Universidade de Coimbra
Faculdade de Ciências e Tecnologia
Departamento de Engenharia Electrotécnica e de Computadores

MLP Co-evolucionária para Definição do Modelo e Selecção de Entradas para Aplicação em Sensores Virtuais

Tiago Ferreira Matias

Coimbra, 2011

MLP Co-evolucionária para Definição do Modelo e Selecção de Entradas para Aplicação em Sensores Virtuais

Orientador: Prof. Doutor Rui Alexandre de Matos Araújo
Co-Orientador: Eng. Francisco Alexandre Andrade de Souza

Júri:

Presidente: Prof. Doutor Paulo José Monteiro Peixoto
Vogais: Prof. Doutor Rui Alexandre de Matos Araújo
Prof. Doutor Paulo Jorge Carvalho Menezes

Tiago Ferreira Matias

Dissertação submetida para obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores

Departamento de Engenharia Electrotécnica e de Computadores

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

Setembro 2011

À minha família,

namorada

e amigos

Agradecimentos

Gostaria de agradecer aos meus pais Arminda Ferreira e Júlio Matias por terem permitido o meu ingresso na faculdade e por serem os melhores pais do mundo. Sem eles nada disto seria possível. Ao meu irmão pelo companheirismo e à minha madrinha Maria da Luz e tia Luísa Ferreira pelo carinho. Também ao resto da minha família, agradeço todo o apoio. Gostaria ainda de deixar um agradecimento especial a minha avó Leonor Bronze, não menosprezando os restantes, pela sua ajuda e infinita generosidade.

Agradeço à minha namorada La-Salete Sousa pelo seu amor, carinho e compreensão. Por estar sempre comigo nos bons e nos maus momentos, quero dar-lhe um especial obrigado.

Agradeço a todos os meus amigos e colegas que fiz durante o percurso académico por terem permitido que esse percurso fosse tão especial.

Gostaria também de agradecer a todos os meus restantes amigos pela sua amizade incondicional.

Agradeço ao orientador Professor Doutor Rui Araújo e ao co-orientador Eng. Francisco Souza pelos conselhos e opiniões que permitiram o desenvolvimento desta dissertação. Agradeço também ao Eng. Pedro Sousa e à empresa Acontrol pelo apoio no desenvolvimento desta dissertação.

Por último agradeço aos colegas do Laboratório de Controlo Inteligente e Robótica (LCIR) do Instituto de Sistemas e Robótica (ISR) pela paciência e pronta disponibilidade.

Este trabalho foi suportado pelo projecto SInCACI "Sistemas Inteligentes de Controlo, Aquisição e Comunicação Industrial"(referência: SInCACI/3120/2009) que é suportado pelo Programa Operacional Mais Centro, financiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), e pela Agência de Inovação (AdI).

Abstract

This dissertation presents a method for model design and variable selection based on a multilayer perceptron model for soft sensors application, called Co-Evolutionary Genetic Multilayer Perceptron (CEV-MLP). Soft sensors (SS) are a growing application due to the fact that they can estimate variables that cannot be measured automatically, or can only be measured with a high cost, inaccurately, sporadically or with high delays (e.g. laboratory tests). These, as the name indicates, are a software sensor that uses the information obtained by the physical sensors available in the process to estimate the desired variable.

The algorithm developed in this dissertation is an automatic tool for the model design and selection of the best variables in SS context, using for this the empirical input and output data of the process variables. Using a co-evolutionary genetic algorithm, the method selects the best subset of pairs of input variables and respective delays and obtains the model that best fits the set of selected entries.

To validate the method, this was applied, along with two state-of-the-art methods for variable selection, in one artificial dataset, in seven real datasets available in public repositories, and in a problem of estimating the amount of fluorine in the effluent of a wastewater treatment plant. After proving the effectiveness of the proposed method, this was applied to a real problem for the estimation of chemical oxygen demand in a pulp production process.

Keywords: Soft Sensors, Multilayer Perceptron, Neural Networks, Co-evolutionary Genetic Algorithm, Variable Selection.

Resumo

Esta dissertação apresenta um método de desenvolvimento de sensores virtuais baseado numa rede neuronal multicamada, denominado CEV-MLP. Os sensores virtuais são uma aplicação em crescimento devido ao facto de poderem estimar variáveis que não podem ser mensuradas automaticamente, ou que podem apenas ser mensuradas com um custo elevado, de forma imprecisa, esporadicamente ou com atrasos elevados (*e.g.* análises laboratoriais). Estes, como o próprio nome indica, são sensores desenvolvidos em *software* que utilizam informação obtida a partir de sensores físicos disponíveis no processo para estimar a variável pretendida.

O algoritmo desenvolvido neste trabalho de dissertação é uma ferramenta automática para o projecto de sensores virtuais utilizando, para isso, apenas o conhecimento empírico das variáveis de entrada e saída do processo. Utilizando um algoritmo genético co-evolucionário, o método selecciona o melhor subconjunto de pares de variáveis de entrada e respectivos tempos de atraso e obtém o modelo que mais se adequa ao conjunto de entradas seleccionado.

Para a validação do método, este foi aplicado, juntamente com outros dois métodos de selecção de variáveis, a um *dataset* artificial, a sete *datasets* reais disponíveis em repositórios públicos e a um problema de estimação da quantidade de flúor no efluente de uma estação de tratamento de águas residuais. Após ter comprovado a eficácia do método proposto, este foi aplicado a um problema real de estimação da carência química de oxigénio num processo de produção de pasta de papel.

Palavras-chave: Sensores Virtuais, *Multilayer Perceptron*, Redes Neuronais, Algoritmos Genéticos Co-evolucionários, Selecção de Variáveis

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AG	Algoritmo Genético
ARX	*Autoregressive with Exogenous Inputs*
CC	Coeficiente de Correlação
CEV-MLP	*Co-evolutionary Genetic Multilayer Perceptron*
CQO	Carência Química de Oxigénio
DDSS	*Data-Driven Soft Sensor*
ETAR	Estação de Tratamento de Águas Residuais
FAD	Friedman Artificial Domain
IM	Informação Mútua
MAE	Erro Absoluto Médio (*Mean Absolute Error*)
MIT	*Massachusetts Institute of Technology*
MLP	Rede Neuronal Multicamada (*Multilayer Perceptron*)
mRMR	Mínima Redundância Máxima Relevância
MSE	Erro Quadrático Médio (*Mean Square Error*)
NARMAX	*Nonlinear Autoregressive Moving Average with Exogenous Inputs*
NARX	*Nonlinear Autoregressive Model with Exogenous Inputs*
NMSE	Erro Quadrático Médio Normalizado (*Normalized Mean Square Error* (NMSE))

PCA Análise de Componentes Principais (*Principal Components Analyses*)

RBF *Radial Basis Function*

RN Redes Neuronais

SBS Pesquisa Sequencial para Trás (*Sequential Backward Search*)

SFS Pesquisa Sequencial para a Frente (*Sequential Forward Search*)

SS Sensor Virtual (*Soft Sensor*)

SSE Erro Quadrático Total (*Sum of Square Error*)

SVM *Support Vector Machine*

T-S Takagi-Sugeno

Lista de Símbolos

b	*Bias* dos neurónios da camada oculta
b_O	*Bias* do neurónio da camada de saída
E	Função de erro
$f(\cdot)$	Função de activação dos neurónios da camada oculta
$g(\cdot)$	Função de activação dos neurónios da camada de saída
h	Número de neurónios da camada oculta
h_{cev}	Número máximo de neurónios da camada oculta no método proposto
h_j	Sinal que sai do neurónio j da camada oculta
h_l	Número de neurónios da camada oculta com função de transferência linear
h_{th}	Número de neurónios da camada oculta com função de transferência tangente hiperbólica
k_{max}	Número máximo de cromossomas no nível 3
l_{max}	Número máximo de cromossomas no nível 2
m_{max}	Número máximo de cromossomas no nível 1
n	Número de entradas
N	Número de amostras

N_{max}	Número máximo de iterações do algoritmo CEV-MLP com o mesmo erro
P_{te}	Proporção de dados de teste
P_{tr}	Proporção de dados de treino
P_{va}	Proporção de dados de validação
r_j	Sinal efectivo na entrada do neurónio j da camada oculta
s	Sinal de entrada do neurónio de saída
T_m	Taxa de Mutação
T_r	Taxa de Reprodução
$U^{(k)}$	Conjunto de entradas
W	Matriz de pesos das ligações das entradas aos neurónios da camada oculta
$w_{i,j}$	Peso da ligação da entrada i ao neurónio j da camada oculta
w_{jO}	Peso da ligação do neurónio j ao neurónio de saída
W_O	Vector de pesos das ligações da camada oculta com a saída
$X^{(k)}$	Conjunto de entradas considerando os respectivos atrasos
y	Saída estimada
y_d	Saída desejada
y_k	Saída do neurónio k
$\Delta w_{i,j}$	Factor de correcção do peso da ligação da entrada i ao neurónio j da camada oculta
Δw_{jO}	Factor de correcção do peso da ligação do neurónio j ao neurónio de saída
γ	Taxa de aprendizagem do algoritmo de retro-propagação

Capítulo 1

Introdução

1.1 Enquadramento

Os sensores são usados na maioria dos processos industriais para a análise em tempo real e como medida de controlo dos mesmos. Contudo, muitas vezes a medição em tempo real de certas variáveis importantes para o controlo dos processos pode ser impraticável ou praticável mas com custos muito elevados, de forma imprecisa, esporadicamente ou com atrasos elevados, como por exemplo quando é necessário recorrer a análises laboratoriais para realizar a medição de uma determinada variável.

Os *soft sensors* (SS) ou sensores virtuais são uma ferramenta baseada em *software* e cujo objectivo é estimar a informação dessas variáveis com base no conhecimento físico do processo ou nos dados recolhidos por outros sensores existentes no processo.

No desenvolvimento dos SS é necessário realizar um pré-processamento das variáveis de entrada de modo a remover os dados inconsistentes e seleccionar as variáveis mais importantes para a estimação da variável alvo pretendida.

A problemática da selecção das melhores variáveis de entrada é um aspecto fundamental na construção do modelo para o desenvolvimento de um SS. Uma rede treinada com o conjunto de todas as variáveis de entrada possíveis, no caso de este ser grande, necessita de um grande tempo de aprendizagem e para além disso, se o conjunto possuir muitas variáveis de entrada irrelevantes ou redundantes, o modelo construído pode ter uma fraca generalização nos dados de teste. A este problema de selecção de variáveis é acrescido o problema da selecção dos melhores atrasos para cada variável, no caso de o processo ter uma longa duração de processamento. Num processo industrial uma entrada

pode não influenciar imediatamente a saída, mas sim após um certo período de tempo, sendo assim necessário determinar qual o tempo que essa entrada demora a manifestar-se na saída e, portanto, o melhor atraso temporal com que esta variável deverá entrar no modelo do SS, caso seja de todo relevante usar a variável.

1.2 Estado da Arte

A selecção das melhores variáveis e dos respectivos atrasos é um processo fundamental na construção de um SS visto possibilitar a diminuição do tempo de treino da rede e o aumento da qualidade de desempenho do modelo construído.

Segundo [1], os métodos mais comuns para a execução da selecção de variáveis em redes neuronais multicamada (*multilayer perceptron* - MLP) (ver Capítulo 3) usam um procedimento de pesquisa sequencial para trás (*sequential backward search* - SBS) e o erro quadrático total (*sum of square error* - SSE) como função de custo. Neste mesmo artigo é discutido o algoritmo de selecção de variáveis SBS tradicional em redes MLP para a selecção de variáveis. Os autores verificam que o retreino da rede, aquando da remoção de uma variável de entrada, é benéfico para o resultado final. Contudo, este método em grandes *datasets* é computacionalmente dispendioso, porque é necessário retreinar a rede $n(n-1)/2$ vezes, onde n é o número de entradas. Para além disso, existe também o problema do número de neurónios ser fixo, podendo assim degradar o resultado, visto que o número óptimo de neurónios não é conhecido. Ao contrário deste, o método apresentado em [2] utiliza um procedimento de pesquisa sequencial para a frente (*sequential forward search* - SFS) para a selecção simultânea das variáveis de entrada e do número de neurónios da camada oculta, denominado CAFS. O método divide as variáveis de entrada em dois grupos baseando-se na sua correlação e, partindo de uma rede inicial constituída por apenas duas entradas e um neurónio na camada oculta, vai procedendo à selecção das variáveis enquanto que aumenta também o tamanho da rede. A principal desvantagem deste método prende-se com a selecção definitiva de uma variável, ou seja, uma vez seleccionada uma variável, esta não pode ser removida. Este problema está presente em todos os métodos de pesquisa SBS e SFS, sendo que para o contornar é necessário utilizar um método de pesquisa global, tal como um algoritmo genético.

Em [3] é apresentado um algoritmo de poda (*pruning*) para a construção de modelos

que utilizam redes neuronais MLP. Neste método de poda, a rede é inicialmente sobredimensionada e depois os neurónios da camada oculta e os pesos mais irrelevantes são sucessivamente removidos, diminuindo assim o tamanho da rede MLP e aumentando a sua generalização. É proposta também uma medida de sensibilidade para verificar a sensibilidade da saída devido a uma perturbação na entrada e uma medida de relevância para verificar quais os neurónios mais importantes.

Em [4], é proposta uma nova função de custo para a selecção simultânea das entradas e do número de neurónios da camada oculta para um modelo MLP. Este método impõe penalizações aos pesos durante o treino da rede para que as variáveis de entrada irrelevantes possam ser excluídas. O desempenho do método depende da afinação do montante de penalização e da forma da função de penalização, isto é, o método não é totalmente automático.

Um novo método rápido para a selecção das variáveis de entrada é apresentado em [5]. É assumido que os modelos não-lineares como modelos polinomiais *nonlinear autoregressive model with exogenous inputs* (NARX), série de Volterra e as redes neurais (RN) podem obter um desempenho equivalente, considerando que certas condições são satisfeitas. É usado um modelo de séries de Volterra que, devido à sua natureza linear-nos-parâmetros, permite o uso, por exemplo, do método dos mínimos quadrados ortogonais para efectuar a identificação do modelo, permitindo uma rápida selecção das variáveis de entrada. Este método permite uma redução significativa da complexidade computacional mas não considera a optimização automática do modelo MLP.

Os algoritmos genéticos (AG) têm comprovado ser um instrumento útil para resolver problemas de optimização. Em [6], é estudado como determinar o tamanho óptimo da tubagem numa rede de gás natural. Em [7] um algoritmo genético é usado para seleccionar as variáveis que maximizam a informação mútua (IM) entre as variáveis de entrada e saída, para a predição do fluxo de petróleo.

Em [8] é proposta uma nova abordagem que usa métodos de selecção de variáveis não-lineares em conjunto com modelos *fuzzy* Takagi-Sugeno (T-S) para o desenvolvimento de SS. Este método usa um algoritmo genético co-evolucionário para identificar as regras do modelo difuso T-S de segunda ordem através dos dados de entrada e saída disponíveis. O desenho do SS é executado em dois passos. Primeiro, as variáveis de entrada do modelo difuso são pré-seleccionadas através de coeficientes de correlação (CC), mapas de

Kohonen e quocientes de Lipschitz. Numa segunda fase, são usados AG hierárquicos para identificar o próprio modelo difuso. A selecção de variáveis de entrada proposta por [8] tem algumas falhas. Primeiro, a selecção do número de neurónios em mapas de Kohonen não é automaticamente executada. Em segundo lugar, os atrasos não são seleccionados juntamente com variáveis de entrada. Esta segunda falha pode ser apontada à maioria das técnicas de selecção de variáveis de entrada para o desenvolvimento de SS. A selecção das variáveis de entrada juntamente com os respectivos atrasos pode levar a uma melhoria da exactidão dos resultados visto que uma variável com o atraso correcto pode conter mais informação sobre a saída do que uma variável com o atraso incorrecto [9].

No que se refere à selecção dos atrasos das variáveis de entrada, em [7] é proposto um método de selecção dos tempos de atraso baseado em IM para modelos não-lineares. Através da utilização de um AG e utilizando como função de custo o princípio da mínima redundância máxima relevância (mRMR) os melhores tempos de atraso são seleccionados. Foi demonstrado que a IM apresenta melhores resultados quando comparada com uma análise da correlação, no caso analisado. Outro método baseado em IM é apresentado em [10]. Utilizando uma heurística geométrica denominada *distância até à diagonal*, a selecção dos melhores atrasos é efectuada. Contudo, a IM não é a única abordagem utilizada. Em [11] é usado um algoritmo genético e um modelo *autoregressive with exogenous inputs* (ARX) para a selecção dos melhores atrasos temporais para a estimação das concentrações de óxidos de azoto (NO_X) e oxigénio (O_2) durante o processo de combustão em caldeiras industriais e em [12] é proposta uma técnica baseada no modelo *radial basis function* (RBF) para determinar os melhores atrasos temporais.

1.3 Objectivos

Esta dissertação tem como principal objectivo desenvolver metodologias para SS. Em particular, o objectivo será desenvolver métodos para seleccionar as melhores variáveis de entrada e os respectivos atrasos em simultâneo com a optimização do modelo, para aplicar à construção de *data-driven* SS.

Especificamente os objectivos desta dissertação são:

- Desenvolver e testar os algoritmos de selecção de variáveis propostos em [1] e [5];
- Desenvolver uma metodologia baseada em algoritmos genéticos para a construção

do modelo e selecção de variáveis de entrada e respectivos atrasos;

- Aplicar as metodologias desenvolvidas na estimação da carência química de oxigénio (CQO) num processo de produção de pasta de papel.

1.4 Trabalho Realizado

Neste trabalho foi desenvolvido um método para o desenho de SS denominado *Co-evolutionary Genetic Multilayer Perceptron* (CEV-MLP). Este foi desenvolvido a partir da ideia do meu Co-orientador Eng. Francisco Souza em construir um método automático para o desenho de SS baseado na estrutura hierárquica apresentada em [8] e utilizando redes MLP.

O método desenvolvido usa uma rede MLP em conjunto com um algoritmo genético co-evolucionário para automaticamente projectar o modelo e seleccionar as variáveis de entrada juntamente com os respectivos atrasos. Para o desenho do SS o método utiliza apenas o conhecimento empírico das variáveis de entrada e saída do processo, não sendo necessário qualquer outro conhecimento do funcionamento do processo e das melhores variáveis.

Para a validação do algoritmo, este foi comparado com dois algoritmos apresentados em [1] e [5] num *dataset* artificial e em oito *datasets* reais (sete disponíveis em repositórios públicos e um problema de estimação da concentração de flúor numa estação de tratamento de águas residuais (ETAR)).

Depois de validado o algoritmo proposto, este foi aplicado num problema real, especificamente na estimação da CQO na etapa de branqueamento num processo de produção de pasta de papel.

Após desenvolvido o algoritmo, foram realizadas experiências comparativas nos *datasets* acima referidos de modo a analisar o comportamento do algoritmo, a sua robustez e eficácia.

Do trabalho resultou também o artigo *Co-evolutionary Genetic Multilayer Percetron for Variable Selection and Model Design* que será publicado e apresentado na conferência *16th IEEE international Conference on Emerging Technologies and Factory Automation*, que decorrerá de 5 a 9 de Setembro de 2011 em Toulouse, França.

1.5 Organização da Dissertação

Esta dissertação está estruturada em seis capítulos. No primeiro, faz-se uma introdução ao trabalho desenvolvido, explica-se a importância do tema escolhido e os objectivos da dissertação.

A aplicação dos SS, a sua importância e desenvolvimento são abordadas no Capítulo 2.

Uma introdução às redes neuronais e mais detalhadamente às redes neuronais multi-camada é apresentada no Capítulo 3.

No Capítulo 4 é apresentado o algoritmo proposto para a realização da selecção de variáveis com os respectivos atrasos e a optimização do modelo.

No Capítulo 5 são apresentados os resultados experimentais de modo a comprovar a eficácia do algoritmo proposto no Capítulo 4.

No Capítulo 6 são apresentadas as conclusões acerca do trabalho, assim como possíveis trabalhos futuros.

Capítulo 2

Sensores Virtuais

Na indústria, o controlo de processos em malha fechada é fundamental. Contudo, o atraso nas medições de algumas variáveis importantes do processo pode provocar um decréscimo do desempenho do sistema de controlo, podendo-se mesmo chegar ao ponto em que o sistema não consiga controlar devidamente o processo.

Os SS são um instrumento muito valioso para situações onde existam problemas de medição em tempo real de certas variáveis do processo de fabrico. Os problemas surgem por não ser possível mensurar essas variáveis, ou apenas ser possível mensurá-las com um custo elevado, de forma imprecisa, esporadicamente ou com atrasos elevados.

Por exemplo, na indústria da pasta de papel é importante obter o valor de CQO, pois trata-se de uma variável que influencía directamente a qualidade do produto. Este valor é medido em laboratório, trazendo uma latência incomportável para sistemas de controlo automático. Com a utilização de um SS para a medição em tempo real desta variável pode proceder-se ao doseamento dos reagentes a serem introduzidos no processo de forma a melhorar a qualidade do produto. Este controlo pode ser executado antecipadamente numa fase inicial do processo, evitando a introdução de reagentes em excesso no processo para satisfazer o mesmo fim. Isto aumenta a qualidade do produto e diminui a quantidade de reagentes, aumentando assim os ganhos e diminuindo os custos.

Um SS pode ser então definido como um sensor baseado em *software* e inteligência computacional que é uma alternativa aos sensores tradicionais. Um SS permite a estimação das variáveis difíceis de mensurar, através do modelo que constrói baseando-se nos sensores/variáveis fáceis de mensurar.

As vantagens na utilização dos SS são as seguintes:

- Representam uma alternativa de baixo custo para os dispositivos de *hardware* economicamente dispendiosos;
- Podem trabalhar em paralelo com os sensores de *hardware*, fornecendo informações úteis para tarefas de detecção de falhas, permitindo a realização de processos mais fiáveis;
- Podem ser facilmente implementados em sistemas de processamento existentes (*e.g.* microcontroladores);
- Permitem a estimação em tempo real de variáveis do processo, superando os atrasos introduzidos por alguns sensores de *hardware* e medidas laboratoriais, melhorando assim o desempenho das estratégias de controlo.

2.1 Aplicações

Existem diversas razões pelas quais os SS podem ser usados proveitosamente em aplicações industriais. Actualmente são ferramentas rotineiras que tendencialmente estão a deixar de se aplicar somente como sensores em esquemas de controlo em malha aberta, passando cada vez mais a serem usados como sensores em esquemas em malha fechada ou esquemas de controlo adaptativos. Além disso, a grande disponibilidade de analisadores *on-line* e sistemas digitais, que são usados para a monitorização e controlo, possibilita a implementação de SS com um aumento mínimo, ou mesmo nulo, dos preços iniciais [13].

Os SS podem ser usados para resolver uma série de problemas diferentes, tais como dispositivos de medida de *back-up*, previsão em tempo real para o controlo da planta, validação de sensores e estratégias de diagnóstico de falhas.

2.1.1 Dispositivo de Medida de *Back-up*

Em algumas plantas industriais existe um grande número de sensores para poder realizar a monitorização e controlo de certas variáveis. Para isso, é exigido aos dispositivos de medida e aos sistemas de transmissão uma grande quantidade de informação, sendo assim necessário que este *hardware* seja robusto e que periodicamente esteja sujeito a manutenção. Para além disso, podem também ocorrer falhas nesses sensores que não

permitam a utilização da informação para o controlo da planta. Nestes casos uma alternativa é o uso de um SS para a substituição momentânea do equipamento de medição não disponível, evitando a degradação do desempenho da planta.

2.1.2 Redução de *Hardware*

A medição em tempo real de algumas variáveis pode obrigar à aquisição de sensores dispendiosos. Os SS são uma alternativa a esses sensores tradicionais, representando assim uma possível fonte de economia.

2.1.3 Medição em Tempo Real

Outra vantagem dos SS é a possibilidade de permitirem a avaliação em tempo real de variáveis importantes ao sistema, que de outra forma só poderiam ser obtidas através de análises laboratoriais. Esta avaliação em tempo real das variáveis não-facilmente mensuráveis permite o controlo e a monitorização das mesmas.

2.1.4 Detecção de Falhas, Validação de Sensores e Diagnóstico

Um sistema de controlo industrial pode ser visto como uma hierarquia de pelo menos três níveis: o primeiro nível é o nível de controlo, que implementa a rede de controlo por meio de controladores *feedback* e *feedforward*, observadores de estado, estimadores de parâmetros, etc. Acima do nível de controlo, o nível de supervisão realiza a tarefa de monitorização continua do processo e o nível mais alto é dedicado à coordenação, gestão e optimização das actividades que constituem o sistema de controlo com as directivas de alto nível, a fim de maximizar o desempenho do sistema em relação a determinados critérios.

A detecção de falhas e processos de diagnóstico são parte integrante do segundo nível e tem como principais objectivos:

- Detecção precoce de possíveis falhas no sistema;
- Fornecer um sistema de apoio à decisão para a manutenção preventiva;
- Fornecer uma base para o desenvolvimento de sistemas tolerantes a falhas.

A validação de sensores é um tipo especial de detecção de falhas, no qual o sistema a ser monitorizado é um sensor ou um conjunto de sensores.

2.2 Desenvolvimento de Sensores Virtuais

Geralmente, consideram-se três modelos distintos para o desenvolvimento de SS, denominados *white-box*, *black-box* e *gray-box*. Os modelos *white-box*, também chamados modelos *model-driven*, são baseados na construção do modelo através do conhecimento físico ou químico do processo. Embora os modelos possam reflectir o processo de forma clara, o seu desenvolvimento exige uma profunda compreensão do processo, conhecimentos estes que nem sempre podem estar disponíveis. Alguns processos reais, tais como processos biológicos ou químicos, podem ser influenciados por muitos factores, o que origina, devido ao grande número de variáveis de entrada (algumas delas com características não-lineares), um aumento da dificuldade na modelação do processo. Tudo isto pode provocar um desvio entre os resultados do modelo e os valores reais, tornando por isso impraticável esta abordagem.

Em contraste com os modelos *white-box*, existem os modelos *black-box*, também chamados *data-driven*. Estes são construídos com base em observações empíricas do processo, através de técnicas de regressão ou estatísticas, sem qualquer informação sobre o funcionamento interno do processo. As observações podem ser utilizadas para construir o modelo usando, por exemplo, o método dos mínimos quadrados (*least-squares method*), *autoregressive with exogenous inputs* (ARX), *nonlinear autoregressive moving average with exogenous inputs* (NARMAX), *multilayer perceptron* (MLP) e *support vector machine* (SVM).

Existe ainda um terceiro tipo de modelo, que é denominado *gray-box* visto ser uma abordagem híbrida entre os dois modelos já descritos. Estes modelos incorporam tanto o conhecimento prévio do modelo dos processos, como a modelação e predição a partir dos dados dos processos.

O desenvolvimento de um *data-driven soft sensor* (DDSS) pode ser dividido em quatro etapas:

1. Recolha de dados e análise;

2. Pré-processamento dos dados;

3. Selecção, treino e validação do modelo;

4. Manutenção do SS.

2.2.1 Recolha de Dados e Análise

A estratégia usada na recolha de dados é fundamental para uma boa identificação do modelo do sistema. Nesta fase, um problema importante é a identificação preliminar das possíveis variáveis importantes com base no conhecimento do processo. Outro problema de igual importância é a sincronização dos dados visto que as amostras das diversas variáveis do processo são obtidas normalmente com diferentes frequências de amostragem.

Depois de recolhidos os dados, é necessária uma análise visual por parte de um operador humano para verificar a existência de, por exemplo, dados corrompidos ou variáveis constantes. Tais tipos de dados não contêm informação importante do processo e, sendo assim são removidos visto que não irão ser utilizados no processo de aprendizagem do modelo.

Nesta primeira etapa são também seleccionados os dados de treino e validação do modelo.

2.2.2 Pré-processamento dos Dados

O objectivo desta fase é processar mais efectivamente os dados de tal forma que possam ser utilizados na construção do modelo. As etapas do pré-processamento dos dados são normalmente a detecção de dados em falta, a detecção de dados inconsistentes (*outliers*) e a selecção de variáveis relevantes com os respectivos atrasos temporais.

2.2.2.1 Detecção de Dados em Falta

Os dados em falta são uma ou mais medidas de uma ou mais variáveis de entrada que não foram realizadas ou foram realizadas com erros. Normalmente estão presentes em todos os processos industriais devido, por exemplo, a amostras em falta devido a problemas de comunicação ou a uma falha nos sensores de *hardware*. Para a substituição desses dados em falta podem ser usadas técnicas de interpolação.

2.2.2.2 Detecção de Dados Inconsistentes (*Outliers*)

Uma falha de *hardware*, um problema de transmissão ou uma falha na medição pode originar a presença de *outliers* no conjunto de dados. Um *outlier* é uma observação que é inconsistente com os dados do restante conjunto e sendo assim, a sua presença prejudica o desempenho dos modelos [14].

Para superar este tipo de problemas podem ser usados alguns métodos de detecção de *outliers*, como por exemplo, análise de componentes principais (*principal components analyses* - PCA) [15, 16, 17] e a regra 3σ [18].

2.2.2.3 Selecção de Variáveis e Atrasos

No caso dos DDSS pode acontecer não haver conhecimento do processo ou não se conhecer o significado das variáveis de entrada. Algumas dessas variáveis podem ser redundantes ou irrelevantes no processo de aprendizagem do modelo e, sendo assim, deve-se proceder a selecção das variáveis que mais influenciam o processo que se pretende estimar. Como já referido, o processo de selecção provoca uma melhoria no processo de aprendizagem do modelo, possibilitando assim um aumento do desempenho de estimação. Associado a este problema, está o problema de seleccionar o melhor tempo de atraso com que cada variável deve ser usada no modelo.

Este tópico será abordado mais detalhadamente no Capítulo 4.

2.2.3 Selecção, Treino e Validação do Modelo

Esta é uma fase crítica na construção do SS, visto que dela dependem em grande parte a qualidade de predição do mesmo.

Quanto à selecção do tipo de modelo não existe, até agora, nenhuma abordagem teórica para realizar esta tarefa e, portanto, o modelo a utilizar e os seus parâmetros são escolhidos de acordo com a aplicação. O tipo de modelo seleccionado depende fortemente das preferências pessoais e experiências passadas do projectista [14].

Depois de definir o modelo que mais se adequa à aplicação, este necessita de ser treinado. Utilizando os dados de treino, os parâmetros do modelo vão sendo ajustados de forma iterativa para que o modelo consiga aprender informação relevante existente nos dados de treino do processo.

Na fase de validação verifica-se o desempenho do modelo obtido na fase anterior, usando os dados de validação que são obrigatoriamente diferentes dos dados de treino. Existem diversas ferramentas para a avaliação do desempenho do modelo como, por exemplo, erro quadrático total (*sum of square error* - SSE), erro quadrático médio (*mean square error* - MSE) ou erro absoluto médio (*mean absolute error* - MAE). A validação de modelos pode também ser suportada por testes de qualidade e verificações realizadas por operadores experientes, de modo a determinar se o modelo desenvolvido apresenta falhas.

2.2.4 Manutenção do Sensor Virtual

Nesta última etapa, o objectivo é manter uma boa resposta do SS ao longo do tempo. A construção do modelo assume que o comportamento do processo é estacionário, contudo isto pode não acontecer [19]. Estas mudanças conduzem a um decréscimo da precisão do SS, originando que este precise de ser calibrado periodicamente.

Capítulo 3

Redes Neuronais

As redes neuronais artificiais, ao contrário dos sistemas heurísticos, assim chamados porque procuram obter sistemas inteligentes baseados em lógica, inspiram-se na estrutura do cérebro, isto é, na maneira como este é organizado e em como é capaz de aprender e executar tarefas computacionais.

Em 1943, o neurofisiologista Warren McCulloch, do MIT, e o matemático Walter Pitts, da Universidade de Illinois, publicam um trabalho pioneiro sobre redes neuronais em [20]. O modelo, originalmente construído com resistências variáveis e amplificadores, foi adaptado aos sistemas computacionais, sendo neste momento uma ferramenta bastante utilizada na aproximação de funções, na previsão de séries temporais, em classificação e reconhecimento de padrões, por exemplo.

Uma rede neuronal biológica é uma complexa estrutura formada por neurónios e suas interligações, e que é capaz de processar informações adquiridas a fim de armazená-las e aprender sobre elas.

Cada neurónio é ligado a outros neurónios por meio de dendritos (por onde a informação chega) e axónios (por onde a informação é transmitida). Existe também a região da soma onde é efectuado o somatório dos impulsos recebidos e, caso o impulso seja suficiente, transmitirá novos impulsos a outros neurónios por meio do axónio.

No caso de uma rede neuronal artificial MLP, os dendritos foram substituídos por entradas, cujas ligações com os neurónios artificiais são realizadas através de elementos chamados de pesos. Os estímulos captados pelas entradas são processados pela função de soma, e o limiar de disparo do neurónio biológico foi substituído por uma função de activação.

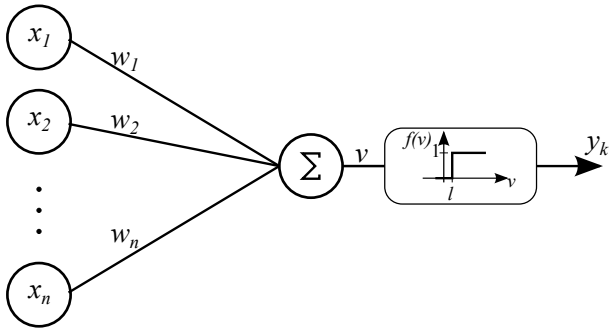


Figura 3.1: Modelo de neurónio de McCullock e Pitts.

Segundo a proposta de McCullock e Pitts em [20], uma unidade de processamento ou neurónio pode ser resumida da seguinte forma (ilustração na Figura 3.1):

1. Sinais são apresentados à entrada;

2. Cada sinal é multiplicado por um peso, que indica a sua influência na saída da unidade;

3. É feita a soma ponderada dos sinais que produz um nível de actividade;

4. Se este nível de actividade exceder um certo limite (*threshold*) a unidade produz uma determinada resposta de saída.

Através da análise da Figura 3.1 e considerando que x_i e $w_i \in \mathbb{R}$, com $i = 1, \ldots, n$, são os sinais de entrada e os pesos, respectivamente, o valor de activação v pode ser escrito como:

$$v = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n. \tag{3.1}$$

Considerando a função de transferência um degrau unitário com transição em l, a saída do neurónio k é dada por:

$$y_k = \begin{cases} 1, & \text{se } v \geq l, \\ 0, & \text{se } v < l. \end{cases} \tag{3.2}$$

Na Figura 3.2 é apresentado um modelo mais actual de um neurónio artificial obtido por generalização do modelo de McCulloch e Pitts.

Neste modelo cada ligação recebe uma entrada x_i e contém um peso sináptico associado w_i, com $i = 1, \ldots, n$, que é multiplicado por essa entrada de modo a obter a saída

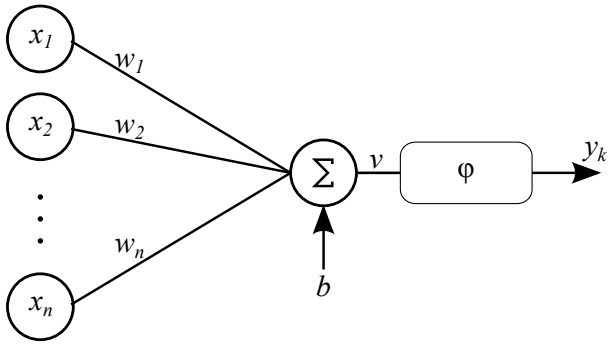


Figura 3.2: Modelo actual de um neurónio artificial.

dessa ligação. A saída no neurónio k é então dada por:

$$y_k = \varphi(v), \tag{3.3}$$

sendo v o valor de activação dado por:

$$v = \sum_{i=1}^{n} w_i x_i + b, \tag{3.4}$$

onde φ é a função de activação e b é o valor de *bias*, que tem como objectivo aumentar ou diminuir a entrada líquida da função de activação.

3.1 Tipos de Funções de Activação

A função de activação, representada por $\varphi(v)$, define a saída de um neurónio, em função do valor de activação v, sendo possível através dela introduzir não-linearidades na rede. Há várias funções de activação possíveis de serem aplicadas, de acordo com o tipo de saída que desejamos. Alguns exemplos das mais comuns são:

- Degrau unitário:

$$\varphi(x) = \begin{cases} 1, & \text{se } x \geq l, \\ 0, & \text{se } x < l. \end{cases} \tag{3.5}$$

- Linear:

$$\varphi(x) = x. \tag{3.6}$$

- Linear por partes:

$$\varphi(x) = \begin{cases} 1, & \text{se } x \geq \frac{1}{2}, \\ x, & \text{se } -\frac{1}{2} < x < \frac{1}{2}, \\ 0, & \text{se } x \leq -\frac{1}{2}. \end{cases} \tag{3.7}$$

- Sigmóide:

$$\varphi(x) = \frac{1}{1 + e^{-ax}}. \tag{3.8}$$

- Tangente Hiperbólica:

$$\varphi(x) = \frac{e^{2x} - 1}{e^{2x} + 1}. \tag{3.9}$$

3.2 Arquitectura de uma Rede Neuronal

Uma rede inclui um conjunto de neurónios (nós) que se interligam entre si e que podem estar organizados em camadas.

Segundo o número de camadas (os neurónios de entrada não contam no momento da contagem do número de camadas), uma arquitectura pode ser classificada como[21]:

- Rede de uma única camada (*singlelayer*) - neste modelo há somente entradas e neurónios de saída;
- Rede de várias camadas (*multilayer*) - neste modelo, além das entradas e dos neurónios (da camada) de saída, há outros neurónios organizados em camadas ocultas que realizam transformações sucessivas entre as entradas e a camada de saída.

Quanto ao arranjo das ligações ou ao fluxo de dados, uma rede pode ser classificada como[21]:

- Redes directas (*feedforward*):
  - Não existem ciclos (*loops*) de ligações;
  - Os dados propagam-se através de uma sequência de camadas, mas não existem realimentações de dados para nós da mesma camada ou de camadas anteriores.
- Redes recorrentes (*feedback*):
  - As ligações apresentam ciclos (*loops*) ao longo da rede;

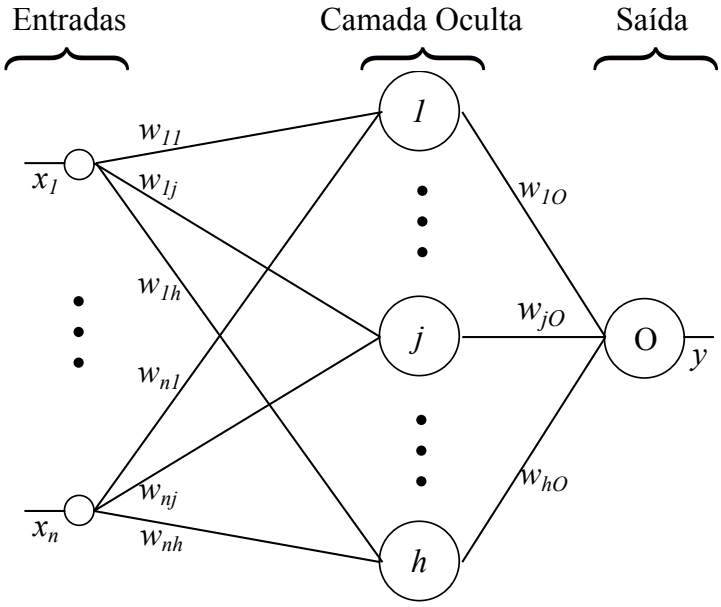


Figura 3.3: Arquitectura de uma rede neuronal multicamada.

– Existem realimentações de dados para nós da mesma camada ou de camadas anteriores.

3.3 Redes Neuronais Multi-Camada MLP

As redes neuronais multicamada *multilayer perceptron* (MLP) são redes neuronais compostas por uma camada de neurónios de entrada, uma camada de neurónios de saída e uma ou mais camadas ocultas (intermédias), sendo que os neurónios que constituem uma camada não partilham qualquer ligação entre eles.

Em [22] Rumelhart, Hinton e Williams desenvolveram um algoritmo de treino de retro-propagação (*backpropagation*) para redes com camadas ocultas, criando o modelo actual MLP. Por sua vez em [23], Cybenko provou que uma rede neuronal *feedforward* com uma camada oculta, desde que com o número suficiente de neurónios na camada oculta, pode aproximar qualquer função contínua.

A Figura 3.3 ilustra a arquitectura da rede neuronal com duas camadas, uma camada oculta e uma camada de saída. Esta será a rede usada nesta dissertação visto ser necessário apenas a aproximação de funções contínuas.

Como é observável na Figura 3.3, a rede é constituída por n entradas, uma camada oculta com h neurónios e uma camada de saída com um neurónio.

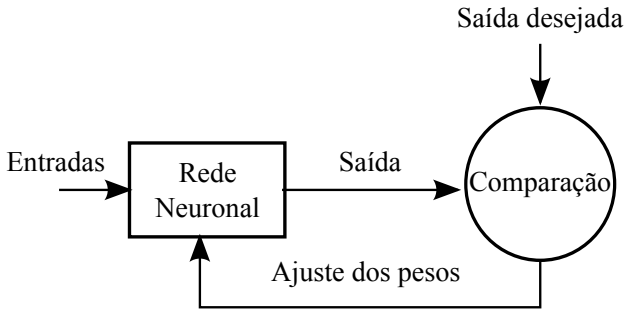


Figura 3.4: Esquema do algoritmo de retro-propagação do erro.

Matematicamente, uma rede neuronal multicamada por ser representada por:

$$y \;=\; g\left(f\left(\mathbf{x}^T\mathbf{W}+\mathbf{b}\right)\mathbf{w}_O + b_O\right), \tag{3.10}$$

onde $\mathbf{x} = [x_1, \ldots, x_n]^T$ é o vector de entrada, y é a saída estimada, $\mathbf{W} = [w_{ij}]$ é a matriz de pesos $n \times h$ das ligações das n entradas aos h neurónios da camada oculta e $\mathbf{b} = [b_1, \ldots, b_h]$ é o vector das *bias* dos neurónios da camada oculta. Os pesos das ligações entre a camada oculta e o neurónio de saída e a *bias* do neurónio de saída são representados por $\mathbf{w}_O = [w_{1O}, \ldots, w_{hO}]^T$ e b_O, respectivamente. $f(\cdot)$ e $g(\cdot)$, representam a função de activação dos neurónios da camada oculta e da saída, respectivamente.

3.4 Algoritmo de Retro-propagação

A aprendizagem de uma rede neuronal de múltiplas camadas MLP pode ser feita através de aprendizagem supervisionada, onde a rede é treinada com pares de dados de treino de entrada e saída, e em que o objectivo da aprendizagem é que a rede aprenda a realizar um mapeamento de entrada-saída que represente bem os dados de treino. Um algoritmo de aprendizagem supervisionada tipicamente usado para a realização desta aprendizagem é o algoritmo de retro-propagação do erro (*error backpropagation*), que é esquematizado na Figura 3.4, e que consiste nas seguintes duas etapas principais:

1. Com os pesos sinápticos fixos, os dados fluem através da rede sofrendo processamento, camada por camada, até que a resposta seja produzida pela camada de saída;

2. A saída obtida é comparada com a saída desejada. Se esta não estiver correcta, o erro é calculado e (retro-)propagado a partir da camada de saída até à camada de

entrada, procedendo-se ao ajuste dos pesos sinápticos.

3.4.1 Modelo Matemático

Durante a retro-propagação do erro desde a saída até à entrada, todos os pesos são ajustados. Dois casos diferentes de ajuste dos pesos devem ser considerados: (1) o ajuste dos pesos da última camada e (2) o ajuste dos pesos da camada oculta (intermédia). Foi considerada apenas uma camada oculta e um neurónio de saída porque foi o modelo usado neste projecto de dissertação. Para simplificação, foi também considerado que tanto as *bias* da camada oculta, como as da saída são nulas.

Eis alguns pressupostos:

- x_i é a entrada i, com $i = 1, \ldots, n$;
- $w_{i,j}$ é o peso da ligação da entrada i ao neurónio j da camada oculta, com $j = 1, \ldots, h$;
- w_{jO} é o peso da ligação do neurónio j da camada oculta ao neurónio de saída;
- r_j é o sinal efectivo na entrada do neurónio j da camada oculta e é dado por:

$$r_j = \sum_{i=1}^{n} x_i w_{i,j}; \tag{3.11}$$

- $f(\cdot)$ e $g(\cdot)$ são as funções de activação dos neurónios da camada oculta e de saída, respectivamente;
- h_j é o sinal que sai do neurónio j da camada oculta e é dado por:

$$h_j = f(r_j). \tag{3.12}$$

- s é o sinal de entrada efectiva do neurónio de saída e é dado por:

$$s = \sum_{j=1}^{h} h_j w_{jO}; \tag{3.13}$$

- y é a saída da rede neuronal (saída do neurónio de saída) e é dada por:

$$y = g(s). \tag{3.14}$$

A actualização dos pesos das ligações entre os neurónios da camada de entrada e da camada oculta é feita da seguinte forma:

$$w_{i,j}(t+1) = w_{i,j}(t) + \Delta w_{i,j}(t), \tag{3.15}$$

onde $\Delta w_{i,j}(t)$ é o factor de correcção de cada um dos pesos definido por:

$$\Delta w_{i,j}(t) = -\gamma \frac{\partial E}{\partial w_{i,j}}, \tag{3.16}$$

onde $\gamma \in [0, 1]$ é a taxa de aprendizagem do algoritmo de retro-propagação. O sinal negativo indica que são realizadas variações dos pesos ao longo da direcção da máxima descida do gradiente no espaço dos pesos.

Da mesma forma, a actualização dos pesos das ligações entre os neurónios da camada oculta e da saída é feita da seguinte forma:

$$w_{jO}(t+1) = w_{jO}(t) + \Delta w_{jO}(t), \tag{3.17}$$

onde $\Delta w_{jO}(t)$ é o factor de correcção de peso definido por:

$$\Delta w_{jO}(t) = -\gamma \frac{\partial E}{\partial w_{jO}}. \tag{3.18}$$

O parâmetro E representa a função de erro ou de custo a ser minimizada (erro da previsão efetuada pela rede). Neste caso definimos a função de erro como o erro quadrático total:

$$E = E(w) = \sum_{p=1}^{N} (y_d^p - y^p)^2, \tag{3.19}$$

onde N é o número de amostras e y_d é a saída desejada.

O factor de correcção dos pesos pode ser calculado usando a Regra Delta [24] na qual se definem valores de δ para os diversos pesos da rede. O delta relativamente às ligações

entre os neurónios da camada oculta e a saída é definido por:

$$\delta^{(w_{jO})} \equiv -\frac{\partial E}{\partial s}. \tag{3.20}$$

Precisamos calcular $\frac{\partial E}{\partial s}$. Podemos escrever o seguinte:

$$\frac{\partial E}{\partial s} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial s}. \tag{3.21}$$

Pela equação (3.19), o primeiro factor é dado por:

$$\frac{\partial E}{\partial y} = -(y_d - y). \tag{3.22}$$

Pela equação (3.14) e considerando que a função de transferência do neurónio de saída é linear (tal como normalmente é adoptado), o segundo factor é dado por:

$$\frac{\partial y}{\partial s} = 1. \tag{3.23}$$

Então, a equação (3.20) é dada por:

$$\delta^{(w_{jO})} = y_d - y. \tag{3.24}$$

Quanto ao delta relativo às ligações entre os neurónios da camada de entrada e o neurónio j da camada oculta, este pode ser definido como:

$$\delta_j^{(w_{i,j})} \equiv -\frac{\partial E}{\partial r_j}. \tag{3.25}$$

Precisamos calcular $\frac{\partial E}{\partial r_j}$. Podemos escrever o seguinte:

$$\frac{\partial E}{\partial r_j} = \frac{\partial E}{\partial h_j}\frac{\partial h_j}{\partial r_j}. \tag{3.26}$$

Pela equação (3.12) e considerando que a função de activação dos neurónios da camada oculta é uma sigmóide, o segundo factor é dado por:

$$\frac{\partial h_j}{\partial r_j} = h_j(1 - h_j). \tag{3.27}$$

O primeiro factor pode ser escrito da seguinte forma:

$$\frac{\partial E}{\partial h_j} = \frac{\partial E}{\partial s}\frac{\partial s}{\partial h_j}. \tag{3.28}$$

Pela equações (3.13) e (3.20), temos:

$$\frac{\partial E}{\partial h_j} = -\delta^{(w_{jO})} w_{jO}. \tag{3.29}$$

A equação (3.25) pode então ser escrita como:

$$\delta_j^{(w_{i,j})} = h_j(1-h_j)\delta^{(w_{jO})} w_{jO}. \tag{3.30}$$

Tendo-se calculado o valor de $\delta_j^{(w_{jO})}$ e $\delta_j^{(w_{i,j})}$ através das equações (3.24) e (3.30), respectivamente, resta determinar os factores de correcção da seguinte forma:

$$\Delta w_{jO} = -\gamma\frac{\partial E}{\partial w_{jO}} = -\gamma\frac{\partial E}{\partial s}\frac{\partial s}{\partial w_{jO}} = \gamma\delta^{(w_{jO})} h_j. \tag{3.31}$$

$$\Delta w_{i,j} = -\gamma\frac{\partial E}{\partial w_{i,j}} = -\gamma\frac{\partial E}{\partial r_j}\frac{\partial r_j}{\partial w_{i,j}} = \gamma\delta_j^{(w_{i,j})} x_i. \tag{3.32}$$

3.4.2 Algoritmo

O treino da rede por retro-propagação é realizado seguindo os seguintes passos:

1. Gerar aleatoriamente os pesos de saída w_{jO} e os pesos de entrada $w_{i,j}$;
2. Injectar o vector x nos neurónios de entrada;
3. Obter a saída da rede neuronal y;
4. Usando a Equação (3.24) calcular o delta dos pesos w_{jO}, utilizando a saída desejada y_d;
5. Usando a Equação (3.30) calcular os deltas dos pesos $w_{i,j}$ (note-se que aqui o delta da saída é retro-propagado nos deltas da camada oculta);
6. Calcular os novos pesos w_{jO} e $w_{i,j}$ usando (3.31) e (3.32), respectivamente;
7. Repetir os passos (2) a (7) até o critério de paragem ser atingido.

Como critério de paragem podemos usar a norma do erro e/ou a norma do gradiente do erro. Por exemplo, se o erro descer abaixo de um certo limiar pré-definido então considera-se que a rede já atingiu um comportamento satisfatório.

Capítulo 4

Selecção de Variáveis e Respectivos Atrasos

A selecção das variáveis de entrada num problema de aprendizagem de um modelo de predição é definida como o problema da selecção do subconjunto óptimo de entradas, de tal forma que esse subconjunto possa ser utilizado de forma a minimizar o erro de estimação à saída.

No entanto, quando este problema é aplicado em processos industriais, pode adicionalmente existir o problema da selecção do atraso temporal de cada variável. Por exemplo, devido à existência de diversas etapas desde a matéria-prima até ao produto final, as variáveis em diferentes fases do processo vão influênciar a saída não de forma instantânea, mas sim após determinados tempos de atraso. Na selecção de atrasos é seleccionado o tempo em que a variável influencía mais a saída. Noutra perspectiva, as variáveis com os respectivos atrasos podem ser agrupadas num único conjunto de dados e o algoritmo de selecção de variáveis escolhe o melhor subconjunto de todo o conjunto. Esta dissertação adopta esta segunda abordagem: o problema da selecção de variáveis em conjunto com os respectivos tempos de atraso é transformado num problema de selecção de variáveis. Esta hipótese implica aumento de dimensionalidade da entrada e multicolinearidade no espaço das entradas.

Na próxima secção é dada a definição matemática dos problemas de selecção de variáveis e de selecção de atrasos. Na secção 4.2 é apresentado o método de selecção de variáveis desenvolvido.

4.1 Definição Matemática do Problema

O problema da selecção de variáveis e respectivos atrasos será matematicamente formulado a seguir. Para qualquer conjunto de elementos (variáveis ou constantes) $A = \{a_1, \ldots, a_n\}$, com $a_i \in \mathbb{R}$ e $i = 1, \ldots, n$, pode definir-se o operador ν que transforma A num vector $\mathbf{a} = \nu(A) = [a_1, \ldots, a_n]^T$. Reciprocamente, $A = \nu^{-1}(\mathbf{a})$. Nesta tese apenas são considerados conjuntos ordenados.

Considerando que $\mathbf{u}(k) = [u_1(k), \ldots, u_n(k)]^T \in \mathbb{R}^n$ e $y_d(k) \in \mathbb{R}$ com $k = 1, \ldots, N$ são o vector das variáveis de entrada e a saída no instante k, respectivamente, o conjunto de dados é dado por $\mathcal{F}^* = \{\mathbf{u}(k), y_d(k) \mid k = 1, \ldots, N\}$. O conjunto de dados de entrada no instante k é dado por $U^{(k)} = \nu^{-1}(\mathbf{u}(k))$.

Considerando que o problema da selecção de variáveis em conjunto com os respectivos tempos de atraso é transformado num problema de selecção de variáveis, podemos escrever o conjunto de variáveis de entrada como $X^{(k)} = \{x_1(k), \ldots, x_n(k)\} = \{u_1(k - d_1^{(1)}), \ldots, u_1(k - d_{n_1}^{(1)}), \ldots, u_n(k - d_1^{(n)}), \ldots, u_n(k - d_{n_n}^{(n)})\}$, onde $\mathbf{x}(k) = \nu(X^{(k)}) = [x_1(k), \ldots, x_n(k)]^T \in \mathbb{R}^n$ é o vector de entrada reformulado e $d^{(j)} = \{d_1^{(j)}, \ldots, d_{n_j}^{(j)}\}$ é o conjunto dos possíveis tempos de atraso para a variável u_j, onde para todos os j e q, $d_q^{(j)} \in \mathbb{N}_0$ são inteiros não negativos.

O conjunto de dados alterado é dado por $\mathcal{F} = \{(\mathbf{x}(k), y_d(k)) | k = 1, \ldots, N\}$, onde, sem perda de generalidade, se considerou que o número de exemplares permanece inalterado depois de considerar os atrasos possíveis em todas as variáveis de entrada. Ocultando o instante temporal, os dois conjuntos de entrada podem ser definidos por $U = \nu^{-1}(\mathbf{u})$ e $X = \nu^{-1}(\mathbf{x})$.

4.2 CEV-MLP

O objectivo do algoritmo CEV-MLP é optimizar o modelo de predição seleccionando simultaneamente os melhores parâmetros da rede MLP e o melhor subconjunto de variáveis e respectivos atrasos. Esta optimização é efectuada recorrendo a um algoritmo genético co-evolucionário.

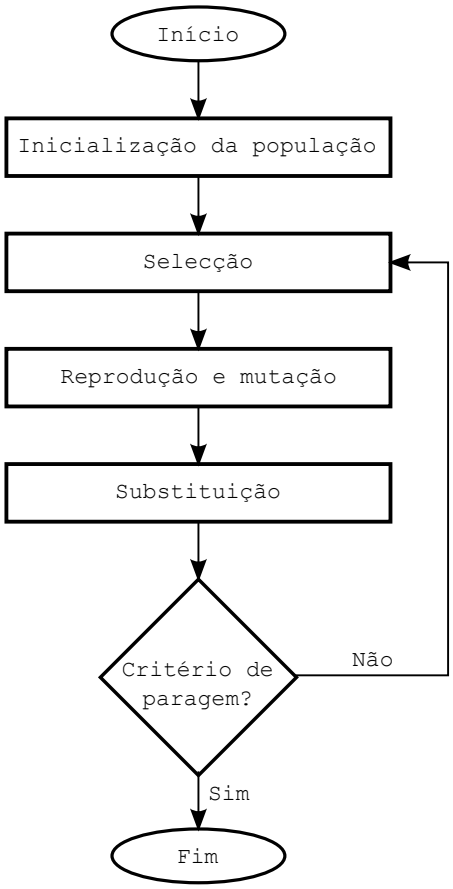


Figura 4.1: Diagrama de fluxo de um algoritmo genético.

4.2.1 Algoritmos Genéticos

Os algoritmos genéticos são métodos de busca e optimização que simulam os processos naturais de evolução, aplicando o princípio da selecção natural de Darwin: as possíveis soluções são codificadas em cromossomas e, tal como na natureza, os cromossomas mais capazes têm mais probabilidade de se reproduzirem, originando descendentes com características de ambos os progenitores. Estes algoritmos foram inicialmente desenvolvidos pelo professor John Holland, da Universidade de Michigan, nos Estados Unidos da América, através dos seus estudos de processos naturais adaptáveis nos finais da década de 1960 e foram formalmente introduzidos em [25].

O diagrama da Figura 4.1 ilustra o funcionamento de um AG. A operação de um AG inicia-se com a geração de uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções do problema. Durante o processo evolucionário, esta população é avaliada e alguns indivíduos são seleccionados para a próxima geração. Os indivíduos que não são seleccionados, são removidos e novos descendentes para a próxima geração são obtidos através de reprodução e mutação dos in-

divíduos seleccionados. A nova população é então utilizada como entrada para a próxima iteração do algoritmo até que a condição de paragem seja satisfeita.

4.2.1.1 Vantagens dos Algoritmos Genéticos

As principais vantagens dos AG são [26]:

- São conceptualmente simples, como se pode verificar na Figura 4.1. Ao longo das diversas iterações, através da selecção, reprodução e mutação são obtidas novas gerações que se aproximam da solução óptima do problema. A eficácia de um AG depende dos operadores aplicados e da inicialização da população inicial;

- Não requerem conhecimento matemático dos problemas a optimizar, ou seja, apenas precisam de avaliar uma função objectivo. As técnicas de busca e optimização tradicionais iniciam-se com um único candidato que, iterativamente, é manipulado utilizando algumas heurísticas directamente associadas ao problema a ser solucionado ou através de uma procura aleatória no espaço de soluções. Ao contrário destes, os AG realizam uma busca adaptando as sucessivas gerações, tentando assim encontrar uma solução óptima;

- Resistentes a serem apanhados em mínimos locais: os AG têm a vantagem de ser menos susceptíveis aos problemas dos mínimos locais, comparado com as técnicas de pesquisa através do gradiente, visto que o operador mutação, geralmente, evita que o AG caia num mínimo local;

- Ampla aplicação: os AG podem ser aplicados a quaisquer problemas que podem ser formulados como problemas de optimização de uma função. Através da representação de possíveis soluções através de estruturas de dados, estes algoritmos podem ser aplicados tanto a problemas de optimização lineares como não-lineares. A sua aplicação também é possível em funções ruidosas e a problemas que não têm solução;

- Facilmente conjugáveis com outros métodos: os AG podem ser facilmente combinados com técnicas de optimização mais tradicionais. Estes podem ser também utilizados para optimizar o desempenho de redes neuronais, sistemas difusos, etc;

- Possibilitam paralelismo: a utilização de computadores de processamento distribuído permitirá um aumento do potencial dos AG. A avaliação de cada solução pode ser tratada como um problema paralelo, permitindo a utilização dos AG em problemas de optimização mais complexos;

- São robustos a alterações dinâmicas: os métodos tradicionais de optimização não são robustos a mudanças dinâmicas na formulação do problema a resolver e requerem um reinício completo para fornecer uma solução. Ao contrário destes, os AG podem ser usados para adaptar as soluções às novas circunstâncias sem ser necessário gerar uma nova população.

4.2.1.2 Desvantagens dos Algoritmos Genéticos

Apesar de os algoritmos genéticos possuírem uma grande quantidade de vantagens, podem-se também citar algumas desvantagens provenientes da utilização desta técnica [26]:

- Necessitam de muito tempo para convergir para a solução óptima: o elevado número de avaliações da função objectivo bem como os operadores usados, podem provocar uma convergência lenta do processo de optimização;

- Não podem ser usados em aplicações em tempo real: o não conhecimento do tempo necessário para a convergência não possibilita a sua utilização em aplicações de tempo real;

- Não garantem uma solução óptima: os AG, tal como os processos estocásticos e de pesquisa aleatória, não garantem que a solução encontrada seja a solução óptima do problema de optimização.

4.2.2 Estrutura do CEV-MLP

Como se pode verificar na Figura 4.2, o CEV-MLP é composto por três níveis hierárquicos: o primeiro é constituído pelos possíveis conjuntos das variáveis de entrada e respectivos atrasos, o segundo é constituído pelas possíveis configurações da camada oculta e o terceiro nível representa a combinação do primeiro nível com o segundo nível, ou seja, o modelo final.

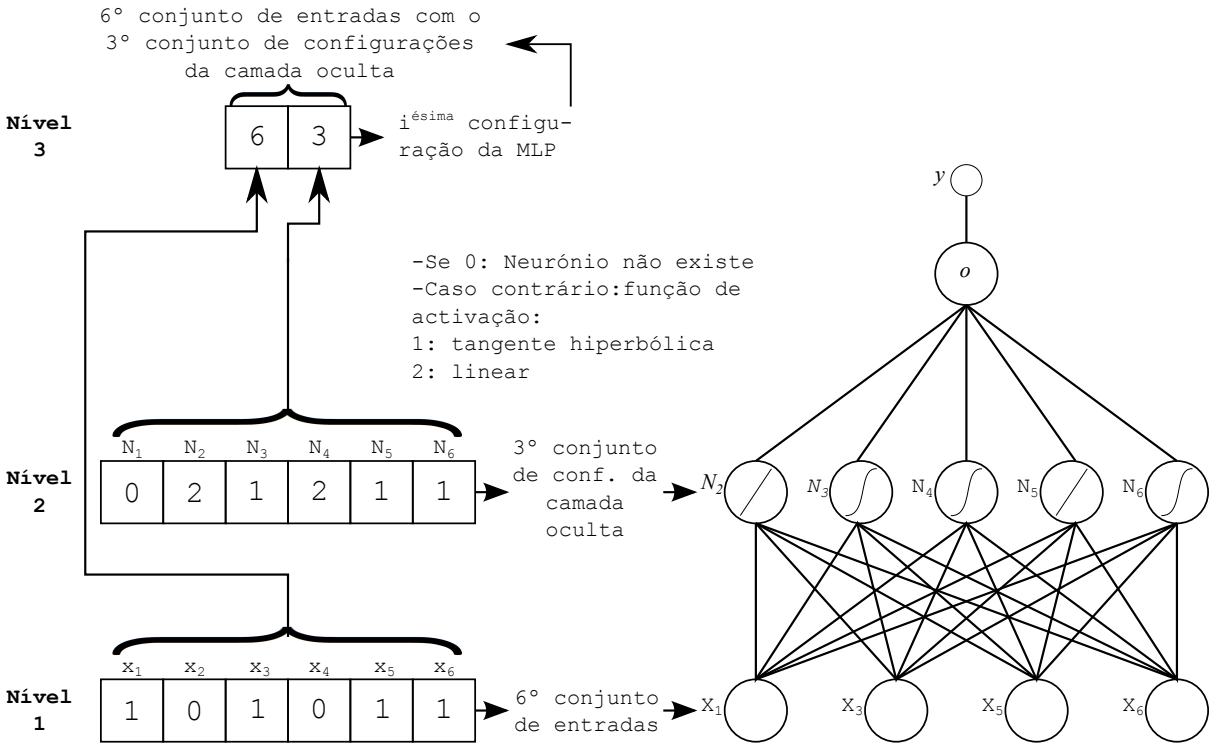


Figura 4.2: Representação gráfica do algoritmo CEV-MLP.

Uma descrição detalhada de cada um dos níveis é apresentada a seguir:

Nível 1 é constituído pelos possíveis conjuntos de variáveis de entrada e os respectivos tempos de atraso que irão ser usados para o desenho do DDSS. O cromossoma é representado através de codificação binária, onde cada alelo (elemento do cromossoma que está localizado numa posição específica) corresponde a cada uma das entradas e ao respectivo atraso (ver Figura 4.2). Em cada um dos alelos, um zero indica que a estrada associada a esse alelo não irá ser considerada.

Nível 2 é constituído pelos possíveis conjuntos de configurações da camada oculta. Cada alelo do cromossoma pode conter valores inteiros entre zero e dois (ver Figura 4.2), sendo que, o valor zero indica que o neurónio da camada oculta associado não irá ser considerado, ou seja, o número de neurónios da camada oculta é dado pelo número de alelos diferentes de zero. Caso o alelo receba o valor um ou dois, isto significa que a função de activação do neurónio da camada oculta associado a este alelo irá ser uma função tangente hiperbólica ou linear, respectivamente.

Nível 3 é constituído pelos possíveis conjuntos de configurações da rede MLP. Cada

cromossoma é constituído por dois alelos onde cada um pode receber um número inteiro positivo, sendo que, o primeiro alelo representa o conjunto/indivíduo do nível 1 e o segundo representa o conjunto/indivíduo do nível 2.

Visto que o nível 3 é constituído pelos conjuntos possíveis de configurações da MLP, o modelo predictor neste nível pode ser definido como $S_3^{(m,l)} = C(S_2^m, S_1^l)$, onde S_2^m e S_1^l são os cromossomas m e l dos níveis 2 e 1, respectivamente, e C é a função que gera o modelo MLP. A saída estimada obtida com os cromossomas S_2^m e S_1^l é designada por $y^{(m,l)}$.

As funções de aptidão para cada indivíduo nos diversos níveis hierárquicos são as seguintes:

- Rede MLP (nível 3):

$$J_3^k = J_3^{(m,l)} = E_{mse}\left(y^{(m,l)}, y_d\right), \tag{4.1}$$

onde k é o número do indivíduo do nível 3 e J_3^k é o erro quadrático médio entre a saída desejada e a saída estimada com os dados de treino. A saída estimada é obtida pelo modelo MLP usando o m-ésimo conjunto de entradas e a l-ésima configuração da camada oculta;

- Configuração da camada oculta (nível 2):

$$J_2^l = \min\left(J_3^{(b,l)}, \ldots, J_3^{(d,l)}\right), \tag{4.2}$$

onde $\{b, \ldots, d\} \subseteq \{1, \ldots, m_{max}\}$ é o subconjunto de cromossomas do nível 1 usados no modelo MLP com a l-ésima configuração da camada oculta;

- Variáveis de entrada (nível 1):

$$J_1^m = \min\left(J_3^{(m,o)}, \ldots, J_3^{(m,q)}\right), \tag{4.3}$$

onde $\{o, \ldots, q\} \subseteq \{1, \ldots, l_{max}\}$ é o subconjunto de cromossomas do nível 2 usados no modelo MLP com o m-ésimo conjunto de variáveis de entrada.

k_{max}, l_{max}, m_{max}, são o número máximo de cromossomas nos níveis 3, 2 e 1, respectivamente. Para não haver redundância nos resultados, $k_{max} \leq m_{max} \cdot l_{max}$, visto que se o

número de cromossomas no terceiro nível for maior que $m_{max} \cdot l_{max}$ iram haver soluções repetidas.

Na Figura 4.2 é apresentado um exemplo da codificação nos diversos níveis da estrutura hierárquica. Neste exemplo, o primeiro alelo do cromossoma i do nível 3 indica que no nível 1 o conjunto de variáveis e respectivos atrasos que irá ser utilizado para construir o modelo é o conjunto número 6, enquanto que o segundo alelo indica que no nível 2 o conjunto de configurações a ser utilizado é o número 3. Focando-nos no cromossoma do segundo nível, podemos verificar que apenas o primeiro alelo do cromossoma é zero e sendo assim, o número de neurónios da camada oculta irá ser 5. Podemos ainda verificar pelos restantes alelos que a função de activação do segundo e quarto neurónios irá ser uma função de activação linear enquanto que os restantes irão ter como função de activação a tangente hiperbólica. Quanto ao nível 1, as variáveis de entrada irão ser as variáveis x_1, x_3, x_5 e x_6.

O algoritmo é composto por três etapas principais. Na primeira etapa do algoritmo, a população dos três níveis é inicializada aleatoriamente. Na etapa seguinte, para todos os cromossomas do nível 3 é determinada a sua aptidão, bem como nos cromossomas utilizados dos níveis 1 e 2. Por fim em todos os níveis os melhores dois cromossomas são escolhidos para serem os progenitores, enquanto que os restantes são substituídos por descendentes obtidos através de reprodução e mutação. As etapas 2 e 3 repetem-se até o método atingir o número máximo de iterações com o mesmo erro N_{max}. Uma descrição mais detalhada do algoritmo pode ser consultada na Secção 4.2.4

4.2.3 Operadores Genéticos

O objectivo básico dos operadores genéticos é de transformar a população através de sucessivas gerações, estendendo a busca até chegar a um resultado satisfatório. Os operadores genéticos são necessários para que a população se diversifique e mantenha as características de adaptação adquiridas pelas gerações anteriores.

Os operadores genéticos usados no algoritmo genético do CEV-MLP são os seguintes:

Inicialização: A população inicial é obtida aleatoriamente com uma distribuição uniforme, apesar de essa distribuição poder afectar o tempo de convergência.

Selecção: O método de selecção usado é a Selecção Elitista [26]. Este método selecciona

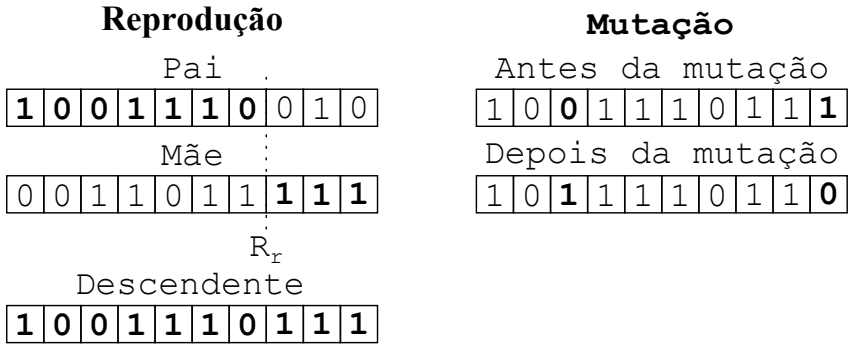


Figura 4.3: Operadores genéticos: *reprodução de um ponto* e *mutação de dois alelos*.

os dois melhores cromossomas para serem os progenitores e os restantes são removidos. Para efectuar a selecção é gerado um número aleatório real R_s entre 0 e 1. Se $R_s < 0.5$ o primeiro cromossoma é usado como pai e o segundo como mãe. Caso contrário, se $R_s \geq 0.5$ o segundo cromossoma é seleccionado como pai e o primeiro como mãe.

Reprodução: A reprodução consiste na técnica de produzir um descendente a partir de dois progenitores [26]. A técnica usada pelo CEV-MLP é a *reprodução single point* ou de *um ponto*. Neste método, todos os cromossomas excepto os progenitores são removidos e novos descendentes são obtidos a partir destes. No processo de reprodução, a probabilidade de um cromossoma se reproduzir com outro é denominada taxa de reprodução (T_r). Para cada descendente, é gerado um número aleatório C_r. Se $C_r > T_r$ não ocorrerá reprodução dos progenitores e o descendente irá ser uma réplica do pai. Caso contrário é gerado um número aleatório R_r e o descendente recebe os primeiros R_r alelos do pai e os restantes da mãe. O processo é ilustrado na Figura 4.3.

Mutação: A mutação é usada para manter a diversidade da população e para prevenir que o algoritmo fique preso num mínimo local. A técnica usada pelo CEV-MLP é a *mutação de dois alelos* (ver Figura 4.3). A probabilidade de um cromossoma sofrer mutação é denominada taxa de mutação (T_m). Para cada novo cromossoma obtido no processo anterior, é gerado um número aleatório C_m. Se $C_m > T_m$ não ocorrerá mutação do cromossoma. Caso contrário o valor de dois alelos aleatoriamente seleccionados do cromossoma são alterados. Para a realização da reprodução e da mutação nos níveis 2 e 3, os cromossomas são convertidos em codificação binária.

4.2.4 Algoritmo

A selecção das melhores entradas e dos respectivos atrasos, tal como a selecção da melhor configuração da MLP é realizada seguindo os seguintes passos:

1. Gerar aleatoriamente a população de todos os níveis;
2. Inicializar i: $i \leftarrow 1$;
3. Inicializar j: $j \leftarrow 1$;
4. Enquanto $j < N_{max}$:
   (a) Para $k = 1, \ldots, k_{max}$: Obter $J_3^k(i)$ usando a Equação (4.1);
   (b) Para todas as configurações da camada oculta utilizadas: Obter $J_2^l(i)$ usando a Equação (4.2);
   (c) Para todas os conjuntos de variáveis de entrada utilizados:: Obter $J_1^m(i)$ usando a Equação (4.3);
   (d) Seleccionar os dois melhores cromossomas em cada um dos níveis 1, 2 e 3 de acordo com $J_1^m(i)$, $J_2^l(i)$ e $J_3^k(i)$ para serem os progenitores.
   Para uma convergência mais rápida, todos os restantes cromossomas são removidos. Através de reprodução e mutação novos cromossomas são obtidos para substituir os removidos;
   (e) Considerando que k^* é o indivíduo mais capaz do nível 3:
   Se $J_3^{k^*}(i) == J_3^{k^*}(i-1)$: $j \leftarrow j + 1$. Caso contrário $j \leftarrow 1$;
   (f) $i \leftarrow i + 1$.

Capítulo 5

Resultados e Discussão

Para validação e verificação da eficiência do método proposto, este foi aplicado a um *dataset* artificial (Secção 5.1.1), a sete *datasets* de referência disponíveis em repositórios públicos (Secção 5.1.2) e a um *dataset* real para estimação da quantidade de flúor no efluente de uma ETAR (Secção 5.1.3). Este foi comparado com um método SBS tradicional usando MLP e com o método proposto em [5]. Contudo, no *dataset* artificial e nos sete *datasets* de referência apenas se procedeu à selecção das variáveis de entrada, ou seja, não se teve em conta possíveis atrasos nas diversas variáveis de entrada. Na Secção 5.2.1 é ainda apresentado um exemplo de aplicação do método proposto na indústria da pasta de papel. Nesta aplicação, o CEV-MLP é utilizado num problema de estimação da CQO, parâmetro fundamental para a qualidade final do papel.

Em todas as experiências foi considerado que a taxa de reprodução e mutação são 80% e 10%, respectivamente, $N_{max} = 30$ e o número máximo de cromossomas em cada nível é $k_{max} = 15$, $m_{max} = 25$ e $l_{max} = 25$.

Foram usadas MLPs com uma única camada oculta, treinada com o método de retropropagação apresentado na Secção 3.4 e utilizando como função de custo o MSE. Foi considerado o número máximo de iterações de treino 1000 e $\gamma = 0.01$. No método proposto na presente dissertação, o número de neurónios da camada oculta e a sua função de activação foram determinados pelo próprio método, sendo apenas necessário indicar o número máximo de neurónios da camada oculta. Nos métodos usados como comparação o número de neurónios foi determinado por validação cruzada *w-folds*, com $w = 5$ e a função de activação empregue foi a função tangente hiperbólica. No método de validação cruzada *w-folds*, o conjunto original de dados é dividido em w subconjuntos. Cada um

desses subconjuntos é então utilizado na validação do modelo, enquanto que dos restantes $(w-1)$ são utilizados no treino do modelo. O número de neurónios foi determinado como sendo o número de neurónios que minimiza a média dos erros de validação obtidos nos w subconjuntos. Quanto à função de activação da camada de saída, em todos os métodos foi usada uma função linear.

O desempenho do SS é medido usando o erro quadrático médio (MSE), o erro quadrático médio normalizado (*normalized mean square error* - NMSE), o erro absoluto médio (MAE) e o coeficiente de correlação (CC) entre a saída desejada e a saída estimada, nos dados de validação e de teste. Os índices de desempenho apresentados nos resultados são a média aritmética simples e o desvio padrão de trinta experiências.

Para a realização das experiências, os dados usados pelos algoritmos foram pré-processados, tendo-lhes sido removidas as entradas constantes. A normalização dos dados foi feita para o intervalo $[-1, 1]$, uma vez que as funções de activação usadas na camada intermédia são lineares ou tangentes hiperbólicas, em que o intervalo de variação das funções é $[-1, 1]$.

Os métodos usados foram implementados em Matlab, usando a toolbox Nnsysid para a realização do treino da rede.

5.1 Validação do CEV-MLP

5.1.1 Aplicação num *Dataset* Artificial: Friedman Artificial Domain

O *dataset* artificial usado nesta secção foi o *dataset* Friedman Artificial Domain (FAD). Num *dataset* artificial, o melhor subconjunto de variáveis de saída é perfeitamente conhecido e, sendo assim, é uma excelente forma de validar o método proposto para a selecção de variáveis de entrada.

O *dataset* Friedman Artificial Domain (disponível em [27]) foi introduzido por Friedman em [28]. Este é constituído por 40768 amostras de dez variáveis de entrada, $U^{(k)} = \{u_1(k), \ldots, u_{10}(k)\}$, e uma saída que apenas depende das cinco primeiras variáveis de entrada. Depois de obtidos os valores de u_i a partir de uma distribuição normal com

Tabela 5.1: Entradas seleccionadas no *dataset* Friedman Artificial Domain.

Método	Variáveis de entrada seleccionadas
SBS	$u_1(k)$, $u_2(k)$, $u_3(k)$, $u_4(k)$
CEV-MLP	$u_1(k)$, $u_2(k)$, $u_3(k)$, $u_4(k)$, $u_5(k)$, $u_6(k)$

Tabela 5.2: Resultados obtidos no *dataset* Friedman Artificial Domain.

Validação		
Medida de desempenho	SBS	CEV-MLP
CC	0.934	**0.979**
MSE	$(12.77 \pm 0.02)e-3$	$\mathbf{(4.20 \pm 0.04)e-3}$
NMSE	$(130.53 \pm 0.25)e-3$	$\mathbf{(42.88 \pm 0.36)e-3}$
MAE	$(92.73 \pm 0.07)e-3$	$\mathbf{(51.63 \pm 0.20)e-3}$
Teste		
Medida de desempenho	SBS	CEV-MLP
CC	0.935	**0.979**
MSE	$(12.89 \pm 0.03)e-3$	$\mathbf{(4.28 \pm 0.04)e-3}$
NMSE	$(128.53 \pm 0.31)e-3$	$\mathbf{(42.51 \pm 0.39)e-3}$
MAE	$(93.54 \pm 0.09)e-3$	$\mathbf{(52.01 \pm 0.24)e-3}$

$u_i \in [0,1]$ e $i = 1, \ldots, 10$, o valor da saída y_d no instante k pode ser obtido por:

$$y_d(k) = 10\sin(\pi u_1(k)u_2(k)) + 20(u_3(k) - 0.5)^2 + 10u_4(k) + 5u_5(k) + \varepsilon, \tag{5.1}$$

onde ε é ruído Gaussiano com média nula e variância unitária.

Os dados foram divididos aleatoriamente em 3/5 para treino, 1/5 para validação e 1/5 para teste. Nos métodos utilizados para comparação, a camada oculta da rede MLP foi constituída por 10 neurónios, enquanto que, no método proposto, foi utilizado como número máximo de neurónios da camada oculta 15 neurónios.

Os resultados da aplicação do método proposto e do método SBS no *dataset* FAD são apresentados nas Tabelas 5.1 e 5.2. Para este mesmo *dataset* não foi possível utilizar o método proposto por [5] devido ao elevado número de amostras. Este algoritmo implica a construção de uma matriz $\mathbf{R} \in \mathbb{R}^{N \times N}$, onde N é o número de amostras, o que não é possível devido a limitações de memória.

Analisando os resultados é possível verificar que o método proposto selecciona uma variável irrelevante para além das cinco variáveis relevantes. O comportamento não é o perfeitamente desejado, mas, contudo, é substancialmente melhor que o comportamento

Tabela 5.3: Descrição dos *datasets* de referência.

Dataset	N	n	h	h_{cev}	P_{tr}	P_{va}	P_{te}
Box and Jenkins	290	10	4	8	3/5	1/5	1/5
Automobile MPG	392	6	2	5	1/3	1/3	1/3
Concrete Compressive Strength	1030	8	10	15	3/5	1/5	1/5
Boston Housing	506	13	2	5	3/5	1/5	1/5
Triazines	186	59	12	18	3/5	1/5	1/5
Parkinson Telemonitoring	5875	26	14	20	3/5	1/5	1/5
Ailerons	7154	39	18	25	3/5	1/5	1/5

apresentado pelo algoritmo SBS. A não selecção de uma variável relevante pelo método SBS vai provocar uma diminuição substancial do desempenho do modelo tanto na fase de validação como de teste em todas as medidas de desempenho. Através da análise dos resultados, foi também possível verificar que a melhor configuração da camada oculta obtida pelo CEV-MLP é constituída por 14 neurónios, dos quais 3 têm uma função de transferência linear e os restantes têm como função de transferência a função tangente hiperbólica.

5.1.2 Aplicação em *Datasets* de Referência

Para a validação do método proposto, foram também utilizados os seguinte sete *datasets* disponíveis em repositórios públicos: Box and Jenkins [29], Automobile MPG [30], Concrete Compressive Strength [31], Boston Housing [32], Triazines [33], Parkinson Telemonitoring [34] e Ailerons [35].

Uma breve descrição destes *datasets* pode ser encontrada na Tabela 5.3, onde N e n representam o número de amostras e de entradas, respectivamente, h e h_{cev} representam o número de neurónios da camada oculta nos métodos usados para comparação e o número máximo de neurónios da camada oculta no método CEV-MLP, respectivamente, e P_{tr}, P_{va} e P_{te} representam a proporção da divisão aleatória do conjunto de dados em dados de treino, validação e teste, respectivamente.

Nas Tabelas 5.4 e 5.5 são apresentados, respectivamente, o número de variáveis seleccionadas em cada método e a configuração da camada oculta obtida pelo método proposto nos diversos *datasets* de referência, sendo que h_l e h_{th} representam o número de neurónios da camada oculta seleccionados pelo método proposto cuja função de activação é linear e o número de neurónios cuja função de activação é uma função tangente hiperbólica,

Tabela 5.4: Número de variáveis de entrada seleccionadas nos *datasets* de referência.

Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	3	4	8
Automobile MPG	3	2	4
Concrete Compressive Strength	3	1	7
Boston Housing	3	1	11
Triazines	8	51	27
Parkinson Telemonitoring	1	13	12
Ailerons	2	22	15

Tabela 5.5: Configurações da camada oculta obtida pelo CEV-MLP nos *datasets* de referência.

Dataset	$h_l + h_{th}$	h_l	h_{th}
Box and Jenkins	6	2	4
Automobile MPG	5	0	5
Concrete Compressive Strength	12	2	10
Boston Housing	5	1	4
Triazines	15	3	12
Parkinson Telemonitoring	13	4	9
Ailerons	22	7	15

Tabela 5.6: Resultados obtidos pelos três métodos nos diversos *datasets* utilizando como medida de desempenho o MSE nos dados de validação e de teste ($\times 10^{-3}$).

Validação			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	2.11 ± 0.06	1.81 ± 0.11	$\mathbf{1.62 \pm 0.14}$
Automobile MPG	26.85 ± 0.30	26.46 ± 0.28	$\mathbf{23.15 \pm 0.52}$
Concrete Compressive Strength	94.51 ± 4.34	129.25 ± 1.69	$\mathbf{29.41 \pm 2.65}$
Boston Housing	30.35 ± 0.59	53.12 ± 0.49	$\mathbf{20.43 \pm 3.58}$
Triazines	127.61 ± 15.09	213.14 ± 57.72	$\mathbf{125.58 \pm 30.36}$
Parkinson Telemonitoring	265.13 ± 0.67	$\mathbf{212.26 \pm 27.27}$	217.71 ± 22.47
Ailerons	35.40 ± 0.07	9.72 ± 0.12	$\mathbf{9.57 \pm 0.06}$
Teste			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	$\mathbf{1.47 \pm 0.05}$	1.95 ± 0.20	1.87 ± 0.27
Automobile MPG	37.53 ± 0.74	37.83 ± 0.61	$\mathbf{35.24 \pm 1.28}$
Concrete Compressive Strength	133.14 ± 4.33	124.85 ± 3.45	$\mathbf{27.58 \pm 2.31}$
Boston Housing	42.34 ± 0.86	44.09 ± 0.35	$\mathbf{30.98 \pm 3.86}$
Triazines	185.07 ± 19.46	192.40 ± 42.23	$\mathbf{179.53 \pm 48.37}$
Parkinson Telemonitoring	266.07 ± 0.71	$\mathbf{213.94 \pm 27.87}$	216.91 ± 22.65
Ailerons	34.59 ± 0.07	10.93 ± 0.09	$\mathbf{10.77 \pm 0.09}$

respectivamente.

Os resultados da aplicação dos três métodos para as medidas de desempenho referidas anteriormente estão presentes nas Tabelas 5.6, 5.7, 5.8 e 5.9.

Tabela 5.7: Resultados obtidos pelos três métodos nos diversos *datasets* utilizando como medida de desempenho o NMSE nos dados de validação e de teste ($\times 10^{-3}$)

Validação			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	11.15 ± 0.29	9.54 ± 0.63	$\mathbf{8.57 \pm 0.72}$
Automobile MPG	127.91 ± 1.42	126.03 ± 1.32	$\mathbf{110.28 \pm 2.48}$
Concrete Compressive Strength	527.90 ± 24.29	721.94 ± 9.45	$\mathbf{164.30 \pm 14.81}$
Boston Housing	162.83 ± 3.14	284.98 ± 2.63	$\mathbf{109.62 \pm 19.22}$
Triazines	786.02 ± 92.93	1312.85 ± 355.56	$\mathbf{773.51 \pm 187.01}$
Parkinson Telemonitoring	1193.01 ± 3.00	$\mathbf{955.09 \pm 122.71}$	979.61 ± 101.12
Ailerons	534.56 ± 1.08	146.83 ± 1.86	$\mathbf{144.47 \pm 0.95}$
Teste			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	$\mathbf{7.56 \pm 0.23}$	9.99 ± 1.02	9.60 ± 1.39
Automobile MPG	170.78 ± 3.35	172.12 ± 2.79	$\mathbf{160.37 \pm 5.84}$
Concrete Compressive Strength	675.93 ± 21.99	633.84 ± 17.55	$\mathbf{140.04 \pm 11.73}$
Boston Housing	286.70 ± 5.82	298.56 ± 2.36	$\mathbf{209.80 \pm 26.16}$
Triazines	1256.98 ± 132.14	1306.74 ± 286.83	$\mathbf{1219.32 \pm 328.49}$
Parkinson Telemonitoring	1198.94 ± 3.20	$\mathbf{964.03 \pm 125.56}$	977.43 ± 102.07
Ailerons	541.44 ± 1.20	171.05 ± 1.40	$\mathbf{168.55 \pm 1.34}$

Tabela 5.8: Resultados obtidos pelos três métodos nos diversos *datasets* utilizando como medida de desempenho o MAE nos dados de validação e de teste ($\times 10^{-3}$).

Validação			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	28.28 ± 0.65	28.14 ± 0.77	$\mathbf{26.87 \pm 1.26}$
Automobile MPG	115.94 ± 1.27	114.71 ± 0.71	$\mathbf{109.89 \pm 1.29}$
Concrete Compressive Strength	255.03 ± 5.22	285.15 ± 3.43	$\mathbf{130.71 \pm 7.41}$
Boston Housing	109.36 ± 4.26	178.49 ± 0.31	$\mathbf{107.45 \pm 6.47}$
Triazines	259.32 ± 16.65	328.50 ± 47.58	$\mathbf{249.91 \pm 23.65}$
Parkinson Telemonitoring	429.06 ± 0.66	$\mathbf{385.28 \pm 27.62}$	391.13 ± 22.31
Ailerons	144.23 ± 0.20	72.68 ± 0.42	$\mathbf{71.91 \pm 0.21}$
Teste			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	27.30 ± 0.57	$\mathbf{27.01 \pm 1.24}$	27.35 ± 2.00
Automobile MPG	142.64 ± 1.55	144.20 ± 1.47	$\mathbf{139.97 \pm 2.23}$
Concrete Compressive Strength	303.91 ± 3.72	283.30 ± 5.87	$\mathbf{121.00 \pm 6.63}$
Boston Housing	130.71 ± 2.11	166.11 ± 0.86	$\mathbf{113.15 \pm 5.00}$
Triazines	321.77 ± 15.90	322.37 ± 32.13	$\mathbf{305.95 \pm 28.59}$
Parkinson Telemonitoring	429.52 ± 0.76	$\mathbf{386.11 \pm 27.87}$	389.00 ± 22.65
Ailerons	141.89 ± 0.19	75.26 ± 0.30	$\mathbf{75.24 \pm 0.30}$

Os resultados obtidos demonstram que em alguns *datasets* o método proposto selecciona o maior subconjunto de variáveis de entrada. Contudo, este subconjunto é o mais adequado em quase todos os *datasets*, relativamente aos subconjuntos seleccionados pelos restantes dois métodos. As únicas excepções observam-se no *dataset* Parkinson Telemonitoring cujo subconjunto mais adequado foi obtido pelo método SBS e no *dataset* Box and Jenkins onde o método apresentado em [5] obteve melhores resultados nos dados de teste e utilizando como medida de desempenho o MSE, o NMSE e o CC. Neste último

Tabela 5.9: Resultados obtidos pelos três métodos nos diversos *datasets* utilizando como medida de desempenho o CC entre a saída desejada e a saída estimada nos dados de validação e de teste.

Validação			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	0.994	0.995	**0.996**
Automobile MPG	0.934	0.935	$\mathbf{0.943 \pm 0.001}$
Concrete Compressive Strength	0.700 ± 0.018	0.531 ± 0.011	$\mathbf{0.922 \pm 0.007}$
Boston Housing	0.915 ± 0.002	0.847 ± 0.001	$\mathbf{0.945 \pm 0.010}$
Triazines	0.611 ± 0.038	0.331 ± 0.125	$\mathbf{0.626 \pm 0.055}$
Parkinson Telemonitoring	0.267 ± 0.004	$\mathbf{0.582 \pm 0.046}$	0.551 ± 0.046
Ailerons	0.683	0.924	**0.925**
Teste			
Dataset	Li & Peng [5]	SBS	CEV-MLP
Box and Jenkins	**0.996**	0.995	0.995
Automobile MPG	0.911 ± 0.002	0.910 ± 0.002	$\mathbf{0.917 \pm 0.003}$
Concrete Compressive Strength	0.611 ± 0.018	0.625 ± 0.017	$\mathbf{0.929 \pm 0.006}$
Boston Housing	0.846 ± 0.003	0.840 ± 0.001	$\mathbf{0.892 \pm 0.013}$
Triazines	0.296 ± 0.108	0.285 ± 0.130	$\mathbf{0.355 \pm 0.121}$
Parkinson Telemonitoring	0.255 ± 0.003	$\mathbf{0.578 \pm 0.048}$	0.557 ± 0.046
Ailerons	0.678	**0.913**	**0.913**

dataset e utilizando a medida de desempenho MAE, é possível verificar também que para os dados de teste o melhor método é o SBS. Este método apresenta também resultados semelhantes ao algoritmo proposto no *dataset* Ailerons, considerando como medida de performance o CC nos dados de teste.

5.1.3 Estimação da Quantidade de Flúor

Nesta secção apresentam-se os resultados obtidos com o conjunto de dados em que a variável alvo a estimar é a quantidade de flúor no efluente de uma estação de tratamento de águas residuais (ETAR). Os dados utilizados são compostos por 11 entradas, $U^{(k)} = \{u_1(k), \ldots, u_{11}(k)\}$, e uma saída a ser estimada y_d. Os valores da variável de saída foram obtidos através de análises laboratoriais. Relativamente às variáveis de entrada, cada variável é obtida a partir de medições por sensores físicos. As variáveis do *dataset* correspondem a grandezas físicas tais como o pH, e a turvação e cor da água (ver Tabela 5.10). O *dataset* é composto por 360 amostras obtidas com um período de amostragem de 2 horas. Considerando os possíveis tempos de atraso $d = \{0, 1, 2\}$ para todas as variáveis de entrada, obtém-se o seguinte conjunto de entradas: $X^{(k)} = \{u_1(k), u_1(k-1), u_1(k-2), \ldots, u_{11}(k), u_{11}(k-1), u_{11}(k-2)\}$.

O conjunto de dados foi dividido aleatoriamente em 3/5, 1/5, e 1/5 para dados de

Tabela 5.10: Variáveis de entrada utilizadas para a estimação da quantidade de flúor.

Variáveis	Descrição
u_1	Quantidade de cloro no afluente (esgoto bruto);
u_2	Quantidade de cloro no efluente (esgoto tratado);
u_3	Turvação da água no afluente (água bruta);
u_4	Turvação no afluente;
u_5	Turvação no efluente;
u_6	pH da água no afluente;
u_7	pH no afluente;
u_8	pH no efluente;
u_9	Cor da água no afluente;
u_{10}	Cor do afluente;
u_{11}	Cor do efluente;
y_d	Flúor no efluente.

Tabela 5.11: Entradas seleccionadas na estimação da quantidade de flúor.

Método	Variáveis de entrada seleccionadas
Li & Peng [5]	$u_7(k)$, $u_7(k-1)$
SBS	$u_4(k)$, $u_1(k-1)$, $u_8(k-2)$
CEV-MLP	$u_1(k)$, $u_3(k)$, $u_7(k)$, $u_8(k)$, $u_{10}(k)$, $u_{11}(k)$, $u_1(k-1)$, $u_2(k-1)$, $u_4(k-1)$, $u_7(k-1)$, $u_3(k-2)$, $u_{11}(k-2)$

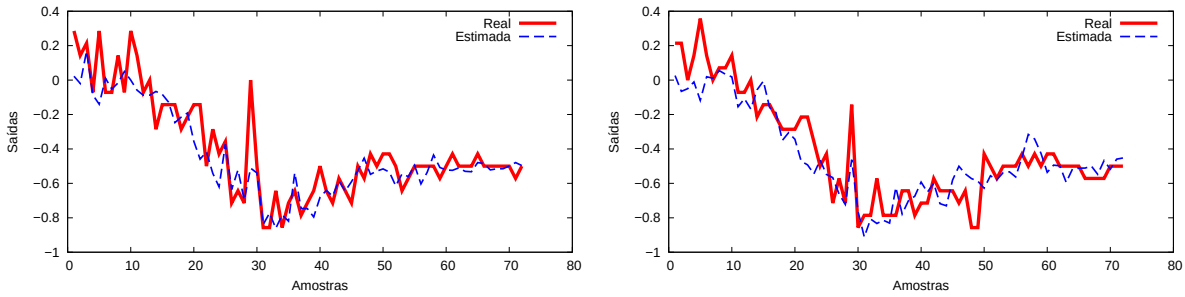
treino, validação e teste, respectivamente. O número máximo de neurónios da camada oculta possíveis para o algoritmo CEV-MLP foi 20. Nos restantes métodos, usando a metodologia de validação cruzada apresentada no início deste capítulo, e considerando todas as variáveis de entrada, foi determinado e fixado em 15 o número de neurónios da camada oculta. Este foi considerado o melhor número de neurónios para esses outros dois métodos.

As Tabelas 5.11 e 5.12 apresentam os resultados obtidos pelos três algoritmos na estimação do flúor na ETAR. O algoritmo proposto é o algoritmo que selecciona o maior número de variáveis de entrada. Contudo este algoritmo apresenta os melhores resultados em todas as medidas de desempenho, logo seguido do algoritmo SBS. Quanto à configuração da camada oculta, foi seleccionada a configuração composta por 18 neurónios, 2 dos quais com função de activação linear e os restantes com função de transferência tangente hiperbólica.

Na Figura 5.1 são representadas a saída desejada e a saída estimada pelo algoritmo CEV-MLP, sendo que a saída estimada corresponde à média de trinta experiências com as

Tabela 5.12: Resultados obtidos na estimação da quantidade de flúor.

Validação			
Medida de desempenho	Li & Peng [5]	SBS	CEV-MLP
CC	0.742 ± 0.004	0.872 ± 0.005	$\mathbf{0.888 \pm 0.011}$
MSE	$(44.21 \pm 0.92)e-3$	$(22.37 \pm 0.80)e-3$	$\mathbf{(19.62 \pm 1.84)e-3}$
NMSE	$(524.36 \pm 10.97)e-3$	$(265.36 \pm 9.51)e-3$	$\mathbf{(232.73 \pm 21.87)e-3}$
MAE	$(155.46 \pm 1.17)e-3$	$(104.09 \pm 1.83)e-3$	$\mathbf{(98.48 \pm 4.46)e-3}$
Teste			
Medida de desempenho	Li & Peng [5]	SBS	CEV-MLP
CC	0.734 ± 0.005	0.874 ± 0.004	$\mathbf{0.885 \pm 0.008}$
MSE	$(44.35 \pm 0.98)e-3$	$(21.67 \pm 0.55)e-3$	$\mathbf{(20.35 \pm 1.33)e-3}$
NMSE	$(496.96 \pm 10.90)e-3$	$(241.78 \pm 6.12)e-3$	$\mathbf{(227.10 \pm 14.89)e-3}$
MAE	$(152.23 \pm 1.83)e-3$	$(107.75 \pm 2.32)e-3$	$\mathbf{(107.73 \pm 4.34)e-3}$



(a) Validação (b) Teste

Figura 5.1: Saída desejada e estimada pelo algoritmo CEV-MLP na estimação da quantidade de flúor.

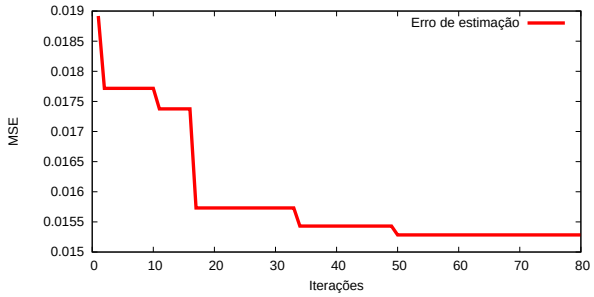


Figura 5.2: Evolução do MSE do melhor indivíduo nos dados de validação no algoritmo CEV-MLP para a estimação da quantidade de flúor.

variáveis de entrada e configuração da camada oculta seleccionadas pelo método proposto. Por sua vez, a Figura 5.2 mostra a evolução do erro quadrático médio (MSE) do indivíduo mais capaz do terceiro nível nos dados de validação, enquanto o algoritmo executa a optimização do modelo da rede e a selecção das melhores entradas e atrasos. Como é possível verificar, ao fim de 80 iterações o algoritmo alcançou a condição de paragem e

devolveu a melhor solução possível encontrada na iteração 50.

5.2 Aplicação do CEV-MLP

5.2.1 Estimação da CQO

A produção da pasta de papel é um processo complexo cujo objectivo é a separação das fibras de celulose para a produção de uma pasta de papel limpa e com qualidade uniforme. Estas fibras são separadas dissolvendo a lignina através de processos químicos.

O principal processo químico para a obtenção da pasta de papel é o *Kraft*. O *Kraft* consiste em produzir pasta de papel a partir de aparas de madeira em reservatórios sob pressão na presença de uma solução alcalina que reage com a lignina da madeira [36]. O processo de Kraft pode ser dividido em cinco etapas:

Preparação da Madeira: A madeira é recebida e é preparada. Depois de retirada a casca, a madeira é reduzida a lascas com dimensões compreendidas entre 2 e 6mm. Nesta etapa também é feita a separação das madeiras consoante a dureza das mesmas, visto que os tempos da cozedura dependem deste factor.

Digestor: As lascas de madeira são cozidas em conjunto com uma solução alcalina de hidróxido de sódio, hidrossulfeto de sódio e água, cujo objectivo é a dissolução da lignina, provocando assim a libertação das fibras de celulose.

Triagem: São retirados da polpa possíveis nós, lascas, ou outros detritos através de peneiras e centrifugação.

Lavagem e deslignificação com oxigénio: Através de 3-4 lavagens as fibras de celulose são separadas da solução alcalina usada no digestor. A pasta é ainda sujeita a uma deslignificação com oxigénio para remoção da lignina que possa não ter sido dissolvida no digestor.

Branqueamento: Se a pasta se destinar a produção de papel branco, esta precisa de ser branqueada recorrendo a produtos químicos.

No processo de produção da pasta de papel existe um factor/variável muito importante denominado/a carência química de oxigénio (CQO). Uma má cozedura no digestor vai

provocar uma má lavagem. Uma má lavagem vai provocar uma maior necessidade de químicos no processo de deslignificação com oxigénio e, sendo assim, vai existir um grande valor de CQO no processo de branqueamento. O controlo da CQO possibilitaria uma redução da quantidade de químicos durante o processo, reduzindo assim os custos da produção. Contudo esta variável não pode ser medida em tempo real, podendo apenas ser medida através de análises laboratoriais. Para contornar este factor podemos recorrer a um SS, estimando assim a CQO através de outros sensores físicos disponíveis no processo.

Os dados utilizados para a estimação da CQO são compostos por 27 entradas, $U^{(k)} = \{u_1(k), \ldots, u_{27}(k)\}$, e uma saída a ser estimada y_d. Cada uma das vinte e sete variáveis corresponde a 10186 medições obtidas por sensores físicos durante a produção da pasta de papel com um período de amostragem de 1 hora. Os valores da saída desejada y_d (CQO) foram obtidos recorrendo a análises laboratoriais. Contudo essas análises laboratoriais não foram realizadas a todas as horas, originando assim que apenas 620 amostras possam ser utilizadas no processo de aprendizagem.

Principalmente devido a problemas nos sensores físicos, os dados para poderem ser usados no processo de aprendizagem do modelo precisam de ser pré-processados. Na Figura 5.3 é apresentado um diagrama esquemático do processo de pré-processamento dos dados. A primeira etapa do pré-processamento consiste na remoção das variáveis constantes e na substituição dos dados corrompidos pela média da respectiva variável de entrada. Na fase seguinte, cada uma das variáveis foi suavizada recorrendo a uma média móvel de 20 pontos. Por último, através da regra 3σ, foi realizada a detecção de *outliers* e procedeu-se à normalização de cada variável de entrada e da saída para o intervalo $[-1, 1]$. Após este processo os dados estão prontos para poderem ser utilizados, restando apenas 14 possíveis entradas $U^{(k)} = \{u_1(k), \ldots, u_{14}(k)\}$, e uma saída a ser estimada y_d, cada uma composta por 620 amostras.

Considerando os possíveis tempos de atraso $d = \{0, \ldots, 8\}$ para todas as variáveis de entrada, temos que o conjunto de entradas é dado por: $X^{(k)} = \{u_1(k), \ldots, u_1(k-8), \ldots, u_{14}(k), \ldots, u_{14}(k-8)\}$.

Os dados de treino, validação e teste foram obtidos dividindo aleatoriamente os conjuntos de dados na proporção de 3/5, 1/5 e 1/5, respectivamente. Quanto à configuração da camada oculta foi considerado que o número máximo de neurónios era 30.

Na Figura 5.4 são representadas a saída desejada e a saída estimada nos dados de

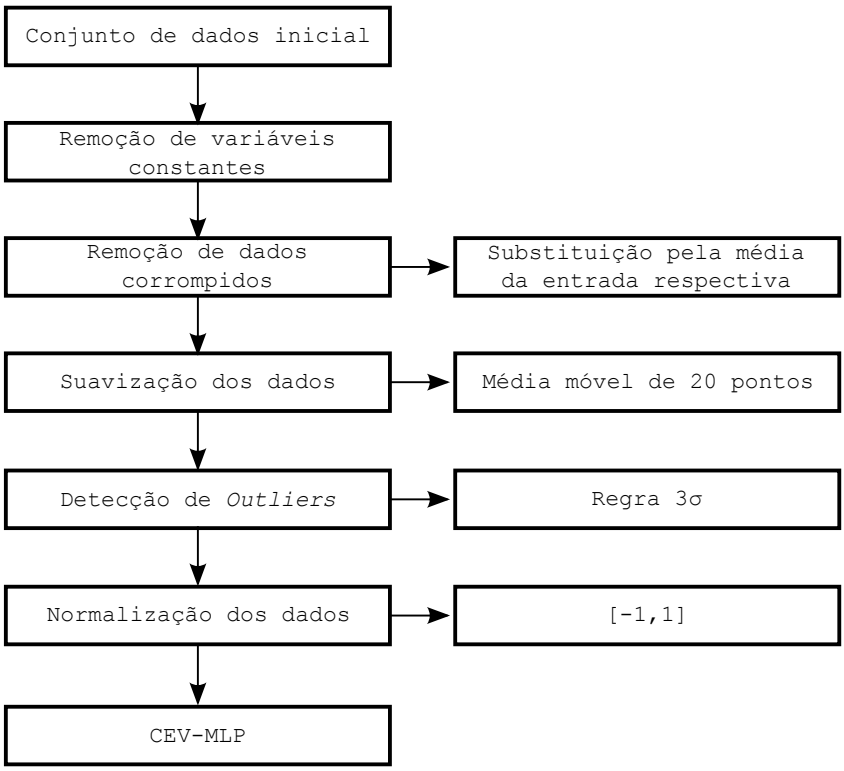
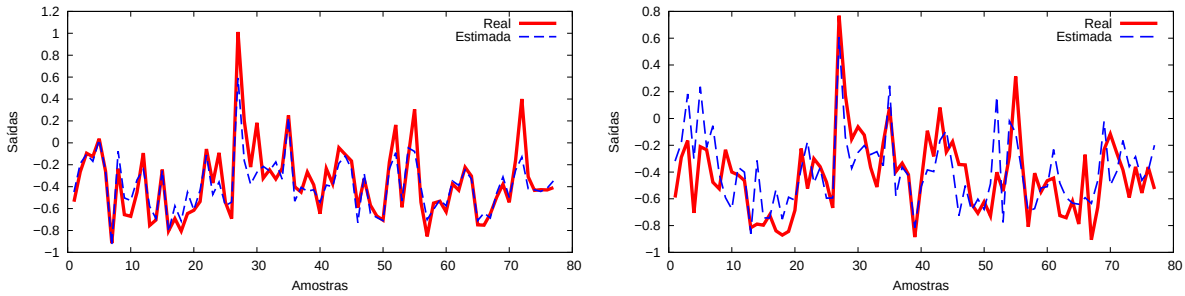


Figura 5.3: Diagrama esquemático do pré-processamento realizado aos dados de entrada na estimação da CQO.



(a) Validação (b) Teste

Figura 5.4: Saída desejada e estimada pelo algoritmo CEV-MLP na estimação da CQO.

validação e teste. A saída estimada corresponde à média da saída de trinta experiências com as variáveis de entrada e configuração da camada oculta seleccionadas pelo método proposto. A evolução do MSE do indivíduo mais apto do terceiro nível nos dados de validação pode ser observada na Figura 5.5. Como descrito anteriormente, o algoritmo partiu de um total de 126 pares de (variáveis de entrada, tempos de atraso) e 30 neurónios como o número máximo de neurónios na camada oculta. Ao fim de 12 iterações, o algoritmo alcançou a sua melhor solução, tendo escolhido 55 pares de (variáveis de entrada, tempos de atraso) e uma configuração da camada oculta composta por 25 neurónios, 7

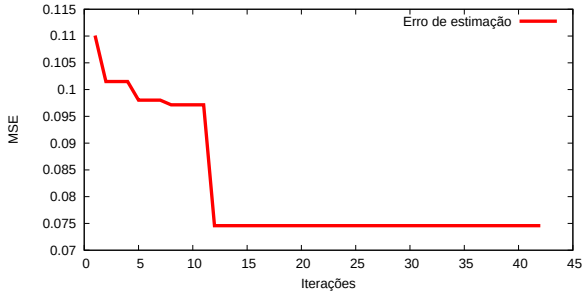


Figura 5.5: Evolução do MSE do melhor indivíduo nos dados de validação para a estimação da CQO.

Tabela 5.13: Resultados obtidos na estimação da CQO.

Medida de desempenho	Validação	Teste
MSE	$21.42e-3$	$50.86e-3$
NMSE	$202.65e-3$	$591.42e-3$
MAE	$99.57e-3$	$186.03e-3$
CC	0.913	0.686

dos quais com função de activação linear e os restantes com função de activação tangente hiperbólica.

Os resultados do desempenho obtido pelo método CEV-MLP podem ser consultados na Tabela 5.13. Neste caso, os resultados para as diversas medidas de desempenho têm em conta a saída média das trinta experiências e não a média e o desvio padrão do erro de cada uma das trinta experiências. Como seria de esperar, os resultados nos dados de validação são bastante melhores do que os resultados obtidos com os dados de teste. Contudo, a partir da análise da Figura 5.4 e dos resultados desta mesma tabela podemos verificar que, mesmo nos dados de teste, o SS desenhado é um bom estimador da CQO e poderia ser implementado no processo de produção de pasta de papel.

5.3 Discussão dos Resultados

Para a validação do método proposto este foi inicialmente aplicado a um *dataset* artificial denominado Friedman Artificial Domain. Comparando o método CEV-MLP com um método SBS tradicional, verificou-se que o desempenho do novo método era bastante melhor, visto não ter removido nenhuma variável relevante para a obtenção de uma boa estimação. A não-remoção de nenhuma variável relevante pelo método proposto originou uma redução de 66.7% do MSE, comparativamente ao método SBS, no conjunto de dados

de teste. Contudo a não remoção de uma variável irrelevante originou que o desempenho da selecção de variáveis não tenha sido perfeito. No que diz respeito ao algoritmo SBS pensa-se que a má decisão de remoção da variável $u_5(k)$ se deva ao ruído próprio das redes MLP. Numa rede MLP dois treinos com as mesmas variáveis de entrada podem levar à obtenção de modelos diferentes, dependendo esse treino, por exemplo, dos pesos iniciais das ligações. Desta problemática pode resultar a remoção de variáveis importantes, sendo que o método proposto é menos propício a que tal problema interfira significativamente nos resultados finais, uma vez que uma variável de entrada nunca é definitivamente removida.

Foram também realizadas experiências comparativas em sete *datasets* de referência disponíveis em repositórios públicos. A partir da análise dos resultados obtidos é possível verificar que o método proposto, na maioria dos *datasets* e das medidas de desempenho, apresenta melhores resultados comparativamente com o método SBS e com o método proposto em [5]. Destacam-se os resultados obtidos nos *datasets* Concrete Compressive Strength e Boston Housing, onde no primeiro *dataset* o método proposto originou uma redução do MSE em 79.3% e 77.9% relativamente ao método proposto em [5] e ao método SBS, respectivamente. No segundo *dataset* o método proposto originou uma redução do MSE em 26.8% e 29.7% relativamente ao mesmos dois métodos. O método proposto nesta dissertação incorpora não só a selecção do melhor subconjunto de variáveis de entrada, como também o desenho do modelo que melhor se adequa às variáveis de entrada seleccionadas, demonstrando assim ser esta a melhor abordagem.

Na experiência realizada para a estimação da quantidade de flúor no efluente de uma ETAR, mais uma vez o método proposto demonstrou ser uma boa ferramenta para o projecto do SS. Apesar de ter seleccionado o maior número de pares (variáveis de entrada, tempos de atraso), estas demonstraram ser as mais adequadas para uma boa estimação da saída, permitindo uma redução do MSE em 51.1% relativamente ao método proposto em [5] e 6.1% relativamente ao método SBS nos dados de teste.

Depois de validado o algoritmo, este foi aplicado a um problema de estimação da CQO num processo de produção de pasta de papel. A utilização de um *dataset* tão complexo envolveu um pré-processamento muito cuidado dos dados devido à existência de muitas medidas corrompidas e inconsistentes. Depois de aplicado o método CEV-MLP foi obtido um SS que permite uma boa estimação da variável pretendida.

Capítulo 6

Conclusões e Trabalho Futuro

6.1 Conclusões

Esta dissertação apresentou um novo método para o projecto de SS. Este método efectua automaticamente o projecto da rede MLP, assim como selecciona o melhor conjunto de variáveis de entrada e os respectivos tempos de atraso. O método proposto não exige qualquer conhecimento prévio sobre o modelo nem sobre as melhores variáveis de entrada, sendo apenas necessário o conhecimento empírico (dados) das entradas e da saída.

Para validar e demonstrar o desempenho e eficácia do método proposto, o algoritmo foi aplicado, juntamente com outros dois métodos de selecção de variáveis, a um *dataset* artificial, a sete *datasets* reais disponíveis em repositórios públicos e a um problema real da estimação da quantidade de flúor no efluente de uma ETAR. Através dos resultados experimentais é possível verificar a eficácia do método proposto. O algoritmo CEV-MLP para além de apresentar o melhor desempenho de estimação na maioria dos *datasets*, permite ainda que o modelo se adeque durante o processo de selecção dos melhores pares de (variáveis, tempos de atraso). Ao contrário dos outros métodos cujo número de neurónios se mantém fixo, a estrutura co-evolucionária permite que a configuração da camada oculta se vá ajustando às variáveis de entrada seleccionadas.

Depois de validado, o método proposto foi aplicado a um problema de estimação da CQO num processo de produção de pasta de papel. A selecção do melhor subconjunto de variáveis de entrada/tempo de atraso permitiu uma redução do tempo computacional na fase de aprendizagem do modelo e permitiu um aumento da generalização do SS, comparativamente a um modelo treinado com todo o conjunto de variáveis de entrada.

Pode-se então concluir que, dos três métodos estudados, o método proposto é o melhor para realizar o projecto do SS.

6.2 Trabalho Futuro

Muitos processos industriais são caracterizados por frequentes alterações nas condições de funcionamento, provocando assim erros de estimação do SS. Para garantir a qualidade desejada no produto final, um SS integrado num sistema de controlo deve ser capaz de lidar com frequentes alterações do modelo e condições de funcionamento. Uma forma de atingir esse objectivo é adaptar o modelo através do *feedback* da medição do desempenho do sistema. Uma solução desejável passa pela adaptação do modelo sem ser necessário um novo treino do mesmo.

Outra possível melhoria poderia passar pela realização da optimização do SS sem realizar o retreino da rede, visto que esta etapa é a mais dispendiosa computacionalmente.

Bibliografia

[1] E. Romero and J. M. Sopena, "Performing feature selection with multilayer perceptrons," *IEEE Transations on neural networks*, vol. 19, no. 3, pp. 431–441, March 2008.

[2] M. M. Kabir, M. M. Islam, and K. Murase, "A new wrapper feature selection approach using neural network," *Neurocomputing*, vol. 73, no. 16-18, pp. 3273–3283, 2010.

[3] X. Zeng and D. S. Yeung, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure," *Neurocomputing*, vol. 69, no. 7-9, pp. 825–837, 2006.

[4] T. Similä and J. Tikka, "Combined input variable selection and model complexity control for nonlinear regression," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 231–236, 2009.

[5] K. Li and J.-X. Peng, "Neural input selection-a fast model-based approach," *Neurocomputing*, vol. 70, no. 4-6, pp. 762–769, 2007.

[6] O. F. M. El-Mahdy. M. E. H. Ahmed, and S. Metwalli, "Computer aided optimization of natural gas pipe networks using genetic algorithm," *Applied Soft Computing*, pp. 1141–1150, 2010.

[7] O. Ludwig, U. Nunes, R. Araújo, L. Schnitman, and H. A. Lepikson, "Applications of information theory, genetic algorithms, and neural models to predict oil flow," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 7, pp. 2870–2885, 2009.

[8] M. R. Delgado, E. Y. Nagai, and L. V. R. de Arruda, "A neuro-coevolutionary genetic fuzzy system to design soft sensors," *Soft Computing*, vol. 13, no. 5, pp. 481–495, 2008.

[9] F. Souza, P. Santos, and R. Araújo, "Variable and delay selection using neural networks and mutual information for data-driven soft sensors," in *Proc. 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, September 2010, pp. 1–8.

[10] G. Simon and M. Verleysen, "High-dimensional delay selection for regression models with mutual information and distance-to-diagonal criteria," *Neurocomputing*, vol. 70, no. 7-9, pp. 1265–1275, 2007.

[11] M. Shakil, M. Elshafei, M. Habib, and F. Maleki, "Soft sensor for nox and o2 using dynamic neural networks," *Computers & Electrical Engineering*, vol. 35, no. 2, pp. 578–586, 2009.

[12] D. Du, C. Wu, X. Luo, and X. Zuo, "Delay time identification and dynamic characteristics study on ann soft sensor," in *Proceedings Sixth International Conference on Intelligent Systems Design and Applications (ISDA 06)*, October 2006, pp. 42–45.

[13] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia, *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer, 2007.

[14] P. Kadlec, B. Gabrys, and S. Strandt, "Data-driven soft sensors in the process industry," *Computers & Chemical Engineering*, vol. 33, no. 4, pp. 795–814, 2009.

[15] S. Serneels and T. Verdonck, "Principal component analysis for data containing outliers and missing elements," *Comput. Stat. Data Anal.*, vol. 52, pp. 1712–1727, January 2008.

[16] I. Stanimirova, M. Daszykowski, and B. Walczak, "Dealing with missing values and outliers in principal component analysis," *Talanta*, vol. 72, no. 1, pp. 172–178, 2007.

[17] B. Walczak and D. L. Massart, "Robust principal components regression as a detection tool for outliers," *Chemometrics and Intelligent Laboratory Systems*, vol. 27, no. 1, pp. 41–54, 1995.

[18] D. Ruan, *Intelligent Data Mining: Techniques and Applications*. Berlin, Heidelberg: Springer-Verlag, 2010.

[19] J. J. Macias, P. Angelov, and X. Zhou, "A method for predicting quality of the crude oil distillation," in *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems*, 2006, pp. 214–220.

[20] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biology*, vol. 5, no. 4, pp. 115–133, 1943.

[21] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall, 1999.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 9, pp. 533–536, 1986.

[23] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[24] B. Widrow and M. E. Hoff, "Adaptive switching circuits," *IRE WESCON Convention Record*, no. 4, pp. 96–104, 1960.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI, USA: University of Michigan Press, 1975.

[26] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Springer, 2007.

[27] L. Torgo, "Friedman artificial domain dataset," January 2006. [Online]. Available: http://www.liaad.up.pt/~ltorgo/Regression/fried.html

[28] J. Friedman, "Multivariate adaptative regression splines," *Annals of Statistics*, vol. 19, no. 1, pp. 1–141, 1991.

[29] G. Reinsel, "Box and jenkins dataset," March 1999. [Online]. Available: http://www.stat.wisc.edu/~reinsel/bjr-data/gas-furnace

[30] UCI Machine Learning Repository, "Automobile mpg dataset," July 1999. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Auto+MPG

[31] UCI Machine Learning Repository , "Concrete compressive strength dataset," August 2007. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

[32] UCI Machine Learning Repository, "Boston housing dataset," July 1993. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Housing

[33] L. Torgo, "Triazines dataset," January 2006. [Online]. Available: http://www.liaad.up.pt/~ltorgo/Regression/triazines.html

[34] UCI Machine Learning Repository, "Parkinson telemonitoring dataset," October 2009. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring

[35] L. Torgo, "Ailerons dataset," January 2006. [Online]. Available: http://www.liaad.up.pt/~ltorgo/Regression/ailerons.html

[36] M. Champagne, M. Amazouz, and R. Platon, "The application of soft sensors in the pulp and paper and cement manufacturing sectors for process and energy performance improvement: Opportunity analysis and technology assessment," Canmet Energy Technology Centre, Tech. Rep., 2005.

Apêndice A

Artigo Publicado

Co-evolutionary Genetic Multilayer Perceptron for Feature Selection and Model Design

Francisco Souza, Tiago Matias, and Rui Araújo
Institute of Systems and Robotics (ISR-UC), and
Department of Electrical and Computer Engineering (DEEC-UC),
University of Coimbra, Pólo II, PT-3030-290 Coimbra
fasouza@isr.uc.pt, tmatias@isr.uc.pt, rui@isr.uc.pt

Abstract

This paper proposes a method for Soft Sensors design using a Multilayer Perceptron model based on co-evolutionary genetic algorithms, called CEV-MLP. This method jointly and automatically selects the best input variables and the best configuration of the network for the prediction setting. The CEV-MLP is constituted by three levels, the first level selects the best input variables and respective delays set, the second level is composed by the parameters of hidden layers to be optimized (number of neurons in the hidden layers and transfer function), and the third level is the combination of first and second level. The method was successfully applied, and compared with two state-of-the-art methods, in three real datasets. In all the experiments, the proposed method shows the best approximation accuracy, while all the design of the prediction setting is performed automatically.

1 Introduction

Data-driven soft sensors (DDSS) are inferential models that use on-line available sensor measures, possibly complemented with measures obtained with laboratory analysis, for on-line estimation of 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) [4, 7, 16].

The selection of input variables and respective delays is essential to obtain an accurate and reliable reproduction of the target variable. If the network is trained with all available input variables in the dataset, it is being assumed that all features are good variables for prediction. However, this assumption is not valid when the data has irrelevant and/or redundant features. Moreover, the problem of selecting the most adequate delays for input variables remains. An MLP trained with irrelevant variables could be more flexible that without them and have better approximations in training set, but also will have a poor generalization performance [13]. Feature selection also decreases the training time of models and prevents network over-fitting [18]. In this work both steps of (1) selecting input variables and delays, and (2) selecting MLP network structure are solved using a multilayer perceptron model (MLP) by means of a co-evolutionary scheme. This approach of variable selection based on, and using, the learning model is called a wrapper approach [8].

The most common methods used to perform variable selection for MLP use a sequential backward search (SBS) or a sequential forward search (SFS) procedure and use the sum of squares error (SSE) as cost function. In [13] the variable selection algorithm using the traditional SBS was discussed, and the authors propose the retraining of the network when a feature is removed for evaluation. However, this method in some datasets becomes computationally expensive, because it is necessary retrain the network $\frac{n(n-1)}{2}$ times, where n is the input dimensionality.

In [17] a pruning algorithm for MLP networks is presented. In a pruning algorithm, the network is oversized and then the least significant hidden neurons and weights are pruned to find the smallest feasible size. In [17] it is proposed a sensitivity measure to verify the output sensitivity due to input perturbation, and a relevance measure to verify the relevance of neurons. Variable selection can be performed using both measures. In [5] it is proposed an evolutionary scheme for feature selection, called SAGA. This algorithm first uses simulated annealing to guide the global search in a solution space, and then uses a genetic algorithm to perform optimization. The main disadvantage of this method is that it does not take into account the optimization of the model, e.g. the number of the neurons in the hidden layer, when used in combination with a MLP model. In [14], a new cost function for simultaneous input variable and hidden node selection for an MLP model is proposed. This method penalizes the weights during fitting so that useless input variables can be excluded. The performance of the method depends on tuning the amount of penalization and the shape of the penalization function, i.e. the method is not fully automatic and depends on the dataset. In [10], a new fast model-based neural input selection method is pre-

sented. It is assumed that nonlinear models like polynomial NARX models, Volterra series and neural networks (NN) can achieve equivalent performance given that certain conditions are met. It is used a Volterra series model that is "linear-in-the-parameters", making it possible the use of existing model selection methods, e.g. orthogonal least-squares method. This method achieves significant reduction in the computational complexity but does not take care of the automatic design of the model used.

Genetic Algorithms (GA) have proved to be a useful tool to solve optimization problems. In [3], it is studied how to determine the optimum pipe size for networks used in natural gas applications. In [11] a Genetic Algorithm is used to maximize mutual information between the input and output variables, to prediction of oil flow. A multi-objective Genetic Algorithm (MOGA) based on the wrapper approach for feature selection is proposed in [9]. The MAGO is used to optimize a multi-objective problem, simultaneously minimizing the error rate and the model complexity. Compared with proposed algorithm, the main disadvantage is that this method does not optimize the hidden layer design, i.e. the number of hidden nodes and the activation functions. An approach using methods for nonlinear variable selection in conjunction with T-S fuzzy models was proposed by [2], for soft sensors applications. It uses T-S fuzzy models from available input/output data by means of a co-evolutionary genetic algorithm and a neuro-based technique. The soft sensor design is carried out in two steps. First, the input variables of the fuzzy model are pre-selected from the secondary variables of a dynamical process by means of correlation coefficients, Kohonen maps and Lipschitz quotients. Such selection procedure considers nonlinear relations among the input and output variables. Second, a hierarchical genetic algorithms is used to identify the fuzzy model itself. The input variable selection proposed by [2] has some shortcomings. First, the selection of the number of neurons in Kohonen maps is not automatically performed. Second, delays are not jointly selected with input variables, which can bring lower-accuracy results because a variable with the correct delay can contain more information about the output than a variable with the incorrect delay [16].

In this paper, a new co-evolutionary multilayer perceptron (CEV-MLP) method is proposed. Differently from previous approaches the method takes into consideration, and jointly selects, both the input variables and the configuration of an MLP prediction neural network. The CEV-MLP is constituted by three levels. The first level selects the best input variables and respective delays set. The second level is composed by, and selects, the parameters of hidden layers to be optimized (number of neurons in the hidden layers and transfer function). The third level is the combination of the first and second levels. The method was successfully applied, and compared with two state-of-the-art methods, in three real datasets, two publicly available datasets (Box Jenkins gas furnace, and Gas Mileage), and a dataset of a problem of flour concentration estimation in a real-world urban wastewater treatment plant (WWTP). The proposed CEV-MLP method exhibits the best prediction performance. This is achieved while the method automatically designs the prediction setting by jointly selecting input variables, the respective delays, and the MLP prediction model.

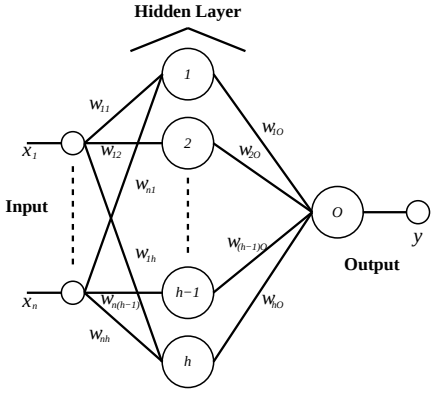


Figure 1: Topology of MLP-TL neural network; O is the output node, $\mathbf{W}_I = \mathbf{W} = [w_{ij}]$ is the $n \times h$ matrix of the weights connecting the inputs to the h hidden layer nodes, and $\mathbf{w}_O = [w_{1O}, \ldots, w_{hO}]^T$ is the output weight vector. The hidden layer biases $\mathbf{b}_I$ and the output bias b_O are omitted to simplify the diagram.

The paper is organized as follows. The MLP architecture is overviewed in Section 2. The new MLP variable selection algorithm proposed in this paper is presented in Section 3. Section 4 presents experimental results. Finally, Section 5 gives concluding remarks.

2 Multilayer Perceptron Architecture

An MLP neural network (NN) with two layers, that is used as the basis for the CEV-MLP, is represented in Figure 1. In [12, 6] it was shown that an MLP with one only hidden layer and a sufficient number of neurons can uniformly approximate any continuous function to any accuracy.

The MLP NN can be mathematically represented by:

$$y \;=\; g\left(f\left(\mathbf{x}^T\mathbf{W}_I + \mathbf{b}_I\right)\mathbf{w}_O + b_O\right), \qquad (1)$$

where $\mathbf{x} = [x_1, \ldots, x_n]^T$ is the input vector, y is the predicted output, $\mathbf{W}_I = \mathbf{W} = [w_{ij}]$ is the $n \times h$ matrix of the weights connecting the inputs to the h hidden layer nodes, $\mathbf{b}_I = \mathbf{b} = [b_1, \ldots, b_h]$ is the vector of biases of the hidden layer nodes. The output weights that connect the hidden neurons with the output neuron and the output bias are represented by $\mathbf{w}_O = [w_{1O}, \ldots, w_{hO}]^T$ and b_O, respectively. $f(\cdot)$ and $g(\cdot)$ represent the activation functions of the nodes of the hidden layer, and output layer, respectively. The network is trained by minimizing the mean square error (MSE) of all network output samples:

$$E_{mse}(y, y_d) \;=\; \frac{1}{L}\sum_{k=1}^{L}\left[y(k) - y_d(k)\right]^2,$$

where, $y(k)$ and $y_d(k)$ are the predicted and desired output of k-th input data sample, and L is the number of exemplars. In this work $f(\cdot)$ can be a tangent sigmoid or a linear function and $g(\cdot)$ is a linear function. To perform the MLP NN design there are several parameters to be considered: the types of the activation function, the number of neurons in the hidden layer and the best subset of input variables and respective delays.

3 CEV-MLP

The objective of the CEV-MLP is to optimize the final prediction model by jointly selecting the appropriate MLP architecture and the best subset of input variables and respective time delays. The optimization is performed by means of genetic algorithms. The CEV-MLP is constituted by three hierarchical levels (Fig. 2). The first level is constituted by the possible sets of input variables and respective delays, the second level is constituted by the possible set of hidden layer configurations and the third level represents the combination of the first and second level, i.e. the final model.

3.1 Hierarchical Architecture

Fig. 2 shows the detailed scheme of the CEV-MLP, it is constituted by 3 levels. The detailed description of each level is given below:

First Level is constituted by the possible sets of input variables and respective time delays that will be used to design the DDSS. The chromosome is represented by a binary encoding, where each allele (element of the chromosome that is located at a specific position) corresponds to each input variable and respective delay (see Fig. 2). One zero in one allele indicates that the input associated to this allele is not considered.

Second Level is constituted by the possible sets of hidden layer configurations. Each allele of the chromosome can get values from zero to two. The zero value indicates that the neuron of the hidden layer associated to this allele will be pruned, i.e. the number of the neurons is given by the number of alleles different from zero. If the allele gets the value one or two, this means that the activation function of the neuron of the hidden layer associated to this allele will be tangent sigmoid or linear, respectively.

Third Level is constituted by the possible sets of MLP configurations. Each chromosome is constituted by two alleles that can get a non-negative integer, being that the first allele represents the individual of level 1 and the second allele represents the set of level 2.

A predictor at the third level is denoted by $S_3^{(m,l)} = C(S_2^m, S_1^l)$, a combination of Levels 1 and 2, where S_2^m, S_1^l are the m-th and l-th, chromosomes of Levels 2 and 1, respectively, and C is an operator that generates a MLP model using the S_2^m and S_1^l chromosomes. The predicted output generated by $S_3^{(m,l)}$ is given by $y^{(m,l)}$.

input: Maximum number of chromosomes for each level, $l_{max}, m_{max}, k_{max}$
output: A optimal MLP model with best variables and configuration

$i \leftarrow 1$;
$int \leftarrow 1$;
while $int < N_{max}$ **do**
  forall $k = 1, \ldots, k_{max}$ **do**
    Evaluate $J_3^k(i)$ using Equ. (2a);
  end
  forall $m = 1, \ldots, m_{max}$ **do**
    Evaluate $J_2^m(i)$ using Equ. (2b);
  end
  forall $l = 1, \ldots, l_{max}$ **do**
    Evaluate $J_1^l(i)$ using Equ. (2c);
  end
  Select the two best chromosomes on Level 1*, Level* 2 *and Level* 3*, according with* J_1^l*,* J_2^m *and* J_3^k *to be parents. For a fast convergence all the other inputs are removed and reproduction is performed to obtain new ones. Perform mutation in all new children for all levels;*
  if $J_3^k(i) == J_3^k(i-1)$ **then**
    $int \leftarrow int + 1$;
  else
    $int \leftarrow 1$;
  end
  $i \leftarrow i + 1$;
end

Algorithm 1: Steps of CEV-MLP algorithm

The cost function for the k-th chromosome of the third level $J_3^k = J_3^{(m,l)}$, which is given by the mean square error between the predicted and real outputs in the training dataset, and cost functions for Levels 1 and 2 are, respectively, given by:

$$J_3^k = J_3^{(m,l)} = E_{mse}\left(y^{(m,l)}, y_d\right), \forall k, \quad (2a)$$

$$J_2^m = \min\left(J_3^{(m,1)}, J_3^{(m,2)}, \ldots, J_3^{(m,l_{max})}\right), \forall k, \quad (2b)$$

$$J_1^l = \min\left(J_3^{(1,l)}, J_3^{(2,l)}, \ldots, J_3^{(m_{max},l)}\right), \forall k. \quad (2c)$$

k_{max}, l_{max}, m_{max}, are the maximum number of chromosomes at Levels 3, 2, and 1, respectively. It is important note that $k_{max} \leq m_{max} \cdot l_{max}$, because the number of chromosomes of Level 3, is always lower than $m_{max} \cdot l_{max}$.

An example of the encoding and the hierarchical relations is given in Fig. 2. In this example, the first allele of the k-th set of Level 3 indicates that the set of input variables and delays for the network will be the 6th set represented at Level 1, while the second allele indicates that the configuration of the hidden layer of the MLP will be the

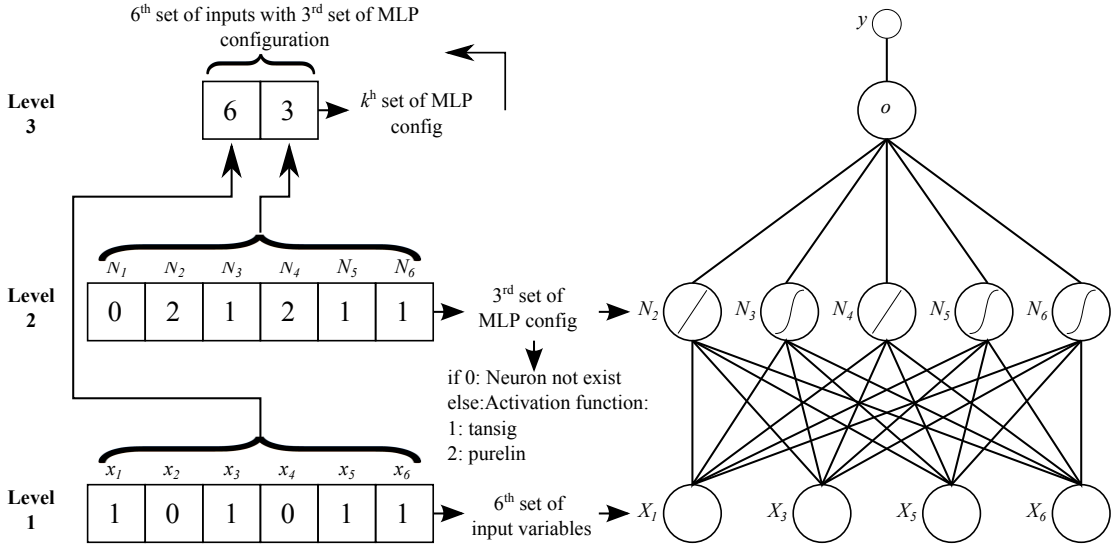


Figure 2: Graphical Representation Scheme of CEV-MLP algorithm.

one described in the 3rd chromosome of Level 2. The first allele of the chromosome of level 2 is zero. So the number of neurons in hidden layer will be five, the size of the chromosome (6) minus the number of zero valued alleles (1). The neurons associated to the 2nd and 4th allele will have a linear activation function and the remaining will have tangent sigmoid transfer function. Level 1, specifies that the inputs of the network will be the variables x_1, x_3, x_5 and x_6.

The CEV-MLP is specified in Algorithm 1. The first step of the algorithm is the random initialization of the populations of all levels. After this, while N_{max} is less than int, i.e. while the method does not reach the maximum number the of iterations with the same error: for all chromosomes of Level 3, the performance of the network is computed; for all levels the best two chromosomes are selected to be the parents and the other are removed; new ones are obtained by reproduction and mutation.

3.2 Genetic Algorithm Operators

The methods used for initialization, selection, reproduction and mutation in the genetic algorithm of the CEV-MLP approach are described in this section.

Initialization: The initial population is chosen randomly with uniform distribution. Is know that random initialization can affect convergence time, but good results have been obtained in CEV-MLP.

Selection: The selection method used was elitist selection [15]. This method selects the best chromosomes to be the parents, the remaining are removed and new chromosomes are generated, in this work the best chromosomes considered are the first two. A random number between 0 and 1, R_s, is generated, if $R_s < 0.5$ the first individual is used as a father and the second as a mother, and if the $R_s \geq 0.5$ the second individual is the father and the first is the mother.

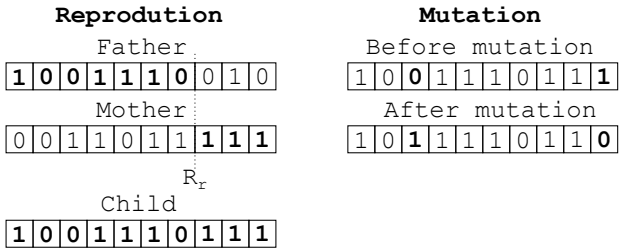


Figure 3: Genetic operators: Single-point Reproduction and Mutation of two alleles.

Reproduction: For reproduction the single point crossover technique was used. The process consists of taking two parents and produce a child [15]. As described above, all chromosomes except the parents are removed and new ones are obtained by the crossover of the two selected parents. For each child, the crossover process generates a random point of crossover, R_r, and the child will receive the alleles from 1 to R_r from the father and the rest of the alleles are received from the mother. This process is illustrated in Fig. 3.

Mutation: Mutation of two alleles was the third operation used. This is used to maintain the diversity of the population and to prevent the algorithm from being trapped in a local minimum. The mutation is an operator that alters the value of one or more randomly selected alleles in a chromosome. To perform crossover and mutation in the second and the third level, the chromosomes are converted in binary encoding. Mutation is also illustrated in Fig. 3.

4 Experiments and Results

This section presents experimental results in three distinct datasets, verifying the performance and demonstrat-

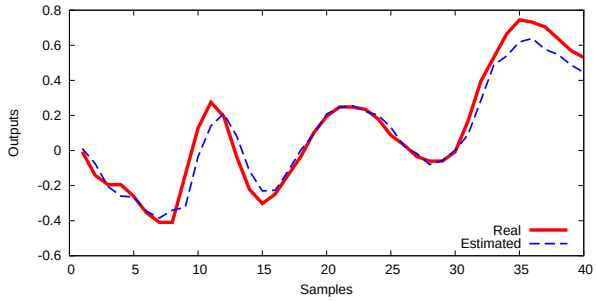


Figure 4: Predicted and target outputs using CEV-MLP for the Box and Jenkins furnace gas test dataset.

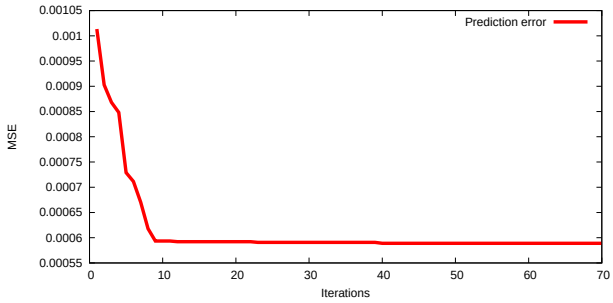


Figure 5: MSE of the network validation of the proposed algorithm for the Box and Jenkins furnace gas dataset.

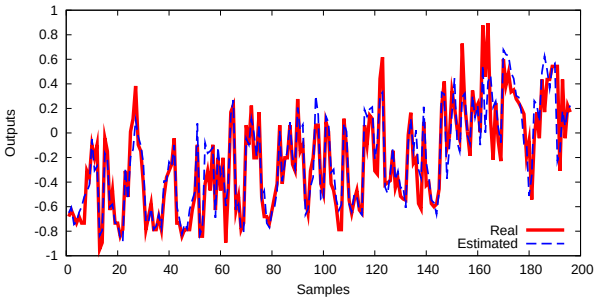


Figure 6: Predicted and target outputs using proposed algorithm for the Automobile MPG dataset.

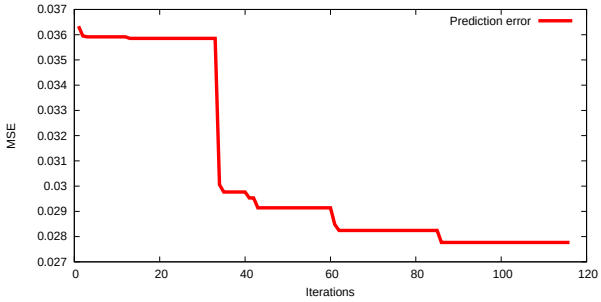


Figure 7: MSE of the network validation of the proposed algorithm for the Automobile MPG dataset.

ing the effectiveness of the proposed methods. The approximation performance of the soft sensors is evaluated using the mean square error (MSE) and the correlation coefficient between predicted and desired output, in the validation and test data. For all experiments the reproduction and mutation probabilities are 80% and 10%, respectively, and the number of chromosomes for each level are: $k_{max} = 20$, $m_{max} = 200$, $l_{max} = 200$ and $N_{max} = 30$. The datasets were divided in training and test data and, in turn the training data was randomly divided in 75% for training and 25% for validation. The proposed method is compared with (i) the traditional SBS method using MLP, and (ii) the method proposed in [10].

4.1 Box and Jenkins Dataset

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 $[y(t), u(t)]$ [1]. The input $u(t)$ is the gas flow rate into the furnace and the output $y(t)$ is the carbon dioxide (CO_2) concentration in the outlet gas. The sampling interval is 9 [s]. To predict $y(t)$ the following set with all variables and possible considered delays is examined $X^{(t)} = \{y(t-1), y(t-2), y(t-3), y(t-4), u(t-1), u(t-2), u(t-3), u(t-4), u(t-5), u(t-6)\}$. The first 250 samples were used for training and the remaining for test/evaluation. The maximum number of neurons in the hidden layer was limited to three.

The results of the application of the three methods is presented in Table 1. Fig. 4 shows the predicted and the target outputs for the test dataset, and Fig. 5 shows the prediction error during the CEV-MLP operation, as can be seen the error stabilizes at iteration 40.

The proposed CEV-MLP chooses more pairs of input variables and delays than the traditional SBS and method of [10], but the mean square error and correlation coefficient has similar values. CEV-MLP attains these results while exhibiting the advantages discussed in Sec. 1, namely automatically selecting both the input variables and respective delays, as well as the MLP structure. The selected configuration for the MLP model is composed by two neurons in the hidden layer and both activation functions are tangent sigmoid.

4.2 Automobile MPG Dataset

The automobile gas mileage dataset corresponds to a problem of predicting the number of miles per gallon (MPG). It is a six input, single output regression problem. The gasoline consumption needs to be predicted based on some input variables. These variables are the number of cylinders, displacement, horsepower, weight, acceleration and model year. The original data is available in the UCI (Univ. of California at Irvine) Machine Learning Repository[2]. The input set considered is $X^{(t)} = \{u_1(t), u_2(t), u_3(t), u_4(t), u_5(t), u_6(t)\}$. Where u_1 is the number of cylinders, u_2 the displacement, u_3 the horsepower, u_4 the weight, u_5 the acceleration, and u_6 is the year. The train and test dataset are composed by 196 samples each. The maximum number of neurons to be se-

[1] Provided by 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 1: Performance Results for the Box and Jenkins dataset

Method	Selected Inputs	MSE Test	Correlation Test
SBS	$y(t-1), y(t-2), u(t-3)$	$5,94e-3$	$\mathbf{0,981}$
Li & Peng [10]	$y(t-1), y(t-2), ut-3)$	$5,75e-3$	$\mathbf{0,981}$
CEV-MLP	$y(t-1), y(t-2), u(t-3), u(t-5), u(t-6)$	$\mathbf{5,69e-3}$	$0,980$

Table 2: Performance Results for the Gas Mileage dataset

Method	Selected Inputs	MSE Test	Correlation Test
SBS	$u_2(t), u_6(t)$	$3,30e-2$	$0,899$
Li & Peng [10]	$u_1(t), u_2(t), u_3(t), u_6(t)$	$2,70e-2$	$0,918$
CEV-MLP	$u_2(t), u_4(t), u_5(t), u_6(t)$	$\mathbf{2,43e-2}$	$\mathbf{0,927}$

lected by the CEV-MLP is three.

The results are presented in Table 2. Fig. 6 shows the predicted and the target outputs and Fig. 7 shows the prediction error of the network while the proposed method performs the optimization of the prediction setting (variables, delays, and network structure).

The CEV-MLP method chooses the most adequate subset of input variables and respective delays when compared with the SBS method and the method proposed by [10]: CEV-MLP has better results in terms of error and correlation coefficient. The MLP model selected by the CEV-MLP is composed by tree neurons in the hidden layer with tangent sigmoidal transfer function.

4.3 WWTP Dataset

In the third experiment the objective is to estimate the flour concentration in the effluent of a real-world urban wastewater treatment plant (WWTP). The dataset of plant variables that is available for learning consists of 11 input variables, $U^{(t)} = \{u_1(t), \ldots, u_{11}(t)\}$, 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, see Table 3 for further details. The possible input variables with respective delays used as input for the optimization problem are: $X^{(t)} = \{u_1(t), u_1(t-1), u_1(t-2), \ldots, u_{11}(t), u_{11}(t-1), u_{11}(t-2)\}$. The input variables are measured online by plant sensors, and the output variable in the dataset is measured by laboratory analysis. The sampling interval is 2 [hours]. The proposed algorithm jointly selects the best variables and delays, as well as the MLP structure, for the flour prediction setting. The train and test dataset are composed by 196 samples.

The results of the application of the three methods (SBS-MLP, [10], CEV-MLP) are presented in Table 4. Fig. 8 shows the predicted and the target outputs, and Fig. 9 shows the prediction error of the test set while the CEV-MLP performs the network optimization and selection of best input variables and delays. The proposed algorithm has chosen less input (variable, delay) pairs than SBS and the method of [10], and the MSE and correlation values

Table 3: Variables of the wastewater treatment plant dataset

Variables	Description
u_1	Amount of chlorine in the influent;
u_2	Amount of chlorine in the effluent;
u_3	Turbidity in the raw water;
u_4	Turbidity in the influent;
u_5	Turbidity in the effluent;
u_6	Ph in the raw water;
u_7	Ph in the influent;
u_8	Ph in the effluent;
u_9	Color in the raw water;
u_{10}	Color in the influent;
u_{11}	Color in the effluent;
y	Flour in the effluent.

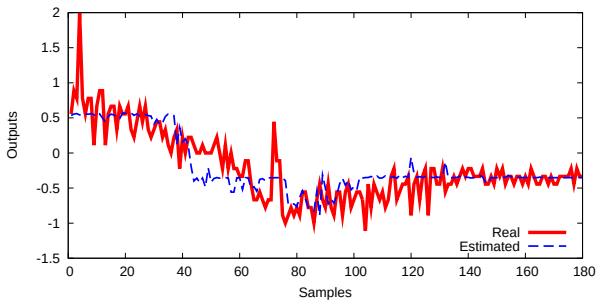


Figure 8: Predicted and target outputs using proposed algorithm for the WWTP dataset.

between the target and predicted outputs show that the proposed CEV-MLP method has better prediction performance results. The selected MLP is composed by eight neurons in the hidden layer, where three of them are linear and the remaining are tangent sigmoidal.

5 Conclusion

The paper proposed a new method for jointly selecting input variables and corresponding delays, as well as constructing the structure of an MLP prediction model. An evolutionary scheme using genetic algorithms selects the best set of input (variable, delay) pairs and the best MLP model, making it suitable for Soft Sensor applica-

Table 4: Performance Results for the WWTP dataset.

Method	Selected Inputs	MSE Test	Correlation Test
SBS	$u_9(t-4)$, $u_{10}(t-4)$, $u_1(t-2)$, $u_3(t-2)$, $u_4(t-2)$, $u_5(t-2)$, $u_6(t-2)$, $u_1(t)$, $u_3(t)$, $u_4(t)$, $u_6(t)$, $u_7(t)$, $u_9(t)$	$8,05e-2$	$0,815$
Li & Peng [10]	$u_1(t-4)$, $u_3(t-4)$, $u_5(t-4)$, $u_7(t-4)$, $u_9(t-4)$, $u_{10}(t-4)$, $u_{11}(t-4)$, $u_3(t-2)$, $u_7(t-2)$, $u_8(t-2)$, $u_1(t)$, $u_5(t)$, $u_7(t)$, $u_8(t)$, $u_9(t)$, $u_{10}(t)$, $u_{11}(t)$	$8,56e-2$	$0,804$
CEV-MLP	$u_4(t-4)$, $u_1(t-2)$, $u_4(t-2)$, $u_8(t-2)$, $u_4(t)$	$\mathbf{6,87e-2}$	$\mathbf{0,844}$

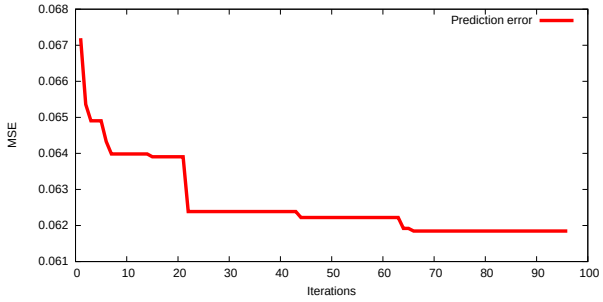


Figure 9: MSE of the network validation of the proposed algorithm for the WWTP dataset.

tions. The proposed method does not require any prior knowledge concerning of the model and about the best input variables; Only the empirical input-output data is required.

To validate and demonstrate the performance and effectiveness of the proposed methodology, the algorithm was applied on three prediction problems with real-world datasets, and compared with two state-of-art methods. The experimental results have shown the effectiveness of the proposed method. The proposed CEV-MLP method exhibits the best prediction performance, while automatically designing the prediction setting (jointly selecting inputs, delays, and prediction model).

Future work will implement different optimization methods, and incorporate more criteria to be optimized.

Acknowledgment

This work was supported by Mais Centro Operacional Program, financed by European Regional Development Fund (ERDF), and Agência de Inovação (AdI) under Project SInCACI/3120/2009.

Francisco Souza has been supported by Fundação para a Ciência e a Tecnologia (FCT) under grant SFRH/BD/63454/2009.

References

[1] G. E. P. Box and G. M. Jenkins. Time series analysis. *Cambridge Univ. Press*, 2003.

[2] M. R. Delgado, E. Y. Nagai, and L. V. R. de Arruda. A neuro-coevolutionary genetic fuzzy system to design soft sensors. *Soft Computing*, 13(5):481–495, 2008.

[3] O. F. M. El-Mahdy. M. E. H. Ahmed, and S. Metwalli. Computer aided optimization of natural gas pipe networks using genetic algorithm. *Applied Soft Computing*, pages 1141–1150, 2010.

[4] L. Fortuna, S. Graziani, A. Rizzo, and M. G. Xibilia. *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer, 2007.

[5] I. A. Gheyas and L. S. Smith. Feature subset selection in large dimensionality domains. *Pattern Recognition*, 43(1):5–13, 2010.

[6] K. M. Hornik, M. Stinchcombea, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[7] P. Kadlec, B. Gabrys, and S. Strandt. Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 2009. 33(4):795 - 814.

[8] R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence Archive*, 97(1-2):273–324, December 1997.

[9] H. C. Lac and D. A. Stacey. Feature subset selection via multi-objective genetic algorithm. In *International Joint Conference on In Neural Networks*, volume 3, pages 1349–1354, 2005.

[10] K. Li and J.-X. Peng. Neural input selection-a fast model-based approach. *Neurocomputing*, 70(4-6):762–769, 2007.

[11] O. Ludwig, U. Nunes, R. Araújo, L. Schnitman, and H. A. Lepikson. Applications of information theory, genetic algorithms, and neural models to predict oil flow. *Communications in Nonlinear Science and Numerical Simulation*, 17(7):2870–2885, 2009.

[12] R. H. Nielsen. Theory of the back propagation neural network. In *International Joint Conference on In Neural Networks*, pages 593–605, June 1989.

[13] E. Romero and J. M. Sopena. Performing feature selection with multilayer perceptrons. *IEEE Transations on neural networks*, 19(3), March 2008.

[14] T. Similä and J. Tikka. Combined input variable selection and model complexity control for nonlinear regression. *Pattern Recognition Letters*, 30(3):231–236, 2009.

[15] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer, Berlin, Germany, 2008.

[16] F. Souza, P. Santos, and R. Araújo. Variable and delay selection using neural networks and mutual information for data-driven soft sensors. In *Proc. 2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, pages 1–8, September 2010.

[17] X. Zeng and D. S. Yeung. Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure. *Neurocomputing*, 69(7-9):825–837, 2006.

[18] G. P. Zhang. Avoiding pitfalls in neural network research. *IEEE Transactions on systems man and cybernetic Part C applications and reviews*, 37(1):3–16, 2007.