João Victor Barbosa Alves

Seleção Incremental de Variáveis para Aprendizado de Máquina utilizando Preditor Linear e Validação Cruzada

Belo Horizonte, Minas Gerais - Brasil

2018

João Victor Barbosa Alves

Seleção Incremental de Variáveis para Aprendizado de Máquina utilizando Preditor Linear e Validação Cruzada

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Universidade Federal de Minas Gerais - UFMG

Escola de Engenharia

Programa de Graduação em Engenharia Elétrica

Orientador: Antônio de Pádua Braga

Belo Horizonte, Minas Gerais - Brasil

2018

João Victor Barbosa Alves

Seleção Incremental de Variáveis para Aprendizado de Máquina utilizando Preditor Linear e Validação Cruzada

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Trabalho aprovado. 10 de dezembro de 2018:

Antônio de Pádua Braga
Orientador

Professor
Convidado 1

Belo Horizonte, Minas Gerais - Brasil
2018

Agradecimentos

Agradeço aos meus pais, Libério e Rosângela, que não mediram esforços para me proporcionar a melhor educação possível e sempre incentivaram o pensamento curioso, o estudo e a dedicação que me permitiram completar essa etapa da minha jornada acadêmica. Agradeço também aos meus amigos, muitos dos quais conheci no decorrer do curso, e que foram porto seguro nos momentos de desânimo e frustração.

Agradeço aos mestres do curso de Engenharia Elétrica que, ao compartilhar seus conhecimentos em sala de aula, ensiraram e inspiraram. Em especial, agradeço ao Professor Antônio de Pádua Braga, com quem pude aprender e trabalhar, ainda que brevemente, no projeto que possibilitou a escrita desse trabalho.

São as perguntas que não sabemos responder que mais nos ensinam.
Elas nos ensinam a pensar.
Se você dá uma resposta a um homem, tudo o que ele ganha é um fato qualquer.
Mas, se você lhe der uma pergunta, ele procurará suas próprias respostas.
(. . .)
Assim, quando ele encontrar as respostas, elas lhe serão preciosas.
Quanto mais difícil a pergunta, com mais empenho procuramos a resposta.
Quanto mais a procuramos, mais aprendemos.
Rothfuss, Patrick - O Temor do Sábio - a Crônica do Matador do Rei - Segundo Dia.

Resumo

Neste trabalho foi apresentado um método de seleção de variáveis *stepwise* utilizando o erro de validação cruzada *leave-one-out* (LOOCV) em regressores lineares como métrica de performance.

O algoritmo proposto faz uso de identidades algébricas conhecidas na literatura para determinar de maneira incremental o erro de LOOCV ao se acrescentar uma variável no conjunto de entradas. Dessa maneira, elimina-se a necessidade do ajuste de novos modelos lineares a cada subconjunto de variáveis avaliado.

Palavras-chave: seleção de variáveis, *features, machine learning, feature selection*, *feature engineering*.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

O aumento da capacidade de armazenamento e processamento de dados tem possibilitado o desenvolvimento de sistemas inteligentes através do aprendizado de máquina. Tal desenvolvimento, por sua vez, permitiu avanços em diversas áreas da computação, microeletrônica e sensoriamento.

Hoje, aplicações tais como pesquisas web, sistemas *anti-spam*, reconhecimento de voz, recomendações de produto e diversas outras são frutos desse progresso. Porém a disponibilidade de quantidades massivas de dados apresenta não só um horizonte vasto de possíveis aplicações, mas também desafios para seleção e processamento desses dados.

1.1 Aprendizado de Máquina

Aprendizado de máquina é o campo da computação responsável por desenvolver e empregar sistemas ou modelos que, através da sua exposição à experiências, são capazes de melhorar sua performance na realização de determinada tarefa (MITCHELL, 1997).

O processo de desenvolvimento de modelos com aprendizado de máquina, ilustrado na Figura 1, pode ser dividido, em linhas gerais, em duas grandes etapas. A primeira delas refere-se à análise e tratamento dos dados. Nessa etapa procura-se identificar correlações entre as informações disponíveis e as variáveis de interesse. Além disso, são explorados possíveis filtros, transformações e outros algoritmos que possam facilitar o aprendizado do modelo. Também é necessário realizar nessa etapa a separação dos dados que serão utilizados para treinamento, validação e teste no decorrer do processo.

Na etapa seguinte objetiva-se obter um modelo capaz de reproduzir o comportamento do sistema que gerou o conjunto de dados. Para tal, um ou mais modelos e algoritmos são selecionados e treinados. Duas características são de fundamental importância e devem ser controladas: a capacidade de representação do sistema (complexidade) e a capacidade de extrapolação das saídas para novas entradas (generalização).

Os algoritmos utilizados nesta segunda etapa podem ser classificados em uma de três categorias, de acordo com as características dos dados disponíveis. (RUSSELL; NORVIG, 2010)

- Na categoria de aprendizado não supervisionado, desenvolve-se sistemas capazes de identificar padrões implícitos em um conjunto de dados não rotulados.
- No aprendizado por reforço, os sistemas se adaptam de acordo com dados de resposta oriundos do ambiente no qual estão envolvidos.

Figura 1: Fluxo geral de desenvolvimento de sistemas de aprendizado de máquina.

Fonte: *Machine Learning: A Gentle Introduction (ABHISHEK, 2018)*. Adaptado.

- Por fim, no aprendizado supervisionado, estão modelos que, a partir de um conjunto de entradas e saídas conhecidas, são capazes de extrapolar a dinâmica do sistema em questão.

1.2 Generalização e Complexidade

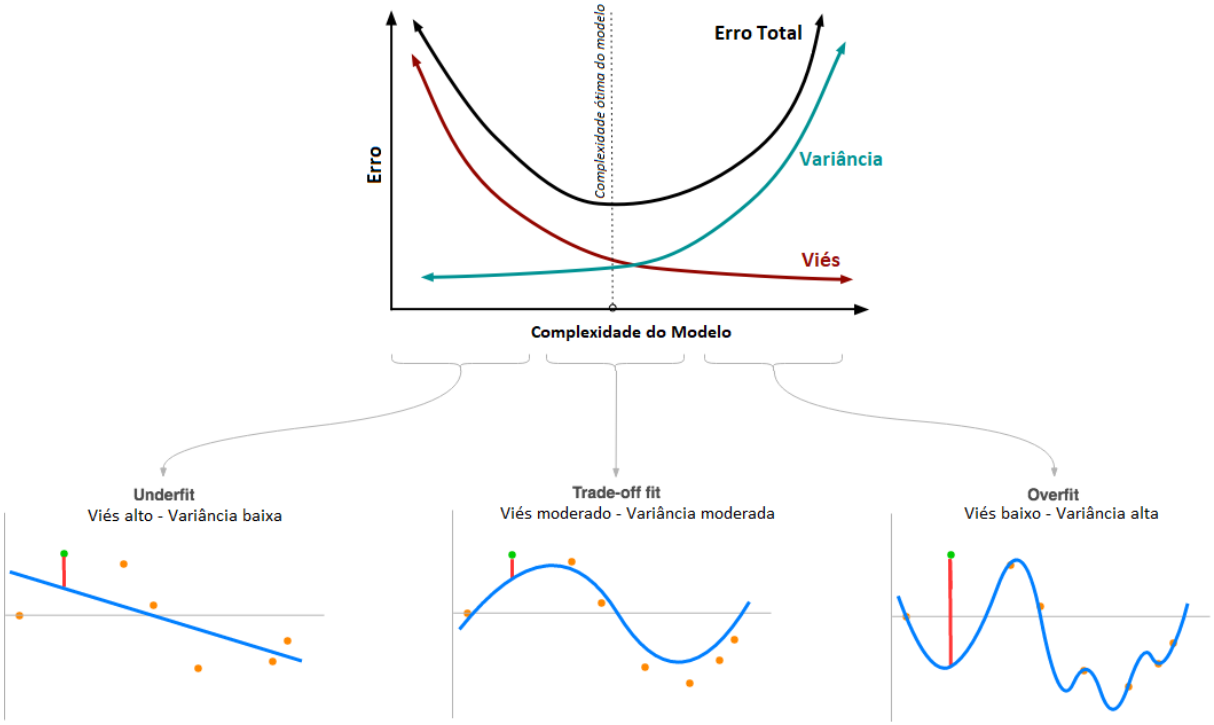
O processo de aprendizado supervisionado é a inferência de um mapeamento de dados de entrada a variáveis de saída a partir de um conjunto de observações. Este pode, portanto, ser interpretado como um ajuste, linear ou não, de uma curva (HAYKIN, 2009). Consequentemente, os modelos obtidos através desse processo também estão sujeitos a *overfitting* e *underfitting*, conforme ilustrado na Figura 2.

Overfitting, ou sobre-ajuste, é caracterizado pela alta performance no conjunto de dados de treinamento e baixa performance em dados nunca observados. Isto é, o modelo assimila desvios causados por erros de medição ou fatores aleatórios presentes no treinamento. Dessa maneira, o erro em relação aos dados de treinamento é reduzido, porém essa melhora não corresponde a uma melhora na representação da realidade.

De maneira similar, o *underfitting*, ou sub-ajuste, é caracterizado pela baixa performance em ambos os conjuntos de dados, indicando que o modelo utilizado não é capaz de

representar de maneira satisfatória a complexidade do sistema real.

Figura 2: Relação entre *underfitting-overfitting*, generalização-complexidade e viés-variância.



Fonte: *Bias and Variance* (CRUELLS, 2017). Adaptado.

Generalização é o termo usado para descrever a capacidade de um sistema de reagir de maneira satisfatória a novos dados. Isto é, após realizado o treinamento, a capacidade de um sistema de realizar predições precisas (sem *overfitting* ou *underfitting*) para dados nunca observados.

Analogamente, pode-se avaliar o modelo em termos de complexidade, onde modelos mais complexos são capazes de ajustar os dados em uma família maior de curvas, porém estão mais sujeitos a *overfitting* e exigem, em geral, uma complexidade amostral (*sample complexity*) maior.

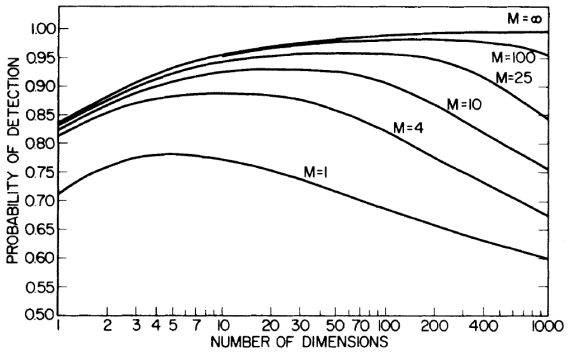
Pode-se ainda avaliar o sistema em termos de viés e variância (*bias-variance tradeoff*), onde o viés representa a parcela do erro atribuída à diferença entre o modelo e a função real do sistema, e a variância representa a parcela devido à sensibilidade do modelo à pequenas variações nos dados.

1.3 O problema da dimensionalidade

Sistemas que se utilizam de aprendizado de máquina apresentam respostas baseadas nos dados de entrada que lhe são apresentados. Dessa maneira, a determinação de quais variáveis são relevantes para o sistema é uma etapa fundamental de seu processo de desenvolvimento.

A inclusão de variáveis irrelevantes resulta no aumento da complexidade do modelo, tornando-o mais suscetível ao *overfitting* e menos interpretável (JAMES et al., 2017, p. 204), possivelmente prejudicando seu desempenho, conforme demonstrado pela Figura 3. Além disso, um número elevado de variáveis de entrada resulta em um efeito conhecido como a maldição da dimensionalidade (*curse of dimensionality*), onde o aumento do número de dimensões (quantidade de variáveis), implica no aumento exponencial do número de amostras necessárias para representar o espaço de variáveis (BISHOP, 2006, p. 34), tal efeito pode ser observado na Figura 4.

Figura 3: Performance em relação ao número de dimensões e amostras.



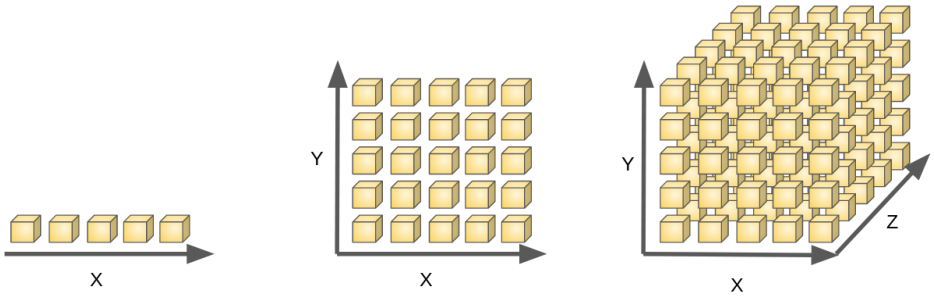
Fonte: A Problem of Dimensionality: A Simple Example (TRUNK, 1979)

É desejável então que se utilize o menor número possível de variáveis de entrada que permitam ao modelo produzir a resposta correta. Uma maneira de realizar essa redução de dimensionalidade é selecionar um subconjunto das variáveis disponíveis, descartando aquelas que apresentarem pouca ou nenhuma relevância para o problema (JAMES et al., 2017, p. 204).

1.4 Motivação

A quantidade e qualidade das variáveis incluídas em um modelo é responsável por parcela significativa de sua complexidade e potencial de generalização. Para o desenvolvimento de um modelo satisfatório, é necessário, portanto, a aplicação de procedimentos adequados para a seleção e processamento dos dados iniciais.

Figura 4: Crescimento do espaço de variáveis em virtude do aumento das dimensões.



Fonte: The Curse of Dimensionality (GLEESON, 2017)

Este trabalho apresenta um método de seleção automática de variáveis de entrada para modelos treinados através de aprendizado supervisionado.

1.5 Objetivos

O algoritmo desenvolvido visa a obtenção de um conjunto de variáveis que possa ser usado para o treinamento de modelos com alta capacidade de generalização.

É um método de seleção de variáveis *stepwise* e, portanto, realiza uma busca gananciosa (*greedy search*) em um conjunto de variáveis candidatas, selecionando, a cada iteração, a variável que apresentar a maior melhora na performance, segundo a métrica adotada.

Neste trabalho, faz-se o uso de um modelo de baixa complexidade (regressão linear regularizada), aliado ao erro de validação cruzada *leave-one-out*, para se determinar a variação da performance ao se incluir uma variável.

2 Fundamentação Teórica

Nesta seção são abordados conceitos fundamentais para a compreensão deste trabalho, assim como tendências atuais da literatura em relação ao tema estudado.

2.1 Regressão Linear

Os modelos de regressão linear constituem uma classe de modelos que utilizam funções lineares com parâmetros ajustáveis. O exemplo mais simples dessa classe é a função que realiza a combinação linear das entradas com os parâmetros ajustados para gerar uma predição (eq. 2.1).

$$\hat{y} = w_0 + w_1 x_1 + ... + w_p x_p = \sum_{i=0}^{p} w_i X = XW \tag{2.1}$$

Onde, X e W correspondem, respectivamente, ao vetor de entradas e de parâmetros, p corresponde ao número de dimensões da variável de entrada e $x_0 = 1$.

Modelos mais complexos e de maior aplicabilidade podem ser obtidos ao se considerar um conjunto fixo de transformações não-lineares ($\phi_n(X)$) em vez das variáveis originais, ou em conjunto com elas. Esses modelos são lineares em relação às suas variáveis independentes, porém são não-lineares em relação às variáveis de entrada (BISHOP, 2006).

2.1.1 Treinamento de modelos lineares

O treinamento de modelos lineares consiste em encontrar o vetor de parâmetros W que maximize a similaridade entre o modelo e o sistema modelado. Esse treinamento é normalmente realizado minimizando-se o somatório dos erros quadráticos (eq. 2.2) em função do vetor W.

$$\begin{aligned} E_{sq}(W) &= \frac{1}{2}\sum_{n=1}^{N}(y_n - \hat{y}_n)^2 \\ &= \frac{1}{2}(Y - \hat{Y})^T(Y - \hat{Y}) \\ &= \frac{1}{2}\left(Y^T Y - 2\hat{Y}^T Y + \hat{Y}^T\hat{Y}\right) \\ &= \frac{1}{2}\left(Y^T Y - 2W^T X^T Y + \sum_{n=1}^{N}(X_n W)^2\right) \end{aligned} \tag{2.2}$$

$$\begin{aligned}\frac{\partial E_{sq}(W)}{\partial W} &= \frac{1}{2}\left(0 - 2X^TY + 2\sum_{n=1}^{N}\left(X_n^TX_n\right)W\right) \\ &= -X^TY + X^TXW\end{aligned} \tag{2.3}$$

Igualando-se a eq. 2.3 a zero e isolando o vetor W, obtém-se a eq. 2.4, conhecida como a equação normal para o problema dos mínimos quadrados.

$$\begin{aligned}X^TXW &= X^TY \\ W &= (X^TX)^{-1}X^TY \\ W &= X^+Y\end{aligned} \tag{2.4}$$

O termo (X^TX) pode se aproximar de uma matriz singular se muitas das variáveis envolvidas forem linearmente dependentes, resultando assim em dificuldades para o cálculo numérico dos valores e possivelmente em um vetor de parâmetros de alta magnitude. Um termo de regularização pode ser adicionado na eq. 2.2 para minimizar esse problema, garantindo que a nova matriz não é singular.

A adição do termo de regularização tem ainda o efeito de limitar a complexidade efetiva do modelo, reduzindo o *overfitting* e possibilitando a utilização de conjuntos de dados menores (BISHOP, 2006).

Comumente utiliza-se a norma-L2 do vetor de parâmetros como termo de regularização, dando origem a *ridge regression*. A dedução da equação normal com termo de regularização é análoga à apresentada e resulta na eq. 2.5.

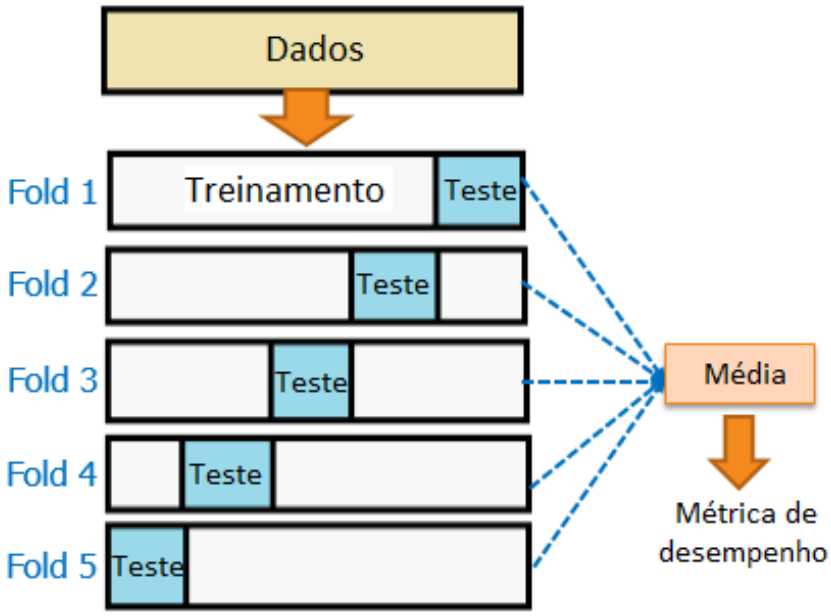
$$W = (\lambda I + X^TX)^{-1}X^TY = A^{-1}X^TY \tag{2.5}$$

2.2 Validação Cruzada

A validação cruzada (*cross-validation*) é um conjunto de metodologias de treinamento e validação que, de maneira simples e efetiva, permitem a estimativa do erro de generalização do modelo.

No método *k-fold* de validação cruzada, os dados de treinamento são divididos em k conjuntos aleatórios de tamanhos aproximadamente iguais. Realizada essa divisão, a cada iteração do treinamento, os parâmetros do modelo são determinados utilizando k-1 conjuntos e sua performance é avaliada no conjunto restante. O processo se repete até que todos os k conjuntos tenham sido utilizados para avaliação e a média das performances observadas fornecem uma estimativa da performance de generalização do modelo (CAWLEY; TALBOT, 2010). O algoritmo é ilustrado na Figura 5.

Figura 5: Validação cruzada: *5-fold*.



Fonte: Different types of Validations in Machine Learning (Cross Validation) (SRINIDHI, 2018)

Um caso especial do método de validação descrito é o *leave-one-out cross-validation* (LOOCV), onde k é igual ao total de amostras disponíveis (N), ou seja, o treinamento é realizado N vezes com N-1 amostras, e validado na amostra excluída. Tal abordagem apesar de apresentar, em geral, um custo computacional elevado, resulta em um modelo com menor variabilidade e viés para modelos de regressão linear (BURMAN, 1989), produzindo assim resultados com melhor performance e menos *overfitting*.

2.2.1 *Leave-one-out Cross-validation* e Regressão Linear

Especificamente para o caso do preditor linear, é possível determinar o resultado do LOOCV sem a necessidade de se calcular N preditores, tornando essa uma estratégia interessante para determinar a capacidade de generalização de um modelo linear. A seguir uma dedução adaptada de (SEBER; LEE, 2012, p. 268) é apresentada.

O erro de LOOCV corresponde à média dos erros quadráticos de cada estimador obtido excluindo-se uma amostra.

$$LOOCV = \frac{1}{N}\sum_{i=1}^{N} e_{[i]}^2 \tag{2.6}$$

$$e_{[i]} = y_i - \hat{y}_{[i]} = y_i - x_i W_{[i]} \tag{2.7}$$

Onde o i indexa a amostra excluída no treinamento, $\hat{y}_{[i]}$ corresponde à saída do preditor calculado sem a amostra (x_i, y_i) e $W_{[i]}$ os parâmetros desse preditor.

Pode-se definir o vetor de parâmetros $W_{[i]}$ conforme a eq. 2.8.

$$W_{[i]} = A_{[i]}^{-1} X_{[i]}^T Y_{[i]} \tag{2.8}$$

É possível perceber as seguintes igualdades:

$$\begin{aligned} A_{[i]} &= \lambda I + X_{[i]}^T X_{[i]} \\ &= \lambda I + X^T X - x_i^T x_i \\ &= A - x_i^T x_i \end{aligned} \tag{2.9}$$

$$X_{[i]}^T Y_{[i]} = X^T Y - x_i^T y_i \tag{2.10}$$

Aplicando a formula de Sherman–Morrison à eq. 2.9, temos:

$$\begin{aligned} h_i &= x_i A^{-1} x_i^T \\ A_{[i]}^{-1} &= A^{-1} + \frac{A^{-1} x_i x_i^T A^{-1}}{1 - h_i} \end{aligned} \tag{2.11}$$

Substituindo as eq. 2.10 e 2.11 na eq. 2.8.

$$\begin{aligned} W_{[i]} &= \left(A^{-1} + \frac{A^{-1} x_i^T x_i A^{-1}}{1 - h_i} \right) \left(X^T Y - x_i^T y_i \right) \\ &= W - A^{-1} x_i^T y_i + \frac{A^{-1} x_i^T x_i W}{1 - h_i} - \frac{A^{-1} x_i^T x_i A^{-1} x_i^T y_i}{1 - h_i} \\ &= W - \frac{A^{-1} x_i^T}{1 - h_i} \left(y_i (1 - h_i) - x_i W + h_i y_i \right) \\ &= W - \frac{A^{-1} x_i^T}{1 - h_i} \left(y_i - \hat{y}_i \right) \end{aligned} \tag{2.12}$$

Substituindo a eq. 2.12 na eq. 2.7.

$$\begin{aligned} e_{[i]} &= y_i - x_i W_{[i]} \\ &= y_i - x_i \left(W - \frac{A^{-1} x_i^T}{1 - h_i} \left(y_i - \hat{y}_i \right) \right) \\ &= y_i - \hat{y}_i + \frac{h_i}{1 - h_i} \left(y_i - \hat{y}_i \right) \\ &= \frac{y_i - \hat{y}_i}{1 - h_i} \end{aligned} \tag{2.13}$$

Seja P a matriz de projeção (BASILEVSKY, 2005, p. 303) do preditor (também conhecida como matriz chapéu (SEBER; LEE, 2012, p. 266)) e a matriz de aniquilação M (HAYASHI, 2001, p. 18) definidas conforme a eq. 2.14

$$\begin{aligned} \hat{Y} = PY &\implies P = XA^{-1}X^T \\ MY = Y - \hat{Y} &\implies M = I - P = I - XA^{-1}X^T \end{aligned} \tag{2.14}$$

Para cada índice i, o numerador da eq. 2.13 corresponde a linha de mesmo índice da matriz MY. Além disso o denominador da equação corresponde ao elemento da diagonal da matriz M de mesmo índice. A partir dessas observações, pode-se agrupar todos os valores de $e_{[i]}$ na forma da matriz E_{LOOCV}, conforme a equação 2.15.

$$E_{LOOCV} = diag(M)^{-1}MY \tag{2.15}$$

Assim, a equação 2.15, juntamente com o conhecimento que M é uma matriz simétrica (FREEDMAN, 2009), permite reescrever a eq. 2.6 na forma matricial apresentada na equação 2.16.

$$\begin{aligned} LOOCV &= \frac{1}{N}\sum_{i=1}^{N} e_{[i]}^2 \\ &= \frac{1}{N}E_{LOOCV}^2 \\ &= \frac{1}{N}E_{LOOCV}^T E_{LOOCV} \\ &= \frac{1}{N}(MY)^T\ diag(M)^{-2}MY \end{aligned} \tag{2.16}$$

2.3 Aprendizado Incremental

A atualização dos parâmetros do modelo de regressão linear para uma nova amostra pode ser realizada de maneira incremental, sem a necessidade de se calcular um novo modelo (ROSASCO, 2015). Neste trabalho, porém, é de interesse calcular a atualização incremental ao se adicionar uma nova variável. Em especial, é de interesse a determinação de uma forma fechada para a matriz de aniquilação, usada no cálculo do erro de validação cruzada.

A matrizes de interesse para o problema de dimensão $p+1$ tem a forma descrita na eq. 2.17

$$\begin{aligned} X_{p+1} &= [\mathbf{X_p}\ x_{p+1}] \\ Y_{p+1} &= [\mathbf{Y_p}\ y_{p+1}] \\ A_{p+1} &= (X_{p+1}^T X_{p+1} + \lambda I) = \begin{bmatrix} \mathbf{A_p} & \mathbf{X_p}^T x_{p+1} \\ x_{p+1}^T \mathbf{X_p} & x_{p+1}^T x_{p+1} + \lambda \end{bmatrix} \\ M_{p+1} &= I - X_{p+1}A_{p+1}^{-1}X_{p+1}^T \end{aligned} \tag{2.17}$$

É necessário então determinar a matriz A_{p+1}^{-1} em função das matrizes já conhecidas. Para tal, aplica-se a identidade da inversa de uma matriz em blocos, conforme a eq. 2.18,

na matriz A_{p+1}.

$$\begin{aligned} \Delta &= (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}) \\ \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} &= \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\Delta^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\Delta^{-1} \\ -\Delta^{-1}\mathbf{C}\mathbf{A}^{-1} & \Delta^{-1} \end{bmatrix} \end{aligned} \tag{2.18}$$

Através da eq. 2.17, obtém-se os termos A, B, C, D e Δ na eq. 2.19.

$$\begin{aligned} A &= A_p \\ B &= X_p^T x_{p+1} \\ C &= x_{p+1}^T X_p = B^T \\ D &= \lambda + x_{p+1}^T x_{p+1} \\ \Delta &= \lambda + x_{p+1}^T x_{p+1} - x_{p+1}^T X_p A_p^{-1} X_p^T x_{p+1} \end{aligned} \tag{2.19}$$

O termo Δ pode ser simplificado para a forma da eq. 2.20.

$$\begin{aligned} \Delta &= \lambda + x_{p+1}^T x_{p+1} - x_{p+1}^T X_p A_p^{-1} X_p^T x_{p+1} \\ &= \lambda + x_{p+1}^T (I x_{p+1} - X_p A_p^{-1} X_p^T x_{p+1}) \\ &= \lambda + x_{p+1}^T (I - X_p A_p^{-1} X_p^T) x_{p+1} \\ &= \lambda + x_{p+1}^T M_p x_{p+1} \end{aligned} \tag{2.20}$$

Substituindo, obtém-se a eq. 2.21

$$\begin{aligned} A_{p+1}^{-1} &= \begin{bmatrix} \mathbf{A}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B} \\ -\mathbf{C}\mathbf{A}^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A_p}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} A_p^{-1}(X_p^T x_{p+1})(X_p^T x_{p+1})^T A_p^{-1} & -A_p^{-1}(X_p^T x_{p+1}) \\ -(X_p^T x_{p+1})^T A_p^{-1} & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A_p}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\Delta} \begin{bmatrix} A_p^{-1}(X_p^T x_{p+1}) \\ -1 \end{bmatrix} \begin{bmatrix} x_{p+1}^T X_p A_p^{-1} & -1 \end{bmatrix} \end{aligned} \tag{2.21}$$

Por fim, substituindo a eq. 2.21 no termo M_{p+1} da eq. 2.17, obtém-se a expressão para a nova matriz de aniquilação.

$$
\begin{aligned}
M_{p+1} &= I - X_{p+1}A_{p+1}^{-1}X_{p+1}^T \\
&= I - X_{p+1}\left(\begin{bmatrix} \mathbf{A_p}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\Delta}\begin{bmatrix} A_p^{-1}(X_p^T x_{p+1}) \\ -1 \end{bmatrix}\begin{bmatrix} x_{p+1}^T X_p A_p^{-1} & -1 \end{bmatrix}\right) X_{p+1}^T \\
&= \left(I - \begin{bmatrix} \mathbf{X_p} & x_{p+1} \end{bmatrix}\begin{bmatrix} \mathbf{A_p}^{-1} & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} \mathbf{X_p}^T \\ x_{p+1}^T \end{bmatrix}\right) - \\
&\qquad \frac{1}{\Delta}\left(\begin{bmatrix} \mathbf{X_p} & x_{p+1} \end{bmatrix}\begin{bmatrix} A_p^{-1}(X_p^T x_{p+1}) \\ -1 \end{bmatrix}\begin{bmatrix} x_{p+1}^T X_p A_p^{-1} & -1 \end{bmatrix}\begin{bmatrix} \mathbf{X_p}^T \\ x_{p+1}^T \end{bmatrix}\right) \\
&= M_p - \frac{1}{\Delta}\left(\mathbf{X_p}A_p^{-1}X_p^T x_{p+1} - x_{p+1}\right)\left(x_{p+1}^T X_p A_p^{-1}\mathbf{X_p}^T - x_{p+1}^T\right) \\
&= M_p - \frac{1}{\Delta}\left(\mathbf{X_p}A_p^{-1}X_p^T - I\right)x_{p+1}x_{p+1}^T\left(X_p A_p^{-1}\mathbf{X_p}^T - I\right) \\
&= M_p - \frac{M_p x_{p+1}x_{p+1}^T M_p}{\Delta} \\
M_{p+1} &= M_p - \frac{M_p x_{p+1}x_{p+1}^T M_p}{\lambda + x_{p+1}^T M_p x_{p+1}}
\end{aligned}
\tag{2.22}
$$

M é uma matriz simétrica, portanto a eq. 2.22 pode ser escrita de maneira simplificada como a eq. 2.23.

$$
\begin{aligned}
M_{p+1} &= M_p - \frac{M_p x_{p+1}x_{p+1}^T M_p}{\lambda + x_{p+1}^T M_p x_{p+1}} \\
&= M_p - \frac{M_p x_{p+1}(M_p x_{p+1})^T}{\lambda + (M_p x_{p+1})^T x_{p+1}}
\end{aligned}
\tag{2.23}
$$

3 Desenvolvimento e Metodologia de Testes

3.1 Algoritmo

3.1.1 Seleção de Variáveis *Stepwise*

O algoritmo proposto nesse trabalho é baseado no método de seleção de variáveis *stepwise* (*Stepwise Selection*), comumente utilizado em conjunto com modelos de regressão linear. É um método de *greedy search*, onde, a cada iteração, a variável que apresentar o melhor ganho de performance é adicionada ao conjunto de entradas.

O modelo é construído incrementalmente até que não haja mais melhora de performance ao acrescentar alguma das variáveis restantes ou não haja mais variáveis para serem consideradas. O método é descrito em pseudocódigo no Algoritmo 1.

Algoritmo 1: *Forward Stepwise Selection* (FSS)

```
Entrada: variáveis: lista contendo as variáveis disponíveis;
Entrada: saídas: lista contendo as saídas do modelo;
Saída: selecionadas: lista de variáveis relevantes.
selecionadas ← { }
melhorErro ← ∞
repita
    melhorVariável ← NULL
    para cada elemento var em variáveis faça
        entradas ← (var ∪ selecionadas)
        model ← ajuste(entradas, saídas)
        erro ← avalia(model, entradas, saídas)
        se erro < melhorErro então
            melhorErro ← erro
            melhorVariável ← var
    selecionadas ← (melhorVariável ∪ selecionadas)
    variáveis ← (variáveis \ {melhorVariável})
até variáveis == { } ou (critério de parada);
```

No método da seleção do melhor subconjunto, que é um algoritmo de busca exaustiva, todas as combinações possíveis das variáveis de entrada são testadas. No algoritmo *Stepwise* porém, há uma redução expressiva na quantidade de modelos ajustados. Na primeira iteração, o modelo nulo deve ser ajustado, em seguida, todas as p variáveis devem ser avaliadas e, a cada iteração posterior, uma delas é retirada.

Porém, ao contrário do método da seleção do melhor subconjunto, não há garantia que o subconjunto determinado é a combinação ótima das variáveis (JAMES et al., 2017, p. 208).

Um fator determinante para a eficácia do algoritmo é a escolha da métrica pela qual os modelos serão avaliados. A utilização de uma métrica que considere simplesmente os dados ajustados não é adequada, uma vez que, nessa situação, o incremento de uma variável no modelo sempre acarretará em uma melhora no erro de treinamento, porém não necessariamente na capacidade de generalização do modelo.

Algumas técnicas tentam determinar a capacidade de generalização através de informações obtidas com os dados de treinamento, tais como o critério de Informação de Akaike (AIC) ou critério de informação Bayesiano (BIC), porém elas se baseiam no comportamento assintótico, isto é, quando a quantidade de amostras é bastante elevada.

Uma alternativa a essas métricas é a utilização de validação cruzada, onde o modelo selecionado é aquele que apresenta a melhor performance no conjunto de testes. Dessa maneira, obtém-se diretamente uma estimativa do erro de generalização, além de se assumir menos condições em relação ao modelo e aos dados utilizados.

O grande empecilho à implementação de validação cruzada é seu custo computacional que, quando associado ao custo do método de seleção de variáveis *stepwise*, pode tornar proibitiva sua implementação.

Em especial, a utilização desse método associada ao *leave-one-out cross-validation* resultaria no ajuste de um número significativamente elevado de modelos, tornando o algoritmo computacionalmente inviável.

3.1.2 Método Incremental

Para minimizar o problema do custo computacional, o método implementado utiliza o resultado apresentado na eq. 2.16 para calcular o erro de validação cruzada sem a necessidade de calcular $N - 1$ modelos de regressão linear. Além disso, o cálculo da matriz de aniquilação é realizado de maneira incremental, conforme deduzido na eq. 2.22.

O Algoritmo 1 é então modificado no Algoritmo 2 para realizar o cálculo incremental, utilizando LOOCV como métrica para seleção das variáveis e atualização incremental do modelo.

O código em R que implementa o algoritmo proposto se encontra no Apêndice A.1 desse trabalho.

Algoritmo 2: *Forward Stepwise Incremental Selection*

Entrada: *variáveis*: lista contendo as variáveis disponíveis;
Entrada: *saídas*: lista contendo as saídas do modelo;
Saída: *selecionadas*: lista de variáveis relevantes.
$selecionadas \leftarrow \{\ \}$
$melhorErro \leftarrow \infty$
$M \leftarrow I_N$
repita
  $melhorVariável \leftarrow NULL$
  para cada *elemento var em variáveis* **faça**
    $M' \leftarrow ajusteIncremental(M, var)$
    $erro \leftarrow erroLOOCV(M, saídas)$
    se $erro < melhorErro$ **então**
      $melhorErro \leftarrow erro$
      $melhorVariável \leftarrow var$
      $melhorM \leftarrow M'$
  $M \leftarrow melhorM$
  $selecionadas \leftarrow (melhorVariável \cup selecionadas)$
  $variáveis \leftarrow (variáveis \setminus \{melhor_{variável}\})$
até *variáveis* $== \{\ \}$ *ou (critério de parada)*;

3.1.3 Complexidade algorítmica

A implementação não incremental do método *stepwise* resulta no ajuste de uma curva para cada conjunto de variáveis considerado. Como apenas uma variável é acrescentada a cada etapa, o total de modelos ajustados pode então ser derivado como na eq. 3.1.

$$1 + \sum_{k=0}^{p-1}(p-k) = 1 + p(p+1) = \frac{1}{2}(p^2 + p + 2) \tag{3.1}$$

Quando associado ao cálculo do erro de validação cruzada *leave-one-out*, para cada conjunto de varáveis considerado, é necessário o cálculo de $(N-1)$ modelos, cada um excluindo uma amostra do treinamento. Dessa forma a equação eq. 3.1 deve ser multiplicada por esse fator.

Analisando a eq. 2.5, é possível assumir a complexidade de uma implementação simples do ajuste de uma regressão linear como $\mathcal{O}\left(Np^2 + p^3\right)$. Implementações utilizando algoritmos otimizados de multiplicação e inversão de matrizes podem reduzir essa complexidade. A custo de uma predição pode ser facilmente percebido como $O(p)$.

Sendo assim, a complexidade da implementação não incremental do método *stepwise*

pode ser derivada conforme a eq. 3.2.

$$\begin{aligned} O(Stepwise_{LOOCV}) &= \frac{1}{2}(p^2+p+2)(N-1)\left(\mathcal{O}\left(Np^2+p^3\right)+\mathcal{O}\left(p\right)\right) \\ &= \mathcal{O}\left(p^2\right)\ \mathcal{O}\left(N\right)\ \mathcal{O}\left(Np^2+p^3\right) \\ &= \mathcal{O}\left(N^2p^4+Np^5\right) \end{aligned} \tag{3.2}$$

O método proposto neste trabalho não realiza o ajuste direto do modelo, mas sim o cálculo de maneira incremental da matriz de aniquilação. Dessa forma, além de não ser necessário o cálculo da inversão da matriz A na eq. 2.5, o cálculo do erro de validação cruzada *leave-one-out* pode ser calculado diretamente, sem a necessidade de $N-1$ passos. Dessa maneira o custo computacional é reduzido para o exposto na eq. 3.3.

$$\begin{aligned} O(Stepwise^{INC}_{LOOCV}) &= \frac{1}{2}(p^2+p+2)\left(\mathcal{O}\left(3N^2\right)\right) \\ &= \mathcal{O}\left(p^2\right)\ \mathcal{O}\left(N^2\right) \\ &= \mathcal{O}\left(N^2p^2\right) \end{aligned} \tag{3.3}$$

3.2 Testes

3.2.1 Conjuntos de dados

Para teste do algoritmo desenvolvido, utilizou-se quatro bancos de dados conhecidos na literatura, especificados na tabela 1.

Tabela 1: Conjuntos de dados utilizados para teste.

Nome	Variáveis	Amostras
Communities and Crime (REDMOND, 2011)	101	2215
Forest Fires (CORTEZ; MORAIS, 2008)	12	517
USA Housing Dataset (GANESH, 2018)	80	1460
Wiscoin Breast Cancer (TORGO, 1996)	32	194

3.2.1.1 Pré-processamento

De maneira geral, em um conjunto de amostras, as variáveis apresentam diferentes unidades, magnitudes e escalas. Tal variação dificulta a comparação de valores e pode prejudicar o treinamento de modelos. Por isso é comum aplicar uma transformação nos dados, de maneira a torná-los comparáveis entre si (*feature scaling*), além de facilitar a detecção de *outliers*.

Embora o algoritmo utilizado se baseie em regressões lineares que, em sua forma mais simples, são robustas em relação a magnitude das variáveis, a adição do termo de

regressão penaliza o crescimento do vetor de parâmetros, tornando o modelo sensível à diferença de magnitudes entre variáveis.

Neste trabalho, todas as variáveis foram normalizadas, de maneira a se obter entradas e saídas que apresentam média nula e desvio padrão unitário. Tal transformação é conhecida como *z-score* (eq. 3.4).

$$X_p = \frac{X_p - \bar{X_p}}{SD(X_p)} \tag{3.4}$$

Além disso, uma variável de entrada com valor unitário foi adicionada posteriormente a cada conjunto de dados, permitindo a consideração do termo livre (w_0) nos cálculos de maneira transparente.

3.2.2 Metodologia de Teste

Com o intuito de identificar um subconjunto ótimo de variáveis para cada conjunto de dados, o algoritmo desenvolvido foi empregado e o erro de validação cruzada, conforme a eq. 2.16, foi registrado ao final de cada iteração. Dessa maneira, é possível analisar o efeito da inclusão de uma nova variável na capacidade de generalização do modelo e avaliar o quão benéfico é o aumento de uma nova dimensão.

Em seguida, para cada conjunto de dados, dois modelos são treinados com parâmetros idênticos, variando-se o conjunto de variáveis. Dessa maneira, é possível estudar quais os efeitos da utilização do subconjunto proposto pelo algoritmo em relação ao conjunto completo de variáveis.

Optou-se pela utilização de um modelo *Perceptron* (MLP) de uma camada oculta (três camadas no total: entradas, camada oculta, saída), devido a disponibilidade de implementações disponíveis tanto para o modelo, quanto para o seu treinamento e validação.

Utilizou-se a biblioteca Keras (ALLAIRE; CHOLLET, 2018) com a interface para a linguagem R e a biblioteca Tensorflow (GOOGLE, 2015) como *backend*. O código fonte que implementa a rede neural desenvolvida se encontra no Apêndice A.2.

4 Resultados e Discussão

4.1 Descrição

A cada iteração "i", um subconjunto candidato de i variáveis é determinado. O melhor subconjunto é aquele que minimiza o erro LOOCV. É importante reiterar que o algoritmo utiliza uma estratégia de *greedy search* para buscar os subconjuntos e, portanto, a otimalidade global desses subconjuntos não é garantida.

O algoritmo foi aplicado a cada um dos bancos de dados indicados na seção anterior, de maneira a determinar subconjuntos otimizados de variáveis para uso em um modelo de regressão. Os gráficos obtidos demonstram, a cada iteração, a variável ainda não utilizada que, ao ser acrescentada no modelo, traz a maior redução do erro de validação cruzada (ou o menor aumento).

Para verificar o efeito da utilização do grupo de entradas proposto, duas redes neurais foram treinadas em cada conjunto de dados. Uma rede utilizando a totalidade das variáveis disponíveis e outra rede utilizando o subconjunto sugerido pelo algoritmo. Os parâmetros de treinamento, tais como taxa de aprendizado e tamanho da camada oculta, foram determinados e fixados por conjunto de dados, permitindo a comparação do efeito isolado da variação do conjunto de entradas.

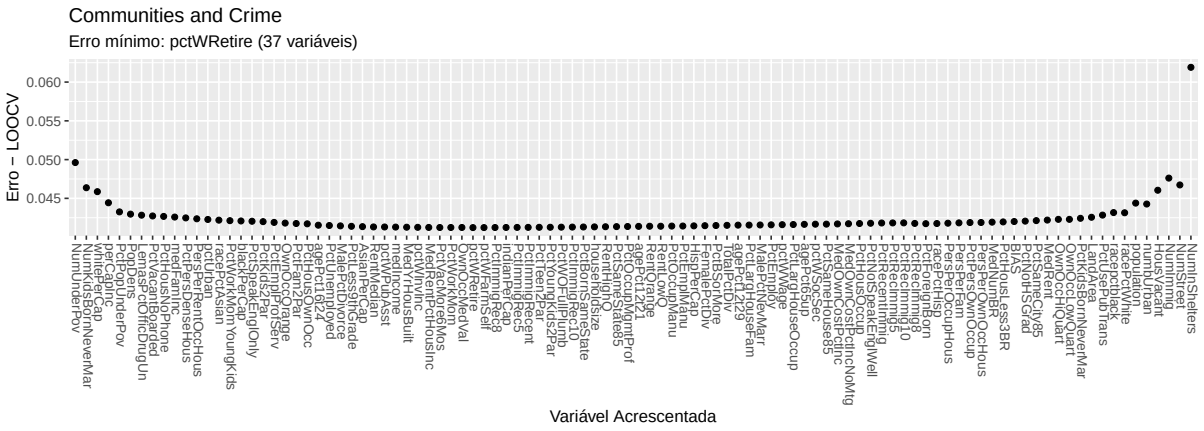
4.2 Seleção de Variáveis

4.2.1 *Communities and Crime*

No banco de dados *Communities and Crime*, o erro de validação cruzada LOOCV mínimo foi observado em um conjunto com apenas 37 das 102 variáveis. A adição de variáveis posteriormente teve pouca influência no erro de validação cruzada, com uma tendência à degradação da performance na inclusão das últimas variáveis.

O resultado para cada iteração pode ser verificado na Figura 6.

Figura 6: Seleção de Variáveis *Stepwise: Communities and Crime*.



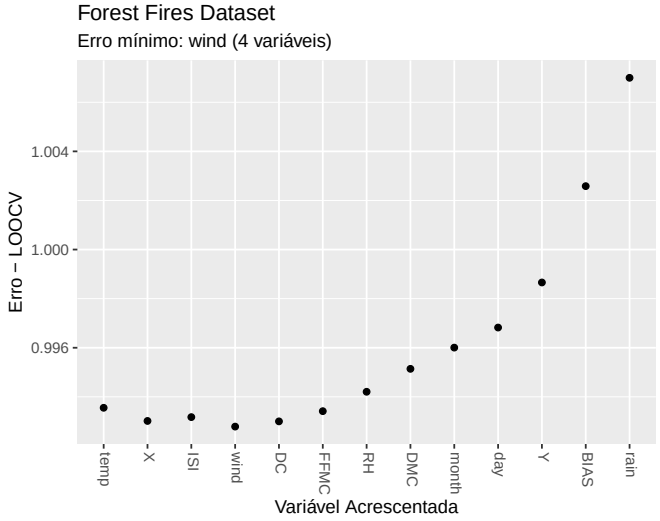
Fonte: Própria.

4.2.2 *Forest Fires*

No banco de dados *Forest Fires*, o erro de validação cruzada LOOCV mínimo foi observado em um conjunto com apenas 4 das 12 variáveis. A adição de variáveis posteriormente resultou em perda significativa de performance.

O resultado para cada iteração pode ser verificado na Figura 7.

Figura 7: Seleção de Variáveis *Stepwise: Forest Fires*.



Fonte: Própria.

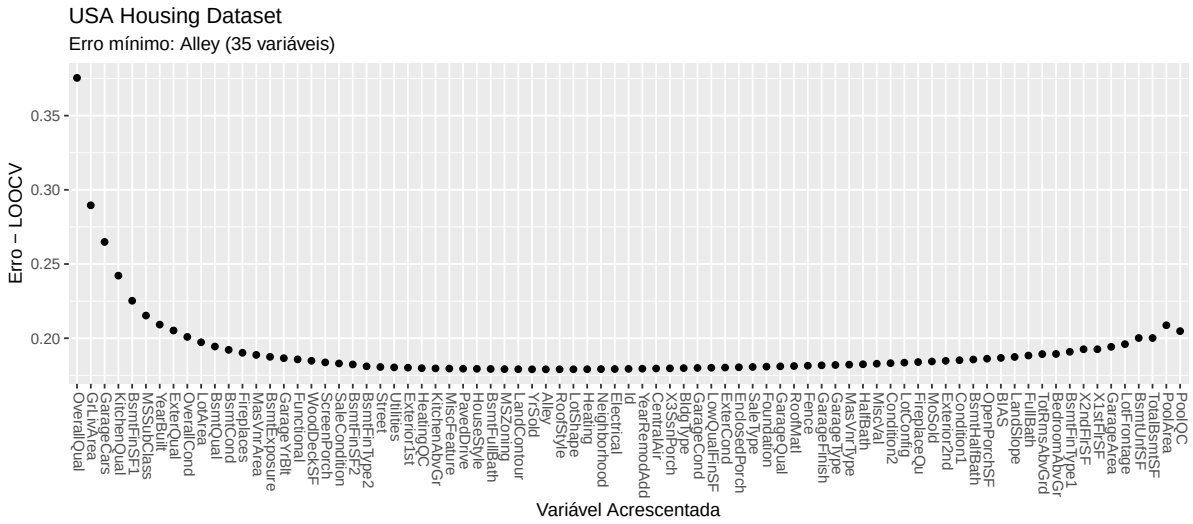
4.2.3 *USA Housing*

No banco de dados *USA Housing*, o erro de validação cruzada LOOCV mínimo foi observado em um conjunto com apenas 35 das 80 variáveis. A adição posterior de variáveis

teve pouca influência no erro de validação cruzada.

O resultado para cada iteração pode ser verificado na Figura 8.

Figura 8: Seleção de Variáveis *Stepwise: USA Housing*.



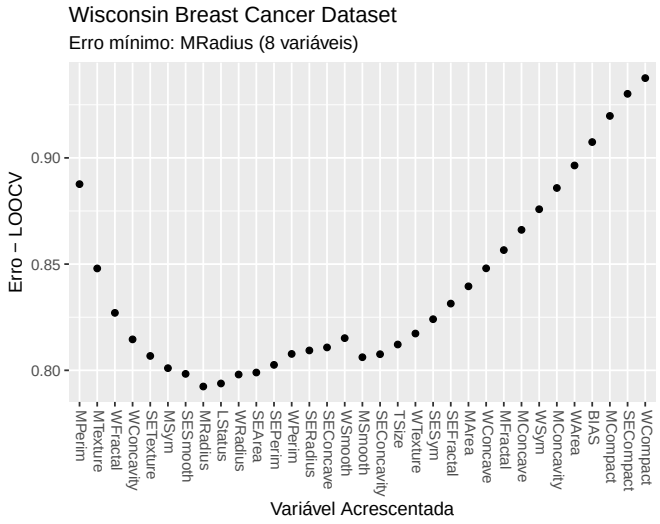
Fonte: Própria.

4.2.4 *Wisconsin Breast Cancer*

No banco de dados *Wisconsin Breast Cancer*, o erro de validação cruzada LOOCV mínimo foi observado em um conjunto com apenas 8 das 32 variáveis, e foi identificada também uma perda significativa de performance quando utilizado um subconjunto candidato de mais de 16 variáveis.

O resultado para cada iteração pode ser verificado na Figura 9.

Figura 9: Seleção de Variáveis *Stepwise: Wisconsin Breast Cancer Dataset.*



Fonte: Própria.

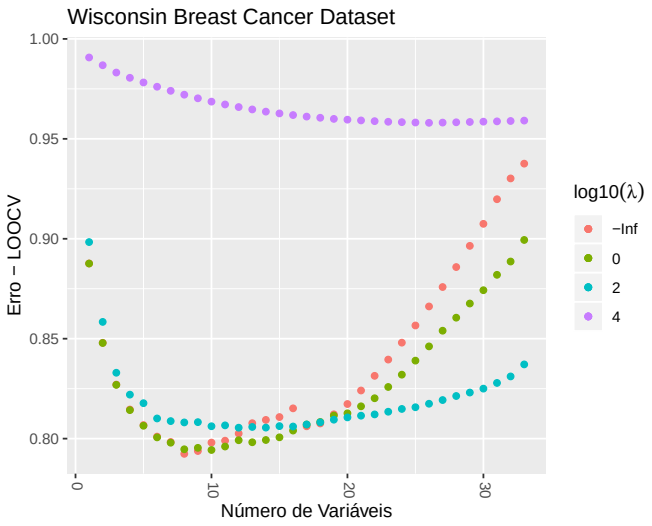
4.2.5 Discussão

Observou-se o decréscimo do erro com o aumento do número de variáveis selecionadas até um ponto de mínimo, a partir do qual a tendência se inverte e o erro passa a aumentar. Tal dinâmica é esperada, uma vez que o aumento do número de parâmetros, na regressão linear, resulta no aumento da complexidade do modelo (HASTIE; TIBSHIRANI; FRIEDMAN, 2017, p. 224), o que por sua vez pode reduzir a capacidade de generalização do modelo.

De fato, ao se reduzir a complexidade do modelo através do aumento do parâmetro de regularização (λ), observa-se a diminuição da tendência ao overfitting, conforme demonstra a Figura 10.

É interessante notar que valores baixos de λ tiveram pouca ou nenhuma influência nos primeiros subconjuntos ótimos selecionados pelo algoritmo, alterando apenas a seleção na região com tendência ao overfitting. Assim, a utilização de um parâmetro de regularização não nulo de pequena magnitude tem pouca influência no conjunto selecionado, enquanto melhora a estabilidade numérica do método, evitando operações com matrizes singulares.

Figura 10: Efeito do parâmetro de regularização (λ): *Wisconsin Breast Cancer Dataset*.



Fonte: Própria.

4.3 Modelos

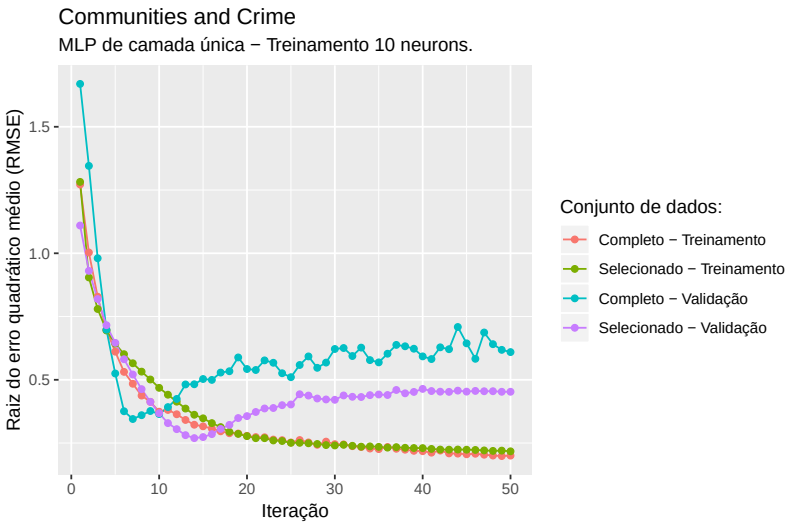
4.3.1 *Communities and Crime*

Tabela 2: Parâmetros de Treinamento: *Communities and Crime*.

	Conjunto de dados
Tamanho (neurônios)	10
Taxa de Aprendizado	1e-4
Iterações	50
Função de Ativação	ReLU
Conjunto de Validação	30%

Observou-se que ambos os modelos apresentaram o mesmo erro de treinamento, porém, a rede neural treinada com o conjunto menor de variáveis de entrada apresentou melhor desempenho no conjunto de validação, indicando uma possível maior capacidade de generalização.

Figura 11: Treinamento do Modelo: *Communities and Crime*.



Fonte: Própria.

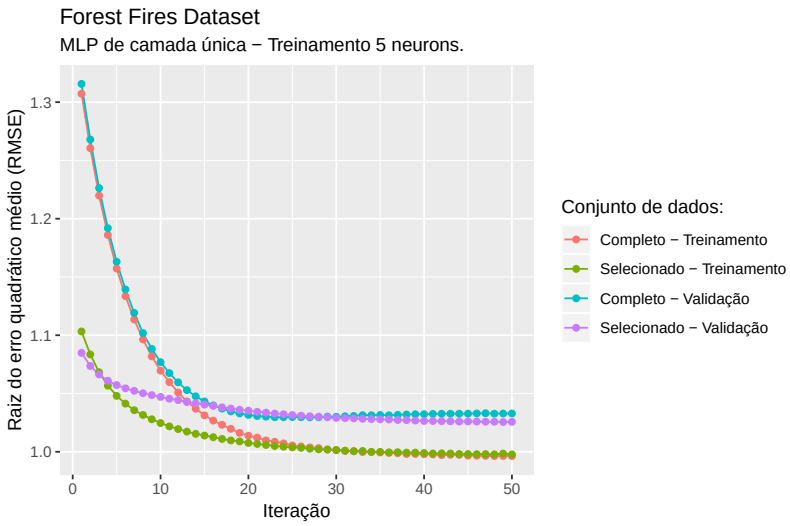
4.3.2 *Forest Fires*

Tabela 3: Parâmetros de Treinamento: *Forest Fires*.

	Conjunto de dados
Tamanho (neurônios)	5
Taxa de Aprendizado	1e-4
Iterações	50
Função de Ativação	Tanh
Conjunto de Validação	30%

Os modelos treinados apresentaram desempenho similar, sendo que a versão que utiliza o conjunto restrito de entradas, apresentou uma convergência inicial mais rápida.

Figura 12: Treinamento do Modelo: *Forest Fires*.



Fonte: Própria.

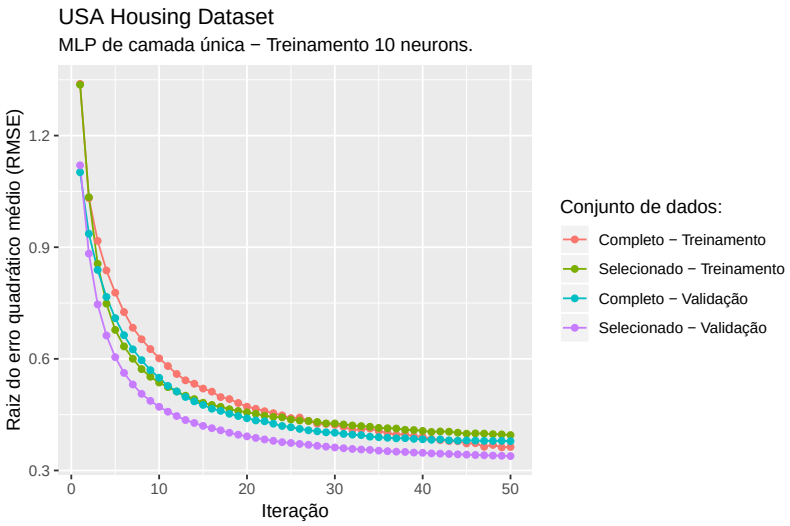
4.3.3 *USA Housing*

Tabela 4: Parâmetros de Treinamento: *USA Housing*.

	Conjunto de dados
Tamanho (neurônios)	10
Taxa de Aprendizado	1e-4
Iterações	50
Função de Ativação	ReLU
Conjunto de Validação	30%

No conjunto de dados *USA Housing*, as redes neurais apresentaram performance similar para os casos de validação e treinamento.

Figura 13: Treinamento do Modelo: *USA Housing*.



Fonte: Própria.

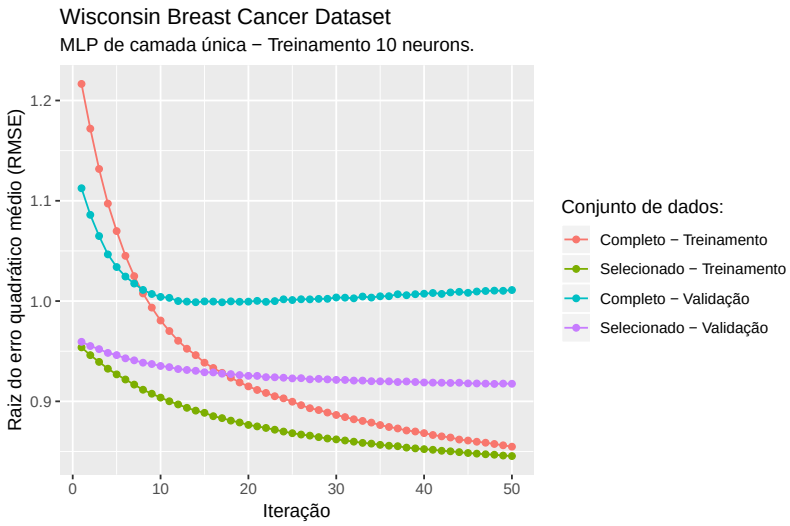
4.3.4 *Wisconsin Breast Cancer*

Tabela 5: Parâmetros de Treinamento: *Wisconsin Breast Cancer*.

	Conjunto de dados
Tamanho (neurônios)	10
Taxa de Aprendizado	1e-4
Iterações	50
Função de Ativação	Tanh
Conjunto de Validação	30%

Por fim, no conjunto de dados *Wisconsin Breast Cancer*, observou-se uma diferença expressiva entre o erro de validação e treinamento para o modelo que utiliza todas as entradas, indicando a ocorrência de *overfitting*. Tal diferença foi reduzida, ao se utilizar o conjunto de variáveis selecionadas pelo algoritmo.

Figura 14: Treinamento do Modelo: *Wisconsin Breast Cancer*.



Fonte: Própria.

4.3.5 Discussão

Em todos os dados avaliados, a utilização do conjunto de variáveis sugerido pelo algoritmo resultou em um modelo com performance similar ou superior ao que utiliza o conjunto completo.

Nota-se através da Figura 11 que, embora ambos modelos tenham demonstrado um padrão que sugere a presença de overfitting, o modelo com menos variáveis de entrada o exibiu em uma etapa posterior do treinamento, além de apresentar a tendência em menor intensidade. Tal fato pode ser constatado observando que, para o mesmo erro de treinamento, seu erro de validação foi menor que sua contraparte com o conjunto completo de entradas. Na Figura 12, também é possível perceber essa tendência se iniciando nas últimas iterações do treinamento.

Na Figura 13, não houve variações significativas ao se utilizar o modelo com número de entradas reduzido. Porém, na Figura 14, observa-se que, embora os modelos tenham apresentado desempenho similar no erro de treinamento, houve uma melhora significava no erro de validação, sugerindo que a utilização do subconjunto proposto pelo algoritmo traz melhorias na capacidade de generalização do modelo para esse conjunto de dados.

Por fim é interessante ressaltar que os impactos da utilização de menos variáveis de entrada não se limitam o desempenho final do modelo. A redução da dimensão de entrada de um sistema tende a produzir modelos menores (com menos parâmetros), e consequentemente mais rápidos e simples de serem treinados.

5 Conclusão

Neste trabalho foi apresentado um método de seleção de variáveis *stepwise* utilizando o erro de validação cruzada *leave-one-out* (LOOCV) em regressores lineares como métrica de performance. O algoritmo proposto faz uso de identidades algébricas conhecidas na literatura para determinar, de maneira incremental, o erro de LOOCV ao se acrescentar uma variável. Dessa maneira, elimina-se a necessidade do ajuste de novos modelos lineares a cada subconjunto de variáveis avaliado.

Os resultados obtidos sugerem a manutenção ou melhora da performance nos bancos de dados considerados ao se utilizar o subconjunto de entradas proposto pelo algoritmo. Além disso, tais subconjuntos são, em todos os casos avaliados, significativamente menores que o total disponível, reduzindo assim o custo computacional e acelerando o treinamento dos modelos.

Diversas técnicas para seleção de variáveis *stepwise* são conhecidas na literatura, porém essas, em geral, utilizam métricas que se baseiam no comportamento assintótico do conjunto de variáveis e presumem diversas restrições em relação ao conjunto de dados e ao modelo a ser desenvolvido. Utilizar o erro de validação cruzada como métrica permite o relaxamento dessas restrições.

A necessidade do armazenamento de uma matriz quadrada da dimensão do tamanho do conjunto amostral é um fator limitante no método proposto. Uma possível solução para tal limitação é a utilização de um número menor de amostras para a seleção de variáveis. Porém a determinação desse número exige ainda um estudo de complexidade amostral (*sample complexity*), aplicando-se teorias tais como complexidade de Rademacher para determinação dos limiares do erro de generalização. Porém tal estudo foge ao escopo deste trabalho.

Por fim, a utilização de regressões lineares é também um fator limitante, uma vez que esses modelos podem falhar em capturar algumas relações não-lineares. Uma possível maneira de minimizar esse problema é utilizar o truque de kernel (*kernel trick*)(THEODORIDIS; KOUTROUMBAS, 2009), isto é, a aplicar uma função não linear aos dados e utilizar esses dados como entradas no método proposto. Dessa maneira, o algoritmo continua sendo linear em relação às variáveis transformadas, porém se torna não linear em relação aos dados originais.

Referências

ABHISHEK, N. *Machine Learning: A Gentle Introduction. – Towards Data Science*. Towards Data Science, 2018. Disponível em: <https://towardsdatascience.com/machine-learning-a-gentle-introduction-17e96d8143fc>. Acesso em: 12/11/2018. Citado na página 16.

ALLAIRE, J. J.; CHOLLET, F. *R Interface to 'Keras'*. 2018. Disponível em: <https://keras.rstudio.com/>. Acesso em: 14/11/2018. Citado na página 33.

BASILEVSKY, A. *Applied matrix algebra in the statistical sciences*. [S.l.]: Dover Publications, 2005. Citado na página 24.

BISHOP, C. M. *Pattern recognition and machine learning*. [S.l.]: Springer, 2006. Citado 3 vezes nas páginas 18, 21 e 22.

BURMAN, P. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, v. 76, n. 3, 1989. Citado na página 23.

CAWLEY, G. C.; TALBOT, N. L. On over-fitting in model selection and subsequent selection bias in performance evaluation. *J. Mach. Learn. Res.*, JMLR.org, v. 11, p. 2079–2107, ago. 2010. ISSN 1532-4435. Disponível em: <http://dl.acm.org/citation.cfm?id=1756006.1859921>. Citado na página 22.

CORTEZ, P.; MORAIS, A. A data mining approach to predict forest fires using meteorological data. *New Trends in Artificial Intelligence, EPIA - Portuguese Conference on Artificial Intelligence*, n. 2007, Feb 2008. Disponível em: <http://archive.ics.uci.edu/ml/datasets/Forest%2BFires>. Acesso em: 14/11/2018. Citado na página 32.

CRUELLS, E. B. i. *Bias and Variance*. 2017. Disponível em: <http://www.ebc.cat/2017/02/12/bias-and-variance/>. Acesso em: 12/11/2018. Citado na página 17.

FREEDMAN, D. A. *Statistical models: theory and practice*. 2. ed. [S.l.]: Cambridge University Press, 2009. Citado na página 25.

GANESH. *USA Housing dataset*. Kaggle, 2018. Disponível em: <https://www.kaggle.com/gpandi007/usa-housing-dataset>. Acesso em: 14/11/2018. Citado na página 32.

GLEESON, P. *The Curse of Dimensionality*. Medium, 2017. Disponível em: <https://medium.freecodecamp.org/the-curse-of-dimensionality-how-we-can-save-big-data-from-itself-d9fa0f872335>. Acesso em: 31/10/2018. Citado na página 19.

GOOGLE. *TensorFlow*. 2015. Disponível em: <https://www.tensorflow.org/>. Citado na página 33.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H. *The elements of statistical learning: data mining, inference, and prediction*. 2. ed. [S.l.]: Springer, 2017. Citado na página 38.

HAYASHI, F. *Econometrics*. Princeton: Princeton University Press, 2001. ISBN 9781400823833. Citado na página 24.

HAYKIN, S. S. *Neural networks and learning machines*. 3. ed. [S.l.]: Prentice Hall, 2009. Citado na página 16.

JAMES, G. et al. *An introduction to statistical learning with applications in R*. [S.l.]: Springer, 2017. Citado 2 vezes nas páginas 18 e 30.

MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-Hill, 1997. Citado na página 15.

REDMOND, M. *Communities and Crime Unnormalized Data Set*. UCI Machine Learning Repository, 2011. Disponível em: <http://archive.ics.uci.edu/ml/datasets/CommunitiesandCrimeUnnormalized>. Acesso em: 14/11/2018. Citado na página 32.

ROSASCO, P. *Machine Learning: a Regularization Approach: Chapter 7 - Online Learning*. [S.l.]: MIT-9.520 Lectures Notes, Manuscript, 2015. Citado na página 25.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence - A Modern Approach*. 3. ed. [S.l.]: Prentice Hall, 2010. Citado na página 15.

SEBER, G. A. F.; LEE, A. J. *Linear regression analysis*. 2. ed. [S.l.]: Wiley, 2012. Citado 2 vezes nas páginas 23 e 24.

SRINIDHI, S. *Different types of Validations in Machine Learning (Cross Validation)*. 2018. Disponível em: <https://blog.contactsunny.com/data-science/different-types-of-validations-in-machine-learning-cross-validation>. Acesso em: 31/10/2018. Citado na página 23.

THEODORIDIS, S.; KOUTROUMBAS, K. *Pattern recognition*. 4th. ed. [S.l.]: Elsevier/Acad. Press, 2009. Citado na página 45.

TORGO, L. *Wisconsin Breast Cancer*. Universidade do Porto, 1996. Disponível em: <http://www.dcc.fc.up.pt/~ltorgo/Regression/wbc.html>. Acesso em: 14/11/2018. Citado na página 32.

TRUNK, G. V. A problem of dimensionality: A simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1, n. 3, p. 306–307, July 1979. ISSN 0162-8828. Citado na página 18.

APÊNDICE A – Códigos Fonte

A.1 *Stepwise Incremental Forward Selection*

```
library(MASS) #para calculo do nr. efetivo de parâmetros

#Calcula nova matriz de aniquilação ao incrementar variável
incrementM =  function(Mprev, Fnew,lambda){
  aux = Mprev %*% Fnew
  scale = as.numeric(lambda + crossprod(Fnew,aux))
  adjust = (tcrossprod(aux,aux))/scale
  Mnew = Mprev - adjust
  return(Mnew)
}

#Calcula erro de LOOCV para uma matriz de projeção M e vetor de saída Y
looMSE = function(M,y){
  N = nrow(M)
  aux = diag(N)*(diag(M)^-2) # mais rápido que diag(diag(M)^-2))
  my = M %*% y
  mse = 1/N * crossprod(my,aux) %*% my #M^t == M
  return(as.numeric(mse))
}

#Função principal
#Detemrina o melhor subconjunto.
#Testa todas as variáveis ou até subconjuntos de lim elementos.
#Pode assumir uma matriz de aniquilação M inicial.
#Aceita o parâmetro de regularização l2 lambda.
rankFeatures = function(X,Y, lim = -1, M = NULL,lambda = 0){

  #Conversão das variáveis
  ##################################
  if(!is.matrix(X))
    X = as.matrix(X)
  if(!is.vector(Y))
```

```
    Y = as.vector(as.matrix(Y))
  if(lim == -1 || lim > ncol(X))
    lim = ncol(X)
  ###################################

  N = nrow(X)
  if(!is.matrix(M))
    M = diag(nrow = N, ncol=N)
  selected_features = c()
  candidates = colnames(X)
  i = 0

  while(length(candidates) && i<lim)
  {

    best = list(name="",error=Inf,M=NULL)

    for(feat in candidates){
      Mcand = incrementM(M,X[,feat],lambda)
      error = looMSE(Mcand,Y)
      improvement = error<best$error
      if(is.na(improvement)){
        print("Degeneration")
        return(-1)
      }else if(improvement){
        best$name = feat
        best$error = error
        best$M = Mcand
      }
    }
    i = i+1
    selected_features[best$name] = best$error
    M = best$M
    candidates = candidates[candidates != best$name]
  }
  return(selected_features)
}

#Extra: Calcula o numero efetivo de parâmetros,
```

```
#considerando regularização L2
dfreedom = function(X,lambda)
{
  if(!is.matrix(X))
    X = as.matrix(X)
  P = ncol(X)
  A = lambda*diag(P) + t(X)%*%X
  P = X%*%ginv(A)%*%t(X)
  return(sum(diag(P)))
}
```

A.2 Rede Neural MLP

```
library(keras)
library(caret)

#Carrega banco de dados (din,dout)
load(db.RData)

#Carrega lista de melhores subconjuntos
load(sets.RData)

N = nrow(din)
P = ncol(din)

############ Embaralha ############
set.seed(57); # repetibilidade
shuffleindexes = sample(N)
din <- din[shuffleindexes,]
dout <- dout[shuffleindexes]
###################################

######################### Particiona os dados ############################
trainIndex <- createDataPartition(dout, p = .7, list = FALSE, times = 1)
db = list()
db$train = list(X = din[trainIndex,],Y = dout[trainIndex])
db$val = list(X = din[-trainIndex,],Y = dout[-trainIndex])
##########################################################################
```

```
############## Normaliza de acordo com o conjunto de testes #############
dmean_out = mean(db$train$Y)
drng_out = sd(db$train$Y)

db$train$Y = (db$train$Y-dmean_out)/drng_out
db$test$Y = (db$test$Y-dmean_out)/drng_out

dmean_in = apply(db$train$X,2,mean)
drng_in = apply(db$train$X,2,sd)

for(col in colnames(db$train$X))
{
  db$train$X[,col] = (db$train$X[,col]-dmean_in[col])/drng_in[col]
  db$val$X[,col] = (db$val$X[,col]-dmean_in[col])/drng_in[col]
}
db$train$X[is.nan(db$train$X)] = 1
db$val$X[is.nan(db$val$X)] = 1

db_trimmed = list()
db_trimmed$train = list(X = db$train$X[,colunas],Y = db$train$Y)
db_trimmed$val = list(X = db$val$X[,colunas],Y = db$val$Y)
#########################################################################


#### Parâmetros de treinamento ####
Nneurons = 20
b_size = 1
epochs = 50
lr = 1e-4
loss = 'logcosh'
act_fnc = "tanh"
optmizer = optimizer_adam
###################################

################ Modelo com todas as variáveis ################
model <- keras_model_sequential()
model %>%
  layer_dense(units = Nneurons,
```

```
              activation = act_fnc,
              input_shape = ncol(db$train$X)) %>%
  layer_dense(units = 1, activation = 'linear')
model %>% compile(
  loss = loss,
  optimizer = optmizer(lr = lr),
  metrics = c('mse')
)
history <- model %>% fit(
  db$train$X, db$train$Y,
  epochs = epochs, batch_size = b_size,
  validation_data = list(db$val$X,db$val$Y)
)
#################################################################



######## Modelo com apenas as variáveis selecionadas #########
model_trim <- keras_model_sequential()
model_trim %>%
  layer_dense(units = Nneurons,
              activation = act_fnc,
              input_shape = ncol(db_trimmed$train$X)) %>%
  layer_dense(units = 1, activation = 'linear')
model_trim %>% compile(
  loss = loss,
  optimizer = optmizer(lr = lr),
  metrics = c('mse')
)
history_trim <- model_trim %>% fit(
  db_trimmed$train$X, db_trimmed$train$Y,
  epochs = epochs, batch_size = b_size,
  validation_data = list(db_trimmed$val$X,db_trimmed$val$Y)
)
#################################################################



########## Plota o resultado dos treinamentos ################
error_trim_v = data.frame(CVerror = history_trim$metrics$val_mean_squared_error,
```

```
                          it = seq(1:epochs), type = "Selecionado - Validação")
error_trim_t = data.frame(CVerror = history_trim$metrics$mean_squared_error,
                          it = seq(1:epochs), type = "Selecionado - Treinamento")

error_full_v = data.frame(CVerror = history$metrics$val_mean_squared_error,
                          it = seq(1:epochs), type = "Completo - Validação")
error_full_t = data.frame(CVerror = history$metrics$mean_squared_error,
                          it = seq(1:epochs), type = "Completo - Treinamento")

error_data = rbind(error_full_t,error_trim_t, error_full_v,error_trim_v)
error_data$CVerror = sqrt(error_data$CVerror)

g = ggplot(data = error_data, aes(y = CVerror, x = it,color = type))
g = g + geom_line()+geom_point()+
  xlab("Iteração")+
  ylab("Raiz do erro quadrático médio (RMSE)")+
  scale_color_discrete(name = "Conjunto de dados:")+
  labs(title=dtitle, subtitle = paste("MLP de camada única - Treinamento",
                                      Nneurons,"neurons."))
g
#################################################################
```