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

Paulo Rui Pereira Susana

Sensores Virtuais Usando Aprendizagem Online para Processos Industriais

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores.

Coimbra

Março de 2015

Universidade de Coimbra

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

Sensores Virtuais Usando Aprendizagem Online para Processos Industriais

por

Paulo Rui Pereira Susana

Dissertação de Mestrado Integrado em Engenharia Electrotécnica e de Computadores, ramo de especialização em Automação.

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

Júri

Presidente:	Prof. Doutor Lino José Forte Marques
Vogais:	Prof. Doutor Gabriel Falcão Paiva Fernandes
	Prof. Doutor Rui Alexandre de Matos Araújo

Março de 2015

Para a minha família, namorada e amigos.

"A person who never made a mistake never tried anything new."
Albert Einstein

Agradecimentos

Agradeço aos meus pais Rui Susana e Glória Susana por terem incentivado o meu ingresso na faculdade, pelo esforço que fizeram para que tivesse todo o conforto nestes anos que passaram.

Agradeço ao meu irmão Nuno Susana por ter vindo estudar para perto de mim e por nunca me deixar adormecer em dias de exame. Quero também agradecer a todos os outros familiares por todo o apoio que me deram.

Quero dar um especial obrigado à minha namorada Ângela Cruz pelo amor, carinho, paciência inesgotável e por estar sempre do meu lado.

Aos meus amigos, Álvaro Oliveira e Pedro Moreira, agradeço as horas de diversão, estudo e distração, que tornaram estes anos tão inesquecíveis. A estes e aos restantes amigos agradeço a sua amizade e apoio ao logo da vida.

O meu obrigado a todos os colegas de curso pela competitividade saudável que mantiveram ao longo destes anos de estudo.

Agradeço ao meu orientador Professor Doutor Rui Araújo e ao meu co-orientador Professor Doutor Francisco Souza pelos conselhos, opiniões, dedicação, rigor científico e paciência que permitiram o desenvolvimento desta dissertação.

Agradeço ainda a todas as outras pessoas que de forma direta ou indireta me ajudaram na realização desta dissertação.

Resumo

Os Sensores Virtuais são ferramentas desenvolvidas em *software* que têm visto a sua aplicação em processos industriais ganhar popularidade. Por vezes nos processos industriais existem variáveis críticas para o funcionamento do processo que não podem ser medidas de forma automática ou precisa por sensores tradicionais, sendo necessário em muitos casos recorrer a análises laboratoriais, o que origina a falta de conhecimento sobre o que acontece com a variável em tempo real. Os Sensores Virtuais devem a sua popularidade ao facto de conseguirem estimar a variável crítica ao processo, através de um modelo aprendido ou treinado com base em dados históricos, normalmente obtidos pelos sensores físicos do processo.

Um dos problemas chave no desenvolvimento de Sensores Virtuais é que a maioria dos processos industriais são variantes no tempo. Ou seja, o processo físico para o qual o modelo foi treinado sofre alterações, sejam elas bruscas ou suaves, ao longo do tempo. Esta situação normalmente traduz-se numa degradação do desempenho do Sensor Virtual. Para ultrapassar este problema devem utilizar-se métodos de estimação adaptativos.

Nesta dissertação são propostos três métodos para desenvolvimento de Sensores Virtuais. O primeiro é um método adaptativo, baseado no conceito de modelos de regressão linear, mais concretamente é uma hibridação do método de regressão linear Recursive Least Squares - Adaptive (RLS-A) [Haykin, 1996] com o método Incremental Forward Stagewise Regression (IFSR) [Hastie *et al.*, 2009] e é denominado de Incremental Forward Stagewise Regression Online (IFSR-ON). Os outros dois métodos propostos são baseados em métodos *ensemble*, e no conceito de *boosting*. Um deles é denominado de Least Squares Boosting Single Variable (LSBST-SV), é não adaptativo e é baseado no método Least Squares Boosting (LSBST) [Friedman, 2001]. O outro método é uma versão adaptativa do método LSBST-SV, é denominado de Least Squares Boosting Single Variable Online (LSBST-SV-ON) e foi baseado numa *framework* para tornar o funcionamento dos métodos *boosting* de modo a terem uma operação *online* [Babenko *et al.*, 2009].

Para validação e discussão dos métodos propostos, eles foram aplicados a treze *data sets* de referência disponíveis em repositórios públicos, e a dois problemas reais variantes no tempo. O desempenho dos métodos propostos foi comparada com a de alguns dos métodos para desenvolvimento de Sensores Virtuais do estado da arte, nomeadamente: Recursive Least Squares (RLS), RLS-A, IFSR, Partial Least Squares (PLS), Recursive Partial Least Squares (RPLS), Least Absolute Shrinkage and Selection Operator (LASSO), Recursive

Least Absolute Shrinkage and Selection Operator (RLASSO), Recursive Least Absolute Shrinkage and Selection Operator with Adaptive Forgetting (RLASSO-AF), LSBST e Least Squares Boosting Online (LSBST-ON).

Nos treze *data sets* de referência foram analisados os desempenhos dos métodos em funcionamento *offline*, tendo os melhores desempenhos sido maioritariamente obtidos pelos métodos PLS e LASSO, embora os métodos LSBST-ON e LSBST-SV-ON tenham obtido desempenhos não muito diferentes dos restantes métodos *offline*. Para os dois problemas de estimação reais variantes no tempo, o melhor desempenho pertenceu ao método RPLS, embora o método LSBST-SV-ON tenha apresentado resultados bastante comparáveis, revelando-se uma ferramenta valiosa para o desenvolvimento de SSs adaptativos.

Palavras-chave: Sensores Virtuais, Inteligência Computacional, Aprendizagem Online, Pré-processamento de Dados, Modelação Adaptativa.

Abstract

Virtual Sensors are tools developed in software that have seen their application gain popularity in industrial processes. Sometimes in industrial processes there are critical variables for operation of the process that can not be measured accurately or automatically by traditional sensors, being necessary in many cases to recourse to laboratory analysis, which leads to lack of knowledge about what is happening with the variable in real time. The Virtual Sensors owe their popularity to the fact that they can estimate the critical variable to the process through a learned or trained model from a historical data base, usually obtained by the physical process sensors.

One of the key issues in the development of Virtual Sensors is that most industrial processes are time-varying. That is, the physical process to which the model has been trained undergoes changes, whether sudden or smooth over time. This usually translates to a degradation of the Virtual Sensor's performance. To overcome this problem adaptive estimation methods should be used.

This dissertation proposes three methods for Virtual Sensors development. The first is an adaptive algorithm based on the concept of linear regression models, more specifically it is an hybridization of Recursive Least Squares - Adaptive (RLS-A) [Haykin, 1996] linear regression method with the Incremental Forward Stagewise Regression (IFSR) method [Hastie *et al.*, 2009] and is called Incremental Forward Stagewise Regression Online (IFSR-ON). Two other methods proposed are based on ensemble methods, and in the boosting concept. One is called Least Squares Boosting Single Variable (LSBST-SV), is nonadaptive, and is based on Least Squares Boosting (LSBST) method [Friedman, 2001]. The other method is an adptative version of the LSBST-SV method, is called Least Sqaures Boosting Single Variable Online (LSBST-SV-ON), and is based on a framework to make the operation of the boosting methods online [Babenko *et al.*, 2009].

For validation and discussion of the proposed methods, they were applied to thirteen reference data sets available in public repositories, and two time-varying real-world prediction problems. The performance of the proposed method was compared with some of the state-of-the-art methods for the Virtual Sensors development, namely: Recursive Least Squares (RLS), RLS-A, IFSR, Partial Least Squares (PLS), Recursive Partial Least Squares (RPLS), Least Absolute Shrinkage and Selection Operator (LASSO), Recursive Least Absolute Shrinkage and Selection Operator (RLASSO), Recursive Least Absolute Shrinkage and

Selection Operator with Adaptive Forgetting (RLASSO-AF), LSBST, and Least Squares Boosting Online (LSBST-ON).

In the thirteen reference data sets analyzed, the performances of the methods in offline operation, with the best performances were mostly obtained by PLS and LASSO methods, although the methods LSBST-ON and LSBST-SV-ON have obtained performances not very different from the other offline methods. For the two real-world time-varying problems, the best performance was obtained by the RPLS method, although the LSBST-SV-ON method had so comparable results, which reveals that it is a valuable tool for developing adaptive SSs.

Keywords: *Soft Sensor*, *Computational Intelligence*, *Online Learning*, *Preprocessing Data*, *Adaptive Modeling*.

Abreviaturas e Símbolos

Lista de Acrónimos

ANN	*Artificial Neural Network*
ARX	*Auto-Regressive with Exogeneous inputs*
CQO	Carência Química de Oxigénio
ETAR	Estação de Tratamento de Águas Residuais
FIS	*Fuzzy Inference System*
GB	*Gradient Boosting*
IFSR	*Incremental Forward Stagewise Regression*
IFSR-ON	*Incremental Forward Stagewise Regression Online*
LASSO	*Least Absolute Shrinkage and Selection Operator*
LSBST	*Least Squares Boosting*
LSBST-ON	*Least Squares Boosting Online*
LSBST-SV	*Least Squares Boosting Single Variable*
LSBST-SV-ON	*Least Sqaures Boosting Single Variable Online*
MAE	*Mean Absolute Error*
ME	Métodos *Ensemble*
MLP	*Multilayer Perceptron*
MSE	*Mean Squared Error*
MWPCA	*Moving Window Principal Component Analiysis*
NFS	*Neuro-Fuzzy Systems*
NRMSE	*Normalized Root Mean Squared Error*
PCA	*Principal Component Analysis*
PLS	*Partial Least Squares*
RLASSO	*Recursive Least Absolute Shrinkage and Selection Operator*
RLASSO-AF	*Recursive Least Absolute Shrinkage and Selection Operator with Adaptive Forgetting*
RLS	*Recursive Least Squares*
RLS-A	*Recursive Least Squares - Adaptive*
RLS-AF	*Recursive Least Squares with Adaptive Forgetting*
RPCA	*Recursive Principal Component Analysis*

RPLS	*Recursive Partial Least Squares*
SS	*Soft Sensor*
SSE	*Sum of Square Error*
SVM	*Support Vector Machine*
VIF	*Variance Inflation Factor*

Símbolos Gerais

c_γ	pequena constante para variar γ
c_λ	pequena constante para variar λ
$h\,()$	função básica
i	índice da nova amostra
j	índice da variável de entrada
l	número de variáveis latentes
m	número de variáveis
n	número de amostras
s	tamanho do passo no IFSR
$\mathbf{X}$	$\in \mathbb{R}^{n\times m}$ matriz de entrada
$\mathbf{x}_i$	$\in \mathbb{R}^{1\times m}$ novo vetor de dados de entrada
$\mathbf{x}_j$	$\in \mathbb{R}^{n}$ vetor das amostras da variável de entrada j
$x_{i,j}$	amostra i da variável de entrada j
$\mathbf{X}_{val}$	Matriz dos dados de entrada utilizados na fase validação
$\mathbf{y}$	$\in \mathbb{R}^{n}$ vetor de saída
$\hat{\mathbf{y}}$	$\in \mathbb{R}^{n}$ vetor de estimativas
$\overline{y}$	média do vetor $\mathbf{y}$
$\overline{\mathbf{y}}$	$\in \mathbb{R}^{n}$ vetor originado pelo produto $\mathbf{1}\overline{y}$
y_i	nova amostra de saída
λ	fator de esquecimento
σ	desvio padrão
σ^2	variância
$\mathbf{1}$	$\in \mathbb{R}^{n}$ vetor de uns

Métodos de Regressão Linear

$\mathbf{A}$	$\in \mathbb{R}^{n\times m}$ matriz de projeto
$\mathbf{a}_i$	$\in \mathbb{R}^{m}$ linha i da matriz de projeto
$\mathbf{B}$	$\in \mathbb{R}^{l\times l}$ matriz diagonal com os coeficientes de regressão
d	índice da iteração do PLS
$\mathbf{E}$	$\in \mathbb{R}^{n\times m}$ matriz dos residuais da entrada

$\mathbf{f}$	$\in \mathbb{R}^n$ vetor dos residuais da saída
$\mathbf{G}$	$\in \mathbb{R}^{m \times m}$ matriz de covariancias
$\mathbf{P}$	$\in \mathbb{R}^{m \times l}$ matriz de *loading* da entrada
$\mathbf{p}$	$\in \mathbb{R}^m$ vetor de *loading* de $\mathbf{P}$
$\mathbf{Q}$	$\in \mathbb{R}^{n \times l}$ matriz de *loading* da saída
$\mathbf{q}$	$\in \mathbb{R}^l$ vetor de *loading* da saída
$\mathbf{R}$	$\in \mathbb{R}^{n \times l}$ matriz residual do modelo interno
$\mathbf{r}$	$\in \mathbb{R}^n$ vetor de resíduos do IFSR
r_i	resíduo da amostra i do IFSR
$\mathbf{T}$	$\in \mathbb{R}^{n \times l}$ matriz de *score* da entrada
$\mathbf{t}$	$\in \mathbb{R}^n$ vetor de *score* da entrada
$\mathbf{U}$	$\in \mathbb{R}^{n \times l}$ matriz de *score* da saída
$\mathbf{u}$	$\in \mathbb{R}^n$ vetor de *score* da saída
$\mathbf{W}$	$\in \mathbb{R}^{m \times l}$ matriz de pesos
$\mathbf{X}_{new}$	$\in \mathbb{R}^{n \times m}$ matriz de entrada para o RPLS adaptativo
$\mathbf{y}_{new}$	$\in \mathbb{R}^n$ vetor de dados de saída para o RPLS adaptativo
$\boldsymbol{\beta}^{ifsr}$	$\in \mathbb{R}^m$ vetor de parâmetros do modelo do IFSR
β_j^{ifsr}	coeficiente j do modelo do IFSR
$\hat{\boldsymbol{\beta}}^{L1}$	$\in \mathbb{R}^m$ vetor de parâmetros do modelo do LASSO
γ	parâmetro de regulação
δ_j	tamanho do passo para a variável j no IFSR
$\boldsymbol{\epsilon}$	$\in \mathbb{R}^n$ vetor de resíduos
$\boldsymbol{\theta}$	$\in \mathbb{R}^m$ vetor de parâmetros do modelo
$\boldsymbol{\theta}_i$	$\in \mathbb{R}^m$ vetor de parâmetros do modelo com a contribuição da amostra i

Métodos Ensemble

$\mathbf{a}^{bst}$	$\in \mathbb{R}^m$ vetor de coeficientes do modelo do *boosting*
$\mathbf{a}_b^{bst}$	$\in \mathbb{R}^m$ vetor de coeficientes do modelo b do *boosting*
$a_{(j,b)}^{bst}$	*b-ésimo* modelo da variável j
b	índice do modelo
$F\,()$	modelo forte
f	predição final do *ensemble*
$F_b\,()$	modelo forte contendo a contribuição do modelo b
$f_{(b)}$	predição do modelo b do *ensemble*
g_i	gradiente negativo para a amostra i
$L\,()$	*loss function*

p	número de modelos
w_b	peso do modelo b do *ensemble*
η_i	taxa de aprendizagem para a amostra i
ρ	tamanho do passo do gradiente descendente
ρ_b	tamanho do passo do gradiente descendente para o modelo b

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Enquadramento

Os processos industriais são normalmente dotados de um elevado número de sensores. Os sensores têm por objetivo entregar dados para processos de monitorização e controlo. Nos últimos vinte anos surgiu uma tecnologia baseada em software, denominada de *Soft Sensor* (SS) ou Sensor Virtual, que é utilizada para a estimação online de variáveis. Os SSs têm ganho nos últimos anos um papel relevante a nível industrial (por exemplo, na deteção de falhas, monitorização de processos, na predição de variáveis críticas e no controlo). Um Sensor Virtual é uma ferramenta que aplica à informação disponível, adquirida por sensores tradicionais e por análises laboratoriais, técnicas de análise multi-variável de modo a realizar uma tarefa específica, como por exemplo a predição de variáveis críticas do processo. Normalmente os SSs são utilizados em situações em que o recurso a sensores convencionais para a medição de uma variável em tempo real torna-se impreciso, apresenta custos demasiado elevados, ou é mesmo impraticável (sendo necessário recorrer a análises laboratoriais). A nível industrial, as latências incutidas pela medição em laboratório de uma variável relevante para a qualidade de um produto, podem tornar impossível um controlo automático do processo de fabrico e levar à degradação da qualidade do produto. Uma estimativa online destas variáveis, oferecida por um Sensor Virtual, pode trazer um aumento significativo à qualidade do produto e uma boa redução ao custo de fabrico.

É usual distinguir-se dois tipos de Sensores Virtuais, *model-driven* (white-box) e *data-driven* (*black-box*). Os modelos *model-driven* pretendem descrever comportamentos físicos de processos através de modelação física-matemática. Por outro lado, os modelos *data-driven* são construídos sem qualquer conhecimento interno do processo (através de técnicas de regressão, como por exemplo: redes neuronais ou redes neuro-difusas). Nesta dissertação será utilizada apenas modelação *data-driven*.

Em [Fortuna *et al.*, 2007], o desenvolvimento de SS é descrito como sendo baseado em quatro passos principais:

1. Recolha dos dados e posterior análise - são selecionados os dados para o treino e validação do modelo;

2. Pré-processamento de dados - são utilizadas técnicas de deteção de *outliers*, de *missing data*, e de filtragem, e é feita a seleção das variáveis mais relevantes para o processo bem como dos respetivos atrasos;

3. Seleção e identificação do modelo - esta fase requer uma meticulosa seleção e treino do modelo, de modo a que este possa reproduzir corretamente a variável crítica do processo;

4. Manutenção do SS - tem o objetivo de manter uma boa resposta do SS sob a presença de alterações no processo.

No desenvolvimento de SSs tradicionais, a seleção de variáveis e o treino do modelo são feitos assumindo que o processo tem um comportamento estacionário (ou seja, que os dados são gerados sempre pela mesma distribuição) e que os dados históricos conseguem representar o processo ao longo do tempo. Contudo é frequente que processos industriais apresentem comportamentos não estacionários [Macias *et al.*, 2006], o que origina uma degradação do desempenho do Sensor Virtual [Kadlec *et al.*, 2011].

Este problema pode ser resolvido desenvolvendo um modelo que se consiga adaptar às alterações do processo de modo a que o SS mantenha um bom desempenho ao longo do tempo. São exemplo de metodologias para o desenvolvimento de SSs adaptativos o *Recursive Least Squares with Adaptive Forgetting* (RLS-AF) descrito em [Haykin, 1996, Sec. 16.10, pp. 734-735] e o *Recursive Least absolute Shrinkage and Selection Operator with Adaptive Forgetting* (RLASSO-AF) proposto em [Anagnostopoulos *et al.*, 2008]. Embora o algoritmo RLS-AF seja completamente online e a calibração dos seus parâmetros seja automática, o RLASSO-AF exige uma calibração rigorosa de alguns dos seus parâmetros.

1.2 Estado da Arte

Muitos processos industriais podem ser bastante complexos. Neste contexto, derivar as equações físicas, químicas, matemáticas (etc) que o descrevem pode ser impraticável, demasiado moroso e requerer um grande conhecimento do processo, o que por vezes é impossível. Os Sensores Virtuais baseados em modelação *data-driven* têm ganho relevo em aplicações industriais. Neste tipo de modelação o modelo do processo é obtido com base em dados medidos, não precisando de conhecimento *a-priori* sobre o processo. Segundo [Kadlec *et al.*, 2009], as técnicas de modelação *data-driven* mais utilizadas para desenvolver SS são *Principal Component Analysis* (PCA) em combinação com um modelo de regressão, *Partial Least Squares* (PLS), *Artificial Neural Networks* (ANN), *Neuro-Fuzzy Systems* (NFS) e *Support Vector Machines* (SVM).

Algumas técnicas de regressão linear têm sido largamente utilizadas no desenvolvimento de SSs. Essas técnicas pretendem estimar o valor de uma variável de saída com base num conjunto de variáveis de entrada, assumindo que as entradas do modelo e a função de regressão são lineares. Em [Jang *et al.*, 1997, Sec. 5.5, pp. 113-115] é descrita uma técnica muito conhecida, é a versão recursiva da abordagem *Least Squares* (LS), denominada de *Recursive Least Squares* (RLS). Contudo o método RLS tem como desvantagem a degradação do desempenho no caso de existir elevada colinearidade nos dados de entrada. A colinearidade nos dados pode ser avaliada através do *Variance Inflation Factor* (VIF) [Robinson and Schumacker, 2009]. O método PLS, descrito em [Qin, 1998], deve a sua popularidade para o desenvolvimento de SS à sua robustez face a ruído e a elevada colinearidade nos dados de entrada. O método PLS foca-se na relação entre as variáveis de entrada e as de saída, tendo em conta a matriz de covariâncias que as relaciona. O algoritmo mantém um bom desempenho para dados colineares, pois consegue decompor o espaço de entrada e de saída simultaneamente, enquanto mantém as restrições de ortogonalidade. O método PLS pode ser útil quando é necessário estimar uma variável com base num elevado número de preditores independentes. Em [Wang *et al.*, 2010] é aplicado um SS, construído com base no PLS, ao processo industrial de uma refinaria. Dois métodos que podem também ter interesse no caso em que a dimensionalidade do espaço de entrada é elevada são o *Least absolute Shrinkage and Selection Operator* (LASSO) e o *Incremental Forward Stagewise Regression* (IFSR), apresentados em [Tibshirani, 1996] e [Hastie *et al.*, 2009] respetivamente. Durante o desenvolvimento de Sensores Virtuais com as técnicas de regressão linear referidas anteriormente, é assumido que o comportamento do processo seria estacionário e que os dados históricos são capazes de representar o processo ao longo do tempo. Contudo, a maioria dos processos industriais sofre alterações ao longo do tempo, e por este motivo o desempenho de um SS baseado nas técnicas de regressão linear já referidas iria degradar-se.

Para contrariar esta perda de desempenho dos SSs, estes podem ser desenvolvidos a partir de modelos com a capacidade de se adaptar às alterações do processo, ou seja o desenvolvimento de SSs deve ser baseado em métodos adaptativos. Um bom exemplo de um método adaptativo é o RLASSO-AF proposto em [Anagnostopoulos *et al.*, 2008], que deve a sua capacidade adaptativa ao método RLS-AF. Em [Dayal and MacGregor, 1997], o método *Recursive Least Squares - Adaptive* (RLS-A) [Haykin, 1996, Sec. 5.6, pp. 116-117] é considerado o mais utilizado *online* para estimação recursiva dos parâmetros do modelo. Em [Souza *et al.*, 2010], o RLS-A foi utilizado para construir um método de seleção de variáveis, uma problemática muito importante para SS. Em [Dayal and MacGregor, 1997], o método RLS-A foi aplicado a um circuito de flotação de uma indústria de minerais e foi aplicado no controlo adaptativo de um tanque reactor agitado contínuo, usualmente utilizado em engenharia química. Apesar da existência de inumeros casos de sucesso do RLS-A, este método mantém a mesma desvantagem que o método não adaptativo RLS exibe perante dados com colinearidade.

Para resolver o problema da perda de desempenho face a elevadas colinearidades nos

dados de entrada do método RLS-A, foram propostos métodos adaptativos como *Recursive Principal Component Analiysis* (RPCA), o *Moving Window Principal Component Analiysis* (MWPCA) e o *time-lagged* PCA, apresentados em [Li *et al.*, 2000], [Wang *et al.*, 2005] e [Lee *et al.*, 2004], respetivamente. Estes algoritmos reduzem o número de variáveis de entrada, associando estas através de combinações lineares, de forma a obter variáveis ortogonais entre si e que cubram uma maior variância do espaço de entrada. Estes algoritmos são ferramentas que conseguem lidar com a colinearidade e reduzir a dimensão do espaço de entrada, o que os torna bastante úteis em processos industriais. Uma das limitações destes algoritmos é que apenas conseguem lidar com relações lineares. Outras limitações, residem na escolha do número ótimo de componentes principais (costuma ser resolvido por técnicas de validação cruzada) e no facto de os componentes principais descreverem otimamente o espaço de entrada mas não a relação entre o espaço de entrada e o de saída.

Em resposta às limitações dos algoritmos baseados em PCA, surgiram métodos como *Recursive Partial Least Squares* (RPLS) [Qin, 1998], e uma versão do PLS utilizando *moving window* [Dayal and MacGregor, 1997]. O método RPLS tem sido aplicado por diversos autores para o desenvolvimento de SSs adaptativos. Em [Dayal and MacGregor, 1997] foi aplicado aos mesmos dois processos que o método RLS-A, ao circuito de flotação de uma indústria de minerais e no controlo adaptativo de um reator tanque agitado contínuo, e em [Qin, 1998] o RPLS foi aplicado a um reformador catalítico. O método RPLS tem como limitação apenas conseguir modelar relações lineares entre os dados.

Segundo [Soares *et al.*, 2011] uma metodologia que tem recentemente ganho protagonismo no desenvolvimento de SSs são os Métodos *Ensemble* (ME). Estes métodos baseiam-se na combinação de um conjunto de modelos individuais de modo a que o desempenho da previsão global seja robusto. Uma das motivações para a utilização de métodos *emsemble* é que geralmente o conjunto de modelos é mais preciso do que qualquer um dos modelos individuais que o constituem, sendo que factores importantes para o seu sucesso são diversidade dos modelos individuais e na forma como estes são combinados. Uma das técnicas mais utilizadas na criação de ME são os métodos de *boosting*, que inicialmente foram projetados para resolver problemas de classificação, sendo mais tarde aplicados com êxito a problemas de regressão. O *boosting* baseia-se no treino de vários modelos fracos e na sua combinação de forma a obter um modelo robusto e poderoso. Uma versão não adaptativa de *boosting*, é apresentada em [Friedman, 2001], denominada de *Least Squares Boosting* (LSBST). O LSBST é um método não adaptativo, o que origina que um SS baseado neste método sofra uma degradação do desempenho quando utilizado em processos com comportamento não estacionário. Uma *framework* que permite tornar o LSBST apto para funcionamento *online* é apresentada em [Babenko *et al.*, 2009]. Os dois métodos *boosting* referidos têm a incapacidade de não conseguir modelar relações não lineares, tal como os métodos de regressão linear referidos anteriormente.

Em [Sliskovic *et al.*, 2011], as *Artificial Neural Networks* (ANNs) são consideradas uma técnica muito conhecida para modelação *data-driven* de sistemas não lineares. A intenção

original das ANNs era construir modelos computacionais baseados no funcionamento dos neurónios biológicos, que são os responsáveis pelo processamento da informação do sistema nervoso. As ANNs são constituídas por elementos de processamento chamados por nós (ou neurónios). As ANNs têm a propriedade de aproximadores universais, no entanto pode acontecer que a relação entre o número de amostras e o número de parâmetros do modelo (pesos das sinopses, número de nós da camada oculta) seja demasiado elevado, o que origina elevados tempos de computação para o treino do modelo. A *MultiLayer Perceptron* (MLP) é a mais popular das ANNs. As redes MLP são compostas por uma camada de neurónios de entrada, uma camada de neurónios de saída, e uma ou mais camadas intermédias (*hidden layers*). Nas MLP todos os neurónios de uma camada têm ligações a todos os neurónios da camada seguinte, não existindo ligações entre nós da mesma camada, e a cada ligação está associado um peso que determina a importância que cada entrada do nó terá na respetiva saída. O algoritmo de aprendizagem mais utilizado pelas MLP é o algoritmo de retro-propagação do erro ou *backpropagation*, sendo uma apresentação deste algoritmo feita em [Rumelhart *et al.*, 1986]. Uma das limitações das ANNs é poderem ficar presas num óptimo local durante o processo de aprendizagem, o que pode originar um desempenho sub-ótimo. Outra dificuldade reside na escolha da topologia correta para que a rede tenha o melhor desempenho. Existe ainda a questão que o conhecimento adquirido existe apenas na forma de pesos entre neurónios, não havendo uma representação perceptível a nível do operador humano.

A *Support Vector Machine* (SVM) é um método de aprendizagem supervisionada, que tal como as ANNs é um aproximador universal. Tem as seguintes vantagens em relação às ANNs, não fica preso em mínimos locais, menor dependência da quantidade e qualidade dos dados de treino e tem melhor qualidade de generalização. Este algoritmo foi implementado com sucesso em SS, em [Gomnam and Jazayeri-rad, 2013]. Embora tenham sido reportados muitos casos de sucesso, este algoritmo ainda tem como limitação o facto de o aumento da complexidade computacional ser diretamente proporcional à dimensão do *data set*.

Outra técnica importante são os *Neuro-Fuzzy Systems* (NFS) [Jang *et al.*, 1997]. Os NFS são um método de inteligência computacional, com características híbridas, que combinam a aprendizagem e a característica de aproximador universal das ANNs, com o raciocínio similar ao humano dos *Fuzzy Inference System* (FIS). A intenção da fusão destas duas técnicas é obter um sistema de aprendizagem que combine as vantagens de ambas as abordagens e ao mesmo tempo lide com as suas desvantagens.

1.3 Objetivos

Esta dissertação tem como principal objetivo desenvolver e comparar metodologias para implementação de Sensores Virtuais. Em particular, o objetivo é desenvolver métodos para a construção de um SS *data-driven* adaptativo. Especificamente, os objetivos desta dissertação

são:

- Implementar e testar os métodos de regressão linear RLASSO-AF e IFSR propostos em [Anagnostopoulos *et al.*, 2008] e [Hastie *et al.*, 2009], respetivamente;
- Com base nos métodos RLS-A e IFSR apresentados em [Haykin, 1996] e [Hastie *et al.*, 2009] respetivamente, desenvolver um algoritmo de Sensor Virtual adaptativo;
- Implementar e testar, os métodos de *boosting* propostos em [Friedman, 2001; Babenko *et al.*, 2009];
- Com base nos métodos de *boosting* de [Friedman, 2001; Babenko *et al.*, 2009] e em alguns conceitos de seleção de modelos presentes em [Bishop *et al.*, 2006], desenvolver duas variantes dos algoritmos implementados no ponto anterior;
- Aplicar os métodos não adaptativos implementados a um conjunto de *data-sets* existentes em repositórios públicos;
- Aplicar os métodos adaptativos implementados a dois *data-sets* variantes no tempo existentes em repositórios públicos, nomeadamente a um reator de polimerização e a uma Estação de tratamento de águas residuais (ETAR);
- Comparar o desempenho dos algoritmos implementados.

1.4 Trabalho Realizado

Neste trabalho foram estudadas e/ou implementadas várias metodologias para desenvolvimento de SSs. Foram implementados os seguintes seis algoritmos com funcionamento não adaptativo:

1. *Recursive Least Squares* (RLS) [Jang *et al.*, 1997];
2. PLS [Qin, 1998];
3. Algoritmo *Least absolute Shrinkage and Selection Operator* (LASSO) [Hastie *et al.*, 2009];
4. *Incremental Forward Stagewise Regression* (IFSR) [Hastie *et al.*, 2009];
5. *Least Squares Boosting* (LSBST) [Friedman, 2001];
6. *Least Squares Boosting Single Variable* (LSBST-SV) [Friedman, 2001; Bishop *et al.*, 2006].

Foram implementados os seguintes sete algoritmos para SSs adaptativos:

1. RLS-A [Haykin, 1996];

2. RPLS [Qin, 1998];
3. *Recursive Least absolute Shrinkage and Selection Operator* (RLASSO) [Anagnostopoulos *et al.*, 2008];
4. RLASSO-AF [Anagnostopoulos *et al.*, 2008];
5. *Incremental Forward Stagewise Regression Online* (IFSR-ON) [Haykin, 1996; Hastie *et al.*, 2009];
6. *Least Squares Boosting Online* (LSBST-ON) [Babenko *et al.*, 2009];
7. *Least Squares Boosting Single Variable Online* (LSBST-SV-ON) [Babenko *et al.*, 2009; Bishop *et al.*, 2006].

Os algoritmos IFSR-ON, LSBST-SV e LSBST-SV-ON são novos e constituem um dos contributos desta dissertação. Outros contributos da dissertação são o desenvolvimento de algoritmos acima listados para implementação de sensores virtuais e a sua comparação em *data sets* reais.

Os algoritmos estudados, constroem o modelo para o SS apenas com base no conhecimento empírico das variáveis de entrada e de saída do processo, não tendo acesso a qualquer informação sobre quais as melhores variáveis ou sobre o funcionamento do processo.

Para a validação dos algoritmos foram utilizados treze *data sets* de referência disponíveis em repositórios públicos e dois *data sets* reais variantes no tempo.

É de realçar que todos os algortimos foram desenvolvidos de raiz por mim, com a exepção dos métodos: LASSO, PLS e RPLS, que foram implementados com base em código disponibiizado já existente. Os algoritmos foram desenvolvidos com recurso à ferramenta Matlab e posteriormente foram efetuados testes com os *data sets* referidos de forma a validar, analisar e comparar o comportamento, eficácia e robustez dos algoritmos.

1.5 Organização da Dissertação

Esta dissertação está estruturada em sete capítulos, organizados da seguinte forma:

- No Capítulo 1, é feita uma introdução ao trabalho desenvolvido, é explicada a relevância do tema escolhido e os objetivos da dissertação;
- No Capítulo 2, são abordadas a aplicação, importância e o desenvolvimento dos Sensores Virtuais;
- No Capítulo 3, é feita uma breve descrição aos algoritmos de regressão linear estudados;
- No Capítulo 4, é feita uma introdução ao conceito de métodos *ensemble*, são apresentadas quatro metodologias de *boosting* sendo duas delas propostas nesta dissertação. É ainda descrita a contribuição dos dois métodos *boosting* propostos;

- No Capítulo 5, são apresentados os resultados experimentais de modo a comprovar a eficácia dos algoritmos propostos nos Capítulos 3 e 4;
- No Capítulo 6, são apresentadas as conclusões acerca do trabalho desenvolvido, assim como possíveis trabalhos futuros.

Capítulo 2

Sensores Virtuais

Os Sensores Virtuais podem ser definidos como sensores baseados em software e inteligência artificial. Por outras palavras, esta tecnologia pode ser definida como um conjunto de metodologias de predição que utilizam dados recolhidos por sensores existentes no processo para a estimação das variáveis que não podem ser medidas devidamente.

Atualmente os SSs são uma mais-valia em aplicações industriais. Em processos industriais os sensores são normalmente obrigados a operar em ambientes hostis, o que exige que estes sejam projetados para estas condições específicas e que sejam elaborados protocolos de manutenção para tentar garantir o seu bom funcionamento mas que não garantem a ausência de falhas. Um dos problemas no controlo de processos industriais surge quando os sensores tradicionais não têm capacidade de medir automaticamente variáveis críticas do processo, ou medem de forma imprecisa, com custos elevados ou com atrasos significativos (por exemplo por meio de análises laboratoriais). Os Sensores Virtuais representam uma solução para a dificuldade de medir algumas variáveis relevantes do processo. Por exemplo, na indústria da pasta de papel é importante obter o valor de carência química de oxigénio (CQO), pois trata-se de uma variável que influencia diretamente a qualidade do produto. Este valor é medido em laboratório, trazendo uma latência incomportável para sistemas de controlo automático.

As metodologias de desenvolvimento de Sensores Virtuais são classificadas como, *white-box*, *black-box* (ou *data-driven*) e *grey-box*. Estas metodologias serão abordadas com mais detalhe na Secção 2.2. Esta tese irá abordar a utilização de sensores virtuais do tipo *data-driven*, pois estes são baseados na obtenção de um modelo a partir de dados históricos do processo, compostos por medições das entradas e da saída do sistema.

Segundo [Fortuna *et al.*, 2007], a utilização de Sensores Virtuais tem as seguintes vantagens:

- Representam uma alternativa de custo reduzido, quando os sensores físicos são demasiado caros;
- Podem operar em paralelo com os sensores físicos disponíveis, fornecendo informações

úteis para a deteção de falhas;

- Possível implementação em hardware existente (e.g. microcontroladores);
- Permitem uma estimação em tempo real das variáveis do processo, contornando os atrasos introduzidos por sensores ou métodos de medição mais lentos e aumentando o desempenho das estratégias de controlo.

2.1 Aplicações

Os Sensores Virtuais têm uma gama de aplicações muito abrangente, o que permite que sejam utilizados em praticamente qualquer campo industrial (por exemplo na indústria do papel, química, refinarias, etc). O aparecimento de aplicações de Sensores Virtuais deve-se maioritariamente à quantidade de sistemas de tempo real que recolhem informação *online* do processo (por exemplo através de informação obtida a partir de sensores) e sistemas digitais usados para monitorização e controlo. Algumas características dos sistemas a serem controlados também encorajam a aplicação deste tipo de sensores (por exemplo medição em ambiente hostil, parâmetros do processo difíceis de medir, *delays* de medição, etc).

Os SSs são ferramentas que têm visto a sua aplicação como sensores de rotina em sistemas de controlo em malha aberta e recentemente verificou-se um rápido aumento das suas aplicações em sistemas de controlo em malha fechada ou em sistemas de controlo adaptativos.

A utilização de Sensores Virtuais pode resolver problemas tais como, dispositivos de medida de *back-up*, predição de variáveis em tempo real para controlo do processo, validação de sensores físicos e estratégias de diagnóstico de falhas.

2.1.1 *Back-up* de Dispositivos de Medição

Algumas plantas industriais exigem um número elevado de sensores para efeitos de monitorização e controlo, e que rotineiramente adquirem uma grande quantidade de dados. Estes dispositivos são obrigados a operar em condições de ambiente hostis, o que exige equipamentos de hardware robustos e procedimentos de manutenção periódicos. Mesmo tendo em conta os cuidados anteriores, é frequente ocorrer falhas nos dispositivos de medição, sejam elas na forma de mudanças abruptas no modo de funcionamento dos dispositivos, ou na forma de uma mudança lenta das características metrológicas.

Tanto as intervenções de manutenção preventiva, como as de manutenção corretiva, exigem, por vezes, que o hardware de medição seja desligado. É típica a utilização de Sensores Virtuais como *back-up* de dispositivos de medição. O SS estima a saída do dispositivo de medida indisponível, com base em entradas auxiliares relacionadas com a variável a estimar. O objectivo é evitar a degradação do desempenho da planta e a subida dos custos, durante a substituição momentânea de dispositivos de medição indisponíveis.

2.1.2 Redução da Necessidade de Sensores Físicos (*Hardware*)

Em processos industriais, medir algumas variáveis em tempo real pode exigir a aquisição de sensores dispendiosos. Assim a utilização de uma ferramenta de software como alternativa a um sensor tradicional pode representar uma possível poupança económica.

O problema desta alternativa, é que usualmente a calibração dos SSs requer a comparação com dados de sensores tradicionais. Este problema é mais evidente em SSs não adaptativos. Uma possível solução é a adicional utilização temporária do sensor físico.

2.1.3 Estimação em Tempo Real

Todos os dispositivos de medição demoram um intervalo finito de tempo para executar uma medição. Em situações em que este tempo é diretamente proporcional à dinâmica do sistema, o tempo de medida pode ser uma fonte significativa de atraso. No caso em que estas medições são usadas em sistemas de controlo em malha fechada este atraso pode ser muito negativo, degradando o desempenho do sistema de controlo até ao ponto em que o hardware de medição deixa de ser adequado para a aplicação de controlo e monitorização.

A estimação de variáveis do sistema, em tempo real, obtidas através de Sensores Virtuais, principalmente em situações em que a medição desta variável por meio de sensores físicos apresenta atrasos, representa a mais comum aplicação dos SSs bem como beneficia o desempenho do sistema.

2.1.4 Validação, Deteção de Falhas e Diagnósticos de Sensores

Pode-se interpretar um sistema de controlo industrial como uma hierarquia de três níveis. O primeiro nível, também chamado nível de controlo, é encarregue por implementar a malha de controlo. O segundo nível é o de supervisão, responsável pela monitorização contínua do processo ao longo do tempo. O terceiro nível trata da gestão, coordenação e atividades de otimização.

A deteção de falhas e diagnóstico são parte das funções no nível de supervisão. Atualmente são realizadas por intermédio de técnicas de modelação matemática avançadas, inteligência computacional, entre outras. Os sistemas de deteção de falhas e diagnósticos modernos, têm como principais objetivos:

- Efetuar a deteção precoce de falhas e sempre que possível fornecer o máximo de informações sobre a falha que ocorreu ou está a ocorrer;
- Fornecer um sistema de apoio às decisões de manutenção e reparações;
- Fornecer uma base ao desenvolvimento de sistemas tolerantes a falhas.

A ideia fundamental de todas as técnicas de deteção de falhas é comparar a informação adquirida a partir do sistema a ser monitorizado com a correspondente informação adqui-

rida de uma fonte redundante, sendo a deteção da falha feita se o conjunto de informações recolhidas for diferente.

Assim, um sensor de validação é considerado a base de um tipo particular de método de deteção de falhas, no qual o sistema a ser monitorizado é observado por um, ou vários, sensores. Num nível básico, o sensor de validação tem como objetivo proporcionar aos utilizadores de um sistema de medição (operadores humanos, bases de dados de medições, sistemas de controlo, outros processos, etc) uma avaliação da confiabilidade da medição realizada. A um nível mais avançado, um sensor de validação pode fornecer uma estimativa da medição, caso o sensor real não se encontre funcional.

2.2 Desenvolvimento de Sensores Virtuais

Geralmente, consideram-se três modelos distintos para o desenvolvimento de SSs, denominados *white-box*, *black-box* e *grey-box*. Os modelos *white-box*, também chamados modelos *model-driven*, são baseados na construção do modelo através do conhecimento físico ou químico, do processo que se deseja modelar. Embora os modelos possam refletir 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 fatores, o que origina, devido ao grande número de variáveis de entrada (algumas delas com características, efeitos e relações 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 esta abordagem impraticável em muitos casos.

Os modelos *black-box* ou *data-driven* são baseados em observações empíricas de processos, 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, os métodos *least-squares* (LS), *auto-regressive with exogenous inputs* (ARX), *nonlinear auto-regressive moving average with exogenous inputs* (NARMAX), *multilayer perceptron* (MLP) ou *support vector machine* (SVM). Os modelos *data-driven* fornecem uma melhor abordagem porque usam dados dos processos reais como base para o modelo. Embora esta abordagem seja pouco transparente quanto ao conhecimento físico sobre o processo que se deseja modelar, a modelação *black-box* torna-se muito mais prática. As dificuldades com este tipo de abordagem residem na escolha do método a ser utilizado e na aprendizagem do modelo.

Finalmente, o terceiro tipo de modelos, designados *grey-box* ou modelos híbridos, são uma abordagem baseada na combinação de características dos outros dois tipos de modelos. Estes modelos incorporam o conhecimento prévio sobre o funcionamento dos processos, e a modelação e predição a partir dos dados dos processos.

Tipicamente, o desenvolvimento de Sensores Virtuais pode ser resumido em cinco passos:

1. Recolha de dados e posterior análise;
2. Pré-processamento dos dados;
3. Seleção e identificação do modelo;
4. Validação do modelo;
5. Manutenção do SS.

Estes cinco passos irão ser abordados seguidamente.

2.2.1 Recolha de Dados e Posterior Análise

A estratégia utilizada na recolha de dados e a análise crítica destes, são fundamentais para a identificação do modelo do sistema. A questão mais importante nesta fase é a escolha da frequência de amostragem. Deve escolher-se uma frequência que não seja demasiado baixa para que o modelo consiga ter a perceção da dinâmica do processo, e que não seja demasiado alta para não prejudicar desnecessariamente o desempenho do Sensor Virtual, em tarefas de aprendizagem ou regulação de outros parâmetros (dependendo do método usado). Do mesmo modo, devem ser também selecionadas as variáveis importantes para o processo. Estas devem ser sincronizadas, pois possuem geralmente frequências de amostragem diferentes.

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 sobre o processo e, sendo assim, são removidos visto que não irão ser utilizados no processo de aprendizagem do modelo.

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

2.2.2 Pré-processamento de Dados

Depois da recolha de dados é necessário efetuar o pré-processamento destes. Esta fase tem como objetivo analisar os dados de forma a que o desenvolvimento e cálculo do modelo se torne mais efetivo. Para tal, são utilizadas técnicas de deteção de *outliers*, *missing data*, filtragem, e a seleção de variáveis relevantes e respetivos atrasos temporais.

O pré-processamento dos dados é realizado de forma iterativa, por exemplo, a seleção de variáveis é realizada repetidamente até se considerar que os dados já estão prontos para serem usados nas fases seguintes.

2.2.2.1 Deteção de Dados Inconsistentes (*Outliers*)

Determinados acontecimentos durante a recolha de dados podem tornar a sua análise mais complexa e incorreta. Por exemplo, uma falha de Hardware, um problema de transmissão ou uma falha na medição pode originar a presença de *outliers* no conjunto de dados recolhidos.

Chama-se *outlier* a uma observação que é inconsistente em relação ao restante conjunto de dados, sendo segundo [Kadlec *et al.*, 2009] a sua deteção crucial no desenvolvimento de SSs, uma vez que a sua presença é prejudicial para o desempenho dos modelos.

Algumas técnicas de deteção de *outliers*, que ajudam a resolver estes problemas são, por exemplo, PCA, PLS, e a regra 3σ. Em [Fortuna *et al.*, 2007] é apresentado um caso de estudo comparando diferentes técnicas para este fim, incluíndo as acabadas de enumerar.

Muitos autores defendem que a deteção de *outliers* não pode ser uma técnica totalmente automática, pelo que para torná-la mais robusta deve usar-se para além de técnicas de processamento automático, procedimentos de validação "manual".

2.2.2.2 Deteção de Falta de Dados (*Missing Data*)

Os dados em falta correspondem a uma ou mais medidas de uma ou mais variáveis de entrada que não foram realizadas, ou foram realizadas com erros. Normalmente este problema existe em todos os processos industriais e é causado, por exemplo, por amostras em falta devido a problemas de comunicação ou a falhas nos sensores de hardware. Para preencher as faltas de dados (*missing data*) podem ser usadas técnicas de interpolação e de correlação. Outra técnica de tratamento de dados em falta pode passar por ignorar e descartar um conjunto de dados. Em [Nelson *et al.*, 1996] as técnicas PCA e PLS foram utilizadas para a deteção de *missing data*.

2.2.2.3 Filtragem

Mesmo depois do tratamento dos dados para a eliminação dos problemas gerados pelos dados em falta e dos *outliers*, pode ocorrer que os mesmos ainda não sejam apropriados para o desenvolvimento de *soft sensors*. Nestes casos os dados são submetidos a operações de pré-filtragem com o intuito de remover ruídos de alta frequência, *offsets* e efeitos sazonais. Em [Kaur and Dewan, 2010] é feita uma pesquisa bibliografica sobre metodologias de pré-filtragem.

2.2.2.4 Seleção de Variáveis e Atrasos

A complexidade do modelo de um processo depende do número de variáveis. Muitas vezes, pode acontecer não haver conhecimento do modelo do processo ou não se conhecer o significado físico das variáveis de entrada (podendo estas ser redundantes ou irrelevantes no processo de aprendizagem), sendo necessário aplicar-se um algoritmo de seleção de variáveis, de modo a determinar quais as variáveis de entrada que mais influenciam a variável a estimar. Assim, o objetivo da seleção de variáveis é garantir um maior desempenho e robustez do sensor virtual, diminuindo a complexidade do modelo do processo.

Mesmo depois de selecionadas as variáveis mais importantes para o processo, pode ocorrer que estas não tenham o maior impacto de forma instantânea no processo físico. Para que a

estimação da variável pretendida seja mais rigorosa é necessário selecionar atrasos específicos para cada variável de entrada, de modo a que cada uma delas tenha a maior contribuição possível para a estimação. Em [Souza *et al.*, 2010] e [Anagnostopoulos *et al.*, 2008] são apresentados vários algoritmos para a resolução destes dois problemas para SSs *data-driven*.

2.2.3 Seleção e Identificação do Modelo

Esta fase é o ponto principal para o desempenho do SS. Como o modelo é a base do Sensor Virtual, a seleção ótima deste é muito importante para a qualidade de predição do sensor. Apesar da sua importância não existe nenhuma abordagem teórica para resolver este problema que seja claramente superior a qualquer outra, sendo a seleção do modelo feita de forma *"ad hoc"* para cada SS. Segundo [Kadlec *et al.*, 2009] a seleção do modelo depende da experiência do projetista e da sua preferência pessoal.

Isto revela um aspeto importante da modelação *black-box*, para a qual quaisquer conhecimentos relativos às variáveis do processo, à ordem do sistema, à gama de operação, a atrasos temporais, ao grau de não-linearidade, às frequências de amostragem, representam uma importante informação ou ferramenta que deve ser tida em conta pelo projetista.

2.2.4 Validação do Modelo

A validação do modelo tem como objetivo a realização de testes para avaliar e verificar o desempenho do modelo escolhido. Os dados utilizados neste procedimento são por regra diferentes dos dados utilizados no treino do modelo (dados de treino).

Para avaliar o desempenho do modelo existem várias ferramentas como, por exemplo, o *Mean Squared Error* (MSE), o Coeficiente de Correlação (CC), o erro quadrático total (*Sum of Square Error*, SSE) e 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.

2.2.5 Manutenção do Sensor Virtual

Depois de desenvolvido o SS, este precisa de manutenção regular. A manutenção do Sensor Virtual tem como objetivo manter o seu desempenho ao longo do tempo.

Normalmente no desenvolvimento do SS, a construção do modelo é feita assumindo que o comportamento do processo é estacionário e que os dados históricos são suficientes para representar o processo ao longo do tempo. Contudo, na prática as suposições anteriores podem não ocorrer e é frequente que os processos industriais apresentem um comportamento não estacionário [Macias *et al.*, 2006]. Nestas situações existe uma degradação no desempenho do Sensor Virtual de acordo com [Kadlec *et al.*, 2011]. Para mitigar este efeito, é necessária calibração periódica do Sensor Virtual.

Este problema pode ser resolvido desenvolvendo um modelo que se consiga adaptar às alterações do processo de modo a que o SS mantenha um bom desempenho ao longo do tempo. Entre outras metodologias referidas no Estado da Arte, em [Souza *et al.*, 2010] e [Qin, 1998] são apresentadas duas metodologias para o desenvolvimento de SSs adaptativos.

No caso em que o SS é adaptativo a periodicidade da manutenção necessária é consideravelmente inferior, o que pode consequentemente originar uma redução nos custos com análises laboratoriais.

Capítulo 3

Métodos de Regressão Linear

Sendo n o número de amostras e m o número de variáveis de entrada, o objetivo da regressão é prever o valor real do vetor de saída $\mathbf{y} \in \mathbb{R}^n$ (que contém as n amostras da saída) com base na matriz de entrada $\mathbf{X} \in \mathbb{R}^{n \times m}$ (que contém as n amostras das m variáveis de entrada), dado um *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$ de observações do modelo, onde $\mathbf{x}_i \in \mathbb{R}^{1 \times m}$ é uma nova amostra das variáveis de entrada (ou seja é uma linha da matriz $\mathbf{X}$), y_i é uma nova amostra da saída (ou seja é o *i-ésimo* elemento de $\mathbf{y}$) e i é o índice da amostra. Um modelo linear assume que a função de regressão é linear. Considerando que a saída y_i pode ser aproximada por uma função genérica $h(\mathbf{x}_i, \boldsymbol{\theta})$, caraterizada pelo vetor de parâmetros do modelo $\boldsymbol{\theta} \in \mathbb{R}^{m+1}$, esta é parametrizada em função das variáveis de entrada $\mathbf{x}_i$, da seguinte forma:

$$y_i = h(\mathbf{x}_i, \boldsymbol{\theta}) + \varepsilon_i, \tag{3.1}$$

onde ε_i é o erro residual da amostra i, ou seja, é o elemento i contido no vetor de resíduos $\boldsymbol{\varepsilon} \in \mathbb{R}^n$ o qual se assume ter uma distribuição normal de média nula e variância σ^2.

3.1 *Least Squares*

Em 1795, Karl Gauss propôs a técnica dos mínimos quadrados para fazer a predição do movimento de planetas e cometas utilizando medidas adquiridas por telescópio [Sorenson, 1970]. Esta técnica não foi utilizada até que Plackett a rescreveu em 1950 em [Plackett, 1950]. Quer a técnica dos *Least Squares*, quer as suas múltiplas variantes têm sido muito utilizadas para resolução de problemas de estimação em diversos campos de aplicação.

Dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$ podemos escrever a saída estimada $\hat{\mathbf{y}} \in \mathbb{R}^n$ pelo modelo de regressão linear na forma parametrizada, denominado *Least Squares*:

$$\hat{\mathbf{y}} = \mathbf{1}\theta_0 + \sum_{i=1}^{n} \mathbf{x}_j \theta_j, \tag{3.2}$$

onde, $\mathbf{x}_j \in \mathbb{R}^n$ é o vetor de amostras da variável j (ou seja uma coluna da matriz $\mathbf{X}$), θ_j é o coeficiente de índice j do vetor de parâmetros do modelo $\boldsymbol{\theta}$, θ_0 é o coeficiente de índice

zero do vetor de parâmetros do modelo $\boldsymbol{\theta}$, m é o número de variáveis de entrada e $\mathbf{1} \in \mathbb{R}^n$ é um vetor cujo valor de todos os seus coeficientes é igual a um. Os coeficientes de $\boldsymbol{\theta}$ que minimizam o SSE são dados por:

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}^{aux}} \sum_{i=1}^{n} \left(y_i - \theta_0^{aux} - \sum_{j=1}^{m} x_{(i,j)} \theta_j^{aux} \right)^2, \tag{3.3}$$

onde $x_{(i,j)}$ é um elemento da matriz $\mathbf{X}$ e corresponde à amostra i da variável de entrada j.

No caso em que $\mathbf{A}^T\mathbf{A}$ é *nonsingular*, o modelo linear do estimador LS pode ser dado na forma matricial:

$$\boldsymbol{\theta} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y}. \tag{3.4}$$

onde, $\mathbf{A} \in \mathbb{R}^{n \times m+1}$, chamada de matriz de projeto e é da forma: $\mathbf{A} = \begin{bmatrix} \mathbf{1}, & \mathbf{X} \end{bmatrix}$. É de notar que $(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ é a pseudo-inversa de $\mathbf{A}$, para o caso particuar em que $\mathbf{A}$ é uma matriz de *full column rank*.

3.2 *Recursive Least Squares*

A variante mais conhecida da família de métodos LS é a sua versão recursiva, denominada de RLS [Jang *et al.*, 1997, Sec. 5.5, pp. 113-115]. O método RLS atualiza o vetor de parâmetros $\boldsymbol{\theta}$ incrementalmente, não sendo necessário guardar os dados históricos. O RLS tem elevada rapidez de convergência, no entanto esta rapidez de convergência é conseguida com um aumento da complexidade computacional. Uma grande desvantagem deste método é mostrar dificuldades na estimação quando utilizado em dados com elevada colinearidade.

Sendo $(\mathbf{a}_i, y_i)$ uma nova amostra, onde $\mathbf{a}_i \in \mathbb{R}^{m+1}$ é uma nova linha da matriz de projeto dada por:

$$\mathbf{a}_i = \begin{bmatrix} 1, & \mathbf{x}_i \end{bmatrix}^T, \tag{3.5}$$

pode-se fazer a atualização incremental, quando uma nova amostra está disponível, do modelo com as seguintes equações:

$$\mathbf{G}_i = \mathbf{G}_{i-1} - \frac{\mathbf{G}_{i-1}\mathbf{a}_i\mathbf{a}_i^T\mathbf{G}_{i-1}}{1 + \mathbf{a}_i^T\mathbf{G}_{i-1}\mathbf{a}_i}, \tag{3.6}$$

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} + \mathbf{G}_i\mathbf{a}_i(y_i - \mathbf{a}_i^T\boldsymbol{\theta}_{i-1}), \tag{3.7}$$

onde $\boldsymbol{\theta}_i \in \mathbb{R}^{m+1}$ é o vetor de parâmetros do modelo com a contribuição da *i-ésima* amostra e $\mathbf{G}_i \in \mathbb{R}^{m+1 \times m+1}$ é chamada de matriz de covariâncias, com a contribuição da amostra i, pois é proporcional à covariância do estimador e é dada por:

$$\mathbf{G}_i = (\mathbf{A}_i^T\mathbf{A}_i)^{-1}. \tag{3.8}$$

Podemos assim descrever o método RLS pelos seguintes passos:

1. Fazer $i = 1$ e inicializar:

- a matriz $\mathbf{G}_i$ como uma matriz de zeros;
- o vetor $\boldsymbol{\theta}_i$ como um vetor de zeros;

2. Repetir os passos 3 e 4 para $i = 2, \ldots, n$, onde n é o número de amostras;
3. Atualizar a matriz de covariâncias, $\mathbf{G}_i$, com a equação (3.6);
4. Atualizar o vetor de coeficientes do modelo, $\boldsymbol{\theta}_i$, com a equação (3.7).

3.3 *Recursive Least Squares - Adaptive*

O método RLS da Secção 3.2 não foi projetado para operar em ambientes não estacionários. Por outro lado, em sistemas variantes no tempo, o uso da abordagem iterativa com um fator de esquecimento [Jang *et al.*, 1997, Sec. 5.6, pp. 116-117], reduz a influência que os dados históricos têm na estimativa da saída, evitando a degradação da estimação à medida que o número de observações do modelo cresce.

Nesta dissertação denominamos o método RLS adaptativo apresentado em [Jang *et al.*, 1997, Sec. 5.6, pp. 116-117] por *Recursive Least Squares - Adaptive* (RLS-A). O método RLS-A contém as vantagens do RLS. Contudo, embora seja um dos métodos mais utilizados em estimação online recursiva dos parâmetros do modelo, este método mostra dificuldades na estimação quando utilizado em dados com elevada colinearidade, uma caraterística muito presente nas variáveis de processos industriais.

O método RLS-A faz a atualização incremental da matriz de covariâncias por:

$$\mathbf{G}_i = \frac{1}{\lambda}\left(\mathbf{G}_{i-1} - \frac{\mathbf{G}_{i-1}\mathbf{a}_i\mathbf{a}_i^T\mathbf{G}_{i-1}}{\lambda + \mathbf{a}_i^T\mathbf{G}_{i-1}\mathbf{a}_i}\right), \tag{3.9}$$

onde o fator de esquecimento λ, pode tomar valores na gama $0 < \lambda \leq 1$, sendo a importância das novas amostras inversamente proporcional ao valor de λ. Para $\lambda = 1$ a equação (3.9) é equivalente a (3.6).

Podemos assim descrever o método RLS-A pelos seguintes passos:

1. Fazer $i = 1$ e inicializar:
   - a matriz $\mathbf{G}_i$ como uma matriz de zeros;
   - o vetor $\boldsymbol{\theta}_i$ como um vetor de zeros;
2. Repetir os passos 3 e 4 para $i = 2, \ldots, n$, onde n é o número de amostras;
3. Atualizar a matriz de covariâncias, $\mathbf{G}_i$, com a equação (3.9);
4. Atualizar o vetor de coeficientes do modelo, $\boldsymbol{\theta}_i$, com a equação (3.7).

3.4 *Partial Least Squares*

O PLS é o método de regressão mais utilizado no desenvolvimento de SS. Este método deve a sua popularidade à sua capacidade para lidar com elevada colinearidade nos dados, elevada dimensionalidade do espaço de entrada e com ruído presente nos dados, sendo estas três caraterísticas comuns em processos industriais onde os SSs são aplicados.

Neste método é construído um conjunto de combinações lineares das variáveis de entrada. O PLS projeta a informação contida nos dados para um espaço de dimensão menor, caraterizado por um pequeno conjunto de l vetores ortogonais, denominados de variáveis latentes.

Segundo [Qin, 1998], pode-se decompor a matriz de entrada $\mathbf{X}$ e o vetor de saída $\mathbf{y}$ da seguinte forma:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T + \mathbf{E}, \tag{3.10}$$

$$\mathbf{y} = \mathbf{U}\mathbf{q}^T + \mathbf{f}, \tag{3.11}$$

onde $\mathbf{T} \in \mathbb{R}^{n\times l}$ e $\mathbf{U} \in \mathbb{R}^{n\times l}$ são chamadas matrizes de *score*, $\mathbf{P} \in \mathbb{R}^{m\times l}$ e $\mathbf{q} \in \mathbb{R}^l$ são denominados matriz e vetor de *loading* respetivamente, $\mathbf{E} \in \mathbb{R}^{n\times m}$ e $\mathbf{f} \in \mathbb{R}^n$ representam a matriz dos residuais da entrada e o vetor do residual da saída. O modelo externo do PLS é formado pelas equações (3.10) e (3.11) e o modelo interno do PLS é dado por:

$$\mathbf{U} = \mathbf{T}\mathbf{B} + \mathbf{R}, \tag{3.12}$$

onde $\mathbf{B} \in \mathbb{R}^{l\times l}$ é uma matriz diagonal com os coeficientes que minimizam o residual $\mathbf{R} \in \mathbb{R}^{n\times l}$.

Assumindo que os vetores $\mathbf{u}$, $\mathbf{t}$, $\mathbf{p}$, $\mathbf{w}$ e $\mathbf{b}$ correspondem às colunas d das matrizes $\mathbf{U}$, $\mathbf{T}$, $\mathbf{P}$, $\mathbf{W}$ e $\mathbf{B}$ respetivamente, onde d representa também o índice da iteração, pode apresentar-se, sumariamente, o algoritmo PLS descrito em [Qin, 1998] da seguinte forma:

1. Normalizar os dados da matriz $\mathbf{X}$ e do vetor $\mathbf{y}$ para uma variância unitária e média nula, e inicializar as matrizes $\mathbf{E}_0 := \mathbf{X}$, $\mathbf{f} := \mathbf{y}$, $d := 0$;

2. Fazer $d := d + 1$ e $\mathbf{u}_d := \mathbf{f}$;

3. Iterar o modelo externo até este convergir (ou seja até l):

$$\mathbf{w}_d = \mathbf{E}_{d-1}^T\mathbf{u}_d/\mathbf{u}_d^T\mathbf{u}_d, \tag{3.13}$$

$$\mathbf{t}_d = \mathbf{E}_{d-1}\mathbf{w}_d/ \parallel \mathbf{E}_{d-1}\mathbf{w}_d \parallel, \tag{3.14}$$

$$\mathbf{q} = \mathbf{f}^T\mathbf{t}_d/ \parallel \mathbf{f}^T\mathbf{t}_d \parallel, \tag{3.15}$$

$$\mathbf{u}_d = \mathbf{f}\mathbf{q}. \tag{3.16}$$

4. Encontrar a matriz de *loading* $\mathbf{P}$:

$$\mathbf{p}_d = \frac{\mathbf{E}_{d-1}^T\mathbf{t}_d}{\mathbf{t}_d^T\mathbf{t}_d} = \mathbf{E}_{d-1}^T\mathbf{t}_d. \tag{3.17}$$

5. Calcular o modelo interno:

$$\mathbf{b}_d = \frac{\mathbf{u}_d^T \mathbf{t}_d}{\mathbf{t}_d^T \mathbf{t}_d} = \mathbf{u}_d^T \mathbf{t}_d. \tag{3.18}$$

6. Atualizar os residuais:

$$\mathbf{E}_d = \mathbf{E}_{d-1} - \mathbf{t}_d \mathbf{p}_d^T. \tag{3.19}$$

$$\mathbf{f} = \mathbf{f} - \mathbf{b}_d \mathbf{t}_d \mathbf{q}^T. \tag{3.20}$$

7. Voltar ao passo 2 até que todos os fatores principais estejam calculados, ou seja, até à convergência do modelo externo.

Na Secção 5.1 é explicado como normalizar $\mathbf{X}$ e $\mathbf{y}$ para média nula e variância unitária.

3.5 *Recursive Partial Least Squares*

O algoritmo PLS não foi projetado para funcionar em ambientes não estacionários. No entanto a versão recursiva do PLS denominada de RPLS, para além de conseguir lidar com elevada colinearidade, também consegue determinar a saída do sistema quando uma nova amostra $\{\mathbf{x}_i, y_i\}_{i=1}^n$ está disponível. Isto é conseguido atualizando o modelo antigo, representado pelas matrizes $\mathbf{P}$, $\mathbf{B}$ e vetor $\mathbf{q}$, com os dados da nova amostra tal que:

$$\mathbf{X}_{new} = \begin{bmatrix} \lambda \mathbf{P}^T \\ \mathbf{x}_i \end{bmatrix}, \quad \mathbf{y}_{new} = \begin{bmatrix} \lambda \mathbf{B}\mathbf{q}^T \\ y_i \end{bmatrix}, \tag{3.21}$$

onde $\mathbf{X}_{new} \in \mathbb{R}^{n \times m}$, $\mathbf{y}_{new} \in \mathbb{R}^n$ e λ é o fator de esquecimento e funciona de forma análoga no RLS-A.

Pode descrever-se o algoritmo RPLS, apresentado em [Qin, 1998], nos seguintes passos:

1. Normalizar $\mathbf{X}$ e $\mathbf{y}$, para uma variância unitária e média nula;

2. Encontrar as matrizes $\mathbf{T}, \mathbf{W}, \mathbf{P}, \mathbf{B}, \mathbf{Q}$ a partir de $\mathbf{X}$ e $\mathbf{y}$, como no método PLS descrito na Secção 3.4;

3. Quando está disponível um novo par de dados $(\mathbf{x}_i, y_i)$, então pode-se recalcular $\mathbf{X}$ e $\mathbf{y}$ da seguinte forma: $\mathbf{X} = \begin{bmatrix} \mathbf{P}^T \\ \mathbf{x}_i \end{bmatrix}$ e $\mathbf{y} = \begin{bmatrix} \mathbf{B}\mathbf{q}^T \\ y_i \end{bmatrix}$; volta para o passo 2.

É de notar que a atualização do passo 3 ou da equação (3.21) está restrito ao caso em que o número de variáveis latentes, l, selecionado é igual ao *rank* da matriz de entrada $\mathbf{X}$. Na Secção 5.1 é explicado como normalizar $\mathbf{X}$ e $\mathbf{y}$ para média nula e variância unitária.

3.6 *Least Absolute Shrinkage and Selection Operator*

Em [Tibshirani, 1996] foi proposto o estimador denominado *Least Absolute Shrinkage and Selection Operator* (LASSO). A versão *offline* do LASSO pode ser descrita como a minimização de uma função com uma restrição, tendo sido inicialmente proposto como uma versão do LS com uma penalização. O método LS, descrito na Secção 3.1, é definido como o valor de $\boldsymbol{\theta}$ que minimiza a SSE, contudo este processo de minimização é instável quando duas ou mais variáveis de entrada são altamente correlacionadas, originando modelos pouco realistas. O papel da penalização é manter a estabilidade da minimização e ao mesmo tempo manter a convexidade da função objetivo.

Segundo [Anagnostopoulos *et al.*, 2008], dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$, pode definir-se o problema do LASSO por:

$$\hat{\boldsymbol{\beta}}^{L1} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} (y_i - \mathbf{x}_i\boldsymbol{\beta})^2 + \gamma \sum_{j=1}^{m} \mid \beta_j \mid \right\}, \tag{3.22}$$

onde, $\hat{\boldsymbol{\beta}}^{L1} \in \mathbb{R}^m$ é o vetor de parâmetros do modelo do LASSO, $\boldsymbol{\beta} \in \mathbb{R}^m$ é o vetor de parâmetros desconhecidos, β_j é o *j-ésimo* coeficiente do vetor $\boldsymbol{\beta}$ e γ é o parâmetro de regulação, sendo que o LASSO coloca coeficientes do modelo a zero à medida que o parâmetro γ aumenta. Devido a colocar os coeficientes de $\hat{\boldsymbol{\beta}}^{L1}$ a zero com o aumento de γ, o LASSO é conhecido por encorajar soluções esparsas e é também desta forma que o LASSO efetua a seleção de variáveis, uma vez que quanto mais coeficientes de $\hat{\boldsymbol{\beta}}^{L1}$ forem colocados a zero menor o número de variáveis selecionadas. O segundo termo da equação (3.22) é denominado de penalização l_1.

Em [Anagnostopoulos *et al.*, 2008] é usado um método denominado por *Shooting Algorithm* para chegar à solução da equação (3.22). Considerando que S_j é o gradiente do SSE penalizado em relação a β_j^{L1} (onde $\hat{\beta}_j^{L1}$ é o *j-ésimo* coeficiente de $\hat{\boldsymbol{\beta}}^{L1}$), é dado por:

$$S_j = -0.5 \sum_{i=1}^{n} x_{(i,j)} \left(y_i - \mathbf{x}_i \hat{\boldsymbol{\beta}}^{L1} \right), \tag{3.23}$$

temos que o coeficiente $\hat{\beta}_j^{L1}$ é atualizado da seguinte forma:

$$\hat{\beta}_j^{L1} \leftarrow \begin{cases} \text{sign}\left(\hat{\beta}_j^{L1} - S_j\right) \left(\mid \hat{\beta}_j^{L1} - S_j \mid -\gamma \right), & \text{Se } \mid \hat{\beta}_j^{L1} - S_j \mid \geq \gamma, \\ 0, & \text{outros casos,} \end{cases} \tag{3.24}$$

enquanto todos os outros coeficientes de $\hat{\boldsymbol{\beta}}^{L1}$ permanecem fixos.

Pode descrever-se o *Shooting Algorithm* nos seguintes passos:

1. Inicializar $\hat{\boldsymbol{\beta}}^{L1} = \boldsymbol{\theta}$;
2. Calcular S_j com a equação (3.23);
3. Atualizar $\hat{\beta}_j^{L1}$ com a equação (3.24);

4. Repetir os passos 2 e 3 para $j = 1, \ldots, m$, onde m é o número de variáveis;

5. Repetir os passos 1, 2, 3 e 4 até convergir.

3.7 Estimadores *LASSO* Adaptativos

O estimador do LASSO descrito na Secção 3.6 não foi projetado para funcionar em ambientes não estacionários mas, segundo [Anagnostopoulos *et al.*, 2008], é possível implementar versões adaptativas do Lasso, combinando o método do LASSO da Secção 3.6 com uma versão adaptativa do RLS.

Uma versão adaptativa do LASSO é o método *Recursive Least Absolute Shrinkage and Selection Operator* (RLASSO), que resulta da combinação do método RLS-A da Secção 3.3 com o método LASSO da Secção 3.6. O método RLASSO pode ser descrito de uma forma resumida pelos seguintes passos:

1. Calcular o estimador RLS-A;

2. Calcular o estimador LASSO usando o *Shooting Algorithm* com base na matriz $\mathbf{G}_i$ obtida no passo 1;

3. Repetir os passos 1 e 2 para $i = 1, \ldots, n$, onde n é o número de amostras.

Como descrito na Secção 3.3, os métodos RLS adaptativos devem a sua capacidade para operar em ambientes não estacionários ao fator de esquecimento, λ. No caso do RLS-A da Secção 3.3 o valor de λ é fixo, mas no caso do RLS-AF [Haykin, 1996, Sec. 16.10, pp. 734-735] o valor de λ varia em relação ao seu valor inicial durante a execução do algoritmo. O método RLS-AF atualiza o valor de λ através da expressão:

$$\lambda_i = \lambda_{i-1} + c_\lambda \operatorname{sign}\left(\frac{\partial SSE}{\partial \lambda_{i-1}}\right), \tag{3.25}$$

onde c_λ é uma pequena constante que representa o incremento usado para variar λ.

Outra versão adaptativa do LASSO é o método *Recursive Least Absolute Shrinkage and Selection Operator with Adaptive Forgetting* (RLASSO-AF) [Anagnostopoulos *et al.*, 2008]. Este método resulta da combinação do RLS-AF com o método LASSO e pode ser descrito de uma forma resumida pelos seguintes passos:

1. Calcular o estimador RLS-AF;

2. Calcular o estimador LASSO usando o *Shooting Algorithm* com base na matriz $\mathbf{G}_i$ obtida no passo 1;

3. Atualizar o valor do fator de esquecimento, λ, com a equação (3.25);

4. Repetir os passos 1, 2 e 3 para $i = 1, \ldots, n$, onde n é o número de amostras.

Ainda em [Anagnostopoulos *et al.*, 2008] é feita uma comparação entre o RLASSO-AF e o RLS-AF, tendo sido o RLASSO-AF a apresentar melhor desempenho.

3.8 *Incremental Forward Stagewise Regression*

O IFSR é um método de regressão linear que gera um vetor de coeficientes $\boldsymbol{\beta}^{ifsr} \in \mathbb{R}^m$ e a cada iteração atualiza apenas o coeficiente β_j^{ifsr} (j-*ésimo* coeficiente de $\boldsymbol{\beta}^{ifsr}$) correspondente à variável mais correlacionada com o vetor de resíduos $\mathbf{r} \in \mathbb{R}^n$. A atualização é feita de forma incremental por uma quantidade s, a qual se espera que assuma valores pequenos. No caso particular em que s toma valores muito próximos de zero, a solução do IFSR é idêntica à solução do LASSO.

Como apenas atualiza um coeficiente de cada vez, o IFSR é considerado um método de convergência lenta e por isso poderia ser considerado ineficiente, contudo ele tem uma boa capacidade para lidar com elevadas dimensionalidades no espaço de entrada.

Dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$, pode apresentar-se o IFSR, proposto em [Hastie *et al.*, 2009], pelos seguintes passos:

1. Inicializar: $\mathbf{r} = \mathbf{y}$; $\boldsymbol{\beta}^{ifsr} = \mathbf{0}$;
2. Encontrar a variável $\mathbf{x}_j$ mais correlacionada com $\mathbf{r}$;
3. Atualizar: $\beta_j^{ifsr} \leftarrow \beta_j^{ifsr} + \delta_j$, onde $\delta_j = s \cdot \text{sign}(< \mathbf{x}_j, \mathbf{r} >)$. Calcular o residual removendo a contribuição de $\delta_j \mathbf{x}_j$: $\mathbf{r} \leftarrow \mathbf{r} - \delta_j \mathbf{x}_j$;
4. Repetir os passos 2 e 3 até que qualquer das variáveis de entrada tenha baixo valor de correlação com o resíduo.

No passo 3, a notação $< \mathbf{x}_j, \mathbf{r} >$ significa calcular os coeficientes do modelo *Least Squares*, neste caso em particular o j-*ésimo* coeficiente do modelo usando o residual, $\mathbf{r}$, como saída. Ainda no passo 3 a função sign$(\cdot)$ devolve o sinal de um número real.

3.9 *Incremental Forward Stagewise Regression - Online*

Nesta secção é proposto o novo método IFSR-ON. Este método teve como motivação a incapacidade do método IFSR, da Secção 3.8, operar em ambientes não estacionários e o conhecimento de que o método RLS adaptativo, da Secção 3.3, quando combinado com outros métodos, pode torná-los adaptativos. Outra motivação foi o facto do algoritmo IFSR ter a capacidade de lidar com elevadas dimensionalidades do espaço de entrada. Elevadas dimensionalidades do espaço de entrada são uma caraterística comum nos processos industriais.

O método IFSR-ON funciona de forma análoga ao método IFSR. No caso do IFSR-ON, para cada amostra i, é atualizado o coeficiente β_j^{ifsr} correspondente à variável j presente em $\mathbf{x}_i$ mais correlacionada com o erro residual r_i, onde r_i é o elemento i do vetor de resíduos $\mathbf{r}$, ou seja, r_i é o residual da i-*ésima* amostra. Para encontrar a variável em $\mathbf{x}_i$ mais correlacionada com r_i é utilizado o método RLS-A da Secção 3.3.

Sendo $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$ o *data set* de observações do modelo, podemos descrever o IFSR-ON nas seguintes etapas:

1. Se $i = 1$:
   - Igualar o residual r_i à saída y_i;
   - Inicializar $\boldsymbol{\beta}^{ifsr} = \mathbf{0}$;
2. Encontrar a variável em $\mathbf{x}_i$ mais correlacionada com r_i:
   - Atualizar os parâmetros do modelo RLS-A, $\boldsymbol{\theta}$, com a base na entrada $\mathbf{x}_i$, utilizando o valor do residual r_i como saída;
   - Encontrar a variável de entrada em $\mathbf{x}_i$ à qual corresponde o parâmetro de $\boldsymbol{\theta}$ com maior valor, sendo o parâmetro representado por θ_j;
3. Atualizar: $\beta_j^{ifsr} \leftarrow \beta_j^{ifsr} + \delta_j$, onde $\delta_j = s \cdot \text{sign}(\theta_j)$;
4. Atualizar o residual removendo a contribuição de $x_{i,j}\beta_j^{ifsr}$: $r_{i+1} = r_i - x_{i,j}\beta_j^{ifsr}$;
5. Repetir os passos 1, 2, 3 e 4 para cada amostra i.

Capítulo 4

Métodos *Ensemble*

4.1 Introdução

Métodos *Ensemble* (ME) baseiam-se na combinação de um conjunto de modelos de modo a produzir uma predição robusta. A motivação para os ME é que, geralmente, uma combinação de modelos tem melhor desempenho do que qualquer um dos modelos do conjunto considerados individualmente [Soares *et al.*, 2011].

Existem diversas abordagens para combinar vários modelos. A escolha ótima da abordagem a seguir pode melhorar a robustez e a precisão dos ME. Duas das abordagens mais comuns para a agregação de modelos são:

1. Média das previsões - a média das saídas de todos os modelos dá-nos a saída final do Método *Ensemble*:

$$f = \frac{1}{p} \sum_{b=1}^{p} f_{(b)}, \tag{4.1}$$

   onde f é a predição final do *ensemble*, $f_{(b)}$ é a predição do modelo b, e p é o número de modelos;

2. Combinação pesada - a soma ponderada das saídas de todos os modelos, onde os pesos são atribuídos de acordo com o desempenho do modelo nos dados de treino:

$$f = \sum_{b=1}^{p} w_b f_{(b)}, \tag{4.2}$$

   onde w_b é o peso do modelo b.

Um elemento chave para o sucesso dos ME é a diversidade, ou seja, para além de ser desejável que os modelos individuais apresentem o menor erro possível, é também desejável que estes tenham diversidade para que o conjunto dos modelos seja mais robusto.

Dois dos métodos mais populares para a criação de ME são o *bagging* e o *boosting*. O *bagging* consiste em criar para cada modelo do *ensemble* um conjunto de dados de treino diferente, obtido a partir do conjunto de treino original, sendo para tal utilizada uma técnica

denominada de *bootstrap*. Esta técnica faz uma amostragem com reposição dos dados de treino originais, criando novos conjuntos do mesmo tamanho. Contudo, nesta dissertação apenas serão utilizados métodos *boosting*.

A técnica de *boosting* foi inicialmente projetada para problemas de classificação, vendo o seu campo de aplicação ser estendido com sucesso para problemas de regressão, sendo considerada uma das mais importantes metodologias de aprendizagem das últimas duas décadas. A ideia base do *boosting* é treinar vários *weak learners* e combiná-los de forma a criar um modelo mais poderoso, com melhor desempenho. O método de *boosting* mais utilizado é denominado de *Adaboost* e é descrito em [Hastie *et al.*, 2009].

As principais motivações do *boosting* são:

- O modelo originado pela combinação dos *weak learners* tem geralmente melhor resposta do que qualquer um dos modelos fracos individualmente;
- Minimiza o efeito da instabilidade dos algoritmos de aprendizagem;
- Consegue lidar com conjuntos de dados em que o número de variáveis é bastante maior do que o número de amostras.

Contudo têm como desvantagens:

- Custa mais a construir, ou seja, como é necessário efetuar a aprendizagem de vários modelos torna-se computacionalmente mais exigente;
- Embora costume ter melhores resultados não há garantias que isso aconteça sempre, pois se os modelos fracos não forem diversos, estes não acrecentam nenhum contributo positivo para a desempenho do ME.

Este capítulo apresenta entre outros métodos *boosting*, dois dos métodos propostos nesta dissertação para o desenvolvimento de SS. O primeiro é um método não adaptativo intitulado de LSBST-SV e o segundo um método adaptativo chamado de LSBST-SV-ON.

Neste capítulo é ainda realizada uma análise comparativa entre o método LSBST implementado nesta dissertação e o método proposto LSBST-SV, para que se perceba de uma forma mais clara as vantagens de cada um dos algoritmos e o porquê destas terem motivado o desenvolvimento dos novos métodos de *boosting*.

Antes de se passar à apresentação dos métodos *boosting* implementados nesta dissertação, é realizada uma pequena descrição do método *Gradient Boosting* (GB) proposto em [Friedman, 2001], sendo que este método será usado como base dos algoritmos *boosting* comparados nesta tese.

4.2 *Gradient Boosting*

Em estimação de funções ou em aprendizagem preditiva, o objetivo é, dado um *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$, aproximar a função desconhecida $\hat{F}(\mathbf{X})$ capaz de prever $\mathbf{y}$ a partir de $\mathbf{X}$ ao longo

da distribuição do conjunto de valores de $\{\mathbf{X}, \mathbf{y}\}$ e de forma a minimizar uma *loss function* $L(\mathbf{y}, F)$ específica.

A solução para este problema, proposta em [Friedman, 2001], faz uma ligação entre a regressão *greedy stagewise* e a minimização *steppest descent* e é denominado de *Gradient Boosting* (GB). O GB é considerado uma técnica de *gradiet descent (steepest descent)*, pois minimiza a função de perda dando passos na direção do gradiente negativo e é considerado uma técnica *greedy stagewise* pois acumula a solução em pequenas etapas e cada etapa é baseada nas anteriores, sendo que a estimativa do modelo final, pode ser vista como a soma das estimativas anteriores. A ideia é restringir $F(\mathbf{X})$ a um membro de uma classe parametrizada de funções $F(\mathbf{X}, \mathbf{Z})$, onde $\mathbf{Z}$ é um conjunto finito de parâmetros ($\mathbf{Z} = \{\rho_b, \mathbf{a}_b^{bst}\}$). Assumindo uma expansão aditiva para o vetor de estimativas $F(\mathbf{X}) \in \mathbb{R}^n$ da forma:

$$F\left(\mathbf{X}; \{\rho_b, \mathbf{a}_b^{bst}\}_{b=1}^{p}\right) = \sum_{b=1}^{p} \rho_b h\left(\mathbf{X}; \mathbf{a}_b^{bst}\right), \tag{4.3}$$

onde ρ_b é o tamanho do passo do *gradient descent* para o modelo b e $\mathbf{a}_b^{bst} \in \mathbb{R}^m$ é o vetor de coeficientes do modelo do *boosting*. Para estimar $F(\mathbf{X})$, temos que resolver o seguinte problema de otimização:

$$\{\rho_b; \mathbf{a}_b^{bst}\} = \arg \min_{\left(\mathbf{a}^{bst}, \rho\right)} \sum_{i=1}^{n} L\left(y_i, F_{b-1}(\mathbf{x}_i) + \rho h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)\right). \tag{4.4}$$

A *b-ésima* atualização de $F(\mathbf{X})$, é feita de uma forma *greedy stagewise* e é dada por:

$$F_b(\mathbf{X}) = F_{b-1}(\mathbf{X}) + \rho_b h\left(\mathbf{X}; \mathbf{a}_b^{bst}\right), \tag{4.5}$$

onde $F_b(\mathbf{X}) \in \mathbb{R}^n$ é o vetor de estimativas com a contribuição do modelo b.

Ajustando $h\left(\mathbf{X}, \mathbf{a}^{bst}\right)$ em relação aos valores do gradiente negativo da *i-ésima* amostra, g_i, usando os mínimos quadrados, simplificamos a equação (4.4) separando-a em duas, na equação (4.6) para ajustar o modelo e na equação (4.8) para calcular o tamanho do passo do *gradient descent*:

$$\mathbf{a}_b^{bst} = \arg \min_{\mathbf{a}^{bst}} \sum_{i=1}^{n} \left[g_i - \rho h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)\right]^2, \tag{4.6}$$

onde o valor de g_i é dado pelo gradiente negativo da *i-ésima* amostra:

$$g_i = -\left[\frac{\partial L\left(y_i, F(\mathbf{x}_i)\right)}{\partial F(\mathbf{x}_i)}\right]_{F_{m-1}(\mathbf{x}_i)} \qquad (i = 1, \ldots, n), \tag{4.7}$$

$$\rho_b = \arg \min_{\rho} \sum_{i=1}^{n} L\left(y_i, F_{b-1}(\mathbf{x}_i) + \rho h\left(\mathbf{x}_i; \mathbf{a}_b^{bst}\right)\right). \tag{4.8}$$

Podemos assim descrever o método GB, proposto em [Friedman, 2001], nos seguintes passos:

1. Inicializar: $F_0(\mathbf{X}) = \arg \min_\rho \sum_{i=1}^{n} L(y_i, \rho)$;

2. Calcular o gradiente negativo usando a equação (4.7), com i a variar de $(i = 1, \ldots, n)$;

3. Ajustar o modelo usando a equação (4.6);

4. Calcular o tamanho do passo do *gradient descent* usando a equação (4.8);

5. Atualizar o vetor de estimativas usando a equação (4.5);

6. Repetir os passos 2, 3, 4 e 5 para $b = 1, \ldots, p$, onde p é o número de modelos.

4.3 *Least Squares Boosting*

O algoritmo *Gradient Boosting* proposto em [Friedman, 2001], descrito na Secção 4.2, foi desenvolvido de forma a permitir a sua aplicação com diferentes métodos de regressão. Para que isto seja possível, é necessário especificar a *loss function* $L\left(y_i, F\left(\mathbf{x}_i\right)\right)$, e a forma funcional, da função básica $h\left(\mathbf{x}_i, \mathbf{a}^{bst}\right)$ a utilizar, onde $F\left(\mathbf{x}_i\right)$ é o *i-ésimo* elemento do vetor de estimativas (ou seja a estimativa para a amostra i).

Least Squares Boosting é uma das aplicações do *Gradient Boosting* mais utilizadas e usa, como *loss function e função básica*, as seguintes equações respetivamente:

$$L\left(y_i, F\left(\mathbf{x}_i\right)\right) = \frac{\left(y_i - F\left(\mathbf{x}_i\right)\right)^2}{2}, \tag{4.9}$$

$$h\left(\mathbf{x}_i, \mathbf{a}_b^{bst}\right) = \mathbf{x}_i \mathbf{a}_b^{bst}. \tag{4.10}$$

Substituindo a equação (4.9) na equação (4.7), resulta o gradiente negativo:

$$g_i = y_i - F_{b-1}\left(\mathbf{x}_i\right), \tag{4.11}$$

onde g_i, para esta função de perda, corresponde ao residual atual (ou seja é o residual da amostra i) e $F_{b-1}\left(\mathbf{x}_i\right)$ foi obtido pela equação (4.5) na iteração anterior do algoritmo.

Dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$, o algoritmo LSBST é definido pelos seguintes passos:

1. Inicializar: $F_0\left(\mathbf{X}\right) = \overline{\mathbf{y}}$;

2. Calcular o gradiente negativo, g_i, usando a equação (4.11) para $i = 1, \ldots, n$;

3. Ajustar o modelo e calcular o tamanho do passo do *gradient descent*:

$$\left(\rho_b, \mathbf{a}_b^{bst}\right) = \arg \min_{\left(\mathbf{a}^{bst}, \rho\right)} \sum_{i=1}^{n} \left[g_i - \rho h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)\right]^2; \tag{4.12}$$

4. Atualizar o vetor de estimativas usando a equação (4.5);

5. Repetir os passos 2, 3 e 4 para $b = 1, \ldots, p$, onde p é o número de modelos.

No passo 1, o vetor $\overline{\mathbf{y}} \in \mathbb{R}^n$ é o produto de um vetor de uns, $\mathbf{1} \in \mathbb{R}^n$, com a média do vetor $\mathbf{y}$, ou seja $\overline{\mathbf{y}} = \mathbf{1}\overline{y}$.

O método LSBST, para além das vantagens já referidas dos métodos *boosting*, atenua o problema do RLS face a elevadas colinearidades nos dados de entrada.

4.4 *Least Squares Boosting - Online*

Em [Babenko *et al.*, 2009] é proposta uma *framework* que permite utilizar diversos algoritmos de *boosting* de forma *online*, nomeadamente o LSBST, dando origem ao LSBST-ON.

Sabendo que o objetivo do *boosting* é minimizar a equação (4.4), assumindo que o erro residual sobre o *data set* inteiro pode ser representado pela soma dos erros residuais em cada amostra, e assumindo que $h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)$ é diferenciável em ordem a $\mathbf{a}^{bst}$, pode ser utilizado o método *Stochastic Gradient Descent* diretamente para minimizar a função de perda: $L\left(y_i, F_{b-1}\left(\mathbf{x}_i\right) + h\left(\mathbf{x}_i; \mathbf{a}_b^{bst}\right)\right)$, onde $F_{b-1}\left(\right)$ representa o modelo forte construído a partir das primeiras $(b-1)$ funções básicas fracas. O método *Stochastic Gradient Descent* consiste em minimizar a função de perda através da soma de funções diferenciáveis.

A *framework*, proposta em [Babenko *et al.*, 2009], atualiza todos os modelos fracos para cada amostra do *data set* em que as amostras $\{\mathbf{x}_i; y_i\}_{i=1}^n$ estão disponíveis uma de cada vez ao longo do tempo, e pode ser descrita pelos seguintes passos:

1. Atualizar os parâmetros (regra):

$$\mathbf{a}_b^{bst} \leftarrow \mathbf{a}_b^{bst} - \eta_i \frac{\partial}{\partial \mathbf{a}^{bst}} L_i\left(y_i, F_{b-1}\left(\mathbf{x}_i\right) + h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)\right) \mid_{\mathbf{a}^{bst}=\mathbf{a}_b^{bst}} . \tag{4.13}$$

   O parâmetro η_i é uma constante de valor reduzido, que representa o fator de aprendizagem, e $F_{b-1}\left(\mathbf{x}_i\right)$ é o elemento i do vetor de estimativas sem a contribuição do modelo atual (para $b = 1$ temos que $F_{b-1}\left(\mathbf{x}_i\right) = \overline{y}$);

2. Repetir o passo 1 para $b = 1, \ldots, p$, onde p é o número de modelos;

3. Repetir os passos 1 e 2 para $i = 1, \ldots, n$, onde n é o número de amostras.

Para utilizar esta *framework* de modo a obter o LSBST-ON é necessário mudar a regra de atualização do passo 1 anteriormente descrita na Secção 4.3. No caso do LSBST-ON temos que a *loss function* é dada pela equação (4.9). Vem que a sua derivada em ordem a $\mathbf{a}^{bst}$ é dada por:

$$\frac{\partial}{\partial \mathbf{a}^{bst}} L_i\left(y_i, F_{b-1}\left(\mathbf{x}_i\right) + h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)\right) \mid_{\mathbf{a}^{bst}=\mathbf{a}_b^{bst}} =$$

$$\left(F_{b-1}\left(\mathbf{x}_i\right) + h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right) - y_i\right) \frac{\partial h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)}{\partial \mathbf{a}^{bst}} \mid_{\mathbf{a}^{bst}=\mathbf{a}_b^{bst}} . \tag{4.14}$$

Para se obter o LSBST-ON a regra de atualização é dada pela equação seguinte em substituição da equação (4.13):

$$\mathbf{a}_b^{bst} \leftarrow \mathbf{a}_b^{bst} - \eta_i \left(F_{b-1}\left(\mathbf{x}_i\right) + h\left(\mathbf{x}_i; \mathbf{a}_b^{bst}\right) - y_i\right) \frac{\partial h\left(\mathbf{x}_i; \mathbf{a}^{bst}\right)}{\partial \mathbf{a}^{bst}} \mid_{\mathbf{a}^{bst}=\mathbf{a}_b^{bst}} . \tag{4.15}$$

Por simplicidade considerou-se que o parâmetro ρ_b foi também representado por $\mathbf{a}_b^{bst}$ nas equações (4.13), (4.14) e (4.15).

Para cada amostra $\{\mathbf{x}_i, y_i\}_{i=1}^n$, a *b-ésima* atualização de $F\left(\mathbf{x}_i\right)$ é dada por:

$$F_b\left(\mathbf{x}_i\right) = F_{b-1}\left(\mathbf{x}_i\right) + \rho_b h\left(\mathbf{x}_i; \mathbf{a}_b^{bst}\right) . \tag{4.16}$$

4.5 *Least Squares Boosting Single Variable*

Nesta secção é proposto o novo método LSBST-SV. Sabendo que a ideia do *boosting* é combinar modelos fracos de forma a criar um modelo forte, onde chamamos de modelo fraco a um preditor que realiza mal a sua função, por vezes com uma precisão pouco superior à que se obteria se a predição fosse feita ao acaso, o método proposto assenta nos conceitos de *boosting* e de modelo fraco, pois o LSBST-SV modifica o método LSBST da Secção 4.3 de modo a que cada modelo fraco do LSBST-SV, em vez de ser construído com base em todas as variáveis de entrada, seja construído apenas com base numa só variável de entrada. Assim o método proposto constrói m vezes mais modelos do que o método LSBST, ou seja, a função básica é:

$$h\left(\mathbf{x}_j; a^{bst}_{(j,b)}\right) = \mathbf{x}_j a^{bst}_{(j,b)}, \tag{4.17}$$

onde $a^{bst}_{(j,b)}$ é o *b-ésimo* modelo da variável j.

Este algoritmo é denominado por LSBST-SV. A motivação deste método é o desenvolvimento de uma variante do LSBST em que os modelos fracos que o constituem sejam ainda mais diversos. E por isso neste método o modelo forte é atualizado incrementalmente mas desta vez com a contribuição de modelos individuais para cada variável.

Dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^n$, pode descrever-se o algoritmo LSBST-SV nos seguintes passos:

1. Inicializar: $F_0(\mathbf{X}) = \overline{\mathbf{y}}$, onde $\overline{\mathbf{y}}$ corresponde ao produto de um vetor de uns, $\mathbf{1} \in \mathbb{R}^n$, com a média do vetor $\mathbf{y}$;

2. Calcular o gradiente negativo, g_i, para $i = 1, \ldots, n$ usando a equação (4.11);

3. Ajustar o modelo e calcular o tamanho do passo do *gradient descent*:

$$\left(\rho_{(j,b)}, a^{bst}_{(j,b)}\right) = \arg\min_{\left(a^{bst},\rho\right)} \sum_{i=1}^{n} \left[g_i - \rho_j h\left(x_{(i,j)}; a^{bst}_{(j,b)}\right)\right]^2. \tag{4.18}$$

   O parâmetro $x_{(i,j)}$ representa a amostra i da variável de entrada j;

4. Atualizar o modelo:

$$F_b(\mathbf{X}) = F_{b-1}(\mathbf{X}) + \rho_{(j,b)} h\left(\mathbf{x}_j; a^{bst}_{(j,b)}\right); \tag{4.19}$$

5. Repetir os passos 2, 3 e 4 para $j = 1, \ldots, m$, onde m é o número de variáveis;

6. Repetir os passos 2, 3, 4 e 5 para $b = 1, \ldots, p$, onde p é o número de modelos;

4.6 *Least Squares Boosting Single Variable - Online*

Nesta secção é proposto o novo método *Least Squares Boosting Single Variable Online* (LSBST-SV-ON), com a motivação de construir um método *boosting* em que os modelos

fracos que o constituem sejam mais fracos do que os que constituem o LSBST, tal como para o método LSBST-SV e com a motivação de responder à incapacidade do método LSBST-SV, proposto na Secção 4.5, de operar em ambientes não estacionários. Esta característica de não estacionaridade é muito comum em processos industriais. O método LSBST-SV-ON efectua a aprendizagem e atualização do modelo de forma online na sequência da chegada de cada amostra. Com a mesma analogia utilizada para modificar o algoritmo LSBST de modo a desenvolver o LSBST-SV, foi modificado o LSBST-ON para desenvolver o método LSBST-SV-ON. Este algoritmo atualiza todos os modelos fracos para cada amostra, atualizando a contribuição de uma variável de entrada de cada vez. O método proposto, LSBST-SV-ON, tem como função básica:

$$h\left(x_{(i,j)}; a^{bst}_{(j,b)}\right) = x_{(i,j)} a^{bst}_{(j,b)}. \tag{4.20}$$

Dado o *data set* $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$, pode descrever-se o algoritmo LSBST-SV-ON nos seguintes passos:

1. Atualizar os parâmetros (regra):

$$a^{bst}_{(j,b)} \leftarrow a^{bst}_{(j,b)} - \eta_i \frac{\partial}{\partial a^{bst}_{j}} L_i\left(y_i, F_{b-1}\left(\mathbf{x}_i\right) + h\left(x_{(i,j)}; a^{bst}_{j}\right)\right) |_{a^{bst}_{j} = a^{bst}_{(j,b)}} . \tag{4.21}$$

   O parâmetro η_i representa o fator de aprendizagem e é uma pequena constante, $x_{(i,j)}$ é a *i-ésima* amostra da variável j e $F_{b-1}\left(\mathbf{x}_i\right)$ é o elemento i do vetor de estimativas sem a contribuição do modelo atual (para $b = 1$ inicializa-se $F_{b-1}\left(\mathbf{x}_i\right) = \overline{y}$);

2. Repetir o passo 1 para $j = 1, \ldots, m$, onde m é o número de variáveis;

3. Repetir o passo 1 e 2 para $b = 1, \ldots, p$, onde p é o número de modelos;

4. Repetir o passo 1, 2 e 3 para $i = 1, \ldots, n$, onde n é o número de amostras.

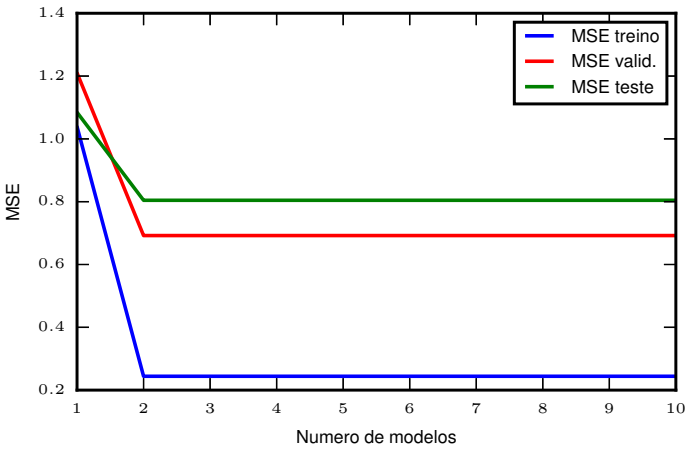
Por simplicidade considerou-se que o parâmetro ρ_b foi também ele representado por $\mathbf{a}^{bst}_{b}$ na equação (4.21).

Para cada amostra $\{\mathbf{x}_i, y_i\}_{i=1}^{n}$, a *b-ésima* atualização de $F\left(\mathbf{x}_i\right)$ é dada por:

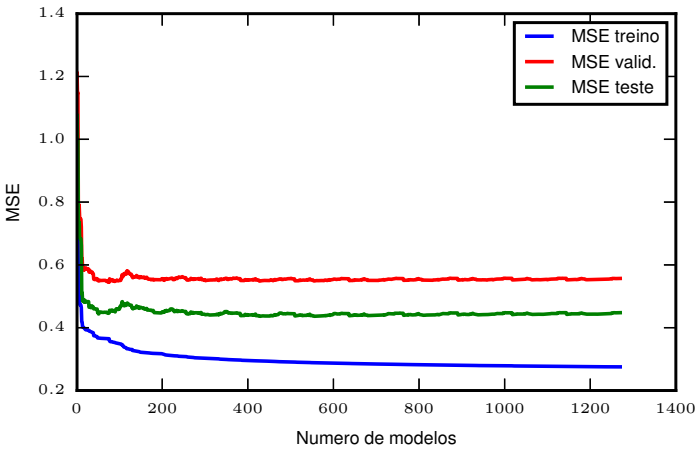
$$F_b\left(\mathbf{x}_i\right) = F_{b-1}\left(\mathbf{x}_i\right) + \rho_b h\left(x_{(i,j)}; a^{bst}_{(j,b)}\right). \tag{4.22}$$

4.7 Uma Análise ao *Boosting*

Como já foi descrito, a motivação para os dois métodos de *boosting* LSBST-SV e LSBST-SV-ON propostos nas Secções 4.5 e 4.6 foi desenvolver métodos *boosting* que construíssem o seu modelo forte com base em mais modelos e mais fracos do que os usados pelos métodos LSBST e LSBST-ON. Nesta secção é realizada uma comparação entre o método proposto LSBST-SV e o método LSBST, para melhor perceber o efeito que esta alteração tem na prática, a qual é aproveitada para abordar um problema bastante frequente no *boosting*, conhecido como *overfitting*.



(a) MSE LSBST



(b) MSE LSBST-SV

Figura 4.1: Evolução dos valores de MSE obtidos para os dados de treino, validação e teste, em função do número de modelos, com os algoritmos LSBST e LSBST-SV.

O *overfitting* é um problema chave em *machine learning* e por isso merece alguma atenção. De uma forma genérica pode dizer-se que ocorre *overfitting* quando um preditor apresenta bons resultados com os dados de treino mas não com os dados de teste, ou seja, tem um erro reduzido quando utilizado nos dados de treino e um erro elevado quando utilizado nos dados de teste.

Segundo [Bishop *et al.*, 2006], o *overfitting* ocorre quando o modelo captura ruídos presentes nos dados de treino. Quanto maior for a complexidade do modelo ou a flexibilidade que este possui para se ajustar aos dados de treino, maior é o risco de *overfitting*. Por sua vez, o risco de *overfitting* diminui à medida que o número de amostras utilizadas para o treino aumenta.

Na Figura 4.1 é possível verificar a ocorrência de *overfitting*, sendo mais acentuada para o algoritmo LSBST, verificando-se também que o *overfitting* acentua-se com o aumento do número de modelos. O motivo pelo qual o método LSBST é mais suscetível à ocorrência

de *overfitting* deve-se ao facto de os modelos serem mais complexos, pois cada um deles é construído com base em todas as variáveis de entrada. É evidente também, pela observação da figura, que o número de modelos criados pelo método proposto é m vezes superior ao número de modelos do LSBST, tornando-os menos complexos.

Pode concluir-se que o método *boosting* proposto, LSBST-SV, embora não tenha sido desenvolvido para suprimir o *overfitting* nos algoritmos *boosting*, oferece uma maior resistência à sua ocorrência quando comparado com o método LSBST.

Pode ainda verificar-se na Figura 4.1 que o método LSBST-SV utiliza m vezes mais modelos, mas modelos mais simples, do que o LSBST e que o aumento do número de modelos se traduz num decréscimo do valor do MSE.

Capítulo 5

Resultados e Discussão

5.1 Introdução

Nesta dissertação foram efetuados dois testes distintos para validar e avaliar o desempenho dos métodos abordados. No primeiro teste, foram aplicados os algoritmos *offline* aos seguintes treze *data sets* de referência disponíveis em repositórios públicos (exceto o *data set* (v)) e referentes a problemas do mundo real: (i) Automobile MPG [Frank and Asuncion, 2010], (ii) Bank [Rasmussen, 1996], (iii) Boston Housing [Frank and Asuncion, 2010], (iv) Box and Jenkins [Reinsel, 1999], (v) Cagece [Souza and Araújo, 2014], (vi) Communities and Crime [Frank and Asuncion, 2010], (vii) Concrete Compressive Strength [Frank and Asuncion, 2010], (viii) CPU [Frank and Asuncion, 2010], (ix) Pumadyn [Corke, 1996], (x) Stock [Torgo, 1991], (xi) Pyrimidines [King *et al.*, 1992], (xii) Spectra [Kalivas, 1997] e (xiii) Triazines [Torgo, 1991]. No segundo teste, os algoritmos adaptativos foram aplicados a dois *data sets* de processos do mundo real variantes no tempo: o (xiv) *data set* da Ativação do Catalisador [Kadlec and Gabrys, 2011] e o (xv) *data set* para a Estimação da Quantidade de Flúor no efluente de uma ETAR [Souza and Araújo, 2014]. Os *data sets* (v) Cagece e (xv) da Estimação da Quantidade de Flúor, retratam o mesmo problema do mundo real, mas com a diferença que o *data set* (v) Cagece diz respeito a um período temporal mais reduzido, podendo ser tratado como um ambiente estacionário. O *data set* (xv) representa um período de três anos pelo que já se trata de um ambiente não estacionário.

Na metodologia utilizada para os algoritmos não adaptativos, os *data sets* foram divididos de forma aleatória nas seguintes proporções: 50% para treino, 20% para validação e 30% para teste.

Uma metodologia diferente foi utilizada para os algoritmos adaptativos [Souza and Araújo, 2014]. Os dois *time-varying data sets* foram divididos em 30% para dados de treino e os restantes 70%, são considerados dados de teste, mas foram utilizados para simulação online e são recebidos num fluxo de amostras. Para cada amostra recebida online é efetuada a predição com o modelo e a aprendizagem/atualização do modelo. Com os *data sets* divididos nestas proporções conseguimos testar o desempenho dos algoritmos em funcionamento online

e a adaptabilidade dos métodos implementados. Para uma melhor compreensão da influência da capacidade adaptativa dos algoritmos desenvolvidos, foram ainda testados todos os algoritmos para diferentes valores de λ ($\lambda = 0.50$, $\lambda = 0.80$, $\lambda = 0.95$, $\lambda = 0.98$, $\lambda = 0.99$ e $\lambda = 1$). Para averiguar o desempenho do SS relativamente à disponibilidade dos dados de saída, foram testados os algoritmos para diferentes disponibilidades ($D = 0\%$, $D = 10\%$, $D = 25\%$, $D = 50\%$ e $D = 100\%$), cobrindo assim o cenário não adaptativo e o caso em que todos os dados da saída real estão disponíveis. Nesta metodologia quando uma nova amostra de entrada-saída, $(\mathbf{x}_i; y_i)$, está disponível, é estimada a saída $\hat{y}_i$ e só depois são atualizados os parâmetros do modelo. Se só existir disponível a amostra da entrada, apenas é estimada a saída $\hat{y}_i$.

Todos os *data sets* utilizados para testar os algoritmos usados nesta dissertação foram normalizados. Os valores das amostras de treino, de validação e de teste foram normalizados, para média nula e variância unitária, utilizando a informação da média e da variância das amostras de treino. Seja $\mathbf{v} \in \mathbb{R}^n$ um vetor que representa uma variável contida em $\mathbf{X}$ ou em $\mathbf{y}$ pode ser normalizado, para média nula e variância unitária, através da seguinte expressão:

$$\mathbf{v}^{norm} = \frac{\mathbf{v} - \overline{\mathbf{v}}}{\sigma}, \tag{5.1}$$

onde $\overline{\mathbf{v}} \in \mathbb{R}^n$ representa o produto do vetor $\mathbf{1} \in \mathbb{R}^n$ pela média do vetor $\mathbf{v}$, σ representa o desvio padrão e $\mathbf{v}^{norm} \in \mathbb{R}^n$ representa o vetor $\mathbf{v}$ normalizado.

Todos os códigos foram implementados de forma autónoma (ou seja implementados de raiz por mim, durante o decorrer do trabalho de dissertação, sem recurso a nenhuma toolbox), com recurso à ferramenta *Matlab*, com as exceções dos algoritmos: PLS, LASSO e RPLS. O algoritmo do LASSO foi implementado com recurso à *Toolbox glmnet* disponível em [Qian *et al.*, 2013] e os algoritmos PLS e RPLS foram implementados com recurso à *Toolbox MVARTOOLS* disponível em [Mathisen, 2001].

5.2 Definição de Métricas

As métricas utilizadas para avaliar o desempenho dos algoritmos testados foram: o *Mean Square Error* (MSE) e o *Normalized Root Mean Square Error* (NRMSE). Elas são descritas pelas seguintes equações, respetivamente:

$$MSE = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \left(\hat{y}_i - y_i\right), \tag{5.2}$$

$$NRMSE = \frac{\sqrt{MSE}}{\max\left(\mathbf{y}\right) - \min\left(\mathbf{y}\right)}, \tag{5.3}$$

onde, n_{test} é o número de amostras do conjunto de teste, $\hat{y}_i$ é a estimação da saída real para a amostra i, $\mathbf{y} \in \mathbb{R}^n$ é o vetor de saída real e y_i é a amostra i de $\mathbf{y}$.

Tanto para o MSE como para o NRMSE, quanto menor for o valor obtido, melhor é a qualidade da estimativa da saída. No caso do NRMSE o valor é apresentado em percentagem.

São considerados aceitáveis valores de NRMSE menores do que 10% [Souza and Araújo, 2014].

5.3 Regulação dos Parâmetros de Configuração dos Algoritmos

A maioria dos algoritmos implementados necessita que os seus parâmetros de configuração sejam calibrados de forma adequada, para cada *data set* em particular, de modo a obter os melhores desempenhos.

Para se selecionar o valor com o qual se deve inicializar os parâmetros de um algoritmo de modo a que este obtenha melhor desempenho para um *data set* específico, são treinados os algoritmos com os dados de treino e em seguida é testado com os dados de validação o modelo obtido durante o treino. No treino são utilizados diferentes valores para o parâmetro a calibrar, pertencentes a uma gama inicial predefinida, sendo escolhido como valor para o parâmetro aquele que obtiver um menor valor de MSE no teste com os dados de validação. A gama foi definida manualmente de forma a tentar garantir que o valor ideal estivesse compreendido no intervalo de valores escolhido. É de notar que os valores ótimos de cada parâmetro, embora seja provável que estejam incluídos na gama escolhida, poderão não coincidir exatamente com os valores utilizados.

No caso dos *data sets* de processos variantes no tempo, como os dados apenas foram divididos em dados de treino e dados de teste, então, apenas para efeitos desta abordagem para obter os parâmetros iniciais do modelo, os dados de treino foram subdivididos num novo sub-conjunto de dados de treino e num sub-conjunto de dados de validação.

Na Tabela 5.1 são apresentadas as gamas de valores que foram testadas para cada parâmetro de cada algoritmo. Foram testados dez a quinze valores igualmente espaçados, pertencentes às gamas de valores da Tabela 5.1, para cada parâmetro.

5.4 Resultados

Nesta secção é feita uma pequena descrição dos *data sets* referidos inicialmente, para que se possa compreender e analisar de forma mais crítica os resultados obtidos. São apresentados os resultados dos algoritmos implementados, de forma a que se possam comparar os indicadores de desempenho referidos anteriormente e avaliar a robustez dos algoritmos estudados.

5.4.1 *Data Sets* de Referência

Uma breve descrição dos *data sets* de referência é apresentada na Tabela 5.2, nomeadamente a nomenclatura pela qual serão referidos, o seu número de amostras, n, e número de variáveis de entrada, m. Mesmo que uma escolha otimizada dos atrasos entre as variáveis de entrada

Tabela 5.1: Gamas de valores testados para os parâmetros de configuração dos métodos.

Algoritmo	Parâmetro 1	Parâmetro 2
RLS	–	–
PLS	$l \in [1; \text{rank}\,(X_{val})]$	–
LASSO	–	–
IFSR	$s \in [0.01; 0.3]$	$z \in [1000; 40000]$
LSBST	$r \in [3; 100]$	–
LSBST-SV	$r \in [1; 100]$	–
RLS-A	–	–
RPLS	$l \in [1; \text{rank}\,(X_{val})]$	–
IFSR-ON	$s \in [0.005; 0.3]$	–
RLASSO	$c_\lambda \in [0.001; 0.2]$	–
LSBST-ON	$r \in [1; 20]$	$\eta \in [0.0000005; 0.5]$
LSBST-SV-ON	$r \in [1; 10]$	$\eta \in [0.00000005; 0.5]$
RLASSO-AF	$c_\lambda \in [0.001; 0.2]$	$c_\gamma \in [0.001; 0.2]$

e de saída no modelo possam influenciar positivamente o desempenho dos métodos, como o objetivo deste trabalho era analisar apenas a capacidade de aprendizagem dos métodos implementados (excluindo o problema da seleção de variáveis e respetivos atrasos), não foram considerados possíveis atrasos entre a entrada e a saída.

Depois de escolhidos os parâmetros, dentro das gamas indicadas a Tabela 5.1, que se traduziam num melhor desempenho de cada algoritmo para cada *data set*, os algoritmos foram aplicados aos *data sets* descritos na Tabela 5.2.

As Tabelas 5.3 e 5.4 apresentam o valor do NRMSE obtidos na aplicação dos métodos não adaptativos aos mesmos treze *data sets*, para os dados de validação e de teste, respectivamente.

Analisando as Tabelas 5.3 e 5.4, podemos verificar que os algoritmos em geral apresentam um desempenho razoável e boa consistência nos resultados para os dados de validação e de teste. Sendo exceção, o algoritmo LSBST-ON que demonstrou uma grande inconsistência entre os valores de NRMSE obtidos com os dados de validação e com os de teste. Pode observar-se nas Tabelas 5.3 e 5.4, que os melhores desempenhos não pertenceram de forma unânime a um só algoritmo, mas foram maioritariamente obtidas pelos métodos PLS e LASSO. Verifica-se que o desempenho dos métodos propostos não diverge muito do obtido pelos métodos existentes e verifica-se ainda que o desempenho dos métodos *boosting online* não diverge muito do obtido pelos restantes métodos *offline* com os quais foram comparados.

5.4.2 *Data Set* da Ativação do Catalisador

O *data set* da Ativação de Catalisador é um *benchmark* para SS adaptativos introduzida em [Kadlec and Gabrys, 2011]. Este *data set* descreve um reator de polimerização, aonde o objetivo é estimar a atividade do Catalisador nos tubos. Na Tabela 5.5 são apresentadas

Tabela 5.2: Descrição dos *data sets* de referência. O número de amostras é n e o número de variáveis de entrada é m.

Data set	Nomenclatura	n	m
Automobile MPG	Auto	392	6
Bank	Bank	8192	32
Boston Housing	Housing	506	13
Box and Jenkins	Box	290	10
Cagece	Cagece	360	33
Communities and Crime	Communities	1994	127
Concrete Compressive Strength	Concrete	1030	8
CPU	CPU	8192	21
Pumadyn	Puma	8192	32
Stock	Stock	950	9
Pyrimidines	Pyrim	74	26
Spectra	Sectra	60	401
Triazines	Triaz	186	59

Tabela 5.3: Resultados obtidos nos *data sets* referência, usando como medida de desempenho o NRMSE, com dados de validação normalizados para média nula e variância unitária.

NRMSE - Validação								
Data set	RLS	PLS	LASSO	IFSR	LSBST	LSBST-SV	LSBST-ON	LSBST-SV-ON
Auto	19.70	18.31	18.29	**18.02**	18.31	18.31	83.88	20.66
Bank	11.72	**11.29**	11.33	11.38	**11.29**	17.25	∞	17.20
Housing	14.98	**10.69**	10.83	11.52	10.70	18.66	79.29	23.17
Box	14.72	02.78	02.81	**02.74**	02.78	06.31	89.08	25.15
Cagece	12.94	**10.00**	10.90	10.62	10.53	16.50	∞	18.70
Communities	14.04	**13.98**	14.03	14.73	14.67	18.71	∞	23.39
Concrete	18.21	15.22	15.00	14.97	**14.95**	19.55	50.36	21.87
CPU	**09.70**	09.93	09.87	09.85	09.87	15.82	31.51	31.51
Puma	15.68	15.65	**15.59**	15.61	15.67	17.49	∞	17.51
Stock	16.95	**09.07**	**09.07**	09.26	09.08	17.70	∞	24.21
Pyrim	34.60	21.13	52.06	**20.73**	59.06	36.32	46.82	20.86
Spectra	20.50	09.22	**06.67**	09.59	07.99	33.96	∞	35.22
Triaz	∞	71.50	**25.71**	26.06	51.32	25.84	∞	**25.71**

as 15 variáveis de entrada e a variável de saída presentes no *data set* inicial, constituído por 8687 amostras, onde a quantidade de *outliers* chega a atingir 80% das variáveis e segundo [Souza and Araújo, 2014] a colinearidade em algumas das variáveis de entrada assume valores bastante elevados. Foi seguido o pré-processamento dos dados usado em [Kadlec and Gabrys, 2011], que se descreve pelos seguintes passos:

1. As primeiras 5800 amostras foram reduzidas para 580 através de um *downsampling*

Tabela 5.4: Resultados obtidos nos *data sets* referência, usando como medida de desempenho o NRMSE, com dados de teste normalizados para média nula e variância unitária.

NRMSE - Teste								
Data set	RLS	PLS	LASSO	IFSR	LSBST	LSBST-SV	LSBST-ON	LSBST-SV-ON
Auto	22.51	19.43	**19.42**	**19.42**	19.43	19.43	20.96	23.76
Bank	10.47	**10.25**	10.28	10.36	**10.25**	14.97	14.32	14.92
Housing	15.06	**11.69**	11.78	12.07	11.71	18.22	17.84	21.78
Box	11.42	**01.89**	01.94	01.95	**01.89**	05.30	25.71	21.60
Cagece	18.53	**14.21**	15.44	14.40	15.18	23.52	25.00	26.41
Communities	**12.58**	12.66	12.67	13.17	12.73	17.03	22.34	21.55
Concrete	17.69	13.70	13.70	**13.68**	13.70	19.22	15.98	21.68
CPU	09.80	09.36	09.29	**09.28**	09.29	15.85	10.97	14.54
Puma	16.12	16.07	**16.06**	16.08	16.10	18.07	18.04	18.08
Stock	18.03	08.80	**08.77**	09.19	08.78	17.88	26.93	26.14
Pyrim	23.45	47.85	74.35	48.01	∞	25.04	23.30	**16.88**
Spectra	27.68	05.42	08.95	09.50	**04.52**	32.52	35.11	32.75
Triaz	∞	40.74	23.17	**21.99**	35.07	22.03	∞	23.18

com um fator de 10;

2. Foram removidas as variáveis de entrada número 3, 4 e 15, devido a serem muito afetadas por dados em falta e *outliers*;

3. Foram removidas todas as amostras em que faltava a observação da saída.

Após o pré-processamento, o *data set* viu-se reduzido a 12 variáveis de entrada, uma variável de saída e 647 amostras.

As Tabelas 5.6, 5.7, 5.8, 5.9 e 5.10 apresentam os resultados de NRMSE obtidos pelos diferentes algoritmos para o *data set* da Ativação do Catalisador, com diferentes valores iniciais para o fator de esquecimento, λ. É de realçar que, para o algoritmo RLASSO-AF da Secção 3.7, o valor para o fator de esquecimento indicado nas tabelas é apenas o seu valor inicial, uma vez que o valor de λ varia em relação ao inicialmente definido. Cada uma das cinco tabelas apresenta os resultados referentes a uma das cinco disponibilidades ($D = 0\%$, $D = 10\%$, $D = 25\%$, $D = 50\%$ e $D = 100\%$), respetivamente.

É notório da observação das Tabelas 5.6, 5.7, 5.8, 5.9 e 5.10 que embora o método LSBST-SV-ON não tenha o melhor desempenho para todas as disponibilidades e todos os valores λ, consegue ter melhores desempenhos do que os restantes algoritmos, para a maioria dos valores de disponibilidade e de λ. Pode observar-se também que com o aumento da disponibilidade dos dados de saída, para atualização do modelo, existe um aumento do desempenho dos algoritmos em geral. Pode ainda observar-se que cada algoritmo tem normalmente um valor de λ com o qual obtém melhor desempenho e que este pode variar de um algoritmo para outro, mas normalmente não varia com a variação das disponibilidades.

Tabela 5.5: Descrição das variáveis do *Data set* da Ativação de Catalisador.

Variáveis	Descrição
x_1	Tempo
x_2	Fluxo de ar
x_3	Fluxo de gases combustíveis
x_4	Concentração de componente combustível nos gases combustíveis (fração de massa)
x_5	Temperatura total de entradas
x_6	Temperatura de Refrigeração
x_7	Temperatura a 1/20 do comprimento do reator
x_8	Temperatura a 2/20 do comprimento do reator
x_9	Temperatura a 4/20 do comprimento do reator
x_{10}	Temperatura a 7/20 do comprimento do reator
x_{11}	Temperatura a 11/20 do comprimento do reator
x_{12}	Temperatura a 16/20 do comprimento do reator
x_{13}	Temperatura a 20/20 do comprimento do reator
x_{14}	Concentração de oxigénio produzido (fração de massa)
x_{15}	Concentração de componente combustível produzido (fração de massa)
y_1	Atividade do catalisador no interior do reator

Tabela 5.6: Resultados obtidos para o *dataset* da Ativação do Catalisador, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 0%.

NRMSE - Teste ($D = 0$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	74.97	72.94	**32.97**	∞	70.34	80.01	∞
0.80	74.97	72.94	**32.97**	∞	70.34	80.01	∞
0.95	74.97	72.94	**32.97**	∞	70.34	80.01	∞
0.98	74.97	72.94	**32.97**	∞	70.34	80.01	∞
0.99	74.97	72.94	**32.97**	∞	70.34	80.01	∞
1	74.97	72.94	**32.97**	∞	70.34	80.01	∞

Tabela 5.7: Resultados obtidos para o *dataset* da Ativação do Catalisador, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 10%.

NRMSE - Teste ($D = 10$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**11.65**	43.03	32.94	95.08	42.34	12.52	32.78
0.80	13.85	27.19	32.95	83.31	42.34	**12.52**	17.81
0.95	18.83	20.23	32.99	55.00	42.34	**12.52**	29.35
0.98	23.60	23.92	32.99	48.96	42.34	**12.52**	31.48
0.99	26.80	25.54	32.99	48.69	42.34	**12.52**	32.08
1	31.23	27.33	32.99	49.60	42.34	**12.52**	32.61

Tabela 5.8: Resultados obtidos para o *dataset* da Ativação do Catalisador, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 25%.

NRMSE - Teste ($D = 25$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**06.31**	20.81	32.82	97.45	27.95	08.99	33.16
0.80	09.91	21.03	32.83	92.00	27.95	**08.99**	14.53
0.95	12.02	16.32	32.89	71.60	27.95	**08.99**	28.00
0.98	17.16	18,86	33.11	53.93	27.95	**08.99**	30.83
0.99	19.23	20.78	33.13	46.95	27.95	**08.99**	31.61
1	25.07	23.60	33.13	42.75	27.95	**08.99**	32.25

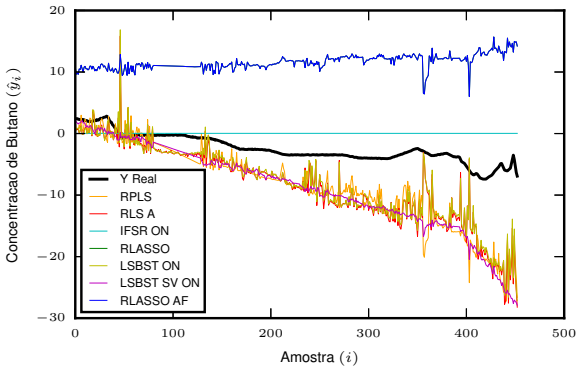
Tabela 5.9: Resultados obtidos para o *dataset* da Ativação do Catalisador, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 50%.

NRMSE - Teste ($D = 50$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**04.68**	30.80	32.43	97.39	21.17	06.28	30.47
0.80	**05.92**	10.07	32.46	96.04	21.17	06.28	12.47
0.95	11.00	15.49	32.48	82.85	21.17	**06.28**	27.45
0.98	12.10	15.19	32.71	66.36	21.17	**06.28**	30.64
0.99	17.07	16.91	33.13	53.86	21.17	**06.28**	31.59
1	22.17	21.23	33.56	40.98	21.17	**06.28**	32.32

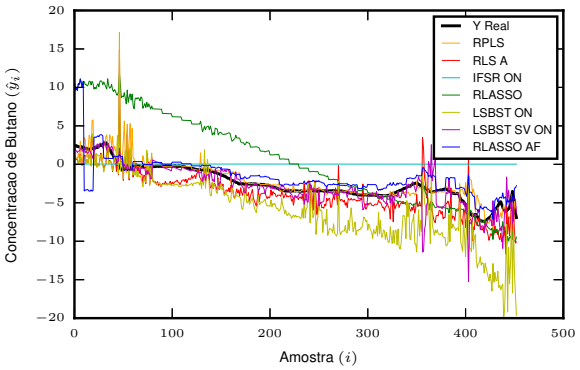
Tabela 5.10: Resultados obtidos para o *dataset* da Ativação do Catalisador, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 100%.

NRMSE - Teste ($D = 100$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**03.58**	20.92	31.51	97.61	19.08	03.61	29.52
0.80	04.14	07.86	31.53	96.46	19.08	**03.61**	11.42
0.95	07.23	09.35	31.64	90.36	19.08	**03.61**	27.20
0.98	11.48	12.10	31.68	79.04	19.08	**03.61**	30.53
0.99	12.17	12.45	32.13	65.91	19.08	**03.61**	31.61
1	19.92	18.56	34.69	39.60	19.08	**03.61**	32.46

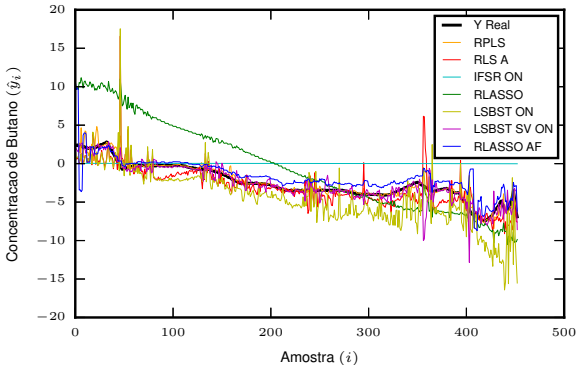
Na Figura 5.1 é representada graficamente a evolução temporal das variáveis estimadas na fase de teste, pelos algoritmos adaptativos, assim como a evolução da saída real para o *data set* da Ativação do Catalisador. Em cada gráfico da figura é apresentado o melhor desempenho de cada algoritmo, ou seja tem valores de λ diferentes dentro da mesma disponibilidade. Os parâmetros foram escolhidos para cada algoritmo de forma a maximizar o seu desempenho e a pertencerem às gamas de valores descritas na Tabela 5.1. Pode observar-se



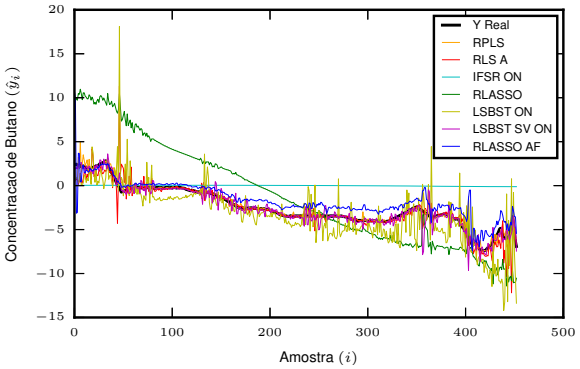
(a) 0% *Disponibilidade*



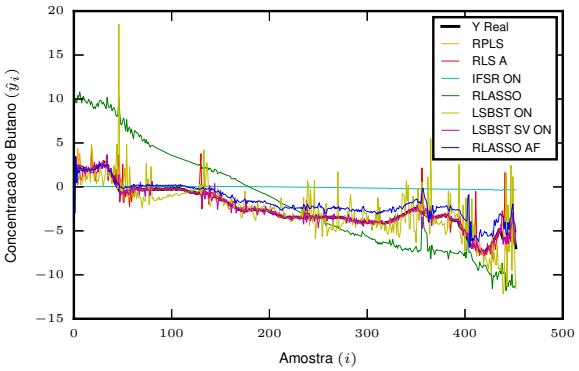
(b) 10% *Disponibilidade*



(c) 25% *Disponibilidade*



(d) 50% *Disponibilidade*



(e) 100% *Disponibilidade*

Figura 5.1: Comparação do desempenho do *Data set* de Ativação do Catalisador entre os algoritmos testados de acordo com a *disponibilidade* dos valores reais da variável de saída.

facilmente pela análise dos gráficos da Figura 5.1, que com o aumento da disponibilidade houve um aumento do desempenho, como já tinha sido verificado pela análise das Tabelas 5.6, 5.7, 5.8, 5.9 e 5.10. Pode ainda verificar-se pela análise da Figura 5.1 que o método proposto IFSR-ON teve pouca capacidade de se ajustar à saída desejada.

Tabela 5.11: Descrição das variáveis do *Data set* da estimação da quantidade de flúor.

Variáveis	Descrição
x_1	Quantidade de cloro no afluente (esgoto bruto)
x_2	Quantidade de cloro no efluente (esgoto tratado)
x_3	Turvação da água no afluente (água bruta)
x_4	Turvação no afluente
x_5	Turvação no efluente
x_6	pH da água no afluente
x_7	pH no afluente
x_8	pH no efluente
x_9	Cor da água no afluente
x_{10}	Cor do afluente
x_{11}	Cor do efluente
y_i	Flúor no efluente

5.4.3 *Data Set* da Estimação da Quantidade de Flúor

Uma ETAR é uma infra-estrutura que visa tratar as águas residuais, que recebe pelas redes de esgotos, para que estas sejam depois libertadas para o meio hídrico com níveis de poluição aceitáveis. O processo de tratamento de águas residuais efetuado numa ETAR é usualmente descrito por fases. A primeira fase, denominada por pré-tratamento, consiste numa filtragem que retêm os materiais sólidos de maiores dimensões. A segunda fase é denominada por tratamento primário, e nela são removidas as gorduras e óleos presentes na água residual, e são separadas a parte líquida das substâncias sólidas em suspensão que se vão depositar no fundo dos decantadores. Na fase seguinte, denominada por tratamento secundário, são adicionados micro-organismos e oxigénio que ajudam a decompor as impurezas que ainda permanecem na água residual transformando-as em lamas. Na quarta fase, conhecida por tratamento terciário, a água é desinfetada com cloro, ozono e radiação ultravioleta. Depois de tratada a água é libertada em rios ou oceanos.

Durante o processo de tratamento de águas residuais efetuado por uma ETAR, uma das dificuldades reside no controlo da quantidade de flúor presente no efluente, uma vez que níveis de flúor elevados podem contaminar o ecosistema aquático [Kurosaki, 1997] e esta quantidade é medida com recurso a análises laboratoriais. Para mostrar a capacidade de um SS solucionar este problema, foi utilizado o *Data set* da Estimação da Quantidade de Flúor.

O *Data set* da Estimação da Quantidade de Flúor é constituído por 11 variáveis de entrada e uma variável de saída, descritas na Tabela 5.11. O objectivo no processo associado a este *data set* é estimar a concentração de flúor no efluente de uma ETAR. O *data set* inicial é constituído por 13512 amostras das variáveis de entrada, adquiridas em 3 anos por sensores físicos, com uma taxa de amostragem de 2 horas. As amostras da quantidade de flúor foram adquiridas por análises laboratoriais realizadas a cada 24 horas. Depois de removidas as amostras em que faltava o valor da quantidade de flúor, o *data set* viu o

Tabela 5.12: Resultados obtidos para o *dataset* da ETAR, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 0%.

NRMSE - Teste ($D = 0$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74
0.80	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74
0.95	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74
0.98	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74
0.99	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74
1	23.63	24.55	**12.69**	28.74	24.55	22.78	28.74

Tabela 5.13: Resultados obtidos para o *dataset* da ETAR, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 10%.

NRMSE - Teste ($D = 10$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**10.18**	46.83	12.72	18.14	∞	15.18	16.98
0.80	11.39	19.83	12.75	12.71	∞	15.18	**10.69**
0.95	16.38	19.24	12.77	**11.26**	∞	15.18	11.30
0.98	18.06	20.06	12.78	11.57	∞	15.18	**11.55**
0.99	19.58	20.43	12.78	11.71	∞	15.18	**11.63**
1	20.45	20.83	12.78	11.85	∞	15.18	**11.71**

Tabela 5.14: Resultados obtidos para o *dataset* da ETAR, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 25%.

NRMSE - Teste ($D = 25$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**08.77**	30.63	12.58	20.08	∞	11.37	18.30
0.80	**09.47**	18.49	12.73	15.06	∞	11.37	09.61
0.95	11.43	12.57	12.90	**10.00**	∞	11.37	10.86
0.98	15.26	15.39	12.89	**10.00**	∞	11.37	11.28
0.99	15.20	16.37	12.88	**10.34**	∞	11.37	11.41
1	17.39	17.51	12.91	**10.75**	∞	11.37	11.54

seu tamanho reduzido a 1002 amostras. Segundo [Souza and Araújo, 2014] as variáveis de entrada apresentam um nível de colinearidade entre elas aceitável, pelo que se espera que os algoritmos baseados nos LS não vejam o seu desempenho comprometida. Nas Tabelas 5.12, 5.13, 5.14, 5.15 e 5.16 são apresentados os resultados de NRMSE obtidos pelos diferentes

Tabela 5.15: Resultados obtidos para o *dataset* da ETAR, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 50%.

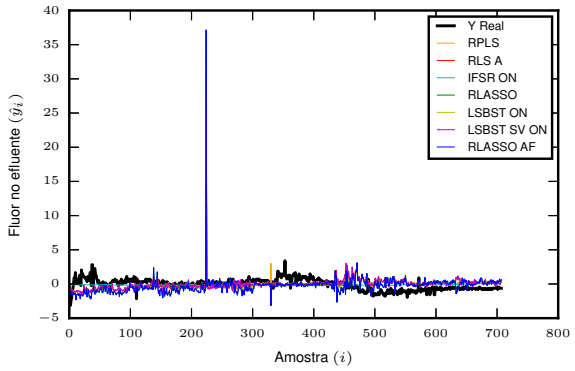
NRMSE - Teste ($D = 50$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**07.91**	∞	12.39	21.92	∞	08.87	19.38
0.80	**08.33**	13.89	12.43	18.32	∞	08.87	09.52
0.95	09.07	09.12	12.99	12.83	∞	**08.87**	10.82
0.98	10.53	10.74	13.25	10.65	∞	**08.87**	11.42
0.99	11.83	12.34	12.66	10.75	∞	**08.87**	11.60
1	17.52	14.41	12.66	12.66	∞	**08.87**	11.76

Tabela 5.16: Resultados obtidos para o *dataset* da ETAR, sendo a medida de desempenho o NRMSE nos dados de teste, com uma disponibilidade de 100%.

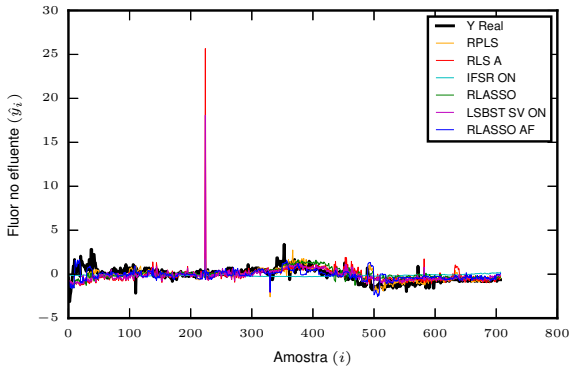
NRMSE - Teste ($D = 100$)							
λ	RPLS	RLS-A	IFSR-ON	RLASSO	LSBST-ON	LSBST-SV-ON	RLASSO-AF
0.50	**07.79**	∞	11.76	23.13	∞	07.90	22.93
0.80	08.00	∞	11.72	18.97	∞	**07.90**	09.46
0.95	08.06	08.62	12.85	14.35	∞	**07.90**	10.80
0.98	09.50	08.69	13.42	10.93	∞	**07.90**	11.50
0.99	10.06	09.59	14.03	10.23	∞	**07.90**	11.72
1	14.71	12.33	12.97	11.86	∞	**07.90**	11.89

algoritmos para o *data set* da ETAR, com diferentes valores para o fator de esquecimento, λ. Para o algoritmo RLASSO-AF da Secção 3.7, o valor de λ apresentado nas tabelas é apenas o valor inicial do seu fator de esquecimento. De forma análoga ao que aconteceu no *data set* da Ativação do Catalisador, cada umas das tabelas representa os resultados de uma das cinco disponibilidades ($D = 0\%$, $D = 10\%$, $D = 25\%$, $D = 50\%$ e $D = 100\%$), respetivamente. Pode verificar-se nas tabelas que o algoritmo com melhor desempenho, neste *data set*, não só mudou com o valor de λ, mas variou também com as disponibilidades. É de notar que o algoritmo LSBST-ON apresentou incapacidade de estimar para todas as disponibilidades, exceto para o caso estático, $D = 0\%$. Pode verificar-se que a melhor desempenho quase sempre pertenceu ao método RPLS, mas que o LSBST-SV-ON por ter um desempenho não influenciado por λ, cobre uma gama mais abrangente para este parâmetro.

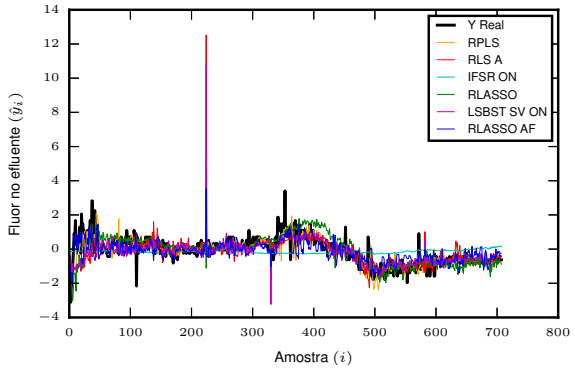
Na Figura 5.2 é representada a evolução temporal das variáveis estimadas na fase de teste, pelos algoritmos adaptativos, assim como a evolução da saída real para o *data set* da ETAR. De forma similar ao *data set* da Ativação do Catalisador, cada gráfico da figura apresenta a melhor desempenho de cada algoritmo, ou seja tem valores de λ diferentes dentro da mesma disponibilidade. Os parâmetros iniciais foram escolhidos para cada algoritmo de forma a



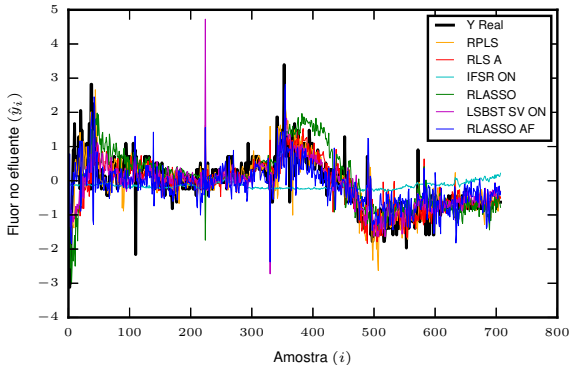
(a) 0% *Disponibilidade*



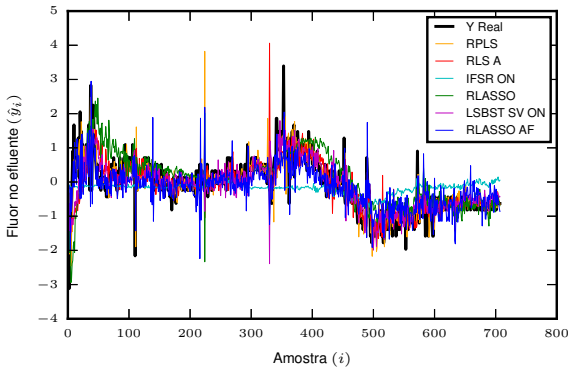
(b) 10% *Disponibilidade*



(c) 25% *Disponibilidade*



(d) 50% *Disponibilidade*



(e) 100% *Disponibilidade*

Figura 5.2: Comparação do desempenho do *Data set* da ETAR entre os algoritmos testados de acordo com a *disponibilidade* dos valores reais da variável de saída.

maximizar o seu desempenho e a pertencerem às gamas de valores descritas na Tabela 5.1. É de notar que apenas na sub-figura 5.2a se representou a evolução do desempenho do algoritmo LSBST-ON, pois para as restantes disponibilidades o algoritmo apresentou incapacidade de efetuar a estimação da saída.

É fácil verificar pela observação da figura 5.2 que todos os algoritmos têm um aumento de desempenho com o aumento da disponibilidade, com a já referida exceção.

5.5 Discussão dos Resultados

Na experiência *offline*, na qual os algoritmos foram aplicados a treze *data sets* de referência, verificou-se consistência entre os resultados obtidos por todos os algoritmos para os dados de validação e os dados de teste. Tendo apenas o algoritmo LSBST-ON quebrado este padrão, mostrando dificuldade ou até mesmo incapacidade em estimar a saída, quando aplicado aos dados de validação de cada um dos *data sets*. O método LSBST-SV proposto, embora não tenha tido o melhor desempenho obtido para nenhum dos *data sets*, obteve resultados comparáveis com os dos outros métodos *offline* e coerência entre os resultados obtidos na fase de validação e na fase de teste. Tanto o método LSBST-ON, como o proposto LSBST-SV-ON, exibiram resultados não muito discrepantes dos métodos *offline*, o que sugere que os métodos *boosting* comparados podem ser boas ferramentas a utilizar em ambientes variantes no tempo.

Na experiência realizada para o *data set* da Ativação do Catalisador, o método IFSR-ON proposto obteve o melhor desempenho no caso estático ($D = 0\%$), embora o NRMSE tenha sido superior ao limite do aceitável. Para as restantes disponibilidades, os melhores desempenhos pertencem ao método RPLS, contudo o método proposto LSBST-SV-ON teve sempre um desempenho equiparável chegando a diferença do valor obtido para o NRMSE a ser de 0.03% (para o valor mais baixo de NRMSE). Para além das vantagens enumeradas para os métodos *booosting* ao longo do Capítulo 4 e da robustez apresentada pelo método LSBST-SV-ON, é de notar que por não depender de λ, apresenta normalmente melhores resultados de NRMSE para os casos em que os outros algoritmos utilizam valores de $\lambda > 0.5$, o que pode ser uma vantagem, pois embora não seja o caso deste *data set* em particular, valores de λ muito baixos podem causar *overfitting*. Comparando o método de *boosting* LSBST-SV-ON proposto com o LSBST-ON, verifica-se que à exceção do cenário ($D = 0\%$), o método LSBST-SV-ON obteve melhores desempenhos, chegando a apresentar uma redução de $81,08\%$ no NRMSE.

Na última experiência efetuada, os algoritmos adaptativos foram aplicados ao *data set* da ETAR para estimar a quantidade de flúor do efluente. À semelhança do que aconteceu com o *data set* da Ativação do Catalisador, no cenário estático o melhor desempenho pertence ao método IFSR-ON. Para as restantes disponibilidades os melhores desempenhos pertencem sempre ao método RPLS, tendo o método LSBST-SV-ON proposto nesta dissertação apresentado resultados comparáveis, chegando a diferença do valor de NRMSE obtida, para o valor mais baixo de NRMSE, ser de apenas 0.11%. Nesta experiência o método LSBST-ON mostrou incapacidade de estimar a saída para todas as disponibilidades ($D \neq 0\%$).

Capítulo 6

Conclusão e Trabalhos Futuros

6.1 Conclusão

Nesta dissertação foram propostos três métodos novos para o projeto de SS, sendo dois deles adaptativos. Qualquer um dos métodos propostos apenas utiliza o conhecimento empírico dos dados de entrada e de saída, não sendo necessário qualquer conhecimento prévio sobre o modelo do processo. Para validar e avaliar o desempenho do métodos propostos, estes métodos bem como os restantes métodos descritos nos Capítulos 3 e 4 foram aplicados a treze *data sets* de referência relativos a problemas do mundo real e a dois *data sets* relativos a problemas do mundo real variantes no tempo.

Pela observação dos resultados dos treze *data sets* de referência, podemos constatar que os resultados obtidos pelos métodos LSBST-ON e LSBST-SV-ON não são muito discrepantes dos obtidos pelos métodos *offline*, o que indica que para além de serem simples de implementar para funcionar *online*, os métodos *boosting* podem ser uma ferramenta útil para ambientes variantes no tempo. Os melhores desempenhos para ambientes *offline* foram maioritariamente obtidos pelos métodos LASSO e PLS.

Da comparação dos resultados, dos diferentes algoritmos adaptativos, obtidos para os dois problemas do mundo real, verifica-se que o melhor desempenho para a disponibilidade de $D = 0\%$ pertence ao método proposto IFSR-ON, mas para ambos os *data sets* o NRMSE é superior a 10% o que, conforme já previamente referido, não é aceitável. Para as restantes disponibilidades os valores mais baixos de NRMSE pertencem ao método RPLS (ocorreram para valores de $\lambda = 0.5$), sendo os resultados deste método, sempre, quase igualados pelo método proposto LSBST-SV-ON. Para o caso disponibilidades ($D \neq 0\%$) e $\lambda \geq 0.8$, os valores mais baixos de NRMSE pertencem ao método proposto LSBST-SV-ON. Os métodos RPLS e LSBST-SV-ON obtêm valores de NRMSE inferiores a 10% para disponibilidades iguais ou superiores a $D = 25\%$ para o *data set* da Ativação do Catalisador, e para disponibilidades iguais ou superiores a $D = 25\%$ e $D = 50\%$ respetivamente para o *data set* da ETAR. É de realçar que o método proposto, LSBST-SV-ON, não depende de λ, o que o torna uma valiosa ferramenta em ambientes onde a aprendizagem é feita por métodos que

utilizam valores de $\lambda < 0.8$, podendo originar que o modelo aprenda ruídos presentes nos dados. Adicionalmente, a não dependência de λ torna mais fácil a realização do projeto de aplicação do método.

Pode ainda concluir-se que a utilização do método proposto, LSBST-SV-ON, para a estimação da quantidade de flúor no efluente da ETAR, deve originar uma descida dos custos com análises laboratoriais, uma vez que se poderia reduzir a frequência destas para metade.

6.2 Trabalhos Futuros

Uma vez que os métodos *boosting* implementados nesta dissertação se mostraram ferramentas úteis para o desenvolvimento de SS, poderia ter interesse experimentar diferentes *loss functions* e diferentes funções básicas, nomeadamente para implementar o IFSR *boosting online*, dado que o método proposto IFSR-ON apresentou um fraco desempenho quando comparado com os métodos *boosting online*.

Visto que a diversidade dos modelos individuais é uma das chaves do sucesso dos ME, seria interessante abordar técnicas de manipulação dos *data sets*, em particular poderia ser vantajoso combinar os métodos *boosting* implementados com o método de *bagging* de modo a alcançar uma maior diversidade nos modelos individuais obtidos.

Bibliografia

[Anagnostopoulos *et al.*, 2008] Christoforos Anagnostopoulos, Dimitris Tasoulis, David J. Hand, and Niall M. Adams. Online optimization for variable selection in data streams. *ECAI-European Conference on Artificial Intelligence*, pp. 132–136, 2008. (Citado nas páginas 2, 3, 6, 7, 15, 22, e 23).

[Babenko *et al.*, 2009] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. A family of online boosting algorithms. In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 1346–1353. IEEE, 2009. (Citado nas páginas iii, v, 4, 6, 7, e 31).

[Bishop *et al.*, 2006] Christopher M Bishop *et al. Pattern recognition and machine learning*, vol. 1. springer New York, 2006. (Citado nas páginas 6, 7, e 34).

[Corke, 1996] Peter I. Corke. *Pumadyn family of datasets*, 1996. URL `http://www.cs.toronto.edu/~delve/data/pumadyn/desc.html`. (Citado na página 37).

[Dayal and MacGregor, 1997] Bhupinder S. Dayal and John F. MacGregor. Recursive Exponentially Weighted PLS and Its Applications to Adaptive Control and Prediction. *Journal of Process Control*, vol. 7, no. 3, pp. 169–179, 1997. (Citado nas páginas 3, e 4).

[Fortuna *et al.*, 2007] Luigi Fortuna, Salvatore Graziani, Alessandro Rizzo, and Maria G. Xibilia. *Soft Sensors for Monitoring and Control of Industrial Processes*. Springer, 2007. (Citado nas páginas 1, 9, e 14).

[Frank and Asuncion, 2010] A. Frank and A. Asuncion. *UCI machine learning repository*, 2010. URL `http://archive.ics.uci.edu/ml`. (Citado na página 37).

[Friedman, 2001] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pp. 1189–1232, 2001. (Citado nas páginas iii, v, 4, 6, 28, 29, e 30).

[Gomnam and Jazayeri-rad, 2013] Ebrahim Gomnam and Hooshang Jazayeri-rad. Development of an Adaptive Soft Sensor Based on FCMILSSVR. *International Journal of Scientific & Technology Research*, vol. 2, pp. 199–203, 2013. (Citado na página 5).

[Hastie *et al.*, 2009] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer, 2 ed., 2009. (Citado nas páginas iii, v, 3, 6, 7, 24, e 28).

[Haykin, 1996] Simon Haykin. *Adaptive Filter Theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996. (Citado nas páginas iii, v, 2, 3, 6, 7, e 23).

[Jang *et al.*, 1997] Jyh-Shing Roger Jang, Chuen-Tsai Sun, and Eiji Mizutani. *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. (Citado nas páginas 3, 5, 6, 18, e 19).

[Kadlec and Gabrys, 2011] Petr Kadlec and Bogdan Gabrys. Local Learning-Based Adaptive Soft Sensor for Catalyst Activation Prediction. *American Institute of Chemical Engineers Journal*, vol. 57, pp. 1288–1301, 2011. (Citado nas páginas 37, 40, e 41).

[Kadlec *et al.*, 2009] Petr Kadlec, Bogdan Gabrys, and S. Strandt. Data-driven Soft Sensors in the Process Industry. *Computers & Chemical Engineering*, vol. 33, pp. 795–814, 2009. (Citado nas páginas 2, 14, e 15).

[Kadlec *et al.*, 2011] Petr Kadlec, Ratko Grbić, and Bogdan Gabrys. Review of adaptation mechanisms for data-driven soft sensors. *Computers & chemical engineering*, vol. 35, no. 1, pp. 1–24, 2011. (Citado nas páginas 2, e 15).

[Kalivas, 1997] John H Kalivas. Two data sets of near infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, vol. 37, no. 2, pp. 255–259, 1997. (Citado na página 37).

[Kaur and Dewan, 2010] Dalvinder Kaur and Lillie Dewan. Pre-Filtering In Robust Model Estimation-A Brief Tour. *International Journal of Engineering (IJE)*, vol. 4, no. 3, pp. 254–261, 2010. (Citado na página 14).

[King *et al.*, 1992] Ross D King, Stephen Muggleton, Richard A Lewis, and MJ Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings of the national academy of sciences*, vol. 89, no. 23, pp. 11322–11326, 1992. (Citado na página 37).

[Kurosaki, 1997] Hisao Kurosaki. Reduction of fluorine-containing industrial waste using aluminum-solubility method. *Oki Denki Kenkyu Kaihatsu*, vol. 64, no. 4, pp. 65–68, 1997. (Citado na página 46).

[Lee *et al.*, 2004] Changkyu Lee, Sang Wook Choi, and In-Beum Lee. Sensor Fault Identification Based on Time-lagged PCA in Dynamic Processes. *Chemometrics and Intelligent Laboratory Systems*, vol. 70, no. 2, pp. 165–178, 2004. (Citado na página 4).

[Li *et al.*, 2000] Weihua Li, H. Henry Yue, Sergio Valle-Cervantes, and S. Joe Qin. Recursive PCA for Adaptive Process Monitoring. *Journal of Process Control*, vol. 10, no. 5, pp. 471–486, 2000. (Citado na página 4).

[Macias *et al.*, 2006] JJ Macias, Plamen Angelov, and Xiaowei Zhou. A method for predicting quality of the crude oil distillation. In: *Evolving Fuzzy Systems, 2006 International Symposium on*, pp. 214–220. IEEE, 2006. (Citado nas páginas 2, e 15).

[Mathisen, 2001] Rune Mathisen. *MVARTOOLS for Matlab*, 2001. URL `http://www.bitjungle.com/mvartools/`. (Citado na página 38).

[Nelson *et al.*, 1996] Philip RC Nelson, Paul A Taylor, and John F MacGregor. Missing data methods in PCA and PLS: Score calculations with incomplete observations. *Chemometrics and intelligent laboratory systems*, vol. 35, no. 1, pp. 45–65, 1996. (Citado na página 14).

[Plackett, 1950] Ronald L Plackett. Some theorems in least squares. *Biometrika*, pp. 149–157, 1950. (Citado na página 17).

[Qian *et al.*, 2013] Junyang Qian, Jerome Friedman, Trevor Hastie, Noah Simon, and Rob Tibshirani. *Glmnet for Matlab*, 2013. URL `http://www.stanford.edu/~hastie/glmnet_matlab/`. (Citado na página 38).

[Qin, 1998] S.J̃oe Qin. Recursive PLS Algorithms for Adaptive Data Modeling. *Computers & Chemical Engineering*, vol. 22, pp. 503–514, 1998. (Citado nas páginas 3, 4, 6, 7, 16, 20, e 21).

[Rasmussen, 1996] Carl Rasmussen. *Bank family of Datasets*, 1996. URL `http://www.cs.toronto.edu/~delve/data/bank/desc.html`. (Citado na página 37).

[Reinsel, 1999] A. Reinsel. *Box and jenkins dataset*, 1999. URL `http://www.stat.wisc.edu/~reinsel/bjr-data/gas-furnace`. (Citado na página 37).

[Robinson and Schumacker, 2009] Cecil Robinson and Randall E Schumacker. Interaction effects: centering, variance inflation factor, and interpretation issues. *Multiple Linear Regression Viewpoints*, vol. 35, no. 1, pp. 6–11, 2009. (Citado na página 3).

[Rumelhart *et al.*, 1986] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning Representations by Back-Propagating Errors. *Nature*, vol. 323, no. 9, pp. 533–536, 1986. (Citado na página 5).

[Sliskovic *et al.*, 2011] Drazen Sliskovic, Ratko Grbic, and Zeljko Hocenski. Methods for Planat Data-Based Process Modeling in Soft-Sensor Development. *Automatika*, vol. 52, pp. 306–318, 2011. (Citado na página 4).

[Soares *et al.*, 2011] Symone Soares, Rui Araújo, Pedro Sousa, and Francisco Souza. Design and Application of Soft Sensor Using Ensemble Methods. In: *Proc. 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2011)*, pp. 1–8. IEEE, Toulouse, France, September 5-9 2011. (Citado nas páginas 4, e 27).

[Sorenson, 1970] Harold W Sorenson. Least-squares estimation: from Gauss to Kalman. *Spectrum, IEEE*, vol. 7, no. 7, pp. 63–68, 1970. (Citado na página 17).

[Souza and Araújo, 2014] A. Souza and R. Araújo. Online Mixture of Univariate Linear Regression Models for Adaptive Soft Sensors. *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 937–945, May 2014. (Citado nas páginas 37, 39, 41, e 47).

[Souza *et al.*, 2010] Francisco Souza, Pedro Santos, and Rui Araújo. Variable and Delay Selection Using Neural Networks and Mutual Information for Data-Driven Soft Sensors. In: *Proc. 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010)*, pp. 1–8. IEEE, Bilbao, Spain, September 13-16 2010. (Citado nas páginas 3, 15, e 16).

[Tibshirani, 1996] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996. (Citado nas páginas 3, e 22).

[Torgo, 1991] L. Torgo. *Stock dataset*, 1991. URL `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html`. (Citado na página 37).

[Wang *et al.*, 2010] David Wang, Jun Liu, and Rajagopalan Srinivasan. Data-driven soft sensor approach for quality prediction in a refining process. *Industrial Informatics, IEEE Transactions on*, vol. 6, no. 1, pp. 11–17, 2010. (Citado na página 3).

[Wang *et al.*, 2005] Xun Wang, Uwe Kruger, and George W. Irwin. Process Monitoring Approach Using Fast Moving Window PCA. *Industrial & Engineering Chemistry Research*, vol. 44, no. 15, pp. 5691–5702, 2005. (Citado na página 4).