{1}{k} \sum_{i=1}^{k} (y(i) - f(\mathbf{x}(i), \boldsymbol{\theta}))^2. \quad (3.5)$$

The expression

$$\text{RSS}(\mathbf{y}, f(\mathbf{X}, \boldsymbol{\theta}^*)) = \sum_{i=1}^{k} (y(i) - f(\mathbf{x}(i), \boldsymbol{\theta}^*))^2 \quad (3.6)$$

in (3.4) is known as the residual sum of squares (RSS). If, instead of the sum, the average of the squared residuals over all k samples is considered, then the well known MSE is defined:

$$\text{MSE}(\mathbf{y}, f(\mathbf{X}, \boldsymbol{\theta}^*)) = \frac{1}{k} \sum_{i=1}^{k} (y(i) - f(\mathbf{x}(i), \boldsymbol{\theta}^*))^2. \quad (3.7)$$

3.3 MLP Neural Network Learning

This section reviews the MLP neural network learning method. From [Hornik *et al.*, 1989] it is known that a MLP neural network model with two-layers (Figure 3.1), a sufficient number of neurons in the hidden layer, h, and proper weights can uniformly approximate any continuous function, i.e. the MLP model is an universal approximator. In the MLP model, the function $f(\mathbf{x}, \boldsymbol{\theta})$ in (3.1) has the following parametrized form:

$$\hat{y}(i) \;=\; f(\mathbf{x}(i); \boldsymbol{\Lambda}) \;=\; \psi\left(g\left(\mathbf{x}^T(i)\mathbf{W}_I + \mathbf{b}_I\right)\mathbf{w}_O + b_O\right), \quad (3.8)$$

where $f(\mathbf{x}; \boldsymbol{\Lambda})$ is the MLP network output, $\boldsymbol{\Lambda} = \{\mathbf{W}_I, \mathbf{b}_I, \mathbf{w}_O, b_O\}$ is the set of weight and bias parameters, $\mathbf{x}(i)$ is the input vector at time instant (i), $\mathbf{W}_I = \mathbf{W} = [w_{lj}]$ is the $D \times h$ matrix of the weights connecting the D inputs to the h hidden layer nodes, $\mathbf{b}_I = \mathbf{b} = [b_1, \ldots, b_h]^T$ is the vector of biases of the hidden layer nodes. The output weights that connect the hidden neurons to the output neuron, and the output bias, are represented by $\mathbf{w}_O = [w_{O1}, \ldots, w_{Oh}]^T$ and b_O, respectively. $g(\cdot)$ and

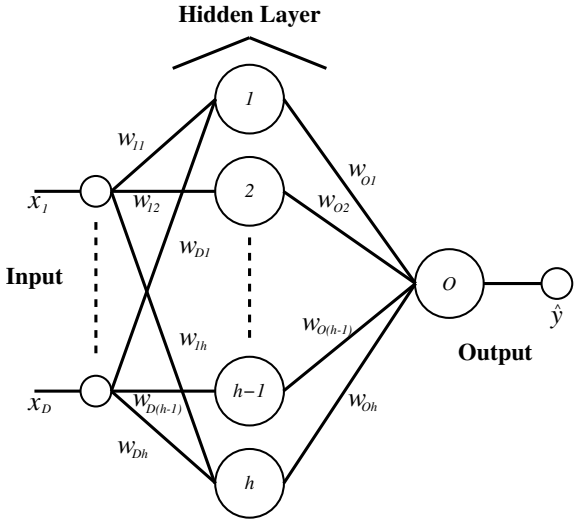


Figure 3.1: Topology of an MLP neural network with two layers; O is the output node, $\mathbf{W}_I = \mathbf{W} = [w_{ij}]$ is the $D \times h$ matrix of the weights connecting the inputs to the h hidden layer nodes, and $\mathbf{w}_O = [w_{O1}, \ldots, w_{Oh}]^T$ is the output weights vector. The hidden layer biases $\mathbf{b}_I$ and the output bias b_O are omitted to simplify the diagram.

$\psi(\cdot)$, represent the activation functions of the nodes of the hidden layer, and output layer, respectively. In this thesis $\psi(\cdot)$ is a linear function, and $g(\cdot)$ is tangent sigmoid function defined by:

$$g(a) = \frac{2}{1 + e^{2a}} - 1, \tag{3.9}$$

which is bounded between -1 and $+1$. For a vector $\mathbf{a} = [a_1, \ldots, a_A]^T \in R^A$ the output of the tangent sigmoid is defined as $g(\mathbf{a}) = [g(a_1), \ldots, g(a_A)]^T$. In the rest of this chapter, for simplicity, and as defined in (3.8), the predicted output $f(\mathbf{x}(i); \mathbf{\Lambda})$ is often denoted as $\hat{y}(i) = f(\mathbf{x}(i); \mathbf{\Lambda})$.

Since it is assumed that $f(\mathbf{x}(i), \boldsymbol{\theta})$ in (3.1) is equal to $f(\mathbf{x}(i); \mathbf{\Lambda})$ in (3.8), then the learning of the MLP parameters $\mathbf{\Lambda}$ given a training data set Φ, is done so as to perform the optimization of (3.4). The MLP is then trained by using some method for minimizing the following equation,

$$E(f(\mathbf{X}; \mathbf{\Lambda}), \mathbf{y}) = \sum_{i=1}^{k} \left(f(\mathbf{x}(i); \mathbf{\Lambda}) - y(i)\right)^2. \tag{3.10}$$

For example, if the error backpropagation algorithm [Rumelhart *et al.*, 1986; Werbos, 1990] is applied for training, the weights and bias in $\mathbf{\Lambda}$ are learned using the following gradient-based delta-rule:

$$w_{ij} \leftarrow w_{ij} - \eta \Delta w_{ij}, \tag{3.11}$$

$$b_j \leftarrow b_j - \eta \Delta b_j, \tag{3.12}$$

$$\Delta w_{ij} = \frac{\partial E}{\partial w_{ij}}, \tag{3.13}$$

$$\Delta b_j = \frac{\partial E}{\partial b_j}, \tag{3.14}$$

where η is the learning rate, $i = 1, \ldots, D, O$, and $j = 1, \ldots, h, O$. There are many variations of the backpropagation algorithm, such as the Quickprop [Fahlman, 1988], the Levenberg-Marquardt algorithms [Hagan and Menhaj, 1994], and the conjugate gradient method [Fletcher and Reeves, 1964]. Apart from the backpropagation algorithms, the parameters $\mathbf{\Lambda}$ can be determined through other optimization algorithms, such as genetic algorithms [Montana and Davis, 1989], or simulated annealing [Sexton *et al.*, 1999], among others.

3.4 Proposed Variable Selection Algorithm

In this section, the proposed variable selection method is derived. The proposed algorithm is a fast implementation of a variable selection that utilizes the MLP prediction error (3.10) as the fitness function, together with the SBS procedure (Appendix B), defined here by the acronyms SBS-MLP. It was inspired in the method proposed in [Castellano and Fanelli, 2000], which for the sake of simplicity is called here the Iteratively Adjusted Neural Network (IANN). The IANN uses an already trained neural network model, and sequentially removes the useless variables according to an exclusion criterion based on the values of the input weights. However, when a variable is removed the IANN performs an adjustment of the existing model instead of retraining again all the network.

The method proposed in this chapter uses the same adjustment rule of the IANN when a variable is removed, but differs on the exclusion criterion. This new exclusion criterion is based on the MLP prediction error (3.10), as the traditional SBS-MLP [Romero and Sopena, 2008a], and at each iteration the variable selected to be removed is the one which contributes least to predict the target output.

3.4.1 Motivation of the Variable Selection Algorithm

The variable selection proposed here holds on the following assumption:

Assumption 1. *Let the difference between the errors of two MLP models, both with h hidden neurons, and one trained with all the variables and the other trained with a subset of variables, be equal to some value ϵ:*

$$E^{all} - E^{rel} = \epsilon, \tag{3.15}$$

where E^{all} and E^{rel} are the errors of the model trained with all the variables, and with the subset of variables, respectively. Then, assume that $|\epsilon|$ is small if and only if (iff) the subset of variables is a sufficient subset of relevant variables. A subset of relevant variables is sufficient iff it contains all the necessary input variables which are required to correctly predict the target variable.

From the above assumption, the error of an MLP model trained with all variables provides similar results when compared with the error of an MLP model trained with a subset of sufficient relevant variables.

3.4.2 MLP Network Adjustment

Assume a MLP model with two layers trained with a dataset Φ by minimizing the error function (3.10) to obtain an approximator of the form (3.8). When a variable $\mathbf{x}_r$ is removed from $\mathbf{X}$, such operation generates a new/derived dataset $\Phi_{-r} = \left\{\left(\mathbf{x}_{-r}^T(i), y(i)\right); i = 1, \ldots, k\right\}$. Retraining a MLP network using the Φ_{-r} dataset with h hidden neurons, the following model is generated:

$$\hat{y}_{-r}(i) = \psi\left(g\left(\mathbf{x}_{-r}^T(i)\mathbf{W}_{I,-r} + \mathbf{b}_{I,-r}\right)\mathbf{w}_{O,-r} + b_{O,-r}\right), \tag{3.16}$$

where $\mathbf{W}_{I,-r}$, $\mathbf{b}_{I,-r}$, $\mathbf{w}_{O,-r}$ and $b_{O,-r}$ are the new matrix of input weights, input bias, output matrix weights, and output bias, respectivelly.

From Assumption 1, the error of an MLP model trained with all variables provides similar results when compared to the error of an MLP model trained with a subset of sufficient relevant variables. Thus, the error $E(\hat{\mathbf{y}}, \mathbf{y}) - E(\hat{\mathbf{y}}_{-r}, \mathbf{y}) = \epsilon$ is small if variable r is irrelevant. On the other hand, if variable r is relevant the error

ϵ will be large. Then, the following approximation holds:

$$E(\hat{\mathbf{y}}, \mathbf{y}) = E(\hat{\mathbf{y}}_{-r}, \mathbf{y}) + \epsilon, \tag{3.17}$$

$$\sum_{i=1}^{k} [\hat{y}(i) - y(i)]^2 = \sum_{i=1}^{k} [\hat{y}_{-r}(i) - y(i)]^2 + \epsilon. \tag{3.18}$$

Define the values $\hat{\epsilon}(i)$ to be the contributions of the individual errors between the models (with and without $\mathbf{x}_r$) for each sample i, so that

$$\hat{y}_{-r}(i) + \hat{\epsilon}(i) = \hat{y}(i), \tag{3.19}$$

and so that (3.18) is valid. This means that for a sample i, the output of one model is equal to the output of the other plus an error $\hat{\epsilon}(i)$. Then,

$$\sum_{i=1}^{k} [\hat{y}(i) - y(i)]^2 = \sum_{i=1}^{k} \left\{[\hat{y}_{-r}(i) + \hat{\epsilon}(i)] - y(i)\right\}^2, \tag{3.20}$$

$$\sum_{i=1}^{k} \hat{y}(i) = \sum_{i=1}^{k} [\hat{y}_{-r}(i) + \hat{\epsilon}(i)]. \tag{3.21}$$

For a sample i, and from (3.8), (3.16), (3.19), or by replacing (3.16) in (3.21):

$$\psi\left(g\left(\mathbf{x}^T(i)\mathbf{W}_I + \mathbf{b}_I\right)\mathbf{w}_O + b_O\right) = \psi\left(g\left(\mathbf{x}_{-r}^T(i)\mathbf{W}_{I,-r} + \mathbf{b}_{I,-r}\right)\mathbf{w}_{O,-r} + b_{O,-r}\right) + \hat{\epsilon}(i) \tag{3.22}$$

Setting the output weights on the right hand side of equation (3.22), equal to $\mathbf{w}_{O,-r} = \mathbf{w}_O$ and $b_O = b_{O,-r}$ and inserting the error $\hat{\epsilon}(i)$ into the first layer of MLP model, equation (3.22) becomes:

$$\psi\left(g\left(\mathbf{x}^T(i)\mathbf{W}_I + \mathbf{b}_I\right)\mathbf{w}_O + b_O\right) = \psi\left(g\left(\mathbf{x}_{-r}^T(i)\mathbf{W}_{I,-r} + \mathbf{b}_{I,-r} + \boldsymbol{\Delta}(i)\right)\mathbf{w}_O + b_O\right), \tag{3.23}$$

where $\boldsymbol{\Delta}(i)$ is the $h \times 1$ vector representation of $\hat{\epsilon}(i)$ into the first layer of the MLP model, so that equation (3.22) is valid. Then, assuming in the input layer, that the bias remain equal, $\mathbf{b}_I = \mathbf{b}_{I,-r}$, and that the weights are $\mathbf{W}_{I,-r} = [w_{lj} + \delta_{lj}]$ for $l = 1, \ldots, r-1, r+1, \ldots, D$, and $j = 1, \ldots, h$, it is possible to update the weights without retraining the network using the parameters of the network trained with all variables. After a variable x_r and the associated weights w_{rj} $(j = 1, \ldots, h)$ are

removed, the remaining weights w_{lj} are adjusted using factors δ_{lj} obtained from the following $h \times k$ equations:

$$\sum_{x_l \in X} \left[x_l(i) w_{lj} \right] = \sum_{x_l \in X_{-r}} \left[x_l(i)(w_{lj} + \delta_{lj}) \right], \quad j = 1, \ldots, h, \;\; i = 1, \ldots, k, \tag{3.24}$$

where for all $l \in L_{-r} = \{\, l : x_l \in X_{-r} \,\}$, and $j = 1, \ldots, h$, δ_{lj} is the adjustment to weight w_{lj} that connects input x_l to hidden neuron j. As can be noticed in (3.24), the δ_{lj} are the adjustment factors for the weights that remain in existence after removing input variable x_r, and so that the input to each node in the hidden layer remains constant. Simple algebraic manipulations of (3.24) yield:

$$\sum_{x_l \in X_{-r}} \delta_{lj} x_l(i) = w_{rj} x_r(i), \quad j = 1, \ldots, h, \;\; i = 1, \ldots, k. \tag{3.25}$$

Equation (3.25) can be rewritten in matrix form, as follows:

$$\mathbf{X}_{-r} \hat{\mathbf{\Delta}} = \mathbf{x}_r \mathbf{w}_r^T, \tag{3.26}$$

$$\mathbf{X}_{-r} \hat{\mathbf{\Delta}} = \mathbf{B}, \tag{3.27}$$

where $\mathbf{B} = [B_{ij}] = [w_{rj} x_r(i)] = [\mathbf{b}_1, \ldots, \mathbf{b}_h]$, and $\hat{\mathbf{\Delta}} = [\hat{\Delta}_{lj}] = [\delta_{lj}] = [\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_h]$ for $j = 1, \ldots, h$, $i = 1, \ldots, k$, and for all $l \in L_{-r}$, where $\boldsymbol{\delta}_j$ is the vector of adjustments of the weights connected to the hidden node j. A way to solve (3.27) for the unknowns δ_{lj} (for all $l \in L_{-r}$, and $j = 1, \ldots, h$) is to use the conjugate gradient precondition normal equation (CGPCNE) method [Björck and Elfving, 1979], which provides a good and fast least-squares solution. The CGPCNE is described in Algorithm 3.1.

Thus, the parameters of the MLP model without the variable x_r can be approximated using an adjustment of the input weights instead of retraining the network

$$\hat{y}^*_{-r} = \psi \left(g \left(\mathbf{x}^T_{-k} \mathbf{W}^*_{I,-r} + \mathbf{b}_I \right) \mathbf{w}_O + b_O \right), \tag{3.28}$$

where $\hat{y}^*_{-r}$ is the new output prediction, and $\mathbf{W}^*_{I,-r}$ is the matrix of input weights updated/adjusted according to the δ_{ij} obtained from (3.25).

The computation complexity per iteration of the MLP trained by the backpropagation algorithm is proportional to the total number of network weights $|W| =$

Algorithm 3.1 CGPCNE Algorithm

Inputs: $\mathbf{X}_{-r}$ input matrix with variable r removed; matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_h]$ from (3.27); $s_{\max}$ is the maximum number of iterations;
Output: δ_{lj} for all $l \in L_{-r}$, and $j = 1, \ldots, h$;
Initialization: Set a value for the convergence rate ω;
Initialize the iteration integer $s = 0$;
Set an initial value for $\hat{\Delta}_{lj}^{(0)} = \delta_{lj}^{(0)}$, for all $l \in L_{-r}$, and $j = 1, \ldots, h$;
Compute:
$\mathbf{D} = \text{diag}\left(||\mathbf{x}_1||_2^2, \ldots, ||\mathbf{x}_n||_2^2\right)$;
$\mathbf{L} = [L_{et}]$, where $\mathbf{L}$ is strictly lower diagonal, and $L_{et} = \mathbf{x}_e^T \mathbf{x}_t$ for $e > t$, $e = 1, \ldots, D$, and $t = 1, \ldots, D$;
$\mathbf{C}_w = (\mathbf{D} + w\mathbf{L})\mathbf{D}^{\frac{1}{2}}$;
for $j = 1, \ldots, h$ **do**
    $\mathbf{r}_j^{(0)} = \mathbf{b}_j - \mathbf{X}_{-r}\boldsymbol{\delta}_j^{(0)}$;
    $\mathbf{p}_j^{(0)} = \mathbf{m}_j^{(0)} = \mathbf{C}_w^{-1}\mathbf{X}_{-r}^T\mathbf{r}_j^{(0)}$
    for $s = 0$ to $s_{\max}$ **do**
        $\mathbf{q}_j^{(s)} = \mathbf{X}_{-r}\mathbf{C}_w^{-T}\mathbf{p}_j^{(s)}$,
        $\alpha_j^{(s)} = \frac{||\mathbf{m}_j^{(s)}||_2^2}{||\mathbf{q}_j^{(s)}||_2^2}$,
        $\boldsymbol{\delta}_j^{(s+1)} = \boldsymbol{\delta}_j^{(s)} + \alpha_j^{(s)}\mathbf{C}_w^{-T}\mathbf{p}_j^{(s)}$,
        $\mathbf{r}_j^{(s+1)} = \mathbf{r}_j^{(s)} - \alpha_j^{(s)}\mathbf{q}_j^{(s)}$,
        $\mathbf{m}_j^{(s+1)} = \mathbf{C}_w^{-1}\mathbf{X}_{-r}^T\mathbf{r}_j^{(s+1)}$
        $\beta_j^{(s)} = \frac{||\mathbf{m}_j^{(s+1)}||_2^2}{||\mathbf{m}_j^{(s)}||_2^2}$,
        $\mathbf{p}_j^{(s+1)} = \mathbf{m}_j^{(s+1)} + \beta_j^{(s)}\mathbf{p}_j^{(s)}$,
        The convergence can be checked by computing $C_o(s) = ||\mathbf{X}_{-r}\boldsymbol{\delta}_j^{(s+1)} - \mathbf{b}_j||_2$. This value is monotically decreasing at each iteration. In a practical perspective, the s-loop ($s = 0$ to $s_{\max}$) can be stopped after attaining $C_o(s) < C_o^{\lim}$ for some small positive $C_o^{\lim}$.
    end for
end for
return $\hat{\boldsymbol{\Delta}} = [\boldsymbol{\delta}_1^{(s+1)}, \ldots, \boldsymbol{\delta}_h^{(s+1)}]$

$h(2 + D) + 1$ and to the number of samples k, so that the overall complexity is equal to $\mathcal{O}(ek|W|)$, where e is the number of training iterations. On the other hand, the computation complexity of each iteration of the CGPCNE to determine the δ_{ij}'s values is proportional to the number of input weights and the number of available samples. Generally, the number of iterations needed to solve equations (3.25) is very small, in a way that the overall computation complexity associated with the CGPCNE algorithm is $\mathcal{O}(k|W_I|)$, where $|W_I| = hD$ is the total number of input weights. Clearly, adjusting the network with the δ_{ij} is computationally much cheaper than to retrain the MLP several times.

3.4.3 Ranking Criterion

In the IANN algorithm the variable x_r selected to be removed in each iteration tends to be the one which has the smallest input weights [Garson, 1991]. More specifically, the variable r selected to be removed at each iteration, is the one that minimizes the following criterion:

$$r = \underset{\{l\,:\,x_l \in X\}}{\arg\min} \sum_{j=1}^{h} \sum_{i=1}^{k} \left[x_l(i) w_{lj}\right]^2, \tag{3.29}$$

where without loss of generality it is assumed that variables x_l, $x_l \in X$ are normalized to have zero mean and unit variance. This variable can be interpreted as the input having the smallest total amount of feedforward propagated information, or it can also be seen as the variable x_r with the smallest energy with respect to the trained network. However, this method has the disadvantage of failing to remove redundant variables which are very commonly occurring in soft sensor applications.

In this chapter, the importance of a variable x_r is measured by considering whether the removal of x_r reduces or increases the error (3.10); it is the same criterion used in the SBS-MLP algorithm. An error reduction indicates that the absence of x_r is irrelevant to the model and an increase suggests that it is relevant to the model. In this way, the exclusion evaluation function of x_r is defined as the following difference:

$$\mathcal{E}_{-r} \;=\; E(\hat{\mathbf{y}}_{-r}, \mathbf{y}) - E(\hat{\mathbf{y}}, \mathbf{y}). \tag{3.30}$$

The above equation computes the difference of the error (3.10) in the presence and absence of x_r, so the higher the value of $\mathcal{E}_{-r}$, the more important is the r-th variable. Moreover, a negative value of $\mathcal{E}_{-r}$ indicates that x_r is irrelevant to the model.

Algorithm 3.2 Steps of the proposed variable selection scheme

1: **Input:** Dataset: $\Phi = \{(\mathbf{x}(i), y(i))\ ;\ i = 1, \ldots, k\}$; Set of variables: X; Ordered set of variables, initially empty $S := \emptyset$;
2: **Output:** Ordered set S, containing the features ranked (in decresing order) according to their importance;
3: Set $\mathcal{MLP}(\Phi) \leftarrow$ "A MLP trained with dataset Φ";
4: **for** $s = D$ down to 1 **do**
5: For each $x_j \in X$, compute $\mathcal{E}^*_{-j}$, eq. (3.31), and let $x_r = \arg\min_{x_j \in X} \left(\mathcal{E}^*_{-j}\right)$.
6: $X \leftarrow (X_{-r} = X \setminus \{x_r\})$, "Remove variable x_r that has the lowest value of $\mathcal{E}^*_{-r}$".
7: Set $S \leftarrow S \cup \{x_r\}$, "Update the set S, adding x_r in the s-th position".
8: Set $\mathcal{MLP}(\Phi_{-r}) \leftarrow$ "MLP network updated by removing input x_r, and adjusting the remaining weights according to (3.24), and $w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \delta_{ij}$".
9: **end for**

In the proposed method, the output $\mathbf{y}^*_{-r}$ of the adjusted network is used instead of the output $\mathbf{y}_{-r}$ of the retrained network in (3.30). This means that instead of retraining the network to obtain (3.16), the input weights are adjusted using the method described in Subsection 3.4.2, generating model (3.28). Thus, the exclusion evaluation function (3.30) is redefined as:

$$\mathcal{E}^*_{-r} \;=\; E(\hat{\mathbf{y}}^*_{-r}, \mathbf{y}) - E(\hat{\mathbf{y}}, \mathbf{y}). \tag{3.31}$$

The proposed variable selection method is detailed in Algorithm 3.2, where the loop starts with a trained MLP network $\mathcal{MLP}(\Phi)$. At each iteration of the loop, the measure of importance of each variable x_j is calculated. This is done by temporarily removing x_j from the dataset Φ, readjusting the network using (3.24) and subsequently measuring the importance of x_j using (3.31). In each iteration of the proposed algorithm, the least important variable x_r is selected and removed from the MLP network. Then, the MLP is readjusted according to the removal of x_r (retaining the most favorable network). At the output of the algorithm, the set S contains the input variables ranked (ordered in a selection rank) in decreasing order of importance (note Step 7) according to the exclusion evaluation function (3.31). After variable ranking, the selection of relevant variables can proceed by selecting the first m variables of the ordered set S.

Table 3.1: Summary of the data sets.

Data set	**#Inputs**	**#Train**	**#Test**	h	**#Epochs**
Friedman	10	250	250	8	500
Box-Jenkins	2	145	145	4	80
Gas-Mileage	6	196	196	3	100
WTP	55	176	176	10	200

3.5 Experimental Results

This section presents experimental results of the proposed variable selection algorithm in one artificial-domain and three real-world prediction problems. A summary of the datasets is given in Table 3.1. In the experiments, for all data sets presented in Table 3.1, the proposed variable selection method will be compared with the IANN [Castellano and Fanelli, 2000], SBS-MLP [Marill and Green, 1963; Romero and Sopena, 2008a], mRMR [Peng *et al.*, 2005], and NMIFS [Estévez *et al.*, 2009] variable selection algorithms. For all algorithms and datasets, half of the available data was used for training and the other half was used for test. All the considered MLP networks have one hidden layer with a tangent hyperbolic activation function, a linear activation function at the output layer, and are trained with the Levenberg-Marquardt error backpropagation algorithm [Hagan and Menhaj, 1994] in batch mode. The weights were initialized using the Nguyen-Widrow method [Nguyen and Widrow, 1990].

In all data sets, the methods were evaluated using both the normalized root mean square error (NRMSE) on the test data, and the execution time. In all methods, the execution time was considered as the time necessary to rank all the variables with the respective criteria. The NRMSE is defined as:

$$\text{NRMSE} = \frac{\sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(y(i)-\hat{y}(i)\right)^2}}{\max(\mathbf{y})-\min(\mathbf{y})}, \tag{3.32}$$

where $y(i)$, and $\hat{y}(i)$ are the observed and predicted targets, respectively, and $\max(\mathbf{y})$, and $\min(\mathbf{y})$ are the maximum and minimum values of the observed target. NRMSE is often expressed in percentage. As can be noticed, the NRMSE is equivalent to the MSE, since $\text{NRMSE} = \frac{\sqrt{\text{MSE}}}{\max(\mathbf{y})-\min(\mathbf{y})}$. The use of this criterion, instead of MSE,

to perform the evaluation of the models, is because the NRMSE criterion is more intuitive. The closer the NRMSE (or the MSE) is to 0 the better is the quality of prediction. In a practical prediction perspective, a NRMSE value of less than 10% is acceptable.

The optimal number of hidden neurons h used in all methods where such h applies was determined by training the MLP model with all variables in a 10-fold cross-validation [Kohavi, 1995] scheme using the training data set. The selected number of hidden neurons was the one that produced the smalest cross-validation MSE among these ten realizations. The number of hidden neurons used in each data set is given in Table 3.1. Additionally, the computations of the exclusion criterion in the MLP model for use in the SBS-MLP algorithm and in the proposed algorithm, were performed by using a 10-fold cross-validation scheme using the training data set. The values considered to decide the exclusion were the average cross-validation of the respective criteria, among these ten realizations.

3.5.1 Artificial Dataset

The Friedman artificial dataset [Friedman, 1991] consists of 10 input variables $\mathbf{x} = [x_1, x_2, \ldots, x_{10}]^T$ generated independently of each other and uniformly distributed over $[0, 1]$. The target variable y is a function of the first five variables:

$$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \mathcal{N}(0, 1), \tag{3.33}$$

where $\mathcal{N}(0, 1)$ is Gaussian noise with zero mean and unit variance. Thus, variables x_1, x_2, x_3, x_4, x_5 are relevant, while the remaining variables are irrelevant. In this dataset the focus is to see the capability of all methods in the selection the relevant variables, and in the removal of the irrelevant variables.

Table 3.2 indicates the selected variables for all methods. All the methods except the IANN method have the capability of correctly selecting the set of relevant input variables. Figure 3.2a shows the error rates on the Friedman test dataset as a function of the number of top ranked variables. It is possible to note that, for all methods except IANN, the error starts to increase after the relevant variables are selected. In the IANN method, the irrelevant variables x_6, x_9 and x_{10} were selected in addition to the five relevant variables. Since the proposed method, the SBS-MLP, the NMIFS, and the mRMR have selected correctly the relevant variables, the fitting

Table 3.2: Selected variables on the Friedman dataset. The selected variables are indicated with a (✓), while the non selected are indicated with a (✕).

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
Proposed	✓	✓	✓	✓	✓	✕	✕	✕	✕	✕
IANN	✓	✓	✓	✓	✓	✓	✕	✕	✓	✓
SBS-MLP	✓	✓	✓	✓	✓	✕	✕	✕	✕	✕
NMIFS	✓	✓	✓	✓	✓	✕	✕	✕	✕	✕
mRMR	✓	✓	✓	✓	✓	✕	✕	✕	✕	✕

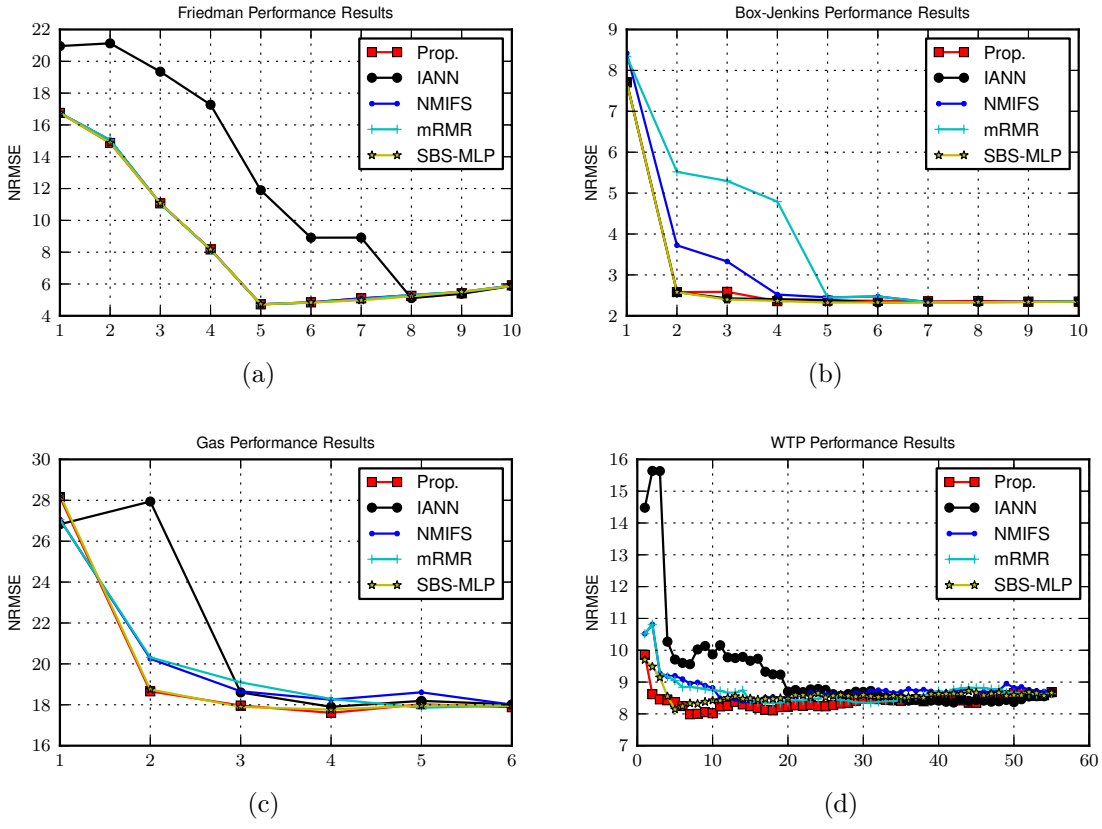


Figure 3.2: Error rates on the test dataset as a function of the number of top ranked variables used in the prediction model on the (a) Friedman, (b) Box Jenkins, (c) Gas Milege, and (d) WTP data sets.

performance in the test data was similar for all of them, as indicated in Table 3.3. Additionally, as expected, the selection of irrelevant variables by the IANN led to a loss of fitting performance in the test data, when compared with the other methods. Concerning the computational time needed to select the set of input variables, the slowest method was the SBS-MLP, followed by the proposed method, and by IANN, and the mRMR and NMIFS filter methods.

3.5.2 Benchmark Datasets

Two benchmark data sets are evaluated, the Box-Jenkins data set and the Gas Mileage data set. A brief description of the benchmark datasets is given as follows.

Box-jenkins: The Box-Jenkins gas furnace process data[1] was recorded from a combustion process of a methane-air mixture, and consists of 296 data points $(y(i), x(i))$. The input $x(i)$ is the gas flow rate into the furnace and the output $y(i)$ is the carbon dioxide (CO_2) concentration in the outlet gas. The sampling interval is 9 [s]. To predict $y(i)$, the following set of possible variables and delays is considered and examined $X = \{y(i-1), y(i-2), y(i-3), y(i-4), x(i-1), x(i-2), x(i-3), x(i-4), x(i-5), x(i-6)\}$.

Gas mileage: The automobile gas mileage data set corresponds to a problem of predicting the number of miles per gallon (MPG). The gasoline consumption needs to be predicted based on the following input variables $x_1, x_2, x_3, x_4, x_5, x_6$, respectively: number of cylinders, displacement, horsepower, weight, acceleration, and model year. The original data is available at the UCI (University of California at Irvine) Machine Learning Repository[2]. The set of considered input variables is $X = \{x_1, x_2, x_3, x_4, x_5, x_6\}$.

For the Box-jenkins data set, all the methods achieved similar performance results in terms of NRMSE, as indicated in Table 3.3. However, the proposed method and the SBS-MLP method selected the lowest number of variables (only five; however, only three variable shows to have enough information regarding the output), while the other tested algorithms, IANN, NMIFS and mRMR, have selected seven

[1]Provided by the IEEE Neural Networks Council Standards Committee Working Group on Data Modeling Benchmarks. Available: http://www.stat.wisc.edu/~reinsel/bjr-data/gas-furnace .

[2]Available: http://archive.ics.uci.edu/ml/datasets/Auto+MPG .

Table 3.3: Performance results of all methods on all data sets.

NRMSE					
	Prop.	**IANN**	**SBS-MLP**	**NMIFS**	**mRMR**
Friedman	4.68	5.01	4.68	4.68	4.68
Box-Jenkins	2.52	2.52	2.69	2.52	2.52
Gas mileage	17.61	18.27	17.52	18.35	18.59
WTP	7.98	8.70	7.99	8.43	8.39
Time [s]					
	Prop.	**IANN**	**SBS-MLP**	**NMIFS**	**mRMR**
Friedman	10.87	1.37	49.54	0.01	0.01
Box-Jenkins	19.7	2.3	4264.3	0.01	0.01
Gas mileage	7.1	1.2	905	0.02	0.03
WTP	2213.3	57.4	42029.1	12.68	10.57
Number of selected variables					
	Prop.	**IANN**	**SBS-MLP**	**NMIFS**	**mRMR**
Friedman	5	8	5	5	5
Box-Jenkins	3	7	3	7	8
Gas mileage	4	6	4	6	6
WTP	6	28	5	14	18

or eight variables. The proposed method has the same performance as the traditional SBS-MLP method, with approximately 200 times lower computational cost, and both methods achieve the best trade off between the lowest number of selected input features while maintaining good results in terms of NRMSE. Figure 3.2b shows that the proposed method attains the best performance value when the set of the top-ranked input variables has five variables. The best variables selected by the proposed method and by SBS-MLP were $\{y(i-1), x(i-3), y(i-2), x(i-6), x(i-2)\}$ and $\{y(i-1), x(i-3), y(i-2), y(i-3), x(i-2)\}$, respectively. As can be noticed, among the two sets of five selected variables there are four variables in common. This divergence can be explained because the SBS-MLP parameters are adjusted using the gradient descent algorithm, while the proposed method uses the model adjustement given by (3.16)-(3.23). Anyway, despite this difference on the selected variables, both results have equal prediction performance, while the proposed method executes the selection of variables faster than the SBS-MLP. On the other hand, it is interesting

to note that with a number of selected variables of 4, 3 or 2 the resulting NRMSE performance would not be much worse than with the five selected variables.

For the Gas mileage data set, the proposed method and the SBS-MLP achieved the best performance in terms of NRMSE, according with Table 3.3. However, the computation time of the SBS-MLP algorithm is more than 100 times greater when compared to the proposed method. Both methods have selected 4 variables, while the remaining methods were not able to select a strict subset from the six available variables. Despite the small number of input variables, the prediction accuracy can be improved by selecting only 4 variables as can be seen in Figure 3.2c. For the SBS-MLP and proposed methods the selected subsets are $\{x_4, x_6, x_2, x_5\}$. Thus, it can be concluded that the variables $\{x_4, x_6, x_2, x_5\}$ have enough representativeness for the prediction setting.

3.5.3 Water Treatment Plant Data Set

In the fourth experiment, the objective is to estimate the fluoride concentration in the effluent of a real-world urban water treatment plant (WTP). This data set comprises 1-year of acquisition. The value of fluoride in the effluent is measured by laboratory analysis once every day, and the objective of the methodology here proposed is to provide the fluoride concentration value at every 2 hours using a soft-sensor. The major concern about this problem is to know what are the best input variables and respective delays for the soft sensor. The dataset of plant variables that is available for learning consists of 11 input variables, $X = \{x_1, \ldots, x_{11}\}$ and 352 exemplars/samples. The variables correspond to physical values, such as pH, turbidity, color of the water and others. Appendix A presents further information on the WTP process and on the WTP data set used for the variable selection experiment.

The WTP is a long duration process, where the incoming water (called raw water) goes to the influent point, and it takes about 24 [h] to reach the effluent point which is the point of measurement of the fluoride. The sampling interval for the variables measured by sensors is 2 [h]. Thus, for the variables which are measured at the point of raw water influent (x_1, x_3, x_4, x_6, x_7, x_9, and x_{10}), the possible time-lags are considered within a range of 18-26 [h]. Let $n_x(j)$ be the set of possible time-lags, measured in time samples, for variable x_j ($j = 1, \ldots, 11$). Then,

$n_x(j) = \{9, 10, 11, 12, 13\}$, for $j = 1, 3, 4, 6, 7, 9, 10$. For those variables measured at the effluent point (x_2, x_5, x_8, and x_{11}), the possible time-lags are considered to be in a range of 0-8 [h]. Thus, $n_x(j) = \{0, 1, 2, 3, 4\}$, for $j = 2, 5, 8, 11$. Then, the size of the input set becomes equal to $|X| = 55$. The variable selection is going to be applied in the set X, to select the most relevant variables.

The five variable selection algorithms were applied and the results are presented in Table 3.3. The proposed method and the SBS-MLP algorithm have both similar results in terms of prediction performance as measured by the NRMSE on the test data set, but the proposed method has a much lower computational time. The worst variable selection algorithm, in terms of NRMSE, was the IANN, followed by the NMIFS and mRMR. Figure 3.2d shows that both the proposed method and SBS-MLP converge fast (in terms of the number of top-ranked variables) to the best solution, while the other methods require more variables to converge to a solution of similar prediction performance.

Moreover, with respect to the number of selected variables and also in the plot of error rates versus the number of top-ranked variables (Figure 3.2d), it is possible to note the slight difference between the results of the SBS-MLP and the proposed method, which contrasts with the results of the previous experiments. However, this divergence is plausible because of the different number of input variables in the experiments. The Box-Jenkins and Gas Mileage data sets have 10 and 6 input variables, respectively, and the WTP data set has 55 input variables. When working with a large number of input variables, as in the case of the WTP problem, the variable selected to be removed at each iteration can differ between the proposed method and the SBS-MLP. This happens because in the SBS-MLP all parameters of the MLP are readjusted following the gradient descent learning algorithm, while the proposed method follows another approach, which is the model adjustment by (3.16)-(3.23) and it is not possible to assure that both adjustments of the network (by the gradient descent learning algorithm and by (3.16)-(3.23)) will produce the same parameters.

The set of variables selected with the proposed algorithm, is composed by 6 variables: $S = \{x_5(i-3), x_3(i-9), x_7(i-12), x_9(i-12), x_1(i-12), x_9(i-10)\}$. The first two variables selected by the proposed algorithm, were the turbidity in the effluent and the turbidity in the coagulated water. These two variables are related with the quality of the process of cleaning the water during the treatment

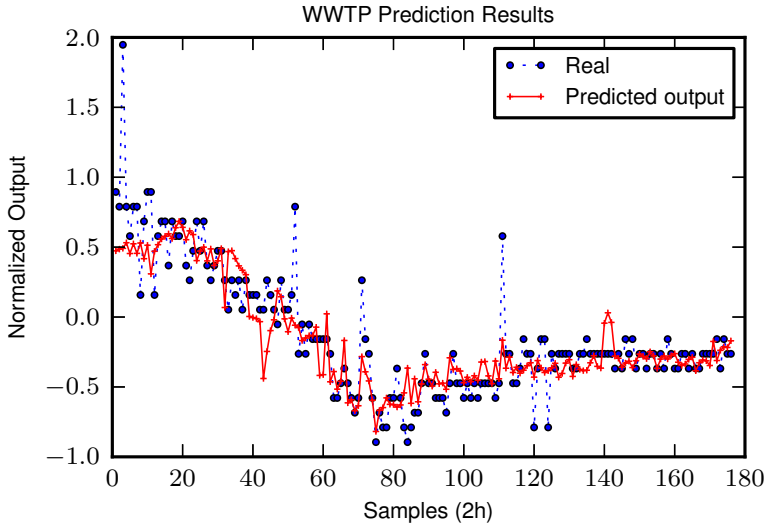


Figure 3.3: Fluoride prediction for the WTP test dataset. The sampling interval is 2 [h].

process (by measuring the difference between the turbidity in the effluent and in the coagulated water, it is possible to observe how effective was the cleaning process) and some portion of fluoride is retained during this process. The third and the fourth selected variables were the pH in the coagulated water and the Color in the raw water. Both variables, as well as the turbidity in the coagulated water (the first selected variable), are related to the Coagulation stage. In the Coagulation stage, the amount of coagulant added to the water is directly linked to the reduction of the concentration of fluoride in the water. This happens because during the Coagulation stage, a portion of fluoride sticks in the floc due to charge neutralization, and is then removed during the subsequent stages. An interesting fact is that the fifth selected variable is the amount Chloride added to the raw water which seems to contribute to the fluoride reduction during the cleaning process, since it is only possible to reduce the fluoride during the process (Appendix A). From the above discussion, it is seen that the selected variables (only five variables, or six variable-delay pairs, despite of the eleven available variables) correspond to a meaningful physical interpretation, while attaining good performance in the prediction of the target variable of interest. The prediction results are shown in Figure 3.3.

3.6 Conclusion

In this chapter a new variable selection algorithm based on the MLP model was presented and compared with four state-of-the-art methods, a wrapper method (SBS-MLP), an embedded method (IANN), and two filter methods (mRMR and NMIFS) methods. All the data sets used in the experiments have a small number of samples, which is a common characteristic in soft sensors applications, and the number of input variables ranged from 5 to 55. In a series of four experiments, the proposed variable selection method has been shown to be feasible and effective. It has been shown that the proposed method has similar performance when compared to the traditional SBS-MLP based on the MLP error algorithm, and has the advantage of having much lower computation cost. The proposed method presents similar or better approximation performance when compared to the other four methods. In the experiments, among all the five methods, the proposed method selects the lowest, or nearly the lowest, number of variables to achieve the best solution. In soft sensors applications, having a lower number of input variables is a positive factor for decreasing implementation costs (e.g. lower numbers of hardware sensors and/or laboratory analysis), or even making the soft sensor feasible at all.

It is necessary to point out that the proposed methodology is dependent on the information content on the dataset. Thus, when applying it, it is necessary to assure that the data set is as representative as possible. The reliability of the method is increased when the number and representativeness of the available samples increase.

Chapter 4

Learning Soft Sensors in Multiple Operating Modes with a Mixture of PLS Experts

Contents

In almost all soft sensor applications, a single model is tuned using all available training samples, without distinguishing the operating modes of the process. However, the existence of multiple operating modes in a process is an inherent characteristic of most industrial applications. Sometimes multiple operating modes result from external disturbances, as for example a change in feedstock or product grade or even changes such as the diurnal load variation of a power plant or the summer-winter operation of a refinery [Matzopoulos, 2010; Wang *et al.*, 2012]. In these situations, consistently training a model for each operating mode or for each set of correlated operating modes of the process has shown to be reasonably consistent and to be beneficial for the prediction accuracy [Facco *et al.*, 2009; Yu, 2012]; During online operation, when a new input sample is made available, the model which is the most adequate for this new sample is identified and then used to make the prediction. The identification of which model will be used is a key issue in the development [Facco *et al.*, 2009; Camacho and Picó, 2006; Lu and Gao, 2005], which can be done using expert knowledge [Facco *et al.*, 2009] or using automatic tools, such as the finite mixture of Gaussian models (FMGM) [Yu, 2012].

In this context, in [Facco *et al.*, 2009] the authors work on modeling the operating modes in a polymerization batch process case study. The correlated operating modes have been grouped, and then a separate PLS model is tuned for each set of correlated operating modes. During online operation, the incoming sample is assigned to the corresponding mode and its model is used for the prediction. However, in [Facco *et al.*, 2009] the expert knowledge of operators has been used to determine the operating modes and in some cases or problems such information can be not available.

Another approach, based on the FMGM, was proposed in [Yu, 2012]. In this work, the FMGM is used to automatically identify the different operating modes of the process. Then, multiple localized Gaussian process regression models in the nonlinear kernel space were built to characterize the different dynamic relationships between process variables (inputs to the prediction setting) and quality variables (outputs of the prediction setting) within the identified operating modes. During

online operation, the incoming sample is assigned automatically to the corresponding submodel, using the FMGM. The major drawback of [Yu, 2012] is that the determination of the operating modes and model tuning is done separately, i.e. the set of operating modes is determined independently of the model used. However, as verified in the case of study of [Facco *et al.*, 2009], a model can be set for more than one operating mode, with the advantage of reducing the number of necessary models and increasing the number of samples available for tuning each model. Another drawback of [Yu, 2012] is that the number of samples used for tuning each model is constrained by the number of samples of each operating mode, which is defined by the FMGM. The approach of [Yu, 2012] leads to "hard" partition boundaries, and consequently just a part of the total of samples is used for tuning the prediction model of each operating mode. Such an approach can lead to poor modeling on the corresponding operating mode, depending on the chosen model and the available samples.

In this chapter a method for dealing with online prediction of critical variables in processes with multiple operating modes is proposed and derived. The method is called mixture of partial least squares (PLS) experts (Mix-PLS). The Mix-PLS is going to be derived based on the mixture of experts (ME) framework [Jacobs *et al.*, 1991]. The ME models input-output observations by assuming that they have been produced by a set of different random sources (the random sources can be thought as operating modes). Each random source in the ME framework is modeled by an expert, and during the online operation the decision about which experts should be used is modeled by a gating function. Figure 4.1 illustrates this approach. The learning of parameters in the ME is done using the maximum likelihood method and the expectation and maximization (EM) algorithm [Dempster *et al.*, 1977]. By modeling the experts by a Gaussian linear regression and the gating functions as a softmax function, the ME is then reduced to a mixture of linear regression experts (MLRE) [Jacobs *et al.*, 1991; Jordan, 1994]. However, the standard MLRE cannot handle input collinearity, and this solution is more prone to overfitting with respect to the number of experts used as will be demonstrated experimentally later [Yuksel *et al.*, 2012]. In this thesis, the parameters of each expert, and of each gating function, are determined using the PLS algorithm. The solution of the parameters using the PLS algorithm overcomes the problem of collinearity of input data and also makes the Mix-PLS less prone to overfitting with respect to the number of

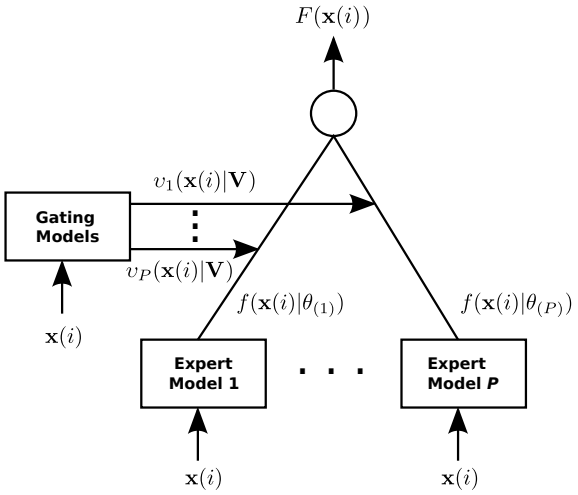


Figure 4.1: Mixture of linear regression models with P experts, where $\mathbf{x}(i)$ is an input sample, $\upsilon_p(\mathbf{x}(i), \mathbf{V})$ is the output of gating function for model p and $f(\mathbf{x}(i), \boldsymbol{\theta}_p)$ is the output of the linear model of expert p.

mixture models. To the best of the author's knowledge, there is no reference in the literature for solving the MLRE using PLS. See [Yuksel *et al.*, 2012] for a recent complete survey about mixture of experts.

In the rest of the chapter, during the derivation of Mix-PLS, some important notions about both the PLS algorithm and its respective parameters selection method, and the mixture of experts, will also be introduced. The notation used throughout this chapter is the one defined in Section 3.1. After the derivation of the Mix-PLS, in the experimental part, the Mix-PLS is then applied to three real-world prediction problems. Moreover, the proposed Mix-PLS is compared with the state of the art methods of soft sensors: a single PLS model, a single layer MLP neural network (SLNN) trained using the gradient descent training algorithm, and a least squares support vector regression (LS-SVR) model with Gaussian kernel [Suykens *et al.*, 2002]. The experimental results indicate that the recursive Mix-PLS outperforms the other methods. Moreover, the Mix-PLS has the advantage of being more interpretable than the nonlinear models with respect to the parameters.

4.1 Partial Least Squares

PLS regression is a method for finding the parameters $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_D]^T$ of a linear model of the form $\hat{y}(i) = f(\mathbf{x}(i); \boldsymbol{\theta}) = \theta_0 + \sum_{l=1}^{D} \theta_l x_l(i)$ from a given a set of input-output samples Φ, where D is the dimensionality of the input space. This model is composed of a linear combination of the inputs to the regression. The objective of the design of the linear combination is to maximize the covariance between the input and output spaces. The PLS estimation method is attractive because it works well on high dimensional data, noisy data, and data with collinearity, which are common characteristics in most industrial applications.

More specifically, PLS projects the information of the data into a low dimensional space defined by a smaller number of orthogonal latent vectors $\mathbf{t}_m$ and $\mathbf{u}_m$, with $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_M] \in \mathbb{R}^{k \times M}$ (where $M \leq D$ is the number of latent variables), $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_M] \in \mathbb{R}^{k \times M}$, where k is the number of data samples:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E} = \sum_{m=1}^{M} \mathbf{t}_m \mathbf{p}_m^T + \mathbf{E}, \tag{4.1}$$

$$\mathbf{y} = \mathbf{T}\mathbf{B}\mathbf{Q}^T + \mathbf{F} = \sum_{m=1}^{M} \mathbf{u}_m \mathbf{q}_m^T + \mathbf{F}, \tag{4.2}$$

where $\mathbf{U} = \mathbf{T}\mathbf{B}$, $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_M] \in \mathbb{R}^{D \times M}$ and $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_M] \in \mathbb{R}^{1 \times M}$ are the loading matrices, $\mathbf{E}$ and $\mathbf{F}$ are the input and output data residuals, $\mathbf{B} = \text{diag}(b_1, \ldots, b_M)$ is a diagonal matrix with the regression weights b_m. Then, the estimated output $\hat{y}$, given an input sample $\mathbf{x}(i)$, is given by:

$$\hat{y}(i) = \mathbf{x}^T(i)\boldsymbol{\theta}, \tag{4.3}$$

where $\boldsymbol{\theta} = \mathbf{P}^{\dagger}\mathbf{B}\mathbf{Q}^T$, and $\mathbf{P}^{\dagger} = (\mathbf{P}\mathbf{P}^T)^{-1}\mathbf{P}$ is the pseudo-inverse of $\mathbf{P}$ [Ben-Israel and Greville, 2003]. The values of $\mathbf{B}$, $\mathbf{T}$, $\mathbf{P}$, $\mathbf{U}$, $\mathbf{Q}$ from the above problem can be computed by using the classical Nonlinear Iterative Partial Least Squares (NIPLS or NIPALS) method [Wold, 1975].

4.1.1 Selecting the Number of Latent Variables

Let $\mathcal{M}$ be such that $M \in \mathcal{M}$, for any possible/eligible number of latent variables, M. The major concern regarding the PLS algorithm is to select the number of

latent variables M. Usually M is determined by a K-fold cross-validation procedure applied on the training data set [Mevik and Cederkvist, 2004; Hawkins, 2004; Toher *et al.*, 2007]. However, the K-fold cross-validation procedure is very efficient as long as k (the number of samples) is not too large, since it needs to run the PLS algorithm $K|\mathcal{M}|$ times. A fast way of selecting the number of latent variables is using information criterion methods, such as the Akaike Information Criterion (AIC) [Akaike, 1974] or the Bayesian Information Criterion (BIC) [Schwarz, 1978], which measure the quality of a model in terms of its accuracy-complexity trade-off (ACT). Using information criterion methods, the PLS algorithm runs just $|\mathcal{M}|$ times [Li *et al.*, 2002].

However, the major concern when applying information criterion methods to evaluate the ACT in the PLS algorithm is to determine the number of its degrees of freedom (DOF) (number of free parameters) of the PLS. Usually the DOF in the PLS model is set to be equal to the number of latent variables, but this is a wrong assumption and does not lead to satisfactory results in the selection of the number of latent variables [Kramer and Braun, 2007; Kramer and Sugiyama, 2011]. This problem of determining the DOF in a PLS model was addressed in [Kramer and Sugiyama, 2011], where an unbiased estimate of the DOF has been proposed. The use of 10-fold cross validation (using the RSS measure), AIC, and BIC criteria (both with the proposed DOF estimate) to select the number of latent variables has been compared. It has been concluded that BIC and 10-fold cross validation provide the best results, with similar performance for both, and with much lower computational cost associated with the BIC computations.

Thus, in this thesis, the BIC criterion will be used to select the number of latent vectors for the PLS algorithm, for each expert and each gate of the Mix-PLS (the proposed implementation will be detailed in Section 4.2). Assume that variable y has an approximation uncertainty modeled by a Gaussian pdf $\mathcal{N}(y(i)|f(\mathbf{x}(i),\boldsymbol{\theta}),\sigma^2)$, where $f(\mathbf{x}(i),\boldsymbol{\theta})$ is the mean, and σ^2 is the variance. For a linear model $f(\mathbf{x}(i),\boldsymbol{\theta}) = \mathbf{x}^T(i)\boldsymbol{\theta}$, where $\boldsymbol{\theta}$ is determined using the PLS method with $m \in \mathcal{M}$ latent vectors, the BIC of the PLS model for the data set $\{\mathbf{X},\mathbf{y}\}$ is equal to:

$$\mathrm{BIC}(m) = -2\ln\prod_{i=1}^{k}\mathcal{N}(y(i)|f(\mathbf{x}(i),\boldsymbol{\theta}),\sigma^2) + \mathrm{dof}(m,\mathbf{X},\mathbf{y},\mathbf{T})\ln(k), \tag{4.4}$$

where the quantity $\ln\prod_{i=1}^{k}\mathcal{N}(y(i)|f(\mathbf{x}(i),\boldsymbol{\theta}),\sigma^2)$ is the log likelihood which accounts

for the model accuracy, and the second term, dof$(m, \mathbf{X}, \mathbf{y}, \mathbf{T})$, is the number of DOF of the PLS regressor, which relates to model complexity (see [Kramer and Sugiyama, 2011] for implementation details of dof$(\cdot)$).

4.2 Mixture of PLS Experts

In this section, the formulas for learning of the Mix-PLS model are going to be derived. For learning, the parameters of the Mix-PLS are tuned using the set of observations Φ. This section also discusses the determination of the number of experts to be used, and gives experimental evidence towards demonstrating that the Mix-PLS is less prone to overffiting when compared with the traditional solution of the ME, the MLRE.

4.2.1 Motivation on Using Mixture of PLS Experts

The motivation for developing the Mix-PLS lies on the fact that there exist few methods adressing the problem of modeling in scenarios with multiple operating modes. Many industrial processes have multiple operating modes, but the majority of soft sensor applications do not take this fact into consideration. To integrate the multiple operating modes into the learning process, the Mix-PLS conjugates two methodologies existing in literature, namely the PLS algorithm which is a popular method in soft sensor applications, and the ME framework, a well know method from the machine learning literature. The proposed Mix-PLS method actually enhances the possible utilization of PLS models in industrial processes.

4.2.2 Mixture of Experts

The ME approximates the true pdf $p(y(i)|\mathbf{x}(i))$ with the following superposition of individual pdfs:

$$p(y(i)|\mathbf{x}(i), \boldsymbol{\vartheta}) = \sum_{p=1}^{P} \upsilon_p \left(\mathbf{x}(i), \mathbf{V}\right) \, p\left(y(i)|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega}\right), \tag{4.5}$$

where P is the number of experts, $\boldsymbol{\vartheta} = \{\mathbf{V}, \boldsymbol{\mathcal{E}}\}$, $\mathbf{V}$ and $\boldsymbol{\mathcal{E}} = \{\boldsymbol{\Theta}, \boldsymbol{\Omega}\}$ are defined as the sets of parameters of the gates and model experts, respectively, $\boldsymbol{\Theta} = \{\, \boldsymbol{\theta}_p \,|\, p =$

$1, \ldots, P\}$, and $\upsilon_p(\mathbf{x}(i), \mathbf{V})$ is the gating function of expert p, which satisfies $0 \leq \upsilon_p(\mathbf{x}(i), \mathbf{V}) \leq 1$ for $p = 1, \ldots, P$, and $\sum_{p=1}^{P} \upsilon_p(\mathbf{x}(i), \mathbf{V}) = 1$. $p(y(i)|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega})$ is the pdf of expert model p, with mean $f_p(\mathbf{x}(i), \boldsymbol{\theta}_p)$ and additional pdf parameters $\boldsymbol{\Omega}$. From (4.5), the prediction equation of the ME is obtained as the following conditional mean of y given $\mathbf{x}$:

$$\begin{aligned} F(\mathbf{x}(i)) &= \mathbb{E}[y|\mathbf{x}(i)] = \mathcal{F}(\mathbf{x}(i), \boldsymbol{\vartheta}) \\ &= \int y\, p\left(y|\mathbf{x}(i), \boldsymbol{\vartheta}\right) \mathrm{d}y \\ &= \int y \sum_{p=1}^{P} \upsilon_p\left(\mathbf{x}(i), \mathbf{V}\right)\, p\left(y|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega}\right) \mathrm{d}y \\ &= \sum_{p=1}^{P} \upsilon_p\left(\mathbf{x}(i), \mathbf{V}\right)\, f_p(\mathbf{x}(i), \boldsymbol{\theta}_p). \end{aligned} \tag{4.6}$$

$\mathcal{F}(\mathbf{x}(i), \boldsymbol{\vartheta})$ is the function which minimizes the expected squared loss $\mathcal{L} = \int\int \left(\mathcal{F}(\mathbf{x}(i), \boldsymbol{\vartheta}) - y\right)^2 p(\mathbf{x}, y) d\mathbf{x} dy$ [Bishop, 2006].

In the ME the log likelihood of (4.5), given a set of observations Φ is given by [Jacobs *et al.*, 1991]:

$$\begin{aligned} \ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\vartheta}) &= \ln \left(\sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \mathbf{V})\, p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\mathcal{E}}) \right) \\ &= \ln \left(\prod_{i=1}^{k} p(y(i)|\mathbf{x}(i), \boldsymbol{\vartheta}) \right) \\ &= \ln \left(\prod_{i=1}^{k} \sum_{\mathbf{z}(i)} p(\mathbf{z}(i)|\mathbf{x}(i), \mathbf{V})\, p(y(i)|\mathbf{x}(i), \mathbf{z}(i), \boldsymbol{\mathcal{E}}) \right), \end{aligned} \tag{4.7}$$

where $\mathbf{Z} = [z_{ip}] = [z_p(i)] \in \mathbb{R}^{k \times P}$ denotes a set of the hidden variables, and $\mathbf{z}(i) = [z_1(i), \ldots, z_P(i)]^T \in \mathbb{R}^P$ is the vector of hidden variables for a sample i, where $z_p(i) \in \{0, 1\}$, for $p = 1, \ldots, P$, and for each sample $\mathbf{z}(i)$, all variables $z_p(i)$ are zero, except for a single value of $z_p(i) = 1$, for some p. The hidden variable $z_p(i)$ indicates which expert p was responsible for generating the data point i. The distributions

$p(\mathbf{z}(i)|\mathbf{x}(i), \mathbf{V})$ and $p(y(i)|\mathbf{x}(i), \mathbf{z}(i), \boldsymbol{\mathcal{E}})$ are defined as follows [Bishop, 2006]:

$$\begin{aligned} p(\mathbf{z}(i)|\mathbf{x}(i), \boldsymbol{\vartheta}) &= p(\mathbf{z}(i)|\mathbf{x}(i), \mathbf{V}) \\ &= \prod_{p=1}^{P} \left[p(z_p(i)|\mathbf{x}(i), \mathbf{V})\right]^{z_p(i)} = p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}\right), \end{aligned} \tag{4.8}$$

$$\begin{aligned} p(y(i)|\mathbf{x}(i), \mathbf{z}(i), \boldsymbol{\vartheta}) &= p(y(i)|\mathbf{x}(i), \mathbf{z}(i), \boldsymbol{\mathcal{E}}) \\ &= \prod_{p=1}^{P} \left[p(y(i))|\mathbf{x}(i), z_p(i), \boldsymbol{\mathcal{E}})\right]^{z_p(i)} = p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right). \end{aligned} \tag{4.9}$$

Then, from (4.7)-(4.9):

$$\begin{aligned} \ln p(\mathbf{y}|\mathbf{X}, \boldsymbol{\vartheta}) &= \ln \left(\prod_{i=1}^{k} \sum_{\mathbf{z}(i)} p(\mathbf{z}(i)|\mathbf{x}(i), \mathbf{V})\, p(y(i)|\mathbf{x}(i), \mathbf{z}(i), \boldsymbol{\mathcal{E}}) \right), \\ &= \ln \left(\prod_{i=1}^{k} \sum_{\mathbf{z}(i)} p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}\right)\, p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \right), \\ &= \sum_{i=1}^{k} \ln \left(\sum_{p=1}^{P} p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}\right)\, p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \right). \end{aligned} \tag{4.10}$$

The maximization of (4.10) is not straightforward [Bishop, 2006; Jacobs *et al.*, 1991]. The common way to maximize (4.10) is by means of the Expectation-Maximization (EM) algorithm. The EM algorithm is a general method for finding the maximum-likelihood estimate of the parameters of an underlying distribution from a given data set when the data has hidden variables [Dempster *et al.*, 1977; Bishop, 2006]. The learning of the mixture of experts by the EM algorithm is summarized in Algorithm 4.1.

During the Expectation step (E step) of the EM, the current parameter values $\boldsymbol{\vartheta}^{\text{old}}$ are used to estimate the posterior distribution of hidden variables $p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\text{old}})$. Then, in the Maximization step (M step), this posterior distribution is used to find the new parameter values $\boldsymbol{\vartheta}^{\text{new}}$, which maximize the expectation of the complete-data (output and hidden variables) log likelihood

$$\begin{aligned} Q_{\text{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}}) &= \mathbb{E}_{\mathbf{Z}}[\ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})] \\ &= \sum_{\mathbf{Z}} \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})\, p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\text{old}}). \end{aligned} \tag{4.11}$$

Algorithm 4.1 EM algorithm for ME

1: **Input:** Φ;
2: **Output:** $\boldsymbol{\vartheta}$;
3: **Initialization:** Initialize $\boldsymbol{\vartheta}$ equal to some initial $\boldsymbol{\vartheta}^{\text{old}}$;
4: **repeat**
5: // E step:
6: Estimate the distribution $p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\text{old}})$ using (4.13);
7: // M step:
8: Find the new parameter values $\boldsymbol{\vartheta}^{\text{new}}$, which maximize the expectation of the complete-data log likelihood $Q_{\text{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$:
$\boldsymbol{\vartheta}^{\text{new}} = \arg\max_{\boldsymbol{\vartheta}} Q_{\text{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}}) = \arg\max_{\boldsymbol{\vartheta}} \left(\sum_{\mathbf{Z}} \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}) p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\text{old}}) \right)$ (Equation (4.19));
$\boldsymbol{\vartheta}^{\text{old}} \leftarrow \boldsymbol{\vartheta}^{\text{new}}$;
9: **until** convergence is attained;
10: **return** $\boldsymbol{\vartheta}^{\text{new}}$.

The convergence of the EM algorithm can be verified by analyzing the convergence of the expectation $Q_{\text{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$. It is also possible to set a pre-specified maximum number of iterations.

To perform the E step, the Bayes theorem and equations (4.7)-(4.9) are used to calculate the posterior distribution of the hidden variables, $p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta})$, as follows:

$$p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\vartheta}) p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})}{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\vartheta})}, \tag{4.12}$$

$$= \prod_{i=1}^{k} \prod_{p=1}^{P} \left(\frac{p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) p\left(z_p(i)|\mathbf{x}(i), \mathbf{V}\right)}{\sum_{p=1}^{P} \left[p\left(z_p(i)|\mathbf{x}(i), \mathbf{V}\right) p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \right]} \right)^{z_p(i)}. \tag{4.13}$$

For the M step, the value of $p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})$, necessary to compute $Q_{\text{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$ (4.11) is obtained using (4.8)-(4.9) as follows:

$$p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}) = p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\vartheta})\, p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}), \tag{4.14}$$

$$= \prod_{i=1}^{k} \prod_{p=1}^{P} \left[p\left(z_p(i)|\mathbf{x}(i), \mathbf{V}\right) p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \right]^{z_p(i)}. \tag{4.15}$$

The expectation of the complete-data log likelihood (4.11) can be computed

using (4.13) and (4.15). First, taking the logarithm of $p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})$ (4.15):

$$\ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}) = \sum_{i=1}^{k} \sum_{p=1}^{P} \Bigg(z_p(i) \Big[\ln p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}\right) + \ln p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \Big] \Bigg), \tag{4.16}$$

and then computing the expectation of $\ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})$ with respect to the posterior distribution of hidden variables $\mathbf{Z}$:

$$\begin{aligned} Q_{\mathrm{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}}) &= \sum_{\mathbf{Z}} \ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})\, p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\mathrm{old}}), \\ &= \sum_{i=1}^{k} \sum_{p=1}^{P} \gamma_p^{\mathrm{old}}(i) \ln p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}\right) \\ &\quad + \sum_{i=1}^{k} \sum_{p=1}^{P} \gamma_p^{\mathrm{old}}(i) \ln p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \\ &= Q_g(\mathbf{V}, \boldsymbol{\vartheta}^{\mathrm{old}}) + Q_e(\boldsymbol{\mathcal{E}}, \boldsymbol{\vartheta}^{\mathrm{old}}), \end{aligned} \tag{4.17}$$

where $\gamma_p^{\mathrm{old}}(i)$, defined as the responsibility of model p, is the expectation of $z_p(i)$ with respect to its distribution (4.13), and it accounts for the probability of model p generating the data sample i:

$$\gamma_p^{\mathrm{old}}(i) = \frac{p\left(z_p(i) = 1|\mathbf{x}(i), \mathbf{V}^{\mathrm{old}}\right)\, p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}^{\mathrm{old}}\right)}{\sum_{l=1}^{P} \left[p\left(z_l(i) = 1|\mathbf{x}(i), \mathbf{V}^{\mathrm{old}}\right)\, p\left(y(i)|z_l(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}^{\mathrm{old}}\right)\right]}. \tag{4.18}$$

In (4.17), Q_g and Q_e are the contributions of the gate and expert parameters for the expectation of complete-data log likelihood. Then, the M step of the EM algorithm can be performed, by separately maximizing the gate and expert contributions, as follows:

$$\begin{aligned} \boldsymbol{\vartheta}^{\mathrm{new}} &= \arg\max_{\boldsymbol{\vartheta}} Q_{\mathrm{ME}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}}), \\ &= \left\{ \arg\max_{\mathbf{V}} Q_g(\mathbf{V}, \boldsymbol{\vartheta}^{\mathrm{old}}), \arg\max_{\boldsymbol{\mathcal{E}}} Q_e(\boldsymbol{\mathcal{E}}, \boldsymbol{\vartheta}^{\mathrm{old}}) \right\}. \end{aligned} \tag{4.19}$$

Thus, the determination of the parameters for the gates $\mathbf{V}$ and the experts $\boldsymbol{\mathcal{E}}$ is independently performed by the maximizations in (4.19). In the Mix-PLS, such maximizations are done using the PLS algorithm, as derived in Subsections 4.2.3 and 4.2.4 below.

4.2.3 Modeling the Experts with the PLS Algorithm

In this thesis, each pdf $p\left(y(i)|z_p(i)=1,\mathbf{x}(i),\boldsymbol{\mathcal{E}}\right)$ in $Q_e(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}})$ (4.17) is described by a Gaussian distribution $\mathcal{N}(y(i)|f_p(\mathbf{x}(i),\boldsymbol{\theta}_p),\omega_p)$, where $f_p(\mathbf{x}(i),\boldsymbol{\theta}_p)$, and ω_p are the mean and variance of the model of expert p, respectively. The mean is modeled by a linear model $f_p(\mathbf{x}(i),\boldsymbol{\theta}_p)=\mathbf{x}(i)^T\boldsymbol{\theta}_p$. Specifically, the expert parameters $\boldsymbol{\mathcal{E}}=\{\boldsymbol{\Theta},\boldsymbol{\Omega}\}$, include the parameters of $\boldsymbol{\Theta}=\{\boldsymbol{\theta}_p|\,p=1,\ldots,P\}$, and $\boldsymbol{\Omega}=\{\omega_p|\,p=1,\ldots,P\}$. Thus, the contribution $Q_e\left(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}}\right)$ of all experts to the expectation of complete data log likelihood (4.17) can be rewritten as:

$$Q_e\left(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}}\right)=\sum_{p=1}^{P}Q_{e,p}\left(\{\boldsymbol{\theta}_p,\omega_p\},\boldsymbol{\vartheta}^{\text{old}}\right), \tag{4.20}$$

$$Q_{e,p}\left(\{\boldsymbol{\theta}_p,\omega_p\},\boldsymbol{\vartheta}^{\text{old}}\right)=\sum_{i=1}^{k}\gamma_p^{\text{old}}(i)\ln\mathcal{N}(y(i)|f_p(\mathbf{x}(i),\boldsymbol{\theta}_p),\omega_p), \tag{4.21}$$

where $Q_{e,p}\left(\{\boldsymbol{\theta}_p,\omega_p\},\boldsymbol{\vartheta}^{\text{old}}\right)$ is the contribution of expert p, and from (4.18) the responsibility $\gamma_p^{\text{old}}(i)$ is equal to:

$$\gamma_p^{\text{old}}(i)=\frac{\upsilon_p^{\text{old}}(i)\mathcal{N}(y(i)|f_p(\mathbf{x}(i),\boldsymbol{\theta}_p^{\text{old}}),\omega_p^{\text{old}})}{\sum_{l=1}^{P}\upsilon_l^{\text{old}}(i)\mathcal{N}(y(i)|f_l(\mathbf{x}(i),\boldsymbol{\theta}_l^{\text{old}}),\omega_l^{\text{old}})}, \tag{4.22}$$

where $\upsilon_p^{\text{old}}(i)=p\left(z_p(i)=1|\mathbf{x}(i),\mathbf{V}^{\text{old}}\right)$ is the probability of model p generating sample i, which is going to be determined in Section 4.2.4.

Then, $Q_e\left(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}}\right)$ is maximized with respect to $\boldsymbol{\mathcal{E}}$ by solving equations $\frac{\partial Q_e\left(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}}\right)}{\partial\boldsymbol{\theta}_p}=0$, and $\frac{\partial Q_e\left(\boldsymbol{\mathcal{E}},\boldsymbol{\vartheta}^{\text{old}}\right)}{\partial\omega_p}=0$, which gives the following solution:

$$\boldsymbol{\theta}_p^{\text{new}}=\left(\mathbf{X}^T\boldsymbol{\Gamma}_p\mathbf{X}\right)^{-1}\mathbf{X}^T\boldsymbol{\Gamma}_p\mathbf{y}, \tag{4.23}$$

$$\begin{aligned}\omega_p^{\text{new}}&=\frac{\sum_{i=1}^{k}\gamma_p^{\text{old}}(i)\left(y(i)-f_p\left(\mathbf{x}(i),\boldsymbol{\theta}_p^{\text{new}}\right)\right)^2}{\sum_{i=1}^{k}\gamma_p^{\text{old}}(i)}\\&=\frac{\left|\left|\mathbf{y}_{(\boldsymbol{\Gamma},p)}-\mathbf{X}_{(\boldsymbol{\Gamma},p)}\boldsymbol{\theta}_p^{\text{new}}\right|\right|^2}{\text{Tr}(\boldsymbol{\Gamma}_p)},\end{aligned} \tag{4.24}$$

where $\boldsymbol{\Gamma}_p=\text{diag}\left(\gamma_p^{\text{old}}(1),\gamma_p^{\text{old}}(2),\ldots,\gamma_p^{\text{old}}(k)\right)$ is a diagonal matrix, and $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$ and $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$ are defined below in (4.25)-(4.26). As can be noticed, the maximization of Q_e (4.20) is equivalent to a weighted least squares problem, where the responsibility $\gamma_p^{\text{old}}(i)$ is the importance of each sample.

In this work, the vector of parameters of each model $\boldsymbol{\theta}_p^{\text{new}}$ (4.23) is going to be solved using the PLS algorithm. In the PLS algorithm, from (4.1)-(4.2), the inputs $\mathbf{X}$ and output $\mathbf{y}$ are traditionally represented through their approximation with M latent and loading variables, i.e. $\mathbf{X} \approx \mathbf{T}\mathbf{P}^T$ and $\mathbf{y} \approx \mathbf{T}\mathbf{B}\mathbf{Q}^T$. However, solving (4.23) after replacing these approximations is not straightforward. A simpler approach is to multiply both $\mathbf{X}$ and $\mathbf{y}$ by $\sqrt{\boldsymbol{\Gamma}_p}$, so that the weighted representation of $\mathbf{X}$ and $\mathbf{y}$ becomes equal to:

$$\mathbf{X}_{(\boldsymbol{\Gamma},p)} = \sqrt{\boldsymbol{\Gamma}_p}\mathbf{X} \approx \mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{P}_{(\boldsymbol{\Gamma},p)}^T, \tag{4.25}$$

$$\mathbf{y}_{(\boldsymbol{\Gamma},p)} = \sqrt{\boldsymbol{\Gamma}_p}\mathbf{y} \approx \mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{B}_{(\boldsymbol{\Gamma},p)}\mathbf{Q}_{(\boldsymbol{\Gamma},p)}^T, \tag{4.26}$$

where $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$ and $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$ are the weighted inputs and output matrices of model p with weight matrix $\boldsymbol{\Gamma}_p$. $\mathbf{T}_{(\boldsymbol{\Gamma},p)}$ and $\mathbf{P}_{(\boldsymbol{\Gamma},p)}$ are the PLS latent and loading matrices of the weighted input $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$, and $\mathbf{B}_{(\boldsymbol{\Gamma},p)}$ and $\mathbf{Q}_{(\boldsymbol{\Gamma},p)}^T$ are the PLS latent and loading matrices of the weighted output $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$. It is assumed that the weighted input and output decomposition for expert p through the PLS algorithm is made with Me_p latent variables.

Then, by replacing (4.25) and (4.26) into (4.23), the parameters of model p can be written as:

$$\begin{aligned}
\boldsymbol{\theta}_p^{\text{new}} &= \left(\mathbf{X}_{(\boldsymbol{\Gamma},p)}^T\mathbf{X}_{(\boldsymbol{\Gamma},p)}\right)^{-1}\mathbf{X}_{(\boldsymbol{\Gamma},p)}^T\mathbf{y}_{(\boldsymbol{\Gamma},p)}, \\
&= \left(\left(\mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{P}_{(\boldsymbol{\Gamma},p)}^T\right)^T\left(\mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{P}_{(\boldsymbol{\Gamma},p)}^T\right)\right)^{-1}\left(\mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{P}_{(\boldsymbol{\Gamma},p)}^T\right)^T\mathbf{T}_{(\boldsymbol{\Gamma},p)}\mathbf{B}_{(\boldsymbol{\Gamma},p)}\mathbf{Q}_{(\boldsymbol{\Gamma},p)}^T, \\
&= \left(\mathbf{P}_{(\boldsymbol{\Gamma},p)}\mathbf{P}_{(\boldsymbol{\Gamma},p)}^T\right)^{-1}\mathbf{P}_{(\boldsymbol{\Gamma},p)}\mathbf{B}_{(\boldsymbol{\Gamma},p)}\mathbf{Q}_{(\boldsymbol{\Gamma},p)}^T.
\end{aligned} \tag{4.27}$$

At each new iteration of the EM algorithm, the values of responsibility $\gamma_p^{\text{old}}(i)$ computed in the expectation step change. Consequently the values of weighted input matrix $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$ and output vector $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$ change. Then, the number of latent variables Me_p necessary to represent $\mathbf{X}_{(\boldsymbol{\Gamma},p)}$ and $\mathbf{y}_{(\boldsymbol{\Gamma},p)}$ should be recomputed for a proper representation.

As discussed before, the use of K-fold cross validation to determine Me_p would computationally overload the EM algorithm, since at each new iteration the cross validation would need to be run $K|\mathcal{M}|$ times. Thus, at each new iteration, the number of latent variables is going to be determined using the BIC measure (4.4), which needs to run just $|\mathcal{M}|$ times. Since each sample $y(i)$ has a weight $\gamma_p^{\text{old}}(i)$, then the weighted log-likelihood (WLL, $\ln \mathcal{L}_w$) [Newton and Raftery, 1994] is going to be used

instead of the log-likelihood in the first term of the r.h.s. of (4.4). Thus, to compute the BIC for expert p, it is necessary to determine the WLL of its approximation model. From the definition of weighted likelihood [Newton and Raftery, 1994], the WLL of a PLS model with sample weights $\gamma_p^{\text{old}}(i)$, is equal to:

$$\begin{aligned}\ln \mathcal{L}_w &= \ln \prod_{i=1}^{k} \mathcal{N}\left(y(i) \mid f_p\left(\mathbf{x}(i), \boldsymbol{\theta}_p\right), \omega_p\right)^{\gamma_p^{(\text{old})}(i)} \\ &= \sum_{i=1}^{k} \gamma_p^{\text{old}}(i) \ln \mathcal{N}\left(y(i) \mid f_p\left(\mathbf{x}(i), \boldsymbol{\theta}_p\right), \omega_p\right), \end{aligned} \tag{4.28}$$

and it is equal to $Q_{e,p}\left(\{\boldsymbol{\theta}_p, \omega_p\}, \boldsymbol{\vartheta}^{\text{old}}\right)$ in (4.21). Then, the BIC when using m latent variables for expert p is:

$$\begin{aligned}\text{BIC}_{\text{E}}(p,m) &= -2Q_{e,p}\left(\{\boldsymbol{\theta}_p, \omega_p\}, \boldsymbol{\vartheta}^{\text{old}}\right) \\ &\quad + \text{dof}\left(m, \sqrt{\boldsymbol{\Gamma}_p}\mathbf{X}, \sqrt{\boldsymbol{\Gamma}_p}\mathbf{y}, \mathbf{T}_{(\boldsymbol{\Gamma},p)}\right) \ln(k), \\ &= -2\sum_{i=1}^{k} \gamma_p^{\text{old}}(i) \ln \mathcal{N}\left(y(i) | f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \omega_p\right) \\ &\quad + \text{dof}\left(m, \mathbf{X}_{(\boldsymbol{\Gamma},p)}, \mathbf{y}_{(\boldsymbol{\Gamma},p)}, \mathbf{T}_{(\boldsymbol{\Gamma},p)}\right) \ln(k), \\ &= \sum_{i=1}^{k} \gamma_p^{\text{old}}(i) \left(\ln(2\pi\omega_p) + \frac{\left(\mathbf{x}^T(i)\boldsymbol{\theta}_p - y(i)\right)^2}{\omega_p}\right) \\ &\quad + \text{dof}\left(m, \mathbf{X}_{(\boldsymbol{\Gamma},p)}, \mathbf{y}_{(\boldsymbol{\Gamma},p)}, \mathbf{T}_{(\boldsymbol{\Gamma},p)}\right) \ln(k), \\ &= \text{Tr}\left(\boldsymbol{\Gamma}_p\right) \ln\left(2\pi\omega_p\right) + \frac{\left|\left|\mathbf{X}_{(\boldsymbol{\Gamma},p)}\boldsymbol{\theta}_p - \mathbf{y}_{(\boldsymbol{\Gamma},p)}\right|\right|^2}{\omega_p} \\ &\quad + \text{dof}\left(m, \mathbf{X}_{(\boldsymbol{\Gamma},p)}, \mathbf{y}_{(\boldsymbol{\Gamma},p)}, \mathbf{T}_{(\boldsymbol{\Gamma},p)}\right) \ln(k). \end{aligned} \tag{4.29}$$

Then, at each iteration of the EM algorithm, the number of latent variables used for the PLS model of expert p is determined by:

$$Me_p = \arg\min_{m \in \mathcal{M}} \text{BIC}_{\text{E}}(p,m). \tag{4.30}$$

4.2.4 Modeling the Gates with the PLS Algorithm

Let the gate parameters be $\mathbf{V} = \{\mathbf{v}_p | \, p = 2, \ldots, P\}$, where $\mathbf{v}_p$ is the regression coefficient of gate p. In this work, the gate of each expert in (4.5) is modeled using

the softmax function as follows:

$$\upsilon_p(i) = p\left(z_p(i)=1|\mathbf{x}(i),\mathbf{V}\right) = \begin{cases} \frac{1}{1+\sum_{l=2}^{P}\exp(\mathbf{x}^T(i)\mathbf{v}_l)}, & p=1, \\ \frac{\exp\left(\mathbf{x}^T(i)\mathbf{v}_p\right)}{1+\sum_{l=2}^{P}\exp(\mathbf{x}^T(i)\mathbf{v}_l)}, & p=2,\ldots,P, \end{cases} \tag{4.31}$$

where $\upsilon_p(i)$ is used as a simplified notation for $\upsilon_p(\mathbf{x}(i),\mathbf{V})$.

It can be seen that (4.31) keeps valid the constraint $\sum_{p=1}^{P} p\left(z_p(i)=1|\mathbf{x}(i),\mathbf{V}\right) = 1$. Then, the gate contribution $Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})$ to $Q_{\text{ME}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})$ (see (4.17), (4.19)) can be rewritten as:

$$\begin{aligned} Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}}) &= \sum_{i=1}^{k}\sum_{p=1}^{P}\gamma_p^{\text{old}}(i)\ln p\left(z_p(i)=1|\mathbf{x}(i),\mathbf{V}\right), \\ &= \sum_{i=1}^{k}\left[\sum_{p=2}^{P}\gamma_p^{\text{old}}(i)\,\mathbf{x}^T(i)\mathbf{v}_p \right. \\ &\quad \left. - \sum_{p=1}^{P}\gamma_p^{\text{old}}(i)\ln\left(1+\sum_{l=2}^{P}\exp\left(\mathbf{x}^T(i)\mathbf{v}_l\right)\right)\right]. \end{aligned} \tag{4.32}$$

In order to find the parameters $\mathbf{V}$ to update the gating parameters in the M step, it is necessary to maximize equation (4.32). The maximization of $Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})$ with respect to each gate parameter $\mathbf{v}_p$ is going to be obtained by the iterative reweighted least squares (IRLS) method [Jordan, 1994; Nabney, 1999] as follows:

$$\mathbf{v}_p^{\text{new}} = \mathbf{v}_p^{\text{old}} + \left[-\frac{\partial^2 Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})}{\partial\mathbf{v}_p\mathbf{v}_p^T}\right]^{-1}\left[\frac{\partial Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})}{\partial\mathbf{v}_p}\right]. \tag{4.33}$$

From (4.32), the derivatives in (4.33) can be obtained:

$$\left[-\frac{\partial^2 Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})}{\partial\mathbf{v}_p\mathbf{v}_p^T}\right]^{-1} = (\mathbf{X}^T\mathbf{R}_p\mathbf{X})^{-1}, \tag{4.34}$$

$$\left[\frac{\partial Q_g(\mathbf{V},\boldsymbol{\vartheta}^{\text{old}})}{\partial\mathbf{v}_p}\right] = \mathbf{X}^T\mathbf{u}_p, \tag{4.35}$$

where $\mathbf{R}_p = \text{diag}(\upsilon_p(1)(1-\upsilon_p(1)), \upsilon_p(2)(1-\upsilon_p(2)), \ldots, \upsilon_p(k)(1-\upsilon_p(k)))$ is a diagonal matrix and $\mathbf{u}_p = \left[\gamma_p^{\text{old}}(1)-\upsilon_p(1), \gamma_p^{\text{old}}(2)-\upsilon_p(2), \ldots, \gamma_p^{\text{old}}(k)-\upsilon_p(k)\right]^T$. After some manipulations, equation (4.33) can be transformed to:

$$\mathbf{v}_p^{\text{new}} = \left(\mathbf{X}^T\mathbf{R}_p\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{R}_p\mathbf{z}_p, \tag{4.36}$$

where $\mathbf{z}_p = \mathbf{X}\mathbf{v}_p^{\text{old}} - \mathbf{R}_p^{-1}\mathbf{u}_p$. Now the parameters $\mathbf{v}_p$ for $p > 1$ can be solved using the PLS algorithm, similarly to the method that was used to determine the expert parameters (Section 4.2.3). Using (4.1)-(4.2), the weighted input and output values are written in terms of their latent and loading variables as follows:

$$\mathbf{X}_{(\mathbf{R},p)} = \sqrt{\mathbf{R}_p}\,\mathbf{X} \approx \mathbf{T}_{(\mathbf{R},p)}\mathbf{P}_{(\mathbf{R},p)}^T, \tag{4.37}$$

$$\mathbf{z}_{(\mathbf{R},p)} = \sqrt{\mathbf{R}_p}\,\mathbf{z}_p \approx \mathbf{T}_{(\mathbf{R},p)}\mathbf{B}_{(\mathbf{R},p)}\mathbf{Q}_{(\mathbf{R},p)}^T, \tag{4.38}$$

where $\mathbf{X}_{(\mathbf{R},p)}$ and $\mathbf{z}_{(\mathbf{R},p)}$ are the weighted input matrix and weighted output vector of model p with weight matrix $\mathbf{R}_p$, and $\mathbf{T}_{(\mathbf{R},p)}$ and $\mathbf{P}_{(\mathbf{R},p)}$ are the latent and loading matrices of weighted input $\mathbf{X}_{(\mathbf{R},p)}$ and similarly, $\mathbf{B}_{(\mathbf{R},p)}$ and $\mathbf{Q}_{(\mathbf{R},p)}^T$ are the latent and loading matrices of weighted output $\mathbf{z}_{(\mathbf{R},p)} = [z_{(\mathbf{R},p)}(1), \ldots, z_{(\mathbf{R},p)}(k)]^T$. It is assumed that the weighted input and output decomposition through the PLS algorithm is made with Mg_p latent variables.

Then, from (4.36)-(4.38) the parameters vector of each gate p is updated using the PLS algorithm as follows:

$$\begin{aligned}
\mathbf{v}_p^{\text{new}} &= \left(\mathbf{X}_{(\mathbf{R},p)}^T\mathbf{X}_{(\mathbf{R},p)}\right)^{-1}\mathbf{X}_{(\mathbf{R},p)}^T\mathbf{z}_{(\mathbf{R},p)}, \\
&= \left(\left(\mathbf{T}_{(\mathbf{R},p)}\mathbf{P}_{(\mathbf{R},p)}^T\right)^T\left(\mathbf{T}_{(\mathbf{R},p)}\mathbf{P}_{(\mathbf{R},p)}^T\right)\right)^{-1}\left(\mathbf{T}_{(\mathbf{R},p)}\mathbf{P}_{(\mathbf{R},p)}^T\right)^T\mathbf{T}_{(\mathbf{R},p)}\mathbf{B}_{(\mathbf{R},p)}\mathbf{Q}_{(\mathbf{R},p)}^T, \\
&= \left(\mathbf{P}_{(\mathbf{R},p)}\mathbf{P}_{(\mathbf{R},p)}^T\right)^{-1}\mathbf{P}_{(\mathbf{R},p)}\mathbf{B}_{(\mathbf{R},p)}\mathbf{Q}_{(\mathbf{R},p)}^T.
\end{aligned} \tag{4.39}$$

As in the case of the expert model parameters, the number of latent variables to represent $\mathbf{X}_{(\mathbf{R},p)}$ and $\mathbf{z}_{(\mathbf{R},p)}$ should be recomputed at each new iteration. The parameter vector solution (4.39) of gate p has a weighted least squares solution, similar to the solution (4.27) of parameter vector of expert p. Then, the BIC for a gate p can be computed by adapting the expression for the BIC of expert p (4.29) by changing the weighted input, $\mathbf{X}_{(\mathbf{\Gamma},p)}$, and output, $\mathbf{y}_{(\mathbf{\Gamma},p)}$, to $\mathbf{X}_{(\mathbf{R},p)}$ and $\mathbf{z}_{(\mathbf{R},p)}$, respectively, and redefining the variance ω_p to ϖ_p. Then, the BIC value for a gate p, represented by $\text{BIC}_\text{G}(p,m)$ is equal to:

$$\begin{aligned}
\text{BIC}_\text{G}(p,m) = \text{Tr}\left(\mathbf{R}_p\right)\ln\left(2\pi\varpi_p\right) + \frac{\left|\left|\mathbf{X}_{(\mathbf{R},p)}\mathbf{v}_p - \mathbf{z}_{(\mathbf{R},p)}\right|\right|^2}{\varpi_p} \\
+ \text{dof}\left(m, \mathbf{X}_{(\mathbf{R},p)}, \mathbf{z}_{(\mathbf{R},p)}, \mathbf{T}_{(\mathbf{R},p)}\right)\ln(k),
\end{aligned} \tag{4.40}$$

where ϖ_p is the variance of the Gaussian model that models the uncertainty of

$z_{(\mathbf{R},p)}(i)$:

$$\varpi_p = \frac{\left|\left|\mathbf{z}_{(\mathbf{R},p)} - \mathbf{X}_{(\mathbf{R},p)}\mathbf{v}_p\right|\right|^2}{\mathrm{Tr}(\mathbf{R}_p)}. \tag{4.41}$$

Then, the number of latent variables Mg_p used for the PLS gate at each iteration is determined by:

$$Mg_p = \arg\min_{m \in \mathcal{M}} \mathrm{BIC}_{\mathrm{G}}(p, m). \tag{4.42}$$

The parameter $\mathbf{v}_p$ for $p = 1, \ldots, P$, of the softmax function, (4.31), is known to suffer from instability in the maximum likelihood estimation of the parameters when the data samples are separable or quasi-separable. In these situations, the vector $\mathbf{v}_p$ tends to infinity in the maximization of log likelihood (4.32). However, the PLS estimation (4.39) tends to alleviate this problem by combining the input variables into a new set of latent variables, reducing the effect of input variables which are responsible for the data separation. Nonetheless, during the Mix-PLS learning by the EM algorithm, it is possible to detect the instability of parameter estimation by using the Hessian matrix (4.34). If the values of the terms in (4.34) are very large or if it is not possible to compute the inverse, then it is possible to restart the learning of Mix-PLS or just reset the value of vector $\mathbf{v}_p$ to its initial value.

4.2.5 Selecting the Number of Mixture Models

The standard mixture of linear regression models (MLRE) is sensitive to the number of experts used to compose the mixture. As the number of expert models increases, the training data is better fitted. However, the mixtures with too many experts tend to overfit the training data and show poor generalization performance.

In contrast, the Mix-PLS is less prone to overfitting, even with a large number of models. This happens because the parameters of each expert and each gate are solved in a low dimensional space spanned by the results of the PLS algorithm. Moreover, the number of latent variables selected to represent each expert and each gate through the PLS algorithm is determined using the BIC criterion which penalizes complex models, then avoiding overfitting.

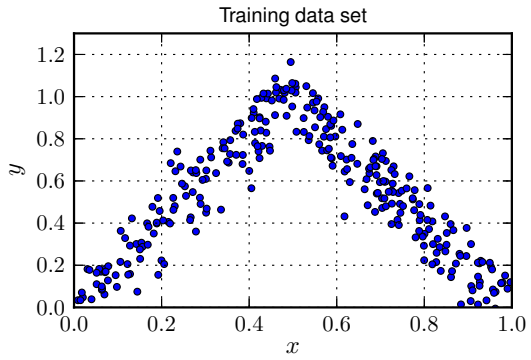


Figure 4.2: Output y defined in equation (4.43).

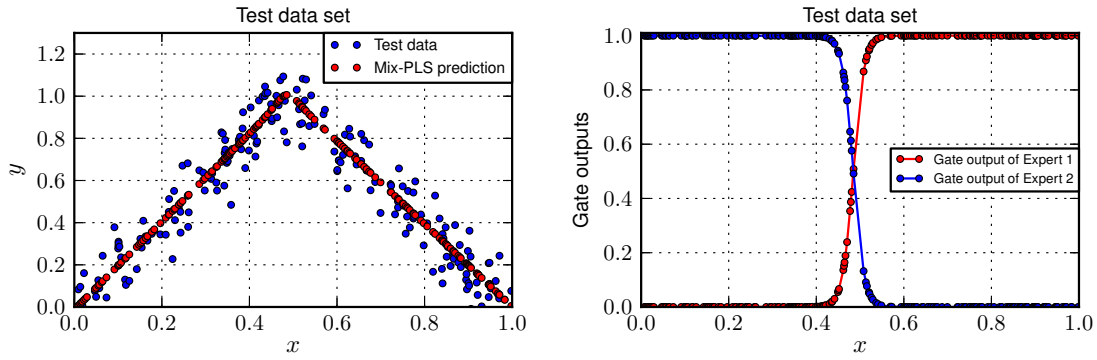
Mix-PLS and Overfitting

Here a small example is studied to demonstrate the robustness of Mix-PLS to overfitting with respect to the number of experts. An artificial data set containing 500 samples was created to compare the performance of Mix-PLS with the MLRE with respect to the number of mixture models. The output y of the artificial model is defined as follows:

$$y(k) = \begin{cases} 2x_1(k) + \mathcal{N}(0, 0.1), \text{if } x_1(k) \leq 0.5, \\ 2 - 2x_1(k) + \mathcal{N}(0, 0.1), \text{if } x_1(k) > 0.5, \end{cases} \tag{4.43}$$

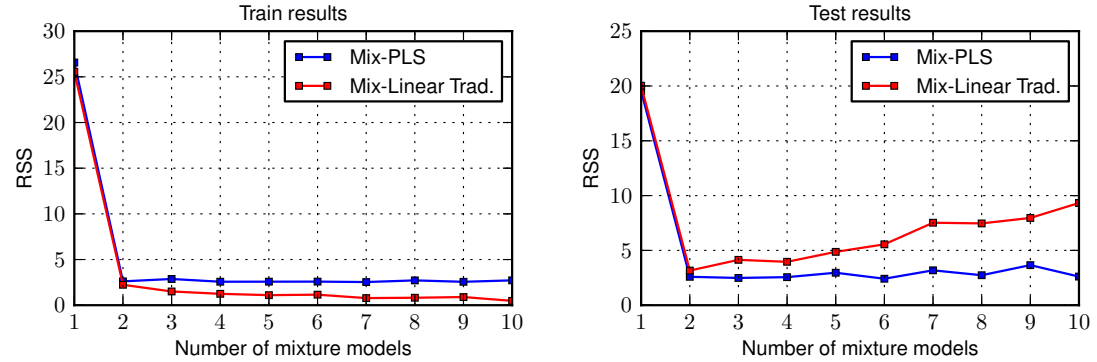
where x_1 was randomly generated with a uniform distribution over $[0, 1]$ and $\mathcal{N}(0, 0.1)$ is a zero-mean Gaussian random variable with 0.1 variance. From the 500 generated samples, 300 were used for training and the remaining 200 were used to testing. The output y of the training data set is represented in Figure 4.2. In this experiment the Mix-PLS and the MLRE were learned using variable x_1 jointly with more 20 irrelevant variables which were added to the data set. The irrelevant variables were generated from a multivariate Gaussian distribution with randomly selected mean and covariance matrix. The values of variables were normalized to be over $[0, 1]$.

The results of using Mix-PLS with two mixture models ($P = 2$) to learn the function (4.43) are shown in Figure 4.3. Figure 4.3a shows the fitting results on the test data set, where it is possible to conclude that the performance of Mix-PLS is good, since it can model the function (4.43). Figure 4.3b shows the output of the gating functions, used to select which model is responsible to predict the output. It is possible to note that two distinct models are used to predict in the test data set.



(a) (b)

Figure 4.3: (a) Prediction results and (b) gate outputs on the Mix-PLS on the test set of the artificial data set.



(a) (b)

Figure 4.4: Performance comparison between the Mix-PLS and the MLRE on the artificial data set for different numbers of mixture models: (a) training data set, and (b) test data set.

Figures 4.4a and 4.4b show the performances, measured by the RSS between the predicted and real output, of the Mix-PLS and the MLRE on the train and test data sets, respectively, when changing the number of mixture models. As can be noticed, on the training data set, the traditional solution fits better as the number of expert models increases. On the other hand, the Mix-PLS results show a constant performance on the training data set. On the test results, it is possible to see that the MLRE tends to overfit the training data as the number of mixture models increases,

then providing poor generalization results. The performance of the Mix-PLS on the test data set is much better, and as mentioned before Mix-PLS is less prone to overfitting.

Number of Experts Selection

To select the number of mixture models, this work will use the criterion suggested by [Jacobs *et al.*, 1997; Ng *et al.*, 2006], where for each expert p, a *worth index* is defined as:

$$I_p = \frac{1}{k} \sum_{i=1}^{k} \gamma_p(i). \tag{4.44}$$

In a mixture of P_e experts, without loss of generality assume that $I_1 \geq I_2 \geq \ldots \geq I_{P_e}$. Then, as defined in [Jacobs *et al.*, 1997], the number of experts, P, is selected as the minimum number of experts with the largest worth indices for which the sum of their worth indices exceeds some threshold value τ, i.e.:

$$P = \min \left\{ P^* : \sum_{p=1}^{P^*} I_p > \tau, \text{ and } P^* \leq P_e, \text{ and } I_1 \geq I_2 \geq \ldots \geq I_{P_e} \right\}. \tag{4.45}$$

The $(P_e - P)$ models with the lowest worth indices can be pruned from the mixture of experts. In [Jacobs *et al.*, 1997] it is suggested the value of $\tau = 0.8$, which has shown to work well in practice.

4.3 Experimental Results

This section presents experimental results of the Mix-PLS applied in three real-world prediction problems. In two of the three processes / data sets, two targets are to be predicted. The prediction will be performed separately for each of the outputs in these data sets. A summary of the data sets is given in Table 4.1. As the objective of this work is to evaluate the proposed method, and not to discuss the process itself, only a short description of each process/dataset is given as follows:

1. SRU: This data set covers the estimation of hydrogen sulfide (H_2S) and sulfur dioxide (SO_2) in the tail stream of a sulfur recovery unit [Fortuna *et al.*, 2006, Chapter 5]. The original data set contains 10072 samples, and in this work the

Table 4.1: Summary of data sets.

Data set	**#Inputs**	**#Train**	**#Test**
SRU: (H_2S)	20	2000	8072
SRU: (SO_2)	20	2000	8072
Polymerization (Viscosity)	24	521	133
Polymerization (Acidity)	24	521	133
Spectra	401	48	12

learning set includes the first 2000 samples for training and the remaining 8072 samples for test (as in the original work [Fortuna *et al.*, 2006]). The data set contains five input variables: x_1, x_2, x_3, x_4, x_5. By considering lagged inputs, the inputs considered in the models, are: $x_1(k), x_1(k-5), x_1(k-7), x_1(k-9), \ldots, x_5(k), x_5(k-5), x_5(k-7), x_5(k-9)$, making a total of 20 input variables. According to the authors [Fortuna *et al.*, 2006], the preferred models are the ones that are able to accurately predict peaks in the H_2S and SO_2 concentrations in the tail gas;

2. Polymerization: The objective on this data set is the estimation of the quality of a resin produced in an industrial batch polymerization process [Facco *et al.*, 2009]. The resin quality is determined by the values of two chemical properties: the resin acidity number (N_A), and the resin viscosity (μ). The data set is composed of 24 input variables and the authors [Facco *et al.*, 2009] have predefined 521 samples for training and 133 for test;

3. Spectra: The objective in this data set is the estimation of octane ratings based on the near infrared (NIR) spectral intensities of 60 samples of gasoline at 401 wavelengths [Kalivas, 1997]. This data set was split in 80% for training and the remaining 20% was used for test.

In all experiments, the values of both the training samples, and the testing samples, were normalized to have zero mean and unit variance. In the experiments, with exception for the Spectra data set, the Mix-PLS, MLRE, and PLS models were tuned by using as input of the model the original variables plus the squared values of these variables; the objective for using the squared values of the input

variables is to introduce some nonlinearity into the linear models (Mix-PLS, MLRE, and PLS). In the experiments, for all data sets presented in Table 4.1, the proposed Mix-PLS method will be compared with the MLRE, a single PLS model, a MLP trained using the gradient descent training algorithm, and a LS-SVR with Gaussian kernel [Suykens *et al.*, 2002, Chapter 3]. From the results, it can be seen that Mix-PLS attains better results when compared with MLRE, and PLS, and when compared with the MLP and LS-SVR non-linear models. Moreover, the Mix-PLS has the advantage of having more interpretability with respect to its parameters when compared with the MLP and LS-SVR non-linear models.

In all data sets, the normalized root mean square error (NRMSE) (3.32) was used as a performance measure to compare the results of the methods.

4.3.1 Evaluation and Discussion

The number of hidden nodes h of the MLP, and the regularization parameter $\gamma_{\text{LS-SVR}}$ and the Gaussian kernel parameter $\sigma_{\text{LS-SVR}}$ of the LS-SVR, were determined using 10-fold cross validation and the MSE as the criterion function. For the PLS model the number of latent variables M, was determined using the BIC criterion as discussed in Section 4.1.1. For the MLRE, and Mix-PLS the numbers of experts P were obtained from (4.45). Additionally, for the Mix-PLS the set that contains the numbers of latent variables for each expert $Me = \{Me_1, \ldots, Me_p\}$ was obtained from (4.30), and the corresponding set of numbers of latent variables for the gates $Mg = \{Mg_2, \ldots, Mg_p\}$ was obtained from (4.42). Table 4.2 shows the parameters obtained for each model and for each data set in the experiments.

SRU Data Set

For the prediction of H_2S in the SRU data set, the NRMSE performances on the test set for all models, are indicated in Table 4.3. These results indicate that the Mix-PLS has the best performance among all the models. Further analysis on the Mix-PLS results, in Figure 4.5, reveals that for the H_2S prediction, the Mix-PLS was able to identify two different operating modes, which are modeled by two experts. The first expert is the most used for predicting in the regular operation and the second expert is most used to predict peaks, as can be verified by the gates output in Figure 4.5. The prediction results on the test set, shown in 4.5b, indicate that,

Table 4.2: Parameters selected for each model and for each data set.

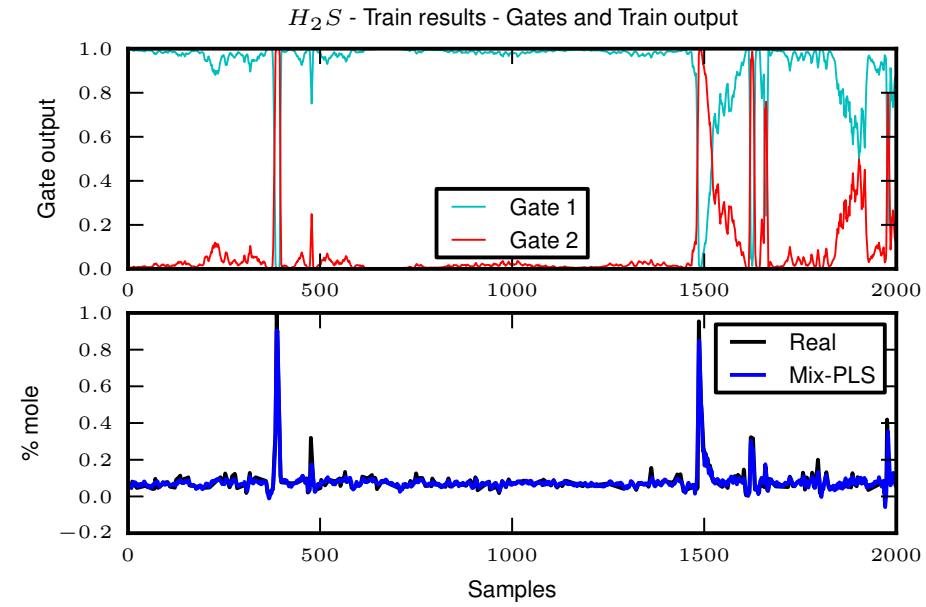
Data set name	**Mix-PLS**	**MLRE**	**PLS**	**MLP**	**LS-SVR**
SRU: (H_2S)	$P = 2$ $Me_p = \{14, 17\}$ $Mg_p = \{7\}$	$P = 2$	$M = 10$	$N = 9$	$\gamma_{\text{LS-SVR}} = 50$ $\sigma_{\text{LS-SVR}} = 5$
SRU: (SO_2)	$P = 2$ $Me_p = \{14, 15\}$ $Mg_p = \{10\}$	$P = 2$	$M = 12$	$N = 3$	$\gamma_{\text{LS-SVR}} = 50$ $\sigma_{\text{LS-SVR}} = 5$
Poly.: (Viscosity)	$P = 2$ $Me_p = \{18, 8\}$ $Mg_p = \{2\}$	$P = 2$	$M = 10$	$N = 3$	$\gamma_{\text{LS-SVR}} = 50$ $\sigma_{\text{LS-SVR}} = 10$
Poly.: (Acidity)	$P = 2$ $Me_p = \{20, 15\}$ $Mg_p = \{2\}$	$P = 2$	$M = 17$	$N = 3$	$\gamma_{\text{LS-SVR}} = 50$ $\sigma_{\text{LS-SVR}} = 25$
Spectra	$P = 4$ $Me_p = \{40, 25, 26, 27\}$ $Mg_p = \{1, 1, 36\}$	$P = -$	$M = 24$	$N = 6$	$\gamma_{\text{LS-SVR}} = 50$ $\sigma_{\text{LS-SVR}} = 25$

Table 4.3: NRMSE results on the test data sets.

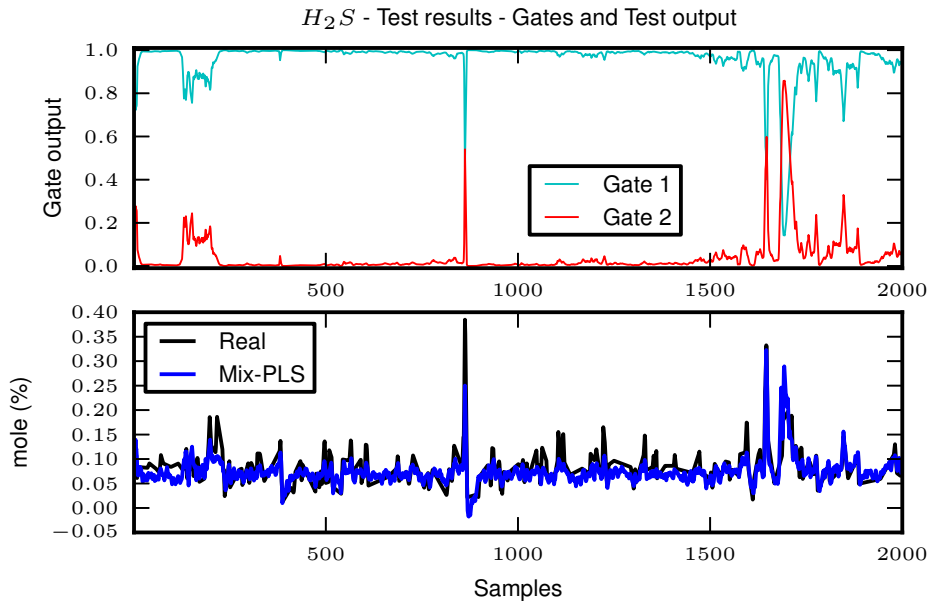
NRMSE					
Data set name	**Mix-PLS**	**MLRE**	**PLS**	**MLP**	**LS-SVR**
SRU: (H_2S)	**4.59**	5.75	6.43	10.41	9.14
SRU: (SO_2)	**3.35**	5.36	3.57	3.95	5.66
Poly.: (Viscosity)	**8.07**	23.43	24.23	9.95	12.38
Poly.: (Acidity)	**3.62**	5.54	4.25	3.93	5.94
Spectra	**6.91**	–	9.14	8.61	28.52

on unseen data, the Mix-PLS performs very well during the prediction, including in the prediction in peak periods.

For the SO_2 prediction, the performances of all models using the NRMSE criterion are indicated in Table 4.3. It is shown that in this experiment, the Mix-PLS has the best performance among all the models, and the PLS and MLP models have results close to Mix-PLS. However, the Mix-PLS is more attractive than the MLP and PLS models, because of the interpretability of its parameters and the ability



(a)



(b)

Figure 4.5: Plots of H_2S prediction on the SRU data set. (a) Train results, gates and prediction. (b) Test results, gates and prediction. For better visualization, only 2000 samples are shown.

to learn multiple operating modes. On this data set, the Mix-PLS was able also to identify two operating modes. The prediction results on the train and test sets are shown in Figure 4.6.

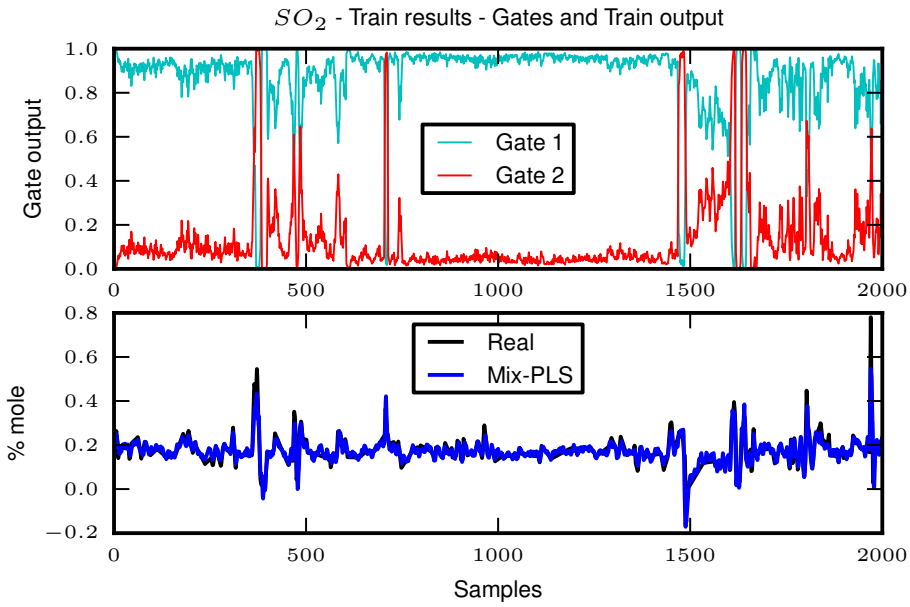
From the H_2S and SO_2 results on the SRU data set, it is possible to conclude that the Mix-PLS was able to identify two different operating modes, in the two data sets. According to [Fortuna *et al.*, 2006], on the SRU data set, the preferred models are the ones that are able to accurately predict peaks. From the SRU results it is possible to note that one expert is more responsible for predicting the regular operation mode, while the other expert is able to predict the peaks.

Polymerization Data Set

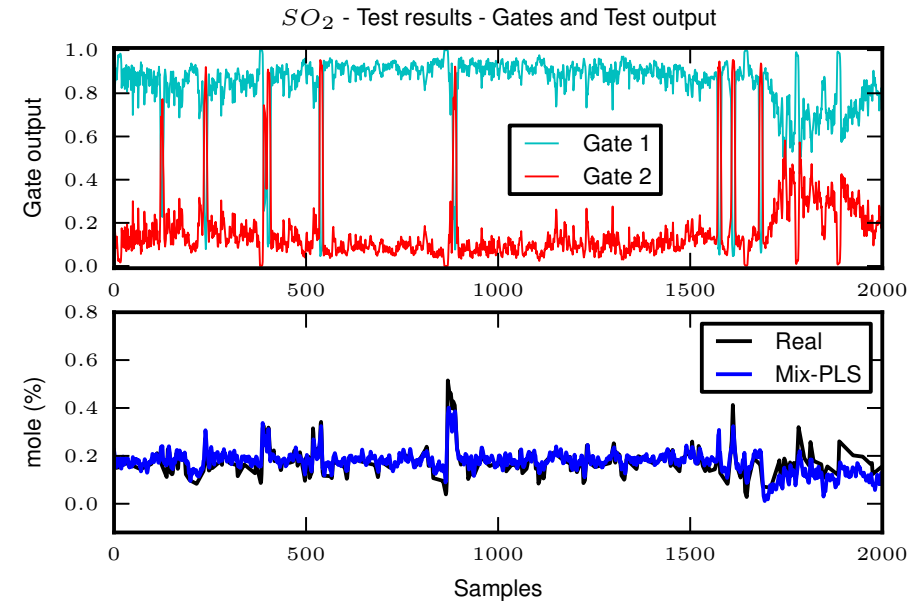
This data set was studied in [Facco *et al.*, 2009], and the objective is to estimate the viscosity and acidity of a resin produced in an industrial batch polymerization process. According to Table 4.3, for predicting the viscosity, the Mix-PLS reached the best results among all the models in terms of NRMSE. Inspecting the results from the gates activation on the train and test sets which are presented in Figure 4.7, it is possible to note that the prediction of the first expert is predominant at the beginning of each batch, and, on the other hand / other areas, the prediction of the two models are combined, usually at the end of each batch. The Mix-PLS suggests, that for viscosity prediction, just two models are necessary and that their prediction should be combined at the end of each batch.

For predicting the acidity, the Mix-PLS also reached the best results in terms of NRMSE, as indicated in Table 4.3. The Mix-PLS used 2 experts to predict the acidity. The plots of gates and prediction on the train and test sets are shown in Figure 4.8. Differently from the viscosity prediction, the models are combined at the beginning of each batch and then, one expert is predominant in the rest of the batch.

As can be seen the Mix-PLS was successfully applied on the Polymerization data set, delivering satisfactory prediction results. Moreover, Mix-PLS has shown better results when compared to the linear models and to the nonlinear models.

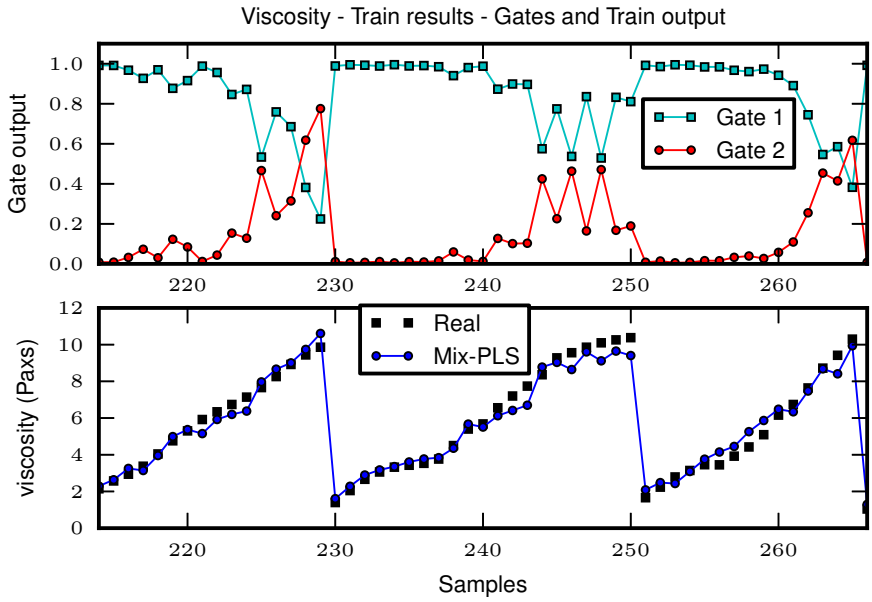


(a)

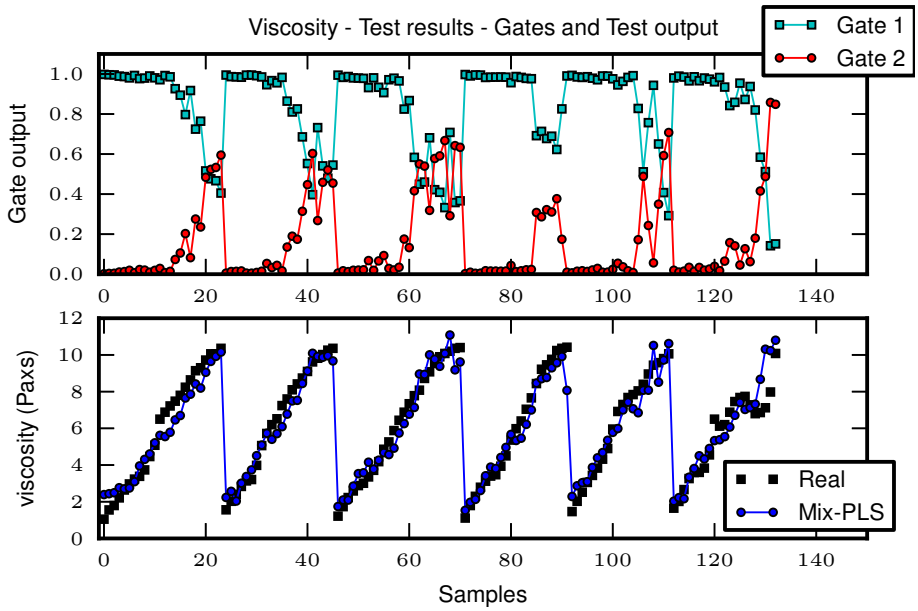


(b)

Figure 4.6: Plots of SO_2 prediction on SRU data set. (a) Train results, gates and prediction. (b) Test results, gates and prediction. For better visualization, only 2000 samples are shown.

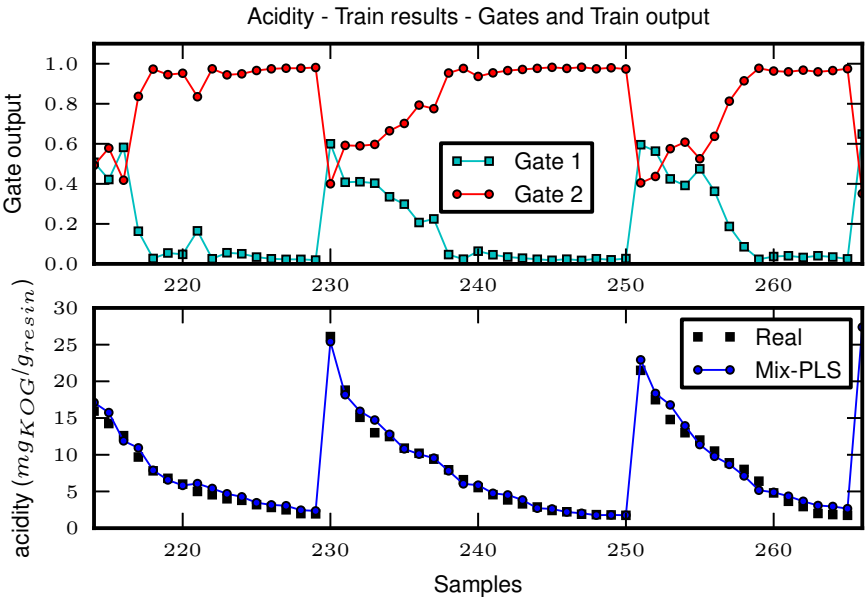


(a)

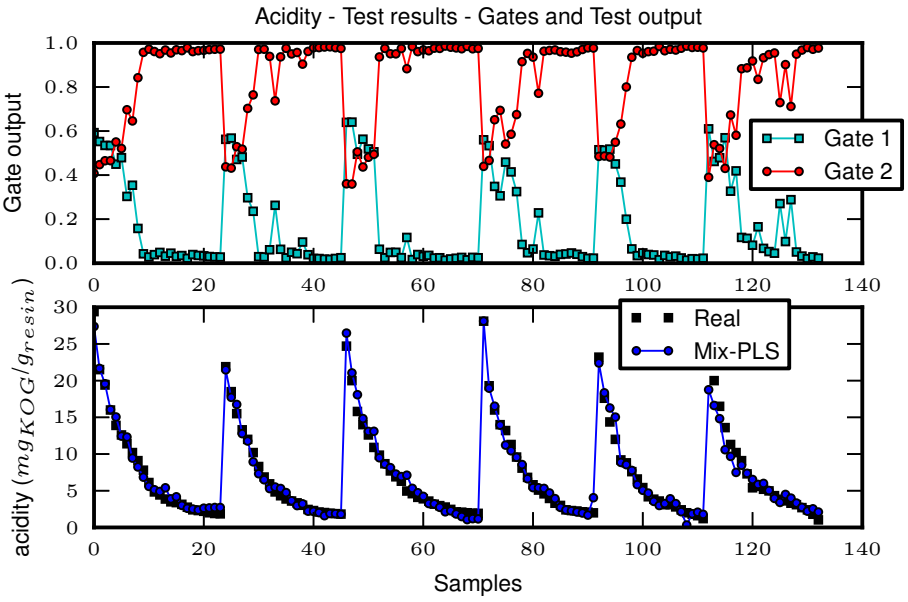


(b)

Figure 4.7: Plots of viscosity prediction on Polymerization data set. (a) Train results, gates and prediction. (b) Test results, gates and prediction.



(a)



(b)

Figure 4.8: Plots of acidity prediction on Polymerization data set. (a) Train results, gates and prediction. (b) Test results, gates and prediction.

Spectra Data Set

This Spectra data set was analyzed in [Kalivas, 1997], and the objective is the estimation of the octane ratings based on the near infrared (NIR) spectral intensities of 60 samples of gasoline at 401 wavelengths. This data set is characterized by having only a few samples and a large number of input variables. Moreover, it is known *a priori* that this data set does not have multiple operating modes. Thus, the analysis is focused in the prediction performance. According to Table 4.3, the Mix-PLS reached the best results among all the models in terms of NRMSE and the MLRE method did not converge in this experiment. Moreover, Mix-PLS has shown much better results when compared with the nonlinear models in this data set.

4.4 Discussion

Performing the selection of the number of latent variables at each iteration of the Mix-PLS algorithm, in our case by the BIC criterion, is not obligatory, but it is recommended. Other options are to run the Mix-PLS algorithm with a fixed number of latent variables or to select the number of variables after the overall run of the algorithm. The use of a validation data set can also be a good option to select the number of latent variables.

The expectation of the complete data log likelihood value (4.11) in the Mix-PLS is monotonically increasing in most iterations. This is more evident in the initial iterations of the algorithm, however, very infrequently, in some iterations the likelihood decreases its value. However, the overall trend is to obtain an increasing likelihood. Such characteristic is expected in the proposed Mix-PLS approach, since the selection of the latent variables by the BIC criterion, which is directly built into the model learning, and is performed at each iteration, avoids overfitting on the training data. By avoiding complex models, the BIC criterion penalizes the likelihood in the algorithm, during the selection of the latent variables.

It is already know that the first two data sets, Polymerization and SRU, have multiple operating modes, and the analysis of the results in both data sets has emphasized this case. From the results it is seen that Mix-PLS is more than a good non-linear regression method, also it picks/assigns different operating modes in/to different experts. However, although these results are representative, they are

also conditioned to the problem under study, i.e. it is not possible to assure that the separate assignment of different modes to different experts is a general property that holds for all other conceivable problems. However, the application of the proposed approach is not limited to multiple operating modes and it can also be used as a general non-linear regression method, as in the case of Spectra data set.

4.5 Conclusion

In this chapter, a method for dealing with multiple operating modes in soft sensor applications was presented. In the proposed Mix-PLS method, the solution of the mixture of linear regression models is done using the PLS model. The formulas for learning were derived based on the EM algorithm. Furthermore, in this work the proposed method has been evaluated and compared with the current state of art methods on three real-world data sets, encompassing the prediction of five variables.

In comparison with the traditional solution of the mixture of linear regression models, the Mix-PLS is much less prone to overfitting with respect to the number of mixture models to be used, while still attaining good prediction results, as demonstrated in an artificial data set experiment. In the real-world data sets experiments, all the results obtained with Mix-PLS were superior when compared with MLRE, a single PLS, MLP, and LS-SVR models, and differently from the non-linear models, the Mix-PLS gives more interpretability to the prediction.

Chapter 5

Mixture of Univariate Linear Regression Models for Adaptive Soft Sensors

Contents

This chapter introduces a novel method for application in soft sensors/regression settings with time-varying characteristics.

When a soft sensor is working online, it is crucial that it can work properly over time, predicting as accurately as possible the target variable, even under "unknown changes" of the process. Unfortunately, in most of the cases reported in the literature, the soft sensor does not react well under "unknown changes", and the response becomes unreliable. Such changes are caused mainly because process plants are rather nonstationary dynamic environments. Such nonstationarity arises from, and is a function of, changes in the process, such as change in feedstock or product grade, changes in the external environments (weather, seasons), production of different product quality grades, process fouling or abrasion of mechanical components [Kadlec *et al.*, 2011]. The ideal soft sensor is the one built with a data set containing all possible future states and conditions of the process. If these temporal changes are present/described in the available historical data, then it is possible to assume the presence of multiple operating modes, and it would be plausible to create a soft sensor based on the Mix-PLS model, described in Chapter 4. However, it may not be an easy task to build such data set.

Typically, laboratory measurements are regularly acquired from the process, even after soft sensor deployment. Such procedure is still necessary to verify the performance of the deployed soft sensor. To overcome the deterioration of the soft sensor in the presence of process variations it is necessary to apply strategies to adapt the soft sensor by using the incoming samples of the process. This can include the use of the laboratory measurements taken from the process, to adapt the soft sensor model. As discussed in Chapter 2, there are three strategies to adapt/build an adaptive soft sensor model, which are the sample selection, sample weighting, and

ensemble learning. Each of these approaches has its advantages and drawbacks, and its utilization is dependent on the problem at hand.

For example, the sample selection strategy, for model adaptation, does not necessarily need a method for adaptive[1] learning of model parameters. The sample selection strategy is based on a window approach, where the samples which are inside of a window are used to retrain/update the model, while samples outside of the window are discarded. Such approach allows the use of any model, even those who do not have an adaptive learning method. The major drawback of such approach is due to the need to retrain the model several times and the necessity of keeping the samples in memory [Lee *et al.*, 2005; Liukkonen *et al.*, 2013].

The other two approaches, sample weighting and ensemble learning, require an online learning strategy for adapting their models. In the case of an ensemble learning strategy, the adaptation is at the level of the model combination, or/and at the level of the models, and it requires a large amount of memory to store the models belonging to the ensemble. However, the ensemble learning has the ability to detect and predict recurrent concepts [Fu *et al.*, 2008; Kadlec and Gabrys, 2011]. On the other hand, the sample weighting strategy is not able to detect recurrent concepts, but the sample weighting does not need the use of memory to store the samples. In the sample weighting approach, the learning/adaptation of parameters is usually done using adaptive learning by means of exponentially recursive learning. The adaptive learning is the same as the recursive or online learning (i.e. each sample is presented once and only once to learn/adapt the parameters) but it has a plus on it because of its ability to forget old examples by exponentially assigning low weights to old samples, usually by setting a forgetting factor $0 < \lambda < 1$. Such approach is very popular in soft sensors applications, mainly in the RPLS model, with several applications in industry [Helland *et al.*, 1992; Komulainen *et al.*, 2004; Li *et al.*, 2005; Mu *et al.*, 2006; Haavisto and Hyötyniemi, 2009; Facco *et al.*, 2010; Wang *et al.*, 2010; Kadlec and Gabrys, 2011; Muradore and Fiorini, 2012].

This forgetting factor λ has influence on the speed of model adaptation and in the model learning. In particular, small values of forgetting factor affect the performance of the existing recursive models, RLS and RPLS. In this work, it is assumed that this is caused mainly because small values of forgetting factor, in adaptive sce-

[1]Synonymous terms are online learning, online identification, real-time identification, adaptive algorithm, sequential estimation, and incremental learning.

narios, result in problems similar to the ones faced when modeling static systems with a small number of samples, such as overfitting, poor prediction performance, etc. To solve these problems, a new model, based on a mixture of univariate linear regression models (MULRM) is proposed, allowing the use of small values of forgetting factor. It has been shown experimentally that the proposed method provides the best results when working with small values of forgetting factor, being suitable to be applied in such scenarios. Thus, looking to reduce the dimensionality of the learning process, the proposed MULRM consists of a mixture of low dimensional models, where the individual models are combined such that the prediction error is minimized. Specifically, the MULRM has the following form:

$$f(\mathbf{x}, \boldsymbol{\Theta}, \boldsymbol{\Upsilon}) = \sum_{p=1}^{D} \upsilon_p f_p(x_p, \boldsymbol{\theta}_p), \text{ with } \sum_{p=1}^{D} \upsilon_p = 1, \tag{5.1}$$

where $f_p(x_p, \boldsymbol{\theta}_p) = \theta_{p0} + \theta_{p1} x_p$ is a univariate linear regression model of variable x_p, $\boldsymbol{\theta}_p = [\theta_{p0}, \theta_{p1}]^T$, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_p | \, j = 1, \ldots, D\}$ denotes the set of all weight parameters, and $\boldsymbol{\Upsilon} = \{\upsilon_p | \, p = 1, \ldots, D\}$ denotes the set of mixing coefficients. The individual models are then combined, and the Expectation-Maximization (EM) algorithm [Dempster *et al.*, 1977; Bishop, 2006] is employed to jointly estimate the model parameters $\boldsymbol{\Upsilon}$ and $\boldsymbol{\Theta}$. The recursive solution for the MULRM parameters, $\boldsymbol{\Theta}$, $\boldsymbol{\Upsilon}$, will be derived in the next sections. A forgetting factor will be introduced in the online solution to discount the information coming from the already learned data, so that it can be applied in time varying scenarios. The solution of the proposed method allows its online and recursive application in any regression problem, without the necessity to store any past value of data. As will be discussed in this chapter, the individual models $f_p(x_p, \boldsymbol{\theta}_p)$ can take other forms in addition to the univariate linear model case, including nonlinear forms.

In the rest of the chapter, during the derivation of MULRM, brief explanations about online learning, as well as a review on the most common adaptive soft sensor models (RLS and RPLS), will be given. In the experimental part, the recursive solution of the MULRM is then applied in two time-varying real-world prediction problems. Moreover, the proposed MULRM method is compared in these problems with four state of the art algorithms: the RLS, the RPLS, the online sequential extreme learning machine (OS-ELM) [Nan-Ying *et al.*, 2006], a fast learning algorithm for single hidden layer feedforward ANN, with offline and online solutions, and

the recently proposed incremental local learning soft sensing (ILLSA) algorithm for adaptive soft sensors [Kadlec and Gabrys, 2011]. The experimental results suggest that the recursive MULRM outperforms the RLS, RPLS, OS-ELM, and ILSSA, when predicting in time-varying scenarios.

5.1 Background

The notation used throughout this chapter is the one defined in Section 3.1, and complementary notation, necessary for this chapter, will be introduced in the next sections. A subscript k will be used as one of the ways to denote the value of the corresponding variable after k samples, for example $y_k = y(k)$.

In regression tasks, the objective is to make use of an input vector $\mathbf{x}$ to describe/approximate a target variable y, where a set of examples $\Phi = \{(\mathbf{x}(i), y(i)); i = 1, \ldots, k\}$ is used to train a model to do this. Similarly to (3.1), it is assumed that y can be approximated by a deterministic function $\hat{y} = f(\mathbf{x}, \boldsymbol{\theta})$, governed by a parameter vector $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_{D(\theta)}]^T$, so that:

$$y = f(\mathbf{x}, \boldsymbol{\theta}) + \xi, \tag{5.2}$$

where ξ, the approximation error, is a zero mean random variable with variance ω. Under the assumption of Gaussian noise, the conditional probability of y given the input $\mathbf{x}$ can be represented by the normal distribution $p(y|\mathbf{x}, \boldsymbol{\theta}, \omega) = \mathcal{N}(y|f(\mathbf{x}, \boldsymbol{\theta}), \omega)$.

5.1.1 Online Learning

In adaptive scenarios, a historical data set $\Phi_k = \{(\mathbf{x}(i), y(i))\ ;\ i = 1, \ldots, k\}$, with k samples is available.

In some cases, including in soft sensor applications, another historical data set is also available, called here the complete data set, $\Phi''_m = \{(\mathbf{x}(e), y(e))\ ;\ e = 1, \ldots, m\}$, where every element of Φ''_m includes the input vector $\mathbf{x}(e)$, and for some elements of Φ''_m the output $y(e)$ is known, while for other elements $y(e)$ is unknown (the NaN notation will be used to denote an unknown value). In many soft sensor applications, $y(e) = \text{NaN}$ for many elements of Φ''_m. Assume that $\Phi'_k = \{(\mathbf{x}(e), y(e)) \in \Phi''_m\ ;\ e = \epsilon(1), \ldots, \epsilon(k)\} = \{(x(i), y(i)) \big| (x(e), y(e)) \in \Phi''_m \wedge y(e) \neq \text{NaN}\}$ is the subset of all the k elements of Φ''_m that have $y(e) \neq \text{NaN}$, where $\{\epsilon(1), \ldots, \epsilon(k)\}$ is a subsequence of

$\{1, \ldots, m\}$. Without loss of generality, a renumbering operation of the independent time variable is performed such that $\Phi_k^{'}$ is transformed into the above defined Φ_k; This corresponds to renumber the time instants $\{\epsilon(1), \ldots, \epsilon(k)\}$ of the elements of $\Phi_k^{'}$ into the time instants $\{1, \ldots, k\}$ of the elements of Φ_k. The dynamics of the system to be modeled/predicted is represented in the datasets $\Phi_k^{'}$ and Φ_k by the fact that, for each time instant, the input vector may contain values of system variables corresponding to several (discrete real-)time instants.

The objecive for the prediction setting is to predict the values of $y(e)$ for $e = k+1, \ldots, k+U$, when these values of $y(e+1), \ldots, y(e+U)$ are unknown, where U is some positive integer. For that purpose, a model $\mathcal{M}_k$ is learned, in batch or recursive mode, using all, or only a selection, of the samples in Φ_k. The inputs $\mathbf{x}(e+1), \ldots, \mathbf{x}(e+U)$ are fed into the model $\mathcal{M}_k$ to predict the outputs, where $\hat{y}(e+u)$ is the predicted output of sample $(e+u)$, for $u = 1, \ldots, U$. At instant $e+U+1$, a sample $y(e+U+1) \neq \text{NaN}$ becomes available (i.e. a complete data pair becomes available). The historical data increases with the addition of sample $(\mathbf{x}(e+U+1), y(e+U+1))$ to Φ_k, and the resulting new data set (including the sample renumbering operation) is represented by $\Phi_{k+1} = \{(\mathbf{x}(i), y(i))\ ;\ i = 1, \ldots, k+1\}$. When a new sample $\mathbf{x}(e+U+2)$ arrives, and assuming $y(e+U+2) = \text{NaN}$, then a model $\mathcal{M}_{k+1}$ is learned using all or a selection of the samples in Φ_{k+1}, to predict $y(e+U+2)$, and so on.

5.1.2 Recursive Least Squares

The RLS is the traditional method of recursive learning. Assume a regression model, linear in the parameters, and composed by a linear combination of functions of the vector of input variables as follows:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \boldsymbol{\Phi}^T(\mathbf{x})\, \boldsymbol{\theta}, \tag{5.3}$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_D]^T$ is the vector of model parameters, and $\boldsymbol{\Phi}(\mathbf{x}) = [\phi_1(\mathbf{x}), \ldots, \phi_D(\mathbf{x})]^T$ is a vector of known functions. Assume a weighted LS (WLS) problem where the weighted squared error is given by $E_{\mathbf{W}}(\boldsymbol{\theta}) = (\mathbf{y} - \mathbf{A}\boldsymbol{\theta})^T\, \mathbf{W}\, (\mathbf{y} - \mathbf{A}\boldsymbol{\theta})$, where $\mathbf{W}$ is a symmetric positive definite weighting ma-

trix, $\mathbf{A}$ is a $k \times D$ design matrix, and $\mathbf{y}$ is a $k \times 1$ output vector, where:

$$\mathbf{A} = \mathbf{A}(k) = \begin{bmatrix} \mathbf{\Phi}^T(\mathbf{x}(1)) \\ \vdots \\ \mathbf{\Phi}^T(\mathbf{x}(k)) \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y(1) \\ \vdots \\ y(k) \end{bmatrix}. \tag{5.4}$$

For example, for a completely linear problem where (5.3) reduces to $f(\mathbf{x}, \boldsymbol{\theta}) = \mathbf{x}^T\boldsymbol{\theta}$, specifically, where $\phi_1(\mathbf{x}) = 1$, $\phi_p(\mathbf{x}) = x_{p-1}$, for $p = 2, \ldots, D+1$, then

$$\mathbf{A} = \mathbf{A}(k) = \begin{bmatrix} 1 & \mathbf{x}^T(1) \\ \vdots & \vdots \\ 1 & \mathbf{x}^T(k) \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y(1) \\ \vdots \\ y(k) \end{bmatrix}. \tag{5.5}$$

When $\mathbf{W} = \mathbf{I}$ is the identity matrix, the WLS problem reduces to a simple LS problem.

Given a training data set with k examples, denoted by Φ_k, the WLS solution to $\boldsymbol{\theta}$ in (5.3) is given by:

$$\boldsymbol{\theta} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{y}, \tag{5.6}$$

When dealing with time-varying environments, and when the examples are delivered sequentially over the time, instead of using the closed form solution (5.6), an incremental learning approach [Jang *et al.*, 1997] with forgetting factor λ which places more emphasis on the more recent data, can be employed to update the weight vector $\boldsymbol{\theta}$, and a weighting diagonal matrix $\mathbf{W} = \text{diag}(\lambda^{k-1}, \lambda^{k-2}, \ldots, 1)$ is defined. Assume a new sample $(\mathbf{x}(k+1), y(k+1))$, or equivalently $(\mathbf{\Phi}^T(\mathbf{x}(k+1)), y(k+1)) = (\mathbf{a}(k+1), y(k+1))$, is available. For example, for the above mentioned completely linear problem, $\mathbf{a}(k+1) = \left[1, \mathbf{x}^T(k+1)\right]^T$. Then, when a new sample is available, the weight vector $\boldsymbol{\theta}$ can be incrementally updated as follows:

$$\mathbf{P}(k+1) = \lambda^{-1}\left(\mathbf{P}(k) - \frac{\mathbf{P}(k)\mathbf{a}(k+1)\mathbf{a}^T(k+1)\mathbf{P}(k)}{\lambda + \mathbf{a}^T(k+1)\mathbf{P}(k)\mathbf{a}(k+1)}\right), \tag{5.7}$$

$$g(k+1) = \mathbf{P}(k+1)\mathbf{a}(k+1), \tag{5.8}$$

$$e(k+1) = y(k+1) - \mathbf{a}^T(k+1)\boldsymbol{\theta}(k), \tag{5.9}$$

$$\boldsymbol{\theta}(k+1) = \boldsymbol{\theta}(k) + g(k+1)e(k+1), \tag{5.10}$$

where $\mathbf{P}(k+1) = \left(\mathbf{A}(k+1)^T\mathbf{A}(k+1)\right)^{-1}$, $g(k+1)$ is the gain, $e(k+1)$ is the error of the present sample calculated with the old parameter $\boldsymbol{\theta}(k)$, and $0 < \lambda \leq 1$ is

the forgetting factor, so that the smaller the λ parameter, the larger is the weight of the recent data, and the more the RLS estimator can track the time-varying parameters. If $\lambda = 1$, then the update rule (5.7)-(5.10) becomes the recursive estimator of equation (5.6).

5.1.3 Recursive Partial Least Squares

The PLS algorithm described in Section 4.1, is designed for the offline case. However, when dealing with time-varying environments, and the examples are delivered sequentially over the time, the PLS solution is achieved by merging the old model, represented by the matrices $\mathbf{P}, \mathbf{B}$ and $\mathbf{Q}$, with the new sample. Assuming that $(\mathbf{x}(k+1), y(k+1))$ represents the new sample, then:

$$\mathbf{X}_{k+1} = \begin{bmatrix} \lambda \mathbf{P}^T \\ \mathbf{x}^T(k+1) \end{bmatrix}; \quad \mathbf{y}_{k+1} = \begin{bmatrix} \lambda \mathbf{B}\mathbf{Q}^T \\ y(k+1) \end{bmatrix}, \tag{5.11}$$

where λ is the forgetting factor, similarly to the LS estimator, where lower values of λ indicate that the recent data will influence more the new model. Then, $\mathbf{X}_{k+1}$ and $\mathbf{y}_{k+1}$ can be applied to the NIPALS estimator [Wold, 1975] to find the new parameters of the PLS model as in Section 4.1. The above update is restricted to the case where the number of latent variables is selected to be equal to the rank of $\mathbf{X}$ [Qin, 1998].

5.2 Motivation for Using Mixture of Univariate Linear Regression Models

So far the RLS solution is the most popular method for recursive learning of linear models. Another common approach is the RPLS algorithm described in the previous section. Both the RLS and the RPLS are learned by minimizing the following exponential weighting error function:

$$E_{\mathbf{W}}(\boldsymbol{\theta}) = \frac{1}{\mathrm{Tr}(\boldsymbol{\Lambda})} \sum_{i=1}^{k} \lambda^{k-i} [y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})]^2, \tag{5.12}$$

where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda^{k-1}, \lambda^{k-2}, \ldots, 1)$. When k tends to infinity, $\mathrm{Tr}(\boldsymbol{\Lambda}) = \frac{1}{1-\lambda}$ (in exponentially recursive learning it is possible to assume that k tends to infinity,

since input samples arrive continually). Then, regarding the learning of parameters using exponential weighting of samples, it is important to note that the amount of data used to construct the multivariate model, represented here as $p(y|\mathbf{x})$, i.e. the effective number of observations being used, defined as d, is related to the forgetting factor λ as $d = \mathrm{Tr}(\mathbf{\Lambda}) = 1/(1-\lambda)$ [Dayal and MacGregor, 1997]. Then, assuming that d plays a similar role as the number of training samples in the static case, it can be concluded that in time-varying scenarios small values of d (or small values of λ) lead to the same problems faced by learning a static model with small number of training samples, such as overfitting or poor generalization on the test set [Raudys and Jain, 1991], mainly if $d < D$. Thus, from this assumption, during learning in time-varying scenarios, the value of λ not only interferes in the speed of the model adaptation, but also it has influence in the model learning (i.e. in the recursive parameters learning) and its problems.

Then, the motivation behind the use of MULRM in time-varying scenarios, is that the MULRM has the benefit of separately estimating the pdf of y conditioned to each individual input variable, $p(y|x_j)$, rather than estimating the conditional pdf $p(y|\mathbf{x})$ with all input variables. In situations where only a small number of samples is available for learning (i.e. when d is small or $d < D$), the estimation of univariate $p(x_j)$, is more accurate than the estimation of the full pdf $p(\mathbf{x})$. The same thought is valid for the conditional pdf $p(y|x_j)$, $p(y|\mathbf{x})$ [Frank *et al.*, 2000]. The amount of data needed to obtain an accurate estimate increases with the dimensionality of the problem. Then, in some cases $p(y|x_j)$ can be estimated more reliably than $p(y|\mathbf{x})$. Moreover, this dimensionality reduction, while estimating $p(y|x_j)$, makes the learning problem much easier. However, the main drawback is that for representing y, $p(y|x_j)$ is less representative than $p(y|\mathbf{x})$. To overcome this effect, or to try to minimize it, the proposed MULRM merges all the individual $p(y|x_j)$ models ($j = 1, \ldots, D$) to estimate $p(y|\mathbf{x})$ by using the mixture of models (MM) framework.

Moreover, due to the above discussed characteristics, the MULRM can be considered a good option for prediction in two possible scenarios. The first scenario is when the dimensionality of the input space is larger than the number of samples; i.e. in situations where $k < D$ (some regularization techniques such as the elastic net [Friedman *et al.*, 2010] can also be employed in this case). The second situation (which is the main target of this work), is in soft sensor applications in time-varying scenarios. In scenarios of this later type, the forgetting factor λ employed to weight

the samples is related to the effective number of samples by $d = \frac{1}{1-\lambda}$, and the value chosen for λ can be small, leading to $d < D$, i.e. MULRM is a good option when the effective number of samples is less than the number of input variables.

5.3 Mixture of Models

The mixture of models (MM) method approximates the true pdf $p(y|\mathbf{x})$ with the following superposition of individual pdfs:

$$p(y(i)|\mathbf{x}(i), \boldsymbol{\vartheta}) = \sum_{p=1}^{P} \upsilon_p \, p(y(i)|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega}), \tag{5.13}$$

where $\boldsymbol{\vartheta} = \{\boldsymbol{\Theta}, \boldsymbol{\Upsilon}, \boldsymbol{\Omega}\}$, $p(y(i)|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega})$ is the pdf describing y given $\mathbf{x}$, with mean $f_p(\mathbf{x}, \boldsymbol{\theta}_p)$ and additional pdf parameters $\boldsymbol{\Omega}$, $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_p | \, j = 1, \ldots, D\}$, and $\boldsymbol{\Upsilon} = \{\upsilon_p | \, p = 1, \ldots, D\}$. $\boldsymbol{\theta}_p$ a the vector of parameters of model p, more precisely $\boldsymbol{\theta}_p$ is the vector of parameters of $f_p(\mathbf{x}, \boldsymbol{\theta}_p)$ (the mean of the probabilistic model), υ_p is the mixing coefficient for model p which satisfies $0 \leq \upsilon_p \leq 1$ for $p = 1, \ldots, P$, and $\sum_{p=1}^{P} \upsilon_p = 1$. From (5.13), the prediction equation of the MM is obtained as the following conditional mean of y given $\mathbf{x}$ [Bishop, 2006, Chapter 1]:

$$\begin{aligned}
\mathbb{E}[y|\mathbf{x}(i)] &= f(\mathbf{x}(i), \boldsymbol{\vartheta}) \\
&= \int y p(y|\mathbf{x}(i), \boldsymbol{\Theta}, \boldsymbol{\Upsilon}, \boldsymbol{\Omega}) \mathrm{d}y \\
&= \int y \sum_{p=1}^{P} \upsilon_p \, p(y|f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\Omega}) \mathrm{d}y \\
&= \sum_{p=1}^{P} \upsilon_p f_p(\mathbf{x}(i), \boldsymbol{\theta}_p).
\end{aligned} \tag{5.14}$$

$f(\mathbf{x}(i), \boldsymbol{\vartheta})$ is the function which minimizes the expected squared loss $\mathcal{L} = \int \int \left(f(\mathbf{x}(i), \boldsymbol{\vartheta}) - y\right)^2 p(\mathbf{x}, y) d\mathbf{x} dy$ [Bishop, 2006].

In the MM, the log-likelihood of (5.13), given a set of observations Φ is given by:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\vartheta}) &= \ln \sum_{\mathbf{Z}} p(\mathbf{Z}) p(\mathbf{y}|\mathbf{X},\mathbf{Z},\boldsymbol{\Theta},\boldsymbol{\Omega}) \\ &= \ln \left(\prod_{i=1}^{k} p(y(i)|\mathbf{x}(i),\boldsymbol{\vartheta}) \right) \\ &= \ln \left(\prod_{i=1}^{k} \sum_{\mathbf{z}(i)} p(\mathbf{z}(i))\, p(y(i)|\mathbf{x}(i),\mathbf{z}(i),\boldsymbol{\Theta},\boldsymbol{\Omega}) \right), \end{aligned} \tag{5.15}$$

where $\mathbf{Z} = [z_{ip}] = [z_p(i)] \in \mathbb{R}^{k \times P}$ denotes a set of hidden variables and $\mathbf{z}(i) = [z_1(i), \ldots, z_P(i)]^T \in \mathbb{R}^P$ is the vector of hidden variables for a sample i, where $z_p(i) \in \{0, 1\}$, for $p = 1, \ldots, P$, and for each sample $\mathbf{z}(i)$, all variables $z_p(i)$ are zero, except for a single value of $z_p(i) = 1$, for some p. The hidden variable $z_p(i)$ indicates which model p is responsible for generating the data sample i. The distribution $p(y(i)|\mathbf{x}(i),\mathbf{z}(i),\boldsymbol{\Theta},\boldsymbol{\Omega})$ is defined in Chapter 4, equation (4.9), and $p(\mathbf{z}(i))$ is defined such that $p(z_p(i) = 1) = \upsilon_p$, thus $p(\mathbf{z}(i))$ is defined by:

$$p(\mathbf{z}(i)) = \prod_{p=1}^{P} \upsilon_p^{z_p(i)}. \tag{5.16}$$

Then, (5.15) becomes:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\vartheta}) &= \ln \left(\prod_{i=1}^{k} \sum_{\mathbf{z}(i)} p(\mathbf{z}(i))\, p(y(i)|\mathbf{x}(i),\mathbf{z}(i),\boldsymbol{\Theta},\boldsymbol{\Omega}) \right) \\ &= \sum_{i=1}^{k} \ln \left(\sum_{p=1}^{P} p\left(\mathbf{z}(i)|z_p(i)=1\right)\; p\left(y(i)|z_p(i)=1,\mathbf{x}(i),\boldsymbol{\Theta},\boldsymbol{\Omega}\right) \right) \\ &= \sum_{i=1}^{k} \ln \left(\sum_{p=1}^{P} \upsilon_p\, p\left(y(i)|z_p(i)=1,\mathbf{x}(i),\boldsymbol{\Theta},\boldsymbol{\Omega}\right) \right). \end{aligned} \tag{5.17}$$

In order to maximize (5.17), the EM algorithm is going to be employed similarly to what was performed in the ME in Chapter 4. The EM algorithm for learning the parameters of the MM model is described in Algorithm 5.1. For Step 6 of Algorithm 5.1 it is necessary to estimate the posterior distribution of hidden variables $p(\mathbf{Z}|\mathbf{y},\mathbf{X},\boldsymbol{\vartheta}^{\text{old}})$ and the distribution of the complete data $p(\mathbf{y},\mathbf{Z}|\mathbf{X},\boldsymbol{\vartheta})$. Following

Algorithm 5.1 EM algorithm for MM

1: **Input:** data set Φ;
2: **Output:** $\boldsymbol{\vartheta}$;
3: **Initialization:** Initialize $\boldsymbol{\vartheta}$ equal to some initial $\boldsymbol{\vartheta}^{\text{old}}$;
4: **repeat**
5: **E step:**
6: Estimate the distribution $p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}^{\text{old}})$ using (5.18);
7: **M step:**
8: Find the new parameter values $\boldsymbol{\vartheta}^{\text{new}}$, which maximize the expectation of the complete-data log likelihood $Q_{\text{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$:
$\boldsymbol{\vartheta}^{\text{new}} = \arg\max_{\boldsymbol{\vartheta}} Q_{\text{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$ (Equation (5.22));
$\boldsymbol{\vartheta}^{\text{old}} \leftarrow \boldsymbol{\vartheta}^{\text{new}}$;
9: **until** convergence is attained;
10: **return** $\boldsymbol{\vartheta}^{\text{new}}$.

equations (4.12) and (4.14), and replacing $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})$ in these equations, by $p(\mathbf{Z})$[2], equation (5.16), then the posterior distribution of hidden variables and the distribution of the complete data are:

$$p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\vartheta}) p(\mathbf{Z})}{p(\mathbf{y}|\mathbf{X}, \boldsymbol{\vartheta})} = \prod_{i=1}^{k} \prod_{p=1}^{P} \left(\frac{\upsilon_p \, p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right)}{\sum_{p=1}^{P} \left[\upsilon_p \, p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right)\right]} \right)^{z_p(i)}, \tag{5.18}$$

$$p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}) = p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\vartheta}) \, p(\mathbf{Z}) = \prod_{i=1}^{k} \prod_{p=1}^{P} \left[\upsilon_p \, p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right) \right]^{z_p(i)}, \tag{5.19}$$

where $\boldsymbol{\mathcal{E}} = \{\boldsymbol{\Theta}, \boldsymbol{\Omega}\}$ is defined as the set of models parameters, and $p(\mathbf{y}|\mathbf{X}, \mathbf{Z}, \boldsymbol{\vartheta})$, $p(\mathbf{y}|\mathbf{X}, \boldsymbol{\vartheta})$, and $p(\mathbf{Z})$ are given by, (4.9), (5.13) and (5.16), respectivelly.

Then, computing the expectation of $p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})$ with respect to $p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta})$,

[2]It is possible to see that $p(\mathbf{Z}|\mathbf{X})$ in the MM is equal to $p(\mathbf{Z})$. In MM $\mathbf{X}$ and $\mathbf{Z}$ are independent, thus, ignoring parameter $\boldsymbol{\vartheta}$, $p(\mathbf{Z}|\mathbf{X}) = \frac{p(\mathbf{Z},\mathbf{X})}{p(\mathbf{X})} = \frac{p(\mathbf{Z})\,p(\mathbf{X})}{p(\mathbf{X})} = p(\mathbf{Z})$.

yields:

$$\begin{aligned} Q_{\mathrm{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}}) &= \mathbb{E}_{\mathbf{Z}}\left[\ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta})\right] \\ &= \sum_{\mathbf{Z}} \left[\ln p(\mathbf{y}, \mathbf{Z}|\mathbf{X}, \boldsymbol{\vartheta}) p(\mathbf{Z}|\mathbf{y}, \mathbf{X}, \boldsymbol{\vartheta})\right] \\ &= \sum_{i=1}^{k} \sum_{p=1}^{P} \gamma_p^{\mathrm{old}}(i) \left[\ln(\upsilon_j) + \ln\left(p\left(y(i)|z_p(i), \mathbf{x}(i), \boldsymbol{\mathcal{E}}\right)\right)\right], \end{aligned} \tag{5.20}$$

where $\gamma_p^{\mathrm{old}}(i) = \mathbb{E}[z_p(i)]^{\mathrm{old}}$, defined as the responsibility of model p, accounts for the probability of model p generating the data sample i. For the MM, using Bayes's Theorem [Bishop, 2006], it is equal to:

$$\begin{aligned} \gamma_p^{\mathrm{old}}(i) &= \mathbb{E}_{\mathbf{Z}}[z_p^{\mathrm{old}}(i)] \\ &= p\left(z_j(i) = 1 | f_p(\mathbf{x}(i), \boldsymbol{\theta}_p), \boldsymbol{\mathcal{E}}^{\mathrm{old}}\right) \\ &= \frac{\upsilon_p^{\mathrm{old}}\, p\left(y(i)|z_p(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}^{\mathrm{old}}\right)}{\sum_{l=1}^{P}\left[\upsilon_l^{\mathrm{old}}\, p\left(y(i)|z_l(i) = 1, \mathbf{x}(i), \boldsymbol{\mathcal{E}}^{\mathrm{old}}\right)\right]}. \end{aligned} \tag{5.21}$$

In the maximization step, function $Q_{\mathrm{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}})$ is maximized with respect to $\boldsymbol{\vartheta}$, using the responsibilities (5.21) computed in the expectation step, as follows:

$$\boldsymbol{\vartheta}^{\mathrm{new}} = \arg\max_{\boldsymbol{\vartheta}} Q_{\mathrm{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}}). \tag{5.22}$$

This maximization can be achieved by finding the solution $\boldsymbol{\vartheta}$ of equating to zero the derivative of (5.20), with respect to $\boldsymbol{\vartheta}$, i.e. the maximization can be achieved by solving $\partial Q_{\mathrm{MM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\mathrm{old}})/\partial\boldsymbol{\vartheta} = 0$.

5.4 Mixture of Univariate Linear Regression Models - Offline Solution

The MULRM is based on the MM framework. Each model p in (5.13) is given by a univariate model $p(y|f_p(x_p, \boldsymbol{\theta}_p), \boldsymbol{\Omega})$, and the total number of models is equal to the number of variables $P = D$. Then, the MULRM approximates the true pdf $p(y|\mathbf{x})$ with the following superposition of individual pdfs:

$$p(y|\mathbf{x}, \boldsymbol{\vartheta}) = \sum_{p=1}^{D} \upsilon_p\, p(y|f_p(x_p, \boldsymbol{\theta}_p), \boldsymbol{\Omega}), \tag{5.23}$$

where $\boldsymbol{\vartheta} = \{\boldsymbol{\Theta}, \boldsymbol{\Upsilon}, \boldsymbol{\Omega}\}$, $p(y|f_p(x_p, \boldsymbol{\theta}_p), \boldsymbol{\Omega})$ is the pdf describing y given x_p, with mean $f_p(x_p, \boldsymbol{\theta}_p)$ and additional pdf parameters $\boldsymbol{\Omega}$. $\boldsymbol{\theta}_p$ is the vector of parameters of model p, more precisely $\boldsymbol{\theta}_p$ is the vector of parameters of the model $f_p(x_p, \boldsymbol{\theta}_p)$, υ_p is the mixing coefficient for model p, which satisfies $0 \leq \upsilon_p \leq 1$ for $p = 1, \ldots, P$, and $\sum_{p=1}^{D} \upsilon_p = 1$. $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_p | \, j = 1, \ldots, D\}$, and $\boldsymbol{\Upsilon} = \{\upsilon_p | \, p = 1, \ldots, D\}$. From (5.23), prediction equation (5.1) is obtained as the following conditional mean of y given $\mathbf{x}$, yielding a mixture of linear univariate models:

$$\begin{aligned}
\mathbb{E}[y|\mathbf{x}] &= f(\mathbf{x}, \boldsymbol{\Theta}, \boldsymbol{\Upsilon}) \\
&= \int y p(y|\mathbf{x}, \boldsymbol{\Theta}, \boldsymbol{\Upsilon}, \boldsymbol{\Omega}) \mathrm{d}y \\
&= \int y \sum_{p=1}^{D} \upsilon_p p(y|f_j(x_p, \boldsymbol{\theta}_p), \boldsymbol{\Omega}) \mathrm{d}y \\
&= \sum_{p=1}^{D} \upsilon_p f_p(x_p, \boldsymbol{\theta}_p). \qquad (5.24)
\end{aligned}$$

Now assume that each individual pdf $p(y|f_p(x_p, \boldsymbol{\theta}_p), \boldsymbol{\Omega})$ in (5.23) is described by a Gaussian distribution. Then, (5.23) can be rewritten as:

$$p(y|\mathbf{x}, \boldsymbol{\vartheta}) = \sum_{p=1}^{D} \upsilon_p \mathcal{N}(y|f_p(x_p, \boldsymbol{\theta}_p), \omega_p), \qquad (5.25)$$

where ω_p is the variance of model p, and $f_p(\cdot)$ can take any form, such as for example:

1. Linear, the canonical MULRM configuration:

$$f_p(x_p) = \theta_{p0} + \theta_{p1} x_p; \qquad (5.26)$$

2. Polynomial: $f_p(x_p) = \sum_{l=0}^{n} \theta_{pl} x_p^l$;

3. Other non-linear forms, linear in the parameters, of the following form:

$$\begin{aligned}
f_p(x_p) &= \boldsymbol{\phi}(x_p)^T \boldsymbol{\theta}_p, \qquad (5.27) \\
\boldsymbol{\phi}(x_p) &= [1, \phi_1(x_p), \ldots, \phi_n(x_p)]^T, \qquad (5.28) \\
\boldsymbol{\theta}_p &= [\theta_{p0}, \theta_{p1}, \ldots, \theta_{pn}]^T, \qquad (5.29)
\end{aligned}$$

where $\phi_j(x_p)$, for $j = 1, \ldots, n$, are non-linear functions. For example: $f_p(x_p) = \theta_{p0} + \theta_{p1} \ln(x_p)$ or $f_p(x_p) = \theta_{p0} + \theta_{p1} \sqrt{x}_p$;

From (5.25), the log likelihood of a given set of observations Φ is given by:

$$\begin{aligned}\ln p(\mathbf{y}|\mathbf{X},\boldsymbol{\vartheta}) &= \ln\left(\prod_{i=1}^{k} p(y(i)|\mathbf{x}(i),\boldsymbol{\vartheta})\right)\\ &= \sum_{i=1}^{k}\ln\left(\sum_{p=1}^{D}\upsilon_p\,\mathcal{N}\left(y(i)\middle|f_p\left(x_p(i),\boldsymbol{\theta}_p\right),\omega_p\right)\right).\end{aligned} \tag{5.30}$$

The log likelihood of the complete joint conditional distribution with the observed and latent variables is given by [Bishop, 2006, Chapter 9]:

$$\ln p(\mathbf{y},\mathbf{Z}|\mathbf{X},\boldsymbol{\vartheta}) = \sum_{i=1}^{k}\sum_{p=1}^{D} z_p(i)\ln\left(\upsilon_p\,\mathcal{N}\left(y(i)\middle|f_p\left(x_p(i),\boldsymbol{\theta}_p\right),\omega_p\right)\right). \tag{5.31}$$

The EM algorithm for MULRM, described in Algorithm 5.2, starts with an initial value for the model parameters $\boldsymbol{\vartheta}$, called here as $\boldsymbol{\vartheta}^{\text{old}}$. Then, $\boldsymbol{\vartheta}^{\text{old}}$ is used to compute, $\gamma_p^{\text{old}}(i) = \mathbb{E}[z_p(i)]^{\text{old}}$, the responsibility of model f_p for sample i, which accounts for the probability of model p in generating the data sample i. From (5.21) the responsibility is equal to:

$$\begin{aligned}\gamma_p^{\text{old}}(i) &= \mathbb{E}[z_p(i)]^{\text{old}}\\ &= p\left(z_p(i)=1|f_p(x_p,\boldsymbol{\theta}_p),\boldsymbol{\mathcal{E}}^{\text{old}}\right)\\ &= \frac{\upsilon_p^{\text{old}}\,\mathcal{N}\left(y(i)\middle|f_p\left(x_p(i),\boldsymbol{\theta}_p^{\text{old}}\right),\omega_p^{\text{old}}\right)}{\sum_{l=1}^{D}\upsilon_l^{\text{old}}\,\mathcal{N}\left(y(i)\middle|f_l\left(x_l(i),\boldsymbol{\theta}_l^{\text{old}}\right),\omega_l^{\text{old}}\right)}.\end{aligned} \tag{5.32}$$

Then, the responsibilities (5.32) are used to determine the expectation of the complete data log likelihood (5.20) with respect to $\mathbf{Z}$, which for MULRM is equal to:

$$\begin{aligned}Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}}) &= \mathbb{E}_{\mathbf{Z}}\left[\ln p(\mathbf{y},\mathbf{Z}|\mathbf{X},\boldsymbol{\vartheta})\right]\\ &= \sum_{i=1}^{k}\sum_{p=1}^{D}\gamma_p^{\text{old}}(i)\left[\ln(\upsilon_p)+\ln\left(\mathcal{N}\left(y(i)|f_p(x_p(i),\boldsymbol{\theta}_p),\omega_p\right)\right)\right].\end{aligned} \tag{5.33}$$

In the maximization step, function $Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})$ is maximized with respect to $\boldsymbol{\vartheta}$, using the responsibilities (5.32) computed in the expectation step. This maximization can be achieved by solving $\partial Q_{\text{MM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})/\partial\boldsymbol{\vartheta} = 0$. The parameters in $\boldsymbol{\vartheta}$ are $\boldsymbol{\theta}_p$, υ_p, and ω_p for all models f_p. Then, maximizing $Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})$ with

respect to $\boldsymbol{\vartheta}$, i.e. solving equations $\frac{\partial Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})}{\partial \boldsymbol{\theta}_p} = 0$, $\frac{\partial Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})}{\partial \upsilon_p} = 0$, and $\frac{\partial Q_{\text{MULRM}}(\boldsymbol{\vartheta},\boldsymbol{\vartheta}^{\text{old}})}{\partial \omega_p} = 0$, gives the following maximizing solution parameters, for the cases where, such as in (5.26) or in (5.27)-(5.29), $f_p(x_p)$ is linear in the parameters:

$$\boldsymbol{\theta}_p = ((\mathbf{A}_p)^T \boldsymbol{\Gamma}_p \mathbf{A}_p)^{-1} (\mathbf{A}_p)^T \boldsymbol{\Gamma}_p \mathbf{y}, \tag{5.34}$$

$$\upsilon_p = \frac{\sum_{i=1}^{k} \gamma_p^{\text{old}}(i)}{k}, \tag{5.35}$$

$$\omega_p = \frac{\sum_{i=1}^{k} \gamma_p^{\text{old}}(i) \left(y(i) - f_p(x_p(i), \boldsymbol{\theta}_p)\right)^2}{\sum_{i=1}^{k} \gamma_p^{\text{old}}(i)}, \tag{5.36}$$

where $\mathbf{A}_p$ is the design matrix of model p, given by

$$\mathbf{A}_p = \mathbf{A}_p(k) = \begin{bmatrix} 1 & \phi_1(x_p(1)) & \dots & \phi_n(x_p(1)) \\ \vdots & \vdots & & \vdots \\ 1 & \phi_1(x_p(k)) & \dots & \phi_n(x_p(k)) \end{bmatrix}, \tag{5.37}$$

and $\boldsymbol{\Gamma}_p = \boldsymbol{\Gamma}_p(k) = \text{diag}(\gamma_p^{\text{old}}(1), \gamma_p^{\text{old}}(2), \dots, \gamma_p^{\text{old}}(k))$ is a diagonal matrix. For the canonical MULRM case where $f_p(x_p)$ is given by (5.26), $\mathbf{A}_p$ (5.37) reduces to

$$\mathbf{A}_p = \mathbf{A}_p(k) = \begin{bmatrix} 1 & x_p(1) \\ \vdots & \vdots \\ 1 & x_p(k) \end{bmatrix}, \tag{5.38}$$

and for the case where $f_p(x_p)$ has the polynomial form of order n:

$$\mathbf{A}_p = \mathbf{A}_p(k) = \begin{bmatrix} 1 & x_p^1(1) & \dots & x_p^n(1) \\ \vdots & \vdots & & \vdots \\ 1 & x_p^1(k) & \dots & x_p^n(k) \end{bmatrix}.$$

The offline learning of MULRM by the EM algorithm is summarized in algorithm 5.2. The convergence of the EM algorithm can be verified by analyzing the convergence of the expectation $Q_{\text{MULRM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$. It is also possible to set a prespecified maximum number of iterations. Equations (5.34)-(5.36) are the solutions of the maximization step in the EM algorithm, for the combination of the univariate models in the form (5.1), for the cases where the component models $f_p(x_p)$ of MULRM are given by (5.26) (or more generally by (5.27)-(5.29)). It is important to note that while running Algorithm 5.2, the values of $\boldsymbol{\theta}_p$ can take inconsistent values when matrix $((\mathbf{A}_p)^T \boldsymbol{\Gamma}_p \mathbf{A}_p)$ in (5.34) becomes ill conditioned. This happens when the values of γ_p^{old} in $\boldsymbol{\Gamma}_p$ take small values, which is also a problem in (5.36). This situation can be detected through the value of $\text{Tr}(\boldsymbol{\Gamma}_p)$. If $\text{Tr}(\boldsymbol{\Gamma}_p)$ becomes small,

Algorithm 5.2 Offline learning of MULRM

Input: data set Φ;
Output: $\boldsymbol{\vartheta}$;
Initialization: Initialize $\boldsymbol{\vartheta}$ equal to some initial $\boldsymbol{\vartheta}^{\text{old}}$;
repeat
E step:
    Compute the responsibilities (5.32) using $\boldsymbol{\vartheta}^{\text{old}}$;
    Compute the expectation $Q_{\text{MULRM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$ using (5.33);
M step:
    Compute the values of $\boldsymbol{\vartheta}^{\text{new}}$ which maximize $Q_{\text{MULRM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$ using (5.34)-(5.36):
        $\boldsymbol{\vartheta}^{\text{new}} = \arg\max_{\boldsymbol{\vartheta}} Q_{\text{MULRM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$;
        $\boldsymbol{\vartheta}^{\text{old}} \leftarrow \boldsymbol{\vartheta}^{\text{new}}$;
until convergence is attained;
return $\boldsymbol{\vartheta}^{\text{new}}$.

then the value of $\boldsymbol{\theta}_p$ is set to its initial value. Setting $\boldsymbol{\theta}_p$ to its initial value does not affect the performance of the model, since v_p (5.35) is small in this situation, and thus the contribution of model p in (5.1), is also small. The next section will provide an online algorithm for the learning of MULRM (5.1), with a sample weighting adaptation approach.

5.5 Mixture of Univariate Linear Regression Models - Recursive/Online Solution

In the online learning of $\boldsymbol{\vartheta}$, the parameters computed by the offline equations (5.34)-(5.36) of the EM algorithm maximization step should be learned recursively, and each available sample should correspond to an iteration in the EM algorithm. In the solution derived here, a forgetting factor λ will be employed to weight more recent data, making the MULRM able to be applied in time varying scenarios. $0 < \lambda \leq 1$, so that the smaller the λ parameter, the larger is the weight of the recent data, as in the traditional RLS. In this section, for simplicity of notation, γ_p^{old} will be written as γ_p, droping the superscript.

From (5.35) it is possible to note that υ_p is given by the average over the samples of the responsibilities $\gamma_p(i)$ of model f_p. Then, υ_p can be learned using the adaptive recursive mean. Specifically, the mean of $\gamma_p(i)$ among k values, indicated by $\upsilon_p(k)$, can be updated when a new sample $\gamma_p(k+1)$ is available using the following adaptive mean formula, where λ is used to discount the information coming from the already learned data:

$$\upsilon_p(k+1) = \lambda\upsilon_p(k) + (1-\lambda)\gamma_p(k+1). \tag{5.39}$$

Using the same idea, equation (5.36) can be seen as a ratio between two means over the samples k:

$$\omega_p = \frac{\sum_{i=1}^{k} \gamma_p(i)\left(y(i) - f_p(x_p(i), \boldsymbol{\theta}_p)\right)^2 / k}{\sum_{i=1}^{k} \gamma_p(i) / k}, \tag{5.40}$$

where the denominator is equal to υ_p (5.35), and the numerator is equal to the weighted error $E_p = \sum_{i=1}^{k} E_{\gamma_p}(i)/k$ of model f_p in predicting y, where $E_{\gamma_p}(i) = \gamma_p(i)\left(y(i) - f_p(x_j(i), \boldsymbol{\theta}_p)\right)^2$. Similarly to (5.39), the recursive formula for E_p is given by:

$$E_p(k+1) = \lambda E_p(k) + (1-\lambda)E_{\gamma_p}(k+1). \tag{5.41}$$

Then, the value of ω_p when a new sample is available is:

$$\omega_p(k+1) = \frac{E_p(k+1)}{\upsilon_p(k+1)}. \tag{5.42}$$

Let $\mathbf{a}_p(k+1)$ be a new available sample corresponding to $(\mathbf{x}(k+1), y(k+1))$. Assuming the canonical MULRM case where $f_p(x_p)$ is given by (5.26) and $\mathbf{A}_p(k)$ is given by (5.38), then define $\mathbf{a}_p(k+1) = [1, x_p(k+1)]^T$. Assuming that $f_p(x_p)$ is given by (5.27)-(5.29) and $\mathbf{A}_p(k)$ if given by (5.37), then define $\mathbf{a}_p(k+1) = [1, \phi_1(x_p(k+1)), \ldots, \phi_n(x_p(k+1))]^T$. Assuming that a new sample $\mathbf{a}_p(k+1)$ is available, then the closed form solution for $\boldsymbol{\theta}_p$ (5.34) is represented in expanded form as follows:

$$\begin{aligned}
\mathbf{P}_p(k+1) &= \left(\begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Gamma}_p(k) & 0 \\ 0 & \gamma_p(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix} \right)^{-1}, \\
\boldsymbol{\theta}_p(k+1) &= \mathbf{P}_p(k+1) \begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\Gamma}_p(k) & 0 \\ 0 & \gamma_p(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ y(k+1) \end{bmatrix},
\end{aligned} \tag{5.43}$$

where $\gamma_p(k+1)$ and $\boldsymbol{\Gamma}_p(k) = \mathrm{diag}(\gamma_p^{\mathrm{old}}(1), \gamma_p^{\mathrm{old}}(2), \ldots, \gamma_p^{\mathrm{old}}(k))$ represent the values of the responsibilities of the current sample $\mathbf{a}_p(k+1)$ and of the previous samples,

respectively. However, in time-varying systems, the update formula for the weights $\boldsymbol{\theta}_p$ of each model f_p, should also take into consideration the forgetting factor λ. Thus, a matrix of weights $\mathbf{W}(k) = \text{diag}(\lambda^{k-1}, \lambda^{k-2}, \ldots, 1)$ is designed to affect the samples, so that the model could take into consideration recent data with more weight. Then, the closed form, full update equation for $\boldsymbol{\theta}_p$, taking into consideration the forgetting factor is given as follows:

$$\begin{aligned}
\mathbf{P}_p(k+1) &= \left(\begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \lambda \mathbf{W}(k) \boldsymbol{\Gamma}_p(k) & 0 \\ 0 & \gamma_p(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix} \right)^{-1}, \\
\boldsymbol{\theta}_p(k+1) &= \mathbf{P}_p(k+1) \begin{bmatrix} \mathbf{A}_p(k) \\ \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \lambda \mathbf{W}(k) \boldsymbol{\Gamma}_p(k) & 0 \\ 0 & \gamma_p(k+1) \end{bmatrix} \begin{bmatrix} \mathbf{y}_k \\ y(k+1) \end{bmatrix},
\end{aligned} \tag{5.44}$$

or equivalently

$$\begin{aligned}
\mathbf{P}_p(k+1) &= \left(\begin{bmatrix} \sqrt{\boldsymbol{\Gamma}_p(k)}\, \mathbf{A}_p(k) \\ \sqrt{\gamma_p(k+1)}\, \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \lambda \mathbf{W}(k) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{\boldsymbol{\Gamma}_p(k)}\, \mathbf{A}_p(k) \\ \sqrt{\gamma_p(k+1)}\, \mathbf{a}_p^T(k+1) \end{bmatrix} \right)^{-1}, \\
\boldsymbol{\theta}_p(k+1) &= \mathbf{P}_p(k+1) \begin{bmatrix} \sqrt{\boldsymbol{\Gamma}_p(k)}\, \mathbf{A}_p(k) \\ \sqrt{\gamma_p(k+1)}\, \mathbf{a}_p^T(k+1) \end{bmatrix}^T \begin{bmatrix} \lambda \mathbf{W}(k) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \sqrt{\boldsymbol{\Gamma}_p(k)}\, \mathbf{y}_k \\ \sqrt{\gamma_p(k+1)}\, y(k+1) \end{bmatrix}.
\end{aligned} \tag{5.45}$$

Therefore, the recursive formulas for computing (5.34) while also taking into consideration the forgetting factor λ, i.e. the formulas to recursively obtain $\mathbf{P}_p(k+1)$ and $\boldsymbol{\theta}_p(k+1)$ in (5.44), are as follows (see Section 5.1.2):

$$\mathbf{P}_p(k+1) = \lambda^{-1} \left(\mathbf{P}_p(k) - \frac{\gamma_p(k+1) \mathbf{P}_p(k) \mathbf{a}_p(k+1) \mathbf{a}_p^T(k+1) \mathbf{P}_p(k)}{\lambda + \gamma_p(k+1) \mathbf{a}_p^T(k+1) \mathbf{P}_p(k) \mathbf{a}_p(k+1)} \right), \tag{5.46}$$

$$g_p(k+1) = \mathbf{P}_p(k+1) \mathbf{a}_p(k+1), \tag{5.47}$$

$$e_p(k+1) = \gamma_p(k+1) \left(y(k+1) - \mathbf{a}_p^T(k+1) \boldsymbol{\theta}_p(k) \right), \tag{5.48}$$

$$\boldsymbol{\theta}_p(k+1) = \boldsymbol{\theta}_p(k) + g_p(k+1) e_p(k+1). \tag{5.49}$$

The MULRM online learning by the EM algorithm is summarized in Algorithm 5.3. The parameter $\mathbf{P}_p(k)$ should be initialized as: $\mathbf{P}_p(k) = \varphi \mathbf{I}$, where $\mathbf{I}$ is the identity matrix with size $(n+1) \times (n+1)$, $(n+1)$ is the number of parameters of the individual model p of the mixture, and φ should be set as a large value. In the canonical MULRM case $\mathbf{P}_p(k) = \varphi \mathbf{I}_2$, where $\mathbf{I}_2$ is the identity matrix with size 2×2. Similarly to the offline case, the value of $\boldsymbol{\theta}_p$ in (5.49) can take inconsistent values when (5.46) and the gain (5.47) take large values, when $\mathbf{P}_p$ becomes large. If the value of (5.47) is very large, then the values of $\boldsymbol{\theta}_p$ and $\mathbf{P}_p$ are set to their initial

Algorithm 5.3 Online learning of the MULRM

1: **Input:** sample $(\mathbf{x}(k+1), y(k+1))$;
2: **Output:** $\boldsymbol{\vartheta}$;
3: **Initialization:** If it is the first iteration, let $\mathbf{P}_p(k) = \varphi\mathbf{I}$, where φ should be set as a large value; And let $\boldsymbol{\vartheta}$ be equal to some initial $\boldsymbol{\vartheta}^{\text{old}}$;
4: If it is not the first iteration $\mathbf{P}_p(k)$ and $\boldsymbol{\vartheta}^{\text{old}}$ are obtained from the previous iteration;
5: **E step:**
6: Compute the responsibilities (5.32) of sample $(k+1)$ using $\boldsymbol{\vartheta}^{\text{old}}$;
7: Compute the expectation of sample $(k+1)$, $Q_{\text{MULRM}}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})(k+1)$ using (5.33);
8: **M step:**
9: Update the values of $\boldsymbol{\vartheta}$ which maximize $Q(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}^{\text{old}})$:
   (i) For each model, update υ_p using (5.39);
   (ii) For each model, update ω_p using (5.42);
   (iii) For each model, update $\boldsymbol{\theta}_p$ using (5.46)-(5.49);
10: $\boldsymbol{\vartheta}^{\text{old}} \leftarrow \boldsymbol{\vartheta}$;
11: **return** updated parameters $\boldsymbol{\vartheta}$.

values. Setting $\boldsymbol{\theta}_p$ and $\mathbf{P}_p$ to their initial values does not affect the performance of the model, since from (5.21) and (5.39) υ_p is small in this situation, and thus the contribution of model p in (5.1), is also small.

The experimental results suggest that the online learning of the MULRM approximates to the offline solution for a static case, when $\lambda = 1$, for a large number of samples. Since the MULRM will be applied in real-time, then after some operating time a large number of samples will be under consideration.

5.6 Remarks on MULRM

5.6.1 Collinearity and MULRM

The objective of the proposed method is on prediction rather than explaining the underlying relationships among the variables, i.e. the MULRM is to be applied on prediction problems and not explanation ones. When working with explanation,

the problem of collinearity becomes a problem of understanding the relationships among the variables [Hocking, 2003], and when working with prediction, the problem becomes to predict an output given a random input. According with Hocking [2003], the collinearity among input variables is not necessarily harmful when working with prediction problems. Additive models are an example of prediction models which ignore the collinearity among the input variables, while providing good results in some applications [Buja *et al.*, 1989].

The proposed MULRM method can be applied for prediction in the presence or absence of collinearity in the input data. Also, the MULRM cannot identify the joint effect among the variables (i.e. interaction terms of input variables are not considered in the MULRM), since in the MULRM the effects of the input variables on the output variables are individually assessed. However, this model formulation facilitates estimation since each component of the model can be addressed separately using (5.34)-(5.36). A special case of the proposed MULRM occurs when $\gamma_p(i) = \frac{1}{k}$, $p = 1, \ldots, D$, $i = 1, \ldots, k$, which reduces to the LS solution in the case where the input variables are mutually independent. Moreover, it is not possible to assure the physical meanings for the slopes and intercepts in the univariate linear regression models, then we assume that the slopes and intercepts do not have physical meaning at all.

5.6.2 Dealing with Outliers

If a variable x_p is affected by an outlier at sample i, and such sample is used to update the parameters, the responsibility $\gamma_p^{\text{old}}(i)$ in (5.32) will take a small value. This happens because if the value of $x_p(i)$ is an outlier, the value of $\mathcal{N}\left(y(i)\middle|f_p(x_p(i), \boldsymbol{\theta}_p^{\text{old}}), \omega_p^{\text{old}}\right)$ in (5.32) will be small. This can affect the performance of MULRM whether variable x_p is relevant to predict the target or not. If x_p is relevant to predict the target and $\lambda < 1$ with $\gamma_p^{\text{old}}(i)$ being small, then, with the update for sample i, the parameters of model p, $\boldsymbol{\theta}_p$ (5.49), ω_p (5.40), and υ_p (5.39) will lose information learned from the previous samples and will not gain the information from the current sample, since it is an outlier. This can decrease the performance of the overall method. However, if variable x_p is not relevant, then the effect of the outlier on the overall performance will not be significant. Moreover, in any case, outlier detection is an essential step while building soft sensors, and this step is encouraged when applying the MULRM

model in real-time applications. See [Kadlec *et al.*, 2011] for a review about methods to deal with outlier detection in soft sensors applications.

5.6.3 Accuracy, Bias and Precision from the Measurement Point of View

All data-driven soft sensors, as well as the proposed method, are data dependent; i.e. they perform the task of being a sensor, based on the learning of a model using the gathered data of the process, represented by $\Phi = \{(\mathbf{x}(i), y(i)); i = 1, \ldots, k\}$. The precision, accuracy and bias, of the soft sensor, from the measurement point of view, are directly related to the representativeness of Φ with respect to the future samples. After the data driven soft sensor is learned using Φ, and deployed for real operation, it is possible to affirm that the soft sensor is always precise, since the same input will always generate the same output. However, this does not mean that the soft sensor is accurate or not biased. In practice, due to the time-varying characteristics of most industrial processes, the soft sensor tends to deteriorate its accuracy over the time and also have the presence of bias. This happens because the data set Φ used for the learning is no longer representative when the time passes. However, this situation motivates the update of the soft sensor model with the most recent data, which is performed online on the proposed MULRM method, increasing its accuracy and reducing its bias. From the measurement point of view the sensor calibration of the soft sensor is done by updating the model using the most recent samples of the process.

5.6.4 Selecting λ Systematically

The value of λ can be fixed to a pre-specified value or its value can vary iterativelly in real-time by using the following gradient descent method proposed in [Anagnostopoulos *et al.*, 2008]:

$$\lambda_{k+1} = \lambda_k + c_\lambda \text{sign}\left(\frac{\partial \text{RSS}}{\partial \lambda_k}\right), \tag{5.50}$$

where c_λ is a small constant. In the above equation the value of λ is moved in the direction in which the move minimizes the residual sum of squares (RSS). If λ is to be fixed, then it can be selected by a K-fold cross-validation procedure applied on the training set.

Table 5.1: Summary of data sets.

	raw data set (before preprocessing)		data set after preprocessing			
Data set	**#Inputs**	**# Samples**	**#Inputs**	**#Samples**	**#Train**	**#Test**
Catalyst	15	5800	12	647	194	453
WTP	11	13152	11	1002	294	708

5.7 Experimental Results

This section presents experimental results of the proposed online MULRM applied in two time-varying real-world data sets. A summary of the data sets is given in Table 5.1. This summary consists on information of the data sets before and after it passes on a pre-processing stage. The pre-processing step is responsible for removing noisy variables (if any) and samples with non-existing output values. The training and testing of the adaptive models are carried out only after the data passes in the pre-processing stage. As the objective of the work is to evaluate the proposed methods, and not to discuss the process itself, only a short description of each process is given as follows:

1. Catalyst process [Kadlec and Gabrys, 2011]: The catalystic/polymerization data set is a benchmark for adaptive soft sensors introduced in [Kadlec and Gabrys, 2011, 2010], where the state of the art ensemble adaptive soft sensor method called ILLSA is also proposed. This data set describes a polymerization reactor. The objective is to predict the activity of the catalyst in the multitube. There are 15 input variables available, and some of them suffer from outliers, missing values, noise and automatic value interpolation by the data acquisition system. The data set covers 1 year of acquisition with 5800 available samples.

2. Water treatment plant (WTP): In this experiment the objective is to estimate the flour concentration in the effluent of a real-world WTP. The data set of plant variables that is available for learning consists of 11 input variables, and one target output variable to be estimated. The historical data set comprises of 3 years of acquisition, with 13512 data samples. The variables correspond to physical values, such as pH, turbidity, color of the water and others. Appendix

A presents the plant and the variables. The variables and respective time-lags selected in Chapter 3 were ignored in the experiments presented here. This is due the fact that the variable selection was ignored in the MULRM method. However, the results while selecting and not selecting the variables are quite similar. Thus, the fact of performing or not performing variables selection does not change or prejudice the conclusions taken here.

The proposed recursive MULRM method is compared with the RLS, and RPLS solutions of the multivariate linear model, and with the OS-ELM [Nan-Ying *et al.*, 2006] and ILLSA [Kadlec and Gabrys, 2011]. In all experiments, the values of both the training samples, and the testing samples, were normalized to be mean centered and with unit variance. More precisely: within all experiments, a same pre-processing linear transformation was performed on the values of both the training samples, and the testing samples. The parameters of this linear transformation were chosen such that the training samples were normalized to be mean centered, and with unit variance, using the information of the sample mean and sample variance of the training samples. Moreover, outliers detection was not considered. It has been assumed that data in both experiments were free of outliers.

In all data sets and tests, the normalized root mean square error (NRMSE), equation (3.32), was used as a performance measure to compare the results of the methods.

5.7.1 Evaluation and Discussion

In the experiments, in both data sets, the training data corresponds to 30% of available data and the remaining 70% was used for testing. All the methods (RLS, RPLS, ILLSA, OS-ELM, and MULRM) were tested for different percentages of availability of target data. Specifically, to verify the performance of the soft sensor with respect to the availability of the target data, it was tested for the cases where 0%, 10%, 25%, 50%, and 100% of the available target values were made available. In these situations, the two extreme cases (i.e., 0% and 100%) represent a non-adaptive/static scenario, and a scenario where all the target values are used for adaptation purposes, respectively. For evaluation purposes, if a new valid input-output pair $(\mathbf{x}(i), y(i))$ is available for update, then the output $y(i)$ will be first predicted using the input $\mathbf{x}(i)$, and then the model parameters will be updated. By

Table 5.2: Parameters selected for each model and for each data set.

Data set name	**RPLS**	**OS-ELM**	**ILSSA**
Catalyst	$M = 6$	$h = 5$	RF = 11, $\sigma = 10^{-3}$, $\sigma^{adapt} = 10^{-4}$.
WTP	$M = 4$	$h = 7$	RF = 11, $\sigma = 10^{-3}$, $\sigma^{adapt} = 10^{-6}$.

analyzing the performance of the models with different percentages of updates, it is possible to note the possibility of reducing, by using a soft sensor, the number of measurements of the hard to measure output variable that needs to be obtained by real-sensors or laboratory analysis. In all experiments, λ was set to be greater than or equal to 0.5. This was motivated by the physical meaning/interpretation of λ, since it is related with the effective number of samples. For example, assuming that the considered values of λ are less than 0.5, then d will lie between $1 \leq d < 2$ in the real interval, which does not have physical meaning or logic interpretation. Moreover, in our experiments, values of λ less than 0.5 did not provide any improvement to the experimental results. In all experiments, the canonical MULRM with component models $f_p(x_p)$ given by (5.26) was used.

The number of latent variables, M, used in the RPLS algorithm and the number of neurons, h, in the OS-ELM were respectively chosen by applying a 10-fold cross validation scheme on the training set and respectively selecting the number of latent variables and neurons which produced the smallest average error in all folds. Regarding the ILLSA algorithm, proposed in [Kadlec and Gabrys, 2011], for the Catalyst data set it was applied with the same parameters as in the original paper [Kadlec and Gabrys, 2011], and in the WTP data set the ILLSA parameters were found by a 10-fold cross-validation. Regarding the ILLSA parameters, RF represents the number of receptive fields and σ, σ^{adapt} are the parameters of kernel width and adaptation, respectively, (check [Kadlec and Gabrys, 2011] for more details on the meaning of the parameters). Table 5.2 shows the parameters obtained for each model and for each data set in the experiments.

Catalyst Data Set

This thesis follows the same pre-processing procedure done in [Kadlec and Gabrys, 2011]: downsampling of the first 5800 samples by a factor of 10, the removal of

variables 3, 4, 15, and removing all samples which have missing values. This pre-processing resulted in a data set with 647 data points, where 194 are used for offline training and the remaining 453 are used to simulate the online data. Moreover, the number of latent variables M used in the RPLS algorithm, the number of neurons h in the OS-ELM, and the parameters used in the ILLSA algorithm, are indicated in Table 5.2. In this data set, the degree of collinearity among the input variables, measured using the largest variance inflate factor (VIF) criterion, is equal to 91.19, which indicates a high degree of multicollinearity [Hocking, 2003; Belsley *et al.*, 2005].

The proposed MULRM method, and the RLS, RPLS and OS-ELM methods were applied for different values of λ (the corresponding value of $d = 1/(1-\lambda)$), and for different scenarios of availability of target data. The performance when using the adaptive forgetting factor (5.50), with $c_t = 0.01$, was also evaluated. The results are summarized in Table 5.3, and indicate that the proposed MULRM method has the best results in almost all scenarios. In the case where the availability of target data is 0% (Table 5.3a), the non-adaptive scenario, the performance of the proposed method reached a NRMSE value of 19%, the smallest when compared with the other methods, but still larger than 10%, which, as mentioned before in Chapter 2, is the threshold below which the values are acceptable in a practical application. Better results are reached with the increase of availability of target data. In the case where 100% of target data is used to update (Table 5.3e), the proposed MULRM method reached its smallest NRMSE value of 2.14% (with $\lambda = 0.50$), which is much better than the state-of-the-art methods. The results of the adaptive forgetting factor are quite good, taking into consideration the non necessity to select its value and to keep it fixed during the online operation. In the catalyst data set, the MULRM method seems to provide the best results when the values of λ, d, are small (which is the main motivation of the proposed approach), on the other hand, when $d = \infty$ (i.e. without forgetting the already learned data), the OS-ELM provides the best results, but they are not satisfactory. The prediction results of all models are exhibited in Figure 5.1.

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	70.41	79.89	24.99	**19.00**
$0.80, 5$	70.41	79.89	24.99	**19.00**
$0.95, 20$	70.41	79.89	24.99	**19.00**
$0.98, 50$	70.41	79.89	24.99	**19.00**
0.99,100	70.41	79.89	24.99	**19.00**
$1, \infty$	70.41	79.89	24.99	**19.00**
ILLSA: 28.74				

(a) 0% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	42.84	**11.65**	47.17	**11.65**
$0.80, 5$	26.17	13.85	14.02	**7.78**
$0.95, 20$	19.05	18.83	13.67	**10.62**
$0.98, 50$	22.63	23.60	13.14	**12.31**
0.99,100	24.19	26.80	14.44	**13.75**
$1, \infty$	25.91	24.59	**15.13**	16.28
Adap.	22.35	28.48	14.73	**13.03**
ILLSA: 9.47				

(b) 10% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	20.82	6.31	46.03	**4.54**
$0.80, 5$	20.83	9.91	16.14	**5.83**
$0.95, 20$	15.72	12.02	13.07	**8.31**
$0.98, 50$	18.31	17.16	12.87	**10.52**
0.99,100	20.19	19.23	12.26	**12.10**
$1, \infty$	22.91	22.98	**13.57**	16.54
Adap.	17.51	17.81	12.13	**10.21**
ILLSA: 8.94				

(c) 25% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	30.73	4.68	100.4	**2.90**
$0.80, 5$	10.17	5.92	16.16	**3.59**
$0.95, 20$	15.55	11.00	10.36	**6.71**
$0.98, 50$	14.80	12.10	11.31	**8.80**
0.99,100	16.38	17.07	12.24	**10.24**
$1, \infty$	20.66	20.55	**12.60**	16.28
Adap.	15.18	11.37	8.67	**6.88**
ILLSA: 6.81				

(d) 50% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	20.92	3.58	6.54	**2.14**
$0.80, 5$	7.85	4.14	10.29	**2.57**
$0.95, 20$	9.27	7.23	8.83	**5.12**
$0.98, 50$	12.02	11.48	8.75	**7.20**
0.99,100	12.26	12.17	10.15	**8.54**
$1, \infty$	18.31	18.64	**12.49**	15.38
Adap.	11.39	9.01	8.68	**6.40**
ILLSA: 5.51				

(e) 100% update

Table 5.3: NRMSE values on the Catalyst data set for different forgetting factors, λ, and different percentages of available target data.

WTP

In this data set, the degree of collinearity among the input variables, measured using the largest VIF value, is equal to 6.19, which is considered an acceptable value for VIF [Hocking, 2003; Belsley *et al.*, 2005], and it indicates that the collinearity will not interfere in the LS solution.

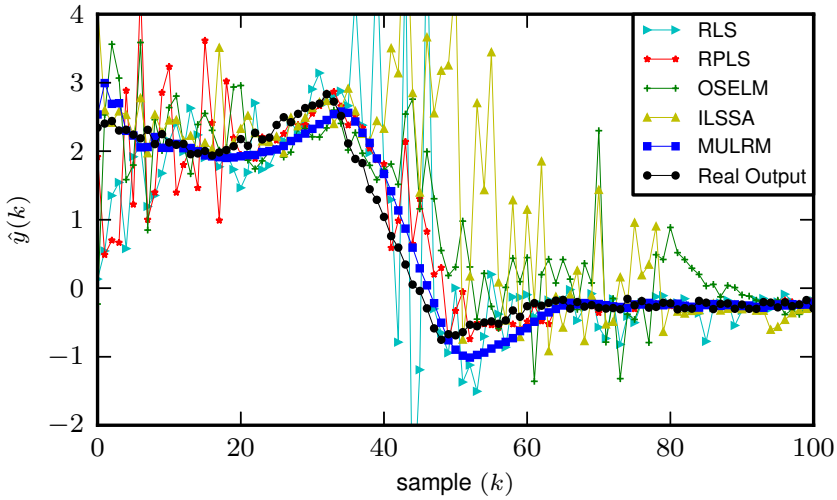


Figure 5.1: Activity of catalyst prediction output of all models for the first 100 samples. The frequency of updates is 100% for all models. The value of λ for each model was chosen based on the best results of the respective model in Table 5.3

The methodology used in the WTP experiment was the same as the one used in the Catalyst experiment. The available data set was split into 30% for training, and the remaining 70% of data was used to simulate the online data, and it is delivered as a stream of samples. The historical data set comprises 3 years of acquisition, with 13512 data samples, with a sample rate of 2 [h], for the variables acquired by sensors (input variables). The target variable, the fluoride, is laboratory measured at every 24 [h]. The samples with missing fluoride data were removed, resulting in a data set with 1002 samples, where 294 were used for training, and the remaining 708 were used to simulate the online data.

The number of latent variables for the RPLS model, and the number of neurons in the OS-ELM model, and the ILLSA parameters are indicated in Table 5.2.

Table 5.4 shows the results on the WTP data set. In the non-adaptive scenario, 0% (Table 5.4a), the results of the proposed MULRM method are the best among all, with a NRMSE = 13.43%. Despite these results, the values of NRMSE are still larger than 10%, which are not acceptable values. The results of all methods get better with the increase in the number/frequency of updates, and all methods reach their minimum NRMSE when 100% (Table 5.4e) of target data is used to update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	24.55	23.63	16.43	**13.43**
$0.80, 5$	24.55	23.63	16.43	**13.43**
$0.95, 20$	24.55	23.63	16.43	**13.43**
$0.98, 50$	24.55	23.63	16.43	**13.43**
0.99,100	24.55	23.63	16.43	**13.43**
$1, \infty$	24.55	23.63	16.43	**13.43**
ILLSA: 14.59				

(a) 0% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	46.83	10.18	17.31	**9.40**
$0.80, 5$	19.83	11.39	10.51	**9.65**
$0.95, 20$	19.24	16.38	**10.53**	11.60
$0.98, 50$	20.06	18.06	**11.01**	12.48
0.99,100	20.43	19.58	**10.74**	12.86
$1, \infty$	20.83	20.40	**11.20**	13.30
Adap.	13.08	15.12	12.70	**11.73**
ILLSA: 11.30				

(b) 10% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	30.63	8.77	16.10	**9.93**
$0.80, 5$	18.49	9.47	10.54	**8.22**
$0.95, 20$	12.57	11.43	9.90	**9.51**
$0.98, 50$	15.39	15.26	**10.28**	10.92
0.99,100	16.37	15.20	**10.79**	11.77
$1, \infty$	17.51	17.39	**11.25**	12.90
Adap.	10.90	10.99	10.67	**9.02**
ILLSA: 10.01				

(c) 25% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	∞	**7.91**	19.28	30.73
$0.80, 5$	13.89	8.33	9.80	**7.66**
$0.95, 20$	9.12	9.07	8.74	**8.59**
$0.98, 50$	10.74	10.53	**9.45**	9.62
0.99,100	12.34	11.83	**10.09**	10.66
$1, \infty$	14.41	17.52	**11.04**	12.53
Adap.	10.44	9.79	9.71	**8.60**
ILLSA: 9.18				

(d) 50% update

λ, d	**RLS**	**RPLS**	**OS-ELM**	**MULRM**
$0.50, 2$	∞	**7.79**	38.40	∞
$0.80, 5$	∞	8.00	9.85	**7.05**
$0.95, 20$	8.62	8.06	8.37	**7.77**
$0.98, 50$	9.59	9.50	8.85	**8.68**
0.99,100	9.59	10.06	9.69	**9.49**
$1, \infty$	12.33	14.71	**11.03**	12.28
Adap.	8.71	9.26	12.83	**7.47**
ILLSA: 9.60				

(e) 100% update

Table 5.4: NRMSE values on the WTP data set for different forgetting factors, λ, and different percentages of target data.

the model, and the proposed method reaches a NRMSE of 7.05 with $\lambda = 0.80$.

For all percentages of availability, when $d \leq 20$, the best results are obtained by the MULRM method, in most of the cases. When $d > 20$, and for updates between 10% and 50%, OS-ELM is the method with the best results. It is important to also note that, motivated by the small values of VIF, the LS solution provides good results in some experiments, which in some cases is even better than the PLS regression.

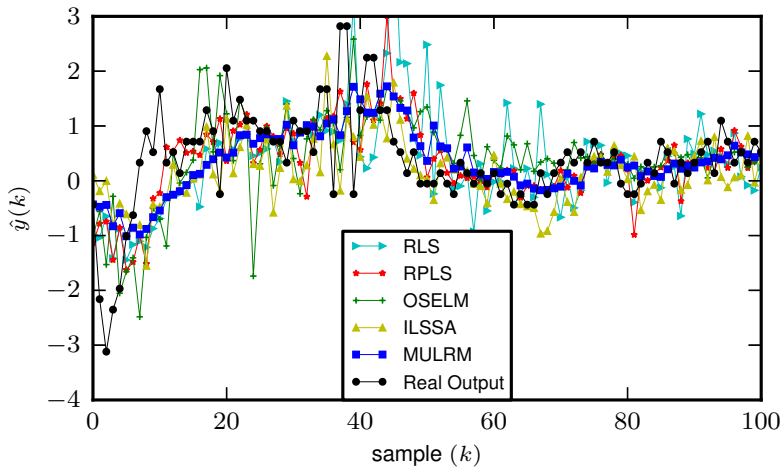


Figure 5.2: Fluoride prediction output of all models for the first 100 samples. The frequency of updates is 100% for all models. The value of λ for each model was chosen based on the best results of the respective model in Table 5.4.

In the WTP experiment, the MULRM produced large values of NRMSE when $\lambda = 0.5$, $d = 2$, in the cases where 50% (Table 5.4d) or 100% of data are used to update the model. This happened because some variables in the WTP data set are very noisy, and with frequencies of updates of 50% and 100% more samples of these variables are used in the updates. Thus, in these scenarios it is not advisable to use $d = 2$, since it can lead to the learn of noise. Values of $\lambda > 0.8$, and $d > 5$, seem to be more appropriate. Another alternative to set λ is to use the adaptive forgetting factor methodology discussed in Section 5.6.4, which provided good results in this data set.

The proposed method also reaches acceptable values of NRMSE for 25% and 50% of target data available for update. This suggests that the frequency of laboratory measurements can be reduced by a half or less, if this soft sensor is going to be applied, reducing the costs associated with the laboratory measurements. The predictions of all models are exhibited in Figure 5.2, validating and showing the effectiveness of the proposed method to perform prediction in time-varying environments.

5.8 Discussion

Both data sets are time-varying real-world data sets. Then, to track the time-varying parameters of each model a forgetting factor was used in the recursive learning of the parameters. For both data sets, it is possible to see that the best results, in almost all models (the OS-ELM seems to be the most constant among the other models), are achieved when $d = 2$, $d = 5$, and $d = 20$ (small values of λ); i.e. the results mean that the best performances are achieved when the most recent samples are used to compose the learning of the models parameters. Moreover, the best prediction performance achieved by the proposed MULRM method, is in general better when using small values of (λ, d). In both data sets, Catalyst and WTP, the proposed MULRM achieved the best results in almost all scenarios where the models were evaluated. Additionally, a major advantage of the proposed method when comparing with the PLS, ILSSA, and OS-ELM, is the necessity to tune few parameters on MULRM (only one parameter, the forgetting factor λ in its online version and no parameter in the offline version). For example, in the PLS it is necessary to tune the number M of latent variables and also λ in its online learning variant, for the ILLSA it is necessary to tune the parameters RF, σ, and σ^{adapt} (and it takes a long time, even for a small data set), and for the OS-ELM it is necessary to tune the number of hidden nodes h and also λ in its online learning variant. This advantage favors the MULRM in real-world applications.

Even under these characteristics of providing good results with small values of λ, it is still necessary to set a fixed value for λ, and this may be a challenge in real applications. For example, in the WTP problem, a small value of $\lambda = 0.5$ lead to noise learning, and provided unsatisfactory results. An alternative is to use the adaptive forgeting factor methodology discussed in Section 5.6.4. The adaptive forgetting factor demonstrated to provide good results, and seems to be a valid approach for real-world solutions. However, although the experimental results are representative, they are also conditioned by the problems under study, i.e. it is not possible to assure that they are general for all other conceivable problems. Nevertheless, MULRM can be a good option for soft sensor applications in time-varying scenarios. In some experiments, the proposed method still does not satisfy the requirement for NRMSE $< 10\%$. This happens mainly when the updating of the model does not occur (0%) or when the frequency of model updating is low (10%

and 25%).

The advantage of MULRM on the presented experiments, in comparison with the other methods, is mainly on the prediction performance, since the execution times for all methods are similar (except ILLSA, which is much more time demanding). The major drawback of MULRM is that it cannot identify the joint effect among the variables, thus it cannot be used as an explanation method.

5.9 Conclusions

This chapter proposed the use of a mixture of univariate linear models (in the canonical case) for adaptive regression, in a new method called MULRM. The method can also use a mixture of nonlinear models, but linear in the parameters. The formulas for the offline and online learning were derived based on the EM algorithm. Furthermore, in this work the proposed method has been evaluated and compared with the current state of art methods on two real-world data sets.

On the polymerization data set, the proposed MULRM method was compared with the RLS, RPLS, OS-ELM, and ILLSA methods, with better results for MULRM in the almost all experiments. On the WTP data set, the performance of the proposed method was much better, in the cases where $d < D$, when compared with the other state of the art RLS, RPLS, OS-ELM, and ILLSA methods. Moreover, the application of the proposed method on the WTP plant, to predict the fluoride, can allow the reduction of costs associated with the laboratory analysis, since it has been verified that the rate of model updates can be reduced by a half or more, while still attaining good prediction results.

Chapter 6

Conclusions and Discussion

The soft sensor technology has important potential for industrial applications and academic research. From the industry perspective, the soft sensor has an enormous potential to be used as a commercial tool to improve performance, efficiency, automation degree, and output quality in industrial systems. From the academy/research perspective, the soft sensors can be stated as a multidisciplinary topic of research, that encompasses several areas of study, such as machine learning, pattern recognition, artificial intelligence, system identification, and statistical learning theory. Moreover, it has several topics to be researched, such as the ones discussed in Chapter 2, where the most emergent topics, from the perspective of the author of this thesis, are the problem of variable selection (including dynamic selection) and soft sensor maintenance. Another topic of research, which corresponds to the area of one of the contributions of this thesis, is regarding the learning of soft sensor models in multiple operating scenarios/modes.

In Chapter 3 of this thesis, the problem of variable selection in nonlinear regression settings was addressed. A variable selection algorithm, based on the MLP model (the most common nonlinear model used in soft sensor applications) was proposed. The proposed variable selection algorithm is a fast implementation of the SBS-MLP procedure ($\sim$10-200 faster), using the error of the MLP model as the metric to judge the quality of a subset o variables. The proposed method has shown to have comparable results with the SBS-MLP and superior performance than other three state of art methods. The number of inputs in the experiments ranged from 6 to 55.

Future research on the topic of variable selection in soft sensors, can include, but is not limited to:

- Comparison of several variable selection algorithms, on soft sensors' data sets with different numbers of inputs and numbers of samples. An interesting topic is to predict the performance of each variable selection method on different problem domains/data sets, by using only the information contained on data (e.g. mean, number of inputs, number of samples, etc.), and not requiring the application of the variable selection method. Such method can be a valuable help while defining the best variable selection method to be used on the problem under study;

- Selection of time-lags/dynamics of input variables;

- Online/adaptive variable selection.

In Chapter 4, a method for model learning in processes with multiple operating modes was proposed. To this end, a PLS model was inserted into the ME framework, to derive the Mix-PLS method. The derivation of Mix-PLS was fully detailed in Chapter 4. First the ME framework was detailed/explained and then the PLS algorithm was used to learn the experts and gates parameters. The Mix-PLS has shown to be much less prone to overfitting when compared with the MLRE. In the experimental part, two real-world data sets with multiple operating modes, and one real-world data set without multiple operating modes, were considered. The Mix-PLS has shown to provide excellent results in the prediction performance and in the ability to detect multiple operating modes. The Mix-PLS can also be used as a non-linear regression model, similarly to MLP and SVR models. The detailed derivation of Mix-PLS opens the way for future further research in multiple operating modes learning.

Future research directions on the topic of model learning, more specifically in the case of multiple operating scenarios, could be follows:

- Evaluate the capability of different linear and non-linear models, for predicting multiple operating modes/scenarios, by integrating them with the ME framework, similarly to what was done with the PLS algorithm;

- Develop a method to automatically detect the number of experts to be used; perhaps using the same idea of [Figueiredo and Jain, 2002] that integrates the model parameters learning and model selection learning in a single algorithm;
- Methods of variable selection in multiple operating scenarios;
- Adaptive/online learning of the Mix-PLS.

In Chapter 5, the problem of learning adaptive soft sensors was studied. The study included the problem related with exponential weighting of samples in adaptive soft sensors. It was assumed that when learning the adaptive models with small values of forgetting factor, the model suffers from problems similar to the ones associated with learning of static models with small number of samples. Then, based on this, a mixture of low dimensional models was proposed and derived, based on the mixture of univariate linear regression models. Mixtures of other types of models, possibly nonlinear, but linear in the parameters were also considered. The proposed method was evaluated in two time-varying real-world data sets, and compared in different settings with the state of the art methods in adaptive soft sensors. The proposed method demonstrated to provide the best results in almost all cases, mainly when using small values of forgetting factor.

Future research on the topic of adaptive model learning, more specifically related to the problem addressed in this thesis are:

- Derive new methods to work efficiently in adaptive scenarios, even with small values of forgetting factor;
- Derive new recursive learning methods for linear and non-linear models, such as LASSO, RR, EN, MLP, SVR, etc.
- A generic framework for adaptive soft sensors.

In this thesis, different novel soft sensor methodologies have been proposed in Chapters 3 to 5. Each of these methodologies covers different aspects of soft sensor development, which are, variable selection, model learning, and soft sensor maintenance. The methods proposed in each of these chapters, do not have any connection among them, i.e. they are independent of each other, and they serve for different purposes. If the soft sensor setting has a large number of input variables, and it

requires a nonlinear model, such as the MLP model, then the method proposed in Chapter 3 should be suitable for application. Moreover, it can also be applied to determine the most relevant variables in a nonlinear prediction settings. On the other hand, if the process has multiple operating modes, one can use the method proposed in Chapter 4, which also deals efficiently with learning in nonlinear prediction settings. If the problem at hands is to be applied in time-varying settings, then the method proposed in Chapter 5 is a good choice.

As can be noticed, there are different possible types of settings in the soft sensors area, and each of these settings brings different challenges to deal with. In this thesis, some of such problems have been placed in evidence and new methods have been proposed and developed. All the proposed methods have been compared with the state of the art and their performances were superior in these comparisons.

Appendix A

Water Treatment Plant

This plant is intended to treat the water incoming into the water treatment plant (WTP) station and the objective is to predict the amount of fluoride in the effluent. In this plant, the following steps are taken to treat the water: Pre-treatment, Coagulation, Flocculation, Sedimentation, Filtration and Disinfection. A short summary about the process is given as follows.

Raw water is pre-treated prior to the main processes within the WTP. The Pre-treatments done in the plant are the algae control, and a treatment to remove metals such as manganese and iron, where the later is done through the addition of chlorine. The Coagulation and Flocculation stages are designed to help the removal of dissolved and suspended particles, causing water clarification (i.e. removal of turbidity). During Coagulation, chemical coagulants are added to raw water aiming to neutralize the electrical charges of the fine particles present in the water. After Coagulation, the water is gentle mixed during the Flocculation stage, facilitating the agglomeration of fine particles, so generating flocs which can then be easily removed in the subsequent stages. The Sedimentation stage prepares the water for effective filtration, by allowing the flocs to settle by gravity. After sedimentation, only small unsettled particles remain in the water, and are then removed in the Filtration stage. In Filtration, the suspended particles from water, and micro-organisms in general, are removed by passing the water through a filter, such as sand. As the water passes through the filter, flocs and impurities get stuck in it and the clean water goes through. The clean water from the Filtration stage is then treated with chlorine in the Disinfection stage. After the last stage, the amount of fluoride in the

Table A.1: Variables of the water treatment plant dataset.

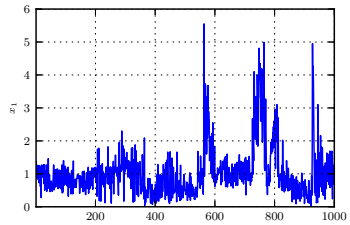
Variable	Description
x_1	Chlorine in the raw water;
x_2	Chlorine in the effluent;
x_3	Turbidity in the raw water;
x_4	Turbidity in the coagulated water;
x_5	Turbidity in the effluent;
x_6	pH in the raw water;
x_7	pH in the coagulated water;
x_8	pH in the effluent;
x_9	Color in the raw water;
x_{10}	Color in the coagulated water;
x_{11}	Color in the effluent;
y	Fluoride in the effluent.

water is determined. The value of concentration of fluoride in the effluent water is necessary to proceed with its correction (which is done by adding more fluoride into the water; this procedure is known as fluoridation).

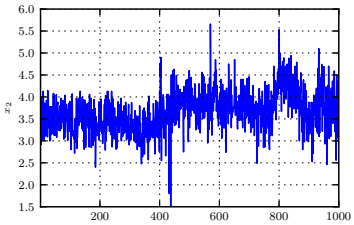
The fluoride is a normal constituent of natural water samples and its concentration in the input of WTP, on raw water, is constant. However, during the water treatment process, the concentration of fluoride in the water decreases, which is caused by the cleaning process. The value of fluoride in the effluent is measured in laboratory once every day, and the objective of the methodology proposed in this thesis is to provide the fluoride concentration value at each 2 hours using a soft-sensor.

The dataset of plant variables that is available for learning consists of 11 input variables, $X = \{x_1, \ldots, x_{11}\}$, and one target output variable to be estimated, y. The variables correspond to physical values, such as pH, turbidity, color of the water and others. Table A.1 presents further details about the variables. Figure A.1 shows the plots of variables described in Table A.1.

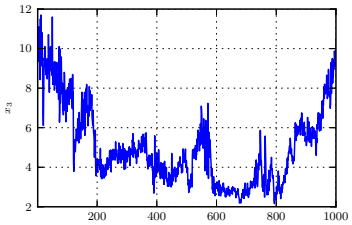
This data set was used in Chapters 3 and 5. In Chapter 3, only one year of acquisiton was considered, since it has a quite time invariant behavior. The problem investigated in Chapter 3 with respect to this data set is the problem of variable



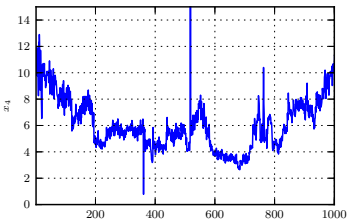
(a) x_1: chlorine in the raw water



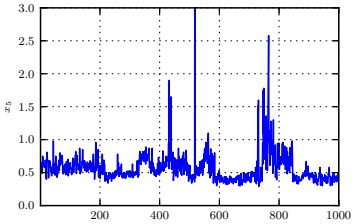
(b) x_2: chlorine in the effluent



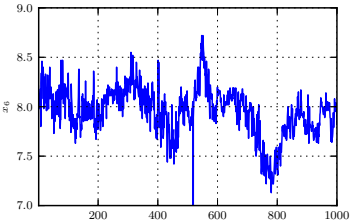
(c) x_3: turbidity in the raw water



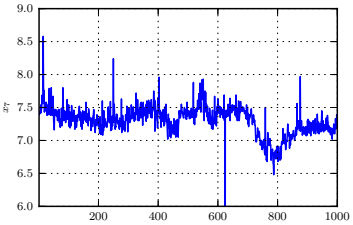
(d) x_4: turbidity in the coagulated water



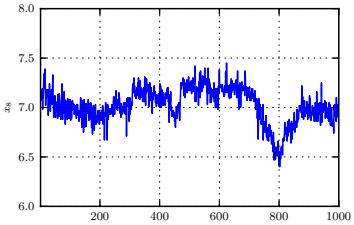
(e) x_5: turbidity in the effluent



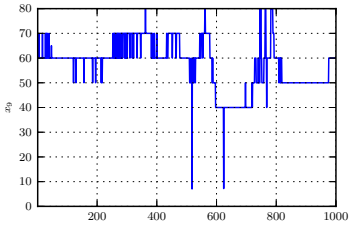
(f) x_6: pH in the raw water



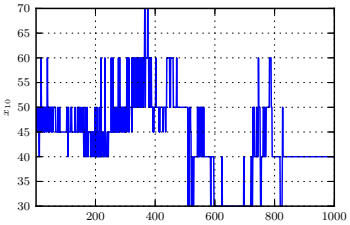
(g) x_7: pH in the coagulated water



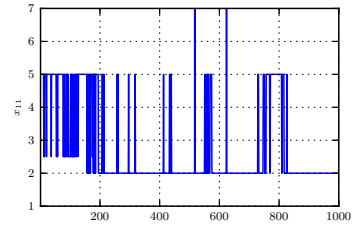
(h) x_8: ph in the effluent



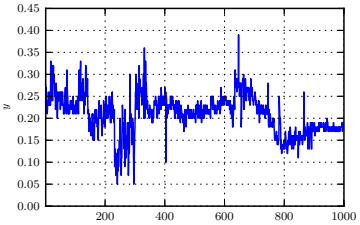
(i) x_9: color in the raw water



(j) x_{10}: color in the coagulated water



(k) x_{11}: color in the effluent



(l) y: fluoride in the effluent

Figure A.1: Plots of WTP variables.

selection in the WTP. This one-year of acquisition contains 352 samples. The time-variant behavior is more accentuated when all the three years of acquisition are considered. The 3-years data set was used in Chapter 5, and this long data set has a time-variant behavior. The total number of samples in the 3-years dataset is 1002 samples. The objective in Chapter 5 was the prediction of fluoride in the tree years.

Appendix B

Search Procedures

In a variable selection algorithm, a search procedure is used to guide the search for the best subset of variables. For D input variables, there are a total of $2^D - 1$ possible subsets, where only some of the subsets attain the optimal solution. Typically, the optimal solution may be attained for only one of the subsets. By searching over all possible subsets (this is called as exhaustive search), it is possible to lead to the optimal solution. However, for exhaustive search there is the problem of the large computational demand. For example if there are only 20 variables, i.e. $D = 20$, there are 1048575 solutions that need to be evaluated, if the criterion to evaluate one subset takes approximately ~ 1(sec) (being optimistic), then it would be necessary ~ 12 days to select the best subset. The branch and bound (B&B) algorithm leads to the optimal solution with less complexity than the exhaustive search, under the constrain that the evaluation function must be monotic [Narendra and Fukunaga, 1977]. However, the algorithm still has an exponential worst case complexity, which may render the approach infeasible when a large number of candidate variables is available [Guyon, 2003].

The large computational costs associated with the exhaustive search and B&B algorithms, caused by the necessity to evaluate so many subsets, can be reduced by using search strategies that prioritize the computational time rather than the quality of the solution, while still providing good results. Such strategies are based on rankers, sequential and stochastic searches. These techniques are briefly reviewed below.

B.1 Ranking

The ranking search proceeds as follows. First, the importance of each input variable, with respect to the target (measured by any criterion, e.g. CC, MI), is computed. Then, the variables are ranked according to their individual merit, with respect to the target variable, in accordance with the chosen criterion. Then, only a subset of the top variables (from the ranked set), are selected, and the remaining variables are excluded. In this search approach only D evaluations are required; a very fast approach. This method gains on the speed of selection, but loses on the quality of the selected variables. This happens because, the variables are selected without taking into consideration the interaction among them.

B.2 Sequential

The sequential search works by removing or adding variables sequentially, following a certain order. The most common sequential search procedures are the sequential forward selection (SFS) and the sequential backward selection (SBS). The SBS procedure, proposed by [Marill and Green, 1963], starts with all variables, and at each step the variable that contributes least to predict the target, according with the subset evaluation criterion, is removed. The SBS procedure stops when a prespecified number of variables are removed or until the results get satisfactory. The SFS, introduced by [Whitney, 1971], starts with an empty subset, and at each step the variable that mostly contributes to predict the target, according with the subset evaluation criterion, is added to the set of selected variables. These methods are largely used in variable selection procedures.

Both SFS and SBS have the same complexity in the worst case scenario (it is necessary to evaluate $\frac{D(D+1)}{2}$ subsets), but in a practical perspective the SFS executes faster than SBS. This happens because the SFS algorithm evaluates smaller subsets than the SBS at the beginning of the search.

The major problem related to the SFS and SBS approaches is that, for example, when a variable is removed in SBS, it cannot be selected again. This results in the so called *nesting effect*, i.e. bad decisions made at the beginning of the search cannot be corrected later. To avoid or alleviate the nesting effect in the sequential selection Stearns [1976] proposed the Plus-l-Minus-r search method. Each iteration of the

Plus-l-Minus-r is divided into two substeps. In the first step, the SFS runs to select l new variables, and in the second step the SBS runs to exclude r variables from those that have already been selected. Pudil *et al.* [1994] proposed modifications on the SFS and SBS to allow them to reselect removed variables, then avoiding the nesting effect, they are called as sequential forward floating selection (SFFS) and sequential backward floating selection (SBFS), and their idea is similar to the Plus-l-Minus-r algorithm.

B.3 Stochastic

Stochastic methods are optimization methods which include some randomness in the search procedure. This can be thought as a good strategy when dealing with a large number of input variables [Kudo and Sklansky, 2000], since it corresponds to search randomly over the input space, but following a certain heuristic. The class of stochastic algorithms includes, but is not restricted to, Genetic Algorithms (GA), Ant Colony Optimization (ACO), and Simulated Annealing (SA).

The GA is inspired by the biological evolution, more specifically by the Darwinian principles of natural evolution, where the best individuals have a high probability of survival; It was first introduced in [Holland, 1992]. In the GA, solutions are encoded into chromosomes (individuals) and the fittest ones are more susceptible, have higher probability, to be selected for reproduction, producing offspring with characteristics of both parents. For some of the offsprings an operation called mutation (inspired by the natural evolution) is applied, to include diversity in the solution.

The ACO is an optimization methodology based on ant behaviors to establish the shortest route paths from their colony to food sources and back [Dorigo *et al.*, 1996]. In nature, ants randomly walk for finding food, then they return to their colony while laying down pheromone trails. Other ants, when finding such path, tend to follow the trail and when they find food, they also walk back to the colony laying down pheromone, thus reinforcing the trail.

SA is a meta-heuristic proposed in [Kirkpatrick *et al.*, 1983] for global optimization problems. SA is inspired in the behavior of a warm particle in a potential field. Generally, a particle tends to move down, to the lower potential energy, but since it has kinect energy (caused by the non-zero temperature), it moves around with some randomness, and occasionally it jumps to higher potentials. The particle is an-

nealed when the time passes in this process, i.e. if temperature decreases gradually, so that the probability to move upwards decreases with time. In SA, the solution is represented by the particle and the potential energy represents the cost function.

Bibliography

[Akaike, 1974] H. Akaike. A New Look at the Statistical Model Identification. *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, December 1974. (Cited in pages 26, and 58).

[Alexandros and Melanie, 2001] Kalousis Alexandros and Hilario Melanie. Model Selection Via Meta-Learning: a Comparative Study. *International Journal on Artificial Intelligence Tools*, vol. 10, no. 4, pp. 525–554, 2001. (No citations).

[Anagnostopoulos *et al.*, 2008] Christoforos Anagnostopoulos, Dimitris Tasoulis, David J. Hand, and Niall M. Adams. Online optimization for variable selection in data streams. In: *Proceedings of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pp. 132–136. IOS Press, Amsterdam, The Netherlands, 2008. (Cited in page 104).

[Andridge and Little, 2010] Rebecca R. Andridge and Roderick J. A. Little. A Review of Hot Deck Imputation for Survey Non-Response. *International Statistical Review*, vol. 78, no. 1, pp. 40–64, April 2010. (Cited in page 12).

[Angelov and Kordon, 2010] P. Angelov and A. Kordon. Adaptive Inferential Sensors Based on Evolving Fuzzy Models. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 40, no. 2, pp. 529–539, April 2010. (Cited in page 27).

[Arakawa *et al.*, 2011] Masamoto Arakawa, Yosuke Yamashita, and Kimito Funatsu. Genetic Algorithm-Based Wavelength Selection Method for Spectral Calibration. *Journal of Chemometrics*, vol. 25, no. 1, pp. 10–19, January 2011. (Cited in page 19).

[Balagani and Phoha, 2010] Kiran S. Balagani and Vir V. Phoha. On the Feature Selection Criterion Based on an Approximation of Multidimensional Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1342–1343, July 2010. (Cited in page 18).

[Battiti, 1994] Roberto Battiti. Using Mutual Information for Selecting Features in Supervised Neural Net Learning. *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, July 1994. (Cited in page 18).

[Beirlant *et al.*, 1997] J. Beirlant, E. J. Dudewicz, L. Gyorfi, and E. C. z van der Meulen. Nonparametric Entropy Estimation: An Overview. *International Journal of Mathematical and Statistical Sciences*, vol. 6, pp. 17–39, 1997. (Cited in page 17).

[Bella *et al.*, 2007] A. Di Bella, L. Fortuna, S. Graziani, G. Napoli, and M. G. Xibilia. A Comparative Analysis of the Influence of Methods for Outliers Detection on the Performance of Data Driven Models. In: *IEEE Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007*, pp. 1–5. 2007. (Cited in pages 13, and 14).

[Bellman, 1961] Richard Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, London, UK, 1961. (Cited in page 15).

[Belsley *et al.*, 2005] David A. Belsley, Edwin Kuh, and Roy E. Welsch. *Regression Diagnostics*, chap. Detecting and Assessing Collinearity, pp. 85–191. John Wiley & Sons, Inc., 2005. (Cited in pages 108, and 109).

[Ben-Gal, 2005] Irad Ben-Gal. Outlier detection. In: O. Maimon and L. Rockach (eds.), *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*, pp. 1–16. Kluwer Academic Publishers, 2005. (Cited in page 13).

[Ben-Israel and Greville, 2003] Adi Ben-Israel and Thomas N.E. Greville. *Generalized Inverses: Theory and Applications*. Springer-Verlag, New York, USA, second ed., 2003. (Cited in page 57).

[Bhartiya and Whiteley, 2001] Sharad Bhartiya and James R Whiteley. Development of Inferential Measurements Using Neural Networks. *ISA Transactions*, vol. 40, no. 4, pp. 307–323, September 2001. (Cited in page 22).

[Billings *et al.*, 1992] S. A. Billings, H. B. Jamaluddin, and S. Chen. Properties of Neural Networks with Applications to Modelling Non-Linear Dynamical Systems. *International Journal of Control*, vol. 55, no. 1, pp. 193–224, 1992. (Cited in page 26).

[Bishop, 1995a] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Springer, 1995a. (No citations).

[Bishop, 1995b] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995b. (Cited in pages 16, 21, 33, and 34).

[Bishop, 2006] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 1 ed., 2006. (Cited in pages 10, 24, 60, 61, 86, 92, 95, and 97).

[Björck and Elfving, 1979] Åke Björck and Tommy Elfving. Accelerated Projection for Computing Pseudoinverse Solutions of Systems of Linear Equations. *BIT Numerical Mathematics*, vol. 19, no. 2, pp. 145–163, 1979. (Cited in page 40).

[Buja *et al.*, 1989] Andreas Buja, Trevor Hastie, and Robert Tibshirani. Linear Smoothers and Additive Models. *The Annals of Statistics*, vol. 17, no. 2, pp. 453–510, 1989. (Cited in page 103).

[Camacho and Picó, 2006] José Camacho and Jesús Picó. Online Monitoring of Batch Processes Using Multi-Phase Principal Component Analysis. *Journal of Process Control*, vol. 16, no. 10, pp. 1021–1035, December 2006. (Cited in pages 25, and 54).

[Castellano and Fanelli, 2000] Giovanna Castellano and Anna Maria Fanelli. Variable Selection Using Neural-Network Models. *Neurocomputing*, vol. 31, no. 1-4, pp. 1–13, March 2000. (Cited in pages 21, 33, 37, and 44).

[Cauwenberghs and Poggio, 2000] Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In: *Advances in Neural Information Processing Systems (NIPS'00)*, pp. 409–415. 2000. (Cited in page 27).

[Chapados and Bengio, 2001] N. Chapados and Y. Bengio. Input Decay: Simple and Effective Soft Variable Selection. In: *Proc. International Joint Conference on Neural Networks (IJCNN'01)*, vol. 2, pp. 1233–1237. 2001. (Cited in pages 21, and 33).

[Chatterjee and Bhattacherjee, 2011] Snehamoy Chatterjee and Ashis Bhattacherjee. Genetic Algorithms for Feature Selection of Image Analysis-Based Quality Monitoring Model: An Application to an Iron Mine. *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 786–795, August 2011. (Cited in pages 19, and 32).

[Choi and Park, 2001] Dong-Jin Choi and Heekyung Park. A Hybrid Artificial Neural Network as a Software Sensor for Optimal Control of a Wastewater Treatment Process. *Water Research*, vol. 35, no. 16, pp. 3959–3967, November 2001. (Cited in page 16).

[Chu *et al.*, 2004] Young-Hwan Chu, Young-Hak Lee, and Chonghun Han. Improved Quality Estimation and Knowledge Extraction in a Batch Process by Bootstrapping-Based Generalized Variable Selection. *Industrial & Engineering Chemistry Research*, vol. 43, no. 11, pp. 2680–2690, May 2004. (Cited in page 19).

[Cover and Thomas, 1991] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991. (Cited in page 17).

[Davies and Gather, 1993] L. Davies and U. Gather. The Identification of Multiple Outliers. *Journal of the American Statistical Association*, vol. 8, no. 423, pp. 782–792, September 1993. (Cited in page 13).

[Dayal and MacGregor, 1997] Bhupinder S. Dayal and John F. MacGregor. Recursive Exponentially Weighted PLS and Its Applications to Adaptive Control and Prediction. *Journal of Process Control*, vol. 7, no. 3, pp. 169–179, 1997. (Cited in pages 28, and 91).

[Delgado *et al.*, 2009] Myriam Regattieri Delgado, Elaine Yassue Nagai, and Lúcia Valéria Ramos Arruda. A Neuro-Coevolutionary Genetic Fuzzy System to Design Soft Sensors. *Soft Computing*, vol. 13, no. 5, pp. 481–495, March 2009. (Cited in pages 17, 23, and 24).

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977. (Cited in pages 55, 61, and 86).

[DeSarbo *et al.*, 1986] S. DeSarbo, Paul E. Green, and J. Douglas Carroll. An Alternating Least-Squares Procedure for Estimating Missing Preference Data in Product Concept Testing. *Decision Sciences*, vol. 17, no. 2, pp. 163–185, April 1986. (Cited in page 12).

[Dimopoulos *et al.*, 1999] Ioannis Dimopoulos, J. Chronopoulos, A. Chronopoulou-Sereli, and Sovan Lek. Neural Network Models to Study Relationships Between Lead Concentration in Grasses and Permanent Urban Descriptors in Athens City (Greece). *Ecological Modelling*, vol. 120, no. 2-3, pp. 157–165, August 1999. (Cited in pages 21, and 32).

[Dimopoulos *et al.*, 1995] Yannis Dimopoulos, Paul Bourret, and Sovan Lek. Use of Some Sensitivity Criteria for Choosing Networks With Good Generalization Ability. *Neural Processing Letters*, vol. 2, no. 6, pp. 1–4, 1995. (Cited in pages 21, and 32).

[Dorigo *et al.*, 1996] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26, no. 1, pp. 29–41, February 1996. (Cited in page 125).

[Duda *et al.*, 2000] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2 ed., 2000. (Cited in pages 10, and 25).

[Enders, 2001] Craig K. Enders. A Primer on Maximum Likelihood Algorithms Available for Use With Missing Data. *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 8, no. 1, pp. 128–141, 2001. (Cited in page 12).

[Eshghi, 2014] Peyman Eshghi. Dimensionality Choice in Principal Components Analysis Via Cross-Validatory Methods. *Chemometrics and Intelligent Laboratory Systems*, vol. 130, no. 0, pp. 6–13, January 2014. (Cited in page 16).

[Estévez *et al.*, 2009] Pablo A. Estévez, Michel Tesmer, Claudio A. Perez, and Jacek M. Zurada. Normalized Mutual Information Feature Selection. *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, February 2009. (Cited in pages 18, 33, and 44).

[Facco *et al.*, 2010] Pierantonio Facco, Fabrizio Bezzo, and Massimiliano Barolo. Nearest-Neighbor Method for the Automatic Maintenance of Multivariate Statistical Soft Sensors in Batch Processing. *Industrial & Engineering Chemistry Research*, vol. 49, no. 5, pp. 2336–2347, March 2010. (Cited in pages 28, and 85).

[Facco *et al.*, 2009] Pierantonio Facco, Franco Doplicher, Fabrizio Bezzo, and Massimiliano Barolo. Moving Average PLS Soft Sensor for Online Product Quality Estimation in an Industrial Batch Polymerization Process. *Journal of Process Control*, vol. 19, no. 3, pp. 520–529, March 2009. (Cited in pages 25, 54, 55, 73, and 77).

[Fahlman, 1988] Scott E. Fahlman. Faster Learning Variations on Back Propagation: An Empirical Study. In: *Proceedings of 1988 Connectionist Models Summer School*. Morgan-Kaufmann, 1988. (Cited in page 37).

[Figueiredo and Jain, 2002] Mario A. T. Figueiredo and Anil K. Jain. Unsupervised Learning of Finite Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 3, pp. 381–396, March 2002. (Cited in page 117).

[Fletcher and Reeves, 1964] R. Fletcher and C. M. Reeves. Function Minimization by Conjugate Gradients. *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964. (Cited in page 37).

[Fortuna *et al.*, 2006] Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo, and M. Gabriella Xibilia. *Soft Sensors for Monitoring and Control of Industrial Processes*. Advances in Industrial Control. Springer, 1 ed., December 2006. (Cited in pages 10, 13, 14, 17, 22, 23, 72, 73, and 77).

[Fortuna *et al.*, 2009] Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo, and M. Gabriella Xibilia. Comparison of Soft-Sensor Design Methods for Industrial Plants Using Small Data Sets. *IEEE Transactions on Instrumentation and Measurement*, vol. 58, no. 8, pp. 2444–2451, August 2009. (Cited in page 14).

[François *et al.*, 2007] D. François, F. Rossi, V. Wertz, and M. Verleysen. Resampling Methods for Parameter-Free and Robust Feature Selection with Mutual Information. *Neurocomputing*, vol. 70, pp. 1276–1288, March 2007. (Cited in page 17).

[Frank *et al.*, 2000] Eibe Frank, Leonard Trigg, Geoffrey Holmes, and Ian H. Witten. Technical Note: Naive Bayes for Regression. *Machine Learning*, vol. 41, no. 1, pp. 5–25, October 2000. (Cited in page 91).

[Frank and Friedman, 1993] Ildiko E. Frank and Jerome H. Friedman. A Statistical View of Some Chemometrics Regression Tools. *Technometrics*, vol. 35, no. 2, pp. 109–135, May 1993. (Cited in page 20).

[Frénay *et al.*, 2013] Benoît Frénay, Gauthier Doquire, and Michel Verleysen. Is mutual Information Adequate for Feature Selection in Regression? *Neural Networks*, vol. 48, no. 0, pp. 1–7, December 2013. (Cited in page 17).

[Friedman *et al.*, 2010] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010. (Cited in page 91).

[Friedman, 1991] Jerome H. Friedman. Multivariate Adaptive Regression Splines. *Annals os Statistics*, vol. 19, no. 1, pp. 1–67, 1991. (Cited in page 45).

[Fu *et al.*, 2008] Yongfeng Fu, Hongye Su, Ying Zhang, and Jian Chu. Adaptive Soft-Sensor Modeling Algorithm Based on FCMISVM and Its Application in PX Adsorption Separation Process. *Chinese Journal of Chemical Engineering*, vol. 16, no. 5, pp. 746–751, October 2008. (Cited in pages 29, and 85).

[Fujiwara *et al.*, 2012] Koichi Fujiwara, Hiroshi Sawada, and Manabu Kano. Input Variable Selection for PLS Modeling Using Nearest Correlation Spectral Clustering. *Chemometrics and Intelligent Laboratory Systems*, vol. 118, no. 0, pp. 109–119, August 2012. (Cited in page 18).

[Garson, 1991] G. David Garson. Interpreting Neural-Network Connection Weights. *AI Expert*, vol. 6, no. 4, pp. 46–51, April 1991. (Cited in pages 21, 32, and 42).

[Gevrey *et al.*, 2003] Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. Review and Comparison of Methods to Study the Contribution of Variables in Artificial Neural Network Models. *Ecological Modelling*, vol. 160, no. 3, pp. 249–264, February 2003. (Cited in pages 21, and 32).

[Gonzaga *et al.*, 2009] J. C. B. Gonzaga, L. A. C. Meleiro, C. Kiang, and R. Maciel Filho. ANN-based soft-sensor for real-time process monitoring and control of an industrial polymerization process. *Computers & Chemical Engineering*, vol. 33, no. 1, pp. 43–49, 2009. ISSN 0098-1354. (Cited in page 17).

[Grbić *et al.*, 2013] Ratko Grbić, Dražen Slišković, and Petr Kadlec. Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models. *Computers & Chemical Engineering*, vol. 58, no. 0, pp. 84–97, 2013. (Cited in pages 18, and 32).

[Guyon, 2003] Isabelle Guyon. An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003. (Cited in pages 16, 17, and 123).

[Guyon *et al.*, 2002] Isabelle Guyon, Jason Weston, Stephen Barnhill, and Vladimir Vapnik. Gene Selection for Cancer Classification using Support Vector Machines. *Machine Learning*, vol. 46, pp. 389–422, 2002. (Cited in page 22).

[Haavisto and Hyötyniemi, 2009] Olli Haavisto and Heikki Hyötyniemi. Recursive Multimodel Partial Least Squares Estimation of Mineral Flotation Slurry Contents Using Optical Reflectance Spectra. *Analytica Chimica Acta*, vol. 642, no. 1-2, pp. 102–109, 2009. Papers presented at the 11th International Conference on Chemometrics in Analytical Chemistry - CAC 2008. (Cited in pages 28, and 85).

[Hagan and Menhaj, 1994] Martin T. Hagan and Mohammad B. Menhaj. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, November 1994. (Cited in pages 37, and 44).

[Hastie *et al.*, 2001] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York, 2001. (Cited in pages 10, 12, 20, 24, and 34).

[Hawkins, 2004] Douglas M. Hawkins. The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, vol. 44, pp. 1–12, 2004. (Cited in page 58).

[Haykin, 1999] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. (Cited in pages 10, and 24).

[Helland *et al.*, 1992] Kristian Helland, Hans E. Berntsen, Odd S. Borgen, and Harald Martens. Recursive Algorithm for Partial Least Squares Regression. *Chemometrics and Intelligent Laboratory Systems*, vol. 14, no. 1-3, pp. 129–137, April 1992. (Cited in pages 28, and 85).

[Hocking, 2003] Ronald R. Hocking. *Methods and Applications of Linear Models: Regression and the Analysis of Variance*, chap. Collinearity in Multiple Linear Regression, pp. 151–192. John Wiley & Sons, 2003. (Cited in pages 103, 108, and 109).

[Hoerl and Kennard, 1970] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, vol. 12, no. 1, pp. 55–67, February 1970. (Cited in page 20).

[Holland, 1992] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, MA, USA, 1992. (Cited in page 125).

[Hornik *et al.*, 1989] Kur Hornik, Maxuel Stinchcombe, and Halber White. Multilayer Feedforward Networks are Universal Approximators. *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. (Cited in page 35).

[Jacobs *et al.*, 1991] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive Mixtures of Local Experts. *Neural Computation*, vol. 3, no. 1, pp. 79–87, March 1991. (Cited in pages 55, 60, and 61).

[Jacobs *et al.*, 1997] Robert A. Jacobs, Fengchun Peng, and Martin A. Tanner. A Bayesian Approach to Model Selection in Hierarchical Mixtures-of-Experts Architectures. *Neural Networks*, vol. 10, no. 2, pp. 231–241, March 1997. (Cited in page 72).

[Jang *et al.*, 1997] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall, 1 ed., September 1997. (Cited in pages 24, and 89).

[Jerez *et al.*, 2010] José M. Jerez, Ignacio Molina, Pedro J. García-Laencina, Emilio Alba, Nuria Ribelles, Miguel Martín, and Leonardo Franco. Missing Data Imputation Using Statistical and Machine Learning Methods in a Real Breast Cancer Problem. *Artificial Intelligence in Medicine*, vol. 50, no. 2, pp. 105–115, October 2010. (Cited in page 12).

[Jolliffe, 2002] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002. (Cited in pages 14, 15, and 24).

[Jordan, 1994] Michael I. Jordan. Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation*, vol. 6, no. 2, pp. 181–214, March 1994. (Cited in pages 55, and 67).

[Kadlec and Gabrys, 2010] Petr Kadlec and Bogdan Gabrys. Adaptive Online Prediction Soft Sensing Without Historical Data. In: *The 2010 International Joint Conference on Neural Networks (IJCNN'10)*, pp. 1–8. 2010. (Cited in page 105).

[Kadlec and Gabrys, 2011] Petr Kadlec and Bogdan Gabrys. Local Learning-Based Adaptive Soft Sensor for Catalyst Activation Prediction. *AIChE Journal*, vol. 57, no. 5, pp. 1288–1301, May 2011. (Cited in pages 11, 28, 29, 85, 87, 105, 106, and 107).

[Kadlec *et al.*, 2009] Petr Kadlec, Bogdan Gabrys, and Sibylle Strandt. Data-Driven Soft Sensors in the Process Industry. *Computers & Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009. (Cited in pages 10, 13, 14, 23, 27, and 31).

[Kadlec *et al.*, 2011] Petr Kadlec, Ratko Grbic, and Bogdan Gabrys. Review of Adaptation Mechanisms for Data-Driven Soft Sensors. *Computers & Chemical Engineering*, vol. 35, no. 1, pp. 1–24, 2011. (Cited in pages 26, 28, 84, and 104).

[Kalivas, 1997] John H. Kalivas. Two Data Sets of Near Infrared Spectra. *Chemometrics and Intelligent Laboratory Systems*, vol. 37, no. 2, pp. 255–259, June 1997. (Cited in pages 73, and 81).

[Kaneko and Funatsu, 2012] Hiromasa Kaneko and Kimito Funatsu. A New Process Variable and Dynamics Selection Method Based on a Genetic Algorithm-BasedWwavelength Selection Method. *AIChE Journal*, vol. 58, no. 6, pp. 1829–1840, June 2012. (Cited in page 19).

[Kaneko and Funatsu, 2013] Hiromasa Kaneko and Kimito Funatsu. Nonlinear Regression Method With Variable Region Selection and Application to Soft Sensors. *Chemometrics and Intelligent Laboratory Systems*, vol. 121, no. 0, pp. 26–32, February 2013. (Cited in page 19).

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, vol. 220, no. 4598, pp. 671–680, May 1983. (Cited in page 125).

[Kohavi, 1995] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: *Proc. 14th International Joint Conference on Artificial Intelligence*, vol. 2, pp. 1137–1143. Morgan Kaufmann, 1995. (Cited in page 45).

[Kohavi and John, 1997] Ron Kohavi and George H. John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, vol. 97, no. 1-2, pp. 273–324, December 1997. (Cited in page 16).

[Komulainen *et al.*, 2004] Tiina Komulainen, Mauri Sourander, and Sirkka Liisa Jämsä Jounela. An Online Application of Dynamic PLS to a Dearomatization Process. *Computers & Chemical Engineering*, vol. 28, no. 12, pp. 2611–2619, 2004. ISSN 0098-1354. (Cited in pages 28, and 85).

[Kramer and Braun, 2007] Nicole Kramer and Mikio L. Braun. Kernelizing PLS, Degrees of Freedom, and Efficient Model Selection. In: *Proc. 24th International Conference on Machine learning*, ICML'07, pp. 441–448. ACM, New York, NY, USA, 2007. (Cited in page 58).

[Kramer and Sugiyama, 2011] Nicole Kramer and Masashi Sugiyama. The Degrees of Freedom of Partial Least Squares Regression. *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 697–705, 2011. (Cited in pages 58, and 59).

[Kraskov *et al.*, 2004] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating Mutual Information. *Phys. Rev. E*, vol. 69, no. 6, p. 066138, June 2004. (Cited in page 17).

[Kudo and Sklansky, 2000] Mineichi Kudo and Jack Sklansky. Comparison of Algorithms that Select Features for Pattern Classifiers. *Pattern Recognition*, vol. 33, no. 1, pp. 25–41, January 2000. (Cited in page 125).

[Kuncheva and Žliobaitė, 2009] Ludmila I. Kuncheva and Indrė Žliobaitė. On the Window Size for Classification in Changing Environments. *Intelligent Data Analysis*, vol. 13, pp. 861–872, December 2009. (Cited in page 27).

[Lee *et al.*, 2005] Min W. Lee, Jea Y. Joung, Dae S. Lee, Jong M. Park, and Seung H. Woo. Application of a Moving-Window-Adaptive Neural Network to the Modeling of a Full-Scale Anaerobic Filter Process. *Industrial & Engineering Chemistry Research*, vol. 44, no. 11, pp. 3973–3982, May 2005. (Cited in pages 27, and 85).

[Lemaire and Féraud, 2006] Vincent Lemaire and Raphael Féraud. Driven Forward Features Selection: A Comparative Study on Neural Networks. In: Irwin King, Jun Wang, Lai-Wan Chan, and DeLiang Wang (eds.), *Neural Information Processing*, vol. 4233 of *Lecture Notes in Computer Science*, pp. 693–702. Springer Berlin Heidelberg, 2006. (Cited in pages 21, and 33).

[Li *et al.*, 2002] Baibing Li, Julian Morris, and Elaine B. Martin. Model Selection for Partial Least Squares Regression. *Chemometrics and Intelligent Laboratory Systems*, vol. 64, no. 1, pp. 79–89, October 2002. (Cited in page 58).

[Li *et al.*, 2005] C. Li, H. Ye, G. Wang, and J. Zhang. A Recursive Nonlinear PLS Algorithm for Adaptive Nonlinear Process Modeling. *Chemical Engineering & Technology*, vol. 28, pp. 141–152, February 2005. (Cited in pages 28, and 85).

[Lin *et al.*, 2007] Bao Lin, Bodil Recke, Jorgen K. H. Knudsen, and Sten Bay Jorgensen. A Systematic Approach for Soft Sensor Development. *Computers & Chemical Engineering*, vol. 31, no. 5-6, pp. 419–425, 2007. (Cited in page 16).

[Lin and Weng, 2004] Chih-Jen Lin and Ruby C. Weng. Simple Probabilistic Predictions for Support Vector Regression. Tech. rep., National Taiwan University, 2004. (Cited in page 22).

[Liu *et al.*, 2010] Guohai Liu, Dawei Zhou, Haixia Xu, and Congli Mei. Model Optimization of SVM for a Fermentation Soft Sensor. *Expert Systems with Applications*, vol. 37, no. 4, pp. 2708–2713, April 2010. (Cited in page 19).

[Liu *et al.*, 2004] Hancong Liu, Sirish Shah, and Wei Jiang. On-line outlier detection and data cleaning. *Computers & Chemical Engineering*, vol. 28, no. 9, pp. 1635–1647, 2004. (Cited in page 13).

[Liu *et al.*, 2000] Yong Liu, Xin Yao, and T. Higuchi. Evolutionary Ensembles With Negative Correlation Learning. *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, November 2000. (Cited in page 24).

[Liukkonen *et al.*, 2013] Mika Liukkonen, Eero Hälikkä, Teri Hiltunen, and Yrjö Hiltunen. Adaptive Soft Sensor for Fluidized Bed Quality: Applications to Combustion of Biomass. *Fuel Processing Technology*, vol. 105, pp. 46–51, January 2013. (Cited in pages 27, and 85).

[Ljung, 1999] Lennart Ljung. *System Identification: Theory for the User*. Prentice Hall, second ed., 1999. (Cited in pages 10, and 24).

[Ljung, 2010] Lennart Ljung. Perspectives on System Identification. *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, April 2010. (Cited in page 10).

[Lu and Gao, 2005] Ningyun Lu and Furong Gao. Stage-Based Process Analysis and Quality Prediction for Batch Processes. *Industrial & Engineering Chemistry Research*, vol. 44, no. 10, pp. 3547–3555, May 2005. (Cited in pages 25, and 54).

[Lu *et al.*, 2004] Ningyun Lu, Yi Yang, Furong Gao, and Fuli Wang. Multirate Dynamic Inferential Modeling for Multivariable Processes. *Chemical Engineering Science*, vol. 59, no. 4, pp. 855–864, February 2004. (Cited in page 11).

[Ludwig *et al.*, 2009] Oswaldo Ludwig, Urbano Nunes, Rui Araújo, Leizer Schnitman, and Herman Augusto Lepikson. Applications of Information Theory, Genetic

Algorithms, and Neural Models to Predict Oil Flow. *Communications in Nonlinear Science and Numerical Simulation*, vol. 14, no. 7, pp. 2870–2885, July 2009. (Cited in pages 17, 18, and 32).

[Ma *et al.*, 2009] Ming-Da Ma, Jing-Wei Ko, San-Jang Wang, Ming-Feng Wu, Shi-Shang Jang, Shyan-Shu Shieh, and David Shan-Hill Wong. Development of Adaptive Soft Sensor Based on Statistical Identification of Key Variables. *Control Engineering Practice*, vol. 17, no. 9, pp. 1026–1034, September 2009. (Cited in page 24).

[Macias-Hernandez *et al.*, 2007] J. J. Macias-Hernandez, P. Angelov, and Xiaowei Zhou. Soft Sensor for Predicting Crude Oil Distillation Side Streams Using Evolving Takagi-Sugeno Fuzzy Models. In: *Proc. IEEE International Conference on Systems, Man and Cybernetics*, pp. 3305–3310. 2007. (Cited in page 20).

[Mallows, 1973] C. L. Mallows. Comments on Cp. *Technometrics*, vol. 15, no. 4, pp. 661–675, November 1973. (Cited in pages 22, and 26).

[Marill and Green, 1963] T. Marill and D. Green. On the Effectiveness of Receptors in Recognition Systems. *IEEE Transactions on Information Theory*, vol. 9, no. 1, pp. 11–17, January 1963. (Cited in pages 33, 44, and 124).

[Matias *et al.*, 2013] Tiago Matias, Dulce Gabriel, Francisco Souza, Rui Araújo, and J. Costa Pereira. Fault Detection and Replacement of a Temperature Sensor in a Cement Rotary Kiln. In: *18th IEEE Conference on Emerging Technologies Factory Automation (ETFA 2013)*, pp. 1–8. 2013. (Cited in page 7).

[Matias *et al.*, 2014] Tiago Matias, Francisco Souza, Rui Araújo, and Carlos Henggeler Antunes. Learning of a Single-hidden Layer Feedforward Neural Network using an Optimized Extreme Learning Machine. *Neurocomputing*, vol. 129, pp. 428–436, April 2014. doi:10.1016/j.neucom.2013.09.016. (Cited in page 7).

[Matzopoulos, 2010] Mark Matzopoulos. Dynamic Process Modeling: Combining Models and Experimental Data to Solve Industrial Problems. In: Michael C. Georgiadis, Julio R. Banga, and Efstratios N. Pistikopoulos (eds.), *Process Systems Engineering*, pp. 1–33. Wiley-VCH Verlag GmbH & Co. KGaA, 2010. (Cited in pages 25, and 54).

[May *et al.*, 2011] Robert May, Graeme Dandy, and Holger Maier. Review of Input Variable Selection Methods for Artificial Neural Networks. In: Prof. Kenji Suzuki (ed.), *Artificial Neural Networks - Methodological Advances and Biomedical Applications*, pp. 19–44. InTech, 2011. (Cited in pages 20, and 32).

[Mendes *et al.*, 2010] Jérôme Mendes, Rui Araújo, and Francisco Souza. Adaptive Fuzzy Generalized Predictive Control Based on Discrete-Time T-S Fuzzy Model. In: *Proc. 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*. Bilbao, Spain, September 2010. (Cited in page 7).

[Mendes *et al.*, 2013] Jérôme Mendes, Rui Araújo, and Francisco Souza. Adaptive Fuzzy Identification and Predictive Control for Industrial Processes. *Expert Systems with Applications*, vol. 40, no. 17, pp. 6964–6975, 2013. (Cited in page 7).

[Mendes *et al.*, 2012a] Jérôme Mendes, Samuel Pinto, Rui Araújo, and Francisco Souza. Evolutionary Fuzzy Models for Nonlinear Identification. In: *17th IEEE Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–8. 2012a. (Cited in page 7).

[Mendes *et al.*, 2012b] Jérôme Mendes, Francisco Souza, Rui Araújo, and Nuno Gonçalves. Genetic Fuzzy System for Data-driven Soft Sensors. *Applied Soft Computing*, vol. 12, no. 10, pp. 3237–3245, 2012b. (Cited in pages 7, 24, and 27).

[Mevik and Cederkvist, 2004] Bjorn-Helge Mevik and Henrik René Cederkvist. Mean squared error of prediction (MSEP) estimates for principal component regression (PCR) and partial least squares regression (PLSR). *Journal of Chemometrics*, vol. 18, no. 9, pp. 422–429, 2004. (Cited in page 58).

[Montana and Davis, 1989] David J. Montana and Lawrence Davis. Training Feedforward Neural Networks Using Genetic Algorithms. In: *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pp. 762–767. 1989. (Cited in page 37).

[Mu *et al.*, 2006] Shengjing Mu, Yingzhi Zeng, Ruilan Liu, Ping Wu, Hongye Su, and Jian Chu. Online Dual Updating with Recursive PLS Model and its Application in Predicting Crystal Size of Purified Terephthalic Acid (PTA) Process.

Journal of Process Control, vol. 16, no. 6, pp. 557–566, May 2006. (Cited in pages 28, and 85).

[Muradore and Fiorini, 2012] Riccardo Muradore and Paolo Fiorini. A PLS-Based Statistical Approach for Fault Detection and Isolation of Robotic Manipulators. *IEEE Transactions on Industrial Electronics*, vol. 59, no. 8, pp. 3167–3175, August 2012. (Cited in pages 28, and 85).

[Nabney, 1999] Ian T. Nabney. Efficient Training of RBF Networks for Classification. In: *Proc. Ninth International Conference on Artificial Neural Networks, 1999 (ICANN 99)*, vol. 1, pp. 210–215. Edinburgh, Scotland, September 7-10 1999. (Cited in page 67).

[Nan-Ying *et al.*, 2006] Liang Nan-Ying, Huang Guang-Bin, P. Saratchandran, and N. Sundararajan. A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks. *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, November 2006. (Cited in pages 28, 86, and 106).

[Narendra and Fukunaga, 1977] Patrenahalli M. Narendra and Keinosuke Fukunaga. A Branch and Bound Algorithm for Feature Subset Selection. *IEEE Transactions on Computers*, vol. C-26, no. 9, pp. 917–922, September 1977. (Cited in page 123).

[Nelles, 2001] Oliver Nelles. *Nonlinear System Identification - From Classical Approaches to Neural Networks and Fuzzy Models.* Springer, 2001. (Cited in pages 15, 23, and 24).

[Newton and Raftery, 1994] Michael A. Newton and Adrian E. Raftery. Approximate Bayesian Inference with the Weighted Likelihood Bootstrap. *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 56, no. 1, pp. 3–48, 1994. (Cited in pages 65, and 66).

[Ng *et al.*, 2006] Shu-Kay Ng, Geoffrey J. McLachlan, and Andy H. Lee. An Incremental EM-Based Learning Approach for On-line Prediction of Hospital Resource Utilization. *Artificial Intelligence in Medicine*, vol. 36, no. 3, pp. 257–267, March 2006. (Cited in page 72).

[Nguyen and Widrow, 1990] Derrick Nguyen and Bernard Widrow. Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights. In: *Proc. 1990 International Joint Conference on Neural Networks (IJCNN 1990)*, vol. 3, pp. 21–26. June 1990. (Cited in page 44).

[Pani and Mohanta, 2011] Ajaya Kumar Pani and Hare Krishna Mohanta. A Survey of Data Treatment Techniques for Soft Sensor Design. *Chemical Product and Process Modeling*, vol. 6, no. 1, pp. 1–21, January 2011. (Cited in page 14).

[Pearson, 2002] R. K. Pearson. Outliers in Process Modeling and Identification. *IEEE Transactions on Control Systems Technology*, vol. 10, no. 1, pp. 55–63, January 2002. (Cited in page 13).

[Peng *et al.*, 2005] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, August 2005. (Cited in pages 18, 33, and 44).

[Penny and Jolliffe, 2001] Kay I. Penny and Ian T. Jolliffe. A Comparison of Multivariate Outlier Detection Methods for Clinical Laboratory Safety Data. *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 50, no. 3, pp. 295–307, 2001. (Cited in page 14).

[Pudil *et al.*, 1994] P. Pudil, J. Novovicová, and J. Kittler. Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, vol. 15, no. 11, pp. 1119–1125, November 1994. (Cited in page 125).

[Qin, 1996] S. Joe Qin. Neural Networks for Intelligent Sensors and Control - Practical Issues and Some Solutions. 1996. (Cited in page 22).

[Qin, 1998] S. Joe Qin. Recursive PLS Algorithms for Adaptive Data Modeling. *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, 1998. (Cited in page 90).

[Rakotomamonjy, 2007] A. Rakotomamonjy. Analysis of SVM Regression Bounds for Variable Ranking. *Neurocomputing*, vol. 70, pp. 1489–1501, March 2007. (Cited in page 22).

[Raudys and Jain, 1991] Sarunas J. Raudys and Anil K. Jain. Small Sample Size Effects in Statistical Pattern Recognition: recommendations for practitioners. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 252–264, March 1991. (Cited in page 91).

[Richman *et al.*, 2009] Michael B. Richman, Theodore B. Trafalis, and Indra Adrianto. Missing Data Imputation Through Machine Learning Algorithms. In: Sueellen Haupt Antonello Pasini and Caren Marzban (eds.), *Artificial Intelligence Methods in the Environmental Sciences*, pp. 153–169. Springer Netherlands, 2009. (Cited in page 12).

[Romero and Sopena, 2008a] Enrique Romero and Josep María Sopena. Performing Feature Selection With Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 431–441, March 2008a. (Cited in pages 20, 32, 33, 37, and 44).

[Romero and Sopena, 2008b] Enrique Romero and Josep María Sopena. Performing Feature Selection With Multilayer Perceptrons. *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 431–441, March 2008b. (No citations).

[Rossi *et al.*, 2006] F. Rossi, A. Lendasse, D. François, V. Wertz, and M. Verleysen. Mutual Information for the Selection of Relevant Variables in Spectrometric Nonlinear Modelling. *Chemometrics and Intelligent Laboratory Systems*, vol. 80, no. 2, pp. 215–226, February 2006. (Cited in page 17).

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Letters to Nature*, vol. 323, pp. 533–536, October 1986. (Cited in page 37).

[Schwarz, 1978] Gideon Schwarz. Estimating the Dimension of a Model. *Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978. (Cited in pages 26, and 58).

[Sexton *et al.*, 1999] Randall S. Sexton, Robert E. Dorsey, and John D. Johnson. Optimization of Neural Networks: A Comparative Analysis of the Genetic Algorithm and Simulated Annealing. *European Journal of Operational Research*, vol. 114, no. 3, pp. 589–601, May 1999. (Cited in page 37).

[Shoorehdeli *et al.*, 2009] Mahdi Aliyari Shoorehdeli, Mohammad Teshnehlab, and Ali Khaki Sedigh. Training ANFIS as an Identifier With Intelligent Hybrid Stable Learning Algorithm Based on Particle Swarm Optimization and Extended Kalman Filter. *Fuzzy Sets and Systems*, vol. 160, pp. 922–948, 2009. (Cited in page 24).

[Similä and Tikka, 2009] Timo Similä and Jarkko Tikka. Combined Input Variable Selection and Model Complexity Control for Nonlinear Regression. *Pattern Recognition Letters*, vol. 30, no. 3, pp. 231–236, February 2009. (Cited in pages 21, and 33).

[Soares *et al.*, 2013] Symone Soares, Carlos Henggeler Antunes, and Rui Araújo. Comparison of a Genetic Algorithm and Simulated Annealing for Automatic Neural Network Ensemble Development. *Neurocomputing*, vol. 121, no. 0, pp. 498–511, December 2013. (Cited in page 24).

[Soares *et al.*, 2011] Symone Soares, Rui Araújo, Pedro Sousa, and Francisco Souza. Design and Application of Soft Sensor Using Ensemble Methods. In: *Proc. 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011)*, pp. 1–8. Toulouse, France, September 2011. (Cited in pages 7, and 24).

[Soderstrom and Stoica, 1989] T. S. Soderstrom and Petre G. Stoica. *System Identification*. Prentice Hall, 1989. (Cited in page 26).

[Souza and Araújo, 2011] Francisco Souza and Rui Araújo. Variable and Time-Lag Selection using Empirical Data. In: *Proc. 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011)*, pp. 1–8. Toulouse, France, September 5-9 2011. (Cited in pages 6, 7, 14, and 18).

[Souza and Araújo, 2012] Francisco Souza and Rui Araújo. An Online Variable Selection Method Using Recursive Least Squares. In: *17th IEEE Conference on Emerging Technologies Factory Automation (ETFA 2012)*, pp. 1–8. 2012. (Cited in pages 6, and 7).

[Souza and Araújo, 2014a] Francisco Souza and Rui Araújo. Online Mixture of Univariate Linear Regression Models for Adaptive Soft Sensors. *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–9, 2014a. (Cited in pages 5, 6, and 7).

[Souza *et al.*, 2013] Francisco Souza, Rui Araújo, Tiago Matias, and Jérôme Mendes. A Multilayer-Perceptron Based Method for Variable Selection in Soft Sensor Design. *Journal of Process Control*, vol. 23, no. 10, pp. 1371–1378, November 2013. (Cited in pages 4, 5, 7, and 8).

[Souza *et al.*, 2010a] Francisco Souza, Rui Araújo, Symone Soares, and Jérôme Mendes. Variable Selection Based on Mutual Information for Soft Sensors Applications. In: *Proc. 9th Portuguese Conference on Automatic Control (CONTROLO 2010)*. Coimbra, Portugal, September 2010a. (Cited in pages 6, and 7).

[Souza *et al.*, 2011] Francisco Souza, Tiago Matias, and Rui Araújo. Co-Evolutionary Genetic Multilayer Perceptron for Feature Selection and Model Design. In: *16th IEEE Conference on Emerging Technologies Factory Automation (ETFA 2011)*, pp. 1–7. 2011. (Cited in pages 6, and 7).

[Souza *et al.*, 2010b] Francisco Souza, Pedro Santos, and Rui Araújo. Variable and Delay Selection Using Neural Networks and Mutual Information for Data-Driven Soft Sensors. In: *Proc. 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*. September 2010b. (Cited in pages 2, 6, 7, 18, and 32).

[Souza and Araújo, 2014b] Francisco A. A. Souza and Rui Araújo. Mixture of Partial Least Squares Experts and Application in Prediction Settings with Multiple Operating Modes. *Chemometrics and Intelligent Laboratory Systems*, vol. 130, pp. 192–202, January 2014b. (Cited in pages 5, 6, 7, and 8).

[Stearns, 1976] S. D. Stearns. On selecting features for pattern classifiers. In: *Proceedings of the 3rd International Conference on Pattern Recognition (ICPR 1976)*, pp. 71–75. Coronado, CA, 1976. (Cited in page 124).

[Suykens *et al.*, 2002] Johan A K Suykens, Tony Van Gestel, Jos De Brabanter, Bart De Moor, and Joos Vandewalle. *Least Squares Support Vector Machiness*. World Scientific, 2002. (Cited in pages 28, 56, and 74).

[Toher *et al.*, 2007] Deirdre Toher, Gerard Downey, and Thomas Brendan Murphy. A Comparison of Model-Based and Regression Classification Techniques Applied

to Near Infrared Spectroscopic Data in Food Authentication Studies. *Chemometrics and Intelligent Laboratory Systems*, vol. 89, no. 2, pp. 102–115, November 2007. (Cited in page 58).

[Tsymbal, 2004] Alexey Tsymbal. The Problem of Concept Drift: Definitions and Related Work. Tech. rep., Department of Computer Science, Trinity College: Dublin, Ireland, 2004. (Cited in pages 26, and 28).

[Walters-Williams and Li, 2009] Janett Walters-Williams and Yan Li. Estimation of Mutual Information: A Survey. In: Peng Wen, Yuefeng Li, Lech Polkowski, Yiyu Yao, Shusaku Tsumoto, and Guoyin Wang (eds.), *Rough Sets and Knowledge Technology*, Lecture Notes in Computer Science, pp. 389–396. Springer Berlin Heidelberg, 2009. (Cited in page 17).

[Wang *et al.*, 2012] Fuli Wang, Shuai Tan, Jun Peng, and Yuqing Chang. Process Monitoring Based on Mode Identification for Multi-Mode Process with Transitions. *Chemometrics and Intelligent Laboratory Systems*, vol. 110, no. 1, pp. 144–155, January 2012. (Cited in pages 25, and 54).

[Wang *et al.*, 2006] Haiqing Wang, Ping Li, Furong Gao, Zhihuan Song, and Steven X. Ding. Kernel Classifier with Adaptive Structure and Fixed Memory for Process Diagnosis. *AIChE Journal*, vol. 52, no. 10, pp. 3515–3531, October 2006. (Cited in page 27).

[Wang *et al.*, 2010] Xinzhe Wang, Min Han, and Jun Wang. Applying Input Variables Selection Technique on Input Weighted Support Vector Machine Modeling for BOF Endpoint Prediction. *Engineering Applications of Artificial Intelligence*, vol. 23, no. 6, pp. 1012–1018, September 2010. (Cited in pages 19, 28, and 85).

[Warne *et al.*, 2004] K. Warne, G. Prasad, S. Rezvani, and L. Maguire. Statistical and Computational Intelligence Techniques for Inferential Model Development: a Comparative Evaluation and a Novel Proposition for Fusion. *Engineering Applications of Artificial Intelligence*, vol. 17, no. 8, pp. 871–885, December 2004. (Cited in pages 14, 15, and 22).

[Werbos, 1990] Paul J. Werbos. Backpropagation Through Time: What it Does and How to do It. *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, October 1990. (Cited in page 37).

[Whitney, 1971] A. W. Whitney. A Direct Method of Nonparametric Measurement Selection. *IEEE Transactions on Computers*, vol. C-20, no. 9, pp. 1100–1103, September 1971. (Cited in page 124).

[Wold, 1975] H. Wold. Path Models With Latent Variables: The NIPALS Approach. In: Hubert M. Blalock et al (ed.), *Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building*, pp. 307–357. Academic Press, 1975. (Cited in pages 57, and 90).

[Wold, 1995] S. Wold. Chemometrics; What Do We Mean with It, and What Do We Want from It? *Chemometrics and Intelligent Laboratory Systems*, vol. 30, pp. 109–115, November 1995. (Cited in page 10).

[Wu and Luo, 2010] Yao Wu and Xionglin Luo. A Novel Calibration Approach of Soft Sensor Based on Multirate Data Fusion Technology. *Journal of Process Control*, vol. 20, no. 10, pp. 1252–1260, December 2010. (Cited in page 11).

[Xie *et al.*, 2013] Li Xie, Huizhong Yang, and Biao Huang. FIR Model Identification of Multirate Processes with Random Delays Using EM Algorithm. *AIChE Journal*, vol. 59, no. 11, pp. 4124–4132, November 2013. (Cited in page 11).

[Xing and Hu, 2009] Hong-Jie Xing and Bao-Gang Hu. Two-Phase Construction of Multilayer Perceptrons Using Information Theory. *IEEE Transactions on Neural Networks*, vol. 20, no. 4, pp. 715–721, April 2009. (Cited in pages 18, and 32).

[Yang and Ong, 2011] Jian-Bo Yang and Chong-Jin Ong. Feature Selection Using Probabilistic Prediction of Support Vector Regression. *IEEE Transactions on Neural Networks*, vol. 22, no. 6, pp. 954 –962, June 2011. (Cited in pages 21, and 22).

[Yang *et al.*, 2009] Jian-Bo Yang, Kai-Quan Shen, Chong-Jin Ong, and Xiao-Ping Li. Feature Selection for MLP Neural Network: The Use of Random Permutation of Probabilistic Outputs. *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1911–1922, December 2009. (Cited in page 28).

[Yeh and Cheng, 2010] I-Cheng Yeh and Wei-Lun Cheng. First and Second Order Sensitivity Analysis of MLP. *Neurocomputing*, vol. 73, no. 10-12, pp. 2225–2233,

June 2010. Subspace Learning / Selected papers from the European Symposium on Time Series Prediction. (Cited in pages 21, and 32).

[Yu, 2012] Jie Yu. Online Quality Prediction of Nonlinear and Non-Gaussian Chemical Processes with Shifting Dynamics Using Finite Mixture Model Based Gaussian Process Regression Approach. *Chemical Engineering Science*, vol. 82, no. 0, pp. 22–30, 2012. (Cited in pages 25, 54, and 55).

[Yuksel *et al.*, 2012] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty Years of Mixture of Experts. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1177–1193, August 2012. (Cited in pages 55, and 56).

[Zahedi *et al.*, 2005] G. Zahedi, A. Elkamel, A. Lohi, A. Jahanmiri, and M. R. Rahimpor. Hybrid Artificial Neural Network-First Principle Model Formulation for the Unsteady State Simulation and Analysis of a Packed Bed Reactor for CO2 Hydrogenation to Methanol. *Chemical Engineering Journal*, vol. 115, no. 1-2, pp. 113–120, December 2005. (Cited in page 23).

[Zamprogna *et al.*, 2005] Eliana Zamprogna, Massimiliano Barolo, and Dale E. Seborg. Optimal Selection of Soft Sensor Inputs for Batch Distillation Columns Using Principal Component Analysis. *Journal of Process Control*, vol. 15, no. 1, pp. 39–52, February 2005. (Cited in page 16).

[Zhou *et al.*, 2002] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling Neural Networks: Many Could be Better Than All. *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, May 2002. (Cited in page 24).

[Zliobaite, 2010] Indre Zliobaite. Learning Under Concept Drift: an Overview. *CoRR*, vol. abs/1010.4784, 2010. (Cited in page 26).

[Zou and Hastie, 2005] Hui Zou and Trevor Hastie. Regularization and Variable Selection Via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, April 2005. (Cited in page 20).


FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA


mais
CENTRO
Programa Operacional Regional do Centro


QREN
QUADRO DE REFERÊNCIA ESTRATÉGICO NACIONAL
PORTUGAL 2007.2013


UNIÃO EUROPEIA
Fundo Europeu de Desenvolvimento Regional


adi
agência de inovação