ANDREA CAROLINA PERES KULAIF

TÉCNICAS DE REGULARIZAÇÃO PARA MÁQUINAS DE APRENDIZADO EXTREMO

CAMPINAS

2014

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e da Computação

ANDREA CAROLINA PERES KULAIF

TÉCNICAS DE REGULARIZAÇÃO PARA MÁQUINAS DE APRENDIZADO EXTREMO

Dissertação apresentada à Pós-graduação da Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na Área de Engenharia de Computação.

Orientador: Prof. Dr. Fernando José Von Zuben

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DE DISSERTAÇÃO DE MESTRADO DA ALUNA ANDREA CAROLINA PERES KULAIF, ORIENTADA PELO PROF. DR. FERNANDO JOSÉ VON ZUBEN.

CAMPINAS
2014

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

Kulaif, Andrea Carolina Peres, 1988-
K958t Técnicas de regularização para máquinas de aprendizado extremo / Andrea Carolina Peres Kulaif. – Campinas, SP : [s.n.], 2014.

Orientador: Fernando José Von Zuben.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Redes neurais artificiais. 2. Análise de regressão. 3. Aprendizado de máquina. I. Von Zuben, Fernando José,1968-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Regularization techniques for extreme learning machines
Palavras-chave em inglês:
Artificial neural networks
Regression analysis
Machine learning
Área de concentração: Engenharia de Computação
Titulação: Mestra em Engenharia Elétrica
Banca examinadora:
Fernando José Von Zuben [Orientador]
Guilherme Palermo Coelho
Romis Ribeiro de Faissol Attux
Data de defesa: 26-08-2014
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE MESTRADO

Candidata: Andrea Carolina Peres Kulaif

Data da Defesa: 26 de agosto de 2014

Título da Tese: "Técnicas de Regularização para Máquinas de Aprendizado Extremo"

Prof. Dr. Fernando José Von Zuben (Presidente): Fernando José Von Zuben

Prof. Dr. Guilherme Palermo Coelho: _____

Prof. Dr. Romis Ribeiro de Faissol Attux: _____

Resumo

Máquinas de Aprendizado Extremo (ELMs, do inglês *Extreme Learning Machines*) são redes neurais com uma camada de entrada, uma camada intermediária e uma camada de saída. Sua arquitetura é equivalente à do perceptron de múltiplas camadas (MLP, do inglês *Multilayer Perceptron*), mas os pesos e o número de neurônios da camada intermediária em ELMs são definidos arbitrariamente e a priori, enquanto os pesos da camada de saída são os únicos parâmetros ajustáveis durante o processo de treinamento supervisionado. O ajuste desses pesos da camada de saída leva a um problema de otimização linear, responsável pelo fato de o treinamento de uma ELM ser ao menos uma ordem de magnitude mais rápido do que o treinamento de uma MLP. No entanto, o desempenho das ELMs é bastante influenciado pelo grau de regularização adotado no ajuste dos pesos da camada de saída. Além disso, por serem definidos de forma arbitrária e não estarem susceptíveis a treinamento, geralmente opera-se com um número elevado de neurônios na camada intermediária, sendo muito comum a presença de neurônios redundantes e pouco funcionais, de modo que sua extração não compromete o desempenho da rede neural como um todo. Em vista desse cenário, esta dissertação apresenta contribuições, junto a ELMs voltadas para tarefas de regressão, em duas frentes principais: (1) Definição de um grau apropriado de regularização no cálculo dos pesos da camada de saída de ELMs; e (2) Ajuste do número de neurônios na camada intermediária, pela extração automática de neurônios redundantes ou pouco funcionais. Mostra-se neste trabalho que, diferentemente do que é praticado na literatura, o grau de regularização mais indicado para uma ELM não varia apenas entre problemas de regressão distintos, mas dentro do mesmo problema de regressão, ainda que se mantenha constante o número de neurônios na camada intermediária. Uma vez detectada a necessidade do ajuste do grau de regularização a cada configuração de ELM, propõe-se aqui uma busca unidimensional mais refinada do que aquela já existente na literatura. Quanto à proposta apresentada para definição do número de neurônios na camada intermediária, faz-se uma análise do espaço de características gerado pelos neurônios dessa camada e aplica-se poda de neurônios que pouco ou nada contribuem para gerar este espaço de características. Além disso, propõe-se o emprego de LASSO e *Elastic Net* como técnicas de regularização, as quais promovem poda adicional de neurônios que não contribuem para a tarefa de regressão. Para as duas contribuições do trabalho, são apresentados resultados experimentais e comparações com outras propostas da literatura. Há um aumento de custo computacional com as propostas deste trabalho, mas é expressivo o ganho em desempenho em alguns cenários.

Abstract

Extreme learning machines are neural networks composed of one input layer, one hidden layer, and one output layer. Their architecture is equivalent to the one of the multilayer perceptron, but the weights and the number of hidden neurons in ELMs are defined a priori and in an arbitrary manner, while the weights at the output layer are the only adjustable parameters during supervised learning. The adjustment of the weights at the output layer leads to a linear optimization problem, responsible for the fact that the training of an ELM be at least one order of magnitude faster than training an MLP. However, the performance of ELMs is greatly influenced by the regularization degree adopted when setting the weights of the output layer. Moreover, by being arbitrarily defined and not susceptible to learning, it is usual to adopt a large number of neurons in the hidden layer, thus promoting the presence of redundant and poorly functional neurons, so that their extraction does not compromise the neural network performance as a whole. Under those circumstances, this dissertation presents contributions to ELMs devoted to regression problems on two main fronts: (1) Definition of a proper degree of regularization when computing the weights at the output layer of ELMs; and (2) Adjustment of the number of neurons at the hidden layer, by the automatic extraction of redundant or poorly functional neurons. It is shown in this work that, different from what is done in the literature, the most suitable degree of regularization for an ELM varies not only among different regression problems, but within the same regression problem and even if it is kept constant the number of neurons at the hidden layer. Once detected the need for adjustment of the degree of regularization for each ELM configuration, it is proposed here a one-dimensional search endowed with more refined steps than that available in the literature. Regarding the proposal to properly define the number of neurons in the hidden layer, an analysis of the feature space generated by neurons in the hidden layer is performed and neurons with no or even a shallow contribution to span the feature space are pruned. Furthermore, the use of LASSO and Elastic Net as regularization techniques are proposed here, which promotes additional pruning of neurons that do not contribute to the regression task. For both contributions of this work, experimental results and comparisons with other proposals in the literature are presented. There is an increase in the computational cost with the proposals of this work, but it is significant the gain in performance in some scenarios.

Sumário

Dedicatória

A meus pais
Fuad e Márcia e
a meu irmão Lucas.

Agradecimentos

Agradeço ao meu orientador Prof. Dr. Fernando José Von Zuben por ter acreditado em mim e ter me guiado pelo caminho do Mestrado. Representa para mim um exemplo de dedicação e atenção com seus alunos e com sua profissão.

Agradeço aos professores da FEEC e IC, onde completei meus créditos. Sua atenção e competência fizeram a diferença no meu crescimento intelectual.

Agradeço aos meus colegas do LBiC pelo apoio nos momentos de dificuldade e por tornar o ambiente do laboratório descontraído e agradável.

Agradeço aos amigos que fiz durante o período que cursei as disciplinas. Por todas as aulas que me deram, por todo o apoio nos momentos de dúvida, pelo carinho com que me receberam em Campinas. Um agradecimento especial a Kamila, Henrique, Hugo e Suelen, que agiram como irmãos nessa caminhada.

Agradeço ao meu namorado Everaldo pela paciência e por estar sempre por perto. Mesmo quando a distância física era grande, sempre pude contar com seu apoio.

Agradeço ao meu irmão Lucas, que sempre torceu por mim, acreditando no meu potencial.

Agradeço especialmente e acima de tudo aos meus pais, Fuad e Márcia. Todas as minhas conquistas sempre serão fruto do que me ensinaram.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo suporte financeiro.

Lista de Figuras

2. Melhorias na Regularização de ELMs com Busca por Seção Áurea

3. *Elastic Net* aplicado a Máquinas de Aprendizado Extremo

Lista de Tabelas

2. Melhorias na Regularização de ELMs com Busca por Seção Áurea

3. *Elastic Net* aplicado a Máquinas de Aprendizado Extremo

Lista de Abreviaturas

ELM	Máquinas de Aprendizado Extremo, do inglês *Extreme Learning Machine*
EQM	Erro Quadrático Médio
EQMv	Erro Quadrático Médio junto ao conjunto de Validação
$EQMv_{médio}$	Média do Erro Quadrático Médio junto ao conjunto de Validação
LARS	*Least Angle Regression*
LARS-EN	*Least Angle Regression - Elastic Net*
LASSO	*Least Absolute Shrinkage and Selection Operator*
MLP	*Multilayer Perceptron*
MRSR	*Multi-Response Sparse Regression*
OLS	*Ordinary Least Square*
OP-ELM	*Optimally Pruned Extreme Learning Machine*
RBF	*Radial Basis Function*
SHLFNN	*Single Hidden Layer Feedfoward Neural Network*
SVM	*Support Vector Machine*
SVD	*Singular Value Decomposition*

Lista de Símbolos

n	Número de neurônios na camada intermediária
N	Número de amostras de entrada-saída
l	Índice da amostra de entrada-saída de dados
x_{li}	Representa o *i*-ésimo componente da *l*-ésima amostra de entrada
v_{ji}	Peso da camada intermediária que conecta a *i*-ésima entrada ao *j*-ésimo neurônio
$f(\cdot)$	Função de ativação de um neurônio da camada intermediária
h_{lj}	Ativação do *j*-ésimo neurônio da camada intermediária para a *l*-ésima amostra de entrada
w_{kj}	Peso da camada de saída que conecta o *j*-ésimo neurônio da camada intermediária ao *k*-ésimo neurônio da camada de saída
$\hat{s}_{lk}$	Corresponde à *k*-ésima saída da rede para a *l*-ésima amostra de entrada
s_{lk}	Representa o *k*-ésimo componente da *l*-ésima amostra de saída
H	Matriz de dimensão $N \times (n+1)$ que representa o resultado de ativação de todos os neurônios da camada intermediária para todas as amostras de entrada, incluindo também um vetor unitário como primeira coluna
$\|\cdot\|_2$	Norma ℓ_2 ou norma euclidiana
$\|\cdot\|_1$	Norma ℓ_1
$H^{\dagger}$	Pseudoinversa Moore-Penrose de H
c	Coeficiente de regularização dos métodos de regressão linear *Ridge Regression*, LASSO e *Elastic Net*
ϕ	Constante real que denota a proporção áurea e tem valor arredondado 1,618
a_k	Limite inferior do intervalo de busca da seção áurea na iteração *k*
b_k	Limite superior do intervalo de busca da seção áurea na iteração *k*
μ_k	Ponto intermediário inferior do intervalo de busca da seção áurea na iteração *k*
λ_k	Ponto intermediário superior do intervalo de busca da seção áurea na iteração *k*
α	Parâmetro da *Elastic Net* que define o *trade-off* entre *Ridge Regression* (α=0) e LASSO (α=1)
U	Matriz ortogonal $N \times N$
D	Matriz retangular $N \times (n+1)$ com elementos não-negativos na diagonal, denominados de valores singulares

V	Matriz ortogonal $(n+1)\times(n+1)$
$\sigma_{\max}$	Maior valor singular da matriz *H*
$\sigma_{\min}$	Menor valor singular da matriz *H*
A	Algoritmo cujos parâmetros serão utilizados pelo algoritmo *ParamILS*
CD	Distribuição de instâncias de um problema
Θ	Espaço de configurações de parâmetros
θ	Configuração de parâmetros, $\theta \in \Theta$
$A(\theta)$	Algoritmo *A* com a configuração de parâmetros θ, $\theta \in \Theta$
$sc(A,\theta,I)$	Custo escalar do algoritmo *A* com configuração de parâmetros θ, para uma instância *I*
$CD(A,\theta,I)$	Distribuição de custos do algoritmo *A* com configuração de parâmetros θ, para uma instância *I*
$c(\theta)$	Estatística de distribuição de custos para a configuração de parâmetros θ, $\theta \in \Theta$
$\hat{c}_N(\theta)$	Aproximação de $c(\theta)$ para *N* amostras
M	Conjunto de dados de treinamento

Capítulo 1

Introdução

1.1 Motivação da proposta

Redes neurais artificiais vêm sendo amplamente aplicadas em problemas de regressão e classificação de dados [10]. Sua capacidade de aproximação universal permite que sejam sintetizados mapeamentos não-lineares contínuos que se aproximam arbitrariamente bem de mapeamentos desejados. Porém, encontrar a arquitetura apropriada a cada problema ainda é um desafio em aberto, já que não existem métodos definitivos para determinar a quantidade de neurônios mais adequada, assim como a configuração de pesos que maximiza a capacidade de generalização da rede neural. Entende-se por capacidade de generalização a habilidade de responder adequadamente a amostras não utilizadas durante a fase de treinamento da rede neural.

Além disso, apesar do crescimento acentuado da capacidade de processamento e memória dos computadores digitais utilizados para implementar redes neurais artificiais, ainda pode ser muito custoso realizar o treinamento dos pesos sinápticos, particularmente quando este envolve o ajuste dos pesos dos neurônios da camada intermediária. No caso das redes com arquitetura do tipo *perceptron* de múltiplas camadas (MLP, do inglês *Multilayer Perceptron*) [30], métodos de otimização não-linear são geralmente aplicados. E, para a definição dos centros das funções de base radial, no caso de uma arquitetura com esse tipo de função de ativação (RBF, do inglês *Radial Basis Function*) [5], são geralmente utilizados métodos auto-organizáveis. São soluções baseadas em processos iterativos e que, além de custosos computacionalmente, são sujeitos a mínimos locais.

Uma das opções para evitar grande parte dessas dificuldades é o emprego de Máquinas de Vetores Suporte (SVMs, do inglês *Support Vector Machines*) [36]. Diretamente baseadas na minimização do risco estrutural [31], as SVMs oferecem melhores soluções na definição da quantidade de neurônios, que são associados com o número de vetores-suporte, e do conjunto de pesos sinápticos, obtido pela definição de um hiperplano que mantenha máxima margem no espaço de características, supondo problemas de classificação binária. A formulação matemática resultante conduz a um problema convexo de otimização quadrática com restrições. Porém, o desempenho das SVMs ainda é condicionado à definição apropriada das funções kernel e seus parâmetros correspondentes [20]. As funções kernel estão associadas ao mapeamento do espaço original dos dados para o espaço de características. Além disso, lidar com o problema de otimização quadrática resultante, sujeito a

restrições de igualdade ou desigualdade [34], pode ser bastante custoso computacionalmente. Há também um coeficiente de regularização a ser determinado, similar ao que será apresentado a seguir para máquinas de aprendizado extremo (ELMs, do inglês *Extreme Learning Machines*) [14].

Como uma alternativa às limitações das abordagens anteriores, particularmente referente ao custo computacional vinculado à fase de treinamento e à necessidade de especificação de múltiplos parâmetros, foram propostas as ELMs [14]. Estas redes são estruturalmente equivalentes às MLPs, porém têm os pesos sinápticos dos neurônios da camada intermediária determinados aleatoriamente, em intervalos pré-definidos, resultando em um tempo de treinamento muito inferior ao obtido para redes MLP, RBF e SVM. Isso ocorre porque apenas os pesos da camada de saída devem ser ajustados, produzindo assim um problema de regressão linear. Normalmente, as ELMs possuem grande quantidade de neurônios, a qual é definida de forma arbitrária. Para evitar sobreajuste no treinamento, os pesos da camada de saída são ajustados através do método de *Ridge Regression* [11], de forma que a capacidade de generalização é diretamente controlada pelo coeficiente de regularização escolhido para esta etapa.

A partir da formulação original das ELMs, surgiram muitas metodologias para definição do número de neurônios da camada intermediária, como o algoritmo de poda proposto por Martínez-Martínez *et al.* [22], que, com intenção de obter uma arquitetura mais parcimoniosa, aplica métodos de regressão regularizada, como LASSO (*Least Absolute Shrinkage and Selection Operator*) [35] e *Elastic Net* [39], através dos quais é feita a seleção dos neurônios da camada intermediária que mais contribuem para o desempenho da rede.

Outra abordagem de definição do número de neurônios na camada intermediária é conhecida como OP-ELM (do inglês *Optimally Pruned* ELM), proposta por Miche *et al.* [24]. Assim como o método apresentado por Martínez-Martínez *et al.* [22], a rede é inicializada com uma estrutura de elevada dimensão e, após efetuar a classificação dos neurônios através do algoritmo MRSR (do inglês *Multi-Response Sparse Regression*) [33], são eliminados os neurônios menos significativos via validação cruzada *leave-one-out* [18].

Foi proposto também por Miche *et al.* [23] um aperfeiçoamento da abordagem OP-ELM, através de dois passos: penalização da norma ℓ_1 para classificar os neurônios da camada intermediária, seguida pela penalização da norma ℓ_2 dos pesos da camada de saída. É uma combinação da ideia original do OP-ELM com o apresentado por Martínez-Martínez *et al.* [22], que aplica LASSO para selecionar os neurônios da camada intermediária.

1.2 Objetivos e contribuições

A primeira contribuição deste trabalho visa aperfeiçoar o cálculo dos pesos da camada de saída das ELMs, empregando uma estratégia de busca refinada para obter o coeficiente de regularização do método *Ridge Regression*, através da aplicação de uma busca unidimensional conhecida como seção áurea. Esta proposta é sustentada pelo fato de que, para cada conjunto de pesos na camada intermediária, desvios no valor do coeficiente de regularização podem gerar resultados significativamente diferentes. São apresentados resultados promissores para tarefas de regressão, incluindo comparações com abordagens alternativas. Nesta pesquisa, não são considerados problemas de classificação.

Como uma evolução da abordagem anterior, a segunda contribuição deste trabalho é representada pela substituição do método de *Ridge Regression*, que tem como característica a penalização da norma ℓ_2 dos pesos da camada de saída, pelo método LASSO, que substitui a norma ℓ_2 pela norma ℓ_1, e *Elastic Net*, que representa um compromisso entre os dois critérios de penalização anteriores.

Através da capacidade de seleção de variáveis de LASSO e *Elastic Net*, aliada com um método de tratamento de condição da matriz *H*, que vai estar associada às ativações dos neurônios da camada intermediária, é possível obter estruturas de rede mais enxutas que alcancem o mesmo nível de desempenho sem aumentar significativamente o custo computacional da etapa de treinamento da rede neural artificial. Além disso, é empregada uma técnica de busca baseada no algoritmo *ParamILS*, proposto por Hutter *et al.* [17], para a definição dos parâmetros da *Elastic Net*.

1.3 Escopo das contribuições alcançadas

Uma das principais características das ELMs está na definição dos pesos sinápticos da camada intermediária de forma aleatória. Em conjunto com os diferentes possíveis particionamentos dos dados em conjuntos de treinamento, validação e teste, pode-se obter redes muito distintas entre si a cada inicialização. Apoiada nesse fato, a primeira contribuição deste trabalho propõe uma busca refinada do valor do coeficiente de regularização *c* para cada inicialização da ELM. Esta iniciativa difere do realizado em Huang *et al.* [14], onde é aplicado apenas um valor de *c* para toda e qualquer inicialização da ELM quando se aborda um mesmo problema de regressão.

Com foco na definição do número de neurônios na camada intermediária, a segunda contribuição deste trabalho utiliza-se de duas fases de poda. A primeira é realizada através do

tratamento do número de condição de $H^{\mathrm{T}}H$, sendo que H é uma matriz que representa a saída da camada intermediária mais uma coluna unitária. A cada iteração, é retirada a coluna de H que, na sua ausência, permite que $H^{\mathrm{T}}H$ tenha a maior melhora no seu número de condição, até que seja alcançado um limiar. Dado que retirar colunas de H é equivalente a podar neurônios da camada intermediária da ELM, já é possível obter uma arquitetura mais parcimoniosa através deste método.

A segunda fase de poda é feita através da aplicação do LASSO e *Elastic Net* sobre a nova matriz H, para a definição dos pesos da camada de saída. Como extensão do apresentado em Martínez-Martínez *et al.* [22], é aplicado o algoritmo *ParamILS* [17] para a definição dos parâmetros da *Elastic Net*, que também variam para um mesmo problema de regressão, de acordo com a inicialização da ELM.

Enquanto o OP-ELM proposto por Miche *et al.* [24] classifica os neurônios da camada intermediária através do algoritmo MRSR, para determinar quais devem ser podados através de validação cruzada *leave-one-out* [18], este trabalho utiliza o tratamento do número de condição de $H^{\mathrm{T}}H$ aliado à capacidade de seleção de variáveis da *Elastic Net* na definição dos pesos da camada de saída.

Também difere da proposta deste trabalho o apresentado em Miche *et al.* [23], já que este último aplica a penalização da norma ℓ_1 para classificar os neurônios da camada intermediária e apenas a penalização da norma ℓ_2 no cálculo dos pesos da camada de saída, representado por *Ridge Regression*.

1.4 Organização do texto

O capítulo 2 descreve a proposta original das ELMs e a aplicação do método de *Ridge Regression* no cálculo dos pesos da camada de saída. Ressalta-se principalmente a necessidade da busca pelo coeficiente de regularização c para cada configuração de ELM, com uma proposta de busca por c através do método da seção áurea. Além disso, são apresentados os experimentos realizados e resultados obtidos por esta abordagem junto a sete conjuntos de dados distintos, concluindo a primeira contribuição deste trabalho.

Com o objetivo de introduzir a segunda contribuição deste trabalho, o capítulo 3 apresenta uma explanação acerca dos métodos de regressão linear LASSO e *Elastic Net*. Duas ferramentas importantes para esta proposta também são descritas neste capítulo: o tratamento da matriz H e o algoritmo de busca *ParamILS*. A partir desses conceitos, é feito o detalhamento do método de poda

para ELM sugerido neste trabalho e a aplicação nos mesmos sete conjuntos de dados utilizados nos experimentos do capítulo 2.

O capítulo 4 contém uma análise dos resultados dos experimentos e das principais contribuições de cada proposta, a qual é seguida pelas considerações finais e perspectivas futuras da pesquisa.

Capítulo 2

Melhorias na Regularização de ELMs com Busca por Seção Áurea

2.1 Máquinas de Aprendizado Extremo

Redes neurais são amplamente estudadas e, dessa forma, é possível encontrar uma grande variedade de arquiteturas, métodos de treinamento e construção das mesmas. Uma das mais populares, no entanto, são conhecidas como redes neurais com conexões à frente (FNN, do inglês *Feedforward Neural Networks*), compostas por uma camada de entrada, pelo menos uma camada intermediária e uma camada de saída. Têm como característica principal o fato de não possuírem sinais realimentados, ou seja, o fluxo de dados se dá sempre no sentido das camadas mais à frente. Dentro desta classe, existe um grupo de redes mais específicas que possuem apenas uma camada intermediária, conhecidas como SHLFNNs (do inglês *Single Hidden Layer Feedfoward Neural Networks*). A Figura 2.1 apresenta a estrutura de uma SHLFNN, também conhecida na literatura como MLP (do inglês *Multilayer Perceptron*).

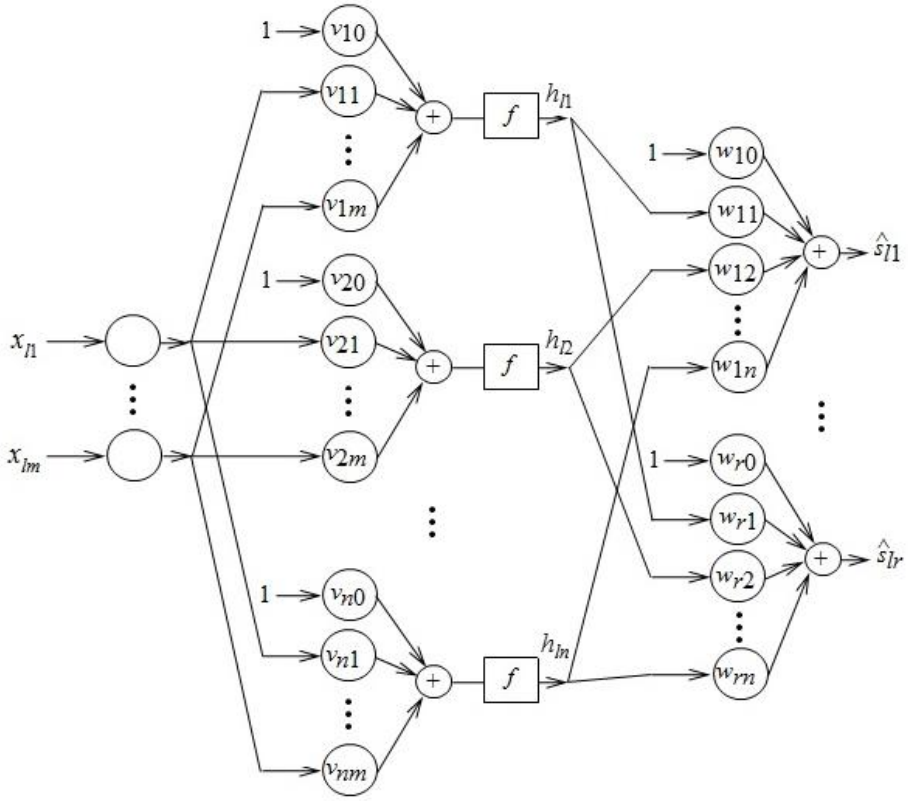

Figura 2.1 - SHLFNN com *m* entradas, *n* neurônios na camada intermediária e *r* saídas.

Na Figura 2.1, x_{li} representa o i-ésimo componente da l-ésima amostra de entrada; v_{ni} corresponde ao peso da camada intermediária que conecta a i-ésima entrada ao j-ésimo neurônio; f é a função de ativação de um neurônio da camada intermediária; h_{lj} é a saída do j-ésimo neurônio da camada intermediária, quando a rede neural recebe na entrada a l-ésima amostra de entrada; w_{kj} representa o peso da conexão entre o j-ésimo neurônio da camada intermediária e o k-ésimo neurônio da camada de saída; $\hat{s}_{lk}$ corresponde à k-ésima saída da rede para a l-ésima amostra de entrada.

Foram desenvolvidas muitas abordagens de treinamento para as FNNs, como as baseadas em métodos de otimização não-linear de 1ª e 2ª ordem para MLPs; as baseadas em auto-organização e no método dos quadrados mínimos para redes RBFs; a baseada em otimização convexa para SVMs, entre outras. Mas todas ainda são computacionalmente custosas. MLPs e RBFs ainda estão sujeitas a mínimos locais, enquanto que SVMs requerem a definição da função kernel e de seus parâmetros.

Buscando evitar esses problemas, foram propostas as ELMs, do inglês *Extreme Learning Machines* (ELMs) [14][16]. São redes do tipo SHLFNN, portanto possuem a mesma estrutura apresentada na Figura 2.1. Tem como vantagem principal o fato de não efetuarem treinamento dos pesos da camada intermediária, que são definidos aleatoriamente dentro de um intervalo pré-estabelecido a ser definido pelo usuário. O objetivo dessas redes é, além de alcançar o menor erro de treinamento, manter baixa a norma dos pesos da camada de saída, calculados através do método *Ridge Regression*. Busca-se assim garantir um baixo custo computacional na fase de treinamento e obter níveis competitivos de desempenho em generalização.

Dois aspectos são formalmente comprovados em Huang *et al.* [14], no tocante à capacidade de aprendizado das ELMs: capacidade de interpolação e capacidade de aproximação universal.

Do ponto de vista de interpolação, as ELMs permitem utilizar uma grande variedade de funções de ativação, como funções sigmoides, funções de base radial, seno, cosseno, exponencial e muitas outras funções não-regulares. Porém, este trabalho vai tratar especificamente de redes com função de ativação tangente hiperbólica e problemas de regressão ou aproximação de funções.

O mapeamento realizado pelas ELMs pode ser representado como na Equação 2.1:

$$\hat{s}_{lk} = \sum_{j=1}^{n} w_{kj} f\left(\sum_{i=1}^{m} v_{ji} x_{li} + v_{j0} \right) + w_{k0}, \tag{2.1}$$

onde $k \in \{1,...,r\}$ é o índice da saída e $l \in \{1,...,N\}$ é o índice da amostra de entrada-saída de dados. Os pesos v_{ji} ($j \in \{1,...,n\}$; $i \in \{1,...,m\}$) e w_{kj} ($k \in \{1,...,r\}$; $j \in \{1,...,n\}$) já foram definidos ao se apresentar a

Figura 2.1. Note a existência de uma entrada constante, fixada em 1, associada a cada neurônio da rede, o que justifica a presença dos pesos v_{j0} ($j \in \{1,...,n\}$) e w_{k0} ($k \in \{1,...,r\}$).

A ativação do j-ésimo neurônio ($j \in \{1,...,n\}$) da camada intermediária para cada amostra de entrada $l \in \{1,...,N\}$ será representada como na Equação 2.2:

$$h_{lj} = f\left(\sum_{i=1}^{m} v_{ji} x_{li} + v_{j0} \right). \tag{2.2}$$

Então, o resultado de ativação de todos os neurônios da camada intermediária para todas as amostras de dados pode ser representado por uma matriz H, dada na Equação 2.3:

$$H = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & & \ddots & \vdots \\ h_{N1} & h_{N2} & \cdots & h_{Nn} \end{bmatrix}. \tag{2.3}$$

Representando os pesos do k-ésimo neurônio da camada de saída por $\mathbf{w}_k = [w_{k0}\ w_{k1}\ ...\ w_{kn}]^T$, e inserindo uma coluna de 1's como primeira coluna de H, a k-ésima saída $\hat{\mathbf{s}}_k$ ($k \in \{1,...,r\}$) para todas as amostras de entrada-saída ($l \in \{1,...,N\}$) é dada pela Equação 2.4:

$$\hat{\mathbf{s}}_k = \begin{bmatrix} \hat{s}_{1k} \\ \hat{s}_{2k} \\ \vdots \\ \hat{s}_{Nk} \end{bmatrix} = \begin{bmatrix} 1 & h_{11} & h_{12} & \cdots & h_{1n} \\ 1 & h_{21} & h_{22} & \cdots & h_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & h_{N1} & h_{N2} & \cdots & h_{Nn} \end{bmatrix} \begin{bmatrix} w_{k0} \\ w_{k1} \\ \vdots \\ w_{kn} \end{bmatrix} \Rightarrow \hat{\mathbf{s}}_k = H\mathbf{w}_k. \tag{2.4}$$

Resulta, portanto, um problema de regressão linear para cada uma das r saídas da rede neural, onde é necessário propor uma solução para um sistema linear apresentado na Equação 2.4. Visto que a matriz H tem dimensão $N \times (n+1)$, e sabendo que o número n de neurônios na camada intermediária é arbitrário, pode-se chegar a menos ($N < (n+1)$), mais ($N > (n+1)$) ou até tantas equações quanto incógnitas ($N = (n+1)$). Desde que não haja coincidência entre quaisquer pares de vetores de pesos dos n neurônios da camada intermediária, é esperado que a matriz H tenha posto completo, ou seja posto(H)=min(N, n+1) [13].

2.2 *Ordinary Least Square* e *Ridge Regression*

Em suas primeiras proposições, o treinamento de uma rede ELM envolvia somente a solução de quadrados mínimos, independente para cada saída k ($k \in \{1,...,r\}$), sem nenhum termo vinculado a aspectos de regularização [16]. Dessa forma, procurava-se resolver o seguinte problema de otimização, denominado *Ordinary Least Square* (OLS), onde $\|\cdot\|_2$ é a norma ℓ_2 ou norma euclidiana:

$$\min_{\mathbf{w}_k} \|H\mathbf{w}_k - \mathbf{s}_k\|_2^2, \tag{2.5}$$

onde $\mathbf{s}_k$ é o vetor de saídas desejadas para a saída k ($k \in \{1,...,r\}$).

A solução de norma mínima para $\mathbf{w}_k$ é obtida utilizando a pseudoinversa Moore-Penrose $H^{\dagger}$ [28][32] como dado na Equação 2.6:

$$\mathbf{w}_k = H^{\dagger}\mathbf{s}_k. \tag{2.6}$$

Se $(n+1) \leq N$ e $H^{\mathrm{T}}H$ é não singular, então $H^{\dagger} = \left(H^{\mathrm{T}}H\right)^{-1}H^{\mathrm{T}}$.

Se $(n+1) \geq N$ e HH^{T} é não singular, então $H^{\dagger} = H^{\mathrm{T}}\left(HH^{\mathrm{T}}\right)^{-1}$.

Neste último caso, ou seja, quando $(n+1) \geq N$, a pseudo-inversa leva a uma regularização "espontânea" dos pesos, sendo que a solução obtida será a de norma mínima dentre as infinitas soluções existentes.

De acordo com Bartlett [1], a redução da norma dos pesos da camada de saída é mais relevante para a capacidade de generalização do que o número de neurônios na camada intermediária. Então, para controlar a norma dos pesos, utiliza-se um coeficiente de regularização $c_k \in R^+$, conduzindo ao problema regularizado apresentado a seguir:

$$\min_{\mathbf{w}_k \in R^{n+1}} \|H\mathbf{w}_k - \mathbf{s}_k\|_2^2 + c_k\|\mathbf{w}_k\|_2^2. \tag{2.7}$$

Resolver o problema da Equação 2.7 é equivalente a adicionar um valor positivo c_k aos elementos da diagonal de $H^T H$ ou de HH^T, produzindo respectivamente:

$$\mathbf{w}_k = (H^{\mathrm{T}}H + c_k I)^{-1}H^{\mathrm{T}}\mathbf{s}_k, \tag{2.8}$$

onde I representa a matriz identidade de dimensão $(n+1)\times(n+1)$ ou

$$\mathbf{w}_k = H^{\mathrm{T}}\left(HH^{\mathrm{T}} + c_k I\right)^{-1}\mathbf{s}_k, \tag{2.9}$$

onde *I* representa a matriz identidade de dimensão *N*×*N*.

A dedução dessas duas soluções para o problema de otimização regularizado apresentado na Equação 2.7 é apresentada a seguir, aplicando diretamente a condição necessária de otimalidade, a qual afirma que o gradiente é nulo nos pontos extremos de uma função.

Para tanto, considere as seguintes propriedades de cálculo matricial:

$\frac{\partial}{\partial \mathbf{y}}\left(\mathbf{y}^T A\mathbf{x}\right) = A\mathbf{x}$; $\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{y}^T A\mathbf{x}\right) = A^T\mathbf{y}$; para $A^T = A$, $\frac{\partial}{\partial \mathbf{x}}\left(\mathbf{x}^T A\mathbf{x}\right) = 2A\mathbf{x}$.

Derivando a função-objetivo da Equação 2.7 em relação a $\mathbf{w}_k$, resulta:

$$\frac{\partial\left[\left\|H\mathbf{w}_k - \mathbf{s}_k\right\|_2^2 + c_k\left\|\mathbf{w}_k\right\|_2^2\right]}{\partial \mathbf{w}_k} = \frac{\partial\left[\left(H\mathbf{w}_k - \mathbf{s}_k\right)^T\left(H\mathbf{w}_k - \mathbf{s}_k\right) + c_k \mathbf{w}_k{}^T\mathbf{w}_k\right]}{\partial \mathbf{w}_k} =$$

$$= \frac{\partial\left[\left(H\mathbf{w}_k - \mathbf{s}_k\right)^T\left(H\mathbf{w}_k - \mathbf{s}_k\right)\right]}{\partial \mathbf{w}_k} + c_k\frac{\partial\left[\mathbf{w}_k{}^T\mathbf{w}_k\right]}{\partial \mathbf{w}_k} =$$

$$= \frac{\partial\left[\mathbf{w}_k{}^T H^T H\mathbf{w}_k - \mathbf{w}_k{}^T H^T\mathbf{s}_k - \mathbf{s}_k{}^T H\mathbf{w}_k + \mathbf{s}_k{}^T\mathbf{s}_k\right]}{\partial \mathbf{w}_k} + c_k\frac{\partial\left[\mathbf{w}_k{}^T\mathbf{w}_k\right]}{\partial \mathbf{w}_k} =$$

$$= 2H^T H\mathbf{w}_k - H^T\mathbf{s}_k - H^T\mathbf{s}_k + 2c_k I\mathbf{w}_k$$

onde *I* representa a matriz identidade de dimensão (*n*+1)×(*n*+1).

Aplicando a condição necessária de otimalidade, obtém-se:

$$\frac{\partial\left[\left\|H\mathbf{w}_k - \mathbf{s}_k\right\|_2^2 + c_k\left\|\mathbf{w}_k\right\|_2^2\right]}{\partial \mathbf{w}_k} = 0 \Rightarrow 2H^T H\mathbf{w}_k^* - H^T\mathbf{s}_k - H^T\mathbf{s}_k + 2c_k I\mathbf{w}_k^* = 0$$

$$\left(H^T H + c_k I\right)\mathbf{w}_k^* = H^T\mathbf{s}_k \Rightarrow \mathbf{w}_k^* = \left(H^T H + c_k I\right)^{-1} H^T\mathbf{s}_k$$

Para o caso em que (*n*+1) > *N*, a dedução é similar e produz $\mathbf{w}_k{}^* = H^T\left(HH^T + c_k I\right)^{-1}\mathbf{s}_k$, onde *I* representa a matriz identidade de dimensão *N*×*N*.

Converge-se então para o método *Ridge Regression* [11] no cálculo dos pesos do *k*-ésimo neurônio da camada de saída de uma rede ELM ($k \in \{1,...,r\}$), ficando sob responsabilidade do usuário a definição do coeficiente de regularização c_k.

2.3 A necessidade de coeficientes de regularização distintos para cada inicialização

Como apresentado na seção 2.1, os pesos da camada intermediária das ELMs são definidos aleatoriamente dentro de um intervalo pré-estabelecido definido pelo usuário, sendo arbitrariamente

distintos de uma inicialização para outra. Apenas isso já produz matrizes H muito diversas, independente de se estar considerando um mesmo problema de regressão com uma quantidade fixa de neurônios na camada intermediária.

Porém, existe ainda o fato de o particionamento dos dados em treinamento, validação e teste poder ser modificado. E já que essas são características que influenciam diretamente na formação de H, é necessário que essas diferenças sejam levadas em conta ao buscar o valor do coeficiente de regularização c_k, associado à k-ésima saída da rede neural. Dessa forma, cada inicialização demanda um valor específico de c_k, o que geralmente não é considerado na literatura.

2.4 Busca por seção áurea

A proporção áurea, ou razão áurea, é uma constante real denotada pela letra grega ϕ. Seu valor é $\phi = \frac{\sqrt{5}+1}{2} \cong 1{,}618$.

Desde a antiguidade, a razão áurea é utilizada no campo das artes. Este número pode ser encontrado na proporção entre partes do corpo dos seres humanos, por exemplo, no tamanho das falanges (ossos dos dedos) e em outros aspectos da natureza em geral, pois tem relação com o processo de crescimento [4].

A razão áurea é definida algebricamente como na Equação 2.10 abaixo:

$$\frac{a+b}{a} = \frac{a}{b}. \tag{2.10}$$

A Figura 2.2 representa o Retângulo Áureo, conhecido assim por obedecer a razão áurea, como apresentado na Equação 2.10.

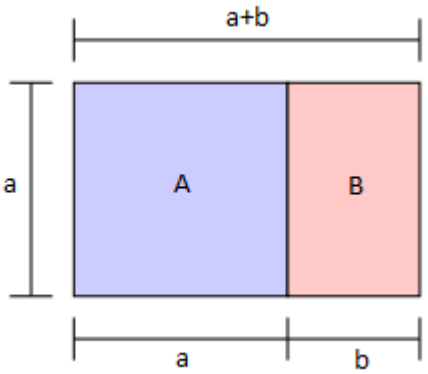

Figura 2.2 - Retângulo Áureo

Ao retirar o quadrado A, obtém-se o retângulo B, que mantém as proporções de lados do retângulo original. E esse processo pode ser realizado novamente com o retângulo B, sendo que essa característica se manterá indefinidamente, independente de quantas vezes esse processo se repetir, como ilustra a Figura 2.3.

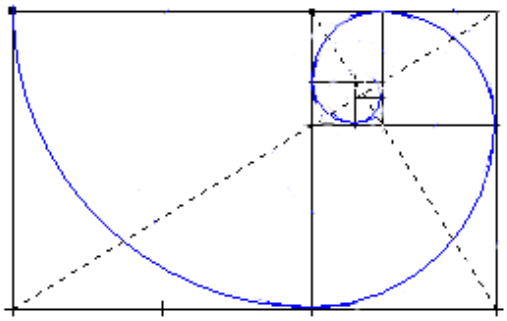

Figura 2.3 - Características do Retângulo Áureo.

É baseado nesses princípios que foi proposto o método de busca por Seção Áurea [21]. Seja $[a_1,b_1]$ o intervalo inicial de busca em um problema de minimização de uma função quase-convexa *f*, por exemplo. A cada iteração *k*, são definidos dois pontos intermediários, μ_k e λ_k, como na Equação 2.11:

$$\begin{aligned} \mu_k &= a_k + \phi(b_k - a_k), \\ \lambda_k &= a_k + (1-\phi)\cdot(b_k - a_k). \end{aligned} \tag{2.11}$$

A Figura 2.4 mostra o resultado da divisão do intervalo e duas situações possíveis que promovem a sua redução de acordo com a razão áurea, ao se verificar qual entre μ_k e λ_k retorna o menor valor da função *f*. O intervalo descartado em cada caso está representado pela linha tracejada em vermelho.

Conforme apresentado na Figura 2.4, se o valor da função *f* em λ_k for menor ou igual que em μ_k, o intervalo $[a_k, \mu_k]$ é descartado. O intervalo de busca *k*+1 passa a ser $[\mu_k, b_k]$ e λ_{k+1} é recalculado. Caso contrário, o intervalo descartado é $[\lambda_k, b_k]$. O intervalo de busca *k*+1 passa a ser $[a_k, \lambda_k]$ e μ_{k+1} é recalculado.

Se $f(\mu_k) \geq f(\lambda_k)$, $a_{k+1} = \mu_k$, $b_{k+1} = b_k$, $\mu_{k+1} = \lambda_k$ e $\lambda_{k+1} = a_{k+1} + (1-\phi)(b_{k+1} - a_{k+1})$.

Se $f(\mu_k) < f(\lambda_k)$, $b_{k+1} = \lambda_k$, $a_{k+1} = a_k$, $\lambda_{k+1} = \mu_k$ e $\mu_{k+1} = a_{k+1} + \phi(b_{k+1} - a_{k+1})$.

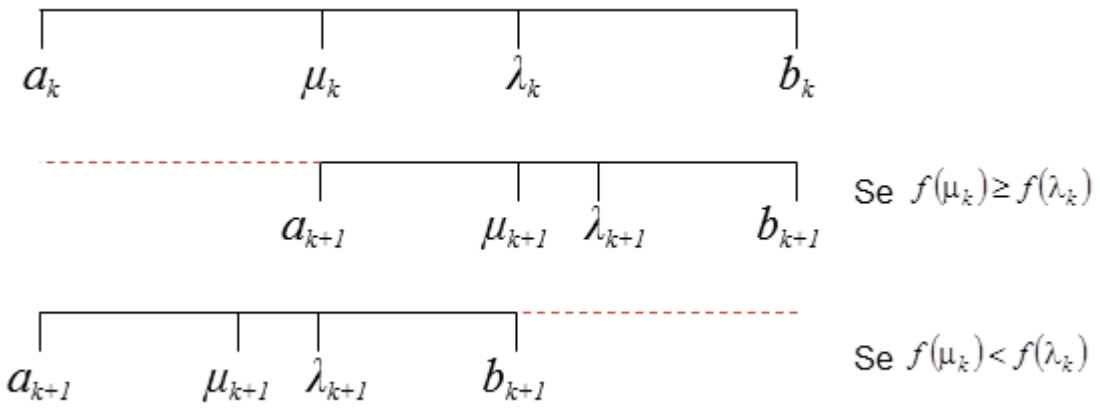

Figura 2.4 - Intervalo de busca e pontos intermediários de acordo com o método da seção áurea, incluindo duas ações possíveis como próximo passo do método, ao supor minimização da função $f(\cdot)$.

O algoritmo finaliza quando o intervalo $[a_k,b_k]$ for menor que um limiar ε definido pelo usuário. O que produzir menor valor de f entre a_k e b_k é definido como solução, ou então toma-se o ponto médio entre eles.

A busca por seção áurea é um processo iterativo e só tem desempenho positivo quando aplicada a intervalos de busca em que a função $f(\cdot)$ se comporta como uma função convexa ou quase-convexa. Caso contrário, o método de busca está sujeito a mínimos locais.

2.5 Busca do coeficiente de regularização por Seção Áurea

No contexto de problemas de regressão, a primeira proposta deste trabalho é uma estratégia de busca do coeficiente de regularização c_k ($k \in \{1,...,r\}$) da ELM quando utilizando *Ridge Regression* no cálculo dos pesos da camada de saída k, método referenciado aqui como ELM com c_k variável. Será considerada sempre uma única saída, visto que o tratamento é independente para cada saída.

Em Huang *et al.* [15], foi proposto o intervalo $[2^{-24}, 2^{+25}]$ para busca do valor ótimo de c_k. Este intervalo é considerado suficientemente amplo em aplicações práticas. Em experimentos preliminares realizados, foi constatado pela autora desta dissertação que aplicar o método de busca por seção áurea no intervalo inteiro $[2^{-24}, 2^{+25}]$ não garante a melhor solução, pois a função-objetivo não tem garantia de ser convexa ou quase-convexa. Logo, a busca fica sujeita a mínimos locais. Sendo assim, decidiu-se por adotar duas etapas de busca.

Primeiramente, em cada execução, são avaliados 50 valores de c no conjunto $\{2^{-24},2^{-23},...,2^{+24},2^{+25}\}$ como proposto em Huang *et al.* [15]. Determina-se o valor de $c=2^a$, com $a \in \{-24,-23,...,+24,+25\}$, que representa a melhor solução do problema formulado na Equação 2.7 e, então, é realizada uma busca por seção áurea no intervalo $[2^{a-1},2^{a+1}]$, de forma a alcançar um valor para c_k mais preciso. Neste intervalo bem mais restrito, e supostamente próximo do valor ótimo de c_k, é bastante improvável que haja mínimos locais e, se houver, a busca, de qualquer modo, não levará a um resultado pior do que o valor já encontrado dentre os 50 valores do intervalo acima. Como critério de parada da busca por seção áurea, foi utilizado um intervalo de busca mínimo de 10^{-5} ou 0,01% do intervalo de busca inicial.

Assim, é possível encontrar o melhor c_k, levando em conta o conjunto de pesos da camada intermediária, diferente do apresentado em Huang *et al.* [15], onde um único c_k é definido para cada conjunto de dados, independente do conjunto de pesos, ou seja, independente da matriz H.

Apesar de computacionalmente mais custosa, constatou-se aqui que a busca por seção áurea trouxe benefícios consistentes no desempenho, contribuindo para a melhora na capacidade de generalização das ELMs para problemas de regressão, como apresentado em Kulaif & Von Zuben [19].

2.6 Simulações e Resultados

Nesta seção, são apresentados os experimentos realizados e os resultados obtidos com a proposta da seção 2.5, assim como a comparação de desempenho com o método ELM proposto em Huang *et al.* [15] e também com a rede neural MLP.

2.6.1 Tratamento de Dados e Configuração de Parâmetros

Para verificar o desempenho dos métodos, foram utilizados conjuntos de dados com dimensões e quantidades de amostras variadas, sendo todos problemas de regressão. Os conjuntos de dados utilizados neste trabalho são coincidentes com alguns dos apresentados nos testes em Huang *et al.* [15]. No entanto, adotamos aqui uma partição diferente dos dados, descrita na seção 2.6.2 deste trabalho.

A Tabela 2.1 especifica as principais características de cada conjunto de dados. Os conjuntos de dados *Bodyfat*, *Space-ga*, *Quake* e *Strike* podem ser encontrados em *Department of Statistics*, *Carnegie Mellon University* [25], e *Abalone*, *Pyrim* e *Housing* estão disponíveis em *University of California at Irvine* (UCI) *Machine Learning Repository* [3].

Tabela 2.1 - Especificações dos Conjuntos de Dados

Conjuntos de Dados	Amostras	Atributos
Pyrim	74	27
Bodyfat	252	14
Housing	506	13
Strike	625	6
Quake	2178	3
Space-ga	3107	6
Abalone	4177	8

As simulações foram realizadas em MATLAB 7.11.0.584 – 64 bits, rodando em Intel Core i5-2430M, 2.4GHz com 4GB de memória RAM.

Além disso, foi feito escalamento dos dados de entrada no intervalo [−1; +1] e dos dados de saída no intervalo [0; +1]. Seguiu-se aqui o pré-processamento realizado em Huang *et al.* [15].

Nas simulações por MLP, foi utilizado o método do gradiente conjugado para o treinamento da rede, proposto por Møller [26]. Os pesos ao final do treinamento são aqueles que minimizam o erro junto ao conjunto de validação.

Como há apenas uma saída, o subíndice k é desnecessário, sendo que o coeficiente de regularização é denotado por c.

Após a realização de testes variando o número de neurônios na camada intermediária, a quantidade de neurônios foi fixada em 100. Com menos do que isso, foi observado que o desempenho do sistema, utilizando o valor de c encontrado pela busca, raramente é superior ao caso sem regularização. Já ao aplicar variações para mais do que 100 neurônios, os resultado alcançados não apresentaram melhora significativa. Portanto, os testes contêm 100 neurônios na camada intermediária, tanto para MLP quanto para ELM com c variável, com particionamento dos dados por *holdout* e *k-fold*. Em todos esses casos, os pesos da camada intermediária foram inicializados aleatoriamente, com distribuição uniforme no intervalo [−0,5; +0,5]. Esse intervalo foi escolhido após estudos preliminares dos pesos definidos pela MLP, para esses problemas de regressão, após a fase de treinamento. Um intervalo pré-definido pode acarretar problemas em tarefas de regressão que exijam pesos com valores fora deste intervalo, porém está ajustado adequadamente à maioria dos conjuntos de dados apresentados na Tabela 2.1. Além disso, não é aconselhável inicializar os pesos da ELM em intervalos de valores mais amplos devido ao seu caráter aleatório.

Para comparação, foi aplicada a mesma técnica de busca encontrada em Huang *et al.* [15] para ELM com função de ativação tangente hiperbólica, referenciada aqui como ELM com c fixo. Como explicado na seção 2.5, nesse caso o valor de c será dado por um dos elementos do conjunto

$\{2^{-24}, 2^{-23}, \cdots, 2^{+24}, 2^{+25}\}$. A Tabela 2.2 apresenta os valores obtidos para cada conjunto de dados nesta configuração com *c* fixo. Não são os mesmo valores apresentados em Huang *et al.* [15] já que lá as configurações da ELM são distintas das aplicadas aqui, como mostra a Tabela 2.3. No entanto, a técnica de busca é a mesma e propõe-se um único valor de *c* para cada problema de regressão, independente da definição da matriz *H* (veja Equação 2.3), obtida com a inicialização dos pesos da camada intermediária.

Tabela 2.2 - Valor de *c* obtido para cada conjunto de dados utilizando a metodologia apresentada em Huang *et al.* [15].

Conjunto de Dados	*c*
Pyrim	2^0
Bodyfat	2^7
Housing	2^2
Strike	2^2
Quake	2^0
Space-ga	2^7
Abalone	2^8

Tabela 2.3 - Diferenças nas configurações da ELM de Huang *et al.* [15] e ELM com *c* fixo deste trabalho.

Configurações	Huang *et al.* [15]	ELM com *c* fixo
Nº de Neurônios	1000	100
Pesos	Inicializados aleatoriamente no intervalo [–1;+1]	Inicializados aleatoriamente no intervalo [–0,5;+0,5]
Dados de Entrada	Escalados no intervalo [–1;+1]	Escalados no intervalo [–1;+1]
Dados de Saída	[0;+1]	[0;+1]
Particionamento	$\frac{2}{3}$ para Treinamento e $\frac{1}{3}$ para Validação	60% para Treinamento, 20% para Validação e 20% para Teste

Na busca por seção áurea, descrita na seção anterior, o limiar utilizado é o menor entre 10^{-5} e 0,01% do intervalo de busca inicial, como já mencionado.

2.6.2 Particionamento *holdout* e *k-fold*

Foram realizados testes com os conjuntos de dados particionados de duas formas diferentes: *holdout* e *k-fold*.

No método *holdout*, os dados foram divididos em três grupos: 60% para treinamento, 20% para validação e 20% para teste. O grupo de dados de validação é utilizado na escolha do valor do

coeficiente de regularização *c*, por isso é necessária a utilização do grupo de teste para verificação da performance da rede ao final do treinamento. Para todas as simulações, antes de cada execução, os dados foram reorganizados e divididos novamente em grupos de treinamento, validação e teste, sendo que para cada execução os grupos de dados são iguais para todos os métodos. Portanto, os testes são realizados em 50 partições diferentes.

No caso da aplicação do particionamento *k-fold*, 20% dos dados formam o conjunto de teste e os outros 80% dos dados foram divididos em *k*=10 pastas. A cada apresentação dos dados, uma das pastas faz o papel de conjunto de validação e o restante funciona como conjunto de treinamento, sendo que cada pasta representa o conjunto de validação apenas uma vez. Antes de cada execução, os dados foram reorganizados e divididos novamente em conjunto de teste e nas *k* pastas, resultando em 50 particionamentos distintos, todos utilizados tanto na MLP quanto na ELM com *c* variável.

No caso da ELM com *c* variável, a busca pelo melhor valor de *c* é realizada com base na média do erro quadrático médio das *k* pastas, encontrando o *c* ótimo para o desempenho médio, para o conjunto de validação, de todas as *k* apresentações dos dados, como representado na Figura 2.5. Posteriormente, o desempenho da rede é medido através da aplicação da rede para o conjunto de teste.

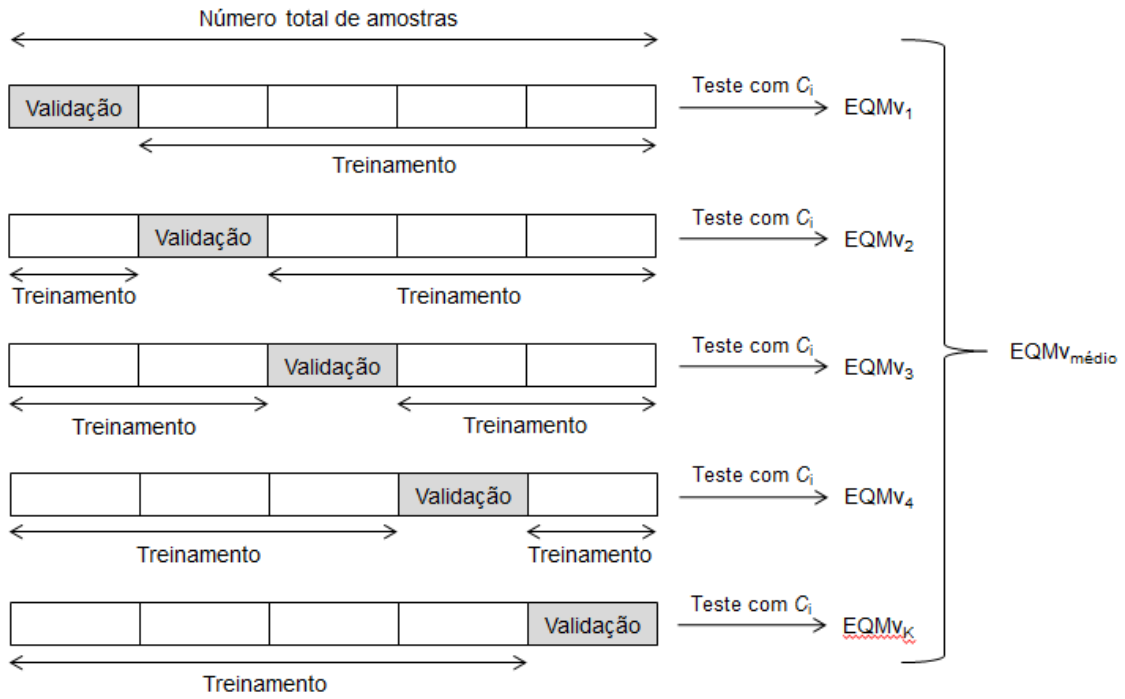

Figura 2.5 - Busca por *c* com particionamento do tipo *k-fold*. EQMv representa o erro quadrático médio junto ao conjunto de validação. O *c* escolhido é o que apresenta melhor desempenho, avaliado através do $EQMv_{médio}$.

Os resultados das simulações com os dois tipos de particionamento são apresentados na próxima seção.

2.6.3 Experimentos e Resultados

Todos os resultados serão apresentados após 50 execuções com os seguintes dados: média e desvio padrão do erro quadrático médio (EQM), e tempo de treinamento.

A Tabela 2.4 mostra os resultados para as simulações da ELM com os valores de *c* fixo apresentados na Tabela 2.2. Este experimento é uma reimplementação da proposta de Huang *et al.* [15] de utilizar um mesmo valor de *c* para todas as execuções de uma tarefa de regressão, independente das configurações de ELM utilizadas. Porém, como mostrado na Tabela 2.2 da seção 2.6.1, as configurações da ELM aqui são diferentes das utilizadas em Huang *et al.* [15]. A Tabela 2.4 também apresenta os resultados para as simulações de ELM com valores de *c* variável e apresenta em negrito os melhores resultados junto aos erros de validação e teste. Na Tabela 2.5, são mostrados os resultados dos testes com *holdout* de MLP, onde são apresentados em negrito os casos em que a MLP superou as ELMs junto aos erros de validação e teste.

Tabela 2.4 - Resultados das Simulações - ELM com *c* Fixo e ELM com *c* Variável - *Holdout*

Conjuntos de Dados	ELM – *c* Fixo					ELM – *c* Variável				
	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo Trein.(s)	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo Trein.(s)
Pyrim	0,1112	0,0372	**0,1152**	0,0394	0,0162	**0,1030**	0,0338	0,1194	0,0426	0,0955
Bodyfat	0,0296	0,0122	0,0340	0,0127	0,0197	**0,0248**	0,0125	**0,0306**	0,0141	0,2346
Housing	0,0857	0,0118	0,0854	0,0136	0,0268	**0,0818**	0,0117	**0,0843**	0,0145	0,3051
Strike	0,2735	0,0119	**0,2735**	0,0118	0,0240	**0,2504**	0,0176	0,2978	0,1283	0,3747
Quake	0,1730	0,0077	**0,1724**	0,0071	0,0636	**0,1725**	0,0077	0,1732	0,0072	0,8964
Space-ga	0,1814	0,1151	0,1756	0,0859	0,0908	**0,1367**	0,0048	**0,1478**	0,0304	1,2787
Abalone	0,0756	0,0027	**0,0770**	0,0029	0,1221	**0,0750**	0,0026	**0,0770**	0,0033	1,7191

Tabela 2.5 - Resultados das Simulações - MLP *Holdout*

Conjuntos de Dados	MLP				
	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo de Trein.(s)
Pyrim	0,1104	0,0321	0,1438	0,0537	3,5131
Bodyfat	**0,0245**	0,0116	0,0321	0,0133	12,8177
Housing	**0,0740**	0,0121	**0,0809**	0,0124	24,8912
Strike	**0,1451**	0,0198	**0,1705**	0,0384	29,8012
Quake	**0,1719**	0,0078	0,1733	0,0070	103,3229
Space-ga	**0,1326**	0,0045	**0,1336**	0,0048	147,3941
Abalone	**0,0748**	0,0026	**0,0766**	0,0025	198,1542

É possível verificar que, apesar do tempo de treinamento ser um pouco maior em relação à ELM com *c* fixo, o ganho na acurácia é significativo junto ao erro de validação para justificar a aplicação da seção áurea na busca local pelo valor mais indicado para *c*. Houve compatibilidade de resultados em todos os casos de estudo, mantendo um custo computacional baixo em relação à rede MLP, e uma performance de generalização competitiva.

No caso do *Strike*, o desempenho da MLP é bastante superior. Através das Figuras 2.6 e 2.7, é possível concluir que esta tarefa de regressão requer que alguns pesos sejam mais elevados para conduzir a uma boa aproximação, saindo do intervalo de valores que restringe a inicialização dos pesos da ELM, o que dificulta a redução dos erros de validação e de teste. A MLP consegue atender a esta demanda, pois os pesos dos neurônios da camada intermediária são ajustáveis. Devido ao caráter aleatório da definição dos pesos da ELM, não é indicado admitir pesos de valor muito elevado.

A Figura 2.6 apresenta a distribuição dos pesos adotados pela MLP para o conjunto de dados *Strike*, contabilizando as 50 execuções. Para comparação, é apresentada na Figura 2.7 a distribuição para o caso *Pyrim*, em que a ELM tem melhor desempenho que a MLP, mostrando que neste caso a tarefa de regressão requer pesos que estão dentro do intervalo de inicialização dos pesos da ELM, tornando essas últimas plenamente capazes de generalizar bem.

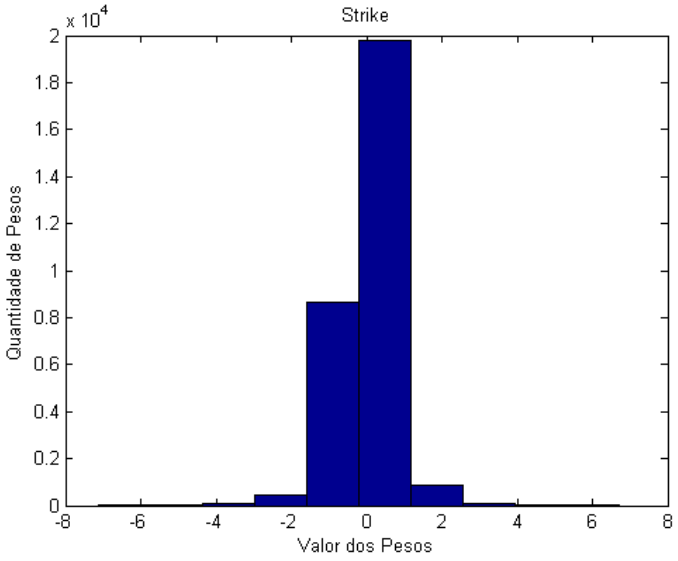

Figura 2.6 - *Strike*: Distribuição dos pesos da MLP

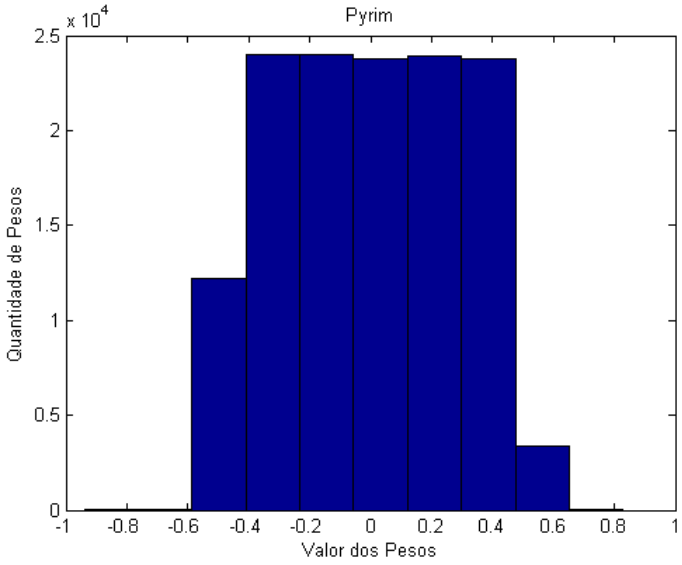

Figura 2.7 - *Pyrim*: Distribuição dos pesos da MLP

Os resultados dos experimentos com particionamento do tipo *k-fold* estão representados na Tabela 2.6. Não há comparações com o apresentado em Huang *et al.* [15], pois seus experimentos são restritos à abordagem *holdout*. Além disso, procurar um valor de *c* através do método de particionamento *k-fold* leva em consideração a variedade na apresentação dos dados, característica

apresentada pela proposta deste trabalho através do método de *c* variável, e que não é preocupação do método apresentado por Huang *et al.* [15], referenciado aqui como ELM com *c* fixo. Por isso, utilizar ELM com *c* fixo com particionamento do tipo *k-fold* faz com que seja perdida a ideia central do *c* fixo, que não leva em conta as peculiaridades da rede.

O tempo de treinamento ao utilizar *k-fold* é maior em relação ao *holdout*, mas os resultados de erro de validação são mais confiáveis e a melhora na acurácia é notória, mesmo em relação à MLP.

Tabela 2.6 - Resultados das Simulações - MLP e ELM com *c* Variável - *K-fold*

Conjuntos de Dados	MLP					ELM – *c* Variável				
	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo Trein.(s)	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo Trein.(s)
Pyrim	0,0872	0,0129	0,1400	0,0298	5,6699	**0,0749**	0,0222	**0,1093**	0,0372	2,3481
Bodyfat	0,0217	0,0055	0,0359	0,0132	19,3082	**0,0187**	0,0076	**0,0318**	0,0151	7,9962
Housing	**0,0671**	0,0029	**0,0784**	0,0111	38,7696	0,0726	0,0102	0,0800	0,0129	16,0558
Strike	**0,1283**	0,0064	**0,1665**	0,0456	31,3439	0,2338	0,0186	0,2783	0,0596	19,8318
Quake	0,1698	0,0017	0,1727	0,0070	117,3793	**0,1656**	0,0074	**0,1724**	0,0070	37,2677
Space-ga	**0,1313**	0,0012	**0,1346**	0,0041	238,0573	0,1330	0,0059	0,1397	0,0058	98,5877
Abalone	0,0743	0,0008	**0,0761**	0,0028	320,0404	**0,0736**	0,0029	0,0764	0,0031	132,5398

Para uma análise mais completa, foi aplicado o teste de significância *Wilcoxon Signed-Rank* [37] com o objetivo de verificar se há diferença estatística na acurácia ao realizar a busca por seção áurea pelo parâmetro *c*, em relação à rede MLP e ELM com *c* fixo. Foi considerado aqui o desempenho junto aos dados de teste. Na Tabela 2.7, é usado o símbolo + quando há ganho significativo com aplicação da busca por seção áurea. Os símbolos ~ e – representam resultados estatisticamente iguais e resultados inferiores, respectivamente. O *p*-valor aparece entre parênteses.

Tabela 2.7 - *Wilcoxon Signed-Rank Test*

	Pyrim	Bodyfat	Housing	Strike	Quake	Space-ga	Abalone
ELM + *Holdout*: *c* Variável *versus c* Fixo	~ (0,6929)	+ ($6{,}355\times10^{-4}$)	~ (0,0794)	~ (0,5606)	– ($1{,}922\times10^{-4}$)	+ ($9{,}715\times10^{-5}$)	~ (0,8567)
Holdout: ELM *c* Variável *versus* MLP	+ (0,0016)	~ (0,1111)	~ (0,0939)	– ($1{,}023\times10^{-9}$)	~ (0,9650)	– ($4{,}126\times10^{-9}$)	~ (0,5873)
k-fold cross-validation: ELM *c* Variável *versus* MLP	+ ($3{,}089\times10^{-8}$)	+ (6,608)	~ (0,1422)	– ($7{,}556\times10^{-10}$)	+ ($1{,}275\times10^{-4}$)	– ($7{,}517\times10^{-10}$)	~ (0,5420)

É possível observar que, em relação à ELM com *c* fixo, os resultados da ELM com *c* variável têm desempenho inferior em apenas um caso. Em comparação com as redes MLP, os resultados são muito parecidos, às vezes perdendo por pouco e até apresentando resultados melhores em alguns experimentos.

As MLPs representam modelos mais flexíveis que as ELMs, pois é possível ajustar os pesos da camada intermediária, além dos pesos da camada de saída. O ajuste desses pesos da camada

intermediária nas MLPs faz com que o espaço de características produzido pela ativação dos neurônios da camada intermediária seja dedicado às demandas da aplicação. Ou seja, a matriz *H*, cujas colunas serão combinadas linearmente para produzir cada uma das saídas da rede neural MLP, não é fornecida a priori e sim moldada durante o processo de minimização do erro de treinamento, com exceção da coluna constante com elementos unitários. Desde que se forneça um número de neurônios suficiente (pode ser maior ou igual ao mínimo necessário) na camada intermediária da MLP, diretamente associado ao número de colunas da matriz *H*, esta maior flexibilidade na definição da matriz *H* pode responder a demandas específicas de cada cenário de aplicação. No entanto, uma possível desvantagem seria um maior risco de sobreajuste do mapeamento produzido, justamente por esta maior flexibilidade para se moldar aos dados de treinamento. Mas como está sendo utilizada a minimização do erro junto a um conjunto de validação para encerrar o ajuste de pesos durante o treinamento das redes MLP, esta desvantagem em potencial é bastante atenuada, o que leva consistentemente a um bom desempenho em termos de generalização com o uso de redes neurais MLP. Portanto, realmente não era esperado que a ELM com *c* variável superasse a rede MLP. Porém, alcançar resultados tão próximos, com a mesma quantidade de neurônios e tempo de treinamento 2 a 3 vezes menor, já representa vantagens para o método proposto neste trabalho.

Com o objetivo de mostrar que uma escolha apropriada de *c* é necessária quando se varia o número de neurônios, foram realizados testes com redes variando de 1 a 100 neurônios, comparando a aplicação da ELM utilizando (*i*) *c* nulo, (*ii*) *c* fixo (os mesmos valores de *c* apresentados na Tabela 2.2), e (*iii*) *c* variável. As Figuras 2.8 a 2.14 mostram a média do erro quadrático junto ao conjunto de validação de 50 execuções com cada arquitetura para cada conjunto de dados, sendo que cada execução utilizou exatamente os mesmos grupos de dados (treinamento, validação e teste) para todos os métodos.

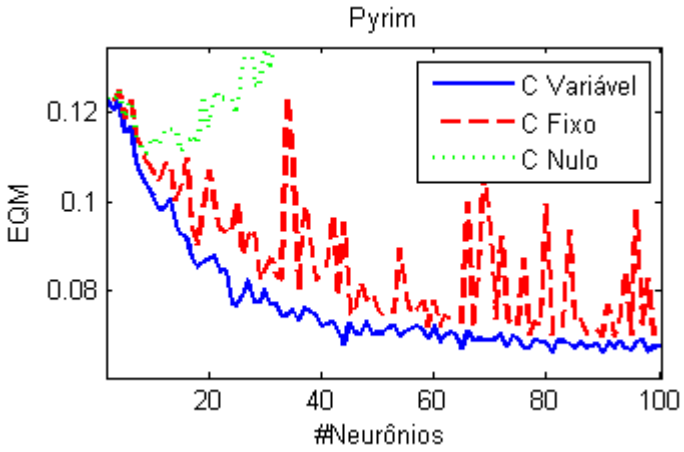

Figura 2.8 - *Pyrim*: Comparação entre *c* Nulo, $c=2^0$ Fixo e *c* Variável.

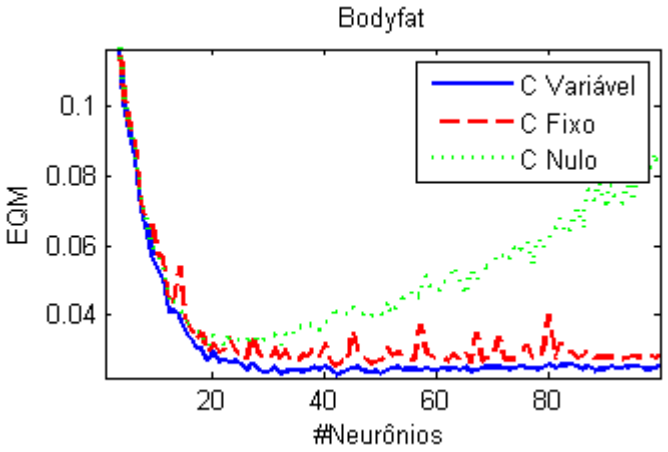

Figura 2.9 - *Bodyfat*: Comparação entre *c* Nulo, $c=2^7$ Fixo e *c* Variável.

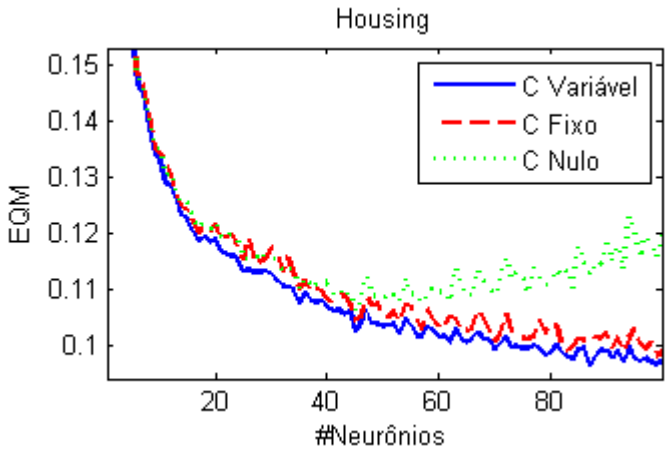

Figura 2.10 - *Housing*: Comparação entre *c* Nulo, $c=2^2$ Fixo e *c* Variável.

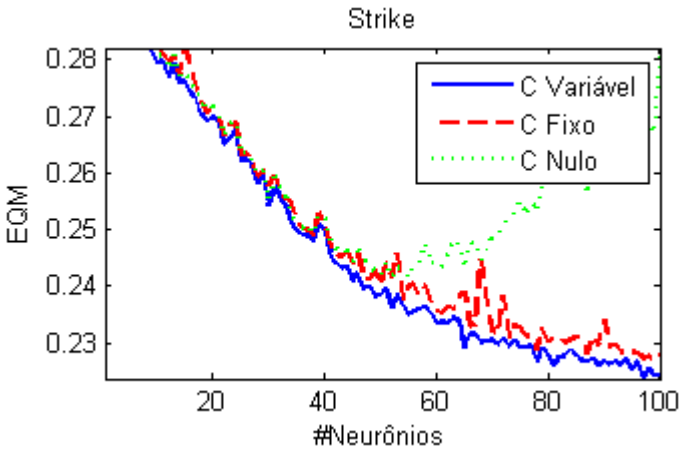

Figura 2.11 - *Strike*: Comparação entre *c* Nulo, $c=2^2$ Fixo, e *c* Variável.

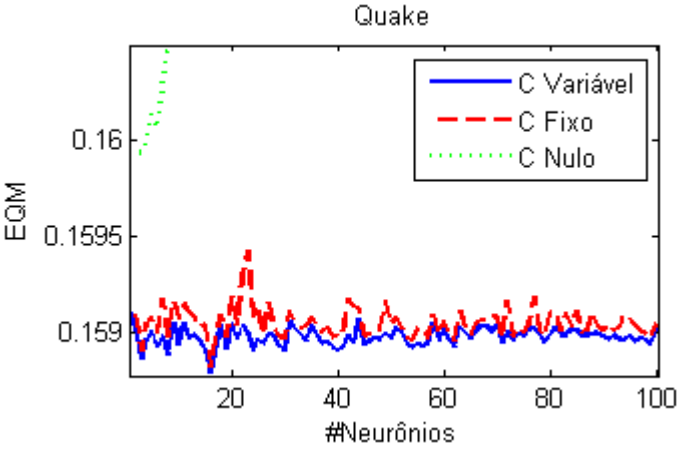

Figura 2.12 - *Quake*: Comparação entre *c* Nulo, $c=2^0$ Fixo e *c* Variável.

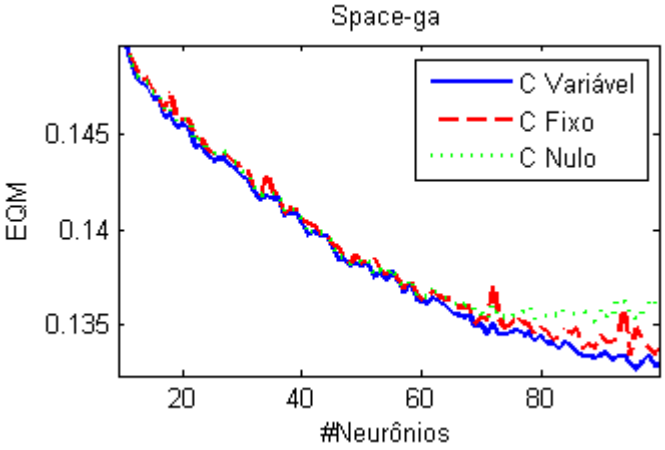

Figura 2.13 - *Space-ga*: Comparação entre *c* Nulo, $c=2^7$ Fixo e *c* Variável.

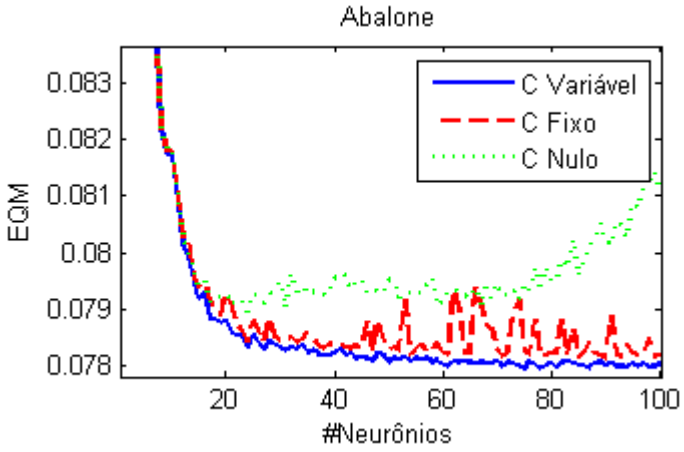

Figura 2.14 - *Abalone*: Comparação entre *c* Nulo, $c=2^8$ Fixo e *c* Variável.

Em todos os experimentos reportados nas Figuras 2.8 a 2.14, a aplicação de ELM com *c* variável tem performance superior às simulações com *c* fixo e nulo, evidenciando a vantagem de buscar um *c* específico para o problema e a forma como ele é apresentado.

Reitera-se aqui que este é o primeiro trabalho que aponta para a necessidade de escolha de um valor específico do coeficiente de regularização *c* para cada ELM aplicada junto a um mesmo problema de regressão. Todos os demais trabalhos da literatura procuram otimizar o valor de *c* para cada problema de regressão. Mesmo fixando-se o número de neurônios da camada intermediária, o problema de regressão e as partições dos dados, inicializações distintas dos pesos levam a matrizes *H* bem diferentes. Logo, o valor de *c* mais indicado para cada matriz *H* vai variar, e este capítulo evidenciou a existência de ganho de desempenho quando este aspecto é devidamente considerado.

Nas Figuras 2.8 a 2.14, para cada quantidade de neurônio na rede, foram realizadas 50 execuções. Para demonstrar a importância da busca pelo valor mais adequado do coeficiente de regularização *c* levando-se em consideração a estrutura da rede ELM¸ a Tabela 2.8 apresenta o valor de *c* selecionado para uma das 50 execuções efetuadas para cada arquitetura, de cada um dos sete conjuntos de dados.

Pode-se perceber que há uma grande variedade nos valores de *c* selecionados para cada execução de um mesmo problema. Essa característica confirma a necessidade de haver a busca individual para cada rede. Foi observado que ainda que o valor de *c* adotado seja alto, o que indica que a solução se preocupa principalmente em minimizar a norma dos pesos, o erro quadrático médio de saída da rede se mantém baixo. Isso é esperado, já que o critério de seleção do valor de *c* entre todos os avaliados está baseado no desempenho junto ao conjunto de validação.

Tabela 2.8 - Valores de *c* selecionados para uma das 50 execuções com cada arquitetura.

#Neur.	*Pyrim*	*Bodyfat*	*Housing*	*Strike*	*Space-ga*	*Quake*	*Abalone*
1	0,228	67108740,000	67108540,000	0,154	67099468,000	67093660,000	0,630
2	0,188	67108740,000	67108817,000	67108540,000	67044460,000	67108340,000	67100315,000
3	0,077	67108740,000	67107493,000	67108540,000	67107493,000	67108740,000	67108464,000
4	0,256	67108799,000	0,876	8,952	0,350	67108493,000	67106646,000
5	0,770	67108718,000	0,336	0,515	1192,840	67108693,000	67108017,000
6	0,124	19,597	0,710	67108788,000	2,265	67108588,000	67106646,000
7	0,300	67108835,000	67108540,000	9,139	67108729,000	42,175	3,674
8	0,310	27,773	1,222	11,528	2,227	67090071,000	11,191
9	0,815	7,429	0,464	67108711,000	0,489	1112,539	41,540
10	0,042	7,134	2,160	55,181	4,376	507,786	9,725
11	0,371	67108835,000	3,143	67108839,000	3,949	74,897	37,305
12	0,083	443,915	67107493,000	13,220	2,561	2181,066	65,913
13	0,181	67108839,000	4,524	10,614	0,218	1784,955	22,838
14	0,059	67108817,000	1,997	67108664,000	0,319	380,826	67106646,000
15	0,107	67107529,000	7,492	1131,581	1,067	67108322,000	67108664,000

16	0,385	16,681	0,898	169,393	1,119	448332,260	30,137
17	0,804	6,744	0,903	509,679	1,929	3151,790	108,447
18	0,059	421,759	0,309	139,270	0,225	554,061	20,314
19	0,228	67108835,000	0,470	125,643	13,196	572,249	196,218
20	0,097	840,181	1,934	17,531	0,078	67108770,000	158,900
21	0,145	25,084	0,519	107,994	7,392	67108740,000	91,551
22	0,062	10,250	3,027	6,942	0,946	2238,594	356,687
23	0,805	3,643	1,117	28,222	0,659	18158,151	11,648
24	0,111	56,444	3,459	321,369	1,109	67108831,000	57,891
25	11,103	65,871	0,878	67108570,000	0,512	67108453,000	371,798
26	0,065	5,401	10,161	709,563	0,279	29437,585	40,950
27	0,149	7,931	3,442	238,667	0,861	76350,614	67108188,000
28	2,641	6,507	0,778	422,982	0,220	67108628,000	764,040
29	2,092	9,599	9,234	67108017,000	0,153	67108347,000	41,879
30	0,138	39,899	16,246	42,841	0,491	87434,329	17,373
31	83,759	17,483	0,248	695,182	0,580	2506,140	9883,074
32	36,311	4,878	0,925	548,443	1,508	15095,545	23,041
33	0,328	7,764	1,404	258,592	0,382	363738,190	21,298
34	0,117	64,053	2,154	48,947	0,372	67108835,000	621,334
35	4,986	3,389	1,009	52,333	0,867	67108689,000	16,976
36	0,282	8,183	1,154	219,916	0,084	8944,254	12,467
37	0,669	9,481	20,021	3749,884	1,854	54838,347	18,350
38	0,085	5,228	0,367	18,239	1,979	21444,325	6,815
39	0,179	7,910	45,136	420,149	0,169	67108536,000	67106682,000
40	0,045	18,326	2,841	3752,405	1,082	141696,440	7,532
41	0,474	17,200	112,603	170,694	0,917	65446,600	680,078
42	0,736	6,296	21,260	7271,088	0,842	130624,330	73,927
43	219,387	3,845	1,897	2397,326	0,171	67107642,000	43,422
44	0,096	2,613	7,254	42,721	2,043	310628,340	18,924
45	0,074	5,760	9,841	38,589	0,255	8098,138	67108071,000
46	0,067	3,902	13,661	5344,214	0,271	420165,220	111,211
47	0,101	14,780	4,454	8022,414	0,178	724695,630	45,147
48	0,083	3,357	4,621	15,536	0,166	252195,130	3307,866
49	0,476	4,962	0,667	131,886	0,460	110549,500	198,368
50	0,177	3,690	4,350	42,040	0,145	439191,370	8,392
51	0,764	11,031	16,800	56,739	2,196	414839,630	305,549
52	0,367	3,258	14,685	9,579	0,359	248100,510	415,624
53	0,056	5,269	193,477	67108657,000	0,359	67108399,000	6,969
54	0,081	2,285	4,781	417,623	0,162	316655,360	377,012
55	0,253	5,083	12,870	220,573	0,294	299160,440	146,886
56	0,698	2,311	1,538	3821,545	0,824	67108046,000	67105846,000
57	0,027	3,149	0,856	10,973	0,244	60404,789	59,453
58	0,069	5,559	0,608	35,745	0,253	67108758,000	5,461
59	0,368	8,386	11,645	42,421	0,182	67097249,000	14,386
60	0,142	71,282	24,901	20,341	0,182	290782,730	17,878
61	263,129	2,479	47,903	145,020	0,271	1837979,600	10,961
62	140,090	3,970	1,555	28,570	0,459	9844500,800	26633,692
63	0,752	8,290	44,623	420,667	0,425	1737386,500	77935,938
64	0,271	2,060	6,050	67108697,000	1,185	64621,333	48,187
65	0,219	3,110	14,794	67107046,000	0,105	67107322,000	104,814
66	139,152	4,065	24,937	676,578	0,141	1702927,400	1000,135
67	41,963	1,704	4,218	17,430	0,615	480614,980	1318,539
68	0,091	3,640	41,936	263,711	0,291	133018,220	8,159

69	0,250	4,226	11,066	827,768	0,190	67108322,000	89,614
70	0,052	7,135	29,050	251,194	0,563	248979,420	32,426
71	0,395	3,193	10,800	449,820	0,579	4974276,000	4,148
72	0,262	2,020	3,016	79,855	0,254	4373400,400	67106664,000
73	29,276	2,395	0,886	20,957	0,223	698761,970	4,721
74	0,553	2,098	4,976	25007,505	0,160	3459432,100	8,402
75	0,152	1,667	3,740	411,024	0,262	153505,990	67103533,000
76	0,047	2,058	48,920	638,931	0,275	195194,280	6,170
77	0,165	1,461	43,571	375,729	0,517	564608,040	78,808
78	0,165	1,798	40,744	399,827	0,192	2384829,700	1078,570
79	0,033	1,934	30,983	53,084	0,212	1467598,800	7,479
80	0,061	1,970	60,850	107009,080	0,075	753647,510	55,158
81	109,965	2,712	13,792	9449,916	0,271	1250101,500	2,463
82	0,286	8,173	37,934	53,098	0,151	1096641,200	13,112
83	3,517	4,155	14,406	5354,006	0,379	7250760,100	355,218
84	0,064	2,331	12,537	2300,207	0,316	1374801,500	87602,204
85	0,059	2,696	15,015	130,141	0,996	66371,551	45,598
86	0,117	2,298	12,679	380174,020	0,372	259890,770	4,688
87	0,025	1,492	37,500	1591,729	0,407	141676,590	8837,390
88	0,029	1,262	13,132	3749,156	0,112	67106668,000	155,210
89	0,191	1,848	40,289	23,106	0,536	1591182,100	55895,495
90	0,198	2,089	101,905	2522,090	0,162	540773,290	280,521
91	10,805	2,215	2,637	438,383	0,245	34521290,000	142,078
92	0,039	1,302	24,834	112224,140	0,388	67107369,000	7555,466
93	368,393	1,625	4,118	27987,212	0,239	1114146,000	42,880
94	0,247	1,461	25,738	6421,230	0,297	7360108,500	483,001
95	0,050	3,185	1,635	4807,447	0,219	2314663,100	16,070
96	0,139	1,307	38,274	2089,116	0,526	1199776,300	2144,555
97	4,211	0,820	71,872	1365,382	0,086	36411570,000	71,366
98	74,750	1,856	21,515	6952,838	0,340	3234617,400	49,549
99	0,180	2,454	1,009	3883,506	0,279	2005686,200	82,907
100	287,651	1,308	28,610	3089,527	0,110	13998867,000	659,534

Mesmo com valores de *c* variando para a mesma tarefa de regressão, há tendências claras de obtenção de valores maiores para *c* junto a alguns problemas, como no caso de *Quake*, quando comparado, por exemplo, a *Pyrim* e *Space-ga*. Isso pode ser um indicativo da presença de mais ruído nos dados, no caso de *Quake*.

Capítulo 3

Elastic Net aplicado a Máquinas de Aprendizado Extremo

Este capítulo descreve a segunda contribuição deste trabalho. Trata-se de um método de poda de neurônios em duas fases. Inicialmente, são apresentados os conceitos de LASSO [35] e *Elastic Net* [39], métodos que substituem o *Ridge Regression* nesta proposta. Além disso, é abordado o método de tratamento do número de condição da matriz $H^{\mathrm{T}}H$, que é aplicado com a finalidade de facilitar o cálculo dos quadrados mínimos, utilizado dentro da solução de LASSO e *Elastic Net*. Na seção 3.3, é descrito o método de busca conhecido como *ParamILS* [17], utilizado na seleção dos parâmetros da *Elastic Net*. Por último, são apresentados os experimentos e resultados obtidos através desta proposta.

3.1 Lasso e *Elastic Net*

Muitos métodos de regressão linear foram desenvolvidos com o objetivo de aproximar modelos regularizados. *Ordinary Least Square* (OLS) e *Ridge Regression* [11] foram abordados no capítulo anterior (veja Equações 2.6, 2.8 e 2.9). Nesta seção, serão apresentadas as formulações para LASSO (*Least Absolute Shrinkage and Selection Operator*) [35] e para *Elastic Net* [39].

Como apresentado na Seção 2.2 deste trabalho, o método *Ridge Regression* penaliza a norma ℓ_2 do vetor $\mathbf{w}_k$. Substituindo a norma ℓ_2 pela norma ℓ_1 do vetor $\mathbf{w}_k$, o LASSO realiza uma seleção de coeficientes de regressão, que na aplicação deste trabalho são representados por ativações de neurônios da camada intermediária, zerando os que têm menor relevância para a tarefa de regressão. A sua formulação é apresentada na Equação 3.1, na forma:

$$\min_{\mathbf{w}_k} \left\| H\mathbf{w}_k - \mathbf{s}_k \right\|_2^2 + c_k \left\| \mathbf{w}_k \right\|_1. \tag{3.1}$$

O LASSO [35] propõe soluções esparsas para c_k suficientemente grande, quando vários elementos do vetor $\mathbf{w}_k$ são levados a zero.

A Figura 3.1, baseada em Flexeder [8], explica no caso bidimensional por que o LASSO tem a capacidade de zerar elementos do vetor $\mathbf{w}_k$. As elipses representam os lugares geométricos dos valores constantes de erro. Os lugares geométricos dos valores constantes da norma ℓ_2 são descritos por uma circunferência (direita), enquanto que para a norma ℓ_1 resulta um quadrado com vértices nos eixos de

coordenadas (esquerda). A solução vai estar num ponto de interseção entre os dois lugares geométricos. Então, o LASSO produz elementos do vetor $\mathbf{w}_k$ iguais a zero quando o ponto de interseção é um vértice do quadrado. Geralmente não ocorre no *Ridge Regression* que o ponto de interseção entre a elipse e a circunferência coincida com algum dos eixos de coordenadas.

Elastic Net [39] surgiu como um método intermediário entre o *Ridge Regression* e o LASSO. Além da regularização e da seleção de variáveis, *Elastic Net* também faz a predição de grupos de variáveis correlacionadas, o que é uma deficiência do LASSO, pois quando há variáveis altamente correlacionadas, este tende a selecionar apenas uma variável entre elas, arbitrariamente [39].

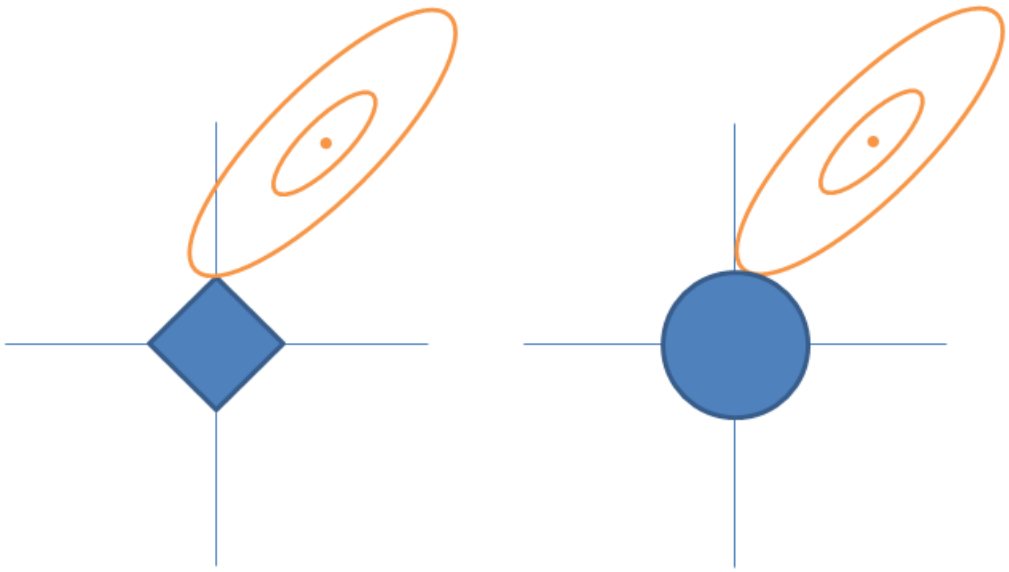

Figura 3.1 - Contornos dos lugares geométricos de erro constante e norma do vetor de pesos constante para LASSO (esquerda) e *Ridge Regression* (direita).

Elastic Net é uma combinação convexa de *Ridge Regression*, com a penalização da norma ℓ_2, e de LASSO, com a penalização da norma ℓ_1, como é possível observar na formulação da Equação 3.2, onde $\alpha_k \in [0,+1]$.

$$\min_{\mathbf{w}_k} \left\| H\mathbf{w}_k - \mathbf{s}_k \right\|_2^2 + c_k \left[(1-\alpha_k) \left\| \mathbf{w}_k \right\|_2^2 + \alpha_k \left\| \mathbf{w}_k \right\|_1 \right]. \tag{3.2}$$

De forma geral, os quatro métodos de solução para o problema de quadrados mínimos, com e sem regularização, estão representados na Equação 3.2:

- OLS: $c_k = 0$;
- *Ridge Regression*: $c_k > 0$ e $\alpha_k = 0$;

- LASSO: $c_k > 0$ e $\alpha_k = 1$;
- *Elastic net*: $c_k > 0$ e $0 < \alpha_k < 1$.

Além da solução através do cálculo da pseudoinversa Moore-Penrose $H^{\dagger}$ apresentada na seção 2.2, há abordagens que lidam com o método de *Ridge Regression* através da decomposição em valores singulares (*Singular Value Decomposition* – SVD) da matriz *H*. A matriz *H* decomposta em valores singulares é representada como na Equação 3.3:

$$H = UDV^T, \tag{3.3}$$

onde $U = [u_1 \quad u_2 \quad \cdots \quad u_N]$ é uma matriz ortogonal $N \times N$; *D* é uma matriz retangular $N \times (n+1)$ com elementos não-negativos na diagonal, denominados de valores singulares, na forma: $d_1 \geq d_2 \geq \ldots \geq d_p \geq 0$, onde $p = N$ se $N \leq (n+1)$ e $p = (n+1)$ se $N \geq (n+1)$; e $V = [v_1 \quad v_2 \quad \cdots \quad v_{n+1}]$ é uma matriz ortogonal $(n+1) \times (n+1)$. A pseudo-inversão da matriz *H* é computacionalmente mais interessante empregando a decomposição da Equação 3.4, pois basta transpor a matriz *D* e substituir os elementos da diagonal por $\frac{d_t}{d_t^2 + c_k}$ ($t \in \{1,\ldots,p\}$), produzindo a matriz $D^{\otimes}$. A pseudoinversa Moore-Penrose $H^{\dagger}$ assume a forma:

$$H^{\dagger} = VD^{\otimes}U^T. \tag{3.4}$$

Diferentemente do método *Ridge Regression*, LASSO não possui solução fechada. Algumas abordagens envolvem técnicas de programação quadrática em otimização convexa. Uma proposta eficiente de aplicação de LASSO é através do algoritmo LARS (*Least Angle Regression)* [7], apresentado no Algoritmo 3.1.

Portanto, originalmente o LARS é um algoritmo para a solução do problema de quadrados mínimos. Para obter a solução de LASSO com ele, inicia-se normalmente o algoritmo; se um coeficiente passa por zero, a respectiva variável é excluída, a melhor direção é recalculada e o processo continua [9].

Baseado no algoritmo LARS, foi proposto em Zou & Hastie [39] o algoritmo LARS-EN que, dado um α fixo, resolve eficientemente o problema da *Elastic Net*. O processo efetuado pelo LARS-EN é resumido a seguir:

- No passo *k*, atualiza-se a fatoração de Cholesky de $H_{\mathrm{A}_{k-1}}^{\mathrm{T}} H_{\mathrm{A}_{k-1}} + \alpha I$, onde A_k é o conjunto ativo no passo *k*.
- Só são registrados os coeficientes diferentes de zero e o conjunto ativo em cada passo do LARS-EN.

Especialmente nos problemas em que $n << N$, a execução do algoritmo é rápida.

Algoritmo 3.1: LARS

Entrada: *H*, resíduo inicial $p = \mathbf{s}_k$

Saída: w_k

1. $w_{k1} = w_{k2} = \ldots = w_{k(n+1)} = 0$.
2. Encontra-se a variável h_{j_1} $\left(j_1 \in \{1,\ldots,(n+1)\}\right)$ mais correlacionada com *p*.
3. Aumenta-se w_{kj} na direção de $\mathrm{sign}\left(\mathrm{corr}\left(p, h_{j_1}\right)\right)$ até que outro competidor h_{j_2} se torne tão correlacionado com o resíduo atual quanto h_{j_1}.
4. Move-se $\left(w_{kj_1}, w_{kj_2}\right)$ (conjunto ativo) no sentido comum dos quadrados mínimos de $\left(h_{j_1}, h_{j_2}\right)$ até que algum outro competidor h_{j_3} se torne tão correlacionado com o resíduo atual quanto o conjunto ativo.
5. Continua-se assim até que todas as variáveis estejam incluídas no processo.
6. Pare quando $\mathrm{corr}\left(p, h_j\right) = 0 \quad \forall j$, ou seja, até que a solução de quadrados mínimos seja alcançada.

Para os experimentos a serem apresentados na seção 3.5, optou-se pelo emprego do programa glmnet, disponível para Matlab e de autoria de Quian *et al.* [27]. Trata-se de um pacote para Matlab contendo procedimentos eficientes para aplicação dos métodos LASSO e *Elastic Net* em regressão linear, logística e regressão multinomial.

3.2 Tratamento da matriz *H*

Após a realização de experimentos, verificou-se um custo computacional muito elevado devido a problemas numéricos relacionados à solução do LASSO e *Elastic Net* nos casos em que a matriz $H^{\mathrm{T}} H$ apresenta características próximas de uma matriz singular.

Por isso, com a intenção de melhorar o número de condição da matriz $H^{\mathrm{T}} H$, é aplicado um tratamento iterativo a ela, considerando o espaço de características gerado pela ativação dos neurônios da camada intermediária, correspondente às colunas de *H*.

O número de condição de uma matriz pode ser obtido como na Equação 3.5:

$$cond(M) = \frac{\sigma_{\max}}{\sigma_{\min}}, \tag{3.5}$$

onde $\sigma_{\max}$ representa o maior valor singular da matriz *M* e $\sigma_{\min}$ representa o menor valor singular da matriz *M*.

Assim, o tratamento da matriz $H^{\mathrm{T}}H$ baseia-se na retirada, uma a uma, das colunas de *H* que, na sua ausência, permitem a máxima redução no valor de $cond(H^T H)$. Esse processo é iterativo e o critério de parada utilizado foi $cond(H^T H)$ ser menor que $1\times10^{+12}$, valor definido após verificação através de testes de que valores acima deste tendem a produzir imprecisão numérica.

Esse processo também pode ser considerado uma forma de poda da ELM, visto que cada retirada de uma coluna de *H* é equivalente a podar um neurônio da camada intermediária da rede. É evidente que supõe-se aqui que $(n+1) \leq N$. Caso ocorresse $(n+1) > N$, o tratamento seria sobre o número de condição da matriz HH^T.

3.3 *ParamILS*

A tarefa de configuração de parâmetros para um algoritmo pode ser muito complexa, e geralmente é tratada como um problema de otimização. Como solução para esse problema, são utilizadas variadas técnicas, como algoritmos evolutivos, de corrida e enumeração exaustiva. O ParamILS [17] é uma abordagem de busca local que permite encontrar o conjunto de parâmetros que satisfaçam variados critérios de otimização.

Seja A o algoritmo cujos parâmetros serão otimizados para uma distribuição D de instâncias do problema; o cruzamento entre o domínio de possíveis valores de cada parâmetro a ser definido representa o espaço de configuração de parâmetros Θ; $A(\theta)$ representa o algoritmo A com a configuração de parâmetros $\theta \in \Theta$. Dessa forma, o objetivo do *ParamILS* é encontrar $\theta \in \Theta$ que resulte no melhor desempenho de $A(\theta)$ na distribuição D.

Esta técnica permite que o usuário defina o conceito de cenário ideal para cada algoritmo A, o que o torna facilmente aplicável para diversos problemas. O único requisito é que seja possível determinar um custo escalar $sc(A,\theta,I)$ para cada execução $A(\theta)$, para uma instância I. Esse custo escalar pode ser representado pelo tempo de execução ou erro aproximado, por exemplo. A distribuição sobre instâncias produz uma distribuição de custo $CD(A,\theta,D)$. A intenção do algoritmo de configuração é minimizar a estatística $c(\theta)$ dessa distribuição de custo, como a média ou mediana por

exemplo. Porém, a distribuição do custo $CD(A,\theta,D)$ é normalmente desconhecida. Pode-se apenas encontrar uma aproximação de $c(\theta)$ baseado em um número limitado de amostras. Denota-se $\hat{c}_N(\theta)$ a aproximação de $c(\theta)$ para N amostras.

ParamILS, apresentado no Algoritmo 3.2, define uma solução inicial padrão e então compara com R soluções aleatórias, adotando uma nova solução apenas quando houver melhor desempenho. O procedimento *Iterative First Improvement* é aplicado como busca local, procurando por soluções melhores na vizinhança, até que seja determinada uma solução que apresenta melhor desempenho que todos os seus vizinhos. Essa solução é comparada a outras soluções selecionadas através de s movimentos aleatórios na vizinhança. Então, o procedimento *Iterative First Improvement* é solicitado novamente. Além disso, com probabilidade p, a busca pode ser reiniciada a partir de uma solução aleatória no espaço de soluções. Esse processo é iterativo e permite ao usuário a definição de um critério de parada mais adequado ao seu problema.

A vizinhança é explorada variando-se um parâmetro de cada vez.

O algoritmo *ParamILS* é capaz de fugir de ótimos locais através da perturbação da posição, ao se movimentar aleatoriamente entre as soluções, e posterior da aplicação de busca local secundária, determinando a aceitação de uma nova solução apenas quando há satisfação de critérios pré-estabelecidos.

Existem duas variações do *ParamILS*, que diferem na construção do procedimento *better*, responsável por fazer a comparação entre duas soluções. No *BasicILS* [17], o procedimento $better(\theta_1,\theta_2)$ compara as aproximações de custo $\hat{c}_N(\theta_1)$ e $\hat{c}_N(\theta_2)$ baseado exatamente em N amostras originárias das respectivas distribuições de custo $CD(A,\theta_1,D)$ e $CD(A,\theta_2,D)$. Para cada configuração θ, os custos são acumulados executando $A(\theta)$ com as mesmas N instâncias e registrando os N custos, que serão reutilizados em futuros cálculos de $\hat{c}_N(\theta)$.

Hutter *et al*. [17] indica dois problemas possíveis ao utilizar a abordagem *BasicILS*. O primeiro foi referenciado como excesso de confiança (do inglês *over-confidence*), que está relacionado com o fato de que o custo estimado para a melhor configuração para o conjunto de treinamento subestima o custo da mesma configuração para o conjunto de teste. O sobretreinamento (do inglês *overfitting*) representa o segundo problema desse método. Isso ocorre quando a configuração se ajusta de forma muito precisa ao conjunto de treinamento, o que resulta em baixo desempenho quando aplicada a amostras não pertencentes a esse conjunto.

Algoritmo 3.2: *ParamILS*

Entrada: Espaço de configuração de parâmetros Θ; relação de vizinhança η, procedimento *better* (que compara $\theta, \theta' \in \Theta$).
Saída: Melhor configuração de parâmetros $\theta \in \Theta$.

```
1  θ0 ← Configuração de parâmetros padrão θ ∈ Θ;
2  para i ← 1...R faça
3      θ ← random θ ∈ Θ;
4      se better(θ, θ0) então θ0 ← θ;
5  θils ← InterativeFirstImprovement(θ, η);
6  enquanto não CritérioParada() faça
7      θ0 ← θils;
       //===Movimentos Aleatórios no Espaço de Solução
8      para i ← 1...s faça θ ← random θ' ∈ η(θ);
       //===Busca Local
9      θ ← InterativeFirstImprovement(θ, η);
       //===Critério de Aceitação
10     se better(θ, θils) então θils ← θ;
11     com probabilidade p faça θils ← random θ ∈ Θ;
12 retorne melhor θ encontrado;
13 Procedimento InterativeFirstImprovement(θ, η)
14 repita
15     θ' ← θ;
16     para cada θ'' ∈ η(θ') em ordem aleatória faça
17         se better(θ'', θ') então θ ← θ''; pare;
18 até θ' = θ;
19 retorne θ;
```

A fim de evitar esses problemas, foi proposta a segunda abordagem do *ParamILS*, o *FocusedILS* [17], que alcança esse objetivo concentrando amostras em configurações de parâmetros promissoras. O procedimento $better(\theta_1, \theta_2)$ é mais complexo nessa versão e está representado na Figura 3.2.

Denota-se $N(\theta)$ a quantidade de amostras utilizadas para estimar a distribuição de custo $CD(A, \theta, D)$. O procedimento $better(\theta_1, \theta_2)$ inicia adquirindo uma amostra para a configuração que possui o menor $N(\theta_i)$, uma para cada em caso de empate. Isso ocorre iterativamente até que uma solução domine a outra. O princípio de dominância é baseado no seguinte: θ_1 domina θ_2 se, e somente se, $N(\theta_1) \geq N(\theta_2)$ e a performance de $A(\theta_1)$ utilizando as amostras $N(\theta_2)$ é superior à performance de $A(\theta_2)$, ou seja, $\hat{c}_{N(\theta_2)}(\theta_1) < \hat{c}_{N(\theta_2)}(\theta_2)$.

Além disso, é armazenado em B o número de amostras adquiridas desde o último passo de melhora. Quando $better(\theta_1, \theta_2)$ retorna verdadeiro, ou seja, θ_1 domina θ_2, é adquirido por θ_1 B amostras bônus de $CD(A, \theta_1, D)$.

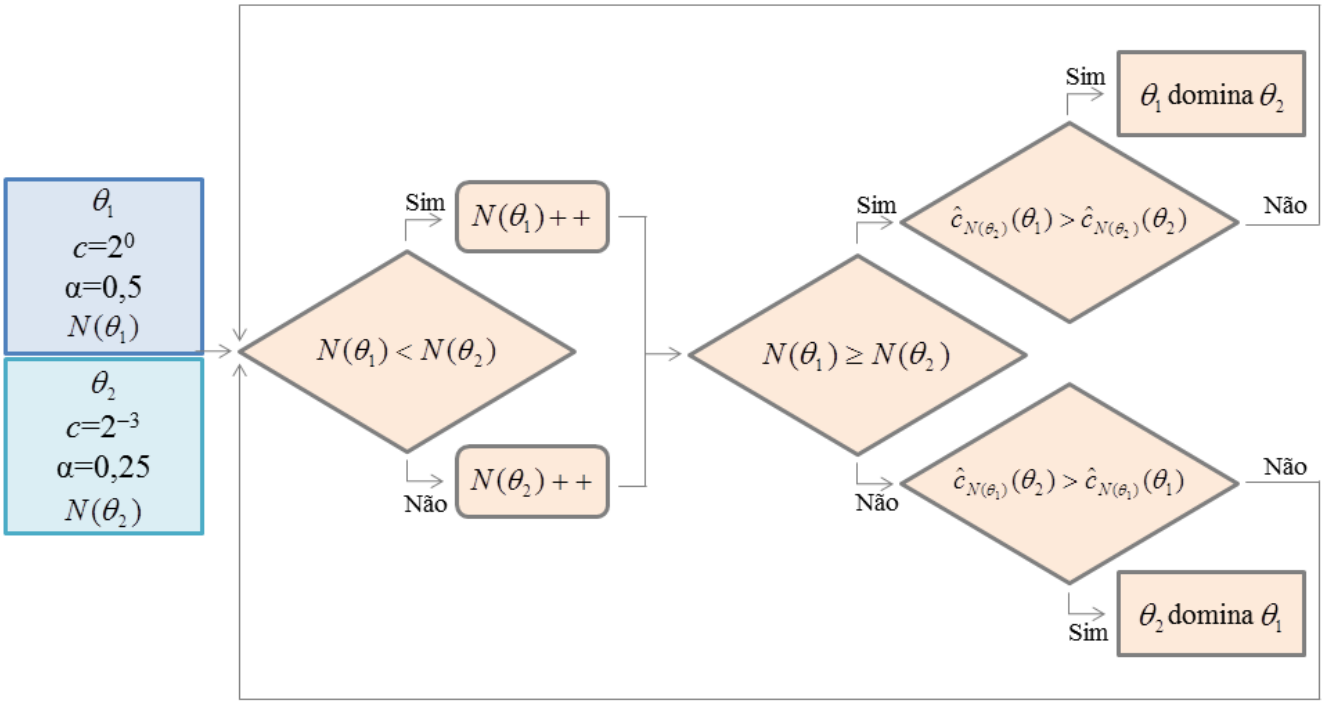

Figura 3.2 - Procedimento $better(\theta_1, \theta_2)$ do algoritmo *ParamILS*.

Esta última, *FocusedILS*, é a abordagem utilizada neste trabalho. É aplicada na busca pelos parâmetros *c* e α do método *Elastic Net*.

3.4 *Elastic Net* aplicada a Máquinas de Aprendizado Extremo

A poda baseia-se na aplicação dos métodos descritos nas seções 3.1, 3.2 e 3.3 para obter uma arquitetura parcimoniosa para a ELM.

Após a realização do cálculo da matriz *H*, foi aplicado o tratamento de número de condição da matriz $H^{\mathrm{T}}H$, de modo a facilitar a inversão necessária durante a aplicação do algoritmo LARS-EN. Além disso, esse processo já permite uma redução significativa no número de neurônios da camada intermediária.

Para a *Elastic Net*, o algoritmo *ParamILS* foi utilizado para determinar os melhores valores para os parâmetros *c* e α, para cada inicialização distinta da ELM. Da mesma forma que na proposta apresentada na seção 2.5, foi mantida a busca pelo coeficiente de regularização e pelo parâmetro α para cada inicialização da rede.

Não são apresentados resultados de simulações com aplicação do LASSO para comparação, pois o LASSO ($c > 0$ e $\alpha = 1$) é uma solução candidata na busca de parâmetros da *Elastic Net*.

Com a capacidade de seleção de variáveis da *Elastic Net*, obtém-se potencialmente uma segunda redução no número de neurônios da camada intermediária, sendo que permanecem os que mais contribuem junto à tarefa de regressão do problema.

3.5 Simulações e Resultados

Nesta seção, são apresentados os experimentos realizados e os resultados obtidos com a proposta da seção 3.4, assim como a comparação com o desempenho do método ELM com regularização via *Ridge Regression*.

3.5.1 Tratamento de Dados e Configuração de Parâmetros

Os conjuntos de dados utilizados para a realização das simulações são os mesmos apresentados na seção 2.6.1 deste trabalho. Porém, diferente do tratamento dos dados descrito naquela seção, aqui foi realizada normalização dos dados de entrada com média zero e variância unitária, como uma exigência do programa glmnet [27] utilizado no cálculo da *Elastic Net*. Os dados de saída foram escalonados no intervalo [0; +1]. Os dados foram particionados através do método *holdout*, sendo 60% para treinamento, 20% para validação e 20% para teste.

As simulações foram realizadas em MATLAB 7.14.0.739 – 64 bits, rodando em Intel Core i5-2430M, 2.4GHz com 4GB de memória RAM.

Serão apresentados resultados para as simulações com *Elastic Net* como proposto na seção 3.4 e, para comparação, foram realizadas simulações com ELM regularizada via *Ridge Regression*. Como há apenas uma saída, o subíndice *k* é desnecessário, sendo que o coeficiente de regularização é denotado por *c*.

Para a *Elastic Net*, a busca por *c* e α foi realizada através do algoritmo *ParamILS*, descrito na seção 3.3. Após a aplicação de uma série de testes variando a condição inicial do *ParamILS*, esta foi definida em $c=2^{-24}$ e α=0,05. Foi possível perceber que valores menores para os dois parâmetros retornam melhor desempenho e, portanto, inicializando a busca com esses valores, a vizinhança passa a ser mais promissora. O espaço de configuração de parâmetros utilizado é o produto cartesiano entre o conjunto de soluções para o parâmetro *c*, representado por $\{2^{-24},2^{-23},...,2^{+24},2^{+25}\}$, e o conjunto de soluções para o parâmetro α, representado por $\{0;0,05;...;0,95;1\}$. Os parâmetros do algoritmo *ParamILS* foram configurados como sugerido em Hutter *et al*. [17]: $R = 10; s = 3; p = 0,01$. O critério

de parada determina o fim da busca ao encontrar três vezes seguidas a mesma configuração de parâmetros como melhor solução.

Após a execução de vários testes, foi observado que, para variadas configurações dos parâmetros *c* e α, a *Elastic Net* retorna todos os pesos da camada de saída zerados. Para ilustrar esse problema, as Figuras 3.3 e 3.4 apresentam o grid de soluções candidatas para *Pyrim* e *Abalone*, marcando em vermelho as soluções que zeram todos os pesos da camada de saída. É possível verificar que valores mais altos do coeficiente de regularização *c* causam esse problema, independente do valor de α, com exceção do caso em que α=0, que representa a solução do *Ridge Regression* e, por isso, não zera nenhum peso. Para evitar esse tipo de solução, no procedimento $better(\theta_1, \theta_2)$, foi acrescentado um critério de descarte de soluções quando essas retornarem todos os coeficientes zerados.

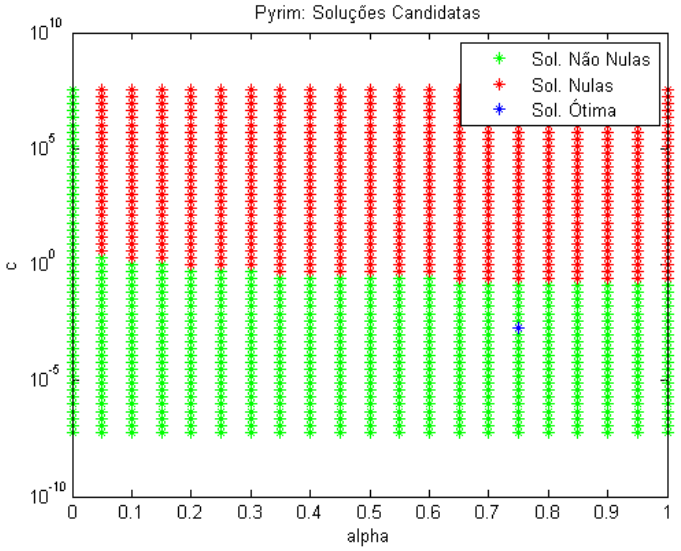

Figura 3.3 - *Pyrim*: Soluções Candidatas do *ParamILS*

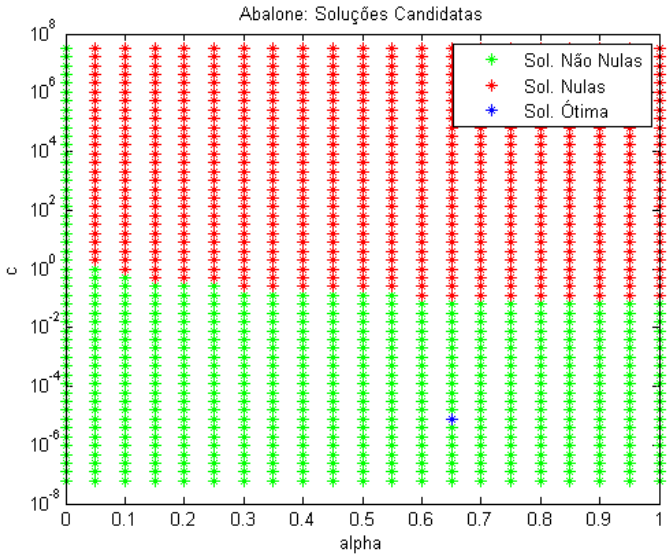

Figura 3.4 - *Abalone*: Soluções Candidatas do *ParamILS*

Para a definição do coeficiente de regularização *c* do *Ridge Regression*, em cada execução, são avaliados 50 valores de *c* no conjunto $\{2^{-24}, 2^{-23}, ..., 2^{+24}, 2^{+25}\}$ como proposto em Huang *et al*. [15], e é selecionado o valor que obteve o menor erro quadrático médio de saída da rede junto ao conjunto de validação.

Tanto para *Elastic Net* quanto para *Ridge Regression*, os parâmetros foram selecionados especificamente para cada execução com inicialização de pesos distinta.

Para os dois métodos, as ELMs iniciam com 100 neurônios na camada intermediária, e os pesos foram inicializados aleatoriamente, com distribuição uniforme no intervalo [–0,5; +0,5]. A função de ativação utilizada foi a tangente hiperbólica.

3.5.2 Experimentos e Resultados

Antes de apresentar os resultados dos experimentos com a proposta da Seção 3.4, é importante expressar a necessidade da seleção adequada dos parâmetros *c* e α, levando em consideração os pesos da camada intermediária e o conjunto de dados de treinamento. São exibidos na Tabela 3.1 a solução ótima encontrada via busca exaustiva para 50 execuções com redes ELM distintas para o *Pyrim*, mas todas partindo com 100 neurônios na camada intermediária. As redes ELMs são distintas, portanto, pelo fato dos pesos associados aos neurônios da camada intermediária serem distintos. Cabe salientar que a busca exaustiva tem um custo bastante elevado, pois são testadas todas as combinações possíveis de valores para o par [*c*, α].

Tabela 3.1 - Soluções ótimas [*c*,α] para 50 execuções do *Pyrim*.

Execução	*c*	α	Execução	*c*	α
1	0.00781250	0.75	26	0.00195313	0.85
2	0.06250000	0.55	27	0.12500000	0.05
3	0.00195313	0.80	28	0.03125000	0.75
4	0.00390625	0.80	29	0.25000000	0.35
5	0.00390625	0.65	30	0.03125000	0.55
6	0.01562500	0.20	31	0.00024414	0.20
7	0.00195313	0.80	32	0.00048828	0.65
8	0.00097656	0.25	33	0.06250000	0.30
9	0.00781250	0.25	34	0.03125000	0.90
10	0.06250000	0.25	35	0.00097656	0.25
11	0.00000381	0.05	36	0.00390625	0.50
12	0.03125000	0.65	37	0.00024414	0.75
13	0.00781250	0.20	38	0.03125000	0.10
14	0.01562500	0.05	39	0.03125000	0.15
15	0.00781250	0.35	40	0.06250000	0.60
16	0.06250000	0.95	41	0.00003052	0.90
17	0.06250000	0.05	42	0.01562500	0.35
18	0.06250000	0.75	43	0.03125000	0.70
19	0.06250000	0.50	44	0.01562500	0.65
20	0.00390625	0.60	45	0.00024414	0.20
21	0.25000000	0.45	46	0.12500000	0.60
22	0.00000763	0.10	47	0.00390625	0.10
23	0.01562500	0.50	48	0.00097656	0.80
24	0.01562500	0.60	49	0.00048828	0.90
25	1.00000000	0.10	50	0.00390625	0.75

Fica evidente na Tabela 3.1 a necessidade de efetuar uma busca pelos parâmetros *c* e α adequados a cada estrutura de ELM. Tanto os valores de *c* quanto os valores de α variam significativamente de uma execução para a outra. O que explica isso são as diferenças contidas nos

pesos da camada intermediária da rede, que também levam a estratégias distintas de poda de neurônios e nos conjuntos de treinamentos aplicados a cada execução.

Todos os resultados da proposta deste trabalho serão apresentados após 50 execuções com os seguintes dados: média e desvio padrão do erro quadrático médio (EQM) e tempo de treinamento. Para os testes com *Elastic Net*, é apresentada a média do número de neurônios resultantes de cada uma das duas fases de poda: tratamento da matriz *H* (fase 1) e seleção de variáveis da *Elastic Net* (fase 2).

A Tabela 3.2 apresenta os resultados para as simulações da ELM regularizada via *Ridge Regression*. Na Tabela 3.3, são mostrados os resultados dos testes com o método de poda para ELM apresentado na seção 3.4 deste trabalho.

Tabela 3.2 - Resultados das Simulações - ELM regularizado via *Ridge Regression* - 100 neurônios

Conjuntos de Dados	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo de Treinamento (s)
Pyrim	0,1277	0,0266	**0,1777**	0,0829	0,0480
Bodyfat	0,0225	0,0148	**0,0253**	0,0157	0,1273
Housing	0,0778	0,0103	**0,0814**	0,0121	0,2100
Strike	0,2560	0,0157	0,3082	0,1030	0,2702
Quake	0,1726	0,0082	0,1732	0,0074	0,6340
Space-ga	0,1453	0,0049	**0,1479**	0,0080	0,8833
Abalone	0,0770	0,0034	0,0908	0,0319	1,2942

Tabela 3.3 - Resultados das Simulações - ELM regularizado via *Elastic Net* - inicializado com 100 neurônios

Conjuntos de Dados	Neurônios Fase 1	Neurônios Fase 2	EQM Validação	Desvio Padrão Validação	EQM Teste	Desvio Padrão Teste	Tempo Trein.(s)
Pyrim	43	22	0,1643	0,0463	0,1888	0,0793	1,0895
Bodyfat	100	67	0,0256	0,0140	0,0284	0,0159	23,1218
Housing	100	89	0,0856	0,0134	0,0851	0,0147	256,6731
Strike	46	37	0,2821	0,0209	**0,3037**	0,0567	118,5149
Quake	9	6	0,1729	0,0081	**0,1731**	0,0072	22,5786
Space-ga	18	18	0,1509	0,0041	0,1514	0,0077	241,7348
Abalone	26	24	0,0796	0,0043	**0,0775**	0,0033	437,2505

Já na primeira fase de poda, é possível observar uma redução bastante significativa do número de neurônios na camada intermediária na maioria dos problemas. Mesmo nos casos como o *Bodyfat* e *Housing*, onde o tratamento de *H* não podou nenhum neurônio devido ao bom número de condição da matriz $H^{\mathrm{T}}H$, a *Elastic Net* faz a função de reduzir o número de neurônios, mantendo apenas os mais significativos para a tarefa de regressão. Além disso, podem ocorrer casos como o *Space-ga*, onde a segunda fase mantém todos os neurônios do sistema, já que as soluções possíveis na busca pelos parâmetros da *Elastic Net* através do *ParamILS* inclui $\alpha=0$, que representa a solução do *Ridge*

Regression e não possui capacidade de seleção de variáveis, utilizando todos os neurônios da camada intermediária da rede. Mesmo assim, pode-se verificar uma estrutura bastante enxuta para o *Space-ga*, alcançada devido à utilização do tratamento do número de condição da matriz $H^{\mathrm{T}}H$. Dessa forma, aliar as duas fases de poda dessa proposta permite obter estruturas mais parcimoniosas.

Para uma análise mais completa, também foi aplicado o teste de significância *Wilcoxon Signed-Rank* [37] para evidenciar se há diferenças estatísticas na acurácia ao utilizar o método de poda proposto neste trabalho em relação à ELM regularizada via *Ridge Regression*. Na Tabela 3.4, é apresentado o símbolo + quando há ganho significativo com a aplicação do tratamento de *H* aliado à *Elastic Net*. Os símbolos ~ e – representam resultados estatisticamente iguais e resultados inferiores, respectivamente. O *p*-valor aparece entre parênteses.

É possível observar que os resultados obtidos pelos dois métodos comparados na Tabela 3.4 não possuem diferenças estatísticas significativas, com exceção do *Space-ga*. Isso significa que o método proposto neste trabalho permite obter resultados muito próximos aos apresentados por uma ELM com 100 neurônios, porém com uma estrutura bem mais enxuta.

Tabela 3.4 - *Wilcoxon Signed-Rank Test*

Conjunto de Dados	ELM + *Elastic Net* *versus* ELM + *Ridge Regression*
Pyrim	~ (0,1122)
Bodyfat	~ (0,0655)
Housing	~ (0,0517)
Strike	~ (0,0733)
Quake	~ (0,0962)
Space-ga	– ($2{,}1610 \times 10^{-5}$)
Abalone	~ (0.4091)

Além da poda de neurônios, o tratamento aplicado à matriz *H* reduz o tempo de execução da *Elastic Net*, que também possui função dupla: efetuar a penalização da norma dos pesos da camada de saída ao calculá-los e realizar a seleção de variáveis, mantendo os neurônios mais significativos à tarefa de regressão na camada intermediária. Embora seja evidente a obtenção de arquiteturas de redes

neurais com um número bem reduzido de neurônios na camada intermediária, o custo total de treinamento sofreu um acréscimo expressivo em termos de tempo de execução.

Foi destacada como uma das principais motivações para o emprego de ELMs a ausência de necessidade de se ajustar os pesos da camada intermediária. De fato, o treinamento das ELMs se apresenta como um problema de regressão linear, o que implica em baixo custo computacional, quando comparado com o problema de regressão não-linear associado ao ajuste de pesos das redes neurais MLP. Ficou evidente, no entanto, que as contribuições desta pesquisa, visando obter melhores resultados de regularização, produziram impactos significativos no custo computacional do processo de treinamento supervisionado das ELMs. Como os custos de treinamento já existentes eram baixos, esse aumento de custo produzido pelas contribuições da pesquisa não inviabiliza a abordagem regularizada, mas torna o treinamento aproximadamente tão custoso quanto no caso de uma MLP de dimensões equivalentes. Por outro lado, as duas iniciativas que contribuem para a eliminação de neurônios pouco representativos na camada intermediária, mais especificamente o tratamento do número de condição da matriz $H^T H$ e o emprego de LASSO e Elastic net, conduzem a ELMs com arquiteturas finais bastante parcimoniosas (em termos do número efetivo de neurônios na camada intermediária), o que reduz muito o custo de implementação dessas redes já treinadas. Supondo aplicações em que as ELMs já treinadas sejam produzidas em escala ou em que haja restrições de recursos computacionais, a obtenção de arquiteturas mais enxutas e com desempenho ainda competitivo mostra-se relevante.

Capítulo 4

Conclusões

Este capítulo apresenta uma análise das principais contribuições alcançadas com a pesquisa e aponta as principais perspectivas futuras de atuação.

4.1 Busca de *c* por Seção Áurea

Ao apresentar esta metodologia de busca pelo parâmetro de regularização, foi mantido como objetivo não perder as principais características apresentadas pelas redes ELMs, que as diferenciam de outras arquiteturas e as fazem atrativas tanto para aplicações de regressão quanto de classificação.

Considerando que o custo computacional de treinamento é um dos principais atributos das redes ELMs, é sensato propor uma metodologia mais exigente, mas que mantenha o custo computacional em níveis baixos. Foi demonstrado, através da comparação desta proposta com redes MLP, que esta característica foi mantida.

O fato de as redes ELM poderem utilizar variados tipos de função de ativação também é muito explorado. Este trabalho aplica exclusivamente funções do tipo tangente hiperbólica, porém esta abordagem pode ser estendida para redes com funções de ativação variadas. Além disso, a flexibilidade das redes ELM em relação à quantidade de neurônios e pesos da camada intermediária também foi levada em conta nos experimentos.

Foi acrescentada à formulação original da ELM a busca por seção áurea para escolha refinada do parâmetro de regularização ao definir os pesos da camada de saída da rede, de forma a alcançar um avanço no desempenho, sem sacrificar os aspectos mais relevantes dessas redes.

A principal contribuição nesta parte da pesquisa foi mostrar que não é adequado empregar um mesmo valor do coeficiente de regularização c para diferentes configurações de redes neurais, dada certa tarefa de aprendizado. Esse procedimento de buscar um valor fixo para c junto a cada problema de regressão é padrão na literatura e deve ser revisto, pois mostrou-se aqui que a melhor escolha de c depende da matriz H. Mesmo que os dados de treinamento sejam os mesmos, a matriz H varia com o número de neurônios e com cada inicialização aleatória dos pesos da camada intermediária.

4.2 *Elastic Net* aplicada a Máquinas de Aprendizado Extremo

A principal contribuição desta proposta está relacionada à definição apropriada da estrutura da ELM. Através de duas fases de poda dos neurônios da camada intermediária, é possível estabelecer uma estrutura reduzida que possui a mesma capacidade de generalização que a estrutura inicial com um número elevado de neurônios.

A primeira fase, representada pelo tratamento do número de condição de $H^{\mathrm{T}}H$, retirando-se colunas da matriz *H*, além de provocar a redução no número de neurônios, também permite que a aplicação posterior da *Elastic Net* seja facilitada, já que matrizes próximas a singulares podem causar problemas no cálculo dos quadrados mínimos que faz parte dos algoritmos LARS-EN e glmnet.

A *Elastic Net* com sua capacidade de seleção de variáveis, ao mesmo tempo que realiza a segunda fase de poda, obtém os pesos da camada de saída, concluindo a fase de treinamento destas redes.

Como diferencial em relação a outros trabalhos que utilizam ELMs aliadas à *Elastic Net*, este trabalho apresenta a importância de selecionar os parâmetros adequados para a rede, levando em consideração a variedade na arquitetura, totalmente dependente do conjunto de pesos da camada intermediária, da quantidade de neurônios e dos dados utilizados no treinamento.

Além disso, é apresentada aqui uma solução na busca destes parâmetros na forma do algoritmo *ParamILS*, que permite efetuar a busca simultânea pelos dois parâmetros *c* e α, sem a necessidade de recorrer a uma busca exaustiva.

Portanto, pode-se concluir que a contribuição deste trabalho tem como ponto fundamental a obtenção de redes ELM ao mesmo tempo mais enxutas e robustas, com capacidade de generalização equivalente à apresentada por redes mais complexas.

4.3 Perspectivas futuras

Como próximos passos da pesquisa, destacam-se:

- Extensão da metodologia para o tratamento de problemas de classificação;
- Consideração de mais camadas intermediárias definidas aleatoriamente, no sentido de *deep learning*, como apresentado em Ribeiro & Noel [29];
- Extensão da metodologia para o tratamento de ELMs construtivas, na linha da abordagem de Huang *et al.* [13].

Referências

[1] Bartlett, P.L., The sample complexity of the pattern classification with neural networks: The size of the weights is more important than the size of the network. IEEE Transactions on Information Theory, vol. 44, no. 2, pp. 525-536, 1998.

[2] Bengio, Y. Learning deep architectures for AI, Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009.

[3] Blake, C.L., Merz, J.C. UCI Repository of Machine Learning Databases, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, 1998 [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[4] Boyer, C. B., Merzbach, U. C. A History of Mathematics. Wiley, 3rd edition, 2011.

[5] Broomhead, D. S., Lowe D. Multivariate functional interpolation and adaptive networks. Complex Systems, vol. 2, pp. 321-355, 1988.

[6] Cambria, E., Huang, G. B., Kasun, L. L. C., Zhou, H., Vong, C. M., Lin, J., Yin, J., Cai, Z., Liu, Q., Li, K., Leung, V. C. M., Feng, L., Ong, Y.-S., Lim, M.-H., Akusok, A., Lendasse, A., Corona, F., Nian, R., Miche, Y., Gastaldo, P., Zunino, R.., Decherchi, S., Yang, X., Mao, K., Oh, B.-S., Jeon, J., Toh, K.-A., Teoh, A. B. J., Kim, J., Yu, H., Chen, Y., Liu, J. Extreme Learning Machines. IEEE Intelligent Systems, vol. 28, no. 6, pp. 30-59, 2013.

[7] Efron B., Hastie T., Johnstone I., Tibshirani R. Least angle regression. The Annals of Statistics, vol. 32, pp. 407-499, 2004.

[8] Flexeder, C. Generalized LASSO Regularization for Regression Models. Ph. D. Thesis, Ludwig-Maximilians-Universidade de Munique, 2010.

[9] Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2nd edition, 2009.

[10] Haykin, S. Neural Networks and Learning Machines. Prentice-Hall, 3rd Edition, 2008.

[11] Hoerl, A.E., Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. Technometrics, vol. 12, pp. 55-67, 1970.

[12] Huang, G.-B., Chen, L. Enhanced random search based incremental extreme learning machine. Neurocomputing, vol. 71, pp. 3460-3468, 2008.

[13] Huang, G.-B., Chen, L., Siew, C.-K. Universal Approximation Using Incremental Constructive Feedfoward Networks with Random Hidden Nodes. IEEE Transactions on Neural Networks, vol. 17, no. 4, pp. 879-892, 2006.

[14] Huang, G.-B., Wang, D.H., Lan, Y. Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics, vol. 2, pp. 107-122, 2011.

[15] Huang, G.-B., Zhou, H., Ding, X., Zhang, R. Extreme Learning Machines for Regression and Multiclass Classification. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics, vol. 42, no. 2, pp. 513-529, 2012.

[16] Huang, G.-B., Zhu, Q.-Y., Siew, C.-K. Extreme learning machine: a new learning scheme of feedforward neural networks. Proceedings of the International Joint Conference on Neural Networks (IJCNN'2004), vol. 2, pp. 985-990, 2004.

[17] Hutter, F., Hoos, H. H., Leyton-Brown, K. and Stüetzle, T. ParamILS: An Automatic Algorithm Configuration Framework. Journal of Artificial Intelligence Research, vol. 36, pp. 267-306, 2009.

[18] Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. International Joint Conference on Artificial Intelligence. vol. 14, pp. 1137–1145, 1995.

[19] Kulaif, A.C.P, Von Zuben, F.J. Improved regularization in extreme learning machines. 11th Brazilian Congress on Computational Intelligence, pp 1-6, 2013.

[20] Lima, C.A.M., Coelho, A.L.V., Von Zuben, F.J. Hybridizing mixtures of experts with support vector machines: Investigation into nonlinear dynamic systems identification. Information Sciences, vol. 177, pp. 2049-2074, 2007.

[21] Luenberger, D. G., Ye, Y. Linear and nonlinear programming. Springer, 3rd edition, 2008.

[22] Martínez-Martínez, J. M., Escandell-Montero, P., Soria-Olivas, E., Martín-Guerrero, J. D., Magdalena-Benedito, R., Gómez-Sanchis, J. Regularized extreme learning machine for regression problems. Neurocomputing, vol. 74, pp. 3716-3721, 2011.

[23] Miche, Y., Heeswijk, M.V., Bas, P., Simula, O. and Lendasse, A. TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization. Neurocomputing, vol. 74, pp. 2413-2421, 2011.

[24] Miche, Y., Sorjamaa, A., Bas, P., Simula, O., Jutten, C., Lendasse, A. OP-ELM: optimally pruned extreme learning machine. IEEE Transactions on Neural Networks, vol. 21, no. 1, pp. 158-162, 2010.

[25] Mike, M. Statistical Datasets, Dept. Statist., Univ. Carnegie Mellon, 1989. [Online]. Available: http//lib.stat.cmu.edu/datasets/

[26] Møller, M. F. A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks, vol. 6, pp. 525-533, 1993.

[27] Qian, J., Hastie, T., Friedman, J., Tibshirani, R. and Simon, N. Glmnet for Matlab, 2013. Available: http://www.stanford.edu/~hastie/glmnet_matlab/

[28] Rao, C.R., Mitra, S.K. Generalized inverse of matrices and its applications. Wiley, 1971.

[29] Ribeiro, B., Noel, L. Extreme Learning Classifier with Deep Concepts. Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. Springer, pp. 182-189, 2013.

[30] Rumelhart, D. E.; McClelland, J.L. Parallel distributed processing: Exploration in the microstructure of cognition. The MIT Press, vol. 1, 1986.

[31] Schölkopf, B., Smola, A.J. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, 2001.

[32] Serre, D. Matrices: theory and applications. Springer, 2002.

[33] Similä, T. and Tikka, J. Multiresponse sparse regression with application to multidimensional scaling. Proceedings of the 15th international conference on Artificial neural networks: formal models and their applications - Volume Part II, ICANN'05, Springer-Verlag, pp. 97-102, 2005.

[34] Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J. Least Squares Support Vector Machines, World Scientific Publishers, 2002.

[35] Tibshirani, R., Regression shrinkage and selection via the LASSO. Journal of the Royal Statistical Society B, vol. 58, pp. 267–288, 1996.

[36] Vapnik V.N. The Nature of Statistical Learning Theory. Springer, 2nd edition, 1999.

[37] Wilcoxon, F. Individual comparisons by ranking methods. Breakthroughs in Statistics. Springer, pp. 196-202, 1992.

[38] Yang, Y., Wang, Y., Yuan, X. Bidirectional Extreme Learning Machine for Regression Problem and Its Learning Effectiveness, IEEE Transactions on Neural Networks, vol. 23, no. 9, pp. 1498-1505, 2012.

[39] Zou, H., Hastie, T. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society B, vol. 67, pp. 301–320, 2005.