

**Exercício de Fixação de Conceitos 1**  
Gustavo Ciotto Pinton - 117136  
EA072 - Inteligência Artificial em Aplicações Industriais

## 2 Seleção de variáveis empregando filtros e *wrappers*

1. A quantidade de dados e de variáveis de entrada podem representar verdadeiros obstáculos no treinamento e validação dos pesos sinápticos de redes neurais, principalmente ao que se refere aos recursos disponíveis de processamento. Sendo assim, algoritmos e técnicas que possam ser capazes de determinar o grau de importância das variáveis em relação à saída e distinguir os dados mais pertinentes tornam-se indispensáveis a essas operações. Destacam-se duas técnicas: a primeira, a chamada técnica **filter**, busca a classificar as variáveis de acordo com algum critério, seja ele a *correlação* ou a *informação mútua* entre as variáveis de entrada  $\mathbf{x}_j$  e as saídas  $\mathbf{y}_i$ . Tais técnicas independem do modelo de predição e são aplicadas durante a fase de pré-processamento. A segunda, chamada **wrapper**, utiliza uma máquina de aprendizado qualquer como uma caixa preta e avalia os subconjuntos de variáveis de acordo com suas respectivas qualidades de predição. Tais características podem impor algumas dificuldades a essa última técnica, à medida que a avaliação dos resultados de predição pode não ser tão trivial e o método de construção dos subconjuntos pode apresentar uma complexidade elevada. Destacam-se, portanto, os métodos de *forward selection* e *backward elimination*.

2. Seja  $\mathbb{S}$  o conjunto de variáveis de entrada cujo efeito na saída do modelo  $\hat{\mathbf{y}}_k$  seja pertinente em relação à saída esperada  $\mathbf{y}_k$ . A abordagem de *forward selection* consiste a aumentar progressivamente  $\mathbb{S}$ , à medida que uma variável se mostre importante ao modelo. A importância de uma variável pode ser determinada através de alguns critérios, como por exemplo o cálculo de  $J(\mathbb{S} \cup \{x_i\}) = \sum_{k=1}^m (\hat{\mathbf{y}}_k - \mathbf{y}_k)^2$ , sendo  $x_i$  um variável candidata à inserção. Neste caso, compara-se  $J(\mathbb{S} \cup \{x_i\})$  e  $J(\mathbb{S})$  e, caso o efeito dessa variável seja positivo, isto é,  $J(\mathbb{S} \cup \{x_i\})$  menor, a acrescentamos em  $\mathbb{S}$ . Para *forward selection*,  $\mathbb{S}$  começa vazio.

A abordagem de *backward elimination*, por sua vez, elimina de  $\mathbb{S}$  gradativamente as variáveis menos pertinentes ao modelo.  $\mathbb{S}$  é, portanto, inicializado com todas as variáveis. Analogamente ao caso anterior, pode-se calcular  $J(\mathbb{S} \setminus \{x_i\})$  e compará-lo com  $J(\mathbb{S})$ . Caso  $J(\mathbb{S} \setminus \{x_i\})$  seja inferior, elimina-se de  $\mathbb{S}$  a variável  $\{x_i\}$ .

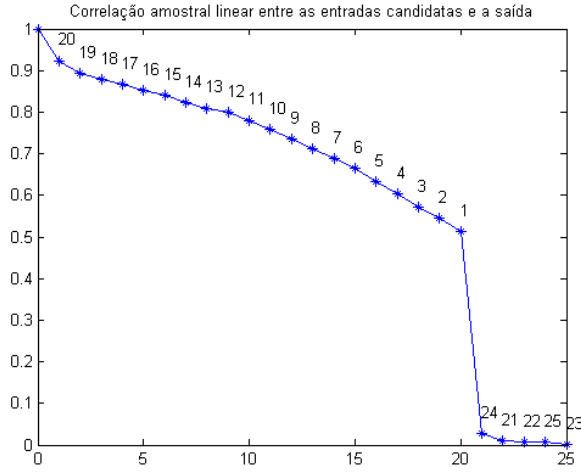
As duas abordagens acima não garantem a melhor combinação de entradas pelo fato da possível existência de *mínimos locais* da função  $J(\mathbb{T})$ , a função que associa o erro com as variáveis presentes no conjunto  $\mathbb{T}$ . Dependendo das condições e ordem de verificação das variáveis  $x_i$ , o método pode tender a diferentes mínimos, que podem ser eventualmente os melhores ou não.

3. (a) A tabela 1 a seguir descreve algumas características estatísticas da série temporal. Todos os valores foram calculados através do MATLAB. Além delas, a série é formada por 3180 entradas, sendo compostas pelos dados de todos os meses de 1749 até 2013.

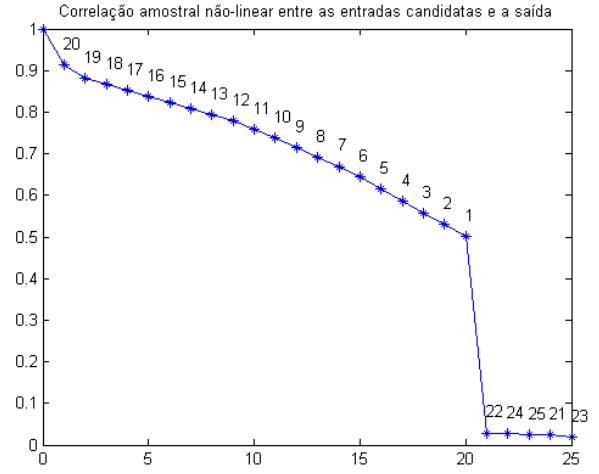
Tabela 1: Características da série temporal

Propriedades	Valor
Média	51.9949
Valor máximo	253.8
Valor mínimo	0
Desvio padrão	41.116

(b) Para o *filtro linear*, as execuções de *filtro\_lin(dados1.mat)* e *filtro\_nlin(dados1.mat)* resultam nas figuras 1a e 1b, respectivamente.



(a) Resultado do *filtro linear*.



(b) Resultado do *filtro não linear*.

Figura 1: Resultados das execuções das funções de filtro.

Conclui-se portanto que as variáveis de 1 a 20 apresentam as maiores correlações, tanto a linear  $R_{linear}$  quanto a não linear  $R_{nlinear}$ , em relação à saída, sendo a 20ª a mais correlata. A ordem de relevância em que elas aparecem também é a mesma, isto é, a sequência decrescente de 20 até 1 é observada em ambos os casos, e a maior diferença percentual entre  $R_{linear}$  e  $R_{nlinear}$  é de 3.12%, correspondente à 13ª variável.

Observa-se ainda que as variáveis 21 até 25 apresentam  $R_{linear}$  e  $R_{nlinear}$  são muito inferiores em relação aos valores das outras variáveis e que a sua ordem de relevância é diferente: para o filtro *linear* temos a sequência 24, 21, 22, 25 e 23, enquanto que para o *não linear* obtemos 22, 24, 25, 21 e 23.

(c) Para o método ***forward selection***<sup>1</sup>, obtem-se para as 5 execuções os seguintes dados:

Tabela 2: Resultados da execução **1**.

Entradas	20	18	17	3	12	19	15	1	16	11	23	5	7
Erro mínimo	1.1765												
No. de Variáveis	13												

Tabela 3: Resultados da execução **2**.

Entradas	20	18	17	3	12	19	15	1	10	5	22	7	21	25	16
Erro mínimo	1.1745														
No. de Variáveis	15														

Tabela 4: Resultados da execução **3**.

Entradas	20	18	17	3	12	15	19	1	16	23	5	10	13	24
Erro mínimo	1.1727													
No. de Variáveis	14													

Tabela 5: Resultados da execução **4**.

Entradas	20	18	17	3	12	15	19	1	16	10	5	23	24	13	21
Erro mínimo	1.1707														
No. de Variáveis	15														

<sup>1</sup>As imagens referentes a estas execuções encontram-se na figura 8 na seção **Anexos** no fim do documento

Tabela 6: Resultados da execução **5**.

Entradas	20	18	17	3	12	19	15	1	10	11	5	7
Erro mínimo	1.1768											
No. de Variáveis	12											

Considerando somente as primeiras cinco variáveis selecionadas em cada execução, percebe-se que, em realidade, elas são todas iguais: 20, 18, 17, 3 e 12, nessa sequência. Na sexta posição, encontra-se a variável 19 (3 ocasiões) ou a 15 (2 ocasiões) e na sétima, a variável 1. À partir dessa posição, as entradas selecionadas variam a cada iteração.

Para o método *backward elimination*<sup>2</sup>, obtem-se:

Tabela 7: Resultados da execução **1**.

Entradas	23	25	24	5	22	16	21	1	18	15	19	12	3	17	20
Erro mínimo	1.1711														
No. variáveis restantes	15														

Tabela 8: Resultados da execução **2**.

Entradas	8	25	11	7	5	16	1	15	18	19	12	3	17	20
Erro mínimo	1.1717													
No. variáveis restantes	14													

Tabela 9: Resultados da execução **3**.

Entradas	8	7	23	21	5	16	11	1	18	15	19	12	3	17	20
Erro mínimo	1.1744														
No. variáveis restantes	15														

Tabela 10: Resultados da execução **4**.

Entradas	6	21	7	8	10	24	16	25	1	18	15	19	12	3	17	20
Erro mínimo	1.1733															
No. variáveis restantes	16															

Tabela 11: Resultados da execução **5**.

Entradas	7	21	13	16	5	10	1	18	15	19	3	12	17	20
Erro mínimo	1.1748													
No. variáveis restantes	14													

Observa-se que o número de variáveis restantes para as execuções do *backward elimination* é ligeiramente superior em relação ao método anterior, apresentando uma média de 15 variáveis escolhidas contra aproximadamente 14 do *forward selection*. A média do erro quadrado médio na construção do modelo é inferior para *backward elimination*: 1.1731 contra 1.1742. Nota-se ainda que algumas variáveis foram escolhidas em todas as execuções de ambos, sendo elas, por exemplo, a 20, 17, 3, 12 e a 15.

- (d) Entradas que possuem as maiores correlações não são as primeiras a serem adicionadas ao modelo pela presença de *redundância* entre elas. Se uma variável é altamente correlata com uma outra, eu não precisaria, em princípio, conhecer as duas, já que a partir de um única, eu sou capaz de determinar a outra. Em outras palavras, variáveis *redundantes* adicionam pouca informação ao sistema. Por exemplo, a correlação entre as variáveis 20 e 19 do arquivo dados1.mat vale 0.9239 (valor obtido pelo comando `corr(X(:,20), X(:,19))` do MATLAB). Isso significa que se  $X_{20}$  é alto, então  $X_{19}$  também o é e, portanto, nenhuma outra informação é adicionada ao modelo. É por essa razão que nas execuções acima,  $X_{20}$  é escolhida inicialmente e  $X_{19}$ , só depois de algumas iterações.

<sup>2</sup>As imagens referentes a estas execuções encontram-se na figura 9 na seção **Anexos** no fim do documento

- (e) Variáveis de baixa correlação com a saída podem proporcionar um grande poder de separação, se consideradas juntamente com outras<sup>3</sup>. Desta maneira, as variáveis de menor correlação são separadas mais facilmente em relação às de maior correlação, permitindo assim a detecção de classes com maior precisão. Em outras palavras, entradas mais próximas das variáveis de baixa correlação podem ser classificadas mais facilmente.
- (f) As variáveis geradas aleatoriamente ( $X_{21} \cdots X_{25}$ ) apresentam correlações em relação à saída praticamente nulas (os resultados de `corr (X(:, 21), S) ... corr (X(:, 25), S)` estão mostrados na tabela 12). Dessa maneira, elas são escolhidas em ambos os modelos pelos mesmos motivos daqueles discutidos nos dois itens anteriores (*redundância e poder de separação*).

Tabela 12: Correlação das variáveis aleatórias execução 1 do *forward selection*.

Variável $X_i$	<code>corr (X(:, i), S)</code>
$X_{21}$	0.001347309212350
$X_{22}$	0.011839959350690
$X_{23}$	-0.037287334573179
$X_{24}$	-0.007005116183944
$X_{25}$	0.001963170472808

4. (a) O arquivo `wineq.mat` é composto por duas estruturas de dados. A primeira, matriz  $X_{1593 \times 11}$ , possui 11 colunas, correspondentes a cada uma das variáveis de entrada, e 1593 linhas, simbolizando 1593 dados disponíveis. De acordo com o *website* de onde tais dados foram retirados, essas variáveis correspondem a diversas propriedades que podem ser extraídas de um vinho, sendo elas *fixed acidity*, *volatile acidity*, *citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide*, *density*, *pH*, *sulphates* e *alcohol*. A tabela abaixo mostra algumas características dessas propriedades.

Tabela 13: Informações correspondentes às variáveis de `wineq.mat`.

Variável	Média	Max.	Min.	<i>Standard Deviation</i>
<i>fixed acidity</i>	0.52324	1	0.28931	0.10932
<i>volatile acidity</i>	0.33385	1	0.075949	0.11333
<i>citric acid</i>	0.27116	1	0	0.19495
<i>residual sugar</i>	0.16374	1	0.058065	0.090957
<i>chlorides</i>	0.14321	1	0.01964	0.077153
<i>free sulfur dioxide</i>	0.2205	1	0.013889	0.14537
<i>total sulfur dioxide</i>	0.16052	1	0.020761	0.11379
<i>density</i>	0.022051	1	0.0098643	0.096464
<i>pH</i>	0.82574	1	0.68329	0.038451
<i>sulphates</i>	0.32903	1	0.165	0.084846
<i>alcohol</i>	0.69949	1	0.56376	0.071404

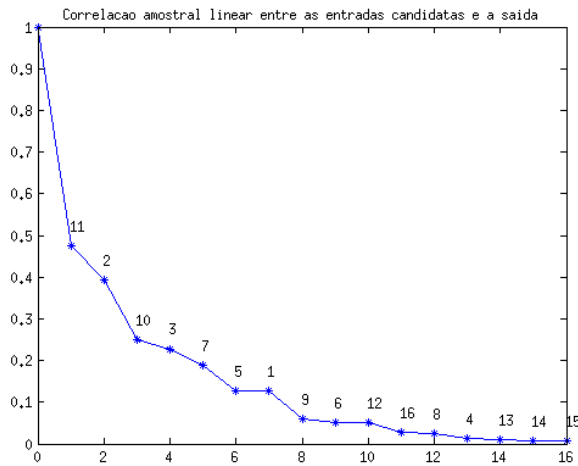
Para a saída encontra-se:

Tabela 14: Informações correspondentes à saída de `wineq.mat`.

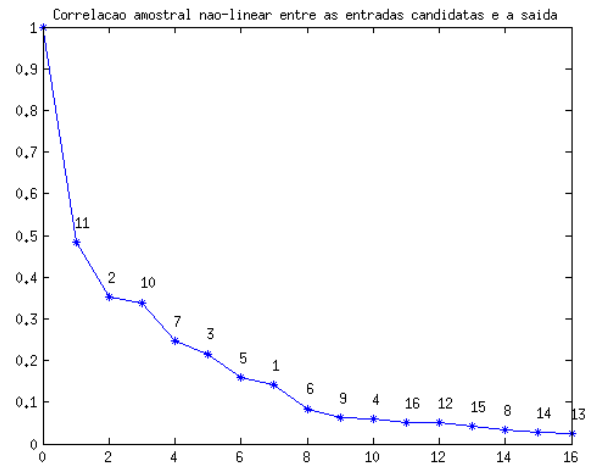
Variável	Média	Max.	Min.	<i>Standard Deviation</i>
Saída	0.70425	1	0.375	0.10101

<sup>3</sup>No artigo [ Guyon, I.; Elisseeff, A. “An introduction to variable and feature selection”, *Journal of Machine Learning Research*, vol. 3, pp. 1157 - 1182 2003 ], o autor dá um exemplo dessa afirmação na seção 3.3.

(b) As execuções de `filtro_lin('dados2.mat')` e `filtro_nlin('dados2.mat')` são mostradas na figuras 2a e 2b a seguir.



(a) Resultado do *filtro linear*.



(b) Resultado do *filtro não linear*.

Figura 2: Resultados das execuções das funções de filtro.

Neste caso, observa-se uma maior diferença entre as filtragens linear e não linear em relação ao estudo de caso anterior. Percebe-se inicialmente que a correlação da variável mais correlata ( $< 0.5$ ) aqui é muito inferior àquela mais correlata ( $> 0.9$ ) para os dados referentes aos *Sunspots*.

A ordem decrescente de correlação em que as variáveis aparecem também muda do caso linear para o não linear. A tabela seguinte evidencia esta afirmação.  $\Delta = \frac{(R_{nlin} - R_{lin})}{R_{nlin}}$ , onde  $R$  é o valor da correlação linear ou não linear, representa a variação percentual entre as respectivas correlações.

Tabela 15: Ordens de aparição das variáveis.

	Ordem															
$V_{lin}$	11	2	10	3	7	5	1	9	6	12	16	8	4	13	14	15
$V_{nlin}$	11	2	10	7	3	5	1	6	9	4	16	12	15	8	14	13
$\Delta$	0.01	-0.12	0.26	0.07	0.13	0.2	0.11	0.31	0.18	0.16	0.46	0back.5	0.68	0.73	0.75	0.72

Observa-se que a ordem muda sobretudo no fim da sequência, onde as variações percentuais são superiores.

(c) Para o método ***forward selection***<sup>4</sup>, obtem-se para as 5 execuções os seguintes dados:

Tabela 16: Resultados da execução **1**.

Entradas	11	2	10	7	5	9	6	16
Erro mínimo	1.0484							
No. de Variáveis	8							

Tabela 17: Resultados da execução **2**.

Entradas	11	2	10	7	5	9	6	13	14
Erro mínimo	1.0491								
No. de Variáveis	9								

Tabela 18: Resultados da execução **3**.

Entradas	11	2	10	7	5	9	6	13
Erro mínimo	1.0495							
No. de Variáveis	8							

Tabela 19: Resultados da execução **4**.

Entradas	11	2	10	7	5	9	6	14
Erro mínimo	1.0509							
No. de Variáveis	8							

<sup>4</sup>As imagens referentes a estas execuções encontram-se na figura 10 na seção **Anexos** no fim do documento

Tabela 20: Resultados da execução **5**.

Entradas	11	2	10	7	5	9	6	13
Erro mínimo	1.0509							
No. de Variáveis	8							

Para o método *backward elimination*<sup>5</sup>, obtem-se:

Tabela 21: Resultados da execução **1**.

Entradas	13	6	9	5	7	10	2	11
Erro mínimo	1.0496							
No. de Variáveis	8							

Tabela 22: Resultados da execução **2**.

Entradas	15	3	14	9	5	7	10	2	11
Erro mínimo	1.0558								
No. de Variáveis	9								

Tabela 23: Resultados da execução **3**.

Entradas	15	12	3	6	9	5	7	10	2	11
Erro mínimo	1.0532									
No. de Variáveis	10									

Tabela 24: Resultados da execução **4**.

Entradas	3	1	14	6	9	5	7	10	2	11
Erro mínimo	1.0548									
No. de Variáveis	10									

Tabela 25: Resultados da execução **5**.

Entradas	12	6	14	9	5	7	10	2	11
Erro mínimo	1.0497								
No. de Variáveis	9								

Observa-se que para a técnica de *forward selection* as primeiras sete entradas selecionadas foram as mesmas para as 5 execuções, sendo elas a 11, 2, 10, 7, 5, 9 e 6. As restantes escolhidas fazem parte das entradas aleatórias, calculadas para cada execução. Em média, foram escolhidas 8 variáveis e produziu-se um erro de 1.04976.

Em relação à técnica de *backward elimination*, as mesmas variáveis citadas no parágrafo anterior estiveram presentes, com exceção à 6, em que esteve ausente somente na execução 2. Foram selecionadas, em média, um número maior de variáveis, 9, e um erro superior, 1.05262. As variáveis restantes foram escolhidas dentre as geradas aleatoriamente.

Finalmente, as variáveis que foram escolhidas nos primeiros lugares na *forward selection* seriam eliminadas por último na *backward elimination*, confirmando assim um certo nível de semelhança nos resultados das duas técnicas.

## 3 Séries temporais e a tarefa de predição

### 3.0 Normalização dos dados

De acordo com enunciado disponibilizado, os dados devem ser normalizados de maneira a obter média nula e desvio padrão unitário. Entradas que excursionam em intervalos muito extensos prejudicam o desempenho das redes neurais MLP.

Sendo assim, faremos uso da seguinte equação para atingir as características necessárias:

$$X_i = \frac{V_i - \hat{\mu}}{\hat{\sigma}} \quad (1)$$

em que  $\hat{\mu}$  é a média de todas as entradas (calcula através de `mean(dengue_SP)`) e  $\hat{\sigma}$  é o estimador do desvio padrão (calculado por `std(dengue_SP)`). Obtem-se, assim, um novo vetor cuja média e

<sup>5</sup>As imagens referentes a estas execuções encontram-se na figura 11 na seção **Anexos** no fim do documento

variância valem, respectivamente,  $1.8288 \times 10^{-16}$  e 1. A figura à seguir mostra um histograma dos dados normalizados. Observa-se que a maioria dos dados encontram-se na vizinhança de 0, mas há uma quantidade razoável deles superior à unidade, correspondentes aos períodos de chuva, onde há, naturalmente, mais casos da doença.

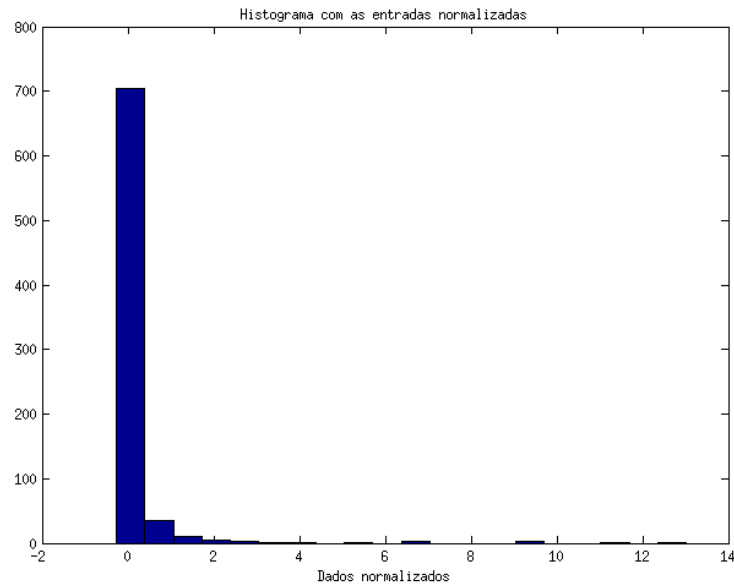


Figura 3: Dados normalizados.

### 3.1 Variáveis participantes do modelo - Filtro de correlação

A fim de obter um resultado mais representativo nos preditores a serem realizados, aplica-se um filtro de correlações nas 20 variáveis que inicialmente foram propostas para determinar a previsão do número de casos de dengue para a próxima semana. Para isso, usamos o programa `calc_corr2.m`, disponibilizado pelo professor. Obtem-se o gráfico mostrado na figura 4 a seguir:

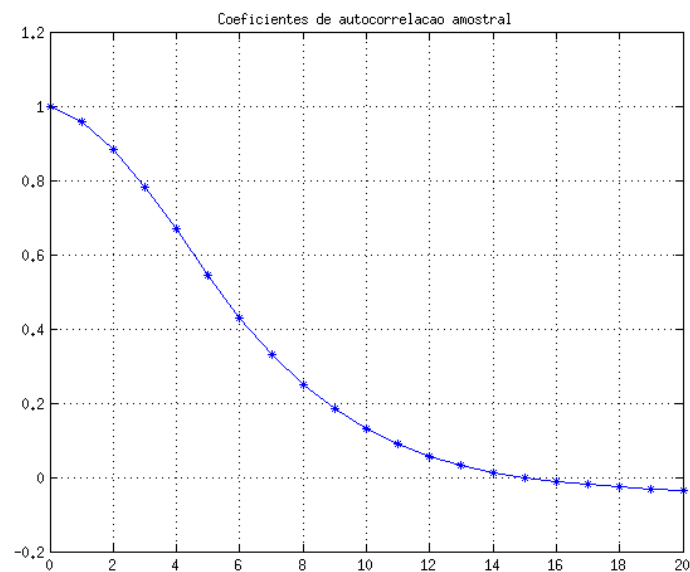


Figura 4: Correlação das 20 variáveis escolhidas como entrada.

Observa-se neste gráfico que as cinco primeiras variáveis utilizadas no modelo são as mais correlatas à saída. Em termos numéricos, as variáveis a partir da sexta apresentam correlações inferiores

a 0.5. Conclui-se, portanto, que em termos de pertinência, as cinco variáveis iniciais são as mais importantes para compor o valor a ser predito.

## 3.2 Síntese de um preditor linear

Para esta etapa, define-se  $M = 5$  a dimensão do espaço de entrada (consultar seção 3.1) e  $R = 1$ , a de saída. Em outras palavras, utilizaremos dados de 5 semanas anteriores para determinar o resultado da 'semana seguinte'. As funções utilizadas para a construção da série e nos cálculos dos preditores encontram-se, respectivamente, nos programas 1 e 2 na seção **Anexos** no fim deste documento.

Utilizaremos a estratégia de *k-folds cross-validation* para estimar o melhor valor do parâmetro  $c$ . Neste caso, adota-se  $k = 10$ .

### 3.2.1 Caso não regularizado

A resolução de  $\vec{b} = (A^T A)^{-1} A^T Y$  utilizando apenas o conjunto de treinamento produz o vetor, cujos coeficientes encontram-se na tabela abaixo, e um erro quadrático médio de 0.2500.

$$\vec{b}_{nreg}^T = [-0.1714 \quad 0.2349 \quad -0.3289 \quad -0.0528 \quad 1.2298 \quad -0.0000]$$

### 3.2.2 Caso regularizado

A utilização de um parâmetro  $c$  adicional e da estratégia *k-fold cross-validation* permite a adequação mais correta dos dados de entrada aos de saída, conforme observado em sala de aula. A execução do programa `resolve_sistema_k_folds.m` da seção **Anexos** gera o gráfico, em escala semi logarítmica, contido na figura 5, que relaciona  $c$  com a média do erro quadrático médio em cada uma das 10 execuções do programa (uma execução para cada uma das pastas de validação).

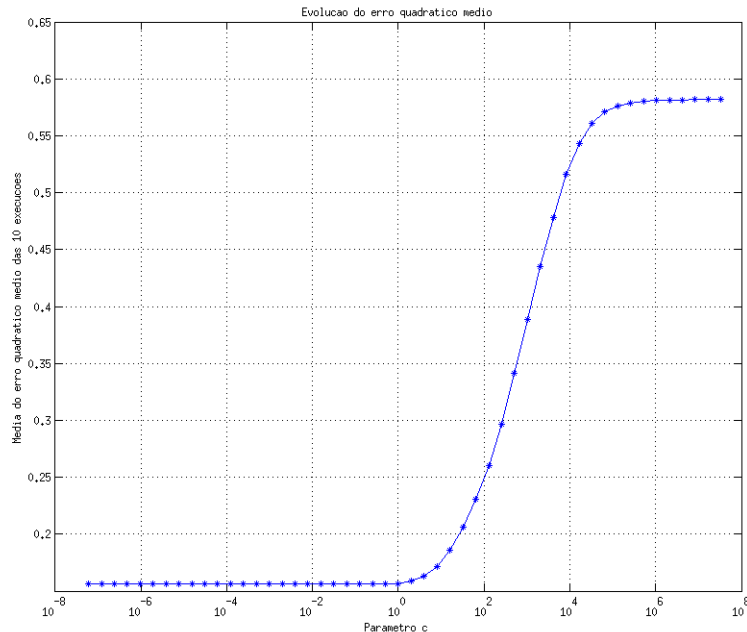


Figura 5: Erro quadrático médio em função do parâmetro  $c$ .

Ressalta-se a presença de um mínimo local, para  $c_1 = 2^{-1} = 0.5$ , valendo 0.15602. Esse erro é inferior a aquele obtido ao caso não regularizado, já que o preditor obtido neste caso adapta-se melhor a todo conjunto dos dados e não somente a aqueles que só foram usados no treinamento.



Enfim, explicitamos os vetores  $\vec{b}$  para  $c = 2^{-1}$ :

$\vec{b}_1$	-0.1673	0.2193	-0.3182	-0.0397	1.2160	0.0023
$\vec{b}_2$	-0.1685	0.2244	-0.3230	-0.0418	1.2195	0.0014
$\vec{b}_3$	-0.1669	0.2195	-0.3199	-0.0379	1.2158	0.0015
$\vec{b}_4$	-0.1674	0.2194	-0.3185	-0.0390	1.2155	0.0025
$\vec{b}_5$	-0.1680	0.2217	-0.3191	-0.0400	1.2164	0.0008
$\vec{b}_6$	-0.1675	0.2199	-0.3191	-0.0390	1.2159	0.0018
$\vec{b}_7$	-0.1662	0.2329	-0.3552	-0.0222	1.2199	-0.0014
$\vec{b}_8$	-0.1676	0.2352	-0.3412	-0.0347	1.2221	0.0008
$\vec{b}_9$	-0.1664	0.2202	-0.3225	-0.0365	1.2160	0.0009
$\vec{b}_{10}$	-0.0832	-0.0739	0.1328	-0.1357	1.0721	-0.0105

Destaca-se que os valores para as componentes de uma mesma coluna não apresentam uma grande variação entre si, isto é, os coeficientes do modelo autoregressivo que produzem o menor erro médio possuem um comportamento bem definido.

### 3.3 Síntese de uma rede neural MLP

Os resultados contidos nas seções 3.3.1 e 3.3.2 foram obtidos do programa `numberNeuronsMLP.m`, contido no trecho de código 3 na seção **Anexos** no fim deste documento. Em poucas palavras, este programa automatiza o processo de treinamento de análise dos resultados de uma rede MLP para diferentes quantidades de neurônios e iterações. Ele utiliza as funções disponibilizadas pelo professor, que foram adaptadas somente para receber parâmetros de entrada no lugar de requerir os dados ao usuário. Neste programa, há dois laços: o mais externo é responsável por modificar o número de iterações do algoritmo otimizador, escolhendo valores no conjunto  $i \in \{50, 100, 150, 200, 300, 400 \dots 1000\}$  e o mais interno, o número de neurônios na camada intermediária no conjunto  $n \in \{5, 6, 7 \dots 20\}$ . Para cada valor de  $i$  e de  $n$  treina-se 10 MLPs (uma para cada configuração das *folds*) e calcula-se as médias dos seus desempenhos. Após obtermos as MLPs para todos os valores de  $n$ , construímos um gráfico com as médias dos erros de validação e de teste e, após todos os valores de  $i$ , desenhamos um último gráfico, com os desempenhos médios para cada valor de  $i$ . As próximas seções serão dedicadas às devidas explicações sobre os resultados.

#### 3.3.1 Determinação do número de iterações

A figura 6 mostra um gráfico que relaciona o desempenho médio das MLPs com o número de iterações. Observa-se que os comportamentos dos erros de teste e de validação são similares, isto é, quando uma apresenta um valor elevado, a outra também apresentará. Esta afirmação explica-se pela capacidade de generalização das MLPs: se uma rede é treinada de forma que o conjunto de validação seja utilizado, é possível atingir um maior grau de flexibilidade a todos os dados e, assim, o erro para dados novos não levados em consideração (como aqueles do conjunto de teste) não se distanciará fortemente do erro de validação. Sendo assim, utilizaremos o valor que possui o maior custo benefício entre os dois erros, isto é,  $i = 1000$ , que produz um erro médio de validação igual a 0.1588 e um erro médio de teste valendo 0.1813.

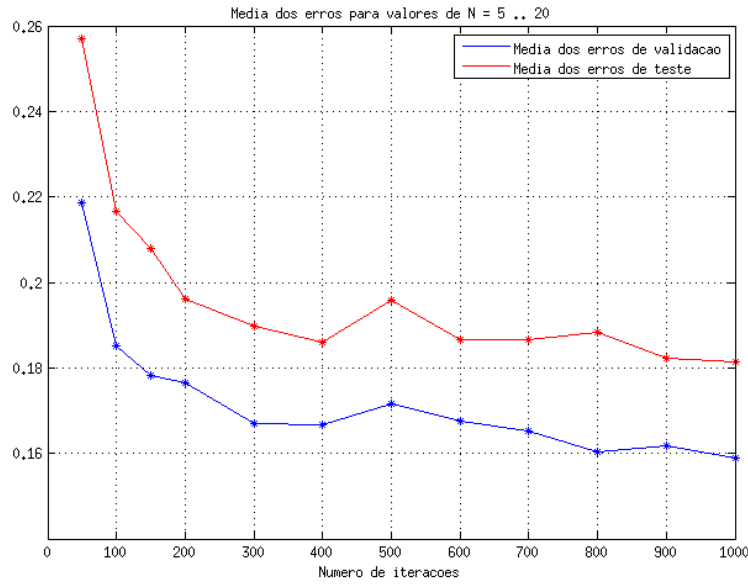


Figura 6: Média de todas as MLPs, calculadas para todo valor de  $n$ , para cada  $i$ .

### 3.3.2 Determinação do número de neurônios na camada intermediária

Uma vez determinado o melhor número de iterações, é possível estabelecer o número de neurônios que melhor se adequa à aplicação. Para isso, utilizamos o gráfico da figura 7.

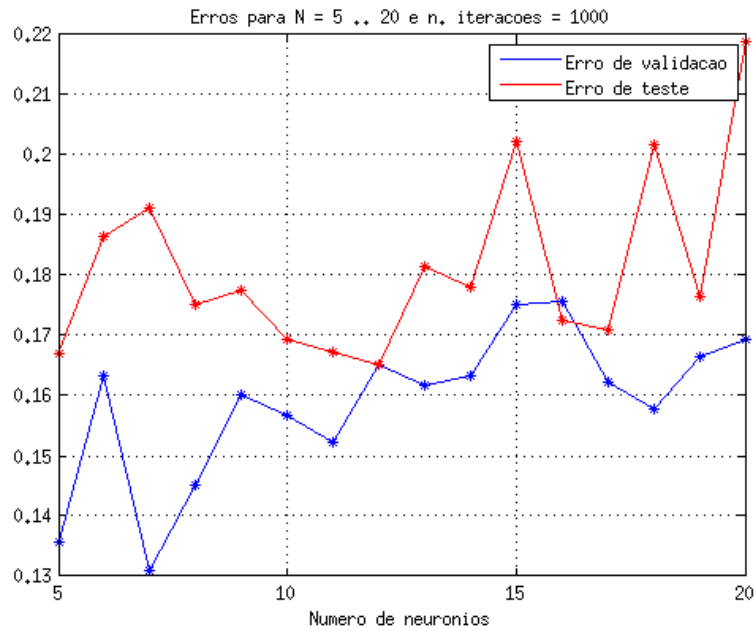


Figura 7: Média de todas as MLPs, calculadas para cada valor de  $n$ , para  $i = 1000$ .

O melhor custo benefício entre os dois erros é, neste caso  $n = 5$ , apresentando erro médio de validação de 0.1356 e erro médio de teste de 0.1671. Destaca-se que, para  $n = 7$ , temos o valor mínimo do erro de validação, mas, ao mesmo tempo, um erro relacionado ao teste muito elevado. Por esta razão, este número de neurônios não foi escolhido. Observa-se também que as duas curvas não apresentam comportamentos definidos, isto é, os erros para diferentes valores de  $n$  são muito distintos entre si.

Os demais resultados para outros valores de  $i$  e  $n$  podem ser observados na figura 12 na seção **Anexos**.

### 3.4 Análise dos resultados dos preditores e MLPs

A pior performance entre os três preditores implementados nesta lista foi a do *preditor linear não regularizado*, para qual só foi utilizado o conjunto de treinamento para o cálculo do erro quadrático médio, que foi 0.2500. Esse resultado era perfeitamente esperado, já que, uma vez o conjunto de validação não foi utilizado, a capacidade de generalização do preditor é comprometido.

Em segundo lugar, destaca-se *preditor linear regularizado*, cujo erro quadrático médio vale 0.15602. A introdução de um parâmetro  $c$  adicional e a sua determinação ótima juntamente ao conjunto de validação provaram que o desempenho obtido pode ser melhorado de maneira significativa, visto que o modelo linear mantém-se o mesmo. Isso significa que nesta oportunidade, observa-se uma maior flexibilidade do preditor a todos os dados.

Enfim, para o caso das MLPs, obtem-se um erro médio quadrático de validação de 0.1356, para  $i = 1000$  e  $n = 5$ . É necessário dizer que este resultado foi conseguido com base em uma quantidade de processamento muito superior aos dois casos precedentes, uma vez que, no total, foram treinadas  $10 * \text{card}(n) * \text{card}(i) = 10 * 15 * 12 = 1800$  MLPs (uma para cada configuração das *folds*, cada valor de  $n$  e  $i$ ). Foram utilizados conjuntos de validação e teste para otimizar ainda mais a adequação da rede aos dados. A rede neural é, portanto, a melhor opção para prever a série temporal da dengue em São Paulo.

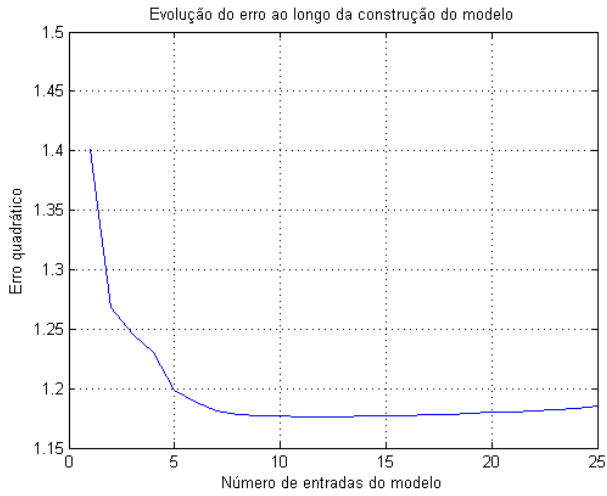
## 4 Conclusões Finais

## Bibliografia

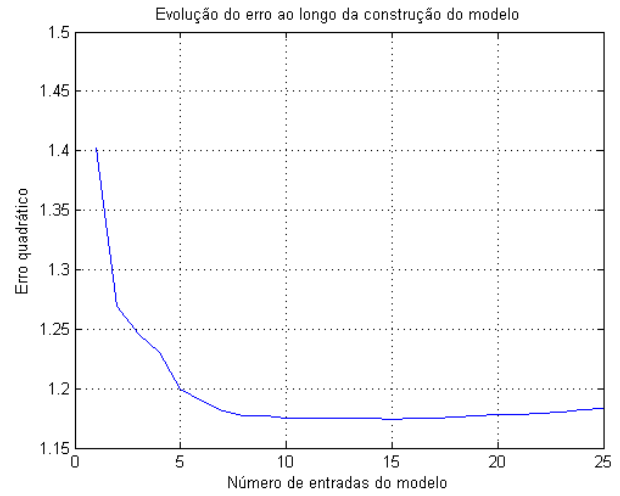
- dasdsa

## 5 Anexos

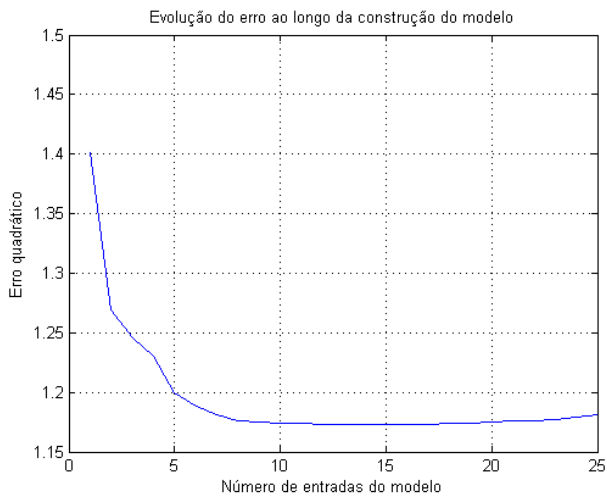
A figura 8 possui todos os resultados das execuções do *forward selection*. Observa-se que o comportamento das curvas em todas elas é muito parecido, apresentando um mínimo para um número de variáveis próximo de 15. (*Rever análise completa na seção 2 item 3c*).



(a) Resultado da execução 1.



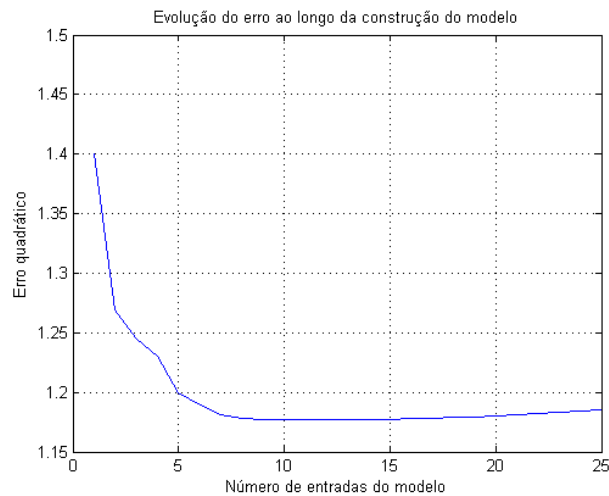
(b) Resultado da execução 2.



(c) Resultado da execução 3.



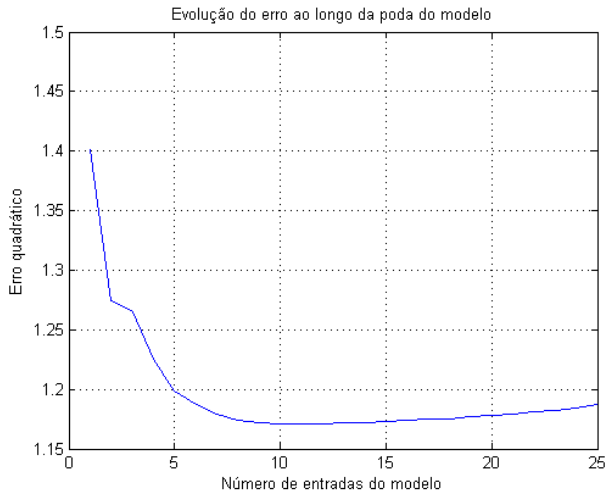
(d) Resultado da execução 4.



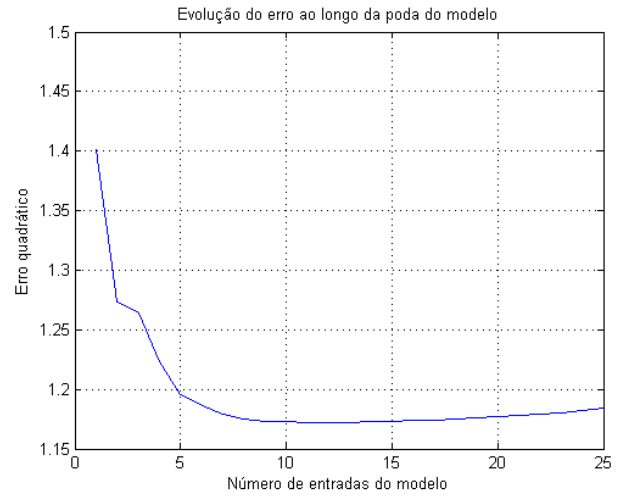
(e) Resultado da execução 5.

Figura 8: Resultados das execuções da *forward selection* para os dados de `sunspot.mat`.

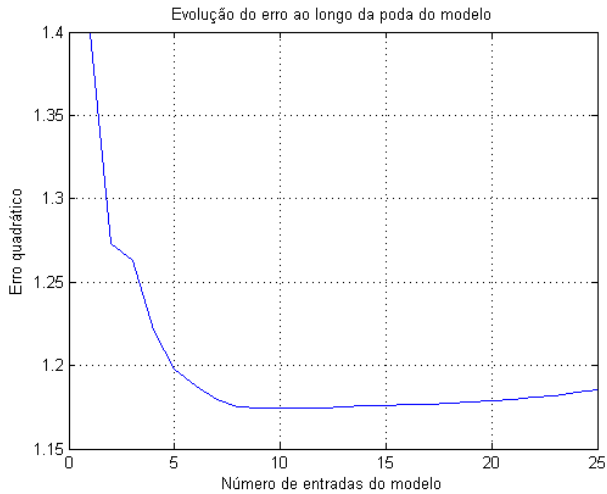
A figura 9 possui todos os resultados das execuções do *backward elimination*. Assim como o caso anterior, observa-se que o comportamento das curvas em todas elas é muito parecido, apresentando um mínimo para um número de variáveis próximo de 15. (*Rever análise completa na seção 2 item 3c*).



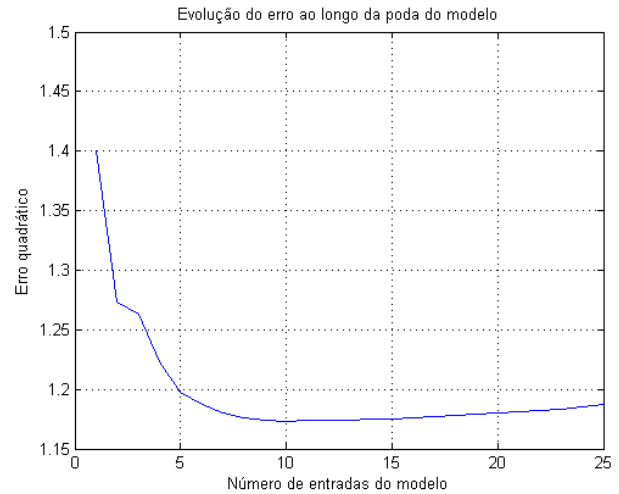
(a) Resultado da execução 1.



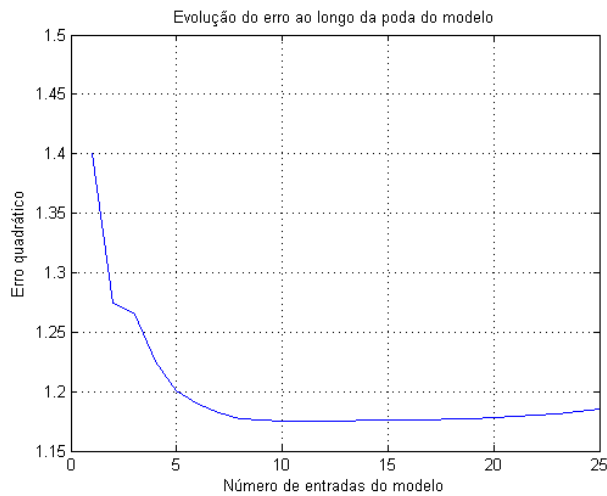
(b) Resultado da execução 2.



(c) Resultado da execução 3.



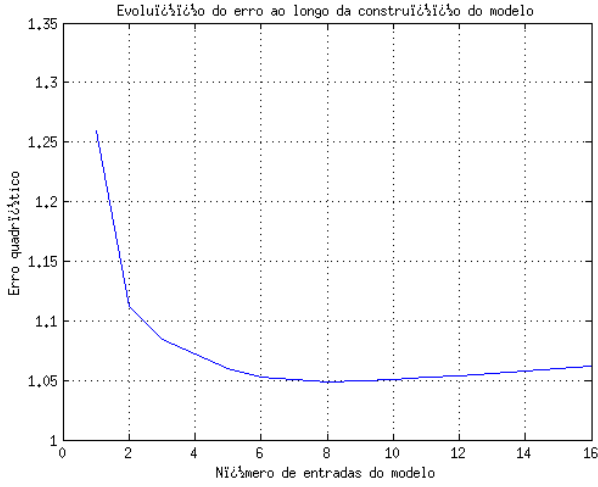
(d) Resultado da execução 4.



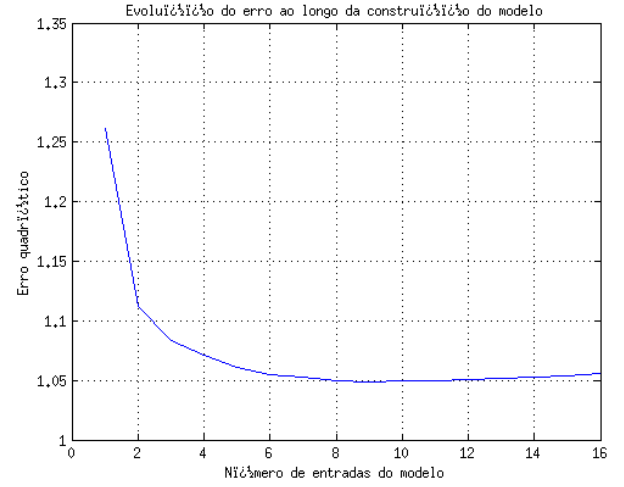
(e) Resultado da execução 5.

Figura 9: Resultados das execuções da *backward elimination* para os dados de `sunspot.mat`.

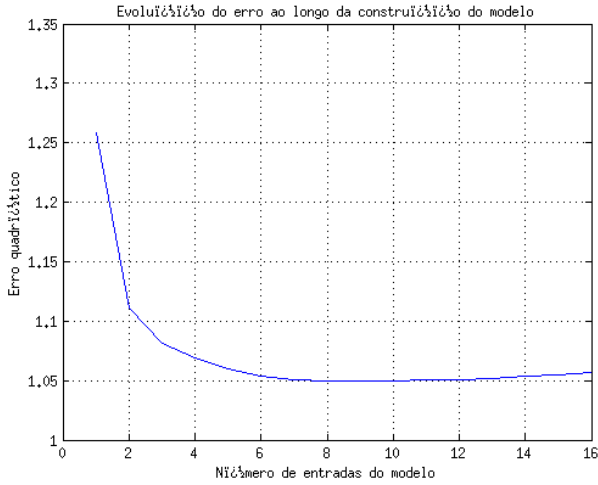
A figura 10 possui todos os resultados das execuções do *forward selection* para os dados contidos em `wineq.mat`. Observa-se que o comportamento das curvas em todas elas é muito parecido, apresentando um mínimo para um número de variáveis próximo de 8. (*Rever análise completa na seção 2 item 4c*).



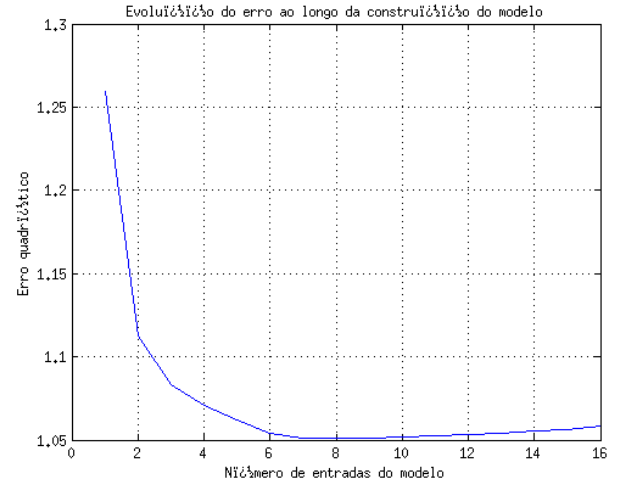
(a) Resultado da execução 1.



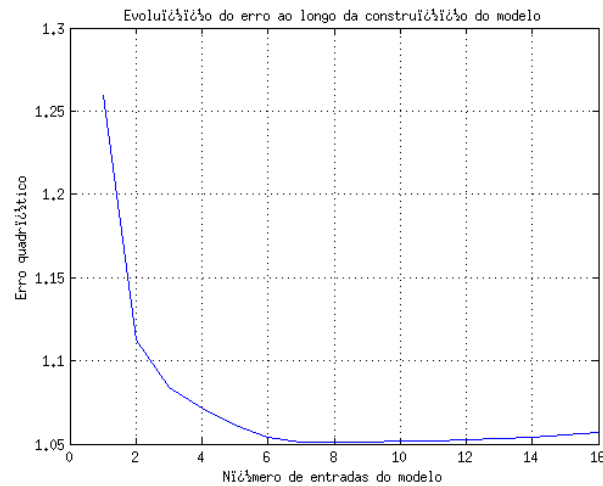
(b) Resultado da execução 2.



(c) Resultado da execução 3.



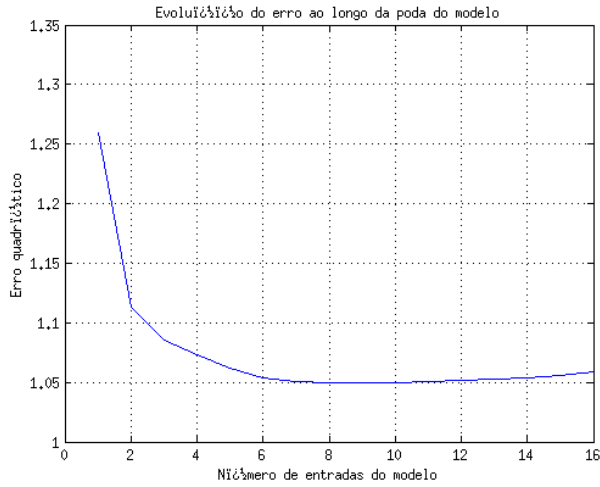
(d) Resultado da execução 4.



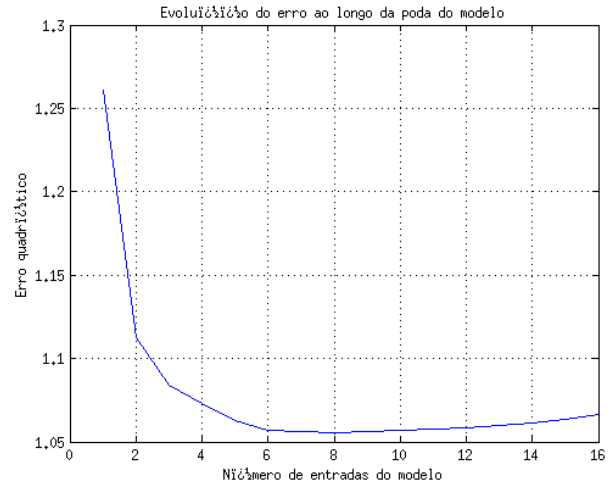
(e) Resultado da execução 5.

Figura 10: Resultados das execuções da *forward selection* para os dados de `wineq.mat`.

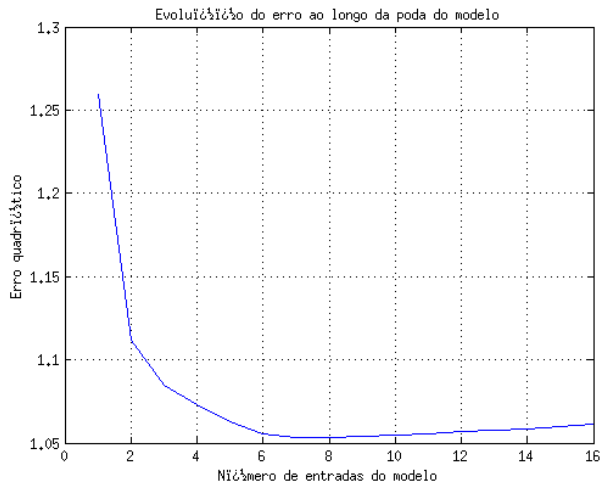
A figura 11 possui todos os resultados das execuções do *backward elimination* para os dados contidos em `wineq.mat`. Observa-se que o comportamento das curvas em todas elas é muito parecido, apresentando um mínimo para um número de variáveis próximo de 10. (*Rever análise completa na seção 2 item 4c*).



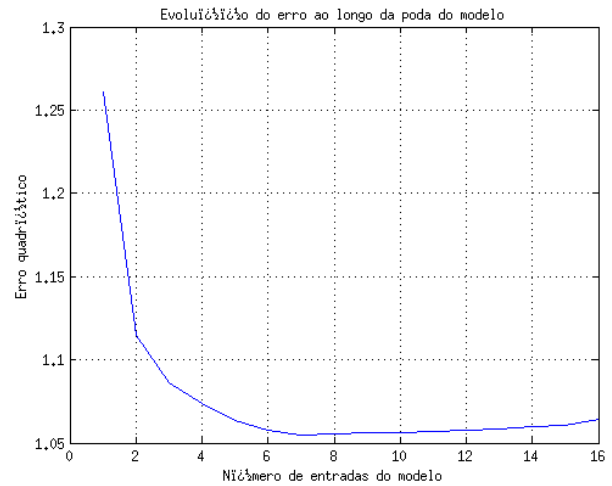
(a) Resultado da execução 1.



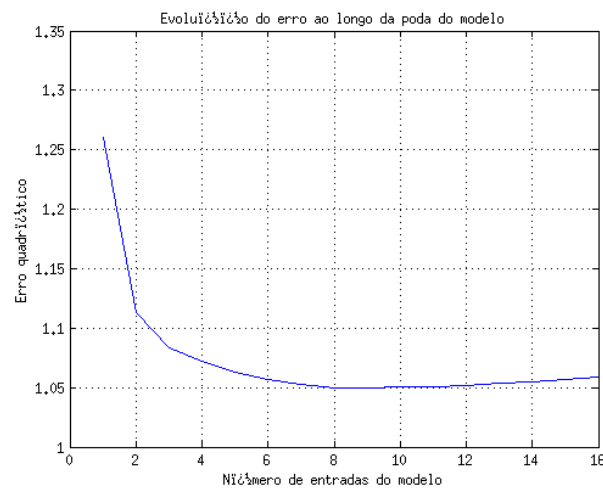
(b) Resultado da execução 2.



(c) Resultado da execução 3.



(d) Resultado da execução 4.



(e) Resultado da execução 5.

Figura 11: Resultados das execuções da *backward elimination* para os dados de `wineq.mat`.

Programa 1: gera\_dados.m - cria a sequência temporal.

```

1 function [X, Y] = gera_dados(dados, saida, m, r)
2
3     load(dados);
4
5     data = dengue_SP;
6
7     data = (data - mean(data))/std(data);
8     n_data = length (data);
9
10    X = [];
11    Y = [];
12    Z = [];
13
14    for j=(m+r):n_data,
15        Z = [Z; data( (j-m-(r-1)):j, 1 )'];
16    end
17
18    X = Z(:, 1:m);
19    Y = Z(:, (m+1):(m+r));
20
21    save(saida, 'X', 'Y');
22
23 end

```

Programa 2: resolve\_sistema\_k\_folds.m - calcula preditores.

```

1 function [b_nreg, erro_nreg, b_reg, erro_reg, erro_reg_tr] = resolve_sistema_k_folds(
2     dados,k)
3
4 close all;
5
6 k = 10;
7 M = 5;
8 R = 1;
9
10 [X, Y] = gera_dados('dengue_SP.mat', 'resultado.mat', M, R);
11
12 Tr = length(X(:,1));
13 V = round(Tr/k);
14 Te = 0;
15
16 A_nreg = [ X ones(Tr ,1) ];
17 ATA = (A_nreg'*A_nreg)\eye(M + 1);
18
19 b_nreg = ATA*A_nreg'*Y;
20 erro_nreg = sqrt((norm(A_nreg*b_nreg - Y))^2/Tr);
21
22 erro_reg_k_folds = [];
23 erro_reg_tr_k_folds = [];
24 b_k_folds = [];
25
26 for i = 1:k
27
28     k_fold_set = (i - 1)*V + 1 : i*V;
29
30     conjunto_treinamento = setdiff(1:Tr, k_fold_set);
31
32     Atr = [ X(conjunto_treinamento, :) ones(Tr - V - Te ,1) ];
33     ATA = (Atr'*Atr)\eye(M + 1);
34     Ytr = Y (conjunto_treinamento,:);
35

```



```

36     b_nreg = ATA*Atr'*Ytr;
37     erro_nreg = sqrt((norm(Atr*b_nreg - Ytr))^2/(Tr - V - Te));
38
39     Aval = [ X(k_fold_set, :) ...
40             ones(V,1) ];
41     Yval = Y(k_fold_set, :);
42
43     erro_reg = [];
44     erro_reg_tr = [];
45     b_reg = [];
46     n_b_reg = [];
47
48     for c_teste = -24:25
49
50         ATA_ = (Atr'*Atr + (2^c_teste)*eye(M + 1)) \ eye(M + 1);
51
52         b = ATA_*Atr'*Ytr;
53
54         b_reg = [b_reg b];
55         n_b_reg = [n_b_reg norm(b)];
56
57         erro_reg = [erro_reg sqrt((norm(Aval*b - Yval)^2)/V)];
58         erro_reg_tr = [erro_reg_tr sqrt((norm(Atr*b - Ytr)^2)/V)];
59     end
60
61     b_k_folds = cat(3, b_k_folds, b_reg);
62     erro_reg_k_folds = [erro_reg_k_folds; erro_reg];
63     erro_reg_tr_k_folds = [erro_reg_tr_k_folds; erro_reg_tr];
64 end
65
66
67 figure
68 semilogx( 2.^(-24:25), mean(erro_reg_k_folds));
69 hold on;
70 semilogx( 2.^(-24:25), mean(erro_reg_k_folds), '*');
71 hold off;
72 title('Evolucao do erro quadratico medio');
73 ylabel('Media do erro quadratico medio das 10 execucoes');
74 xlabel('Parametro c');
75 grid on
76
77 [menor_erro, menor_index] = sort(mean(erro_reg_k_folds), 'ascend');
78
79 disp(['Vetores b para c = ' int2str(menor_index(1) - 25) ' e erro medio de ' num2str(
80     menor_erro(1))]);
81 b_k_folds(:,menor_index(1),:)
82 end

```

Programa 3: numberNeuronsMLP.m - Automatiza treinamento e análise de redes MLP.

```

1 function numberNeuronsMLP (N)
2
3 close all;
4
5 erro_tot_avg_iter = [];
6 mean_eqmv_min_avg_itr = [];
7
8 neurons_interval = 5:N;
9 iter_interval = [50:50:200 300:100:1000];
10
11
12 for n = iter_interval

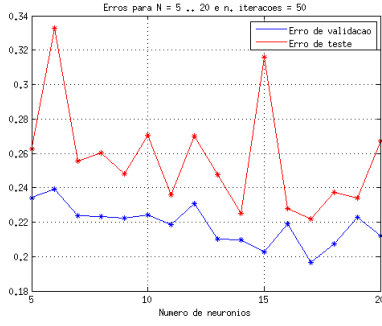
```

```

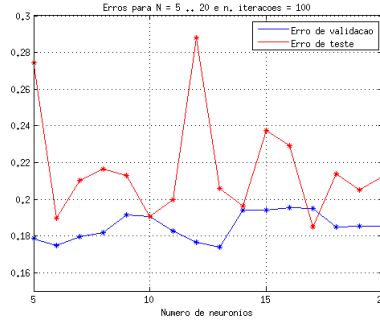
13
14 mean_eqmv_min_avg_array = [];
15 error_tot_avg_array = [];
16
17 for i=neurons_interval
18
19     disp(['=====' num2str(n) ' ITERACOES =====' num2str(i) ' NEURONS
           =====']);
20
21     nnlh_k_folds('matrizes', 10, i, 1, n);
22
23     [error_tot_avg mean_eqmv_min_avg eqm_ens] = analysis('matrizes', 1, 10);
24
25     mean_eqmv_min_avg_array = [mean_eqmv_min_avg_array mean_eqmv_min_avg];
26     error_tot_avg_array = [error_tot_avg_array error_tot_avg];
27     disp('=====');
28
29 end
30
31 erro_tot_avg_iter = [erro_tot_avg_iter error_tot_avg_array'];
32 mean_eqmv_min_avg_itr = [mean_eqmv_min_avg_itr mean_eqmv_min_avg_array'];
33
34 [A B] = sort(mean_eqmv_min_avg_array, 'ascend');
35 disp(sprintf('Numero de neuronios para minimizar erro de validacao: %d', B(1) +
              min (neurons_interval) - 1));
36
37 figure
38 plot(neurons_interval, mean_eqmv_min_avg_array);
39 xlabel('Numero de neuronios');
40 hold on;
41 plot(neurons_interval, error_tot_avg_array, 'r');
42 legend('Erro de validacao', 'Erro de teste');
43 plot(neurons_interval, mean_eqmv_min_avg_array, '*');
44 plot(neurons_interval, error_tot_avg_array, 'r*');
45 title(sprintf('Erros para N = 5 .. %d e n. iteracoes = %d', i, n));
46
47 grid on;
48 hold off;
49 end
50
51 figure
52 plot (iter_interval, mean(mean_eqmv_min_avg_itr));
53 hold on;
54 plot (iter_interval, mean(erro_tot_avg_iter), 'r');
55 legend('Media dos erros de validacao', 'Media dos erros de teste');
56 xlabel('Numero de iteracoes');
57 plot (iter_interval, mean(mean_eqmv_min_avg_itr), '*');
58 plot (iter_interval, mean(erro_tot_avg_iter), 'r*');
59 title('Media dos erros para valores de N = 5 .. 20');
60 hold off;
61 grid on;
62
63 end

```

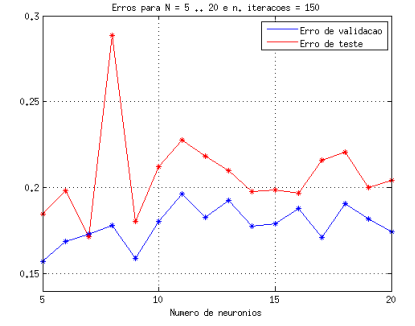
Resultados obtidos para diversos valores do número de iterações do algoritmo otimizador. Destaca-se que os resultados não apresentam nenhum comportamento bem definido, mas tendem a diminuir com o aumento de  $i$ . (*Rever análise completa na seção 3.3.1.*)



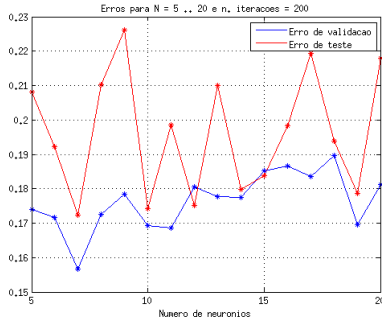
(a)  $i = 50$ .



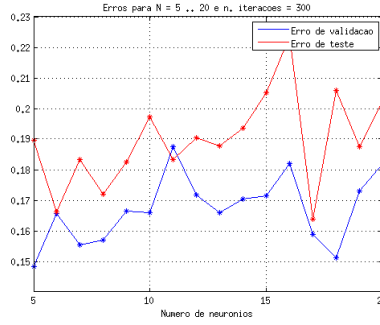
(b)  $i = 100$ .



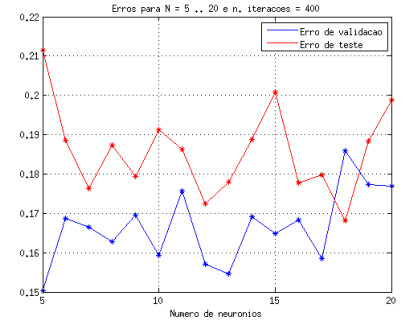
(c)  $i = 150$ .



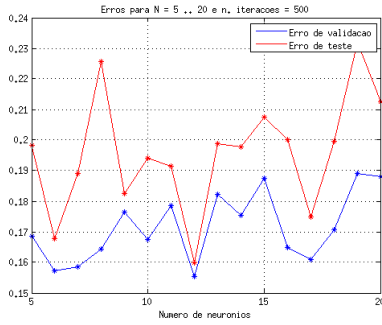
(d)  $i = 200$ .



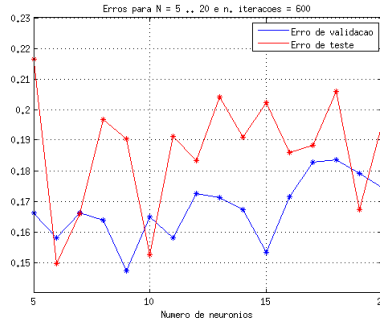
(e)  $i = 300$ .



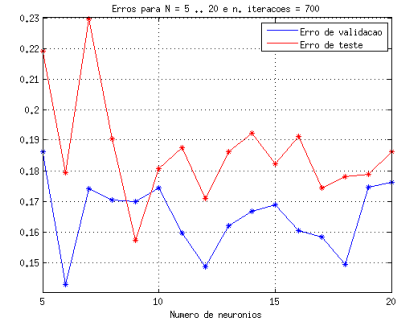
(f)  $i = 400$ .



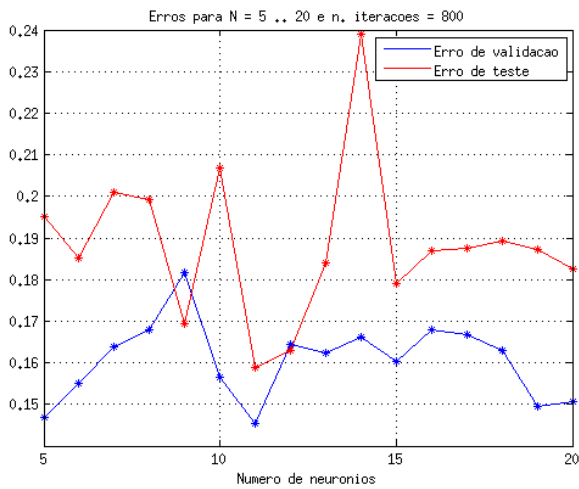
(g)  $i = 500$ .



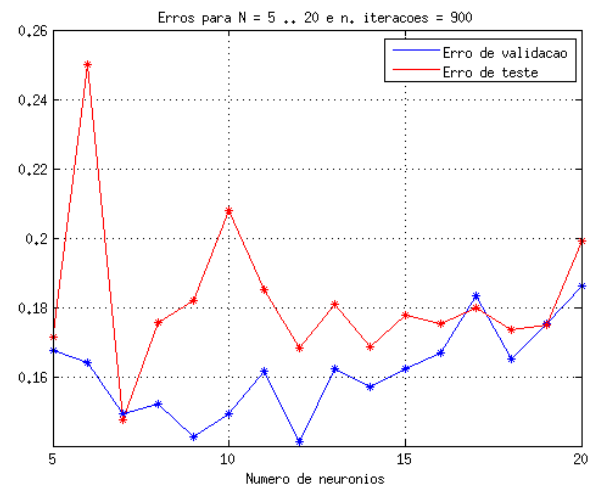
(h)  $i = 600$ .



(i)  $i = 700$ .



(j)  $i = 800$ .



(k)  $i = 900$ .

Figura 12: Médias dos erros de validação e de teste para vários valores possíveis  $i$  de iteração.