UNIVERSIDADE FEDERAL DO RIO DE JANEIRO
PROGRAMA DE PÓS-GRADUAÇÃO EM TECNOLOGIA DE PROCESSOS QUÍMICOS E BIOQUÍMICOS

Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data*

Dissertação de Mestrado

Felipo Doval Rojas Soares

Rio de Janeiro

Setembro de 2017

Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data*

Felipo Doval Rojas Soares

Dissertação submetida ao Corpo Docente do Curso de Pós-Graduação em Tecnologia de Processos Químicos e Bioquímicos da Escola de Química da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências.

Aprovado por:

_____

Prof. Maurício Bezerra de Souza Júnior, D.Sc.
(orientador)

_____

Prof. Brunno Ferreira dos Santos, D.Sc.

_____

Profa. Andrea Valdman, D.Sc.

_____

Prof. Álvaro José Boareto Mendes, D.Sc.

Rio de Janeiro, RJ - Brasil
Setembro de 2017

Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data*

Felipo Doval Rojas Soares

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Tecnologia de Processos Químicos e Bioquímicos da Escola de Química da Universidade Federal do Rio de Janeiro, para a obtenção do grau de Mestre em Ciências.

Orientador: Prof. Maurício Bezerra de Souza Júnior, D.Sc.

Rio de Janeiro, RJ – Brasil

Setembro de 2017

DS676t Soares, Felipo Doval Rojas

Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data* / Felipo Doval Rojas Soares --2017

XVIII, 90 f, il.

Dissertação (Mestrado em Tecnologia de Processos Químicos e Bioquímicos) – Universidade Federal do Rio de Janeiro, Escola de Química, Rio de Janeiro, 2017

.

Orientador: Maurício Bezerra de Souza Júnior

1. Redes neuronais. 2. Máquina de Vetor de Suporte. 3. *Deep Learning Networks*. 4. Redes Recorrentes. – Teses. I. De Sousa Jr, Maurício (Orient.). II. Universidade Federal do Rio de Janeiro. Programa de Pós-Graduação em Tecnologia de Processos Químicos e Bioquímicos. Escola de Química. III. Título

AGRADECIMENTOS

Agradeço a minha família: meus pais George e Ina e meus irmãos Luana e Ian, pelo apoio que têm me dado. Não conseguiria fazer esse mestrado sem vocês.

Aos meus amigos, que deixam a vida mais leve e mais divertida.

Ao professor Frederico Jandre, do Programa de Engenharia Biomédica da COPPE, por ter cedido o computador usado nos cálculos e pela hospitalidade.

Ao meu Orientador Maurício de Souza Jr. pela dedicação e ajuda ao desenvolver esse trabalho.

Resumo da Dissertação de Mestrado apresentada ao Curso de Pós-Graduação em Tecnologia de Processos Químicos e Bioquímicos da Escola de Química/UFRJ como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências.

Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data*

Felipo Doval Rojas Soares

Setembro, 2017

Orientador: Prof. Maurício Bezerra de Souza Júnior, D. Sc.

Estamos vivendo a 4ª Revolução Industrial, com grande aumento na capacidade de se gerar, armazenar e acessar dados. Isso cria uma demanda por uma análise de informações, de forma a otimizar a produção. Atualmente esses dados são analisados por algoritmos de estatística multivariada, que não são ideais para essa tarefa por terem uma capacidade de análise limitada. Algoritmos de *Machine Learning* já são empregados em diversas áreas, como robótica e reconhecimento de imagens. Nesta dissertação, utilizamos algoritmos de *Machine Learning*, incluindo novos paradigmas de redes neuronais, para tratar dois problemas comuns na indústria química: *i*. inferência estatística e predição *ii*. detecção e diagnóstico de falhas. Foram usados respectivamente dois *benchmarks* da engenharia de processos: o reator de van de Vusse, ilustrativo de um processo complexo, e a planta completa Tennessee Eastman. Também inserimos desafios comuns desse ambiente, como ruído e ausência de dados e espaço de estado incompleto. Foram empregadas grandes quantidades de dados, a fim de simular um contexto *Big Data*. Para inferência estatística, foram usadas redes neuronais tradicionais *feedforward* e novos paradigmas de redes recorrentes (Redes *Echo State*), as quais foram comparadas entre si e com as técnicas baseadas em modelos lineares. Para detecção e diagnóstico de falhas, foram usadas redes recorrentes, simples e *gated*; redes de convolução 2D e máquinas de vetor de suporte, de *kernel* radial e linear. Os resultados foram comparados entre si e com os métodos lineares: análise de componentes principais dinâmicos para detecção e análise do discriminante de Fischer dinâmico para diagnóstico. Para inferência estatística, as redes foram mais estáveis que os métodos lineares. Elas foram capazes de predizer a inversão de ganho do reator e tiveram um erro quadrático médio para predição em ciclo fechado reduzido até aproximadamente 1/3 do erro dos modelos lineares. Para detecção e diagnóstico de falha, os resultados obtidos foram superiores aos métodos lineares, uma melhora de até 8% em acurácia e 34% na área da curva característica de operação do receptor, porém o treinamento foi complexo e custoso computacionalmente. Concluimos que as técnicas de *Machine Learning* trazem benefícios à engenharia de processos químicos industriais, mas, no momento, o custo de seu emprego deve ser criteriosamente avaliado.

Abstract of Thesis presented to Graduate Program on Technology of Chemical and Biochemical Processes - EQ/UFRJ as partial fulfillment of the requirements for the degree of Master of Sciences.

Machine Learning Techniques Applied to Inference and Fault Detection and Diagnosis of Industrial Chemical Processes in Big Data Context.

Supervisor: Prof. Maurício Bezerra de Souza Júnior, D.Sc.

We are now living the 4th Industrial Revolution. There is a great increase in the quantity, quality and accessibility of data, due to lowering costs of sensors, communication devices and data storage. This creates a demand for data analysis, to optimize production. Currently multivariate statistical methods are used with that purpose, but they are incapable of handling all possible features of data. Machine learning is a set of algorithms created to deal with huge quantities of data and is used in several areas. In this work we used machine learning to investigate two common problems in the industry: statistical inference and fault detection and diagnosis. We used two benchmark problems: the van de Vusse reactor and the Tennessee Eastman process. We have inserted common limitations occurring in the industry, like noise, incomplete state space and data absence. For statistical inference, we employed traditional feedforward and new recurrent neural network paradigms and they were compared to two analog linear models: autoregressive with exogenous input model and state space models. For fault detection and diagnosis, we adopted new (recurrent and convolutional) neural networks paradigms and support vector machines. We compared the results with dynamic principal components analysis, for fault detection, and dynamic Fischer discriminant analysis, for diagnosis. For statistical inference the results were quite good. The networks were capable of predicting the static gain inversion in the reactor and proved to be more stable than the linear models, with closed loop prediction mean squared error reduced down to one third. The fault detection and diagnosis results for the machine learning methods were better than the linear model. The best neural network had an increase of 8% of accuracy and 34% of receiver operating characteristic curve area, however the computational costs were much higher due to the higher number of parameters and gradients. We have concluded that machine learning techniques bring benefits to industrial chemical process engineering, but at the moment the cost of its usage should be carefully evaluated.

Índice

1) Introdução 1
1.1) Motivação: 1
1.2) Objetivo 2
1.3) Estrutura da dissertação 2
2) Revisão Bibliográfica 4
2.1) *Machine Learning* 4
2.2) Desafios à aplicação de *Machine Learning* a processos químicos 8
2.3) Inferência estatística e detecção e diagnóstico de falha 11
2.4) Máquinas de Vetores de Suportes 12
2.5) Redes Neuronais 19
2.5.1) Redes neuronais recorrentes 26
2.5.2) Redes neuronais convolucionais 30
2.6) Métodos lineares de classificação e regressão 32
2.6.1) ARX 32
2.6.2) Modelo de espaço de estado 33
2.6.3) PCA dinâmico 34
2.6.4) FDA dinâmico 35
2.7) Modelagem de incerteza 37
3) Inferência estatística 39
3.1) Estudo de caso: Reator de van de Vusse 39
3.2) Resultados 46
3.2.1) Modelos lineares 49
3.2.2) Rede *Echo State* 51
3.2.3) Rede neuronal com atraso temporal 53
3.3) Sumário 57
4) Detecção e Diagnóstico de Falhas 60
4.1) Problema Tennessee Eastman 60
4.2) Resultados 70
4.2.1) PCA e FDA dinâmico 71
4.2.2) Redes neuronais recorrentes 75
4.2.3) Redes neuronais convolucionais 78
4.2.4) Máquinas de vetores de suporte 82

4.3) Análise por perturbação 86
4.4) Sumário da classificação de falha 87
5) Conclusão 89
6) Referências Bibliográficas 91
Apêndice I - Código do reator de Van der Vusse 97
Apêndice II - Perturbações adicionadas ao reator de van de Vusse. 100
Apêndice III - Perturbações adicionados ao processo Tennessee Eastman. 103
Apêndice IV - Valores utilizados na estandardização da base de dados 105

Lista de símbolos

Símbolo	Descrição
%ab	Porcentagem da abertura das válvulas
**	Multiplicação matricial elemento a elemento
A	Substância química hipotética
A^{SS}	Pesos do modelo de espaço de estados relacionando h_t com h_{t+1}
A_{TT}	Área de troca térmica
b	Termo de viés
B	Substância química hipotética
B^{BFGS}	Aproximação da Hessiana do método BFGS
B^{SS}	Pesos do modelo de espaço de estados relacionando x_t com h_{t+1}
C^{MVS}	Parâmetro de regularização da MVS
C	Substância química hipotética
C_i	Concentração da substância i
cp	Capacitância térmica
C^{SS}	Pesos do modelo de espaço de estados relacionando h_t com y_t
C_v	Constante da válvula
d	Direção da otimização
D	Substância química hipotética
D^{SS}	Pesos do modelo de espaço de estados relacionando x_t com y_{t+1}
E	Substância química hipotética
e	Erro dado pela diferença entre o *setpoint* e o valor da variável medida
E_i	Energia de ativação da reação i
F()	Portão de esquecimento
F	Substância química hipotética
f()	Função
F_{in}	Corrente de entrada
F_{out}	Corrente de saída
F_α (a, n-a)	Distribuição F com α% de confiabilidade e a, n-a graus de liberdade
G	Substância química hipotética
g()	Função
H	Diferença dos gradientes
h	Estado interno do modelo
h_r	Altura do líquido no reator
i	Portão de entrada
k_{0i}	Fator pré exponencial da reação i
k_i	Constante da reação i
K_i	Ganho integral do controlador
K_p	Ganho proporcional do controlador
o	Portão de saída
p	Passo da otimização
P	Loadings dos componentes principais
q_g	Calor gerado pelas reações
q_r	Calor removido pelo trocador de calor
r	Portão de redefinição
R	Constante dos gases
r_i	Taxa da reação i
S_b	Matriz de espalhamento entre as classes
S_j	Matriz de espalhamento interno

S_t	Matriz de espalhamento total
S_w	Matriz de espalhamento dentro das classes
t	Alvo
T	Temperatura
T^2	Estatística de Hotelling
T_{H2O}	Temperatura da camisa de resfriamento
T_i	Tempo de integração
T_{in}	Temperatura da corrente de entrada
T_s	Tempo de amostragem
U	Matriz unitária linha x linha da SVD
U_k	Ação do controlador
U_{TT}	Coeficiente global de troca térmica
v	Saída da camada escondida da rede neuronal
V	Matriz unitária coluna x coluna
V	Volume
V_r	Volume da camisa de resfriamento
W	Pesos de modelo de *Machine Learning*
w	Vetor de pesos
W^{in}	Pesos da camada de entrada
W^{out}	Pesos da camada de saída
x	Vetor de entrada de modelo
X	Matriz de dados
x*	Sinal filtrado
x_m	Vetor médio das características
XMEAS	Variável medida
x_{mj}	Vetor médio das características dos dados pertencentes a classe j
XMV	Variável controlada
y	Saída de modelo
z	Portão de atualização
α	Fator de amortização
α^{ARX}	Pesos da seção autorregressiva do ARX
β	Fator de momento
β^{ARX}	Pesos da seção exógena do ARX
Δh_{ri}	Calor da reação i
ε	Ruído aleatório
η	Taxa de aprendizagem da rede neuronal
λ	Multiplicador de Lagrange
ξi	Variável de folga
ρ	Densidade
Σ	Matriz diagonal dos valores singulares

Lista de abreviações

Abreviação	Descrição
ARX	modelo autorregressivo exógeno
ADAM	*adaptive moment estimation*
BFGS	método de Broyden–Fletcher–Goldfarb–Shanno
CSTR	Reator continuamente agitado
ESN	*Echo State Network*
DDF	Detecção e diagnóstico de falhas
FDA	Análise do discriminante de Fischer
GRU	*Gated Recurrent unit*
kW	Quilowatts
kPa	Quilopascal
kscmh	1000 metros cúbicos padrão
LSTM	*Long-Short Term Memory Network*
MLP	Perceptron multicamadas
MLP-AT	Rede neuronal multicamadas com atraso temporal
MNIST	Base de dados para classificação de dígitos
MVS	Máquina de vetor de suporte
NRMSE	Raiz do erro médio quadrático normalizado pela variância
PCA	Análise de componentes principais
RBF	Função de base radial
ReLU	Retificadora linear
RGA	Matriz de ganhos relativos
RMSE	Raiz do erro médio quadrático
RMSprop	*Root mean square propagation*
RN	Rede neuronal
ROC	Característica de operação do receptor
sig	sigmoide
SP_i	*Setpoint* de i
SS	Modelo de espaço de estados
SVD	Decomposição de valores singulares
tanh	Tangente Hiperbólica
TE	Tennessee Eastman

Índice de Figuras

Figura	Descrição	Página
1	Ilustração do fenômeno de Runge	9
2	MVS classificando vetores binários	14
3	Dados de duas classes diferentes não linearmente separáveis no R^2	16
4	Dados de duas classes diferentes não linearmente separáveis no R^3	16
5	Diferença de classificação *one-vs-rest* e *one-vs-one*	18
6	exemplo de neurônio	19
7	Exemplo de rede neuronal	20
8	Treinamento de rede convolucional	21
9	Exemplos de diferentes topologias de redes recorrentes	27
10	Rede *Echo State*	28
11	Operação de convolução	31
12	Operação de *max pooling*	32
13	Exemplo de rede convolucional completa	32
14	Representação gráfica do modelo de espaço de estados contínuo	34
15	Fluxograma do reator de van de Vusse	42
16	Modelo do reator de van de Vusse implementado no Simulink	43
17	Ilustração da saída do reator de van de Vusse	46
18	Todos os valores na base de dados para concentração de B	47
19	Curva C_b vs F/V característica do reator de van de Vusse	47
20	Curva C_b vs F/V característica do reator de van de Vusse, feita com os dados de treinamento	48
21	Autocorrelação parcial de C_b	48
22	Saída do modelo simulado comparado a saída dos modelos lineares	49
23	Saída do modelo simulado comparado a saída da ESN	52
24	RMSE para predição de passo adiante de C_b em relação ao tamanho da camada oculta	53
25	Saída do modelo simulado comparado a saída da MLP-AT. Dados de teste	54
26	Saída do modelo simulado comparado a saída da MLP-AT. Dados de validação	55
27	Curva F/V do modelo simulado, comparado as curvas produzidas pelo ARX e pela MLP-AT	56
28	Todos os modelos de inferência estatística para a produção de B	57
29	Todos os modelos de inferência estatística para C_b	58
30	Fluxograma do processo TE atualizado	60
31	Custo operacional durante toda a operação	69
32	Efeito da falha 1, variação da razão de A/C, na corrente de entrada de A	70
33	Efeito da falha 22, variação da temperatura da corrente de entrada de E, na temperatura do reator	70
34	variância cumulativa explicada por cada componente principal	71
35	Coeficiente de Hotelling em função do tempo, na presença ou ausência de perturbações	72
36	Curva ROC da detecção de falha por PCA.	73
37	Saída da rede neuronal recorrente. Dados de validação e treino	76
38	Curva ROC da rede recorrente	76
39	Saída da rede neuronal convolucional. Dados de validação e treino	79

40 Curva ROC da rede convolucional 79
41 Mapa de calor da acurácia da MVS em relação a C^{MVS} e gamma 82
42 Curva ROC da MVS 83
43 Saída da MVS. Dados de validação e treino 84

Índice de Tabelas

Tabela	Descrição	Página
1	Tipos usuais de *kernels* usados em MVS	17
2	Parâmetros do reator de van de Vusse	41
3	Parâmetros dos controladores	42
4	Hiperparâmetros dos modelos lineares	49
5	Resultados dos modelos lineares	50
6	Hiperparâmetros da ESN	50
7	Resultado da busca em grade pelos melhores parâmetros	51
8	Resultados da ESN	51
9	Hiperparâmetros da MLP-AT	53
10	Resultados da MLP-AT	53
11	Resumo dos resultados para inferência estatística	57
12	Variáveis manipuláveis do Tennessee Eastman	61
13	Variáveis medidas continuamente do Tennessee Eastman	61
14	Variáveis medidas discretamente do Tennessee Eastman	62
15	Variáveis medidas adicionadas ao Tennessee Eastman	63
16	Perturbações programadas no Tennessee Eastman	64
17	Variáveis controladas e manipuladas diretamente pelo esquema de controle	65
18	Ciclos de controle do processo Tennessee Eastman	65
19	Modos de operação do TE	66
20	Detecção de falha através de PCA, dados de treino	72
21	Detecção de falha através de PCA, dados de teste e validação	72
22	Diagnóstico de falha através de FDA dinâmico	75
23	Hiperparâmetros da rede recorrente	75
24	Detecção de falha pela rede recorrente, dados de teste e validação	75
25	Diagnóstico de falha pela rede recorrente	77
26	Hiperparâmetros da rede convolucional	77
27	Detecção de falha pela rede convolucional, dados de teste e validação	78
28	Diagnóstico de falha pela rede convolucional	80
29	Diagnóstico de falha pela rede convolucional dados limpos	81
30	Detecção de falha pela MVS, dados de teste e validação	83
31	Acurácia da classificação em relação ao posição do ponto dentro da margem	84
32	Diagnóstico de falha pela MVS, estratégia *one-vs-rest*	85
33	Diagnóstico de falha por tipo de perturbação	86
34	Sumário do resultado da classificação de falha	87

Índice de Equações

Equação	Descrição	Página
2.4.1	Perda *hinge*	13
2.4.2	Entropia cruzada	13
2.4.3	Máquina de vetor de suporte	13
2.4.4	Algoritmo de classificação	13
2.4.5	Restrição da otimização na forma primal	14
2.4.6	Forma dual da otimização da Máquina de vetor de suporte	14
2.4.7	Resultado da otimização na forma dual	14
2.4.8	Resultado da otimização na forma dual	15
2.4.9	Resultado da otimização na forma dual	15
2.4.10	Resultado da otimização na forma dual	15
2.4.11	Forma final da otimização da MVS na forma dual	15
2.4.12	Restrição da equação primal com variável de folga	15
2.4.13	Otimização da MVS de margem suave na forma primal	15
2.4.14	Otimização da MVS de margem suave na forma dual	16
2.4.15	*Kernel* quadrático	17
2.5.1	Equação de um neurônio da camada escondida	19
2.5.2	Equação de um neurônio da camada de saída	19
2.5.3	Função sigmoidal	20
2.5.4	Função retificadora linear	20
2.5.5	Função tangente hiperbólica	20
2.5.6	Função de base radial	20
2.5.7	Função softmax	20
2.5.8	Equação da atualização do resultado da otimização	22
2.5.9	Definição do passo e direção de busca do gradiente descendente	22
2.5.10	Algoritmo de gradiente descendente	22
2.5.11	Direção de busca do gradiente descendente com momento	22
2.5.12	Direção de busca do gradiente conjugado	23
2.5.13	Cálculo da direção conjugada	23
2.5.14	Busca em linha	23
2.5.15	Direção de busca do método BFGS	23
2.5.16	Diferença entre gradientes	23
2.5.17	Estimação da hessiana	23
2.5.18	Equação da camada recorrente simples	27
2.5.19	Equação da camada de saída da rede neuronal recorrente	27
2.5.20	Formação da entrada com dados atrasados	27
2.5.21	Camada escondida da rede *Echo State*	28
2.5.22	Amortização do espaço interno	28
2.5.23	Equação da saída da rede *Echo State*	28
2.5.24	Portão de esquecimento da LSTM	29
2.5.25	Portão de entrada da LSTM	29
2.5.26	Portão de saída da LSTM	29
2.5.27	Memória interna da LSTM	29
2.5.28	Estado interno da LSTM	29
2.5.29	Portão de atualização da GRU	29
2.5.30	Portão de redefinição da GRU	29
2.5.31	Estado interno da GRU	30
2.5.32	Equação da convolução	30

2.6.1.1	Modelo ARX	33
2.6.2.1	Estado interno do modelo de espaço de estado	33
2.6.2.2	Saída do modelo de espaço de estado	33
2.6.3.1	Transformação ortogonal do PCA	34
2.6.3.2	Decomposição por valores singulares	35
2.6.3.3	Cálculo dos componentes principais	35
2.6.3.4	Estatística T^2	35
2.6.3.5	Limite de 95% de confiabilidade para estatística T^2	35
2.6.4.1	Matriz de espalhamento total	36
2.6.4.2	Matriz de espalhamento interno	36
2.6.4.3	Matriz de espalhamento dentro das classes	36
2.6.4.4	Matriz de espalhamento entre as classes	36
2.6.4.5	Problema de autovalores e autovetores	36
2.6.4.6	Função discriminante do FDA	36
3.1.1	Reação de formação de B no reator de van de Vusse	39
3.1.2	Cinética da formação de B	39
3.1.3	Reação de formação de C no reator de van de Vusse	39
3.1.4	Cinética da formação de C	39
3.1.5	Reação de formação de D no reator de van de Vusse	39
3.1.6	Cinética da formação de D	39
3.1.7	Equação de Arrhenius	39
3.1.8	Equação da acumulação	40
3.1.9	Regra da multiplicação	40
3.1.10	Equação diferencial de C_a	40
3.1.11	Equação diferencial de C_b	40
3.1.12	Equação diferencial de C_c	40
3.1.13	Equação diferencial de C_d	40
3.1.14	Equação diferencial do volume	40
3.1.15	Equação diferencial da temperatura do reator	41
3.1.16	Geração de calor na reação	41
3.1.17	Potência de troca térmica	41
3.1.18	Equação diferencial da temperatura da camisa de resfriamento	41
3.1.19	Filtro exponencial	44
3.1.20	Raiz do erro quadrático médio	45
3.1.21	Raiz do erro quadrático médio normalizado	45
4.1.1	Reação de formação de G	60
4.1.2	Reação de formação de E	60
4.1.3	Primeira reação de formação de F	60
4.1.4	Segunda reação de formação de F	60
4.1.5	Equação para cálculo do *setpoint* do XMEAS(2)	66
4.1.6	Equação para cálculo do *setpoint* do XMEAS(3)	66
4.1.7	Primeira equação do controlador discreto	66
4.1.8	Segunda equação do controlador discreto	66

1) Introdução

1.1) Motivação:

Estamos vivendo a chamada Revolução 4.0 ou 4ª Revolução Industrial. Assim se denomina a tendência atual do aumento da comunicação entre os diversos setores e equipamentos na indústria. Na indústria química, a Revolução 4.0 se manifesta a partir do barateamento de sensores e estocagem de dados; do aumento da instrumentação disponível e da melhora na comunicação entre servidores, plantas e operadores. Isso aumenta muito a quantidade de dados históricos e on-line disponíveis, gerando uma demanda para analise desses dados. Por outro lado, aplicações mais complexas, que requerem um controle mais fino, regulações mais rígidas e uma alta demanda por produtos e processos inovadores, se beneficiam da análise desses dados. (YIN, *et al*. 2015)

Como exemplo do grande volume de dados gerados, vamos supor um tanque simples com medidores para vazão de entrada e nível, e taxa de amostragem de 1 segundo. Cada medida equivale a 4 bytes, padrão para valores de sensores. Por dia deve-se gerar 24*3600*4= 345,6 kB. Em um ano geram-se 126,1 MB de dados. A maioria das plantas armazena terabytes de informação por ano.

Para processar esses dados são utilizados estatística multivariada e uma série de técnicas chamadas de *Machine Learning*, uma parte do tópico de inteligência artificial e do tópico interdisciplinar de *Big Data* (QIN, 2014). O potencial do *Big Data* é incalculável. Já se utiliza para detectar falhas, melhorar o controle de qualidade e aprimorar o planejamento de *supply chain*. Junto com lógica Fuzzy, vem sendo usada para auxiliar em decisões executivas (QIN, 2014).

Machine Learning é tradicionalmente definida como “ algoritmos que permitem que o computador aprenda sem ser explicitamente programado” (SIMON, 2013). Em geral ela é empregada quando a atividade exigida do programa é muito complexa para ser explicitamente programada, por exemplo: identificar se uma foto contém um rosto. Como se define um rosto? Dois olhos, um nariz e uma boca? E se o rosto estiver meio coberto? Como se define um olho? Para essa tarefa é muito mais fácil introduzir dezenas de milhares de fotos com e sem rostos para o algoritmo aprender quando a foto tem um rosto.

Na indústria química, *Machine Learning* já possui um espaço estabelecido desde os anos 1990, com a aplicação, por exemplo, de redes neuronais para fazer aproximação de

processos para fins de controle automático, quimometria e diagnóstico de falhas . (BAUGHMAN & LIU, 1995; DE SOUZA JR, 1993)

O maior desafio atual com *Machine Learning* na indústria química é a qualidade dos dados. Todos os métodos de *Machine Learning* dependem de muitos dados para um aprendizado satisfatório. Ruídos, falhas nos sensores e temporalidade dos dados são fatores a serem levados em conta antes de se aplicar *Machine Learning* a engenharia de processos. Um dos focos desse trabalho é analisar essas limitações e desenvolver meios de superá-las, como utilizar filtros para reduzir ruído e utilizar modelos próprios para série temporais .

Machine Learning traz muitas possibilidades para a engenharia de processos, porém a sua aplicação não é simples. Avaliações do conhecimento gerado, da eficiência da máquina e comportamento em situação anormal são etapas essenciais antes de se aplicar na planta os algoritmos de *Machine Learning*. Provar a sua eficiência e segurança é essencial para maior penetração das técnicas na indústria.

1.2) Objetivo

O objetivo desse trabalho é demonstrar como aplicar os métodos de Machine Learning em processos químicos industriais.

(1) Nesta dissertação aplicamos métodos para superar as limitações de *Machine Learning* em relação a processos químicos, assim facilitando a sua aplicação em processos reais, em um contexto *Big Data*. (2) Novos paradigmas de redes neuronais, como redes profundas, são estudados. (3) Para investigar sua eficácia usamos dois processos *benchmark* em controle e simulação de processos, o reator van de Vusse (VAN DE VUSSE, 1964), como ilustrativo de um processo complexo, e o problema Tennessee Eastman (DOWNS & VOGEL, 1993), como exemplo de uma planta completa. (4) Buscamos aqui modelar a incerteza dos dados para melhorar a sua similaridade com dados de plantas, uma vez que a qualidade dessas informações é essencial para demonstrar a capacidade de *Machine Learning* de operar em casos reais.

1.3) Estrutura da dissertação

Essa dissertação é dividida em 5 partes. Primeiro esta introdução, em que expomos seus objetivos e motivações. O segundo capítulo apresenta a revisão bibliográfica a respeito

de *Machine Learning*, inferência estatística, detecção e diagnóstico de falhas e modelos de aproximação baseados em dados. Os algoritmos de *Machine Learning* são detalhados e suas vantagens e desvantagens expostas. O terceiro capítulo traz o estudo de caso utilizado para avaliar a capacidade de inferência estatística dos modelos: o reator de van de Vusse, com expansões adicionadas para melhorar a aproximação com um reator real. As metodologias de inferência estatística são explicadas e os resultados discutidos. O quarto capítulo expõe o estudo de caso utilizado para investigar a capacidade de detecção e diagnóstico de falhas das abordagens: o modelo Tennessee Eastman expandido por Ricker (BATHELT, *et al.* 2014). As metodologias de detecção e diagnóstico de falhas são explicadas e os resultados discutidos. O quinto capítulo expõe as conclusões do trabalho e discute possíveis linhas de pesquisa futuras.

2) Revisão Bibliográfica

2.1) *Machine Learning*

Existem muitos algoritmos de inteligência artificial: Bayes ingênuo, Árvores de decisão, k-vizinhos mais próximos, regressão linear regularizada etc (HASTIE, *et al.* 2008). Nesse trabalho daremos foco em redes neuronais (RN) (BAUGHMAN & LIU, 1995; MONTAVON, *et al.*, 2012) e máquinas de vetores de suporte (MVS), que são os métodos que têm obtido os melhores resultados atualmente. O ponto em comum entre todos os algoritmos é que os programas possuem uma estrutura básica muito maleável e simples que se adapta aos dados apresentados. No caso de RN, a rede possui diversos pesos que são treinados a fim do algoritmo ser capaz de aproximar a função desejada. Nas MVS, o uso de *kernels* que permitem o algoritmo trabalhar em espaços com maior dimensionalidade e margens suaves que permitem uma regularização inerente ao algoritmo. Esses dois atributos flexibilizam as MVS. (LORENA & CARVALHO, 2007)

Machine Learning está intimamente ligada com *Data Mining*, estatística multivariada e otimização, sendo que a divisão entre os assuntos muitas vezes não é clara (MANNILA, 1996). *Machine Learning* tem sido muito utilizada para resolver problemas tradicionais de estatística multivariada, como regressão, redução de dimensionalidade e clusterização (MONTAVON, *et al.* 2012).

Em processos químicos, *Machine Learning* se enquadra na abordagem baseada em dados. Essa abordagem contrasta com a abordagem tradicional baseada em modelos. A abordagem tradicional tem suas vantagens, ela é capaz de explicar o porquê dos fenômenos e tenta simular situações novas. Porém a abordagem baseada em modelos tem seus problemas. Em geral é difícil criar um modelo a partir das equações básicas. Alguns modelos podem estar errados ou não avaliarem todas as variáveis possíveis que afetam o sistema. Alguns sistemas, como dinâmica molecular, são difíceis de modelar com alto grau de precisão. Muitas vezes sistemas complexos não convergem ou precisam de muito tempo para convergirem para uma solução (BAUGHMAN & LIU, 1995).

A rotina básica para utilização de *Machine Learning* é bem simples (HASTIE, *et al*. 2008). O primeiro passo consiste em selecionar e tratar os dados: remover ruídos usando filtros ou média móveis; descartar ou inferir pontos ausentes e selecionar as características relevantes ao método. A seleção de características é complicada e um tópico de estudo profundo dentro de estatística. Características de menos reduzem a informação disponível ao sistema e características demais atrasam muito o treinamento do sistema, podendp confundir o

algoritmo, aumentando a sua variância. Estatísticos recomendam fazer PCA quando há muitos dados correlacionados. Por exemplo, na análise de espectros infravermelhos de moléculas similares. Porém, PCA ignora a correlação temporal caso os dados sejam séries temporais, mesmo no uso de PCA dinâmico (HASTIE, *et al*. 2008).

No segundo passo, escolhe-se o método. Se houver disponibilidade de programas e tempo, é recomendado testar mais de um e encontrar o que melhor se aplica usando configurações padrões e depois otimizar os hiperparâmetros do modelo até achar o melhor. Dependendo do tamanho da base de dados e da complexidade do algoritmo, o treinamento pode demorar de alguns segundos a semanas. Nem sempre o método mais demorado ou complexo é o melhor, portanto é recomendado testar métodos simples com MVS lineares ou árvores de decisão simples. Existem diversas heurísticas na literatura de como sintonizar hiperparâmetros para diversos métodos, entretanto nenhuma heurística encontrada é universal. (MONTAVON, *et al.* 2012) A sintonia de hiperparâmetros depende da complexidade da relação que se procura encontrar, número de dados disponíveis, características usadas, etc. Sintonizar por experimentação é a única forma garantida de se produzir um resultado ótimo.

"Treinamento" é o termo usado para o ajuste dos parâmetros do modelo aos dados. Em uma MVS, os parâmetros são o vetor de pesos; em uma RN, são os pesos da rede. Geralmente é feita a minimização da função custo C, manipulando os parâmetros do modelo. A função C pode ter diversas formas, dependendo dos métodos e do problema. Para problemas de regressão, a média dos quadrados do erro, definido como a média da diferença entre a saída do algoritmo (y) e os dados de treinamento (t) ao quadrado. Para problemas de classificação, a função custo entropia cruzada, $t*\ln(y)+(1-t)*\ln(1-y)$ tem obtido ótimos resultados(BAUGHMAN & LIU, 1995).

Depois de treinado o modelo, é necessário validá-lo. Métodos mais flexíveis sempre convergirão para uma função custo menor, mas isso não quer dizer que generalizem melhor para um conjunto de dados diferente dos usados para treinamento. Em *Machine Learning*, a base de dados é separada em 3 categorias: dados de treino, de teste e de validação. Os dados de treino são usados para treinar o modelo, os dados de teste são usados para dar a ideia de como o modelo trabalha fora dos dados de treinamento. Em geral o ajuste dos hiperparâmetros é realizado até encontrar o menor erro nos dados de teste. Quando o mínimo for encontrado, testa-se o algoritmo contra os dados de validação. Esses dados de validação dão a verdadeira capacidade de generalização do algoritmo, pois há o risco dos hiperparâmetros serem sobreajustados para o conjunto dos dados de teste. Existem várias

formas de fazer validação. As mais usadas são amostragem randômica e k-fold (HASTIE, *et al*. 2008).

Amostragem randômica consiste em separar os dados aleatoriamente nos três conjuntos. É recomendada quando os dados não são sequenciais, bem variados e quando o treinamento é demorado.

K-fold se baseia em separar primeiro os dados de validação do conjunto. Depois de separados os dados de validação, separam-se os dados em um número k de conjuntos, chamados de *folds* em inglês. São usados k-1 conjuntos de treino e 1 para teste. Depois se refaz o treino usando outro conjunto para teste. Faz-se esse procedimento k vezes, até que todos os dados tenham sido utilizados. O algoritmo resultante pode ser aquele com melhor performance ou a média de todos os algoritmos. É recomendado caso o treinamento seja rápido e não haja tantos dados. (HASTIE, *et al*., 2008)

O treinamento muitas vezes exige algoritmos avançados de otimização e uma boa análise matemática para garantir a convergência e velocidade de obtenção dos resultados. Isso é especialmente verdade para as RN, que dependem muito da estimativa inicial dos pesos e de algoritmos que não se prendam a mínimos locais (LECUN, *et al.* 2012). Existem 3 tipos de problemas típicos em *Machine Learning*: aprendizado supervisionado, aprendizado não supervisionado e aprendizado reforçado (HASTIE, *et al.* 2008), conforme detalhado a seguir:

- Aprendizado supervisionado é o que tem sido mais utilizado no mundo. Esse tipo de problema se baseia em relacionar uma entrada X com uma saída Y, a partir de dados fornecidos pelo usuário. A formulação de problemas nesse caso é bem direta, sem precisar de uma formalização matemática rígida ou de conhecimentos estatísticos profundos. É a preferida entre os engenheiros e operadores. Aprendizado supervisionado é divido também em problemas de classificação, onde a saída é um booleano, e de regressão, em que a saída é um valor numérico. Problemas típicos são: detecção de falha (classificação), em que a saída é verdadeiro ou falso, 1 ou 0, para a pergunta, “há falha?”; simulação da saída de um sistema é um problema típico de regressão, onde tenta mapear uma saída numérica Ya partir de uma entrada X.

- Aprendizado não supervisionado é quando apenas a entrada X é fornecida ao sistema. Entre os cientistas da computação é o tipo de aprendizado mais estudado, por ser mais próximo ao realizado pelo cérebro humano. Problemas típicos são clusterização, redução de dimensionalidade e remoção de ruídos. Ele exige uma formulação mais

cuidadosa do problema, uma vez que o objetivo nem sempre é claro. Uma vantagem interessante do aprendizado não supervisionado é que ele pode ser automatizado a partir da captura de dados em tempo real. Em métodos supervisionados é necessário ter certeza de que um X corresponde a um Y, já que muitas vezes X e Y são avaliados por sistemas diferentes e analisar se o dado adquirido não é um ponto excepcional ou repetido. Se há apenas o X essa etapa manual é dispensada.

- Em aprendizado reforçado o programa interage com um sistema dinâmico e tenta alcançar um objetivo. Os famosos carros que se dirigem sozinhos são um tipo de aplicação de aprendizado reforçado. Aprendizado reforçado é uma generalização do campo da programação dinâmica, que é estudada a um tempo em processos químicos para um controle mais fino de processos variantes no tempo. (LEWIS & VRABIE, 2009; LEE & LEE, 2005).

A eficiência dos métodos de *Machine Learning* está intrinsecamente relacionada com a qualidade e quantidade dos dados disponíveis. Uma vez que todo o conhecimento do algoritmo provém dos dados, são necessários muitos exemplos para ele aprender as relações entre os dados. Por isso é necessário garantir uma boa variabilidade dos dados, assegurar que se tenha uma boa quantidade de informações diferentes. A aplicação de *Machine Learning* em processos químicos só é possível devido à imensa quantidade de dados históricos disponíveis.

Não se sabe qual é o tamanho ideal de um banco de dados para ser utilizada para treinamento de algoritmos. Geralmente cientistas de dados dizem: quanto mais melhor. Porém a qualidade e variabilidade dos dados influenciam. É uma prática comum duplicar dados raros e difíceis de serem corretamente classificados e descartar dados redundantes, muito comuns e facilmente classificáveis (HASTIE, *et al*. 2008).

Existem muitas vantagens em aplicar *Machine Learning* se comparado a métodos estatísticos tradicionais. Como não existem suposições sobre a estrutura do espaço de estados, ideias erradas ou modelagem imprecisa do processo são ignoradas. Os algoritmos de *Machine Learning* também são capazes de capturar não linearidades do processo não representadas por modelos lineares. Não são necessárias análise das dinâmicas internas do processo e estimação de parâmetros das equações, que muitas vezes são aspectos desconhecidos do processo.

Além disso, a abordagem baseada em dados permite detectar padrões que não são claramente descobertos por abordagens tradicionais como árvores de falha. Isso é

especialmente verdadeiro em processos com reciclos e múltiplos laços de controle mal ajustados (SHU, *et al*. 2016).

Entretanto, *Machine Learning* possui desvantagens. A abordagem baseada em dados não explica a razão dos fenômenos, pode, por exemplo, demonstrar que o grupamento amina aumenta a solubilidade da molécula A, mas não explica o porquê. Apenas abordagens fenomenológicas são capazes de explicar os fenômenos ocorrendo. *Machine Learning* também é ruim em extrapolação de dados (BAUGHMAN & LIU, 1995).

Outra questão é que a natureza "caixa-preta" de muitos modelos deixa engenheiros receosos por aplicar os modelos, pois fica difícil saber exatamente o que modelo aprendeu, então não se sabe como ele se comportará em uma situação anormal. Na indústria química isso pode resultar em acidentes graves, com um custo financeiro e humano alto (SHU, *et al.* 2016).

2.2) Desafios à aplicação de *Machine Learning* a processos químicos

Aplicar *Machine Learning* a processos industriais é bem diferente de pesquisar sobre *Machine Learning*. A maioria da pesquisa acadêmica busca utilizar problemas definidos com bases de dados estabelecidos, como o MNIST (grande base de dados de dígitos manuscritos), e endereça o refinamento dos métodos pesquisados. Industrialmente há um foco maior na formulação dos problemas, coleta e tratamento dos dados, análise do resultado do algoritmo e aplicação no processo (VENKATASUBRAMANIAN, 2009).

O uso de dados em processos químicos não é novidade. Reconciliação de dados é essencial para controle de processos em tempo real em processos modernos. Filtro de Kalman é utilizado há quase meio século e modelos de série temporais como Box-Jenkins e ARX são muito empregados para controle e modelagem de processos. A principal diferença desses métodos para *Machine Learning* é que eles exigem um conhecimento a priori do processo e dos balanços fundamentais para funcionarem (SHU, *et al.* 2016).

Existem muitos desafios relativos à aplicação de *Machine Learning* a processos químicos. A maioria é relativo aos dados. Dificilmente o conhecimento a priori do processo é introduzido aos algoritmos de *Machine Learning*, então é necessário ter uma grande quantidade de dados. Os problemas usuais com os dados dos sensores, como desvio e ruído, reduzem a aplicabilidade do modelo (BAUGHMAN & LIU, 1995). Este trabalho pretende investigar esses problemas e a robustez dos métodos de *Machine Learning*.

Um grande desafio é como inserir nos algoritmos que os dados são variantes com o tempo. Isso é uma característica importantíssima dos dados de processos que a maioria dos métodos estatísticos, como PCA, não consegue captar. Existem certas estratégias, como usar janelas móveis, para introduzir dados atrasados. Porém esses métodos tratam os dados atrasados como variáveis independentes, isso é, ele não assume que x(t) é relacionado a x(t-1). o que é uma suposição que não é verdadeira (HASTIE, et al., 2008). Alguns algoritmos como redes neuronais recorrentes conseguem entender os dados como variantes no tempo.

Outro método utilizado é empregar apenas dados em estado estacionário e estimar uma dinâmica de primeira ou segunda ordem usando conhecimento dos processos. Essa metodologia é complicada, pois dificilmente processos permanecem plenamente estacionários.

Um problema recorrente de métodos estatísticos é o sobreajuste. Sobreajuste ocorre quando o número de parâmetros a ser otimizado é maior do que o necessário e o algoritmo acaba decorando os dados de treinamento e não a relação entre os dados. Um algoritmo sobreajustado generaliza muito mal para fora da região de treinamento. Esse resultado é relacionado com o fenômeno de Runge , que demonstra que o maior grau de liberdade de um método tende a melhorar o erro de treino mas não melhora necessariamente a aproximação real da relação interna entre os dados, como ilustrado na Figura 1. Nela, apesar do erro ser menor na regressão polinomial, espera-se que a função linear generalize melhor. De uma maneira geral, é impossível saber qual o número ótimo de parâmetros a serem ajustados; tipicamente esse número é estimado empiricamente (HASTIE, *et al*. 2008).

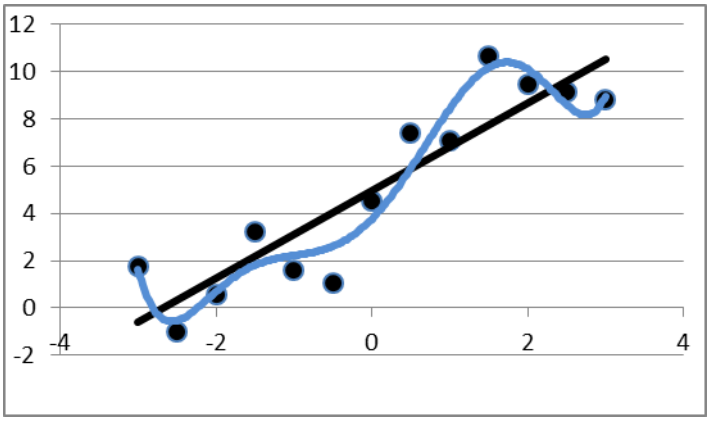

Figura 1: Ilustração do fenômeno de Runge. Dados ruidosos são ajustados para uma função linear (linha preta) e outra polinomial de grau 6 (linha azul).

Para compensar o sobreajuste, geralmente se usa regularização. Regularização é uma restrição á complexidade do algoritmo de forma a compensar os graus de liberdade extras

fornecidos pelo excesso de parâmetros do modelo. Existem diversas formas de regularização e muitas vezes várias delas são usadas em conjunto. Tipicamente a regularização busca limitar o tamanho dos parâmetro dos algoritmos, criando uma penalidade para parâmetros muito grandes, assim garantindo uma função mais suave. Alguns métodos de regularização removem coeficientes.

Um desafio que aparece em todos os métodos de aproximação de função em processos químicos é a extrapolação dos resultados. A maioria dos métodos é excelente para interpolação, mas funciona muito mal fora do estado de espaços da base de dados usados em seu treinamento. É necessário que seja alimentado ao algoritmo dados que ocupem o máximo do espaço de estados possível. Em processos industriais isso é arriscado por restrições de segurança e economia. Essa é a maior dificuldade em aplicar *Machine Learning* em processos reais, pois muitos engenheiros e operadores possuem receio de que os métodos falhem em situações imprevistas (SHU, *et al.*, 2016).

Existem métodos que mitigam esse problema. Um método comum usado em controle é utilizar um termo de compensação de erro do modelo. Outro método é inserir dados de plantas similares que estejam disponíveis. É necessário compensar as diferenças entre as plantas, então esse método exige cuidado. Adicionar dados de simulação computacional ou proveniente de interpolações é outra técnica disponível (LECUN, *et al.* 2012; HASTIE, *et al*. 2008).

Uma questão é a rigidez das entradas. *Machine Learning* depende de ter sempre as mesmas entradas, então troca de equipamentos e falhas de sensores podem exigir um retreinamento do algoritmo (NEUNEIER & ZIMMERMANN, 2012). Isso é muito perigoso em processos industriais pois pode inutilizar o algoritmo, obrigando o uso de sistemas secundários e a transição raramente é suave (ALBERTOS & SALA, 2004).

Esse problema também é comum em controle preditivo e existem métodos para contornar isso. Um método simples, porém pouco recomendado, é utilizar o valor histórico médio para a entrada ausente, assim o algoritmo recebe um sinal relativamente razoável. Um método melhor é ter um sensor virtual, que estima uma variável do processo a partir de outras, pronto para ser usado caso haja perda de um sinal específico, estimando assim o sinal perdido. Redes neuronais têm sido muito utilizadas para a criação de sensores virtuais por ser um aproximador universal (MONTAVON, *et al.* 2012). Os maiores usos atualmente de *Machine Learning* em processos químicos são detecção e diagnóstico de falha e inferência estatística.

Não existem muitos trabalhos no contexto de *Big Data* em processos químicos. A maioria das aplicações de *Big Data* são feitas pelas empresas que possuem esses dados e elas

não costumam compartilhar resultados e dados por questões de propriedade intelectual e sigilo industrial (QIN, 2014). SHANG, e*t al.*(2014) utiliza uma rede profunda como *soft sensor*, porém a base de dados utilizada contêm apenas 251 pontos. GAO, *et al*. (2014) utilizou uma base de dados maior, 6364 pontos para treinar uma rede *deep belief* que modela o regime de operação de uma refinaria. Nenhum trabalho encontrado lida com bases de dados temporais do tamanho que nós utilizamos nesse projeto, 200 dias de dados.

2.3) Inferência estatística e detecção e diagnóstico de falha

Inferência estatística é, em um contexto mais geral, estimar qual a distribuição probabilística de um certo dado. No contexto de processos químicos, é usada para estimar o valor de certas variáveis em um determinado espaço de tempo. Inferência estatística e identificação de processo são similares. Muitos dados do processo, como concentrações, só são medidos em análises laboratoriais, podendo demorar horas até se obter o valor real no instante de tempo de retirada da amostra.

Exemplos de usos de inferência estatística são *soft sensors* e controle preditivo baseado em modelo. *Soft sensors* são algoritmos que estimam em tempo real certas variáveis do processo que não podem ser obtidas automaticamente, a partir de dados que são obtidos por sensores do processo como vazões, temperaturas etc. Esses sensores virtuais são muito úteis para controle de concentrações, que são difíceis de medir (YAN, *et al.* 2004).

Em controle preditivo, muitos modelos químicos feitos a partir de modelagem matemática são rígidos e podem ter milhares de equações algébrico-diferenciais. Alguns modelos dependem de parâmetros desconhecidos ou cuja estimação é muito imprecisa. Integrar essas equações pode demorar horas, de forma que quando uma ação de controle é planejada já é tarde demais. Inferência estatística é usada para criar modelos simplificados a partir desses modelos complexos cuja ordem de cálculo seja bem menor.

Inferência estatística é um problema de regressão, em que há saídas e entradas numéricas. Qualquer método de regressão pode ser usado, como regressão linear, redes neuronais, máquinas de vetor de suporte, etc. (GARCIA, *et al.* 1989). Uma vantagem dos métodos de *Machine Learning* é que eles lidam bem com as não linearidades, o caso da maioria dos processos químicos.

Detecção e diagnóstico de falha (DDF) é saber quando ocorreu uma falha no processo e qual falha ocorreu respectivamente. Uma falha é o desvio não autorizado de uma ou mais

variáveis do sistema. Tipos de falhas comuns na indústria química são: escalonamento de trocadores de calor, envenenamento do catalisador, vazamentos etc. Essa é uma área relativamente nova na engenharia química (SHU, *et al.* 2016).

O maior objetivo de detecção e diagnóstico de falhas é garantir uma operação que satisfaça as especificações de desempenho, incluindo restrições ambientais, econômicas e de segurança. Essa área de estudo tem se mostrado cada vez mais relevante dado o endurecimento da legislação ambiental e trabalhista e o aumento da complexidade das plantas, é considerado difícil um operador ficar atento a todas as variáveis do processo. Um bom DDF reduz os custos de operação e o tempo ocioso da planta, melhora a segurança e a qualidade dos produtos (CHIANG, *et al.* 2001).

Tradicionalmente são usados métodos de análise estatística multivariada ou modelagem matemática para DDF. Em análise multivariada são observadas as características dos dados, como média e desvio-padrão e analisado se há um desvio relevante dos dados históricos do sistema. Usando modelagem matemática a saída do processo é comparada com a resposta esperada de um modelo rigoroso. Caso haja um desvio forte do modelo em relação ao processo é analisada a causa do desvio, podendo se usar simulações para estimar a causa (CHIANG *et al.* 2001).

Detecção e diagnóstico de falhas é um problema de classificação, onde há entradas numéricas e saídas categóricas. Nos últimos anos, o número de algoritmos disponíveis para classificação aumentou muito, porém a maioria é usado por cientistas de dados e estatísticos, não fazendo parte do repertório de um engenheiro químico comum. (VENKATASUBRAMANIAN, 2009). Essas técnicas têm-se mostrado capazes de aprender padrões inesperados em desafios de visão computacional e análise. Nesta dissertação investigaremos a utilização dessas técnicas para identificação das perturbações do problema Tennessee Eastman, que são falhas pré-programadas no processo (DOWNS & VOGEL, 1993). No contexto do problema Tennessee Eastman usaremos o termo falhas e perturbações como sinônimos.

2.4) Máquinas de Vetores de Suportes

Máquinas de vetores de suporte procuram construir um modelo que separa duas categorias com o máximo de margem possível. O método é visualizável na Figura 2. A linha sólida é um hiperplano que separa as duas categorias, preta e branca, e as duas linhas

pontilhadas são as margens. A margem é a maior distância entre um hiperplano e um exemplo separado. Esse objetivo é conhecido como perda de articulação, ou perda *Hinge*, equação 2.4.1. MVS lineares são análogos a regressão logística, só que regressão logística minimiza a entropia cruzada, equação 2.4.2 (LORENA &. CARVALHO, 2007).

$$Hinge = max(0,1 - y * t) \vee t = -1\ ou\ 1 \qquad \text{Eq 2.4.1}$$

$$entropia\ cruzada = -t * log(y) + (1 - t) * log(1 - y) \vee t = 0\ ou\ 1 \qquad \text{Eq 2.4.2}$$

Sendo y a saída do modelo e t a classe real dos dados.

Uma vantagem das MVS linear é que a classificação é visualizável. Matematicamente uma MVS divide os dados em hiperplanos, e a menor distância entre um hiperplano e um ponto (x,y) é a margem. Um MVS com maior margem é mais robusto. O MVS é definida pela equação 2.4.3

$$f(x) = w * x - b = 0 \qquad \text{Eq. 2.4.3}$$

em que w é o vetor normal do hiperplano, x é o ponto no hiperplano e b é o desvio da origem do plano, mostrado melhor na Figura 2. O algoritmo para classificação é a equação 2.4.4 (VENKATASUBRAMANIAN, 2009)

$$g(x) = sinal\big(f(x)\big) \qquad \text{Eq 2.4.4}$$

A Figura 2 ilustra um exemplo de separação feita por MVS. A linha sólida é o vetor normal, as linhas pontilhadas são as margens, o espaço entre as margens é proporcional ao inverso de W, os parâmetros do modelo.

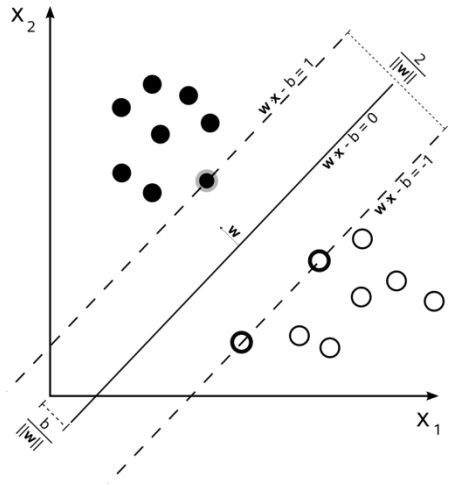

Figura 2: Máquina de vetor de suporte classificando duas classes (LORENA & CARVALHO, 2007).

O algoritmo para o cálculo de uma MVS é uma minimização de $|w|^2$ tal que para todos os pontos (x_i, y_i), o alvo esteja fora ou na margem, dada pela equação 2.4.5

$$\min_{w} |w|^2 \; restrito \; a \; t_i * f(x_i) \geq 1 \qquad \text{Eq 2.4.5}$$

Essa forma de cálculo é chamada de forma primal e é menos eficiente que a forma dual. Esse problema de otimização é quadrático, e possui um único mínimo global. Problemas desse tipo podem ser solucionados com a introdução de uma função Lagrangiana, que engloba as restrições à função objetivo, associadas a parâmetros denominados multiplicadores de Lagrange αi. Essa forma de resolução usando multiplicadores de Lagrange é conhecido como forma dual e é mais eficiente (LORENA &. CARVALHO, 2007).

$$L(w, b, \lambda) = \frac{||w||^2}{2} - \sum_{i=1}^{n} \lambda_i * (t_i * (w * x_i + b) - 1) \qquad \text{Eq. 2.4.6}$$

Maximizar a Lagrangiana implica maximizar λ e minimizar w e b. Encontra-se um ponto de sela tal que:

$$\sum_{i}^{n=1} \lambda_i * t_i = 0 \qquad \text{Eq. 2.4.7}$$

$$\sum_{i}^{n=1} \lambda_i * t_i * x_i = w \qquad \text{Eq. 2.4.8}$$

O resultado dessas expressões é

$$\frac{\partial L}{\partial b} = 0 \qquad \text{Eq 2.4.9}$$

$$\frac{\partial L}{\partial w} = 0 \qquad \text{Eq 2.4.10}$$

E o resultado final é o seguinte problema de otimização

$$\max_{\lambda} \sum_{i=1}^{n} \lambda_i - \sum_{i,j=1}^{n} \lambda_i * \lambda_j * t_j * t_i * (x_i * x_j) \; restrito \; a \; \lambda_i \geq 0 \; e \sum_{i=1}^{n} \lambda_i * t_i = 0 \qquad \text{Eq 2.4.11}$$

A solução é que λ são zero, exceto nos pontos nas margens do hiperplano. Esses pontos são os vetores de suporte que nomeiam o algoritmo. A equação dual tem a vantagem de trabalhar com restrições simples e apenas com o produto interno dos dados, o que é importante na técnica "*kernel trick*", que será apresentado a seguir. Essa forma MVS é conhecida como MVS de margem rígida. Ela é muito pouco utilizada para aplicações reais pois geralmente os dados possuem ruídos, *outliers* e/ou não são linearmente separáveis. As MVS mais utilizadas são as de margem suave (LORENA & CARVALHO, 2007).

MVS de margem suave são criadas introduzindo variáveis de folga e regularização, ξ_i e C^{MVS} respectivamente. A variável de folga transforma a restrição da equação primal na equação 2.4.12:

$$t_i * f(x_i) \geq 1 - \xi_i \qquad \text{Eq 2.4.12}$$

Suavizando as margens permite-se que o modelo funcione mesmo com erros de classificação e que haja pontos entre as margens. Se $\xi_i > 1$, ξ_i representa um erro de treino. Para minimizar o erro de treino a função objetivo se torna a equação 2.4.13:

$$\min_{w} |w|^2 + C^{MVS} * \sum_{i=1}^{n} \xi_i \; restrito \; a \; t_i * f(x_i) \geq 1 - \xi_i \qquad \text{Eq 2.4.13}$$

Na forma dual o problema é reescrito na equação 2.4.14

$$\max_{\lambda} \sum_{i=1}^{n} \lambda_i - \sum_{i,j=1}^{n} \lambda_i * \lambda_j * t_j * t_i * (x_i * x_j) \; restrito \; a \; C^{MVS} \geq \lambda_i \geq 0 \; e \sum_{i=1}^{n} \lambda_i * t_i = 0 \qquad \text{Eq 2.4.14}$$

Esse problema tem a vantagem de ser convexo, isto é, possui apenas um mínimo global. Isso facilita o treinamento da MVS, evitando os problemas normalmente encontrados por outros métodos, principalmente RN (LORENA & CARVALHO, 2007).

Para lidar com dados não lineares, usa-se uma técnica conhecida como *kernel trick*. O *kernel trick* consiste em mapear o conjunto de treinamento para um espaço de maior dimensão, onde ele pode ser separado nesse espaço (LORENA & CARVALHO, 2007). Por exemplo, o mapeamento polinomial de grau 2, com κ=0 e δ=1 um espaço X=[x1, x2] se torna $\Phi(X)=[x1^2, \sqrt{2}*x1*x2, x2^2]$. A transformação pode ser vista nas Figuras 3-4, onde o espaço passa de R^2 para R^3.

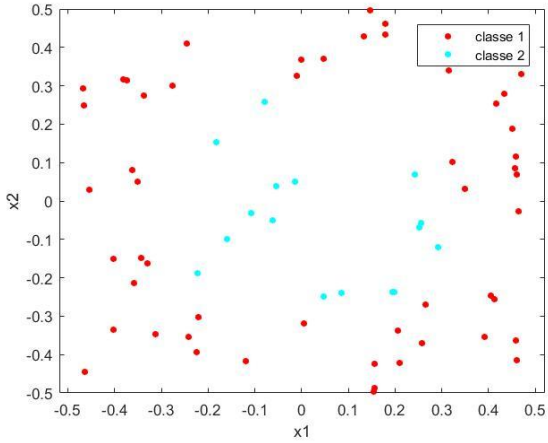

Figura 3: Dados de duas classes diferentes não-linearmente separáveis no R^2.

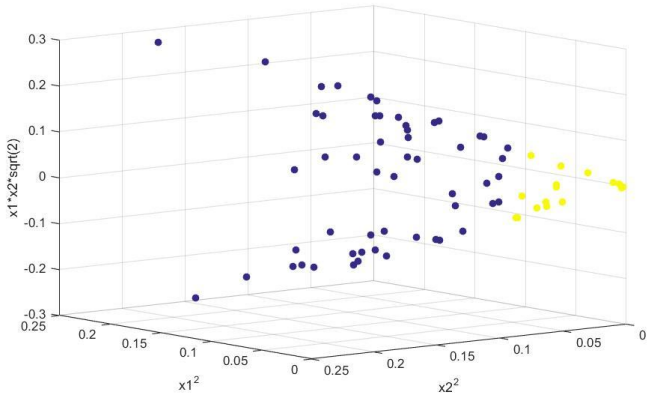

Figura 4: Dados de duas classes diferentes linearmente separáveis no R^3.

Como pode ser visto na Figura 4, no R^3 os dados são linearmente separáveis. Um *kernel* é uma função que recebe o espaço original X e calcula o produto escalar das variáveis depois do mapeamento. No exemplo anterior o *kernel* é:

$$K(x_i, x_j) = (x_{1i}^2, \sqrt{2} * x_{1i} * x_{2i}, x_{2i}^2) * (x_{1j}^2, \sqrt{2} * x_{1j} * x_{2j}, x_{2j}^2) = (x_i * x_J)^2 \qquad \text{Eq 2.4.15}$$

Kernels comuns incluem sigmoidal, polinomial e gaussiano, também conhecido como função de base radial. Eles são definidos na Tabela 1:

Tabela 1: Tipos usuais de *kernel* usados em MVS. (LORENA, A. & CARVALHO, 2007)

Tipo de *Kernel*	**Função $K(x_i,x_j)$**	**Hiperparâmetros**
Polinomial	$(\delta(x_i * x_j) + k)^d$	δ, k e d
Gaussiano	$\exp\left(-\gamma * \left\|\left\|x_i - x_j\right\|\right\|^2\right)$	γ
Sigmoidal	$tanh(\delta(x_i * x_j\ ) + k)$	δ e k

É possível usar a função *kernel* sem saber o mapeamento explicito, isso simplifica muito o cálculo. Uma das maiores dificuldades em utilizar MVS é a necessidade de usar dados completamente classificados. A classificação é limitada a apenas 2 classes. Às vezes os parâmetros são difíceis de interpretar, principalmente quando se usa o *kernel trick*. A escolha dos parâmetros ótimos geralmente é feita a partir de experimentação (HASTIE, *et al*., 2008).

Máquinas de vetores de suporte são classificadores binários por natureza. Para expandir a sua capacidade de classificação para casos multiclasses são usadas várias estratégias. As estratégias mais comuns são: "*one-vs-rest*", "*one-vs-one*" e Crammer-Singer. A estratégia *one-vs-rest*, (um contra os outros) se baseia em treinar n classificadores, sendo n o número de classes, cada classificador tentando classificar uma classe contra todas as outras. Para fazer a classificação, o X que se deseja classificar é analisado por todas as máquinas e a máquina com maior distância do hiperplano, do lado da classe analisada, é escolhida. A distância do hiperplano funciona como um indicador de confiança, pois quanto mais longe o vetor X estiver do hiperplano maior a chance de classificação correta. Essa estratégia é rápida quando se há muitas classes, porém é muito sensível ao balanceamento da base de dados, pois a classe "outros" possui um volume muito maior (HASTIE, *et al.* 2008).

A estratégia *one-vs-one* (um contra um) treina n*(n-1)/2 classificadores para n classes. Cada classificador aprende a diferenciar entre duas classes. Depois de treinados, o vetor que se deseja classificar é analisado por todos os modelos. A classe mais votada é a escolhida. Essa estratégia em geral é mais lenta, pois o número de classificadores aumenta quadraticamente em relação ao número de classes, porém é menos sensível ao balanceamento da base de dados. A Figura 5 ilustra a diferença entre classificação *one-vs-one* de *one-vs-rest*. (HASTIE, *et al.* 2008)

Existem expansões de MVS para o caso multiclasse, como o método Crammer-Singer. Apesar de parecer a escolha mais lógica aplicar um método desse tipo os resultados em geral são menos precisos, consomem mais memória e o treinamento pode ser mais devagar que para as outras estratégias (HASTIE, *et al.* 2008).

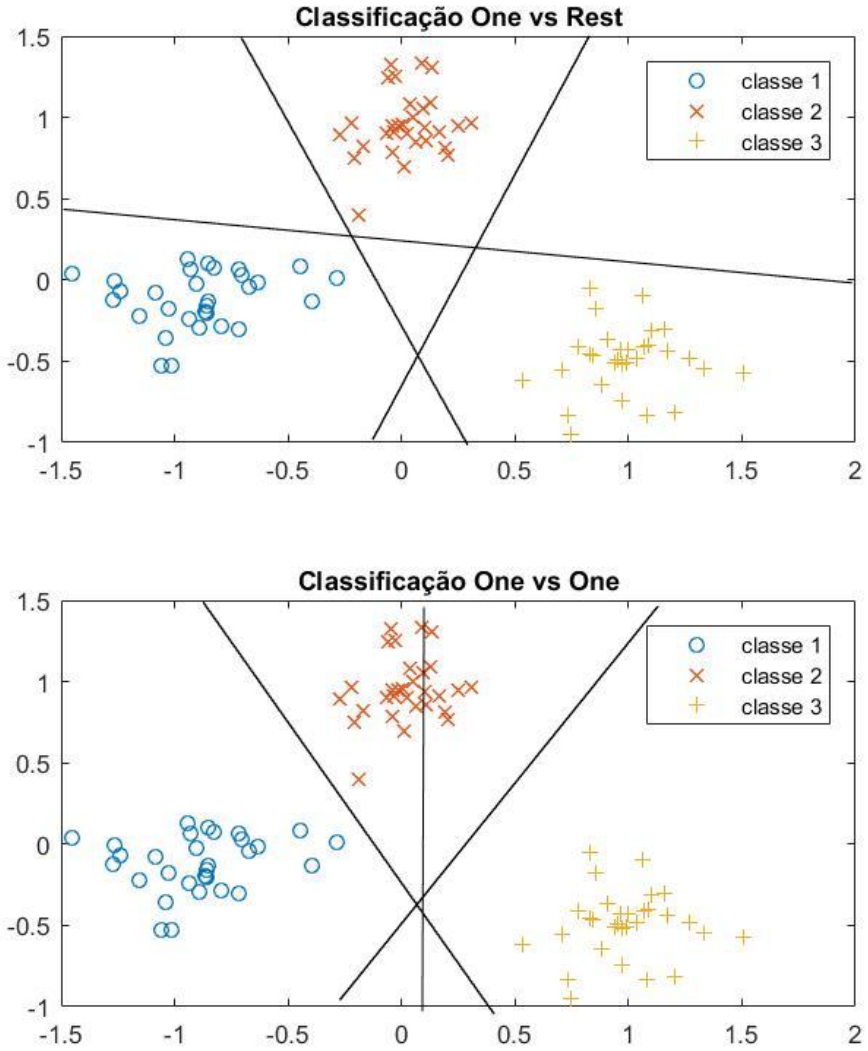

Figura 5: Diferença entre classificação *one-vs-rest* e *one-vs-one*.

2.5) Redes Neuronais

As redes neuronais são um algoritmo baseado em reproduzir como o cérebro humano funciona, usando vários "neurônios" conectados uns aos outros. Cada neurônio realiza uma soma ponderada de diversas entradas, o resultado é comparado a um limite interno de ativação (ou bias) e esse resultado passa por uma função de ativação, como ilustrado na Figura 6. A saída dessa função de ativação é usada como entrada para outros neurônios. A rede tenta aprender a função que relaciona as entradas e as saídas dos dados fornecidos ajustando seus pesos e biases (BAUGHMAN & LIU, 1995).

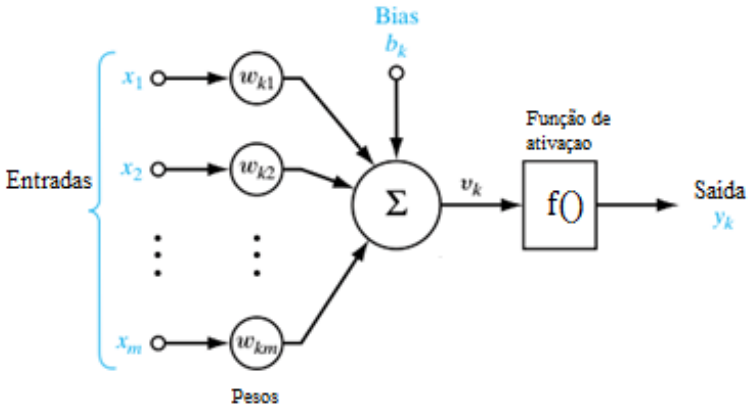

Figura 6: Exemplo de neurônio (MONTAVON, *et al.* 2012).

A equação que dita a saída de uma rede simples de duas camadas é

$$v_i = f(w_i * x + b_i) \qquad \text{Eq 2.5.1}$$

$$y_j = w_j^{out} * v + b_j^{out} \qquad \text{Eq 2.5.2}$$

Onde x são as entradas da rede, w_i é o vetor de pesos do neurônio i, b_i é o bias do neurônio i, y_j é a saída j da rede, w^{out}_j é o vetor de pesos da saída j e b^{out}_j é o bias da saída j; f é a função não linear aplicada a cada elemento. Camadas ocultas podem ser adicionadas replicando a primeira equação, substituindo x por v e v por um v2.

Funções de ativações comuns para camadas ocultas são: sigmoidal (sig), retificadora linear (ReLU), tangente hiperbólica (tanh) e função de base radial (RBF), indicadas nas equações 2.5.3-6. Para classificação multiclasse é usada a ativação softmax, equação 2.5.7 na camada de saída.

$$sig(x) = \frac{1}{1 + exp(-x)} \qquad \text{Eq 2.5.3}$$

$$ReLU(x) = max(0, x) \qquad \text{Eq 2.5.4}$$

$$tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \qquad \text{Eq 2.5.5}$$

$$RBF(x) = exp(-x^2) \qquad \text{Eq 2.5.6}$$

$$softmax(x) = \frac{e^{x_i}}{\sum_i e^{x_i}} \qquad \text{Eq 2.5.7}$$

Usualmente uma rede consiste de várias camadas de neurônios dispostas uma em frente da outra, como exibido na Figura 7. Os dados X entram na primeira camada e vão ativando os neurônios camada por camada até sair na última camada. A primeira camada é conhecida como camada de entrada, e geralmente contém neurônios com a função de transferência identidade. A última camada é chamada de camada de saída, e geralmente possui uma função de transferência linear, para problemas de regressão, função de transferência sigmoidal, para problemas de classificação binária e função de ativação Softmax, equação 2.5.7, para classificação multiclasse (BAUGHMAN.& LIU,1995).

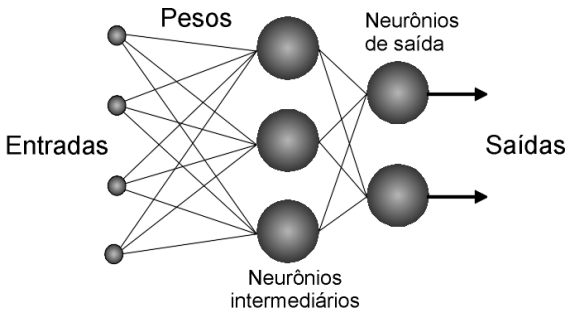

Figura 7: Exemplo de rede neuronal. (TAFNER, 1998)

Redes neuronais têm a vantagem de serem aproximadores universais. Dado o número apropriado de neurônios e dados de treinamento uma rede neuronal de uma camada oculta consegue aproximar qualquer função (BAUGHMAN & LIU,1995). Redes neuronais possuem muita flexibilidade de forma, podendo ser adaptada ao tipo de dado utilizado. Por exemplo: redes convolucionais, que são redes que funcionam analogamente ao olho humano e serão descritas com mais detalhes adiante, são indicadas para tratar dados bidimensionais como imagens, redes recorrentes tratam dados sequenciais como texto e séries temporais. Isso tem

feito as redes neuronais serem o algoritmo de *Machine Learning* mais pesquisado no mundo (MONTAVON, *et al.* 2012).

As redes, no entanto, impõem muitos desafios. Elas dependem de muitos dados e não há garantias de que o treinamento resulte na melhor rede possível. Também não há garantias de que a rede generalize bem. As características escolhidas também são um fator importante, ainda mais porque o treinamento é consideravelmente mais demorado que os outros algoritmos (BAUGHMAN & LIU, 1995; NEUNEIER & ZIMMERMANN, 2012).

As redes multicamadas típicas, também chamadas de MLP (*MultiLayer Perceptrons*, ou Perceptrons Multicamadas) são treinadas usando um algoritmo chamado de backpropagation, geralmente usado em conjunto com gradiente descendente ou outro algoritmo de otimização. O método primeiro calcula a saída da rede e com essa saída calcula o gradiente da função objetivo, geralmente soma quadrática dos erros ou entropia cruzada, em relação aos pesos e os biases. Subtrai-se esse gradiente multiplicado pela taxa de aprendizagem dos pesos antigos para corrigir os pesos entre a camada de saída e a última camada escondida e a correção vai progredindo para trás até a camada de entrada. Depois se inicia o processo de novo até os critérios de parada sejam alcançados. Cada ciclo de processo é chamado época. Os erros de treino e teste de cada época geralmente são salvos e exibidos em um gráfico para verificar sobreajuste, como na Figura 8. Para se usar backpropagation as funções de ativação precisam ser diferenciáveis (BAUGHMAN & LIU, 1995).

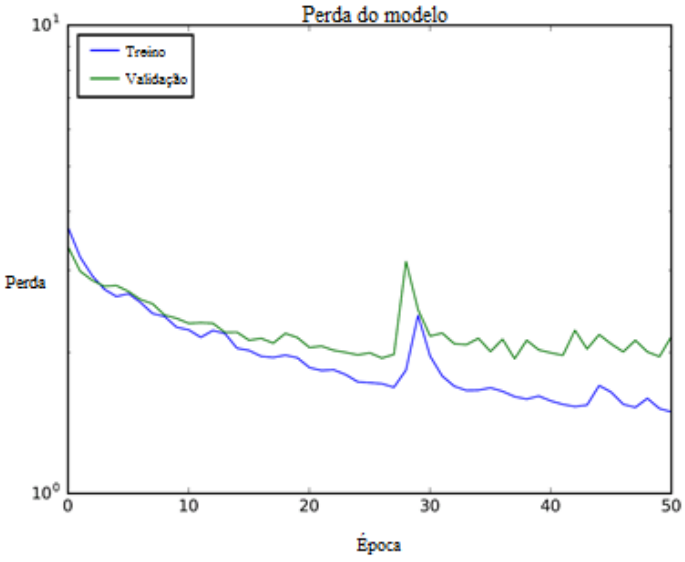

Figura 8: Treinamento da rede neuronal para diagnóstico de falha.

Backpropagation simples raramente é utilizado, pois ele demora muito e tende a ficar preso em mínimos locais. Existem diversos algoritmos de treino mais refinados usados em

redes neuronais. Os mais relevantes são: gradiente descendente com momento, gradiente conjugado e BFGS (Broyden–Fletcher–Goldfarb–Shanno), cada um com diversas variações. Não existe uma heurística de que método escolher, depende basicamente do problema, da base de dados e da arquitetura da rede. Em geral é recomendado testar mais de um, caso a rede seja pequena, com menos de 1000 parâmetros. Na maioria das vezes a diferença de performance não é tão grande dado um bom ajuste dos hiperparâmetros, mas há diferenças consideráveis na memória utilizada e no tempo de treinamento. Como o número de parâmetros é muito grande, métodos de otimização baseados em inversão da hessiana não são utilizados (MONTAVON, *et al.* 2012).

- Gradiente descendente:

$$x_{k+1} = x_k - p * d_k \qquad \text{Eq 2.5.8}$$

$$p * d_k = \eta * \nabla f(x_k) \qquad \text{Eq 2.5.9}$$

$$x_{k+1} = x_k - \eta * \nabla f(x_k) \qquad \text{Eq 2.5.10}$$

Com momento

$$d_k = \beta * d_{k-1} - \eta * \nabla f(x_k) \qquad \text{Eq 2.5.11}$$

$$x_{k+1} = x_k - p * d_k \qquad \text{Eq 2.5.8}$$

O método do gradiente descendente com momento tem a vantagem de ganhar velocidade quando o gradiente não varia muito e ser capaz de ultrapassar mínimos locais. Além disso, ele se adapta muito bem ao treinamento por batelada. Atualmente é o método preferido para treino de RN com muitas camadas ocultas. Quando a base de dados é pequena faz-se uma busca em grade para se achar o p ótimo, mas geralmente p =η= constante (AVRIEL, 2003). Nesse trabalho usaremos duas variantes de gradiente descendente: RMSprop (*root mean square propagation*), que adapta a taxa de aprendizagem de acordo com os gradientes de cada termo e ADAM (*adaptive moment estimation*), que além de adaptar as taxas de aprendizagem também estima o momento

- Gradiente conjugado:

$$x_{k+1} = x_k - p * d_k \qquad \text{Eq 2.3.8}$$

$$d_k = \beta * \Delta x_{k-1} - \eta * \nabla f(x_k) \qquad \text{Eq 2.3.12}$$

$$\beta = \frac{\nabla f(x_k)^T * \nabla f(x_k)}{\nabla f(x_{k-1})^T * \nabla f(x_{k-1})} \qquad \text{Eq 2.3.13}$$

$$p = \min_p f(x - p * d_k) \qquad \text{Eq 2.3.14}$$

Este método consiste em fazer uma busca em linha na direção conjugada do gradiente descendente. Em geral é mais rápido que o gradiente descendente caso não haja muitos parâmetros. Existem diversas formas de se calcular o β, essa forma é chamada Fletcher-Reeves (FLETCHER & REEVES, 1964). Esta técnica não converge bem quando usado com treinamento em batelada, não sendo usado em aplicações de *Big Data* (LECUN, *et al.* 2012).

- BFGS

$$x_{k+1} = x_k - p * d_k \qquad \text{Eq 2.5.8}$$

$$d_k = B_k^{-1,BFGS} * -\nabla f(x_k) \qquad \text{Eq 2.5.15}$$

$$p = \min_p f(x - p * d_k) \qquad \text{Eq 2.5.14}$$

$$H_k = \nabla f(x_{k+1}) - \nabla f(x_k) \qquad \text{Eq 2.5.16}$$

$$B_{k+1}^{BFGS} = B_k^{BFGS} + \frac{H_k * H_k^T}{H_k^T * (-p * d_k)} - \frac{B_k^{BFGS} * (-p * d_k) * (-p * d_k)^T * B_k^{T,BFGS}}{(-p * d_k)^T * B_k^{BFGS} * (-p * d_k)} \qquad \text{Eq 2.3.17}$$

B^{BFGS} é uma aproximação da hessiana do problema, fazendo este método ter convergência superlinear. O método BFGS é uma expansão do método da secante para otimização. O B^{BFGS} inicial geralmente é a identidade. A princípio B^{BFGS} possui certas propriedades que tornam a sua inversão mais fácil que a inversão da Hessiana. BFGS em geral dá passos muito grandes, o que faz o método divergir em treinamento por batelada (FLETCHER, 1987), fazendo com que BFGS não seja usado para tratar *Big Data*.

A rede neuronal retém a sua propriedade de aproximador universal caso a função de ativação seja diferenciável e tenha limites de mínimo e máximo. As mais comuns atualmente

são a tangente hiperbólica, retificadora linear e sigmoidal. Em geral tangente hiperbólica é preferida em redes rasas, com 1 ou 2 camadas ocultas, por ter uma inversão de sinal e maior curva que a sigmoidal, gerando assim uma resposta menos linear. Em redes profundas, com muitas camadas ocultas, sigmoidal e ReLU são preferidas por gerarem esparsidade, facilitando o treino. Em problemas de classificação RBF também tem tido bons resultados, porém é necessário um número maior de neurônios para treiná-la devido à região de ativação limitada da função de base radial e ela funciona bem em redes profundas (BAUGHMAN & LIU, 1995).

Existe muito debate sobre o número de camadas ocultas e de neurônios que deve-se usar. Existem várias heurísticas mas a maioria funciona muito mal fora dos problemas em que ela foi testada. O ideal é testar várias redes, até achar a rede que possui o menor erro de generalização e o menor número de neurônios. Pode se entender o número de neurônios por camada como quantas características devem ser analisadas para chegar à solução do problema. Pode-se entender o número de camadas como o nível de abstração necessária para se chegar à solução do problema (MONTAVON, *et al.* 2012)

Voltando ao exemplo da identificação de rosto, quantos mais detalhes tem que se procurar maior o número de neurônios por camada. Tem que se procurar por 2 olhos, 1 nariz e 1 boca? Ou também por orelhas, cabelo e queixo? Quanto mais dessas características é necessário observar, maior o número de neurônios requisitados. O que define um olho? Como se define uma boca? Agora sabendo que tem os dois, o posicionamento importa? A cor? Cada nível de pensamento é mais uma camada (LECUN, *et al.* 2015).

Treinar redes profundas (*deep*) é bem mais complicado de que treinar redes de 1 ou 2 camadas ocultas. A maioria dos métodos de treinamento não converge para uma solução, pois os gradientes necessários para o treinamento começam a “sumir” nas primeiras ou últimas camadas, em geral porque as derivadas das funções tanh e sigmoide tendem a zero nos infinitos positivo e negativo, e o gradiente para de regredir para trás da rede no *Backpropagation* (LECUN, *et al.* 2012).

Existem diversas técnicas e metodologias para contornar esses problemas. Essas técnicas melhoraram muito o desempenho das redes neuronais profundas na última década. As mais comuns são: novas funções de ativação, como ReLU; novos métodos de inicialização, como Glorot, Lecun e He; arquiteturas novas que exigem menos pesos como camadas convolucionais; pré-treino das camadas mais profundas como autoencoders, depois juntando as camadas pré-treinadas e usando essas camadas como chute inicial para os pesos de uma rede profunda.(MONTAVON, *et al.* 2012)

Regularização é um aspecto importante das redes neuronais. Como as redes podem possuir dezenas de milhares de pesos é natural que ocorra sobreajuste, quando a rede decora os dados de treino mas não aprende os padrões reais dos dados. Regularização busca restringir as liberdades do modelo, mas não a ponto dele ser incapaz de aprender os dados. Nas RN as técnicas de regularização mais comuns são: regularização L2 nos pesos, *early stopping* (parada cedo), regularização L2 nas ativações, e *dropout*

- Regularização L2 nos pesos, também conhecida como weight decay, é adicionar um termo à função objetivo equivalente ao quadrado dos pesos multiplicado por um peso γ. Na diferenciação aparece um termo $2*\gamma*w$. Isso reduz o tamanho dos parâmetros, evitando grandes variações na saída da rede, deixando-a mais suave. Essa regularização é vista como uma penalidade às derivadas superiores da rede. Os pesos acabam adquirindo uma distribuição normal depois do treino. Existe também regularização L1, em que é adicionado um termo equivalente ao absoluto dos pesos à função objetivo. Essa regularização tende a gerar uma matriz de pesos esparsa, pois durante a diferenciação aparece um termo γ ou $-\gamma$, dependendo do sinal do peso, e se o peso for menor que γ ele é zerado. As vezes isso aumenta a velocidade do treino, pois reduz o número de gradientes a serem calculados. É possível usar qualquer número nessa regularização, como L0.5 ou L4, mas a grande maioria dos trabalhos utiliza regularização L2 ou L1, sendo L2 o preferido. Esse tipo de regularização existe em quase todos os métodos de *Machine Learning*.

- *Early stopping* se baseia em alocar uma parte da base de dados em dados de validação. Esses dados não são utilizados no treinamento, mas o erro da saída deles é calculada depois de cada época. *Early stopping* se baseia em parar o treinamento quando o erro dos dados de validação para de diminuir. A ideia é que primeiro a rede aprende os padrões gerais dos dados, mas depois de um tempo começa a decorar a base de dados de treino, quando o erro dos dados de validação começa a aumentar. Definir como usar *early stopping* é difícil, as vezes o erro de validação aumenta e depois volta a cair, as vezes o erro não aumenta, mas diminui muito pouco em relação ao erro de treinamento, indicando sobreajuste.

- Regularização L2 nas ativações se baseia em reduzir o número de neurônios ativados ao mesmo tempo na rede. Isso induz uma certa esparsidade durante o treinamento, o que facilita o treino, pois evita redundância da rede. De outra forma é possível que alguns neurônios lidem com uma grande parte das tarefas da rede enquanto outros ficam ociosos. Não é bem entendido como essa regularização funciona e quando deve ser utilizada. É usada em autoencoders, redes em que a saída é igual a entrada, quando se quer criar um autoencoder com um número de neurônios na camada oculta maior do que a camada de entrada.

- *Dropout* é uma técnica relativamente nova mas bem poderosa. Consiste em desativar uma porcentagem dos neurônios durante o treino, sendo os neurônios desligados diferentes em cada batelada. Isso diminui consideravelmente a memória utilizada e o tempo computacional requerido, pois muito menos gradientes têm de ser computados. É possível demonstrar que o *dropout* equivale a treinar várias redes diferentes com pesos compartilhados. *Dropout* é muito popular em redes profundas.

2.5.1) Redes neuronais recorrentes

Um modelo de rede importante para esse trabalho é a rede recorrente. Redes recorrentes possuem conexões internas, onde a saída de uma camada se conecta com ela mesma ou com uma anterior. Isso cria uma memória dentro da rede neuronal e ela adquire comportamento dinâmico. Existem diversos tipos de conexões possíveis entre as camadas ocultas, como exemplificado na Figura 9. Esse tipo de rede é excelente para trabalhar com dados temporais, pois leva em conta a correlação temporal entre as variáveis. (ZIMMERMANN, *et al.* 2012). O funcionamento de uma rede recorrente simples, com recorrência interna, é mostrado nas equações 2.3.20 e 2.3.21. h_0 é normalmente uma matriz 0, mas uma matriz aleatória no treino também pode ser usada, caso se queira fazer a RN mais resistente a inicialização ruim.

$$h_t = f(W * x_t + W_h * h_{t-1}) \qquad \text{Eq.2.5.18}$$

$$y_t = W^{out} * h_t \qquad \text{Eq.2.5.19}$$

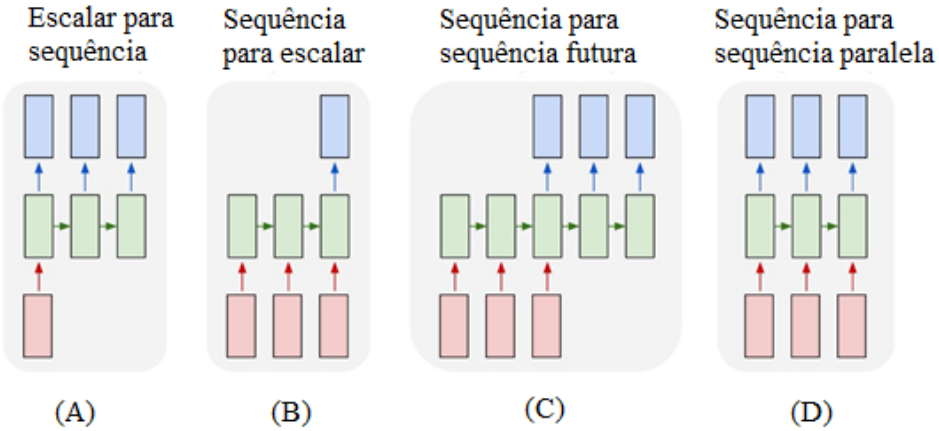

Figura 9: Exemplos da topologia de redes neuronais recorrentes. (KARPATHY, 2015)

Redes recorrentes têm sido usadas para previsão econômica, reconhecimento de fala (ZIMMERMANN, *et al.* 2012) e simulação de processos (BAUGHMAN & LIU, 1995). Treinar redes recorrentes é relativamente difícil, pois é necessário otimizar os pesos da rede anterior e as bifurcações podem fazer a rede divergir e quando convergem, o treinamento é relativamente devagar. A forma mais simples de fazer uma rede recorrente é adicionar atrasos a entradas da rede. A princípio isso faz a rede atuar como um modelo autorregressivo com entrada exôgena (ARX) não linear. Esse tipo de rede será chamada nesse trabalho de MLP-AT (*Multilayer Perceptron* com atraso temporal).

$$x = [x_t x_{t-1} \dots x_{t-n} y_{t-1} \dots y_{t-m}] \qquad \text{Eq. 2.5.20}$$

$$v_i = f(x * W_i + b_i) \qquad \text{Eq. 2.5.1}$$

$$y_j = v * W_j^{out} + b_j^{out} \qquad \text{Eq. 2.5.2}$$

Nesse trabalho também exploramos um tipo de rede recorrente chamada de rede *Echo State* (ESN), cuja representação gráfica pode ser vista na Figura 10 (LUKOŠEVIČIUS, 2012). A proposta das ESN é criar uma rede recorrente com um grande número de neurônios com conexões esparsas, sendo conhecida como reservatório. No treinamento apenas as saídas dos reservatórios são lidas e ajustadas aos dados de treinamento. Para uso em regressão, o treinamento é uma simples regressão linear; para uso em classificação o treinamento é uma regressão logística. Não há necessidade de treinar a rede recorrente desde que o reservatório seja grande o suficiente. Como as regressões são bem mais simples e menos custosas

computacionalmente que o *backpropagation* a ESN se adapta melhor ao *Big Data*. Vale lembrar que os métodos de regularização disponíveis para regressões também estão disponíveis para ESN, como LASSO, regressão *Ridge* e rede elástica (LUKOŠEVIČIUS, 2012).

O número de neurônios numa ESN pode chegar a dezenas de milhares e elas aceitam centenas de milhares de exemplos com facilidade. As equações 2.5.21-22 ditam a atualização do estado.

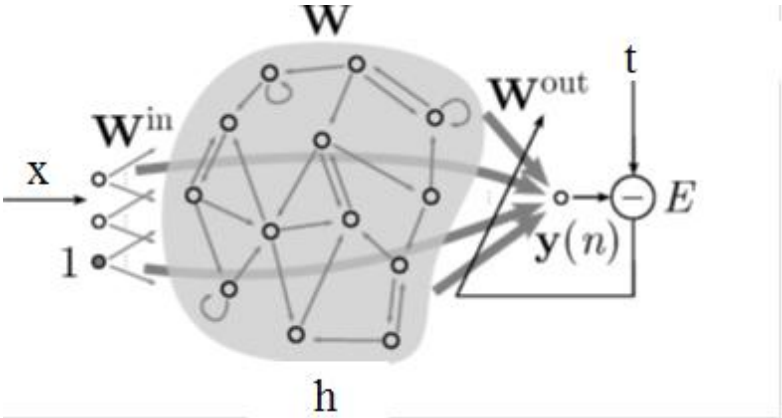

Figura 10: Rede *Echo State* (LUKOŠEVIČIUS, 2012).

$$h'_t = f\left(W^{in} * x_t + W * h_{t-1}\right) \qquad \text{Eq 2.5.21}$$

$$h_t = (1-\alpha) * h_{t-1} + \alpha * h'_t \qquad \text{Eq 2.5.22}$$

$$y_t = W^{out} * [x_t\ h_t] \qquad \text{Eq 2.5.23}$$

Em que h é o análogo ao vetor de estados, α é o fator de amortização, x_t são as entradas do processo no tempo t e y_t são as saídas do processo no tempo t, f() é uma função não linear, geralmente tangente hiperbólica.

Apenas W^{out} é treinado, W^{in} e α são hiperparâmetros escolhidos pelo usuário e W é gerado randomicamente. W^{in} é equivalente à matriz B, W e α são equivalente à matriz A na equação 2.6.2.1, e W^{out} é equivalente às matrizes C^{SS} e D^{SS} na equação 2.6.2.2. W^{in} é escolhido em termos da linearidade do processo, quanto mais linear for a relação entre as entradas e as saídas, mais a tangente hiperbólica se aproxima da função identidade e a saída do modelo se torna mais linear. α é escolhido em termos da dinâmica do processo. Se o processo tiver dinâmica bem mais lenta que a taxa de amostragem, α se aproxima de 0, pois

os estados não devem mudar tão rapidamente; se as dinâmicas do processo forem rápidas α se aproxima de 1 (LUKOŠEVIČIUS M., 2012).

Outra arquitetura bastante popular atualmente são as redes recorrentes *gated*. Redes recorrentes tradicionais não possuem memória longa, os elementos mais recentes acabam contribuindo para a maior parte da memória interna. As redes *gated* possuem camadas ocultas dedicadas a controlar o estado interno da rede, permitindo assim à rede possuir uma memória maior. A rede recorrente *gated* mais popular é a LSTM (*Long-short term memory*, memória longa-curta) (CHUNG, *et al.* 2014). Uma LSTM possui 3 camadas para o controle do estado interno da rede, chamado de portões: um portão para apagar parte do estado, chamado portão de esquecimento; um portão para adicionar elementos ao estado, portão de entrada; e um portão para liberar o estado para ser usado como saída da rede, chamado portão de saída.

$$F_t = f_g\left(W_f * x_t + W_{hf} * h_{t-1}\right) \qquad \text{Eq. 2.5.24}$$

$$i_t = f_g(W_i * x_t + W_{hi} * h_{t-1}) \qquad \text{Eq. 2.5.25}$$

$$o_t = f_g(W_o * x_t + W_{ho} * h_{t-1}) \qquad \text{Eq. 2.5.26}$$

$$c_t = F_t * c_{t-1} + i_t * f_c(W_c * x_t + W_{hc} * h_{t-1}) \qquad \text{Eq. 2.5.27}$$

$$h_t = o_t * f_h(c_t) \qquad \text{Eq. 2.5.28}$$

$$y_t = W^{out} * h_t \qquad \text{Eq. 2.5.19}$$

Redes tipo GRU (*Gated Recurrent Unit*, Unidade de portão recorrente) (CHUNG *et al.* 2014) também se mostraram bem úteis. Redes tipo GRU são parecidas com redes LSTM, mas combinam o portão de esquecimento com o portão de entrada, agora chamado de portão de redefinição. A lógica é que adicionar informação e esquecer informação devem ocorrer ao mesmo tempo. O portão de saída é renomeado portão de atualização Esse tipo de rede possui menos parâmetros, sendo assim mais fácil de treinar.

$$z_t = f_g(W_z * x_t + W_{hz} * h_{t-1}) \qquad \text{Eq 2.5.29}$$

$$r_t = f_g(W_r * x_t + W_{hr} * h_{t-1}) \qquad \text{Eq 2.5.30}$$

$$h_t = z_t * h_{t-1} + (1 - z_t) * f_h\left(W_h * x_t + W_{hh}(r_t * h_{t-1})\right) \qquad \text{Eq 2.5.31}$$

$$y_t = W^{out} * h_t \qquad \text{Eq 2.5.19}$$

2.5.2) Redes neuronais convolucionais

Outro tipo de RN utilizado nesse trabalho é a rede convolucional (LECUN, *et al.* 2015). Redes convolucionais são muito utilizadas em reconhecimento de imagem. Enquanto nas redes tradicionais é recebido um vetor na camada de entrada, em redes convolucionais recebe-se uma matriz, geralmente uma imagem. A saída de uma camada convolucional é outra matriz. As vezes, as redes recebem várias matrizes de mesmo tamanho ao mesmo tempo, como uma imagem contendo canais para as cores verde, azul e vermelho. As dimensões das camadas são chamadas de filtros. Do mesmo modo que uma camada normal pode conter muitos, neurônios uma camada convolucional pode conter muitos filtros. Depois da convolução uma não linearidade é aplicada, geralmente ReLU.

No pseudocódigo a seguir, alt e lar são a altura e a largura da janela de convolução, respectivamente. A operação de convolução pode ser visualizada na Figura 11 Pode-se observar que uma convolução retorna uma matriz menor do que a entrada. Isso é um problema para redes mais profundas, pois acarreta em perda de informação. Para compensar a perda de tamanho adicionam-se linhas e colunas contendo zeros às bordas da matriz y, uma prática conhecida como *padding* (BRITZ, 2015).

```
for i=1:size(x,1)-alt
     for j= 1:size(x,2)-lar
         y(i,j)=sum(W**x(i:i+alt,j:j+lar))+b
```

$$CONV_{xy} = b + \sum_{i=1}^{alt} \sum_{j=1}^{lar} W_{ij} * X_{x+i-1,y+j-1} \qquad \text{Eq 2.5.32}$$

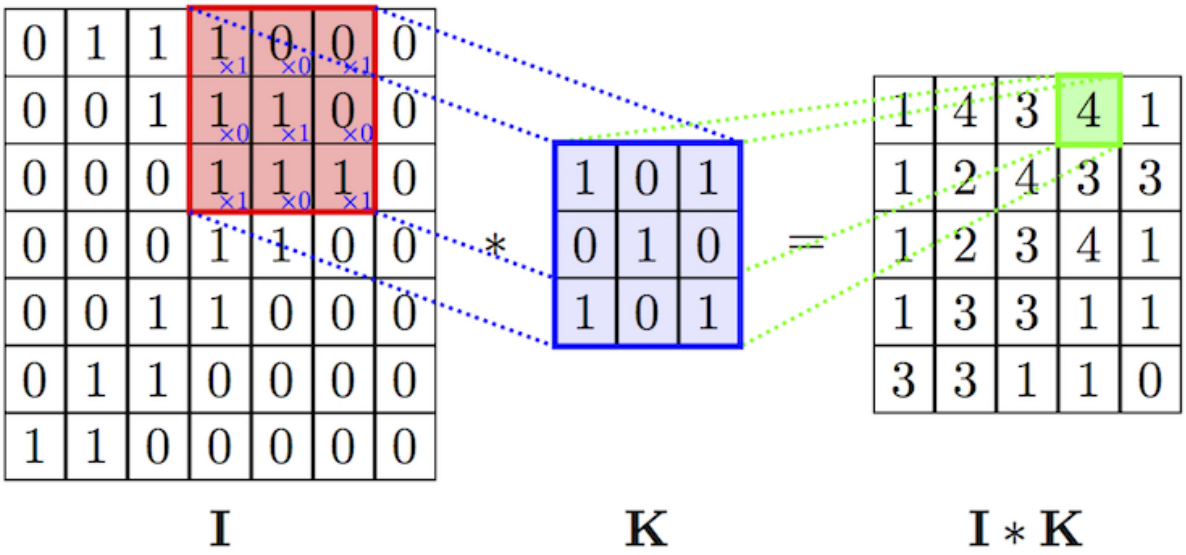

Figura 11: Exemplo de operação de convolução (BRITZ, 2015).

Na maioria das redes convolucionais, a saída das camadas convolucionais é vetorizada, uma matriz m por n vira um vetor com m*n elementos, e conectada a uma camada regular. Essa camada regular em geral contém a maior parte dos parâmetros da rede, pois a vetorização cria vetores bem grandes (LECUN, *et al.* 2015).

Uma parte importante das redes convolucionais é a camada de *pooling*. *Pooling* é uma operação de convolução não treinada, seguindp uma equação definida pelo usuário. A camada de *pooling* é usada para reduzir o tamanho das saídas das camadas convolucionais, reduzindo o número de parâmetros a serem calculados. Os tipos mais comuns de *pooling* são *pooling* médio, que tira a média dos valores mais próximos, e *max pooling*, que escolhe o valor máximo. *Max pooling* é mais comum, pois exige menos operações, o que torna o cálculo da rede mais rápido, como pode ser visto na Figura 12. No final as camadas convolucionais e de *pooling* são concatenadas com uma rede simples, como demonstrado na Figura 13 (BRITZ, 2015).

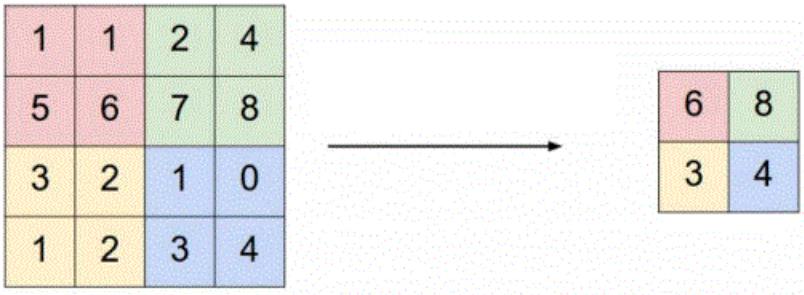

Figura 12: Operação de *Max pooling* (BRITZ, 2015).

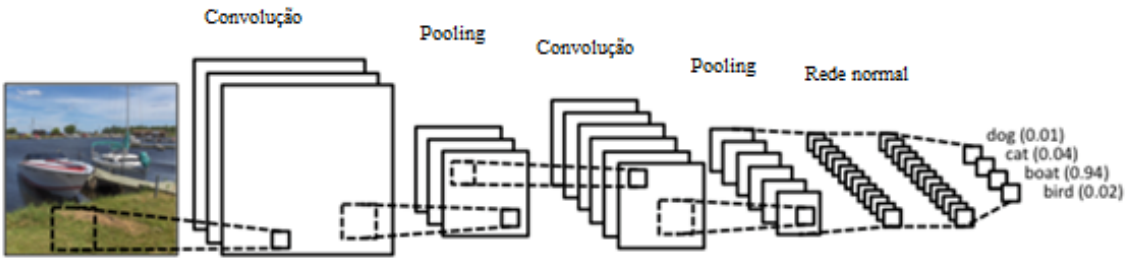

Figura 13: Exemplo de rede neuronal convolucional (BRITZ, 2015).

2.6) Métodos lineares de classificação e regressão

Existem diversos métodos lineares de classificação e regressão aplicados a dados temporais. Eles em geral são mais conhecidos e usados do que os métodos de *Machine Learning*. Há um debate entre os cientistas de dados se esses métodos podem ser chamados de *Machine Learning*. A princípio eles também utilizam uma base de dados para obter um modelo do processo, por outro eles assumem certas características dos modelos, como uma relação linear entre as características, além de terem sido desenvolvidos sem uma preocupação com *Big Data*. Nesse trabalho usaremos modelo de estado de espaços (SS) e modelo autorregressivo exógeno (ARX) para regressão e análise de componentes principais (PCA) e análise discriminante de Fischer (FDA) para classificação.

2.6.1) ARX

ARX é uma representação de um processo randômico, que depende linearmente dos seus valores anteriores, de perturbações externas e de um termo estocástico de ruído. Ela é definida pela equação 2.6.1.1 (LJUNG, 1999).

$$y_t = b + \sum_{i=1}^{p} \alpha_i^{ARX} * y_{t-i} + \sum_{i=1}^{q} \beta_i^{ARX} * x_{t-i} + \varepsilon \qquad \text{Eq 2.6.1.1}$$

Em que y_t são as variáveis do modelo no tempo t, x_t são as perturbações do modelo no tempo t e ε é um ruído branco. Em engenharia química Y é visto como as saídas do processo e x como as entradas do processo. α^{ARX} e β^{ARX} são os parâmetros dos modelos, que podem ser estimados pelo método da máxima verossimilhança ou por regressão linear. Os hiperparâmetros "p" e "q" indicam o quanto o modelo "olha para trás" do processo. Quanto maior p e q mais importante são os eventos passados.

As fontes de incerteza possíveis do ARX são a estimação dos coeficientes, o ruído branco e as saídas dos modelos. ARX assume que as entradas são perfeitamente conhecidas. A ordem do modelo em geral é estimada por autocorrelação parcial. A autocorrelação parcial analisa o quanto o valor atual de uma variável depende dos seus valores atrasados.

2.6.2) Modelo de espaço de estado

Modelos de espaço de estado usam variáveis de estado para descrever um sistema diferencial de primeira ordem. As variáveis de estado são calculadas durante o treinamento, mas não são diretamente medidas. Um modelo SS é bom para se fazer uma estimação de parâmetros rápida, pois o modelo possui apenas 1 hiperparâmetro, o número de variáveis de estado, também conhecido como ordem. O modelo pode ser visto como um diagrama de blocos na Figura 14.

Os modelos de espaço de estado são descritos na forma discreta pelas equações 2.6.2.1-2:

$$h_{t+1} = A^{SS} * h_t + B^{SS} * x_t \qquad \text{Eq 2.6.2.1}$$

$$y_t = C^{SS} * h_t + D^{SS} * x_t \qquad \text{Eq 2.6.2.2}$$

Em que y_t são as saídas do modelo no tempo t, x_t são as entradas do modelo no tempo t e h_t são as variáveis de estado. Em engenharia química, y é visto como as variáveis medidas do processo, x como as entradas do processo e h como as equações diferenciais internas do processo. A^{SS}, B^{SS}, C^{SS} e D^{SS} são as matrizes que são calculadas na identificação. As matrizes e o x inicial são calculadas a partir do método da máxima verossimilhança. A ordem do modelo é estimado pelos autovalores de Hankel (LJUNG, 1999).

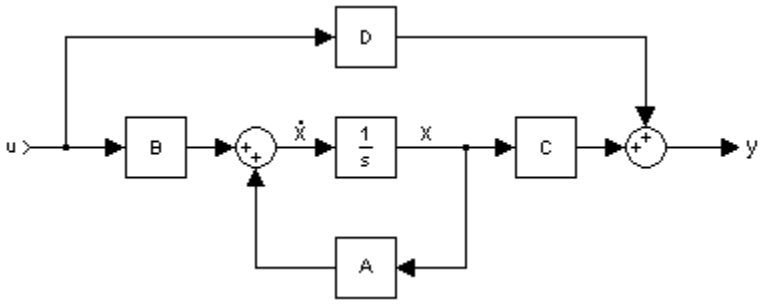

Figura 14: representação gráfica do modelo de espaço de estados contínuo (LJUNG, 1999).

2.6.3) PCA dinâmico

"*Principal components analysis*" (análise de componentes principais, comumente conhecido como PCA) é uma técnica de redução de dimensionalidade que busca expressar os dados em uma base ortogonal que explica a maior parte da variabilidade dos dados. Os vetores da base ortogonal são os componentes principais, em que o primeiro explica a maior parte da variância, o segundo componente explica uma parte menor da variância e assim por diante. PCA também permite a observação das estruturas dos dados. Visualizando quais características fazem partes dos mesmos vetores, pode-se descobrir estruturas dos processos.

Em geral, os últimos vetores são excluídos para a redução da dimensionalidade, pois tendem a ser a representações dos ruídos do sistema. Então os dados são passados para a base ortogonal de dimensão reduzida e analisados.

Supondo uma matriz X contendo n observações de m variáveis, o primeiro componente principal da base ortogonal w_1 é encontrado maximizando a equação 2.6.3.1, o segundo componente principal (PC) é encontrado subtraindo a projeção do primeiro PC na base de dados e depois otimizando a mesma equação (CHIANG, *et al.* 2001).

$$\frac{w^T * X^T * X * w}{w^T * w} \qquad \text{Eq 2.6.3.1}$$

Esse problema equivale a fazer decomposição dos valores singulares de X, equação 2.6.3.2.

$$\frac{1}{\sqrt{(n-1)}} * X = U * \Sigma * V^T \qquad \text{Eq 2.6.3.2}$$

$$P = U * \Sigma \qquad \text{Eq. 2.6.3.3}$$

Em que as colunas de V são os vetores w e Σ contêm os valores singulares de X em sua diagonal principal, ordenadas em ordem decrescente. A variância explicada pelo modelo é igual ao quadrado dos valores singulares. Os componentes principais são agrupados na matriz P. Para redução da dimensionalidade, retiram-se os últimos valores de P, que correspondem à menor parcela da variabilidade da base de dados. Esses cálculos são dependentes da escala das características da matriz X, então é recomendado estandardizá-la, transformar a média e a variância das características para 0 e 1 respectivamente.

PCA é muito utilizado em controle multivariável, pois permite a observação de múltiplas variáveis ao mesmo tempo. Nesse trabalho foi usado a estatística T^2, também conhecida como estatística de Hotelling, dada pela equação 2.6.3.4. A estatística T^2 pode ser interpretada como a distância entre um ponto com os dados normais usados no cálculo do PCA. Dado um vetor de observações x sendo reduzido para "a" componentes principais:

$$T^2 = x^T * P_a * {\Sigma_a}^{-2} * {P_a}^T * x \qquad \text{Eq 2.6.3.4}$$

O limite para se detectar com α% de certeza se o modelo está em situação anormal é dado pela equação 2.6.3.5:

$$T_\alpha^2 = \frac{a * (n-1) * \big((n+1)\big)}{n * (n-a)} * F_a(\alpha, n-a) \qquad \text{Eq 2.6.3.5}$$

em que Fα(a, n-m) é o ponto crítico de α% da distribuição F com a e n-a graus de liberdade (CHIANG, *et al.* 2001).

2.6.4) FDA dinâmico

FDA, também conhecido análise do discriminante de Fischer, é similar ao PCA. Só que ao invés de buscar uma representação linear que explique a maior parte da variância, ele

busca uma representação linear que separe melhor 2 ou mais classes. Geralmente é usado para redução de dimensionalidade seguido de classificação.

Para uma matriz de dados X contendo n observações e m características pertencentes a p classes, a matriz de espalhamento total S_t é calculado pela equação 2.6.4.1 (CHIANG, *et al.* 2001).

$$S_t = \sum_{i=1}^{n} (x_i - x_m) * (x_i - x_m) \qquad \text{Eq 2.6.4.1}$$

Sendo x_m o vetor das médias de cada característica para toda a base de dados. A matriz de espalhamento interno é definida para cada classe j pela equação 2.6.4.2:

$$S_j = \sum_{i=1}^{nj} \left(x_i - x_{mj}\right) * \left(x_i - x_{mj}\right) \qquad \text{Eq 2.6.4.2}$$

Sendo x_{mj} o vetor das médias de cada característica cuja observação pertença a classe j. A matriz de espalhamento dentro das classes é dado pela equação 2.6.4.3:

$$S_w = \sum_{i=1}^{p} S_j \qquad \text{Eq 2.6.4.3}$$

A matriz de espalhamento entre as classes é dado pela equação 2.6.4.4:

$$S_b = \sum_{j=1}^{n} \left(x_{mj} - x_m\right) * \left(x_{mj} - x_m\right) \qquad \text{Eq 2.6.4.4}$$

O primeiro vetor do FDA maximiza o espalhamento entre classes e cada vetor seguinte explica melhor a diferença entre as classes. Os vetores são a coluna da matriz w_k. É possível resolver esse problema pela resolução do problema de autovalores e autovetores dado pela equação 2.6.4.5:

$$S_b * w_k = \lambda * S_w * w_k \qquad \text{Eq 2.6.4.5}$$

A função discriminante é dada pela equação 2.6.4.6:

$$g_j(x) = \frac{-\left(x - x_{jm}\right)^T}{2} * W_a * \left(\frac{1}{n_j - 1} * W_a^T * S_j * W_a\right)^{-1} * W_a^T * \left(x - x_{mj}\right) + ln\left(p_j\right) - \frac{ln\left(det\left(\frac{1}{n_j - 1} * W_a^T * S_j * W_a\right)\right)}{2} \qquad \text{Eq 2.6.4.6}$$

A saída da função discriminante é um vetor contendo as probabilidades do vetor x pertencer a tal classe. O maior valor de g é o valor ao qual a classe pertence. A princípio é possível fazer detecção de falha atribuindo uma classe a operação normal. O termo dinâmico, tanto no PCA quanto no FDA refere-se ao fato que as entradas dos métodos não são apenas a observação dos sensores em um instante no tempo e sim observações de uma janela temporal, contendo todos os valores dentro de período de tempo.

2.7) Modelagem de incerteza

Existem poucos trabalhos de modelagem de incerteza. A grande maioria dos estatísticos e engenheiros busca minimizar a incerteza dos sensores usando diversos métodos: filtros passa-baixa, filtro de Kalman, reconciliação de dados etc. (NARASIMHAN & JORDACHE, 2000). Em geral fabricantes de sensores e pesquisadores da área de instrumentação buscam modelar as fontes de incerteza, como pontos de solda, isolamento ruim, etc e como a incerteza se propaga (BAGAJEWICZ, 2001).

Neste trabalho, para aumentar a similaridade entre a simulação e uma situação real de uma planta é introduzida incerteza aos dados obtidos. Quatro tipos de incerteza são adicionados aos dados do processo:

- Ruído branco: ruído branco são números randômicos de distribuição normal. É a forma mais usual de modelagem de incerteza. Assume-se que as fontes de ruído são múltiplas e independentes entre si. Logo, mesmo que cada fonte de ruído tenha a sua própria distribuição de probabilidade, pelo teorema central do limite, a média dos ruídos segue uma distribuição normal de média zero (BAGAJEWICZ, 2001).

- Desvio: Esse tipo de ruído é o desvio do sensor do padrão de calibração. Isso faz com que a suposição de que a média dos erros do sensor é igual a zero não seja válida. Desvio também é conhecido por reduzir a precisão dos sensores. As origens de um desvio são váriadas: desgaste do sensor, principalmente em medidores de fluxo, alterações no processo, acumulo de sujeira etc. O desvio é tratado a partir de recalibração, em que a saída do sensor é comparada com um padrão de valor conhecido e a saída do sensor é reescalonada de acordo (BAGAJEWICZ, 2001; KRISHNAMURTHY & CHAN, 2009).

- *Outliers*: *outliers* são pontos fora da curva. Podem surgir de várias fontes: erros crassos do sensor, situações de operação anormal, vazamentos, dano físico ao sensor ou a condição do sistema está fora da zona de confiabilidade do sensor. Removê-los é essencial pois causam desvios graves na maioria dos métodos de reconciliação de dados. Há quem argumente que esses dados não devam ser descartados e sim separados e analisados para se tentar obter informação sobre casos de operação anormal. (BAGAJEWICZ, 2001; NARASIMHAN & JORDACHE, 2000) Neste trabalho eles ocorrem naturalmente durante as falhas do Tennessee Eastman.

- Dados faltantes: dados podem sumir do banco de dados devido a erro humano, um operador esqueceu de anotá-los ou enviá-los à base de dados, isso é comum em medidas offline como concentrações e especificações de produto ou por erro no sistema de comunicação entre a planta e a base de dados. Um cabo de comunicação pode ter ser partido, por exemplo (YAN, *et al.* 2004). Também ocorre eventualmente dos dados sumirem por falha no sensor. Diferentes bases de dados tratam dados ausentes de forma diferente. Algumas atribuem valor 0, outras atribuem valor NaN, "*not-a-number*". Nesse trabalho, modelaremos a falta de dados excluindo a concentração de D e a temperatura da camisa de resfriamento do reator de van de Vusse.

3) Inferência estatística

Nesse trabalho realizamos um estudo de caso para a aplicação de *Machine Learning* em inferência estatística: inferir a concentração e a produção de um produto de interesse no reator de van de Vusse. O reator de van de Vusse é um processo bem conhecido na literatura, com muitas não linearidades, principalmente uma inversão de ganho que torna o seu controle muito difícil (GONÇALVES, *et al.* 2015).

3.1) Estudo de caso: Reator de van de Vusse

O reator de van de Vusse é um CSTR (*Continous stirred tank reactor*), ou seja, um reator contínuo perfeitamente agitado, de forma que não haja gradientes das suas variáveis em termos da localização espacial dentro dele. Neste reator ocorrem 3 reações: A →B→C e 2A →D. As cinéticas são de primeira ordem:

$$A \rightarrow B \qquad \text{Eq. 3.1.1}$$

$$r_1 = -dC_a/dt = k_1 * C_a \qquad \text{Eq. 3.1.2}$$

$$B \rightarrow C \qquad \text{Eq. 3.1.3}$$

$$r_2 = -dC_b/dt = k_2 * C_b \qquad \text{Eq. 3.1.4}$$

$$2A \rightarrow D \qquad \text{Eq. 3.1.5}$$

$$r_3 = \frac{-dC_a}{dt} = k_3 * C_a^2 \qquad \text{Eq. 3.1.6}$$

os k seguem a equação de Arrhenius, Eq. 3.1.7

$$k_i = k_{0i} * exp(-E_i/(R * T)) \text{ i=1,2,3} \qquad \text{Eq. 3.1.7}$$

Nessas equações E_i é a energia de ativação i, r_i é a taxa da reação, k_i é a constante da reação e k_{0i} é o fator pré-exponencial. T é a temperatura em Kelvin. C_a e C_b são as concentrações de A e de B.

Esse processo representa a síntese de ciclopentanol (B) a partir do ciclopentadieno (A), procurando minimizar a produção de ciclopentadienol (C) e diciclopentadieno (D). Este

sistema foi proposto por van de Vusse para exemplificar as a dificuldade de escolhas de reator entre um reator contínuo tubular de escoamento empistonado (ou PFR - *Plug Flow Reactor*) e um reator CSTR (VAN DE VUSSE, 1964).

As equações do modelo dinâmico partem da aplicação dos princípios fundamentais de conservação aos balanços de massa:

$$\frac{d(V * C_a)}{dt} = entra - sai + gerado - consumido \qquad \text{Eq. 3.1.8}$$

$$\frac{d(V * C_a)}{dt} = \frac{C_a * dV}{dt} + V * dC_a/dt \qquad \text{Eq. 3.1.9}$$

$$\frac{d(C_a)}{dt} = (F_{in} * C_{ain} - F_{out} * C_a)/V - k_1 * Ca - k_3 * C_a^2 - \left(\frac{C_a * dV}{dt}\right)/V \qquad \text{Eq. 3.1.10}$$

$$\frac{d(C_b)}{dt} = \frac{-F_{out} * C_b}{V} + k_1 * C_a - k_2 * C_b - \frac{C_b * dV}{dt}/V \qquad \text{Eq. 3.1.11}$$

$$\frac{d(C_c)}{dt} = \frac{-F_{out} * C_c}{V} + k_2 * C_b - \frac{C_c * dV}{dt}/V \qquad \text{Eq. 3.1.12}$$

$$\frac{d(C_d)}{dt} = \left(-F_{out} * C_d + k_3 * C_a^2 - \frac{C_d * dV}{dt}\right)/V \qquad \text{Eq. 3.1.13}$$

A variação de volume é baseada na equação do volume de tanque simples

$$\frac{dV}{dt} = F_{\text{in}} - F_{out} = F_{\text{in}} - Cv * \%ab * \sqrt{h_r} \qquad \text{Eq. 3.1.14}$$

F_{in} é a vazão de entrada e F_{out} é a vazão de saída. A vazão de saída é proporcional à raiz quadrada do nível de líquido, Cv é a constante de válvula, %ab é a porcentagem de abertura e h_r é o nível de líquido no reator. No algoritmo implementado, Cv e %ab são incluídos na mesma constante.

No balanço de energia, a fim de dissociar os efeitos da temperatura da água de resfriamento e a vazão, assumimos que a água de resfriamento é adicionada em um tanque de resfriamento perfeitamente misturado de volume muito inferior ao reator. Assumimos volume constante a esse tanque de resfriamento e que dentro dele as propriedades da água são homogêneas, uma suposição válida uma vez que a vazão da água de resfriamento é muito superior ao volume do tanque. Assim:

$$\frac{dT}{dt} = (F_{in} * T_{in} - F_{out} * T)/V - (q_r - q_g)/(\rho * V * cp) - \left(\frac{T * dV}{dt}\right)/V \qquad \text{Eq 3.1.15}$$

T_{in} é a temperatura da corrente de entrada, T é a temperatura do reator, q_r é o calor removido pelo trocador de calor, q_g é o calor gerado pelas reações, ρ é a densidade e c_p é a capacitância térmica da mistura.

As seguintes equações auxiliares podem ser escritas:

$$q_g = -(\Delta H_{r1} * r_1 + \Delta H_{r2} * r_2 + \Delta H_{r3} * r_3) \qquad \text{Eq. 3.1.16}$$

$$q_r = U_{TT} * A_{TT} * (T - T_{H20}) \qquad \text{Eq. 3.1.17}$$

ΔH_{ri} é o calor de reação da reação i, T_{H2O} é a temperatura da água de resfriamento, U_{TT} é o coeficiente global de troca térmica e A_{TT} é a área de troca térmica. A variação temporal de T_{H2O} é descrita por:

$$\frac{d(T_{H2O})}{dt} = \frac{(T_{H2Oin} - T_{H2O}) * F_r}{V_r} + q_r/(\rho_{H2O} * V_r * cp_{H2O}) \qquad \text{Eq. 3.1.18}$$

tal que $T_{H2O\ in}$ é a temperatura da água de resfriamento que entra, F_r é a vazão de água de resfriamento que entra, V_r é o volume da jaqueta, ρ_{H2O} e cp_{H2O} são, respectivamente, a densidade e a capacitância térmica da água.

A Tabela 2 contém os parâmetros utilizados na simulação do reator de van de Vusse. (VAN DE VUSSE, 1964).

Tabela 2: Parâmetros do reator de van de Vusse (VAN DE VUSSE, 1964).

$A = 4\ m^2$	$E_1 = 81{,}091$ kJ
$h_{max} = 2$ m	$E_2 = 81{,}091$ kJ
$k_1 = 2{,}145 * 10^{11}\ m^3/s$	$E_3 = 71{,}133$ kJ
$k_2 = 2{,}145 * 10^{11}\ m^3/s$	$\Delta H_{r1} = 4200$ kJ/kmol
$k_3 = 1{,}507 * 10^{8}\ m^3/kmol/s$	$\Delta H_{r2} = -11000$ kJ/kmol
$Cp_{H2O} = 4{,}185$ kJ/kg/K	$\Delta H_{r3} = -41850$ kJ/kmol
$Cp = 2{,}8121$ kJ/kg/K	$UA = 75{,}8$ kW/K
$\rho = 1000\ kg/m^3$	$V_j = 0{,}05\ m^3$
$\rho_{H20} = 1000\ kg/m^3$	

O reator de van de Vusse é muito utilizado em exemplos de controle preditivo, pois possui em uma ampla faixa de operação, comportamento de fase não mínima e comportamento muito não linear. A capacidade das redes neuronais para o controle desse processo para situações normais está bem estabelecido (GONÇALVES, *et al.* 2015; ANURADHA, *et al.* 2009).

Nesse modelo existem 6 entradas, 7 equações diferenciais e 8 saídas. A concentração inicial de A e a temperatura da água de resfriamento são pertubações, enquanto a vazão da água de resfriamento, válvula de saída do reator, vazão de entrada do reator e temperatura do reagente são variáveis manipuladas. A Figura 15 exibe um fluxograma do reator.

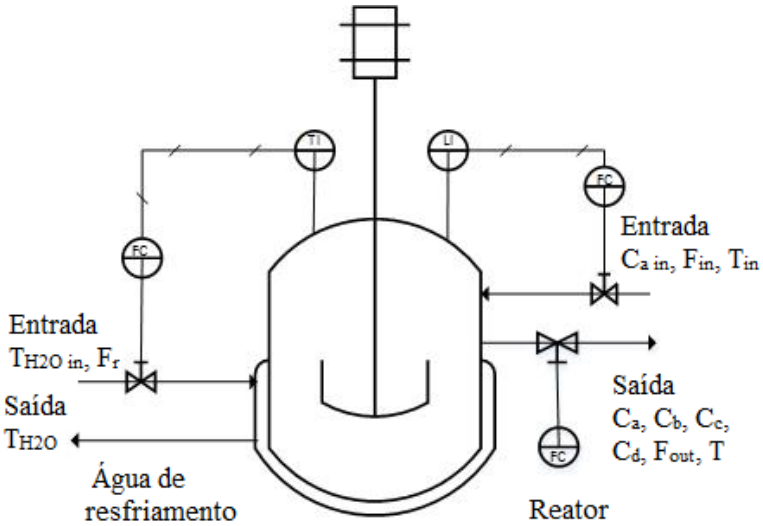

Figura 15: Fluxograma do reator de van de Vusse.

O modelo para o reator de van de Vusse foi implementado em Simulink 6.0, MATLAB 7.0. O código está nos apêndices deste texto. Foi feita preliminarmente uma análise RGA, matriz de ganhos relativos (não mostrada aqui), que determinou que a vazão da água de resfriamento deveria ser usada para controlar a temperatura e a vazão de entrada para controlar o nível do reator. O RGA determinou também que um controlador da válvula de saída seria o ideal para se controlar C_b. A Tabela 3 contém os parâmetros dos controladores.

Tabela 3: Parâmetros dos controladores.

Parâmetro	**Nível**	**Temperatura**
K_p	9.113 (ação reversa)	10.32 (ação direta)
K_i	82.34 (ação reversa)	55.72 (ação direta)
Limites	[800 -790]	[300 -299]
Coeficiente de *back calculation*	5	8

Para percorrer o espaço de estados foram feitas perturbações tipo degrau em 3 entradas e em 2 *setpoints*: concentração de A na saída, temperatura da corrente de entrada e posição da válvula de saída; *setpoint* da temperatura do reator e *setpoint* do volume do reator. Foram adicionados atrasos de 1 segundo para evitar a existência de equações algébricas diferenciais implícitas, acelerando assim o código e deixando-o mais robusto. A Figura 16 é o modelo implementado no Simulink 6.0.

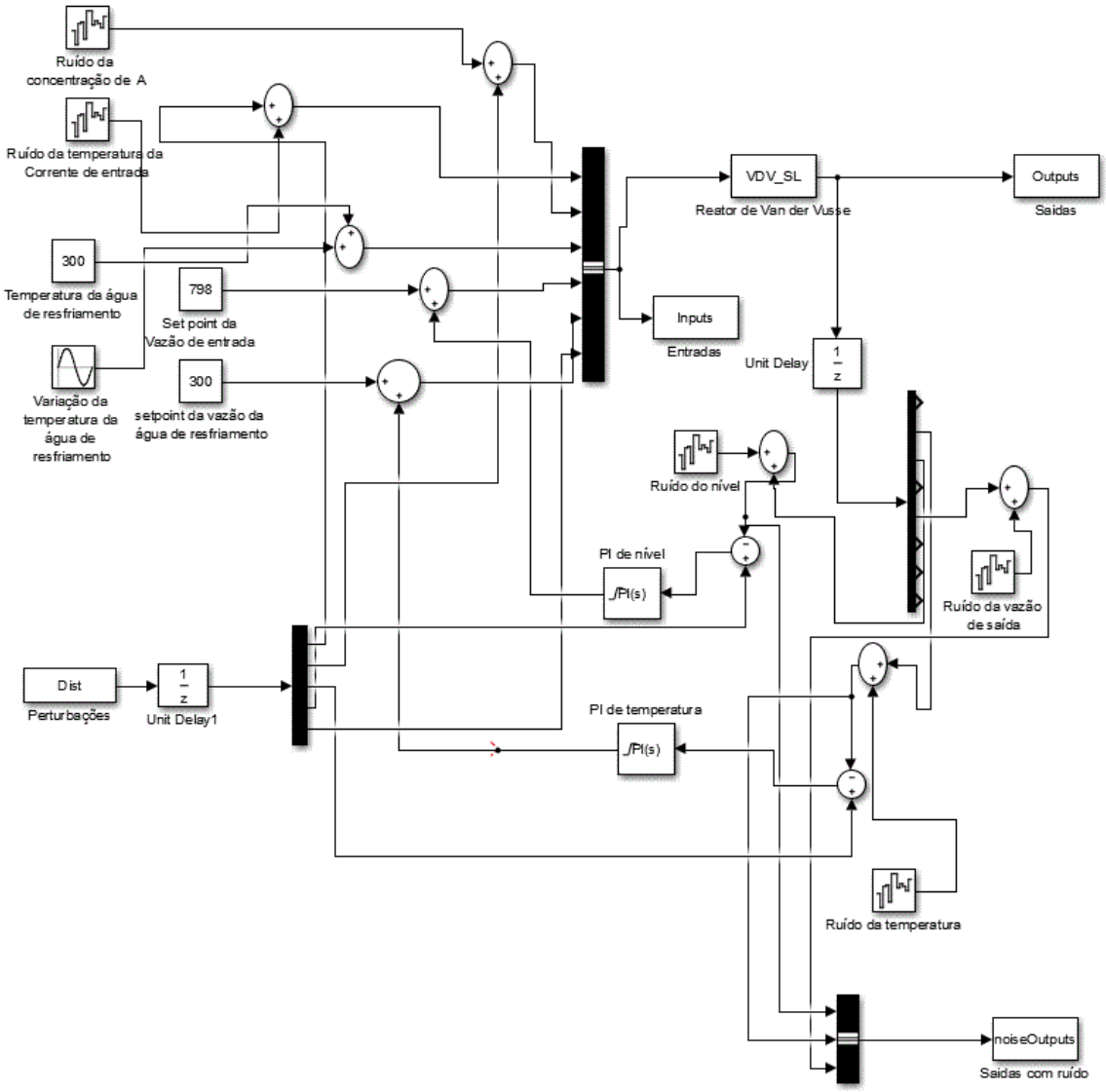

Figura 16: Modelo do reator de van de Vusse implementado no Simulink 6.0.

O processo foi simulado por 400 horas com tempo de amostragem de 1 segundo. Essa taxa de amostragem foi reduzida para 6 s depois da simulação, pois 1 segundo produz muitos dados redundantes. Na base de dados alocamos as primeiras 200 horas para treino, 100 horas para validação, da hora 200 a hora 300; 100 horas para teste, da hora 300 a hora 400, resultando em um total de 120000 pontos para treino, 60000 para validação e 60000 para teste. Nas variáveis que não receberam ruído durante a simulação, foi adicionado ruído depois da base de dados estar completa. Antes de passar pelas análises, os dados foram submetidos a um filtro exponencial com fator de amortização de 70% para reduzir o ruído. Nas análises, C_d e T_{H2O} foram excluídos, para caracterizar a falta de informações sobre o espaço de estados completo.

$$filtro\ exponencial = x^*{}_k = a * x^*_{k-1} + (1-a) * x_k \qquad \text{Eq. 3.1.19}$$

x^* é o sinal filtrado, x o sinal original e α o fator de amortização

Neste estudo de caso, identificamos o sistema para prever C_b e a produção total de B. Manter um C_b alto é o principal objetivo da maioria das aplicações do reator de van de Vusse, mas nesse trabalho decidimos adicionar a produção total de B já que a vazão de entrada não é a mesma da vazão de saída, diferentemente dos modelos tradicionais que assumem volume constante.

Para fazer a identificação do sistema, usamos 2 modelos de rede neuronal e 2 modelos lineares análogos às redes. Os modelos lineares análogos escolhidos foram o modelo SS e modelo ARX. Os modelos lineares foram encontrados e analisados usando o *System Identification Toolbox* do MATLAB. Para avaliar os modelos foram usados as médias do quadrado dos erros, tanto para a predição de um passo adiante tanto para predição em ciclo fechado. Na primeira, nós calculamos C_b e a vazão de B no momento seguinte, seis segundos adiantes no tempo. Na predição de ciclo fechado, nós calculamos C_b e a vazão de B realimentando a saída do modelo como uma entrada. Essa predição é importante, pois avalia a robustez dos modelos e se eles não estão apenas reproduzindo os valores anteriores. Os modelos de redes neuronais escolhidos foram os Redes tipo *Echo State* e redes com atraso temporal na entrada (MLP-AT). A ESN foi criada usando uma *toolbox* pública desenvolvida e mantida pelo professor Herbert Jaeger (JAEGER, *et al.* 2007). A MLP-AT foi criada usando a biblioteca Keras, em Python (CHOLLET, 2015). O computador utilizado foi um laptop com Windows 7,8 GB de ram, processador Intel Core i7 3612QM com 2 núcleos e 2.10 GHz de processamento, sem placa gráfica dedicada. Os parâmetros das redes foram iguais aos análogos lineares onde essa comparação faz sentido estatisticamente. A rede com entradas com atraso temporal usaram também 2 termos de atraso, porém a ordem da rede *Echo State* foi bem maior, pois o treinamento ocorre apenas na saída. A otimização dos hiperparâmetros foi feita a partir de busca em grade, em que vários valores são testados em sequência e o melhor escolhido.

Para avaliar a qualidade dos modelos foram usadas 2 métricas: raiz do erro quadrático médio (RMSE, *root mean square error*) e RMSE normalizado pelo desvio padrão (NRMSE, *normalized root mean square error*), conforme definições a seguir:

$$RMSE = \sqrt{\sum (t - y)^2 / n} \qquad \text{Eq 3.1.20}$$

$$NRMSE = 1 - \frac{\sqrt{\sum (t - y)^2 / n}}{\sqrt{\sum \left(t - E(t)\right)^2 / n}} \qquad \text{Eq 3.1.21}$$

A métrica RMSE foi escolhido por ter a mesma unidade das variáveis, dando assim uma ideia da efetividade dos modelos contra os seus valores base; NRMSE foi escolhida por ser adimensional, assim dando uma ideia da efetividade dos modelos entre C_b e produção de B.

Não foi feita análise da normalidade dos ruídos, pois, pela natureza do experimento, espera-se que eles não sejam normais. Espera-se que os ruídos se concentrem nas regiões mais distantes do estado estacionário, devido à inversão de ganho, logo a distribuição não é normal. Por certas propriedades das redes neuronais os parâmetros não afetam o modelo em toda a região possível e a regularização limita a liberdade dos modelos, por isso análises relativas ao número de parâmetros como AIC não são aplicáveis.

3.2) Resultados

Os valores utilizados na estandardização da base de dados seguem no apêndice IV.

A Figura 17 ilustra o nível de ruído para C_b e a produção de B. O modelo do reator de van de Vusse foi simulado com sucesso, possuindo a característica curva C_b vs F_{out}/V no estado estacionário, demonstrada na Figura 18 e conhecida na literatura (GONÇALVES, *et al.* 2015). Também foi feita a curva de C_b vs F_{out}/V usando os dados de treino, Figura 19, para demonstrar o nível de ruído na base de dados e o quão incompleto é o espaço de estados disponível para treino, um problema industrial comum, conforme citado anteriormente. É interessante notar que transientes na Figura 19 também são bem comuns, como pode ser visto pelas regiões mais esparsas e com pontos isolados.

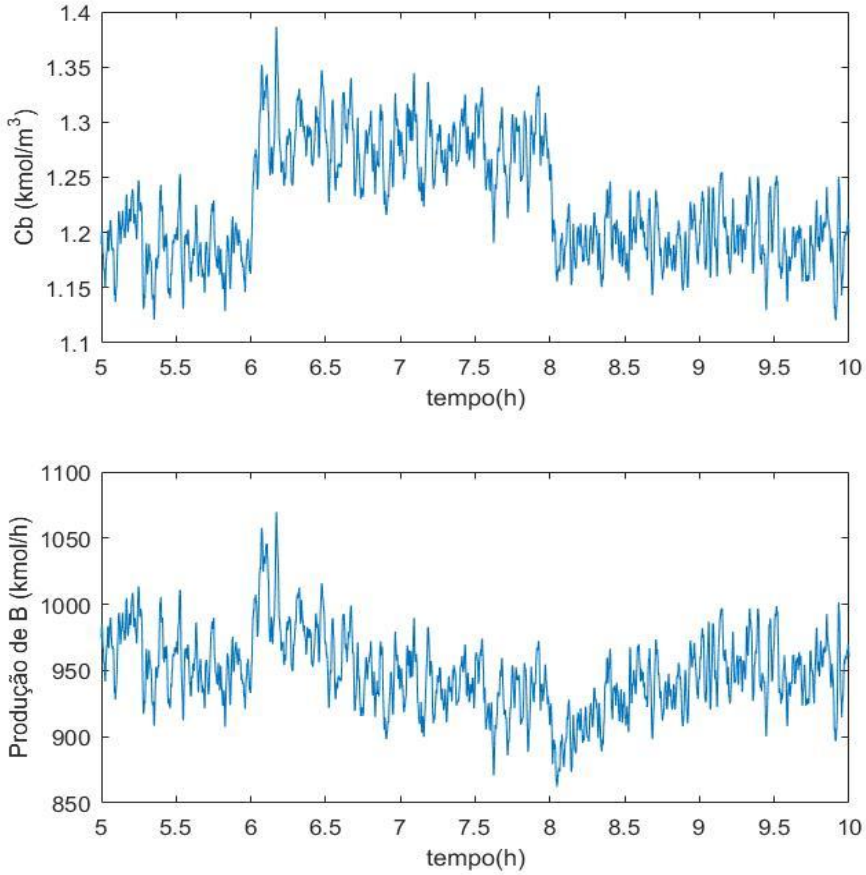

Figura 17: Ilustração da saída do reator de van de Vusse.

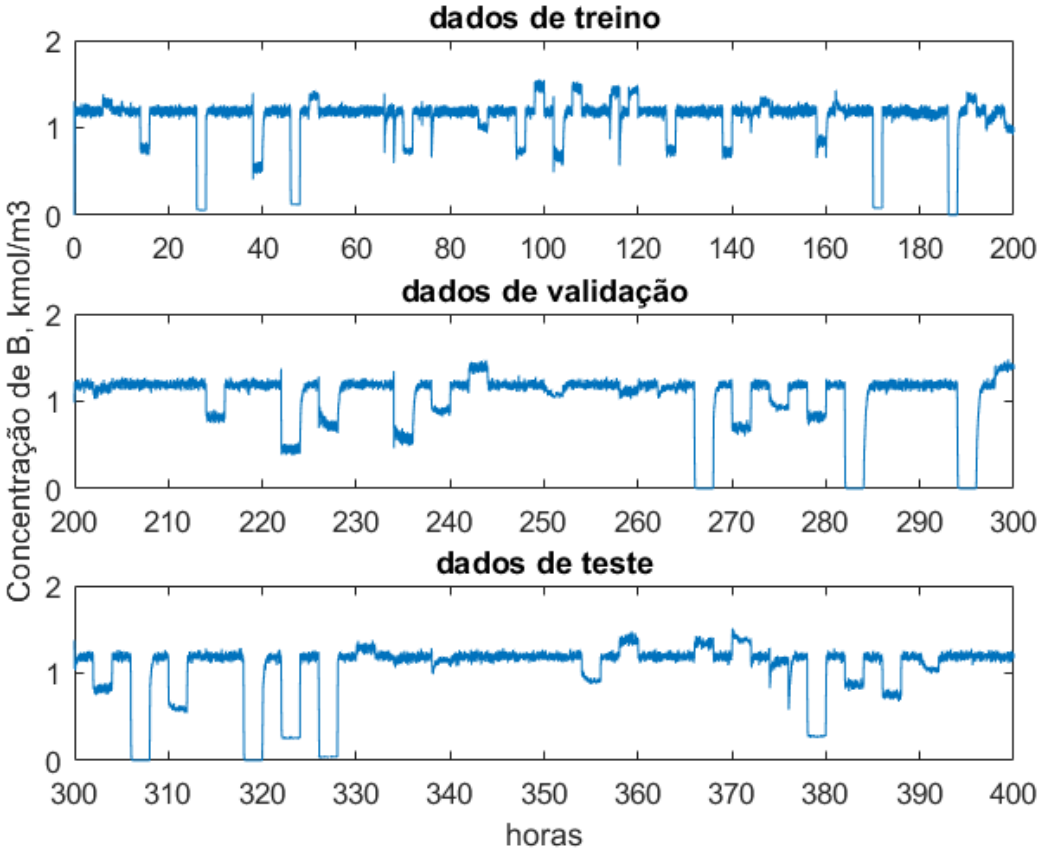

Figura 18: todos os valores na base de dados para concentração de B

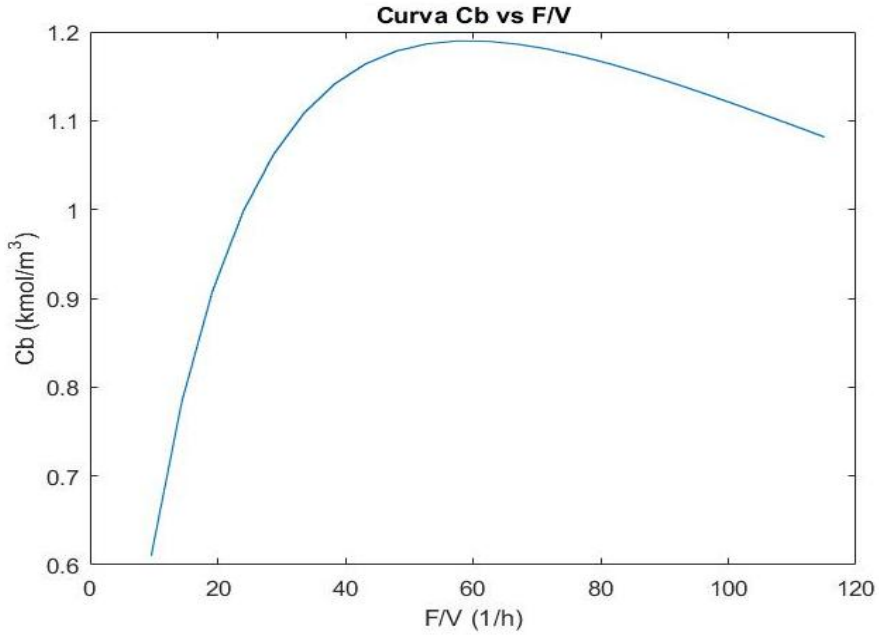

Figura 19: Curva C_b vs F/V característica do reator de van de Vusse. Vale notar que a inversão de ganho é bem pronunciada.

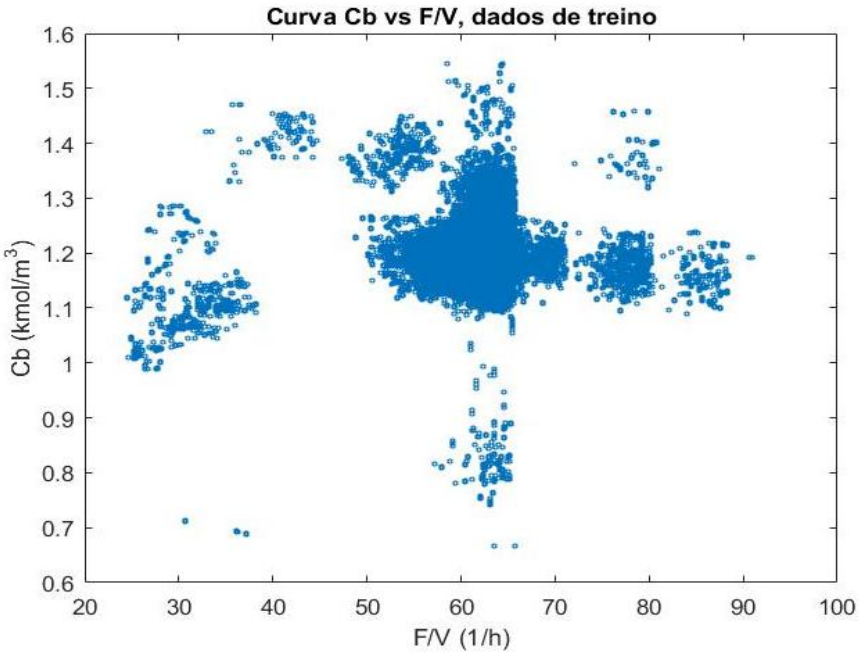

Figura 20: Curva C_b vs F/V característica do reator de van de Vusse, feita com os dados de treinamento.

3.2.1) Modelos lineares

A Figura 21 exibe a avaliação da autocorrelação parcial de C_b, usada na definição dos atrasos usados no modelo ARX.

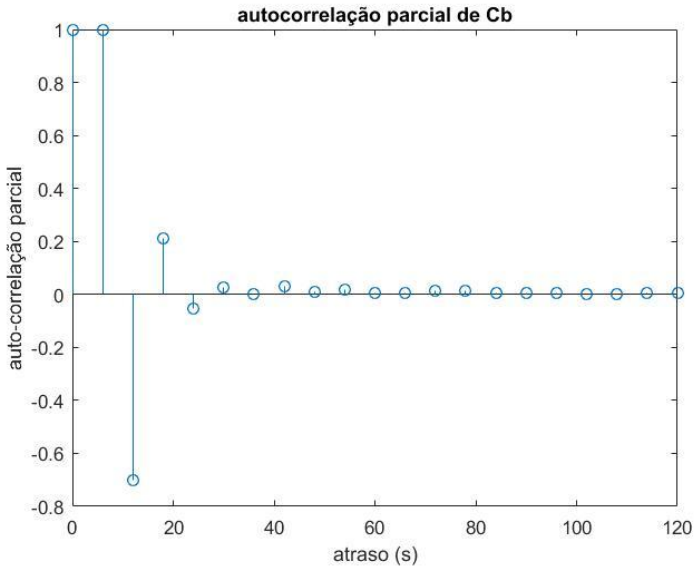

Figura 21: Autocorrelação parcial de C_b, usado para se definir os atrasos usados no ARX.

Como pode-se ver o primeiro e segundo atrasos possuem valores muito superiores ao terceiro em diante, portanto apenas eles foram utilizados. Assim, para ARX, o melhor resultado obtido foi com 2 termos de atraso, que corresponde aos pontos com maior autocorrelação parcial. Para o modelo de Espaço de estados o melhor resultado foi com um modelo de ordem 10. Estes parâmetros estão sumarizados na Tabela 4.

Tabela 4: Hiperparâmetros dos modelos lineares.

Atraso do ARX	2
Ordem do SS	10

Os resultados dos modelos lineares foram muito bons para a predição do valor da variável no instante seguinte, praticamente coincidindo com as curvas dos valores reais. Além disso a pouca diferença entre os modelos. Por essas razões essas figuras não são interessantes e não serão mostradas. Para a predição em ciclo fechado, o SS diverge fortemente depois de algumas horas, como pode ser percebido na Figura 22.

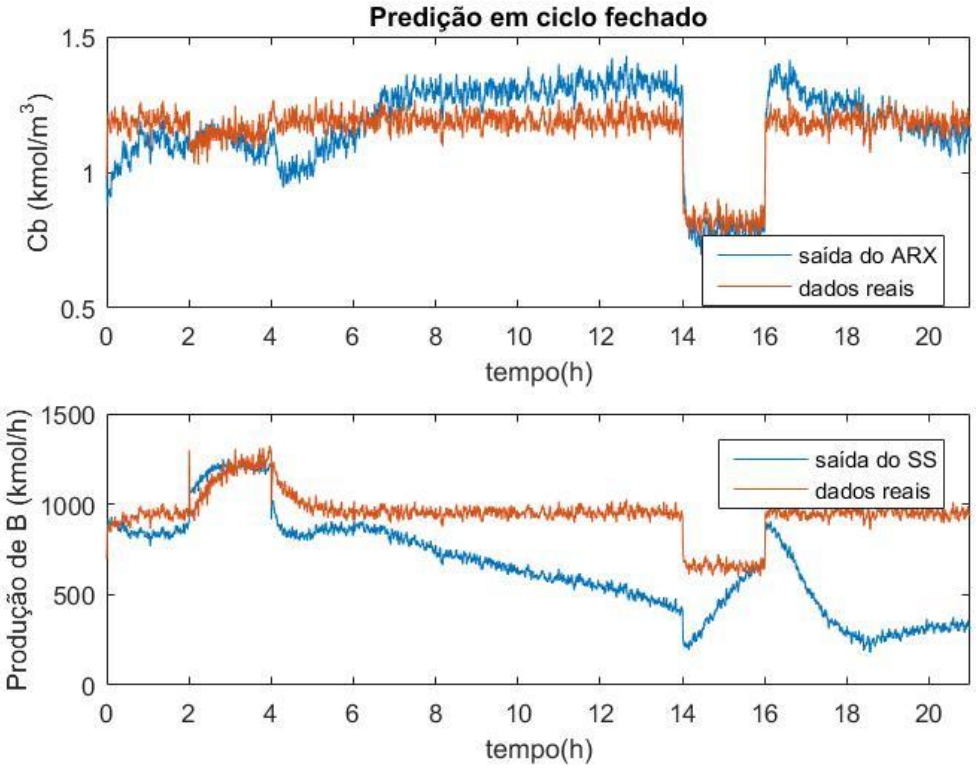

Figura 22: Saída dos modelo simulado comparado à saída dos modelos lineares.

Os resultados gerais para todos os dados de teste e validação estão resumidos na Tabela 5 em termos das métricas empregadas. NRMSE vai de 1 até -∞. NMRSE de 1 indica

um ajuste perfeito; 0, um ajuste ruim, cuja variância dos ruídos é igual à variância do processo e negativo, um modelo cuja variância dos ruídos do modelo predito é maior que a variância dos dados originais, indicando que o modelo pode ser pior que um chute aleatório.

Assim, para passo seguinte, pela análise do RMSE, encontram-se menores erros (métrica próxima a zero) para o ARX, passo seguinte, confirmado para NRMSE próximo de 1.

Para o uso em ciclo fechado, nenhum modelo linear é satisfatório, pois o NRMSE é negativo para o ARX e o modelo SS diverge, não possibilitando a análise por tais métricas. O tempo de treinamento do modelo ARX foi de 56 segundos e o tempo de treino do modelo SS foi de 227 segundos.

Tabela 5: Resultados dos modelos lineares para todos os padrões de teste e validação.

	RMSE		NRMSE	
ARX	C_b (kmol/m^3)	B (kmol/h)	C_b	B
Passo seguinte	2.37E-03	2.148837	0.9902	0.9895
Ciclo fechado	0.382492	3.07E+02	-0.1532	-0.1193

SS	C_b (kmol/m^3)	B (kmol/h)	C_b	B
Passo seguinte	5.20E-02	5.159835	0.8421	0.9821
Ciclo fechado	DIVERGE			

3.2.2) Rede *Echo State*

A Tabela 6 lista os hiperparâmetros ótimos da ESN. A Tabela 7 é o resultado de uma busca em grade para encontrar os valores ótimos do raio espectral (maior autovalor da matriz W) e do fator de amortização. O número em cada retângulo é o erro, a cor vai de verde para vermelho conforme o erro aumenta.

Tabela 6: Hiperparâmetros da ESN.

Raio espectral	0.4
Número de neurônios	800
Fator de amortização	0.95
Nível de ruído	0.3
Método de treinamento	Regressão linear regularizada por ruído
Não-linearidade	tangente hiperbólica

Tabela 7: Resultado da busca em grade pelos melhores hiperparâmetros.

		Raio espectral					
		1.58E-01	2.51E-01	3.98E-01	6.31E-01	1.00E+00	1.58E+00
Fator de amortização	1.19E-01	1.61E-01	7.95E-02	6.02E-02	6.30E-02	5.20E-02	4.87E-02
	2.69E-01	9.06E-02	8.66E-02	8.79E-02	4.65E-02	4.34E-02	4.88E-02
	5.00E-01	2.27E-02	4.42E-02	5.15E-02	2.10E-02	6.32E-02	4.81E-02
	7.31E-01	3.21E-02	3.06E-02	2.93E-02	2.70E-02	2.94E-02	2.83E-02
	8.81E-01	2.22E-02	1.91E-02	2.89E-02	2.58E-02	1.95E-02	3.18E-02
	9.53E-01	1.91E-02	1.48E-02	8.21E-03	1.36E-02	1.16E-02	2.07E-02

Os resultados de predição foram bons. Em predição de ciclo fechado a rede *Echo State* foi um modelo melhorado que o modelo de espaço de estados, como pode ser visto na Tabela 8, Mas em malha aberta ARX e modelo de estados de espaço se mostraram superiores, como pode ser visto na Tabela 5.

Tabela 8: Resultados da ESN para todos os padrões de teste e validação.

	RMSE		NRMSE	
ESN	C_b (kmol/m^3)	B (kmol/h)	C_b	B
Passo seguinte	8.00E-03	7.3072	0.9761	0.9736
Ciclo fechado	0.2076	167.8895	0.4675	0.4899

A Figura 23 mostra a saída da ESN comparada ao modelo simulado. Pode se ver que mesmo depois de 52 horas de simulação a saída da ESN continua próxima à do processo real, isso ilustra a estabilidade do modelo. Uma propriedade interessante dessa rede é que a saída dela é comprimida entre -1 e 1 pela tangente hiperbólica, impedindo que o estado interno da rede atinja valores muito altos e que o modelo se torne instável. Isso, junto com o "α" definido pelo usuário, impede que o modelo varie bruscamente, aumentando a estabilidade da ESN. O tempo de treinamento do modelo foi de 60 segundos

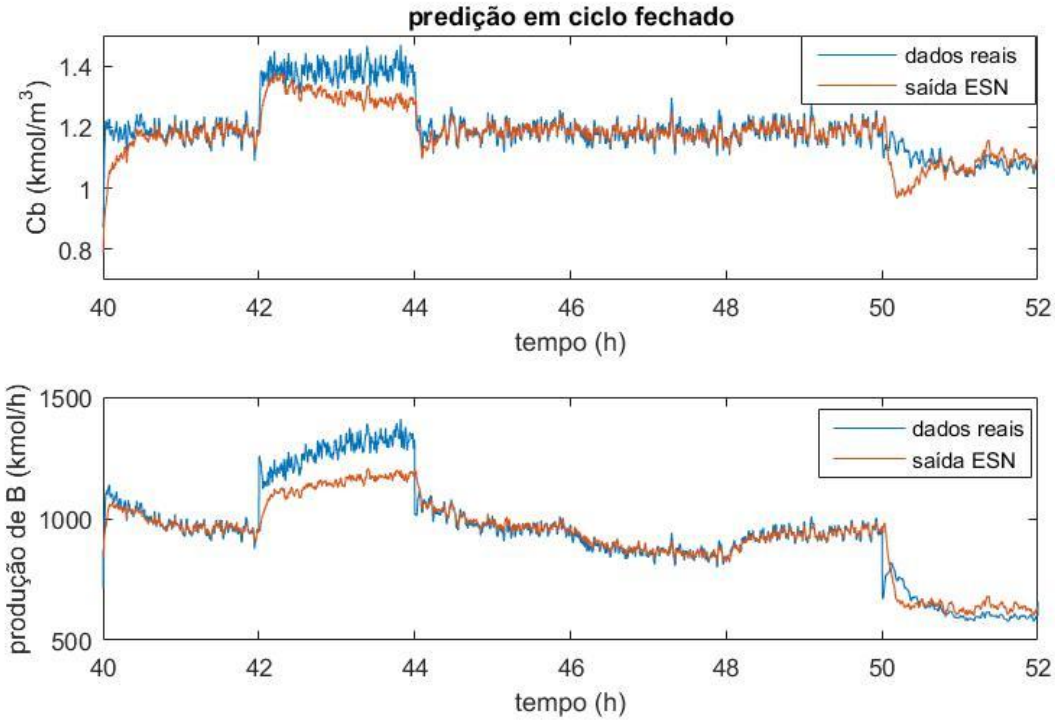

Figura 23: Saída da modelo simulado comparado a saída da ESN.

3.2.3) Rede neuronal com atraso temporal

A Figura 24 apresenta os resultados da busca para o melhor tamanho da camada oculta. Foi escolhido um número de 90 neurônios para esta camada, por levar ao menor erro RMSE para predição de passo adiante de C_b. A Tabela 9 lista os hiperparâmetros utilizados para o treinamento da rede,

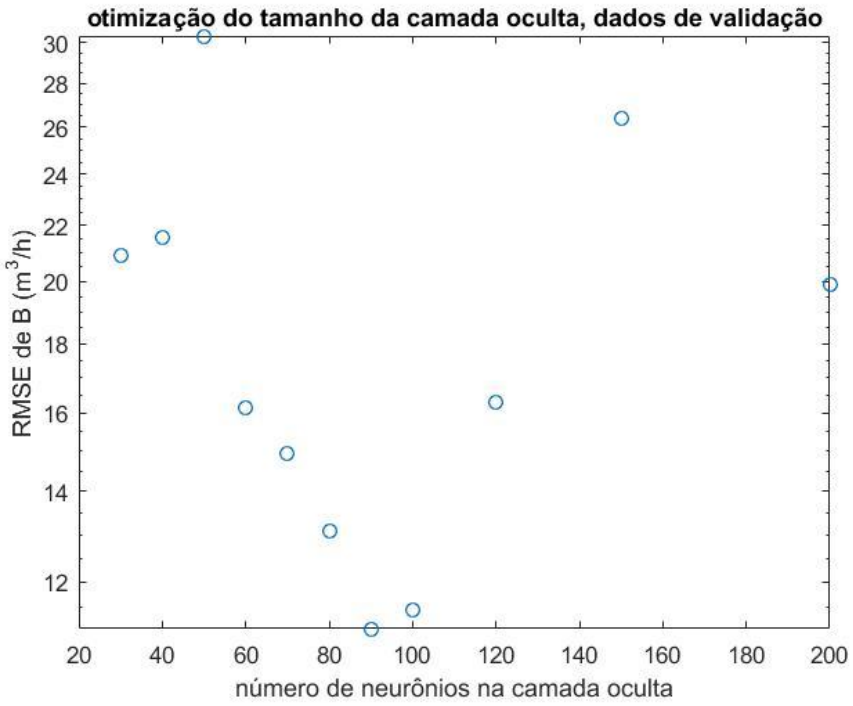

Figura 24: RMSE para predição de passo adiante de C_b em relação ao tamanho da camada oculta.

Tabela 9: Hiperparâmetros da MLP-AT.

Dropout	30%
Número de neurônios	90
Termo de regularização L2	$5*10^{-3}$
Épocas máximas	200
Taxa de aprendizagem	0.001
Fator de decaimento da taxa de aprendizagem	5e-3
Método de treinamento	RMSprop
Não linearidade	ReLU

Em ciclo fechado os resultados da MLP-AT foram os melhores encontrados, como pode ser visto na Tabela 10.

Tabela 10: Resultados da MLP-AT para todos os padrões de teste e validação.

	RMSE		NRMSE	
MLP-AT	C_b (kmol/m^3)	B (kmol/h)	C_b	B
Passo seguinte	1.75e-2	15.72	0.947	0.943
Ciclo fechado	8.674e-2	77.789	0.739	0.718

Os resultados foram bem melhores que a ARX, indicando que a rede conseguiu aprender bem as não linearidades do reator de van de Vusse. As Figuras 25 e 26 mostram a saída da MLP-AT durante toda a base de dados. É interessante notar que a rede funciona como um modelo de caixa-preta, não sendo possível entender o seu funcionamento interno. Por exemplo: em torno da hora 60 dos dados de validação, na figura 26 o modelo prediz um aumento de C_b que não aconteceu e não é possível saber o porquê. O tempo de treinamento do modelo é em torno de 400 segundos.

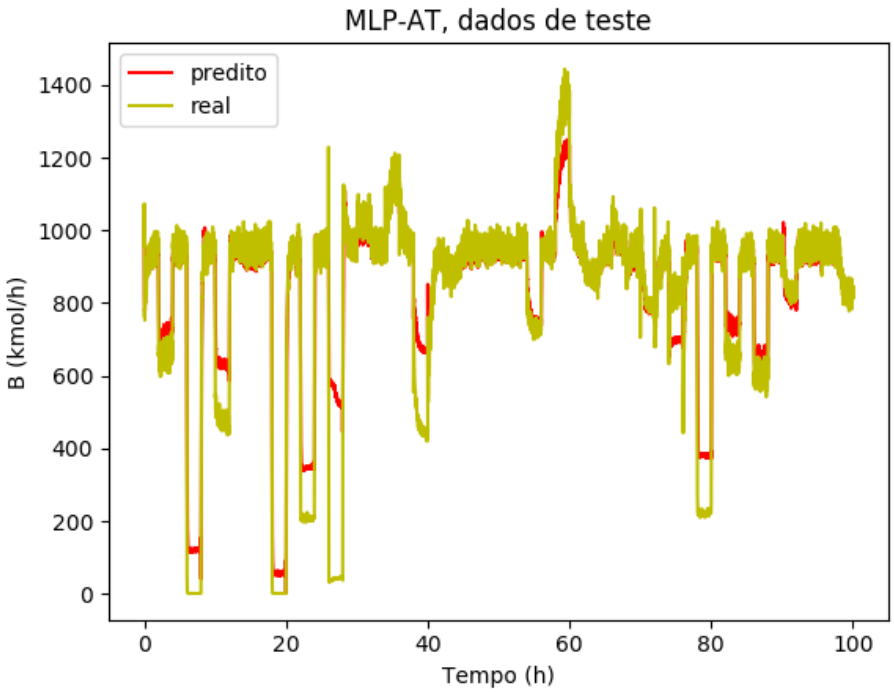

Figura 25: Saída da modelo simulado comparado a saída da rede neuronal com atraso temporal. Base de dados de teste completo.

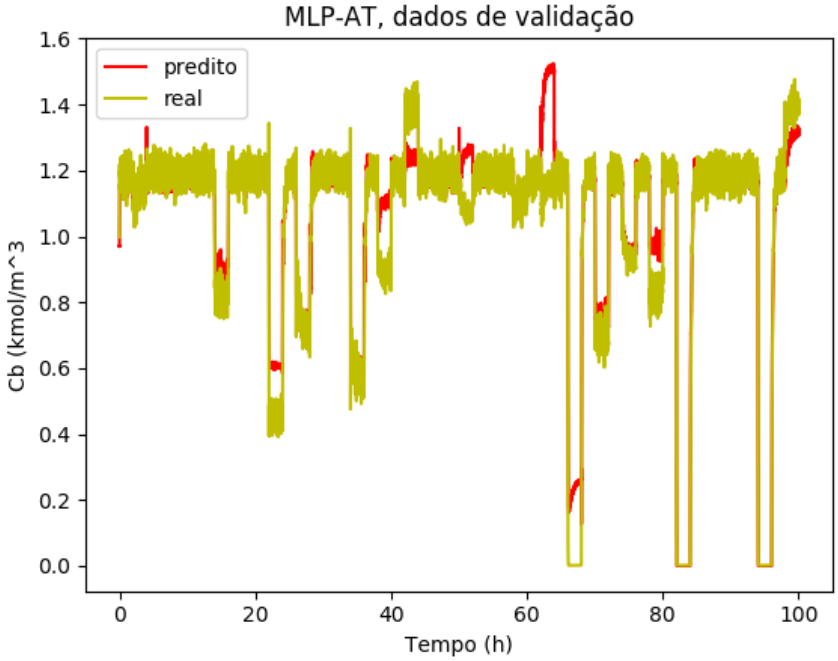

Figura 26: Saída da modelo simulado comparado a saída da rede neuronal com atraso temporal. Base de dados de teste completo.

Aproximação da não-linearidade.

Para demonstrar que a rede neuronal aproxima melhor a não linearidade, foram plotadas as curvas C_b vs F/V preditas pela MLP-AT, pelo ARX, pelo ESN e pelo modelo do reator, na Figura 27. É possível ver que a MLP-AT aprendeu a inversão de ganho do sistema, apesar de divergir quanto mais se distância do ponto de inflexão, onde os dados de treino eram menos comuns. O ARX aproxima bem o modelo depois de F/V = 55 1/h, mas diverge consideravelmente quando F/V < 50 e ainda é monotônico, sendo incapaz de aprender a inversão de ganho.

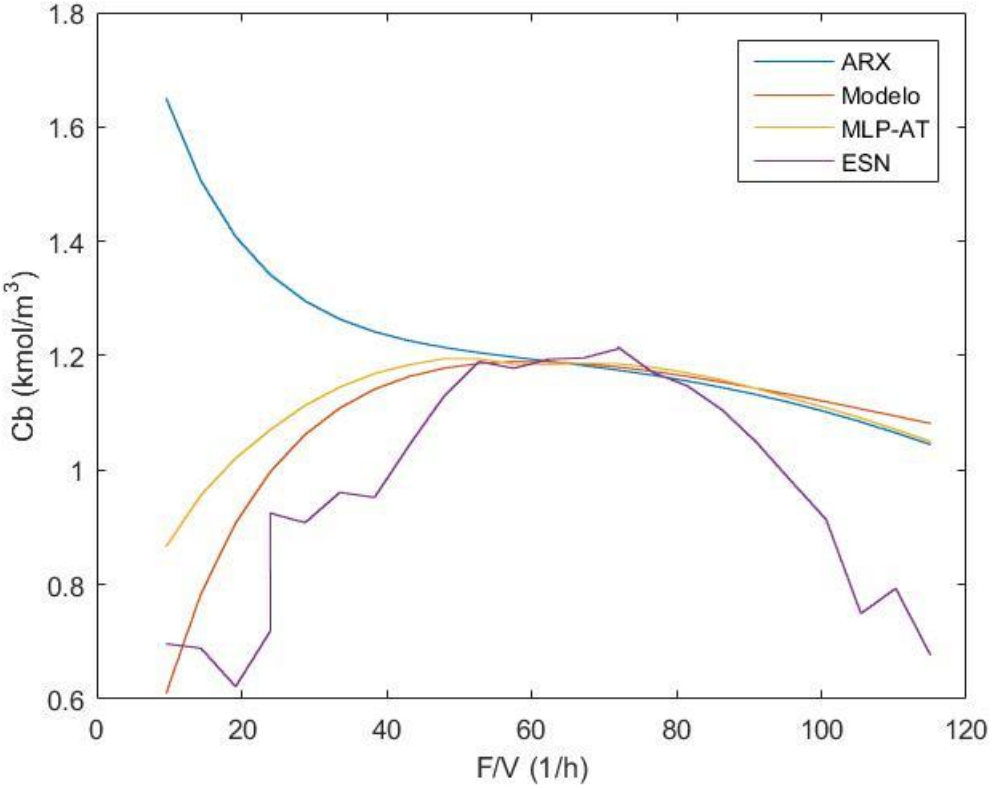

Figura 27: Curva F/V do modelo simulado, comparado as curvas produzidas pelo ARX, MLP-AT e ESN.

3.3) Sumário

A Tabela 11 apresenta o resumo dos resultados e as Figuras 28 e 29 exibem as curvas de todos os modelos juntos, para fins de comparação.

A análise desses resultados evidencia que as redes conseguiram fazer bem a inferência estística do reator de van de Vusse. Elas são capazes de aprender as não linearidade do reator e suas propriedades, principalmente a inversão de ganho do reator de van de Vusse, o maior desafio ao controle do reator. O erro para predição em ciclo fechado é menor comparado ao modelo ARX; para a predição da produção de B, o erro da MLP-AT foi de aproximadamente 1/3 do erro da ARX; para C_b o erro da ESN foi metade do erro do modelo ARX. As redes também se mostraram estáveis, podendo ser usadas para predição de longos períodos de tempo. É difícil saber o porque a MLP-AT foi melhor que a ESN para predição em ciclo fechado. Nós supomos que no estado interno os ruídos se acumulam e o comportamento se torna bem mais errático

Tabela 11: Resumo dos resultados dos modelos de inferência estatística.

	RMSE		NRMSE		Tempo de treinamento (segundos)
ARX	C_b (kmol/m^3)	B (kmol/h)	C_b	B	
Passo seguinte	2.37E-03	2.148837	0.9902	0.9895	56
Ciclo fechado	0.382492	3.07E+02	-0.1532	-0.1193	
SS					
Passo seguinte	5.20E-02	5.159835	0.8421	0.9821	227
Ciclo fechado	DIVERGE				
ESN					
Passo seguinte	8.00E-03	7.3072	0.9761	0.9736	60
Ciclo fechado	0.2076	167.8895	0.4675	0.4899	
MLP-AT					
Passo seguinte	1.75e-2	15.72	0.947	0.943	400
Ciclo fechado	8.674e-2	77.789	0.739	0.718	

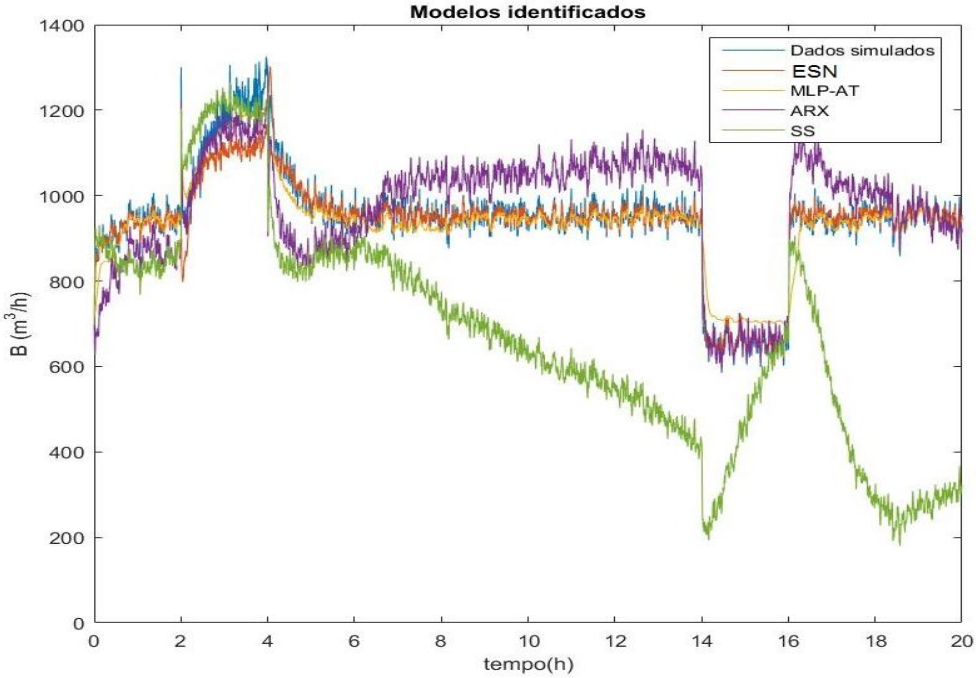

Figura 28: Todos os modelos de inferência estatística testados para a produção de B.

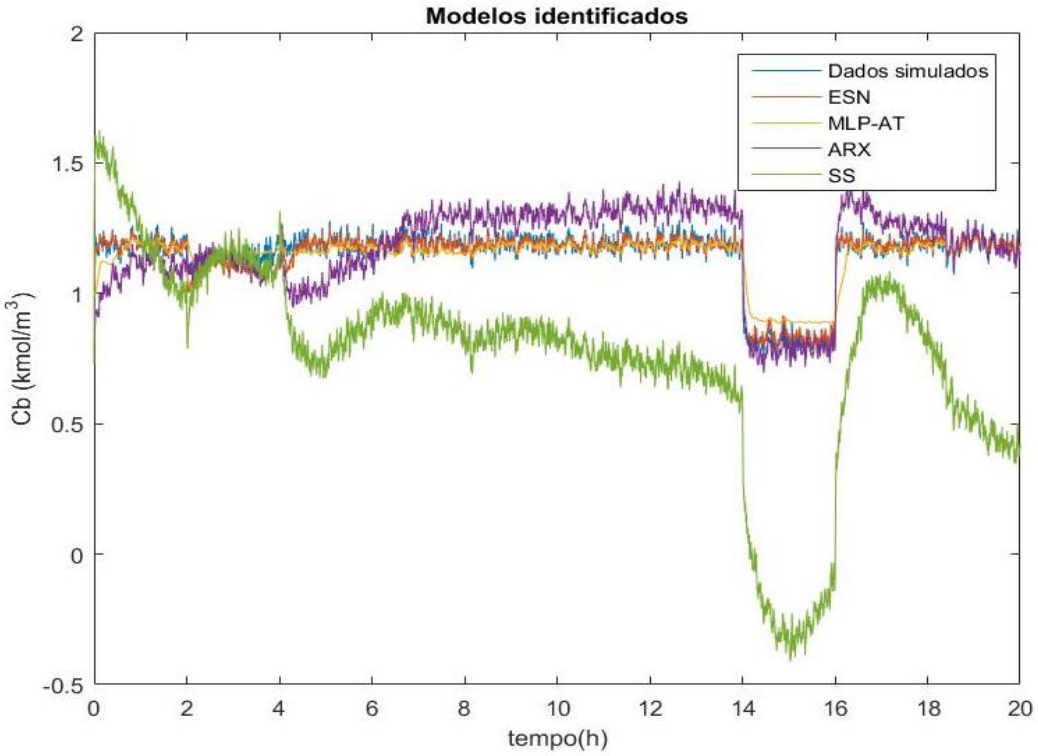

Figura 29: Todos os modelos de inferência estatística testados para a concentração de B.

4) Detecção e Diagnóstico de Falhas

Realizamos um estudo de caso para a aplicação de *Machine Learning* em DDF: detectar se uma perturbação está afetando o processo Tennessee Eastman e qual perturbação é essa. O Tennessee Eastman é um modelo muito usado para testar, além de DDF (CHIANG, *et al.* 2001), controle multivariável (FARINA, 2000), otimização (LARSSON, 2000) etc. É uma planta com não linearidades, atrasos e ruídos (DOWNS, & VOGEL, 1993).

4.1) Problema Tennessee Eastman

O problema Tennessee Eastman (TE) é um problema de controle proposto por Downs e Vogel (1993), muito usado como *benchmark* em estudo de controle, por ter um comportamento próximo ao de uma planta real completa e ser um problema complexo (DOWNS & VOGEL 1993; CHIANG, *et al.* 2001).

Neste trabalho utilizamos uma implementação em MATLAB do problema feita pelo Professor N. Lawrence Ricker, da Universidade de Washington, que implementou alterações ao modelo original, adicionando mais perturbações possíveis, melhorando a reprodutibilidade e aumentando o número de medidores no sistema (BATHELT, *et al.* 2014). A Figura 30 ilustra o processo atualizado.

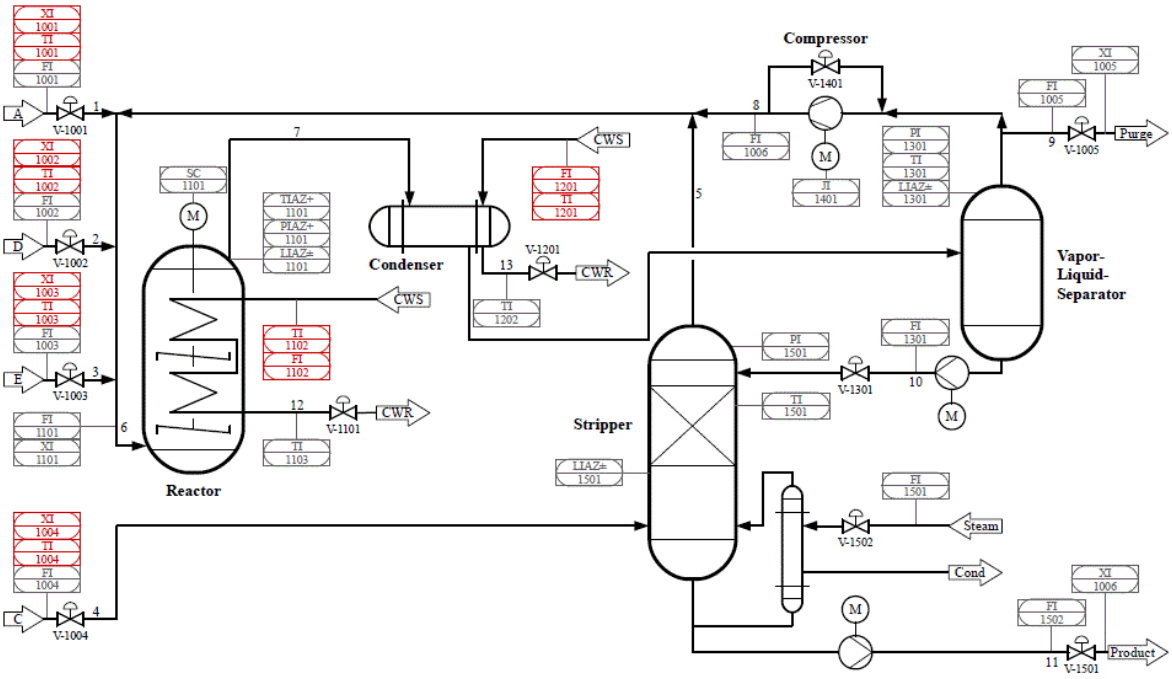

Figura 30: Fluxograma do processo Tennessee Eastman atualizado (BATHELT, *et al.* 2014).

Nessa planta os reagentes gasosos A, C, D e E e o inerte B são alimentados no reator onde os produtos líquidos G e H são formados, assim como o subproduto F. As reações são:

$$A_{(g)}+C_{(g)}+D_{(g)} \rightarrow G_{(liq)} \qquad \text{Eq. 4.1.1}$$

$$A_{(g)}+C_{(g)}+E_{(g)} \rightarrow H_{(liq)} \qquad \text{Eq. 4.1.2}$$

$$A_{(g)}+E_{(g)} \rightarrow F_{(liq)} \qquad \text{Eq. 4.1.3}$$

$$3D_{(g)} \rightarrow 2F_{(liq)} \qquad \text{Eq. 4.1.4}$$

As reações são exotérmicas e aproximadamente de primeira ordem, que seguem a equação de Arrhenius. A energia de ativação da reação de G é maior do que a da reação de H. A saída do reator é condensada e alimentada para um separador gas-líquido. A saída de topo do separador é reciclada através de um compressor, com uma certa fração purgada para evitar o acumulo de subproduto e inerte no processo. A saída de fundo do separador é bombeada para um *stripper* e a corrente 4 é utilizada como corrente de vapor para remover os reagentes que ficaram. A saída de fundo do *stripper* é enviada para outro processo, não incluído na simulação.

XMV são as variáveis diretamente manipuláveis, XMEAS são as variáveis medidas. As variáveis manipuladas são dadas na Tabela 12:

Tabela 12: Variáveis manipuláveis do Tennessee Eastman.

Variável	Descrição
XMV(1)	Vazão de entrada de D (corrente 2)
XMV(2)	Vazão de entrada de E(corrente 3)
XMV(3)	Vazão de entrada de A (corrente 1)
XMV(4)	Vazão de entrada total (corrente 4)
XMV(5)	Válvula de reciclo do compressor
XMV(6)	Válvula de purga (corrente 9)
XMV(7)	Vazão de saída de líquido do separador (corrente 10)
XMV(8)	Saída de líquido do stripper (corrente 11)
XMV(9)	Válvula do vapor do stripper
XMV(10)	Vazão da água de resfriamento do reator
XMV(11)	Vazão da água de resfriamento do condensador
XMV(12)	Velocidade do agitador

As variáveis medidas do processo estão na Tabela 13. A Tabela 14 contém as variáveis das medidas de composição. Essas medidas possuem um tempo morto equivalente ao intervalo de amostragem. A unidade é % molar.

Tabela 13: Variáveis medidas continuamente do Tennessee Eastman.

Variável	Descrição	Unidade
XMEAS(1)	Vazão de entrada de A (corrente 1)	kscmh
XMEAS(2)	Vazão de entrada de D (corrente 2)	kg/h
XMEAS(3)	Vazão de entrada de E (corrente 3)	kg/h
XMEAS(4)	Vazão de entrada total (corrente 4)	kscmh
XMEAS(5)	Vazão de reciclo (corrente 8)	kscmh
XMEAS(6)	Vazão de entrada do reator (corrente 6)	kscmh
XMEAS(7)	Pressão do reator	kpa
XMEAS(8)	Nível do reator	%
XMEAS(9)	Temperatura do reator	^{0}C
XMEAS(10)	Vazão de purga (corrente 9)	kscmh
XMEAS(11)	Temperatura do separador	^{0}C
XMEAS(12)	Nível do separador	%
XMEAS(13)	Pressão do separador	kPa
XMEAS(14)	Vazão de fundo do separador (corrente 10)	m^3/h
XMEAS(15)	Nível do stripper	%
XMEAS(16)	Pressão do stripper	kPa
XMEAS(17)	Vazão de fundo do stripper	m^3/h
XMEAS(18)	Temperatura do stripper	^{0}C
XMEAS(19)	Vazão de vapor do stripper	kg/h
XMEAS(20)	Trabalho do compressor	kw

Variável	Descrição	Unidade
XMEAS(21)	Temperatura da saída da água de resfriamento do reator	^{0}C
XMEAS(22)	Temperatura da saída da água de resfriamento do condensador	^{0}C

Tabela 14: Variáveis medidas discretas do Tennessee Eastman.

Variável	Descrição	Corrente	Intervalo de amostragem (minutos)
XMEAS(23)	Componente A	6	6
XMEAS(24)	Componente B	6	6
XMEAS(25)	Componente C	6	6
XMEAS(26)	Componente D	6	6
XMEAS(27)	Componente E	6	6
XMEAS(28)	Componente F	6	6
XMEAS(29)	Componente A	9	6
XMEAS(39)	Componente B	9	6
XMEAS(31)	Componente C	9	6
XMEAS(32)	Componente D	9	6
XMEAS(33)	Componente E	9	6
XMEAS(34)	Componente F	9	6
XMEAS(35)	Componente G	9	6
XMEAS(36)	Componente H	9	6
XMEAS(37)	Componente D	11	15
XMEAS(38)	Componente E	11	15
XMEAS(39)	Componente F	11	15
XMEAS(40)	Componente G	11	15
XMEAS(41)	Componente H	11	15

As variáveis na Tabela 15 foram adicionadas no modelo atualizado pelo professor Ricker. Elas não foram utilizadas na detecção e diagnóstico de falha, pois, uma vez que elas são medidas, acreditamos que ficaria muito claro quando as falhas ocorrem, facilitando a solução do problema.

Tabela 15: Variáveis medidas adicionadas ao Tennessee Eastman.

Variável	Descrição	Unidade
XMEAS(42)	Temperatura da entrada de A (corrente 1)	^{0}C
XMEAS(43)	Temperatura da entrada de D (corrente 2)	^{0}C
XMEAS(44)	Temperatura da entrada de E (corrente 3)	^{0}C
XMEAS(45)	Temperatura da entrada total (corrente 4)	^{0}C
XMEAS(46)	Temperatura de entrada da água de resfriamento do reator	^{0}C
XMEAS(47)	Vazão da água de resfriamento do reator	m^3/h
XMEAS(48)	Temperatura de entrada da água de resfriamento do condensador	^{0}C
XMEAS(49)	Vazão da água de resfriamento do condensador	m^3/h
XMEAS(50)	Composição de A na corrente 1	% molar
XMEAS(51)	Composição de B na corrente 1	% molar
XMEAS(52)	Composição de C na corrente 1	% molar
XMEAS(53)	Composição de D na corrente 1	% molar
XMEAS(54)	Composição de E na corrente 1	% molar
XMEAS(55)	Composição de F na corrente 1	% molar
XMEAS(56)	Composição de A na corrente 2	% molar
XMEAS(57)	Composição de B na corrente 2	% molar
XMEAS(58)	Composição de C na corrente 2	% molar
XMEAS(59)	Composição de D na corrente 2	% molar
XMEAS(60)	Composição de E na corrente 2	% molar
XMEAS(61)	Composição de F na corrente 2	% molar
XMEAS(62)	Composição de A na corrente 3	% molar
XMEAS(63)	Composição de B na corrente 3	% molar
XMEAS(64)	Composição de C na corrente 3	% molar
XMEAS(65)	Composição de D na corrente 3	% molar
XMEAS(66)	Composição de E na corrente 3	% molar
XMEAS(67)	Composição de F na corrente 3	% molar
XMEAS(68)	Composição de A na corrente 4	% molar
XMEAS(69)	Composição de B na corrente 4	% molar
XMEAS(70)	Composição de C na corrente 4	% molar
XMEAS(71)	Composição de D na corrente 4	% molar
XMEAS(72)	Composição de E na corrente 4	% molar
XMEAS(73)	Composição de F na corrente 4	% molar

O problema TE original possui 20 pertubações pré-programadas, expandidas para 28 no processo atualizado. Elas são adicionadas ao processo como entradas do modelo, um vetor com booleanos e numéricos. Na Tabela 16 as descrições das perturbações são apresentadas.

Tabela 16: Perturbações programadas no Tennessee Eastman.

Perturbação	Descrição	Tipo
IDV(1)	Razão de entrada A/C composição de B constante (corrente 4)	Degrau
IDV(2)	Composição de B, razão A/C constante (corrente 4)	Degrau
IDV(3)	Temperatura de entrada de D (corrente 2)	Degrau
IDV(4)	Temperatura de entrada da água de resfriamento do reator	Degrau
IDV(5)	Temperatura de entrada da água de resfriamento do condensador	Degrau
IDV(6)	Perda de entrada de A (corrente 1)	Degrau
IDV(7)	Queda de pressão de C, disponibilidade reduzida (corrente 4)	Degrau
IDV(8)	composição da entrada de A,B e C (corrente 4)	Variação randômica
IDV(9)	Temperatura de entrada de D (corrente 2)	Variação randômica
IDV(10)	Temperatura de entrada de C (corrente 4)	Variação randômica
IDV(11)	Temperatura de entrada da água de resfriamento do reator	Variação randômica
IDV(12)	Temperatura de entrada da água de resfriamento do condensador	Variação randômica
IDV(13)	Cinética da reação	Desvio lento
IDV(14)	Válvula de água de resfriamento do reator	Agarramento
IDV(15)	Válvula de água de resfriamento do condensador	Agarramento
IDV(16)	Desconhecido	
IDV(17)	Desconhecido	
IDV(18)	Desconhecido	
IDV(19)	Desconhecido	
IDV(20)	Desconhecido	
IDV(21)	Temperatura de entrada de A (corrente 1)	Variação randômica
IDV(22)	Temperatura de entrada de E (corrente 3)	Variação randômica
IDV(23)	Pressão de entrada de A (corrente 1)	Variação randômica
IDV(24)	Pressão de entrada de D (corrente 2)	Variação randômica
IDV(25)	Pressão de entrada de E (corrente 3)	Variação randômica
IDV(26)	Pressão de entrada de A e C (corrente 4)	Variação randômica
IDV(27)	flutuação de pressão da unidade de recirculação de água de resfriamento do reator	Variação randômica
IDV(28)	flutuação de pressão da unidade de recirculação de água de resfriamento do condensador	Variação randômica

Uma adição interessante ao TE é que ele possui um calculador de custo operacional, sendo possível realizar otimização econômica do processo. Uma otimização baseado em

Machine Learning pode ser um projeto no futuro. Neste trabalho usaremos esse custo operacional para ilustrar a variabilidade do processo.

Diversas soluções para o problema de controle foram propostas ao longo dos anos, com vantagens e desvantagens cada uma. Neste trabalho usaremos a estratégia de controle proposta por pelo grupo do Professor Skogestad (LARSSON, *et al.* 2001). Os controles simplificados, com uma variável manipulada e uma controlada estão representados na Tabela 17.

Tabela 17: Variáveis controladas e manipuladas diretamente pelo sistema de controle.

Variável controlada	**Variável manipulada**
Nível do stripper XMEAS(15)	Saída de líquido do stripper XMV(8)
Nível do separador XMEAS(12)	Vazão de saída de líquido do separador XMV(7)
Nível do reator XMEAS(8)	Vazão da água de resfriamento do condensador XMV(11)
Pressão do reator XMEAS(7)	Válvula de purga XMV(6)
% molar de C XMEAS(31)	Entrada de C, XMV(4)
Taxa de reciclo XMEAS(5)	Entrada de A, XMV(3)
Temperatura do reator XMEAS(9)	Vazão da água de resfriamento do reator XMV(10)

O controle do nível de produção e o controle da % de G na saída do produto são feitos controlando simultaneamente a vazão de entrada de D, XMV(1), e E, XMV(2). A estratégia completa de controle segue na Tabela 18. A taxa de amostragem do controlador é de 5e-4 ms. Note que na Tabela 18 Fp é uma variável manipulada.

Tabela 18: Ciclos de controle do processo.

Ciclo	**Variável manipulada**	**Variável controlada**	**Ganho R – ação reversa; D – ação direta**	**Tempo integral (h)**
1	XMV(3)	XMEAS(1)	0.01*Fp (R)	0.001/60
2	XMV(1)	XMEAS(2)	1.6e-6*Fp (R)	0.001/60
3	XMV(2)	XMEAS(3)	1.8e-6*Fp (R)	0.001/60
4	XMV(4)	XMEAS(4)	0.003*Fp (R)	0.001/60
5	XMV(6)	XMEAS(10)	0.01*Fp (R)	0.001/60
6	XMV(7)	XMEAS(14)	4e-4*Fp (R)	0.001/60
7	XMV(8)	XMEAS(17)	4e-4*Fp (R)	0.001/60
8	Fp	XMEAS(17)	3.2 (R)	120/60
9	XMEAS(17)	XMEAS(15)	2e-4 (D)	200/60
10	XMEAS(14)	XMEAS(12)	1e-3 (D)	200/60
11	XMEAS(11)	XMEAS(8)	0.8 (R)	60/60
12	XMEAS(10)	XMEAS(7)	1e-4 (D)	20/60
13	Eadj	XMEAS(40)	0.032 (D)	100/60
14	XMEAS(4)	XMEAS(31)	0.0009 (R)	562/60
15	XMEAS(1)	XMEAS(5)	0.00125 (R)	120/60

Ciclo	Variável manipulada	Variável controlada	Ganho R – ação reversa; D – ação direta	Tempo integral (h)
16 17	XMV(10) XMV(11)	XMEAS(9) XMEAS(11)	8 (D) 4 (D)	7.5/60 15/60

Os *setpoints* de XMEAS(2) e XMEAS(3) são calculados a partir das seguintes fórmulas:

$$SP_{xmeas2} = \left(1.5192e3 * SP_g^2 + 5.9446e - 1 * SP_g + 2.769e - 1\right) - \frac{Eadj * 32}{Fp} \qquad \text{Eq 4.1.5}$$

$$SP_{xmeas3} = \left(-1.1377e - 3 * SP_g^2 - 8.0893e - 1 * SP_g + 91.06\right) - \frac{Eadj * 46}{Fp} \qquad \text{Eq. 4.1.6}$$

Os controladores são discretos, seguindo as equações

$$C_k = K_p * \left(e * \left(1 + \frac{T_s}{T_i} - z\right)\right) \qquad \text{Eq. 4.1.7}$$

$$U_k = U_{k-1} + C_k \qquad \text{Eq. 4.1.8}$$

tal que K é o ganho do controlador e T_i é o tempo de integração, T_s é a taxa de amostragem, *e* é o erro dado pela diferença entre *setpoint* e a variável medida

O processo Tennessee Eastman pode operar em 6 modos, descritos na Tabela 19. Nesse trabalho apenas o modo 1 é utilizado, mas o esquema de controle permite passar para qualquer um dos modos de operação.

Tabela 19: Modos de operação do TE.

Modo	razão G/H	Taxa de produção
1	50/50	7038 kg/h G&H
2	10/90	1048 kg/h G 12669 kg/h H
3	90/10	10000 kg/h G 1111 kg/h H
4	50/50	Máximo
5	10/90	Máximo
6	90/10	Máximo

O modelo do Tennessee Eastman foi implementado em Simulink 6.0 e teve simulados 200 dias de operação. O sistema começa sem perturbações e, depois de um intervalo aleatório entre 18 e 36 horas, ocorre uma perturbação, depois de um intervalo aleatório de 18 e 36 horas

a perturbação cessa. O intervalo aleatório das perturbações é necessário senão os modelos de *Machine Learning* podem aprender apenas a contar um certo tempo ao invés de prever a existência ou não de perturbações. As perturbações que ocorrem são aleatórias, cada uma das 28 perturbações disponíveis pode ocorrer com igual probabilidade, independentemente se a mesma perturbação ocorreu antes. Um total de 140 dias de operação foi alocado como dados de treino, 47040 padrões; 30 dias como dados de validação e 30 dias como dados de teste, 10080 padrões cada. Os dados de treino foram estandardizados para terem média zero e variância 1 e os dados de validação e testes foram estandardizados com os mesmos parâmetros. Foram testadas várias arquiteturas de redes para classificação: Convolução, redes recorrentes simples e *gated*. Também foi testado o uso de MVS de *kernel* radial e linear. Para comparação com os modelos de *Machine Learning* foram usados PCA dinâmico para detecção de falha e FDA dinâmico para diagnóstico de falha.

Existem muitos trabalhos que estudam a classificação de falhas no Tennessee Eastman baseada em dados: CHIANG, *et al.* (2001) usaram diversas técnicas baseadas em dados com o objetivo de fazer uma análise comparativa entre os métodos: variantes de PCA, FDA, análise de variáveis canônica e mínimos quadrados parciais. NASHALHI, *et al.* (2010) utilizaram um PCA melhorado por algoritmo genético para pré-tratamento dos dados e redes neuronais para fazer a detecção de falha. RATO & REIS (2013) utilizaram uma PCA dinâmico com resíduos de correlacionados. RAGAB, *et al.* (2017) testaram um novo método de *Machine Learning*, análise lógica de dados, e o comparou com redes neuronais e máquinas de vetores de suporte.

A principal diferença deste trabalho para os outros trabalhos é a geração dos dados. Todos os outros trabalhos realizam diferentes corridas para cada perturbação, com o mesmo tempo de simulação. Este trabalho utiliza uma corrida contínua em que diferentes perturbações são inseridas e removidas ao longo do tempo, para simular melhor o período de operação de uma planta, em que paradas na operação são raras e caras. Nós também não controlamos quais perturbações ocorrem e por quanto tempo, isso afeta a estimativa de probabilidade obtida a partir dos dados de treino. Em contrapartida, não é possível saber como o sistema classifica certas perturbações que não ocorrem na base de teste e validação.

Uma questão interessante desse método de simulação é o retorno ao estado estacionário. É possível em um sistema complexo que o estado estacionário encontrado depois de uma perturbação seja diferente do estado estacionário inicial. Isso é comum em plantas industriais, porém não é um fenômeno avaliado pelos outros trabalhos. Outro ponto é a utilização do *Big Data*. Enquanto os outros trabalhos criam diferentes bases de dados para

cada falha e trabalham com cada base de dados separadamente, nós usamos uma única base de dados com 227 MB.

A simulação do modelo do Tennessee Eastman foi amostrado a cada 5 segundos, depois a base de dados foi reduzida para 30 segundos usando um filtro exponencial com fator de amortização de 70%. As variáveis mantidas constantes durante a operação foram eliminadas: XMV(5), a posição da válvula de reciclo, XMV(9), a posição da válvula de vapor do stripper e XMV(12), o ajuste do agitador. Para o FDA e PCA dinâmicos e MVS, foi alimentada uma matriz com 1 hora de dados atrasados, sendo concatenado à matriz original os dados com 3 minutos de atraso por vez, deixando a matriz com 1000 colunas. As redes neuronais são próprias para receberem dados temporais, portanto respondem bem a uma janela temporal maior, tendo sido usadas janelas de 2 horas. Para o PCA e o FDA, o número de componentes usados foram os que corresponderam a 95% da variância, 450 componentes de 1000. Os dados foram separados a partir do momento que ocorre a falha.

Os cálculos dos modelos lineares foram feitos em MATLAB, os modelos de *Machine Learning* foram implementados em Python, pois Python lida bem melhor com modelos que utilizam muita memória e porque possui bibliotecas com recursos bem mais avançados que MATLAB. As redes neuronais foram criadas usando a biblioteca Keras (CHOLLET, 2015) e as MVS foram criadas usando o pacote Sklearn (PEDREGOSA, *et al*. 2011). Os métodos de *Machine Learning* foram aplicados usando um PC com Linux Ubuntu 16.04, 16GB de ram, processador i5-6500 de 4 núcleos com 3.2 GHz de processamento e placa de vídeo GeForce GTX 1070, 1GB de ram. Para identificar os melhores hiperparâmetros para os modelos, foi usado busca em grade para as MVS, e busca randômica para as redes, pois as redes neuronais possuem muitos parâmetros para serem treinados e um tempo enorme de treino. Foram treinadas redes profundas, com até 6 camadas, e redes largas, com até 200 neurônios. Para os modelos lineares e as MVS foi avaliada a acurácia dos modelos.

Para as redes neuronais, além da acurácia foi avaliada a entropia cruzada dos modelos. Essa métrica é importante pois as redes retornam uma probabilidade, e uma rede que dá mais certeza é uma rede mais confiável. Outra métrica avaliada foi a curva ROC (Característica de Operação do Receptor). Ela é obtida pela representação da fração de positivos verdadeiros dos positivos totais versus a fração de positivos falsos dos negativos totais, em várias configurações do limite de classificação. A vantagem da curva ROC é que ela pode ser usada para avaliar a qualidade dos modelos em diversos níveis de sensibilidade, ao invés de dar um resultado binário de sim ou não. Outra métrica da curva ROC é a área da curva, que varia de 1, predição perfeita a 0.5, predição randômica (HASTIE, *et al.* 2008) .

4.2) Resultados

Para demonstrar a variância do processo durante a simulação, usamos um gráfico do custo operacional, citado anteriormente ao longo do tempo, Figura 31, pois o custo operacional combina diversos valores do processo. É possível observar picos e vales, alguns ocorrendo frequentemente, outros isolados; regiões de maior variabilidade do custo e um momento próximo ao dia 125 em que o sistema ficou com uma média de custo maior.

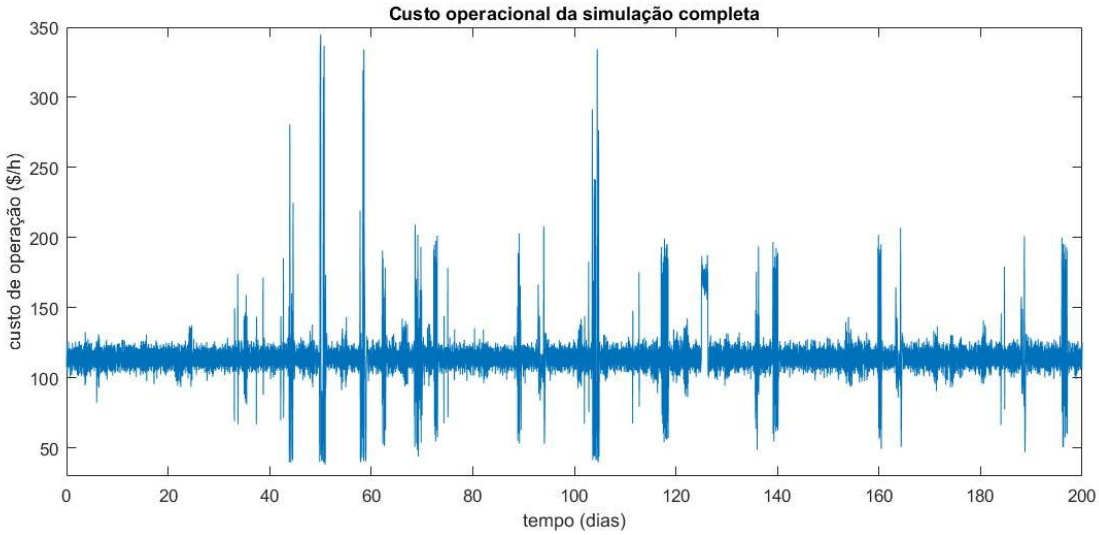

Figura 31: Custo operacional durante toda a simulação.

Algumas perturbações são facilmente identificáveis, outras não. Nem sempre o nível de uma perturbação será o mesmo da seguinte, pois o gerador de números randômicos usado internamente pelo modelo não é fixado. Como pode se observar na Figura 32, o efeito da falha 1, variação da razão A/C, mantendo B constante, é bem claro, exibindo uma resposta ao degrau clássica na entrada de A, resultando em um valor 3 vezes maior que o de base. Na falha 22, variação da temperatura de E na entrada, a diferença é muito pequena, sendo quase imperceptível perto do ruído da temperatura, como pode ser visto na Figura 33. No apêndice IV encontram-se os valores utilizados na estandardização dos dados.

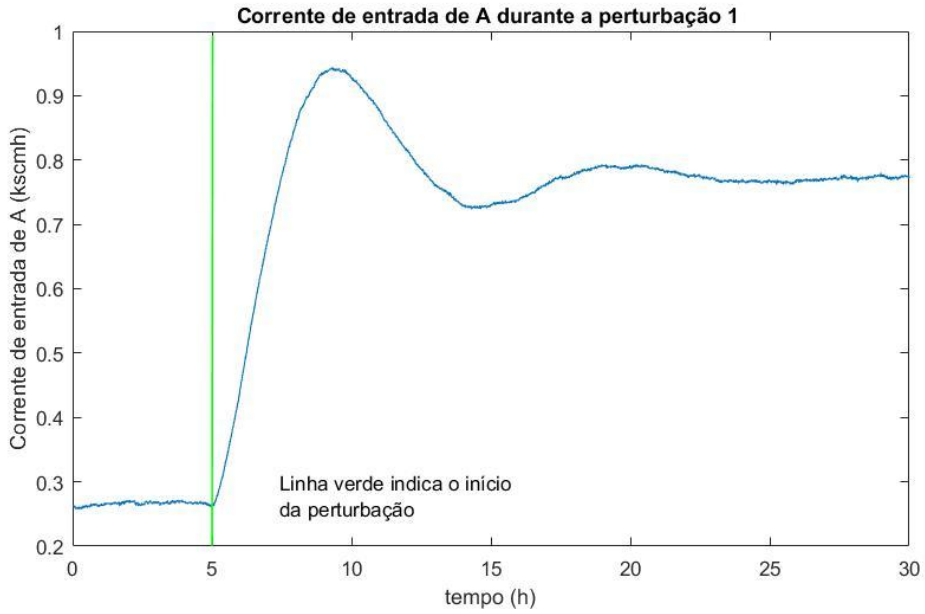

Figura 32: Efeito da falha 1, variação da razão de A/C, na corrente de entrada de A.

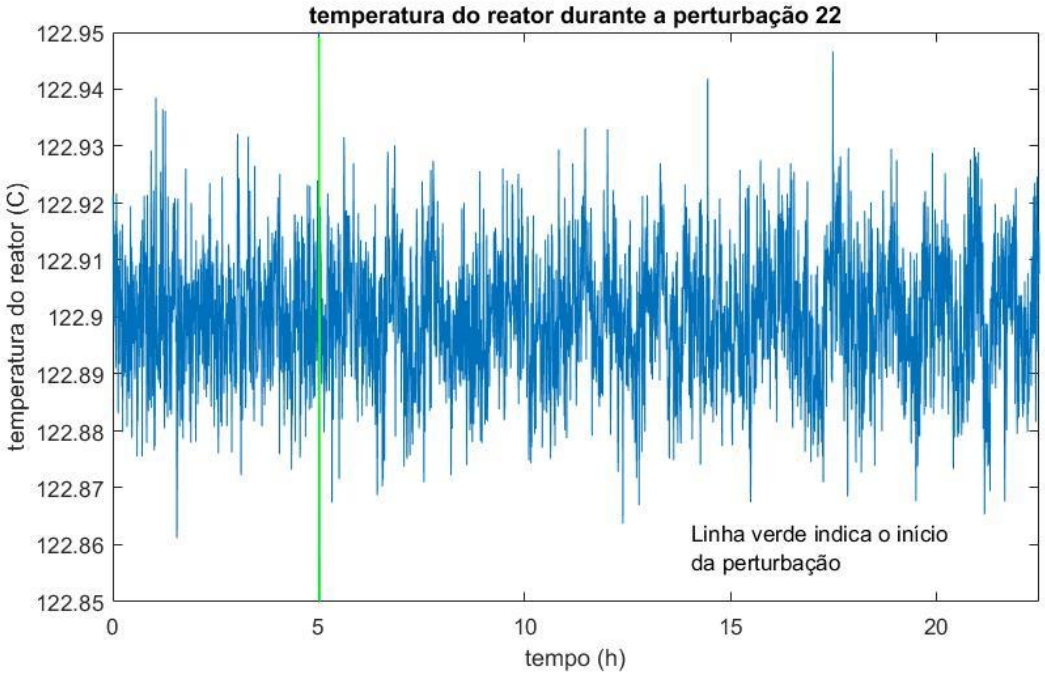

Figura 33: efeito da falha 22, variação da temperatura da corrente de entrada de E, na temperatura do reator.

4.2.1) PCA e FDA dinâmico

A Figura 34 demonstra o quanto da variância cumulativa da base de dados é explicada por cada componente principal. Nós usamos 450 componentes principais que correspondem a aproximadamente 95% da variância, descartando 550 componentes principais.

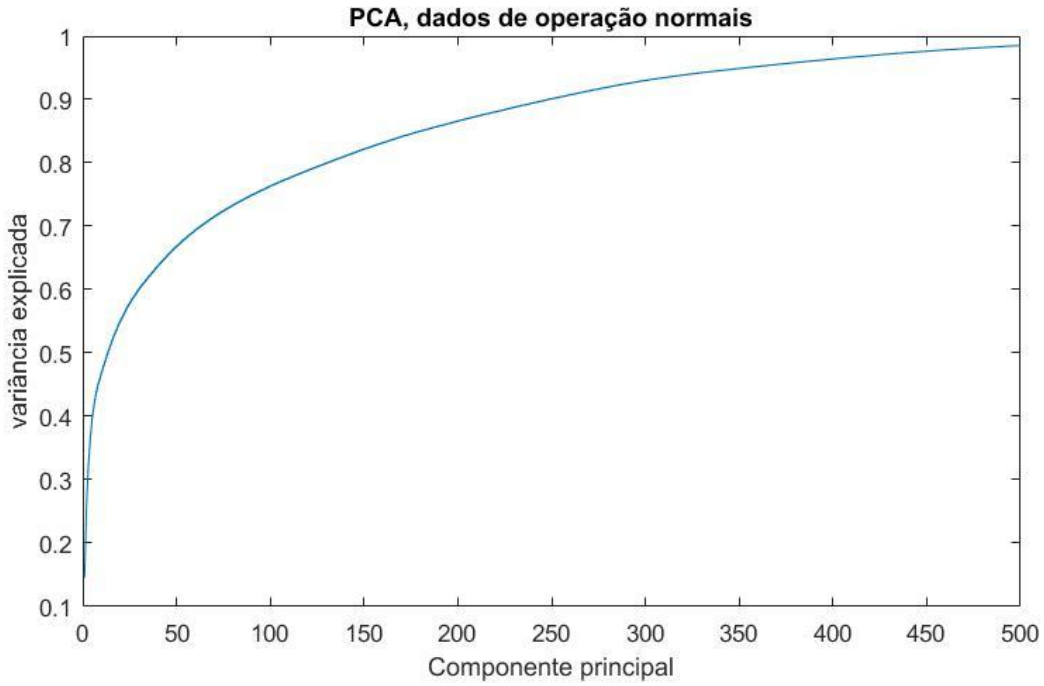

Figura 34: Variância cumulativa explicada por cada componente principal.

A Figura 35 demonstra como é o comportamento da estatística de Hotelling ao longo do tempo. Às vezes ela continua estável quando ocorrem perturbações; noutras, ela dispara quando elas acontecem.

O principal problema da detecção de falhas dos métodos lineares são os falsos negativos, que ocorrem quando o sistema está sob falha, mas o método não detecta a perturbação. A taxa de acerto da base de dados de treino é de 71,9% como pode ser visto na Tabela 20. A acurácia de detecção de falha é de 68.1% na base de dados de teste e validação, como pode ser visto na Tabela 21. Uma certa taxa de erro é esperada, pois demora para o processo voltar ao normal. Para diagnóstico de falha a acurácia foi de 64.0%.

A área da curva ROC, Figura 36 foi baixa, apenas 59,7%, ligeiramente melhor que predição randômica.

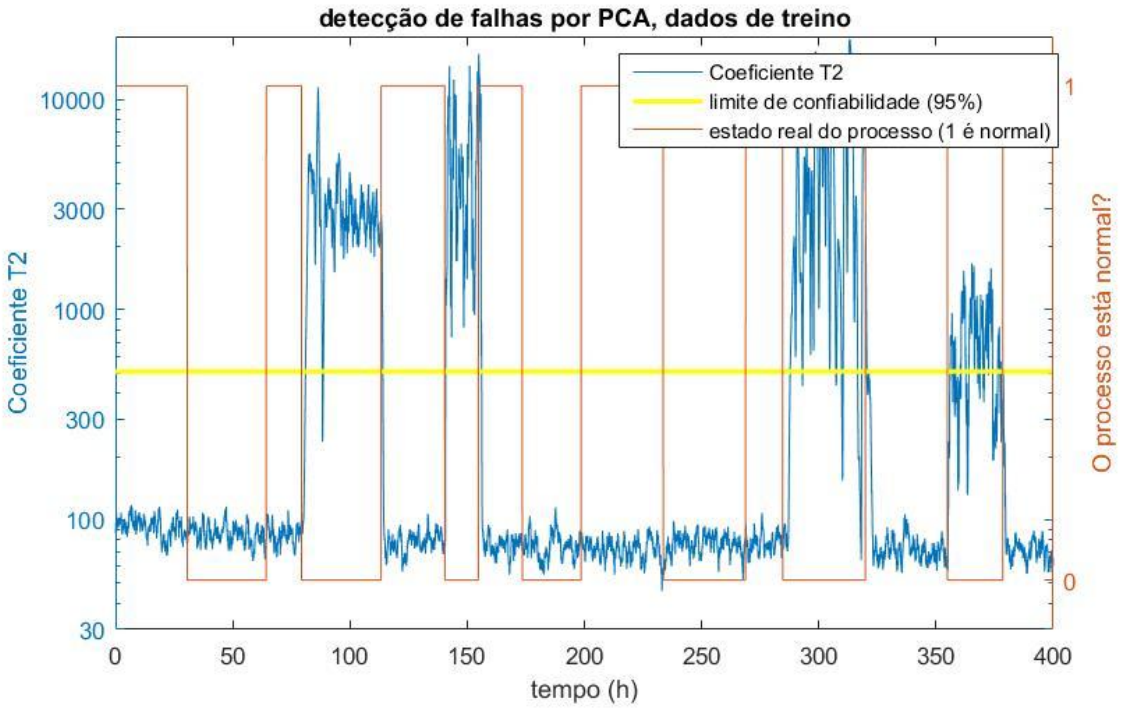

Figura 35: Coeficiente de Hotelling em função do tempo, na presença ou ausência de perturbações.

Tabela 20: Detecção de falha através de PCA, dados de treino. Acurácia de 71.90%.

O sistema está normal? Dados de treino		
	predito	
real	0	1
0	17426	16564
1	2293	30897

Tabela 21: Detecção de falha através de PCA, dados de teste e validação. Acurácia de 68.06%.

O sistema está normal? Dados de teste e validação		
	predito	
real	0	1
0	5239	8218
1	972	14352

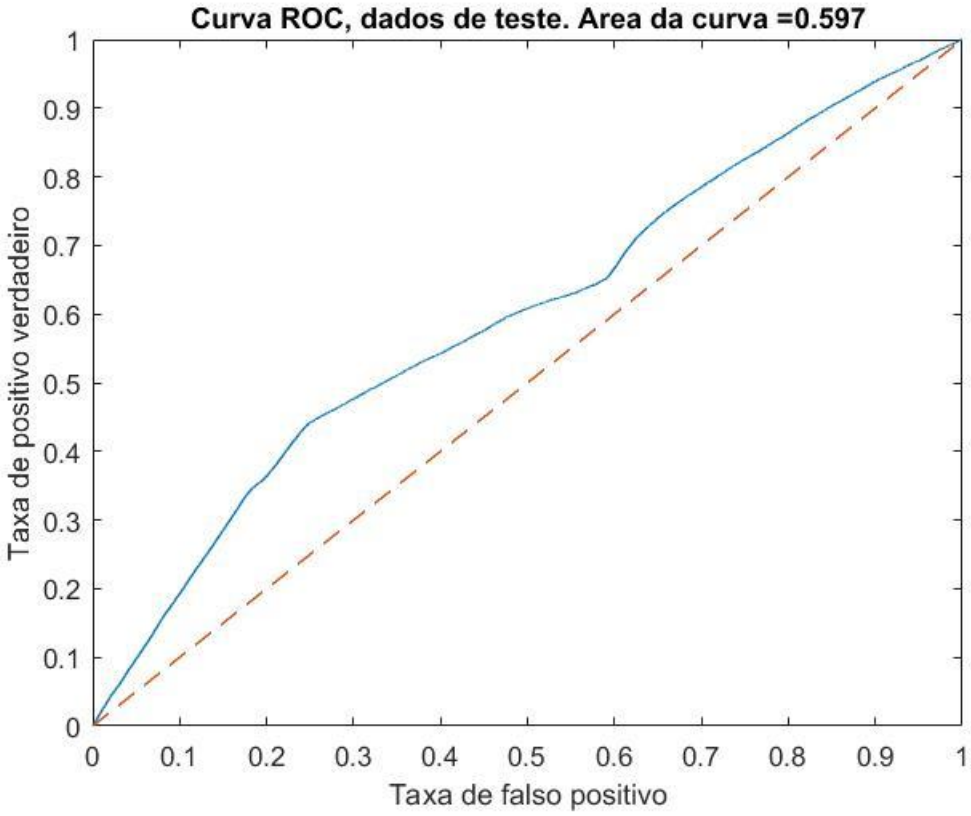

Figura 36: Curva ROC da detecção de falha por PCA.

A Tabela 22 é uma matriz de confusão, as linhas representam a classe do dado e as colunas a predição do modelo. Na Tabela 22 os quadrinhos verdes indicam os acertos. Linhas e colunas contendo apenas 0 foram excluídas para melhor formatação. A classe 0 indica operação normal. É interessante notar que as falhas classe 3, 6, 14, 15, 19, 21, 22, 23, 26 e 28 não foram detectados, sendo que dessas apenas a classe 23 não estava presente nos dados de treino. O tempo de treinamento do PCA dinâmico foi de 57 segundos, o FDA dinâmico demorou 127 segundos.

Tabela 22: Diagnóstico de falha através de FDA dinâmico.

		Predito															
		0	1	4	5	7	8	9	10	12	13	17	18	20	23	25	27
Real	0	14115	74	10	12	10	91	1	61	13	0	44	125	144	2	67	555
	1	69	628	0	0	0	0	0	0	0	15	0	0	0	0	0	0
	3	487	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	11	0	299	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	14	0	0	435	0	0	0	0	0	0	0	0	0	0	0	0
	6	759	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	10	0	0	0	318	0	0	0	0	0	0	0	0	0	0	0
	10	382	0	0	0	0	0	0	290	0	0	0	0	0	0	0	0
	14	1179	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	15	1476	0	0	0	0	0	0	0	0	0	0	0	0	63	0	0
	17	110	0	0	0	0	0	0	0	0	0	272	0	0	0	0	0
	18	200	0	0	0	0	0	0	0	0	0	0	370	0	0	0	0
	19	275	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	20	110	0	0	0	0	0	0	0	0	0	0	0	668	0	0	0
	21	930	0	0	0	0	0	0	0	0	0	0	0	0	0	0	176
	22	355	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	455	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
	25	961	0	0	0	0	0	0	0	0	0	0	0	0	3	643	0
	26	1097	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0
	28	387	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

4.2.2) Redes neuronais recorrentes

As melhores redes encontradas foram as largas e rasas, com apenas uma camada oculta com muitos neurônios. LSTM se mostrou bem difícil de treinar, portanto GRUs foram usadas. A melhor rede foi uma contendo apenas uma camada GRU de 100 neurônios. Redes recorrentes simples também obtiveram bons resultados caso usem redes bem largas, com mais de 150 neurônios, mas as melhores foram as GRU. A Tabela 23 contém os hiperparâmettros utilizados na melhor rede recorrente.

Tabela 23: Hiperparâmetros da rede recorrente GRU.

Dropout	30%
Número de neurônios	100
Número de pesos	45.519
Termo de regularização l2	$5*10^{-3}$
Épocas máximas	200
Taxa de aprendizagem	0.0015
Fator de decaimento da taxa de aprendizagem	8e-3
Método de treinamento	RMSprop
Não linearidade	Relu

A taxa de acerto para a detecção de falhas foi de 76.65%, como pode ser visto na Tabela 24, um valor superior à base, que é de 68.1%. A Figura 37 é o gráfico da saida da rede recorrente para detecção de falha. Vale lembrar que a saída da rede recorrente é uma probabilidade, isso é, qual a chance de do processo estar realmente perturbado. É interessante notar que a rede geralmente é insegura, raramente dando mais de 90% de certeza que o processo está normal. A curva ROC, Figura 38, possui uma área de 80.5%.

A taxa de acerto para diagnóstico de falhas é de 72.30%, como pode ser visto na Tabela 25, uma boa melhora em relação ao FDA dinâmico, de apenas 64% de acerto. O treinamento da RN recorrente demorou em média 2560 segundos

Tabela 24: Detecção de falha para os dados de teste e validação.

O sistema está normal?		
	predito	
real	0	1
0	8699	4759
1	1962	13363

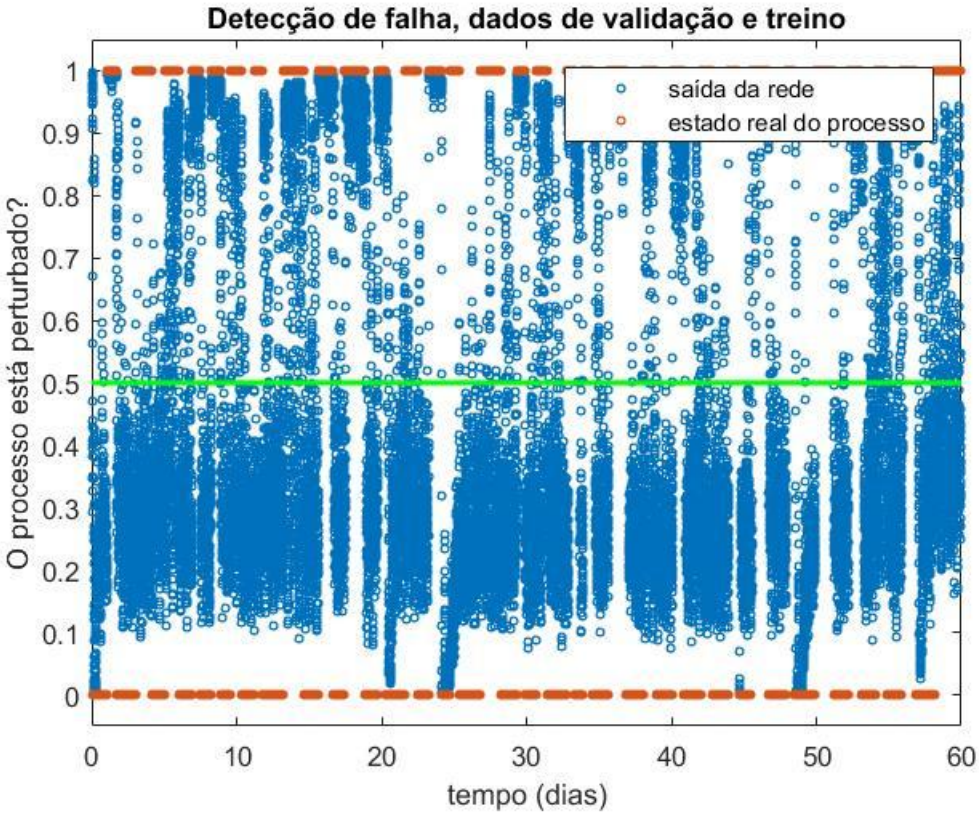

Figura 37: Saída da rede neuronal recorrente. Dados de validação e treino.

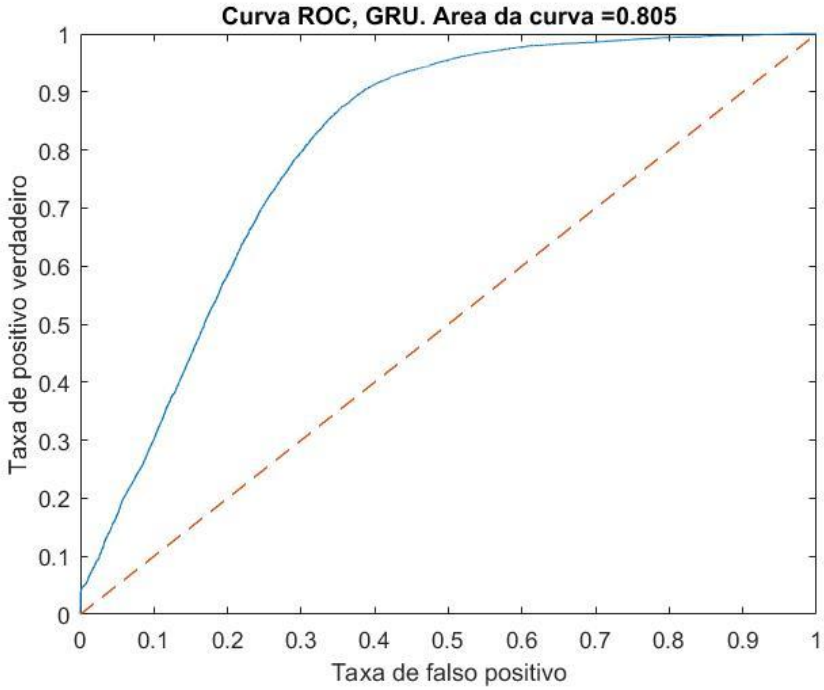

Figura 38: Curva ROC da rede recorrente para os dados de treino.

Tabela 25: Diagnostico de falha para os dados de teste e validação da RN Recorrente.

		Predito														
		0	1	4	7	8	10	12	13	14	17	18	20	25	26	27
Real	0	14318	21	1	1	39	72	19	6	26	76	246	184	189	49	78
	1	14	697	0	0	2	0	0	0	0	0	0	0	0	0	0
	3	487	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	1	0	309	0	0	0	0	0	0	0	0	0	0	0	0
	5	269	0	0	0	0	0	166	0	0	0	0	0	0	0	14
	6	758	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	1	0	0	327	0	0	0	0	0	0	0	0	0	0	0
	10	136	0	0	0	15	489	16	0	0	0	0	0	0	0	16
	14	25	0	2	0	0	1	0	0	1125	0	0	0	0	0	27
	15	1540	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	17	72	0	1	0	0	0	0	0	0	309	0	0	0	0	0
	18	90	0	0	0	0	0	0	0	0	0	479	0	0	0	0
	19	144	0	0	0	0	0	125	3	0	0	0	0	0	4	1
	20	75	0	0	0	0	0	0	0	0	0	0	703	0	0	0
	21	1076	0	0	0	5	0	0	0	0	0	0	0	0	0	25
	22	355	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	457	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	126	0	0	0	0	0	15	0	0	0	0	0	1466	0	0
	26	515	0	0	0	0	0	0	0	0	0	0	0	0	587	1
	28	386	0	0	0	0	0	1	0	0	0	0	0	0	0	0

4.2.3) Redes neuronais convolucionais

As redes convolucionais foram as mais difíceis de treinar. Elas ficam presas em mínimos locais facilmente e exigiam o cálculo de muitos gradientes. Foi necessário um certo ajuste nos hiperparâmetros recomendados, pois diferentemente de imagens, os vetores são densos em informação e retangulares, o tamanho temporal da sequência é bem maior que o número de sensores, por isso a janela de convolução da rede também é retangular e os passos são retangulares também. A Tabela 26 contem os hiperparâmetros utilizados no treinamento da rede convolucional

Tabela 26: Hiperparâmetros da rede convolucional.

Dropout	30%
Número de parâmetros	240715
termo de regularização l2	$5*10^{-3}$
épocas máximas	300
taxa de aprendizagem	0.005
Fator de decaimento da taxa de aprendizagem	4e-3
Método de treinamento	ADAM
Não linearidade	Relu

A arquitetura da maior rede encontrado foi: 3 camadas convolucionais, uma *max pooling*, um *batch normalization* e uma camada densa plenamente conectada. A primeira camada tem 5 filtros, janela de convolução 8x3 e passo 3x1; A segunda camada tem 10 filtros, janela de convolução 6x2 e passo 2x1. A terceira camada tem 12 filtros, janela de convolução 3x2e passo 2x1. A última camada antes da saída tem 40 neurônios.

A taxa de acerto para a detecção foi de 74.15%, como pode ser visto na Tabela 27. Como pode se ver na Figura 39, a falta de confiança em dizer se o processo está normal se mostrou ainda mais pronunciada na rede convolucional. Isso provavelmente é o resultado de um mínimo local em que a rede ficou presa; mesmo assim a acurácia é boa. Essa falta de confiança pode ser vista no desvio da curva ROC, Figura 40, no ponto (0,9; 0,4), onde a taxa de falsos positivos aumenta subitamente. A entropia cruzada média é de 0.276. A acurácia de diagnóstico de falha foi de 70.71%, como pode ser visto na Tabela 31. O tempo de treinamento da RN convolucional demorou em média 1184 segundos.

Tabela 27: Detecção de Falha com a Rede Neuronal Convolucional. Dados de teste e validação.

O sistema está normal?		
	predito	
real	0	1
0	7067	6391
1	1048	14277

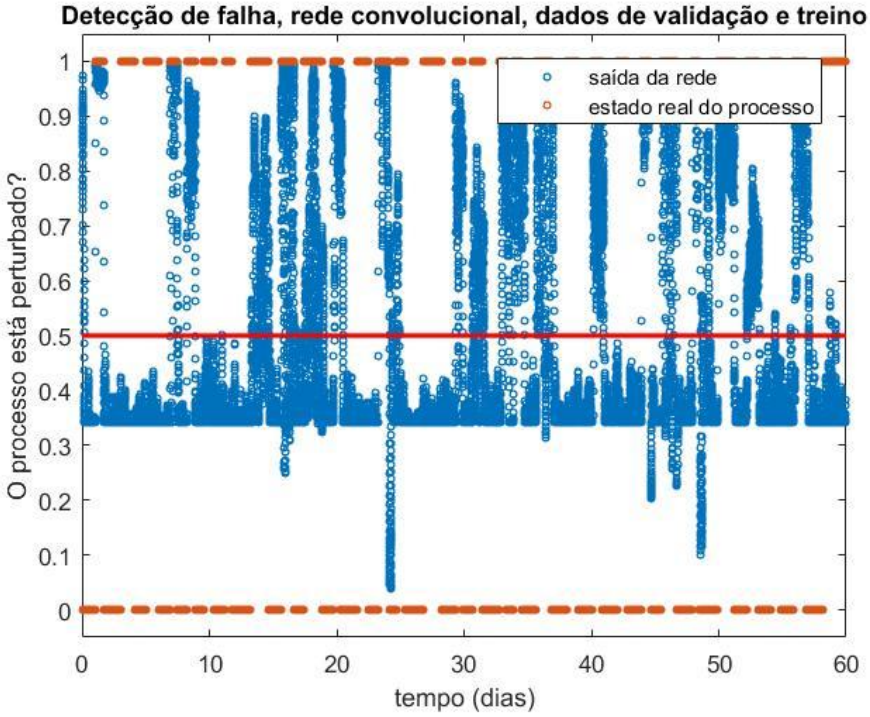

Figura 39: Saída da rede convolucional.

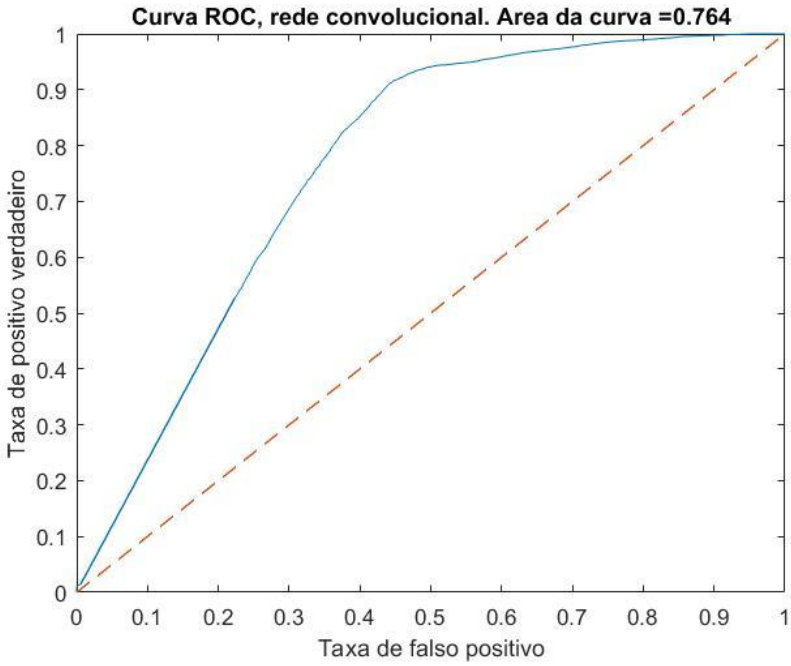

Figura 40: Curva ROC da rede convolucional.

Tabela 28: Diagnóstico de falha para os dados de teste e validação da Rede Neuronal convolucional.

		Predito														
		0	1	4	6	7	8	10	12	14	17	18	19	20	25	26
Real	0	14025	68	24	11	6	196	85	25	85	83	242	28	268	179	0
	1	22	661	0	0	0	30	0	0	0	0	0	0	0	0	0
	3	487	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	6	0	304	0	0	0	0	0	0	0	0	0	0	0	0
	5	27	0	0	0	0	0	0	422	0	0	0	0	0	0	0
	6	758	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	4	0	0	0	324	0	0	0	0	0	0	0	0	0	0
	10	92	0	0	0	0	0	580	0	0	0	0	0	0	0	0
	14	72	0	0	0	0	0	0	0	1105	0	0	0	0	0	0
	15	1537	0	0	0	0	0	0	0	0	0	0	0	0	0	3
	17	40	0	0	0	0	0	0	0	0	342	0	0	0	0	0
	18	110	0	0	0	0	0	0	0	0	0	459	0	0	0	0
	19	22	0	0	0	0	0	0	0	0	0	0	255	0	0	0
	20	62	0	0	0	0	0	0	0	0	0	0	0	716	0	0
	21	1106	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	355	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	457	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	25	76	0	0	0	0	0	0	0	0	0	0	0	0	1531	0
	26	1053	0	0	0	0	0	0	0	0	0	0	0	0	0	50
	28	387	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Para tentar melhorar a classificação da rede convolucional, uma estratégia utilizada foi limpar a base de dados, removendo exemplos durante a transição entre o estado normal e perturbado e vice-versa. Foram removidos pontos no período de até 1 hora depois de uma falha acontecer e de uma falha cessar. Isso faz com que as sequências inseridas no modelo não contenham dados do período de transição, dando mais clareza no aprendizado no modelo. Isso melhorou a acurácia para 75.98%. Uma analise mais aprofundada mostra que boa parte do aumento da acurácia se deve apenas à exclusão dos momentos de transição, como pode ser visto no aumento de zeros na primeira linha e coluna na Tabela 32 e que a melhora real da capacidade do modelo de aprendizado é pouca.

Tabela 29: Diagnóstico de falha para os dados de teste e validação da RN convolucional com base de dados limpa.

		Predito																
		0	1	4	7	8	10	12	13	14	17	18	19	20	23	25	26	27
Real	0	10557	1	0	0	300	45	0	11	0	26	102	0	92	18	11	28	36
	1	0	503	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0
	3	354	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	4	0	0	218	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	327	0	0	0	0	0	0	0	0	0	0
	6	544	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	7	0	0	0	231	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	50	0	0	0	0	447	0	0	0	0	0	0	0	0	0	0	2
	14	0	0	0	0	0	0	0	0	843	0	0	0	0	0	0	0	9
	15	1155	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	17	1	0	0	0	0	0	0	0	0	273	0	0	0	0	0	0	0
	18	23	0	0	0	0	0	0	0	0	0	374	0	0	0	0	0	0
	19	0	0	0	0	0	0	0	0	0	0	0	192	0	0	0	0	0
	20	5	0	0	0	0	0	0	0	0	0	0	0	611	0	0	0	0
	21	791	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	22	253	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	23	331	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	25	0	0	0	0	0	0	0	0	0	0	0	0	0	32	1149	0	0
	26	215	0	0	0	0	0	0	0	0	0	0	0	0	0	0	598	0
	28	276	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

4.2.4) Máquinas de vetores de suporte

Dos modelos MVS testados, o que obteve o melhor resultado foi a MVS com *kernel* RBF. O kernel RBF inclui um hiperparâmetro gamma, como pode ser visto na Tabela 1. A Figura 41 segue o resultado da otimização do modelo. Quanto mais amarelo é o quadrado maior a acurácia do modelo. O melhor modelo tinha gamma de 3.1627e-3 e C^{MVS} de 3.5481 para a detecção de falha. Gamma é o hiperparâmetro mais importante para o MVS. A princípio o C^{MVS} não influencia tanto desde que ele não seja muito baixo, o que aumenta muito a regularização ao ponto que a máquina não aprende a classificar o modelo.

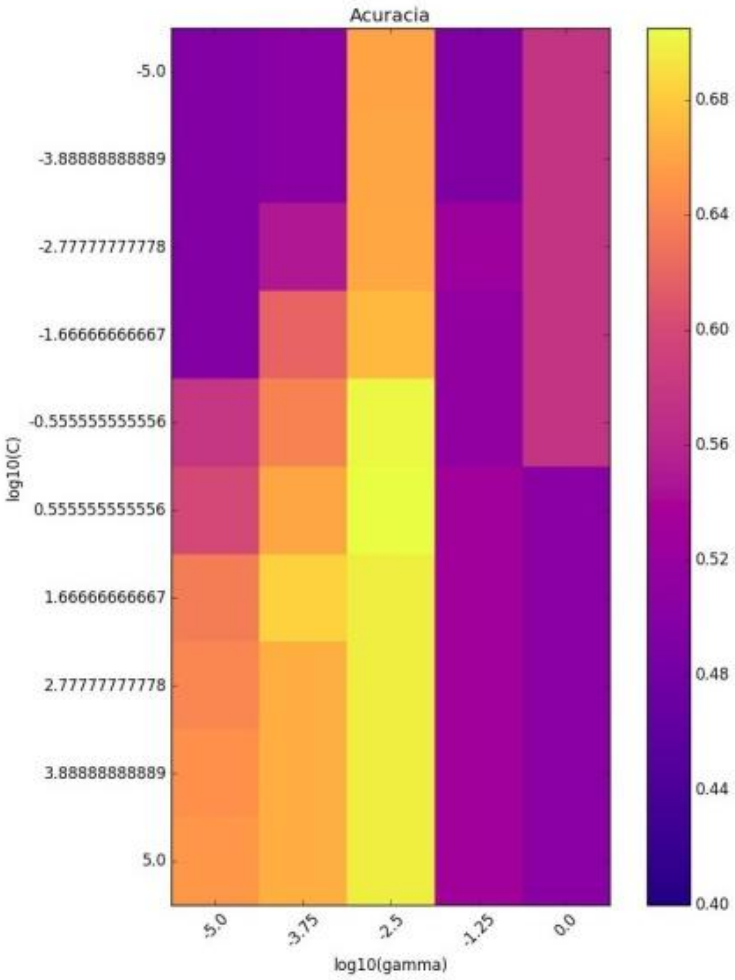

Figura 41: Mapa de calor da acurácia da MVS em relação a C^{MVS} e gamma.

A acurácia para detecção de falha foi de 72,9%, como pode ser visto na Tabela 30. A Figura 43 mostra a perda *Hinge* ao longo do tempo. Quando a perda *Hinge* é maior que 1 ocorre um erro de classificação; se a perda *Hinge* estiver entre 0 e 1 o ponto é corretamente classificado; mas dentro da margem da MVS. Se a perda for zero, o ponto está corretamente classificado e fora das margens da MVS. Analisando o gráfico da perda *Hinge*, Figura 43 é possível observar que a maior parte dos vetores se encontra dentro das margens. Esse resultado indica que é possível usar as margens como uma medida extra de confiabilidade da MVS. A área da curva ROC, Figura 42, é de 78,3%.

Tabela 30: Detecção de falha da MVS. . Dados de treino e validação. O sistema está normal?

	predito	
real	0	1
0	8676	4788
1	2992	12339

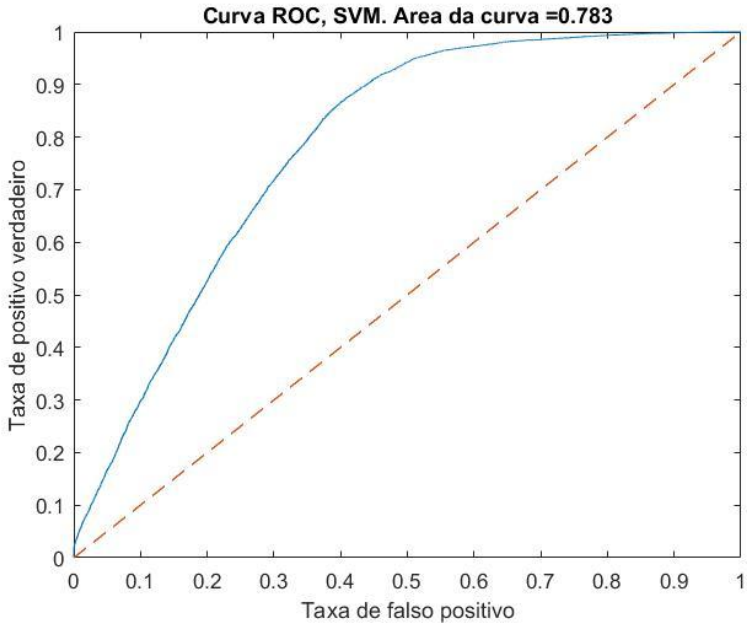

Figura 42: Curva ROC da MVS, Dados de treino e validação.

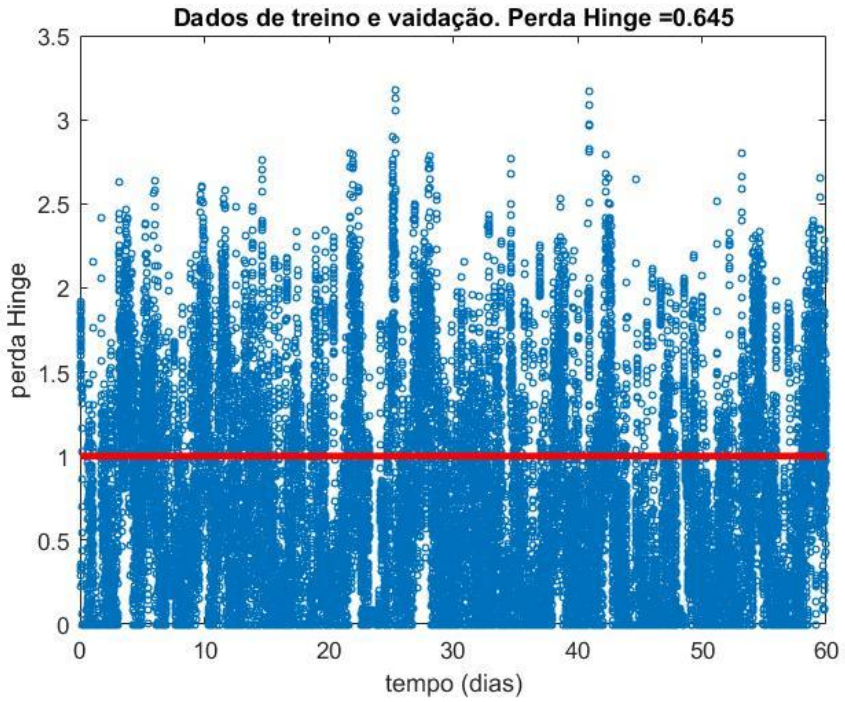

Figura 43: Perda *Hinge* da MVS, dados de treino e validação. Abaixo da linha vermelha são os pontos corretamente classificados, os pontos acima da linha vermelha são os pontos incorretamente classificados.

A Tabela 31 contém a classificação dos pontos em relação as margens. A maioria dos pontos incorretamente classificados estão dentro das margens, indicando que o posicionamento fora da margem pode ser usado como um indicativo a mais de confiabilidade da classificação.

Tabela 31: Resultado da acurácia da classificação em relação a posição do ponto dentro da margem.

	Classificação	
Margem	Correta	Incorreta
Dentro	52,49%	24,12%
Fora	20,49%	2,90%

A estratégia de classificação multiclasse utilizada foi *one-vs-rest*, pois a base de dados é muito grande e uma classificação *one-vs-one* demoraria muito, exigindo 406 classificadores, enquanto *one-vs-rest* exige 29. Como a classificação é o resultado de vários classificadores juntos, há uma variedade bem maior dos erros preditos do que com as redes, que usam apenas

um classificador. A taxa de acerto foi baixa, de 59,6%, como pode ser visto na Tabela 32. O tempo de treinamento para detecção de falha foi 2 horas, o tempo de treinamento para diagnóstico de falha foi dois dias.

Tabela 32: Matriz de confusão da MVS para diagnóstico de falha, estratégia *one-vs-rest*.

		Predito																							
		0	1	3	4	5	6	7	8	9	10	12	13	14	15	16	17	18	20	21	23	25	26	27	28
Real	0	10383	20	13	2	8	973	2	0	917	62	2	6	23	874	234	56	178	178	60	711	98	198	129	204
	1	16	677	0	0	0	0	0	18	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	213	0	82	0	0	3	0	0	42	0	0	0	0	43	9	0	0	0	0	91	0	0	0	4
	4	2	0	0	304	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	2	0
	5	16	0	0	0	425	0	0	0	5	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0
	6	398	0	0	0	0	28	0	0	63	0	0	0	0	126	27	0	0	0	0	81	0	16	0	19
	7	2	0	0	0	0	0	326	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10	130	0	0	0	0	0	0	0	0	520	0	0	0	0	8	0	0	0	0	14	0	0	0	0
	14	189	0	15	0	0	0	0	2	1	0	0	7	928	5	0	0	0	0	2	10	0	0	20	1
	15	577	0	0	0	0	80	0	0	83	0	0	0	0	206	14	0	0	0	9	556	0	1	0	14
	17	18	0	2	0	0	13	0	0	1	0	1	8	0	2	0	337	0	0	0	0	0	0	0	0
	18	84	0	0	0	0	1	0	0	2	0	0	0	0	1	0	0	481	0	0	0	0	0	0	0
	19	57	0	0	0	0	0	0	0	1	0	0	219	0	0	0	0	0	0	0	0	0	0	0	0
	20	33	0	0	0	0	0	0	0	0	0	0	0	0	7	10	0	0	729	0	0	0	0	0	5
	21	687	0	0	0	0	71	0	0	75	0	0	0	0	56	42	0	0	0	0	30	0	0	118	27
	22	271	0	5	0	0	34	0	0	14	0	0	0	0	18	0	0	0	0	0	9	0	4	0	0
	23	288	0	0	0	0	92	0	0	32	0	0	0	0	13	9	0	0	0	1	0	0	19	0	3
	25	135	0	0	0	0	11	0	0	0	0	0	33	0	5	0	0	0	0	0	0	1398	23	0	2
	26	637	0	0	0	0	62	0	0	90	0	0	0	0	36	15	0	0	0	0	16	0	241	0	6
	28	219	0	0	0	0	69	0	0	26	0	0	0	0	46	3	0	0	0	0	3	0	0	0	21

4.3) Análise por perturbação

Como mencionado anteriormente algumas perturbações são fáceis de diagnosticar, outras são mais difíceis. A Tabela 36 apresenta o diagnóstico de falhas para cada abordagem estudada por tipo de perturbação. As redes neuronais detectam a perturbação 5, mas elas a classificam como a falha 12. Esse resultado é compreensível, pois ambas as perturbações incidem sobre a entrada da temperatura da água de resfriamento, sendo a 5 um degrau e a 12, variação randômica. O FDA nem sempre diagnostica corretamente as perturbações 17 e 18, pois a malha de controle evita que o sistema fique fora da região de operação normal por muito tempo. No geral, pode se notar que as perturbações detectadas são as mesmas, tirando a perturbação 14, detectada apenas pelos métodos de *Machine Learning*. A perturbação 19 é diagnosticada apenas pela rede convolucional. Ela é detectada parcialmente pela RN recorrente e MVS, porém é classificada como perturbação 12 e 13 respectivamente.

Tabela 33: Diagnóstico de falha por tipo de perturbação.

	Diagnosticada por			
Perturbação	FDA dinâmico	RN recorrente	RN convolucional	MVS
1	Sim	Sim	Sim	Sim
3	Não	Não	Não	Não
4	Sim	Sim	Sim	Sim
5	Sim	Não	Não	Sim
6	Não	Não	Não	Não
7	Sim	Sim	Sim	Sim
10	Parcialmente	Sim	Sim	Sim
14	Não	Sim	Sim	Sim
15	Não	Não	Não	Não
17	Parcialmente	Sim	Sim	Sim
18	Parcialmente	Sim	Sim	Sim
19	Não	Não	Sim	Não
20	Sim	Sim	Sim	Sim
21	Não	Não	Não	Não
22	Não	Não	Não	Não
23	Não	Não	Não	Não
25	Sim	Sim	Sim	Sim
26	Não	Parcialmente	Não	Parcialmente
28	Não	Não	Não	Não

4.4) Sumário da classificação de falha

A Tabela 34 resume os resultados encontrados. Os métodos de *Machine Learning* foram superiores aos métodos lineares, principalmente na questão de sensibilidade vs sensitividade, como evidenciado pela área da curva ROC. No diagnóstico de falha as máquinas de vetores de suporte se mostraram inferiores aos métodos lineares, pois a classificação multiclasse não é intrínseca ao método.

Tabela 34: Sumário do resultado do problema de classificação de falha.

Modelo	**Acurácia de detecção de falha**	**Área da curva ROC**	**Acurácia de diagnóstico de falha**	**Função de perda**	**Tempo médio de treinamento**
Métodos lineares	68,06%	0,597	64,0%	_____	PCA 56 s FDA 127 s
Rede recorrente	76,65%	0,805	72,23%	0,264	2560 s
Rede convolucional	74,15%	0,764	70,71%	0,276	1184 s
MVS	72,9%	0,789	59,39%	0,645	detecção 2 h diagnóstico 2 dias

5) Conclusão

Esta dissertação apresenta inovação e pioneirismo nos seguintes aspectos:

- técnicas clássicas de ML – baseadas em redes neuronais tradicionais (MLP) e em MVS – foram comparadas com novas técnicas de ML – baseadas em redes neuronais recorrentes (ESN, GRU, LSTM) e profundas convolucionais, tendo ainda como referência técnicas estatísticas clássicas (modelos ARX, SS, PCA, FDD);

- os dados empregados se aproximavam a condições industriais (dados ruidosos, incompletos etc.);

- os dados foram empregados em grandes volumes, 26,8 Mb inferência estatística e 227 Mb para DDF, reproduzindo um contexto *Big Data*;

- problemas de inferência – no reator complexo de van de Vusse – e DDF – na planta completa Tennessee Eastman foram estudados.

As redes conseguiram fazer bem a inferência estatística do reator de van de Vusse. Elas são capazes de aprender as não linearidade do reator e suas propriedades, principalmente a inversão de ganho do reator de van de Vusse, o maior desafio ao controle do reator. As redes também se mostraram estáveis, podendo ser usadas para predição de longos períodos de tempo e não apenas em um instante seguinte.

Esse resultado indica que os modelos de *Machine Learning* podem ser usados com boa robustez em controle preditivo, sem preocupação com a instabilidade quando se calcula a janela de predição, sendo esse um projeto a ser pensado no futuro.

Quanto a detecção e diagnósticos de falha os resultados foram menos animadores. Os modelos de *Machine Learning* se mostraram melhores que os modelos lineares, mesmo assim sendo incapazes de prever uma grande quantidade de falhas. Por outro lado os modelos de *Machine Learning* se mostraram bem robustos ao indicar a falta de normalidade em limites de detecção mais robustos, como indicado pela curva ROC, enquanto os modelos lineares mantêm níveis similares de especificidades ao longo de todos os limites de detecção. Como pode ser visto na curva ROC da reder recorrente, é possível ter 90% dos positivos verdadeiros com apenas 40% dos positivos falsos.

Existe um espaço para aplicação industrial de *Machine Learning*, dado os melhores resultados em relação aos métodos tradicionais empregados. De maneira geral, as técnicas baseadas em *Machine Learning* se mostram aptas para emprego na engenharia de processos químicos, contudo o esforço analítico e computacional para resultados superiores ainda pode

ser considerado significativo. Há a necessidade de se fazer esses métodos mais simples para uma maior inserção deles na indústria.

No futuro recomendamos usar bases de dados maiores, com até 1 Gb, para melhor simular as bases de dados disponíveis na indústria, e buscar novas aplicações de *Machine Learning* ainda não estudadas, como reconciliação de dados, um problema tipico de regressão.

Outra questão interessante a ser explorada é o uso de aprendizado reforçado para desenvolver um esquema de controle robusto. Aprendizado reforçado utiliza dados obtidos *online*, sendo assim um sistema auto-atualizável.

6) Referências Bibliográficas

ALBERTOS, P; SALA, A; 2004 Multivariable control systems: An engineering approach. Springer 2004.

ANURADHA, D.B.; REDDY, G.P; MURTHY, J.S.N.; Direct Inverse Neural Network Control of A Continuous Stirred Tank Reactor (CSTR) IMECS 2009.

AVRIEL, M. Nonlinear Programming: Analysis and Methods. Courier Corporation. ISBN 978-0-486-43227-4. 2003.

BAGAJEWICZ, M. J. ; Process Plant Instrumentation: Design and Upgrade, Technomic. 2001.

BATHELT, A. N.; RICKER,L; JELALI, M; Revision of the Tennessee Eastman Process Model. 9th International Symposium on Advanced Control of Chemical Processes 2014.

BAUGHMAN, D. R. & LIU, Neural Networks in Bioprocessing and Chemical Engineering,. Academic Press 1995.

BRITZ, D; Understanding Convolutional Neural Networks for NLP, http://www.wildml.com/ 2015 /11/understanding-convolutional-neural-networks-for-nlp/, 2015, acessado 01/03/2017.

CHIANG, L. H., RUSSELL, E.L., BRAATZ, R. D. ; Fault Detection and Diagnosis in Industrial Systems,Springer-Verlag London, ISBN 978-1-4471-0347-9 2001.

CHOLLET, F. *et al.*; Keras, GitHub, https://github.com/fchollet/keras, 2015. Acessado janeiro de 2017.

CHUNG, J; GULCEHRE, C, CHO, K, BENGIO, Y; Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv:1412.3555 [cs.NE], 2014.

DE SOUZA JUNIOR, M. B.; Redes neuronais multicamadas aplicadas a modelagem e controle de processos quimicos, Tese de doutorado PEQ/COPPE/UFRJ, 1993).

DOWNS, J. J.; VOGEL, E. F. ; A plant-wide industrial process problem, Computers &Chemical engineering. Vol. 17, No. 3, pp. 245-255, 1993.

FARINA, L. A.; RPN-Toolbox: uma ferramenta para o desenvolvimento de estruturas de controle. Dissertação de mestrado: programa de pós-graduação em engenharia química UFRGS. 2000

FLETCHER, R. ; REEVES, C. M. "Function minimization by conjugate gradients", Comput. J. 7, 149–154. 1964.

FLETCHER, R., Practical methods of optimization (2nd ed.), New York: John Wiley & Sons, ISBN 978-0-471-91547-8 1987.

GAO, X; YANG, F; SHANG, C; HUANG, D.; A review of control loop monitoring and diagnosis: Prospects of controller maintenance in big data era. Chinese Journal of Chemical Engineering , 2016.

GAO, X.; SHANG, C.; HUANG, D.; JIANG, Y.; CHEN, T.; Refinery scheduling with varying crude: a deep belief network classification and multi-model approach AIChE J. 60 (7) 2525–2532. 2014

GARCIA, C. E.; PRETT, D. M.; MORARI, M. Model Predictive Control: Theory and Practice -A survey. Automatica, v. 25, p. 335-348, 1989.

GONÇALVES, G. A. A.; ALVARISTO, E. L.; SILVA, E. L.; SOUZA JR, M. B.; SECCHI, A. R.; A comparison of performance and implementation characteristics of nmpc formulations, XX Congresso Brasileiro de Engenharia Química, Blucher Chemical Engineering Proceedings, Volume 1, 2015, Pages 11746-11753, ISSN 2359-1757, 2015

HASTIE, T; TIBSHIRANI, R.; FRIEDMAN, J., The elements of statistical learning, 2^0 Edition, 2008.

IVANCIUC,O; Applications of Support Vector Machines in Chemistry Reviews in Computational Chemistry, Volume 23, 2007.

JAEGER, H., Matlab toolbox for reservoir computing.http://organic.elis.ugent.be/software. 2007. Acessado em agosto de 2016.

KADLECA, P; GABRYS B.A., STRANDTB,S; Data-driven Soft Sensors in the process industry, Computers and Chemical Engineering 33 (2009) 795–814, 2009.

KARPATHY, A;. The Unreasonable Effectiveness of Recurrent Neural Networks, May 21, 2015.

KRISHNAMURTHY, A.; CHAN, W.K.V.; Modeling Uncertainty due to Instrument Drift in Clinical Laboratories; Proceedings of the 2013 Industrial and Systems Engineering Research Conference, 2009.

LARSSON, T.. Studies on plantwide control. Tese submetida para o título de doutor em engenharia Department of Chemical Engineering. Norwegian University of Science and Technology, 2000

LARSSON, T.; HESTETUN, K.; HOVLAND, E.; SKOGESTAD,S. Self-Optimizing control of a large-scale plant: the Tennessee Eastman process", Ind. Eng. Chem. Res., Vol. 40, pp. 4889-4901, 2001.

LECUN Y,; BENGIO, Y; HINTON, G; Deep Learning, doi:10.1038/nature14539, Nature, 2015.

LECUN, Y.; BOTTOU, L.; ORR, G. B.; MÜLLER, K.-R;. Efficient BackProp. In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686, 2012.

LJUNG, L., System Identification: Theory For the User, Second Edition, Appendix 4A, pp 132-134, Upper Saddle River, N.J: Prentice Hall, 1999.

LEE, J. M.; LEE. J, H,; Approximate dynamic programming-based approaches for input–output data-driven control of nonlinear processes. Automatica 41 (2005) 1281 – 1288, 2005.

LEWIS, F, L; VRABIE, D; Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control IEEE CIRCUITS AND SYSTEMS MAGAZINE. 2009.

LORENA, A.; CARVALHO, A. R. Uma Introdução às Support Vector Machines. Revista de Informática Teórica e Aplicada, RITA Volume XIV Número 2 2007.

LUKOŠEVIČIUS M. A Practical Guide to Applying Echo State Networks. In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686, 2012.

MANNILA, H., Data mining: machine learning, statistics, and databases.. Int'l Conf. Scientific and Statistical Database Management. IEEE Computer Society. 1996.

The MathWorks, Inc. MATLAB 7.0 e Simulink 6.0, Natick, Massachusetts, United States.

MONTAVON, G; GENEVIÉVE B., O; MÜLLER, K.R. *et al.* Neural Networks: Tricks of the Trade.. 2^0 edition, 2012. DOI 10.1007/978-3-642-35289-8, 2012.

NARASIMHAN,S; JORDACHE, C.;. Data Reconciliation and Gross Error Detection, Gulf Publishing Company, 2000.

NASHALJI, M. N.; SHOOREHDELI, M. A.; TESHNEHLAB, M.; Fault Detection of the Tennessee Eastman Process Using Improved PCA and Neural Classifier In X.Z. Gao et al. (Eds.): Soft Computing in Industrial Applications, Springer-Verlag Berlin Heidelberg 2010.

NEUNEIER, R.; ZIMMERMANN, H. G.; How to Train Neural Networks In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686, 2012.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V., VANDERPLAS, J.; PASSOS, A.; COURNAPEAU,D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, É.; Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research 12 2825-2830, 2011.

PRECHELT, L.; Early Stopping — But When In: G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686, 2012.

QIN, J. Process Data Analytics in the Era of Big Data S. September 2014 Vol. 60, No. 9. Aiche Journal, 2014.

RAGAB, A; EL-KOUJOK, M.; AMAZOUZ, M.; YACOUT, S.; Fault detection and diagnosis in the Tennessee Eastman Process using interpretable knowledge discovery, Annual Reliability and Maintainability Symposium (RAMS), doi: 10.1109/RAM.2017.7889650, 2017.

RATO, T, J.; REIS, S. M.; Fault detection in the Tennessee Eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (DPCA-DR) Chemometrics and Intelligent Laboratory Systems, June 2013 vol 125, 2013.

RUSSELL, S.; NORVIG, P.,. Artificial Intelligence: A Modern Approach (2nd ed.). Prentice Hall. ISBN 978-0137903955. 2003.

SANT'ANNA, H. R; BARRETO, JR.; A. G.; TAVARES, F. W.; DE SOUZA JR., M. B.; Machine learning model and optimization of a PSA unit for methane-nitrogen separation Computers & Chemical Engineering, Volume 104, 2 September 2017, Pages 377-391

SCALES, L.E., *Introduction to Non-Linear Optimization*, New York, Springer-Verlag, 1985.

SHANG, C.; YANG, F.; HUANG, D.; LYU, W.; Data-driven soft sensor development based on deep learning technique, J. Process Control 24 (3) 223–233,2014

SHU, Y.; MING, L.; CHENG, F.; ZHANG, Z; ZHAO, J., Abnormal situation management: Challenges and opportunities in the big data era. Computers and Chemical Engineering 91 (2016) 104–113, 2016.

SIMON, P., Too Big to Ignore: The Business Case for Big DataWiley. p. 89. ISBN 978-1-118-63817-0 , 2013.

TAFNER, M. A. Redes Neurais Artificiais: Aprendizado e Plasticidade, Revista "Cérebro & Mente" 2(5), Março/Maio 1998.

VAN DE VUSSE, J. G. Plug-flow Type Reactor Versus Tank Reactor. Chemical Engineering Science, v. 19, p. 964-997, 1964.

VENKATASUBRAMANIAN, V.; DROWNING IN DATA: Informatics and Modeling Challenges in a Data-Rich Networked World January 2009 Vol. 55, No. 1, 2009.

VENKATASUBRAMANIAN, V; RENGASWAMY, R; KAVURI, S. N.; YINM,K; A review of process fault detection and diagnosis Part III: Process history based methods Computers and Chemical Engineering 27 (2003) 327/346, 2003.

WIDODO, A.; YANG B. S., Support vector machine in machine condition monitoring and fault diagnosis Mechanical Systems and Signal Processing Volume 21, Issue 6, August 2007, Pages 2560–2574 , 2007.

YAN, W; SHAO,H; WANG, H; Soft sensing modeling based on support vector machineand Bayesian model selection, Computers and Chemical engineering 28 (2004) 1489–1498, 2004.

YIN, S.; LI, X.; GAO, H.; KAYNAK, O. Data-Based Techniques Focused on Modern Industry: An Overview, IEEE transactions on industrial electronics, vol. 62, no. 1, January, 2015.

ZIMMERMANM, H.-G.; TIETZ, C.; GROTHMANN, R.; Forecasting with Recurrent Neural Networks: 12 Tricks, G. Montavon, G. B. Orr, and K.-R. Müller (eds.) Neural Networks: Tricks of the Trade, 2nd ed. Springer LNCS 7700, 659-686, 2012.

Apêndice I - Código do reator de Van der Vusse

```
function [sys,x0,str,ts] = VDV_SL(t,x,u,flag)
%This function implements A Van der Vusse model for Matlab
%parameters are constant here
A=3.14; %m^2
Vj=0.12; % m^3
k_01=2.145e10*60; %1/h
k_02=2.145e10*60; %1/h
k_03=1.507e5*60000; %1/h
E1=81091;   %kJ
E2=81091; %kJ
E3=71133;  %kJ
R=8.31; %kJ/kmol*K
H1= 4200; %kJ/kgmol
H2=-11000;%kJ/kgmol
H3=-41850; %kJ/kgmol
HD=2812.1; %Reaction heat capacity
HH2O=4185; %Water heat capacity
HA = 75.8*360000; %heat transfer coefficient kJ/h*K

switch flag,

  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3,
HD,HH2O, HA);

  case 1,
    sys=mdlDerivatives(t,x,u, A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3, HD,HH2O,
HA);

  case 3,
    sys=mdlOutputs(t,x,u,A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3, HD,HH2O, HA);
```

```
  case { 2, 4, 9 },
    sys = [];

  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));

end
% end csfunc

function [sys,x0,str,ts]=mdlInitializeSizes(A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3,
HD,HH2O, HA)

sizes = simsizes;
sizes.NumContStates  = 7;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 8;
sizes.NumInputs      = 6;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;

sys = simsizes(sizes);
x0  = [2.360; 410;12.56; 301;1.300;1.500;0.370];
str = [];
ts  = [0 0];

% end mdlInitializeSizes

function sys=mdlDerivatives(t,x,u,A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3,
HD,HH2O, HA)
Fout= sqrt(x(3)/A)*u(6); %output flow
dvdt= u(4)-Fout; %Volume difference

r1=k_01*exp(-E1/(R*x(2)))*x(1);
```

```
r2=k_02*exp(-E2/(R*x(2)))*x(5);
r3=k_03*exp(-E3/(R*x(2)))*(x(1).^2);

sys = [(u(4)*u(1)-Fout*x(1))/x(3)-r1-r3-(x(1)/x(3))*dvdt; ...
       (u(4)*u(2)-x(2)*Fout)/x(3)-(H1/HD)*r1-(H2/HD)*r2-(H3/HD)*r3-
(HA/(HD*x(3)))*(x(2)-x(4))-(x(2)/x(3))*dvdt; ...
      dvdt;...
      (u(5)/Vj)*(u(3)-x(4))+(HA/(HH2O*Vj))*(x(2)-x(4)); ...
      (-Fout*x(5))/x(3)+r1-r2-(x(5)/x(3))*dvdt; ...
      (-Fout*x(6))/x(3)+r2-(x(6)/x(3))*dvdt;  ...
      (-Fout*x(7))/x(3)+r3/2-(x(7)/x(3))*dvdt];


function sys=mdlOutputs(t,x,u,A, Vj, k_01, k_02,k_03, E1,E2,E3, R, H1,H2,H3, HD,HH2O,
HA)


 sys = [x(1);              ... % Concentration of substance A in the reactor.
      x(2) ;              ... % Reactor temperature.
      x(3) ;               ... % volume of reactor
     x(4);                    %temperature of the water
     x(5);                    ...   %Cb
     x(6);                   ...    %Cc
     x(7);                   ... %Cd
    sqrt(x(3)/A)*u(6)];    ... %outlet flow
```

Apêndice II - Perturbações adicionadas ao reator de van de Vusse.

Hora	Ca na entrada	Temperatura na entrada	Abertura da válvula	Set point da Temperatura do reator	Set point do Volume	Hora	Ca na entrada	Temperatura na entrada	Abertura da válvula	Set point da Temperatura do reator	Set point do Volume
0	0	0	0	0	0	200	0	0	0	0	0
2	0	0	497.77	419.58	0	202	0	0	546.29	0	0
4	0	0	0	0	0	204	0	0	0	0	0
6	5.96	0	0	0	10.49	206	0	0	0	413.13	0
8	0	0	0	0	0	208	0	0	0	0	0
10	6.21	0	0	399.54	0	210	0	0	0	0	0
12	0	0	0	0	0	212	0	0	0	0	0
14	3.33	0	0	0	0	214	3.6	0	0	0	0
16	0	0	0	0	0	216	0	0	0	0	0
18	0	0	0	0	0	218	0	0	0	0	0
20	0	0	0	0	0	220	0	0	0	0	0
22	0	0	0	0	10.67	222	0	593.83	0	454.42	0
24	0	0	0	0	0	224	0	0	0	0	0
26	6.9	341.11	0	342.29	0	226	0	557.06	463.97	444.42	0
28	0	0	0	0	0	228	0	0	0	0	0
30	0	0	0	0	0	230	0	0	0	0	0
32	0	0	0	0	0	232	0	0	0	0	0
34	0	0	0	0	0	234	0	590.79	456.38	452.22	10.77
36	0	0	0	0	0	236	0	0	0	0	0
38	0	591.66	0	448.94	0	238	0	0	497.79	396	0
40	0	0	0	0	0	240	0	0	0	0	0
42	0	423.36	435.92	0	0	242	6.91	0	533.21	0	10.09
44	0	0	0	0	0	244	0	0	0	0	0
46	0	353.54	0	355.24	0	246	0	0	0	0	10.29
48	0	0	0	0	0	248	0	0	0	0	0
50	6.42	0	332.66	0	10.83	250	0	0	275.45	394.11	0
52	0	0	0	0	0	252	0	0	0	0	0
54	0	0	0	0	9.34	254	0	0	0	0	0
56	0	0	0	0	0	256	0	0	0	0	0
58	0	0	0	0	9.38	258	0	0	555.17	0	8.93
60	0	0	0	0	0	260	0	0	0	0	0
62	0	0	0	0	0	262	0	0	231.69	0	9.39
64	0	0	0	0	0	264	0	0	0	0	0
66	0	515.25	446.07	418.96	9.23	266	0	274.4	0	274.43	10
68	0	0	0	0	0	268	0	0	0	0	0
70	3.68	0	0	398.68	0	270	3.08	0	0	0	0
72	0	0	0	0	0	272	0	0	0	0	0
74	0	510.39	0	411.84	0	274	0	0	0	393.73	0
76	0	0	0	0	0	276	0	0	0	0	0
78	0	0	0	0	9.35	278	3.81	0	526.24	0	10.33
80	0	0	0	0	0	280	0	0	0	0	0

82	0	0	0	0	9.46	282	0	180.76	0	181.02	0
84	0	0	0	0	0	284	0	0	0	0	0
86	0	0	0	396.35	0	286	0	0	0	0	9.49
88	0	0	0	0	0	288	0	0	0	0	0
90	0	0	0	0	0	290	0	0	0	0	10.24
92	0	0	0	0	0	292	0	0	0	0	0
94	3.45	0	229.68	0	0	294	0	189.77	0	189.97	0
96	0	0	0	0	0	296	0	0	0	0	0
98	6.98	0	0	0	0	298	6.65	443.09	347.25	0	0
100	0	0	0	0	0	300	0	0	0	0	0
102	0	565.19	430.4	444.89	0	302	3.65	0	0	0	0
104	0	0	0	0	0	304	0	0	0	0	0
106	6.97	0	0	0	10.29	306	0	283.37	0	283.39	0
108	0	0	0	0	0	308	0	0	0	0	0
110	0	0	0	0	9.16	310	3.63	0	0	391.18	0
112	0	0	0	0	0	312	0	0	0	0	0
114	6.72	515.83	0	412.78	0	314	0	0	0	413.04	0
116	0	0	0	0	0	316	0	0	0	0	0
118	6.97	0	270.41	0	0	318	0	266.69	0	266.7	0
120	0	0	0	0	0	320	0	0	0	0	0
122	0	0	0	0	0	322	0	363.06	0	366.41	0
124	0	0	0	0	0	324	0	0	0	0	0
126	3.24	0	0	0	0	326	0	346.14	544.14	346.79	10.33
128	0	0	0	0	0	328	0	0	0	0	0
130	0	0	0	0	10.62	330	6.08	0	0	0	0
132	0	0	0	0	0	332	0	0	0	0	0
134	0	0	0	0	9.8	334	0	0	493.12	0	0
136	0	0	0	0	0	336	0	0	0	0	0
138	3.14	0	0	0	9.35	338	0	0	218.27	0	9.82
140	0	0	0	0	0	340	0	0	0	0	0
142	0	461.88	0	0	0	342	0	0	0	0	10.83
144	0	0	0	0	0	344	0	0	0	0	0
146	6.03	0	0	0	0	346	0	0	0	0	0
148	0	0	0	0	0	348	0	0	0	0	0
150	0	0	0	0	9.29	350	0	0	0	0	0
152	0	0	0	0	0	352	0	0	0	0	0
154	0	0	485.99	417.99	0	354	0	0	0	392.64	0
156	0	0	0	0	0	356	0	0	0	0	0
158	3.78	488.79	0	0	0	358	6.62	0	497.84	0	0
160	0	0	0	0	0	360	0	0	0	0	0
162	6.92	0	322.82	393.03	0	362	0	0	0	0	10.44
164	0	0	0	0	0	364	0	0	0	0	0
166	0	0	0	415.3	10.12	366	6.36	0	0	0	10.2
168	0	0	0	0	0	368	0	0	0	0	0
170	0	348.47	0	349.68	0	370	6.98	0	286.23	397.46	0
172	0	0	0	0	0	372	0	0	0	0	0

174	0	0	493.72	0	0	374	0	538.03	365.1	417.82	0
176	0	0	0	0	0	376	0	0	0	0	0
178	0	0	0	419.24	0	378	0	364.1	0	367.54	0
180	0	0	0	0	0	380	0	0	0	0	0
182	0	0	0	419.02	0	382	3.84	0	0	0	10.69
184	0	0	0	0	0	384	0	0	0	0	0
186	3.4	276.32	0	276.36	10.94	386	3.26	0	0	0	0
188	0	0	0	0	0	388	0	0	0	0	0
190	6.32	0	0	0	0	390	6.34	0	0	393.21	0
192	0	0	0	0	0	392	0	0	0	0	0
194	0	0	587.06	0	9.96	394	0	0	0	0	0
196	0	0	0	0	0	396	0	0	0	0	0
198	0	0	0	397.55	9.89	398	0	0	0	0	9.48

Apêndice III - Perturbações adicionados ao processo Tennessee Eastman.

Dia	Perturbação	Dia	Perturbação	Dia	Perturbação	Dia	Perturbação	Dia	Perturbação
0.00	0	42.69	0	82.93	0	122.43	0	160.34	0
1.32	23	43.63	13	83.78	15	123.04	28	161.62	6
2.72	0	44.51	0	85.11	0	124.50	0	162.51	0
3.35	19	45.78	9	86.20	3	125.00	2	163.25	1
4.76	0	47.07	0	87.25	0	126.28	0	164.15	0
5.89	11	47.76	26	88.66	8	127.59	25	164.75	21
6.49	0	48.75	0	89.45	0	128.96	0	165.38	0
7.27	23	49.70	13	90.71	23	129.55	26	166.82	15
8.32	0	50.84	0	91.96	0	130.45	0	168.28	0
9.77	15	52.05	6	92.84	1	131.21	23	169.35	14
11.24	0	53.31	0	93.91	0	132.51	0	169.91	0
11.90	10	54.08	26	94.49	27	133.44	3	170.65	26
13.37	0	55.26	0	95.04	0	134.85	0	171.50	0
14.82	27	56.42	28	96.07	21	135.53	8	172.82	18
15.81	0	57.08	0	97.35	0	136.29	0	173.34	0
17.11	25	57.70	13	98.78	14	136.94	10	173.88	18
17.75	0	58.70	0	99.41	0	137.58	0	174.55	0
18.67	16	60.16	4	100.48	17	138.95	20	175.70	25
20.09	0	61.00	0	101.45	0	140.03	0	176.93	0
21.38	18	62.08	8	101.96	7	141.08	4	178.08	23
22.84	0	62.81	0	102.80	0	141.72	0	179.03	0
24.00	17	64.06	12	103.46	13	143.07	21	180.08	17
24.53	0	64.81	0	104.76	0	144.19	0	180.87	0
25.88	6	65.82	17	105.57	27	145.05	3	182.12	6
27.32	0	67.02	0	106.60	0	146.06	0	182.81	0
28.49	9	68.41	8	107.26	16	146.96	19	183.99	7
29.75	0	69.87	0	108.36	0	147.54	0	184.68	0
31.00	14	70.91	17	109.13	15	148.28	14	185.55	25
31.89	0	71.55	0	110.28	0	148.90	0	186.67	0
33.04	7	72.20	20	111.47	7	149.58	22	187.95	1
33.71	0	72.96	0	112.72	0	150.32	0	188.53	0
34.92	24	74.30	7	113.67	14	151.24	21	189.96	14
35.45	0	75.06	0	114.25	0	151.79	0	191.24	0
36.23	6	76.37	4	114.98	18	153.19	26	192.22	5
36.78	0	77.11	0	116.39	0	154.64	0	193.16	0

37.37	7	78.54	9	117.05	20	155.63	25	194.11	28
38.70	0	79.39	0	118.37	0	156.62	0	194.91	0
39.89	5	80.09	9	119.41	12	157.46	10	195.92	20
40.71	0	80.84	0	120.91	0	158.86	0	196.93	0
42.16	7	81.96	12	121.49	11	159.73	20	198.25	15

Apêndice IV - Valores utilizados na estandardização da base de dados

Médias e desvio padrões usados na estandardização dos dados do reator de van de Vusse.

Entradas	**C_a na entrada**	**Temperatura da entrada**	**Temperatura de entrada da água de resfriamento**	**Vazão de entrada**	**Vazão da água de resfriamento**	**Abertura da válvula**	
Média	5.46	410.66	300.09	788.70	86.69	402.85	
Desvio padrão	0.92	44.62	3.53	74.76	116.10	41.75	
Saídas	**C_a na saída**	**Temperatura do reator**	**Volume do reator**	**C_b**	**C_c**	**vazão de saída**	**Vazão de B**
Média	2.66	403.56	12.12	1.08	1.06	789.08	848.49
Desvio padrão	0.86	31.24	1.38	0.30	0.67	74.13	248.21

Médias e desvios-padrão usados na estandardização das variáveis manipuláveis do processo TE.

Variável	**Média**	**Desvio padrão**
xmv(1)	62.9171	0.145562376
xmv(2)	53.21404	0.231457526
xmv(3)	26.80271	6.254630709
xmv(4)	60.98203	2.654523521
xmv(6)	25.9407	6.094842006
xmv(7)	37.33135	0.202349613
xmv(8)	46.45628	0.126662436
xmv(10)	35.96027	0.620734841
xmv(11)	12.56042	1.656362227

Tabela 22: Médias e desvios-padrão usados na estandardização das variáveis medidas do processo TE.

Variável	**média**	**Desvio padrão**		**Variável**	**média**	**Desvio padrão**
xmeas(1)	0.272468	0.063593088		xmeas(22)	91.82856	0.777734
xmeas(2)	3656.129	19.51819898		xmeas(23)	31.86783	0.520966
xmeas(3)	4444.883	24.72973602		xmeas(24)	15.03378	0.29185
xmeas(4)	9.229507	0.086384515		xmeas(25)	18.74023	0.43567
xmeas(5)	32.20033	0.294392194		xmeas(26)	6.036943	0.105895
xmeas(6)	47.39441	0.299383519		xmeas(27)	16.93998	0.349521
xmeas(7)	2799.724	4.770463124		xmeas(28)	4.065867	0.119299
xmeas(8)	64.99938	0.550010136		xmeas(29)	32.2347	0.70598
xmeas(9)	122.9	0.040146054		xmeas(30)	21.98439	0.410682
xmeas(10)	0.210725	0.049350108		xmeas(31)	13.1	0.563874
xmeas(11)	91.76388	0.300224393		xmeas(32)	0.916592	0.102973
xmeas(12)	49.99958	1.357682473		xmeas(33)	16.49554	0.41002
xmeas(13)	2703.829	5.464624551		xmeas(34)	5.417661	0.154867
xmeas(14)	25.3428	0.178391303		xmeas(35)	6.616432	0.080638
xmeas(15)	49.99859	3.153709196		xmeas(36)	3.229776	0.061289
xmeas(16)	3331.465	5.98766933		xmeas(37)	0.011033	0.009966
xmeas(17)	22.88984	0.123814733		xmeas(38)	0.597592	0.022662
xmeas(18)	66.35374	0.292882774		xmeas(39)	0.189977	0.011923
xmeas(19)	-0.00016	1.150613796		xmeas(40)	53.80156	0.516004
xmeas(20)	271.1085	1.847633045		xmeas(41)	43.92682	0.517469
xmeas(21)	102.4258	0.641115864				