EFC 01 – Regressão Linear

Luís Antônio Almeida Lima Vieira (221045) | Nathan Shen Baldon (242448)

28 de setembro de 2022

1 Primeira Parte

1.1 Introdução

A primeira parte do exercício buscou prever valores de uma série temporal dada, utilizando regressão linear com um horizonte de predição L=7. Foi também feita a escolha da melhor quantidade de valores passados (K) a serem usados na predição, através de validação cruzada, tal que o melhor valor de K era aquele que minimizava a raiz quadrada do erro quadrático médio (RMSE).

1.2 Importação e interpretação do conjunto de dados

Inicialmente, o conjunto de dados "*mackeyglass.csv*"foi importado e analisado conforme técnicas expostas por Géron (2019). Observou-se a forma da série temporal (Fig. 1), a organização de suas colunas e como eram os valores disponíveis de t e p.

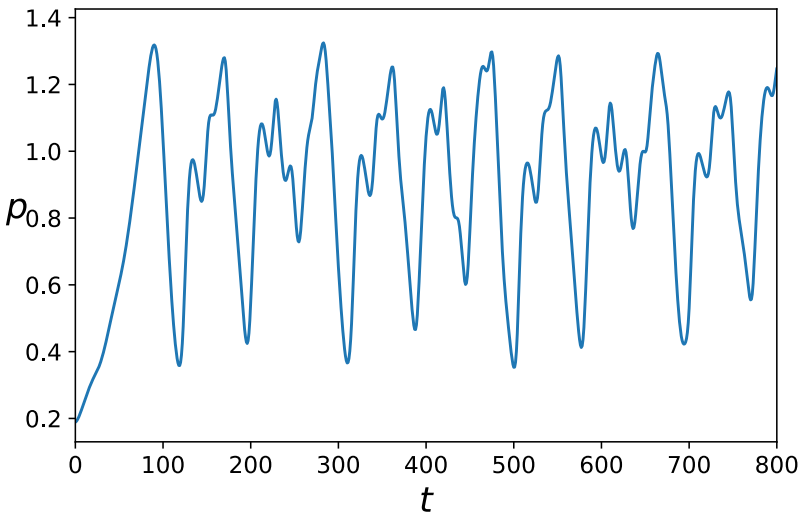

Figura 1: Trecho da série temporal fornecida associada ao sistema de Mackey-Glass no intervalo $t \in [0, 800]$.

1.3 Separação dos Conjuntos

Utilizando a função ".iloc[]"da biblioteca *pandas*, o conjunto original foi separado em três menores, para: treinamento (68% do conjunto inicial), validação (17%) e teste (15%). Para teste, foram escolhidas as 750 últimas amostras, conforme recomendado no enunciado. Para validação cruzada, foi utilizada a estratégia de *holdout*, sendo separadas as últimas 850

amostras. Optou-se por separar as amostras mais recentes por tratar-se da predição de uma série temporal, em que deseja-se sempre prever valores futuros.

1.4 Preparação dos dados para regressão

Para que os dados pudessem ser utilizados em algoritmos de regressão linear, foi necessário separá-los em matrizes que satisfizessem a Equação 1, onde a matriz $\mathbf{\Phi}$ é composta pelos atributos de entrada, a matriz $\mathbf{w}$ é composta pelos parâmetros a serem escolhidos e a matriz $\hat{\mathbf{y}}$, pelos valores futuros estimados.

$$\mathbf{\Phi w} = \hat{\mathbf{y}} \tag{1}$$

Foram montadas as matrizes $\mathbf{\Phi}$ e $\mathbf{y}$ (valores reais esperados), conforme a Figura 2, para serem usadas na entrada da função de regressão linear. Para treinamento, as matrizes foram montadas utilizando apenas valores do conjunto de treinamento (sem validação). Para validação, utilizou-se valores do conjunto de validação e do de treinamento.

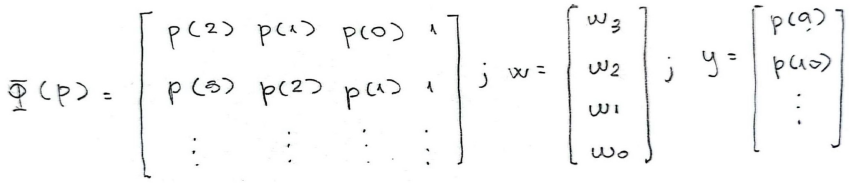

Figura 2: Exemplo do formato das matrizes montadas (L=7 e K=3).

1.5 Regressão Linear e Validação Cruzada

Foi utilizada a função de regressão linear da biblioteca *Scikit-learn* (*sklearn.linear_model.LinearRegression()*), baseada em pseudo-inversa. Com as matrizes montadas a partir do conjunto de **treinamento**, para cada valor de $K \in [1, 50]$, foram estimados os coeficientes $\mathbf{w}$ que minimizavam o MSE (erro quadrático médio), utilizando a função *LinearRegression.fit()* com $\mathbf{\Phi}$ e $\mathbf{y}$ como entradas.

Então, para cada K, com os coeficientes ótimos obtidos, foi feita a estimativa $\hat{\mathbf{y}}$ para o conjunto de **treinamento** e para o conjunto de **validação** (com a função *LinearRegression.predict()*) e foi calculado o RMSE (raiz quadrada do erro quadrático médio, Equação 2) da estimativa feita para cada conjunto (validação e treinamento), conforme Figura 3.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{2}$$

Obteve-se $K = 50$ como melhor número de amostras passadas a serem utilizadas, sendo este o valor que apresentou menor RMSE ($RSME_{min_validation,K=50} = 0,06253$ e $RSME_{min_train,K=50} = 0.04088$), tanto para o conjunto de treinamento, como para o de validação. Assim, estimou-se os valores do conjunto de teste com $K = 50$ (Fig. 4), obtendo-se $RMSE_{teste} = 0.06275$.

Note que esperava-se haver certa concavidade no gráfico da predição na Figura 3, de forma que o melhor K seria menor que 50. No entanto, a função utilizada da biblioteca *Scikit-learn* tem fatores de estabilização, de forma que o $RMSE$ continua caindo conforme K aumenta.

Analisando a predição obtida do conjunto de teste, nota-se que foi possível obter um resultado bom, de forma que os valores estimados oscilam de forma muito semelhante e próxima

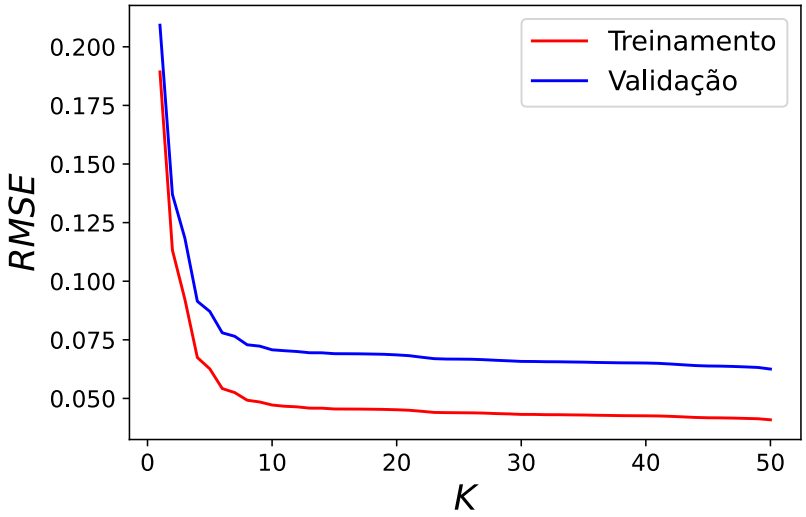

Figura 3: Valores de RMSE das estimativas para os conjuntos de treinamento e de validação em função de $K \in [1, 50]$, utilizando regressão linear.

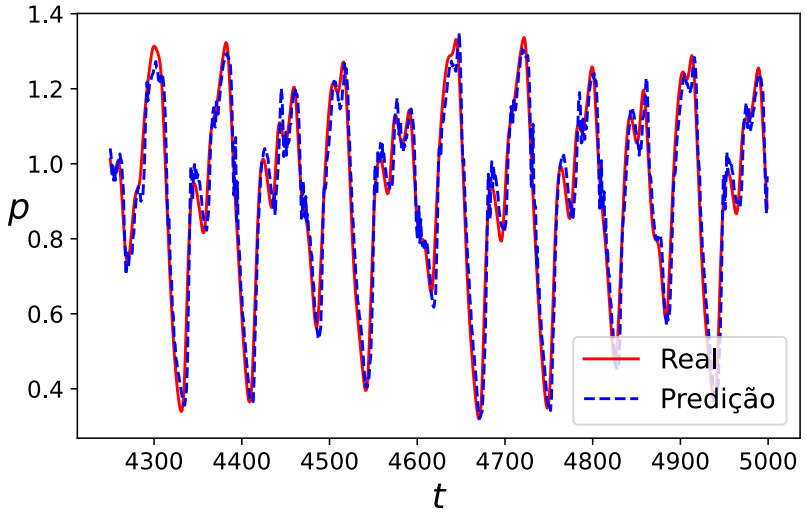

Figura 4: Valores reais e estimados do conjunto de teste da série temporal dada ($t \in [4250, 5000]$) com regressão lineare $K = 50$.

dos valores reais, destoando mais nos picos e vales. Veja ainda que o RMSE (raiz do erro quadrático médio) do **teste** foi muito próximo daquele mínimo encontrado na **validação**.

2 Segunda Parte

2.1 Introdução

Para esta parte do exercício, também foi explorada a regressão linear para a predição linear de parâmetros, porém neste caso foram utilizadas como entradas do preditor valores obtidos a partir de transformações não lineares do vetor $x(n)$, que contém os atrasos da série temporal. No caso anterior, o preditor recebeu diretamente os vetores de atrasos $x(n)$ para fazer

a regressão baseada nos valores de y com horizonte de predição 7. Para este caso foi realizada a seguinte transformação da matriz $x(n)$ de forma a obter uma nova matriz de atrasos $x'(n)$:

$$x'_k(n) = tanh(W_k^T x(n)) \tag{3}$$

Sendo W_k^T uma matriz de vetores w_k compostos de números gerados aleatoriamente de acordo com uma distribuição uniforme, para k = 1,....,V e n = 0,....., N-1. Como no item anterior, K é número de colunas da matriz x(n) (denominada $\mathbf{\Phi}$ na primeira parte), sendo K o número de amostras passadas utilizadas. O novo atributo V se refere ao número de colunas da nova matriz x'(n). N se refere à quantidade de amostras no conjunto de dados.

A estrutura utilizada nesta questão é uma rede neural denominada ELM (*Extreme Learning Machine*) Huang, Zhu e Siew (2006). Ao contrário da primeira questão, neste exercício, devido à presença da função tanh(), há um intervalo admissível para os coeficientes aleatórios das projeções. Para tratar isso, foram utilizadas técnicas de normalização para alterar a escala da série. Ao final as normalizações foram desfeitas para a apresentação dos resultados.

2.2 Importação e separação do conjunto de dados

Como no caso anterior, o conjunto de dados "mackeyglass.csv"foi importado e analisado conforme técnicas expostas pelo livro Géron (2019). A partir desses dados importados, foi realizada a separação dos dados em um conjunto de teste, composto pelos últimos 750 dados da série temporal, e um conjunto de treinamento composto pelos dados restantes. Para a validação cruzada, foi escolhida a técnica de *holdout*, escolhendo 20% dos últimos dados de treinamento para validação. Com isso, assim como no caso anterior, os primeiros 3400 dados do conjunto "mackeyglass.csv"foram reservados para treinamento, os próximos 850 dados foram reservados para validação cruzada e os últimos 750 dados foram reservados para teste. A separação dos conjuntos de dados foi feita a partir de uma função com o uso de funcionalidades da biblioteca *pandas*. Para o tratamento posterior, foi feita a conversão destes conjuntos em listas da biblioteca *numpy*.

2.3 Normalização/Padronização dos dados

Como neste caso ocorrerá uma transformação não linear com aplicação da função tanh(), é necessário alguns cuidados. Esta função é simétrica em relação ao eixo x, possui saturação para valores elevados em módulo, e apresenta comportamento aproximadamente linear para valores pequenos em módulo. Para valores de entradas maiores do que 1, a função começa a saturar e apresentar valores de saída constantes, sendo que para qualquer entrada maior que 2 a função apresenta resultado unitário. No caso do exercício, não é interessante que dados diferentes fossem mapeados para valor 1 após aplicar a função tanh(). Por isso, é interessante manter todos os dados do conjunto de dados em valores pelo menos entre -2 e 2, de forma que a saída de tanh() não sature para nenhum valor. Devido ao comportamento simétrico em relação ao eixo x da função tanh(), foi escolhida a padronização como forma de escalonar as variáveis de entrada, para manter os dados centrados em 0, simétricos em relação ao eixo x, e com valores pequenos, de forma a não perder informação ou ocasionar em erros elevados após a regressão. A padronização possui a seguinte formulação:

$$x'_i = \frac{x_i - mean(x)}{std(x)} \tag{4}$$

Em que mean(x) representa a média de todos os valores do conjunto de dados e std(x) representa o desvio padrão de todos os valores do conjunto de dados. Nesta operação, cada elemento do

conjunto de dados é subtraído pela média dos valores, com o resultado desta subtração dividido pelo desvio padrão dos valores. Para voltar à escala original, basta fazer a operação inversa ao multiplicar cada valor da lista normalizada pelo desvio padrão e somar o resultado com a média. A padronização foi feita após a separação dos conjuntos, porém como foi utilizada a média e o desvio padrão do conjunto completo, esta operação foi equivalente à padronizar o conjunto completo e separar após normalizado.

2.4 Montagem dos dados para a regressão

Primeiramente, assim como no primeiro caso, foi montada a matriz x(n) (denominada **Φ** na primeira parte) composta pelos atrasos da série temporal. Essa matriz tem dimensão (NxK), sendo K o número de atrasos utilizados representados no número de colunas da matriz, e N o número de linhas, referentes ao número de saídas observadas. Foi considerado um horizonte de predição L de 7, ou seja, a primeira saída observada se refere ao dado com 7 posições a frente da posição do último dado de atraso. Além da matriz de atrasos x(n) também foi montado o vetor de saídas y com dimensão N. A dimensão de y equivale à dimensão do conjunto utilizado subtraída do horizonte de predição e do valor de K e somada com 1. Com isso, como no caso anterior, foram montadas as matrizes x(n) e os vetores y(n) para os conjuntos de treino, validação e teste, lembrando que a matriz x para a validação e teste começa com dados do conjunto anterior. A única diferença na montagem dessas matrizes no exercício 2 é a ausência da coluna de uns na matriz x(n), já que neste caso, ainda haverá um mapeamento antes da regressão. Além disso, estas matrizes e vetores foram montados agora com os dados já normalizados.

Antes da regressão, foi feito o mapeamento de acordo com a equação 3, e para isso houve algumas adaptações. Primeiramente foi realizada a transposição da matriz x(n), antes (NxK) para (KxN). Esta operação foi necessária para efetuar o produto da matriz x(n) com a matriz de valores aleatórios W_k^T, que foi definida como uma matriz (VxK). Esta matriz foi definida a partir de uma distribuição normal de valores de -1 a 1, e multiplicada por um fator de 0,1 de forma a manter os valores do produto $W_k^T x(n)$ da transformação em valores pequenos, de forma a não ocasionar saturação após aplicar a função tanh(). Com isso, o produto $W_k^T x(n)$ se tornou uma matriz (VxN). Ao aplicar tanh na matriz resultante do produto anterior, foi obtida a nova matriz x'(n) de dimensão (VxN). À essa matriz x'(n), foi adicionada uma linha de uns para ficar pronta para a aplicação da regressão linear.

2.5 Regressão e Validação Cruzada

Após a transformação realizada, o processo de regressão linear ocorre igual ao exercício 1, com o vetor y agora normalizado, e com o uso da matriz x'(n). Para este caso, foram fixados o horizonte de predição em 7 e o número K de atrasos em 10. Neste exercício foi adicionada a técnica de *Ridge Regression* para a regularização, de forma à estabilizar e evitar *overfitting* nos dados previstos, ao adicionar à regressão um coeficiente *alpha*, proporcional à norma do vetor de pesos, de forma à melhorar a generalização do modelo. Assim como no caso anterior, foi utilizada a biblioteca *scikit learn*, utilizando neste caso o pacote *Ridge* para a regressão.

Foram feitos treinamentos para valores de V de 1 a 1000 variando com passos de 10, para os conjuntos de validação e treinamento. Primeiramente foi realizado o treinamento sem o *Ridge Regression*, para fins de comparações, os valores do RMSE para este caso estão presentes na figura 5. Após isso, foi feita a *Ridge Regression* variando o parâmetro alpha para os valores 0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 100, 1000. Para cada um desses valores foram feitas

as previsões nos conjuntos de treinamento e teste e calculados os valores do RMSE para as previsões, de acordo com a equação 2. Os valores do RMSE foram calculados após os valores voltarem para a escala original pela operação inversa da padronização. Com isso, foram plotados os gráficos do melhor alfa em função do número V, presente na figura 6, e do melhor RMSE para o melhor alfa em cada V para validação e teste, presente na figura 7.

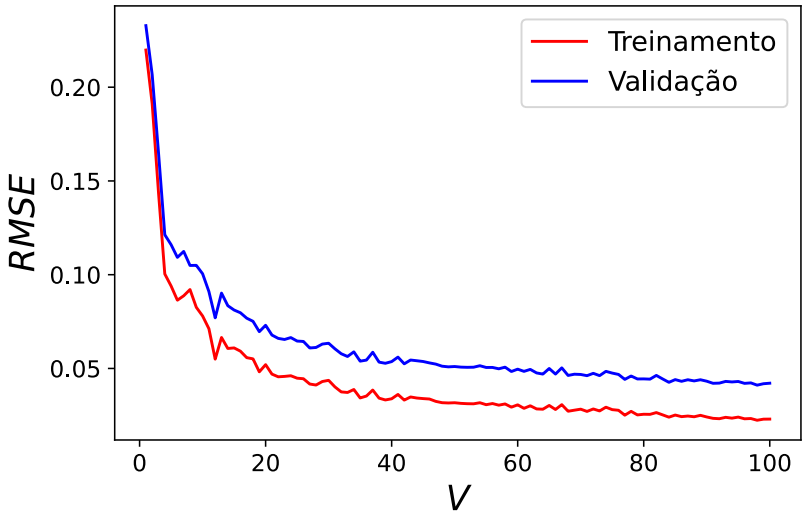

Figura 5: Valores de RMSE das estimativas para os conjuntos de treinamento e de validação em função de $V \in [1, 100]$, utilizando regressão linear.

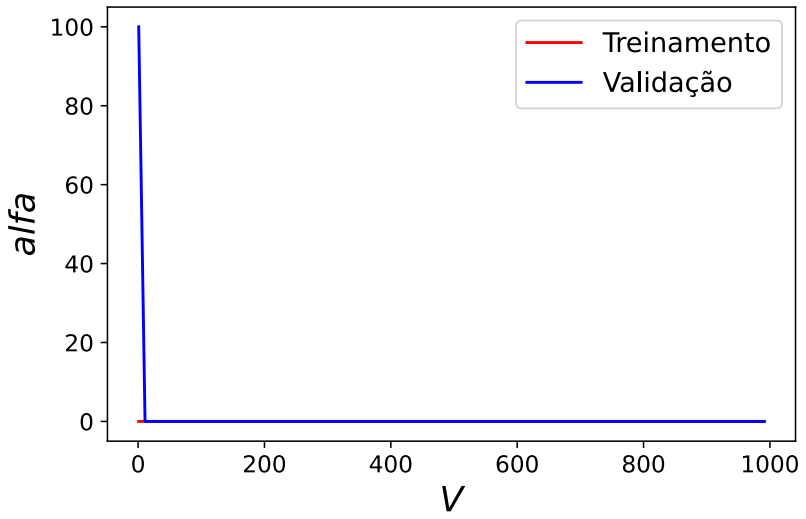

Figura 6: Melhores alfas para cada V para a Ridge Regression.

Com os valores ótimos de V = 939 e alfa = 0, foi refeito o treinamento, agora para todos os dados de validação e treinamento juntos. Com a o modelo resultante desses dados, foi feita a previsão para o conjunto de treinamento. Esta previsão está na figura 8.

É possível observar uma melhora nos resultados nesse segundo exercício com relação ao primeiro exercício. O RMSE ótimo para a validação reduziu pela metade, e para o treinamento reduziu em uma ordem de grandeza, atingindo o valor de 0,0028. O parâmetro alfa não interferiu

muito para o modelo, exceto para valores de V pequenos, em que se faz necessário grandes ajustes. A biblioteca *scikit learn* para a regressão possivelmente tem algum mecanismo de estabilização durante a regressão. Ao calcular a regressão pela fórmula da pseudo-inversa, é possível notar uma piora de resultados com muitas amostras passadas, o que não ocorreu com o uso da biblioteca. Devido à isso, o parâmetro alfa não se mostrou útil ao modelo, sendo que na maioia dos casos as soluções ótimas ocorreram sem o uso do *Ridge Regression*. O RMSE do conjunto de teste foi de 0,0328.

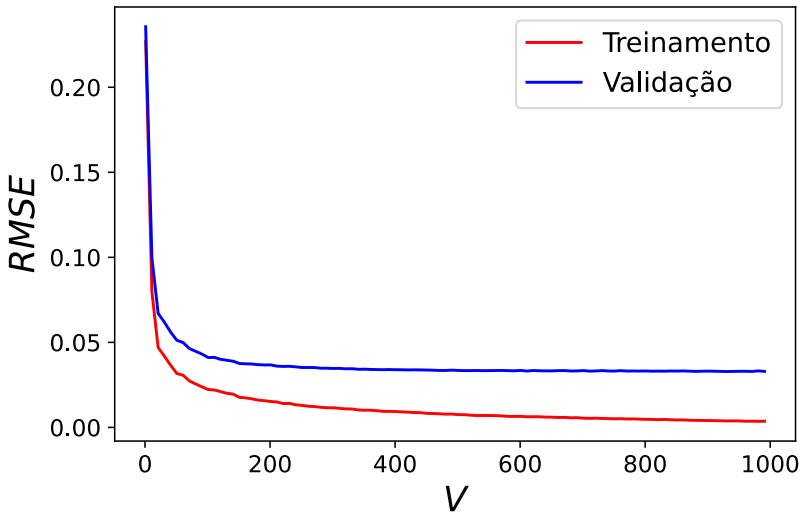

Figura 7: Valores de RMSE das estimativas para os conjuntos de treinamento e de validação em função de $K \in [1, 1000]$, utilizando Ridge Regression

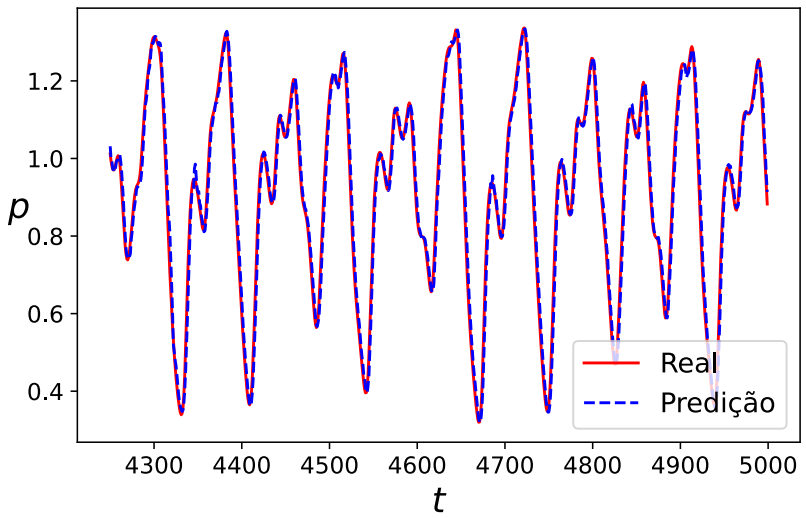

Figura 8: Valores reais e estimados do conjunto de teste da série temporal dada ($t \in [4250, 5000]$) com regressão lineare $V = 939$ e $alpha = 0$.

Referências

GÉRON, A. **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow**. [S.l.]: Alta Books, 2019. ISBN 9788550809021. Disponível em: <https://books.google.com.br/books?id=Z0mvDwAAQBAJ>.

HUANG, Guang-Bin; ZHU, Qin-Yu; SIEW, Chee-Kheong. Extreme learning machine: theory and applications. **Neurocomputing**, Elsevier, v. 70, n. 1-3, p. 489–501, 2006.