Avaliação de Classificadores utilizando Técnicas de Estimativa de Densidades

Victor São Paulo Ruela
Programa de Pós-Graduação em Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
Email: victorspruela@ufmg.br

***Resumo*— A modelagem de dados não-lineares com redes neurais artificiais depende da qualidade projeção aplicada sobre as entradas, geralmente feita através de funções de kernel. A otimização de seus parâmetros é uma etapa importante e pode ser feita via técnicas de estimativa de densidade. Além de fornecer uma forma automática de seleção dos parâmetros ótimos, estas técnicas possuem a tendência de gerar projeções ortogonais das entradas no espaço de projetado. A partir desta observação, este trabalho tem como objetivo avaliar o desempenho de classificadores lineares sobre esta projeção, considerando problemas de benchmark presentes na literatura. Considerando 15 bases de dados de benchmark, os modelos ELM, Hebbiano e Perceptron sobre a projeção do espaço de verossimilhanças e SVM/RBF com otimização de largura foram comparados estatisticamente. Os resultados do experimento mostraram que, para um intervalo de confiança de 95%, o uso de otimização de largura para os modelos SVM e RBF é bastante promisor, obtendo resultados estatisticamente equivalentes para a maioria das bases de dados. Além disso, o modelo Hebbiano apresentou resultados abaixo do esperado, repetindo o que foi observado no trabalho anterior. Uma breve análise é feita sobre as projeções e característica dos datasets, onde se observou uma maior tendência do Hebbiano ter piores resultados quando há um desbalanceamento entre as classes.**

I. Introdução

O desempenho de redes neurais artificiais sobre dados não-lineares é altamente dependente da projeção aplicada sobre as entradas, geralmente feita através de kernels. Estas funções possuem diversos parâmetros a serem ajustados, os quais irão afetar diretamente a qualidade do modelo obtido. O ajuste de seus paramêtros é geralmente realizado via técnicas de busca exaustiva, como validação cruzada [1]. Embora amplamente utilizadas, tais técnicas não utilizam informações presentes nos dados. Isso motivou o desenvolvimento de, por exemplo, técnicas baseadas em estimativa de densidade para analisar a estrutura dos dados e reduzir a necessidade de interação do usuário [2].

Baseado no KDE (*Kernel Density Estimation*) [3], esta abordagem exploram o comportamento da projeção das funções de similaridade calculadas sobre kernel escolhido. É determinada uma função sobre os parâmetros do kernel, a qual pode ser utilizada para determinar os parâmetros que melhor se ajustam aos dados. É possível, por exemplo, usar este conceito para a determinação da largura ótima de kernels radiais (RBF) para o algoritmo SVM [2]. Um resultado importante desta técnica está no fato dela poder gerar projeções ortogonais no espaço de verossimilhanças. Isso sugere a possibilidade de utilizar modelos lineares sobre esta projeção, como o Perceptron e aprendizado Hebbiano.

Este trabalho tem como objetivo avaliar o desempenho dos modelos Perceptron simples [4] e Hebbiano [5] sobre as projeções no espaço de verossimilhanças. Serão considerados tanto os kernels gaussiano e perceptron de múltiplas camadas (MLP), disponibilizados pelo professor. Além destes modelos, serão considerados também o ELM com regularização [6], SVM [7], [2] e RBF com otimização de largura, aplicados sobre o espaço das entradas. Este último modelo não possui publicação associada e é proposto como forma de extensão ao enunciado original do trabalho. Um experimento será desenhado para compará-los estatisticamente sobre diferentes bases de dados de *benchmark* disponíveis na literatura.

II. Revisão da literatura

A. Métodos baseados em Kernel

Em geral, a solução de um problema de classificação consiste em encontrar um discriminador linear $g(\mathbf{x})$, $\mathbf{x} = x_1, x_2, \ldots, x_d$, definido como:

$$g(\mathbf{x}) = w_0 + \sum_{i=1}^{d} w_i x_i \tag{1}$$

onde $\mathbf{w}$ é um vetor de pesos deste hiper-plano. Entretanto, caso o problema não seja linearmente separável, não é possível encontrar um vetor $\mathbf{w}$ que consegue separar adequadamente todas as amostras. Para lidar com este problema, foram desevolvidos métodos baseados em kernel, os quais realizam um mapeamento não-linear das entradas para um espaço onde seja possível encontrar este vetor. A partir da escolha de um kernel apropriado [8], [9], o problema poderá ser resolvido por um modelo linear.

Essa transformação é justificada pelo teorema de Cover sobre a separabilidade de padrões [10], o qual diz que um problema de classificação complexo projetado não-linearmente para um espaço de alta dimensão é mais provável de ser separável do que em um espaço de baixa dimensão, desde que o espaço não seja densamente povoado. Este princípio é explorado por diversos modelos como o SVM, redes ELM, RBF e MLP [11].

O KDE é um estimador de densidade não-paramétrico baseado na soma de funções de kernel em cada dimensão do espaço de entradas [3]. Tais funções são tipicamente Gaussianas e possuem somente um parâmetro, a sua largura σ, dado que os centros serão as amostras em si. O problema de estimar esta densidade resulta no σ que satisfaz um conjunto de restrições ou função objetivo [2]. Considerando um problema de classificação binário, podemos definir um classificador linear no espaço de verossimilhanças estimado pelo KDE como:

$$P(\mathbf{x}|C_1) = kP(\mathbf{x}|C_2) \quad (2)$$

onde $P(\mathbf{x}|C_i)$ é a probabilidade condicional de $\mathbf{x}$ dado uma classe C_i, e k é a inclinação da reta que separa linearmente as verossimilhanças. Conforme observado por [2], encontrar um k que garante a melhor separação pode ser feito através do ajuste correto de σ.

A defininição de σ é tipicamente realizada por técnicas de estimativa de erro como a validação cruzada. Entretanto, observando as características do mapeamento aplicado via funções de Kernel, [2] propôs uma forma simples de determinar a largura ótima como a maximização da uma função de similaridade sobre o espaço de verossimilhanças. Os autores mostram evidências que esta função possuirá somente um máximo para diferentes problemas reais. Além disso, a técnica foi aplicada com sucesso em conjunto com o algoritmo SVM, o qual obteve resultados equivalentes em relação aos obtidos com validação cruzada sobre a largura.

B. SVM

Introduzido por Vapnik em 1992 [12] as máquinas de vetores suporte (SVM) são considerados um dos algoritmos mais robustos e poderosos para aprendizado supervisionado até os dias atuais. No trabalho de Vapnik, o autor apresenta de forma bem completa os fundamentos teóricos deste modelo, apresentando justificativas para sua ótima capacidade de generalização, bem como os limites para a sua validade. Seu princípio de treinamento está na maximização da margem entre os padrões de treino e a superfície de decisão, que é uma representação convexa do compromisso entre viés e variância. Estas propriedades são uma consequência do uso dos chamados vetores de suporte no seu treinamento, que são os sub-conjuntos de dados mais próximos da superfície de decisão, ou seja, os mais difícies de classificar e relevantes para sua otimalidade. O problema então consiste em encontrar o hiperplano que maximiza a margem de separação entre os padrões, que é equivalente a minimizar a norma Euclidiana do seu vetor de pesos [11].

C. ELM

Inicialmente proposto por [6], as máquinas de aprendizado extremo (ELM) são redes neurais com uma única camada escondida, as quais possuem o atrativo de poucos parâmetros a serem ajustados, generalização maior ou similar e redução do tempo de treinamento das redes em relação aos métodos convencionais. Seu treinamento é baseado na teoria de minimização de risco empírico, necessitando de somente uma iteração para este processo, evitando múltiplas iterações e algoritmos de otimização local [13].

ELMs são capazes de definir adaptivamente o número neurônios da rede e aleatoriamente escolher os pesos das entradas e viéses da camada escondida [14]. Isso faz com o que a rede possa ser considerada como um sistema linear, o qual pode ser calculado de forma analítica através de uma operação de inversão da matriz de pesos da camada de saídas [14]. Essa característica permite uma drástica redução do esforço computacional do treinamento, geralmente de 10 vezes ou mais [15].

III. Metodologia

A. Bases de Dados

Neste trabalho serão consideradas quinze bases de dados referentes a problemas de classificação binários disponíveis no repositório da UCI [16]. Antes do treinamento, os dados de entrada serão normalizados para o intervalo $[0, 1]$ e filtrados para remoção de valores inválidos. Um sumário das bases de dados pode ser visto na Tabela III-A.

Dataset	Instâncias	Atributos	Proporção
ILPD	578	10	0.71/0.29
appendicitis	106	7	0.80/0.20
australian	690	14	0.56/0.44
banknote	1372	4	0.56/0.44
breastcancer	569	30	0.37/0.63
bupa	345	6	0.42/0.58
climate	540	18	0.09/0.91
fertility	100	9	0.88/0.12
glass	214	9	0.86/0.14
haberman	306	3	0.74/0.26
heart	270	13	0.56/0.44
ionosphere	351	33	0.64/0.36
sonar	208	60	0.47/0.53
segmentation	2310	19	0.86/0.14
pima	768	8	0.65/0.35

B. Desenho do experimento

A partir das recomendações para desenho de experimento para comparação de algoritmos proposta em [17], a seguinte metodologia será adotada:

- Para cada base de dados:
  1) Particionar os dados D em k partições para validação cruzada, mantendo a mesma proporção entre os rótulos
     a) Criar o conjunto de treino $T = D - k$
     b) Para cada modelo:
        i) Executar busca exaustiva com validação cruzada de z sobre T para os coeficientes de regularização λ
        ii) Escolher λ que obtém o melhor ajuste médio
        iii) Avaliar a métrica do modelo sobre k
  2) Estimar o intervalo de confiança de 95% do valor médio da métrica sobre k usando *bootstraping*

O número de partições consideradas será de $k = 10$. Para o item 1(b)i, serão considerados um número fixo de valores

igualmente espaçados dentro de um intervalo pré-definido. Além disso, serão considerados $z = 5$ partições, como forma de controlar um pouco o tamanho do experimento. A acurácia será utlizada como a métrica para comparação dos algoritmos. A implementação deste experimento, bem como dos modelos utilizados, será feita em Python e utilizando principalmente os pacotes *numpy* [18] e *scikit-learn* [19]. O experimento será realizado em um Notebook Intel Core i7 Quad Core com 8Gb de memória RAM, sendo que o uso de paralelização será utilizado sempre que possível visto a enorme quantidade de vezes que os modelos serão treinados.

C. Aprendizado Hebbiano

O uso de aprendizado Hebbiano após a camada escondida é uma abordagem simples e eficiente para se controlar a generalização do modelo. Conforme sugerido em [20], podemos substituir o cálculo da pseudo-inversa da matriz de projeção, o que caracteriza a regra de aprendizado do Percepton simples, por um Perceptron Hebbiano com pesos normalizados. Essa abordagem é melhor descrita em [5], da qual podemos retirar a seguinte regra para problemas de classificação binários:

$$w = \frac{\sum_{i=1}^{N} y_i \mathbf{h}_i}{\left\| \sum_{i=1}^{N} y_i \mathbf{h}_i \right\|} \tag{3}$$

onde $\mathbf{h}_i$ é á i-ésima linha da matriz de projeção. É importante ressaltar que esta regra assume que os dados foram normalizados para possuir média zero e desvio padrão unitário. Conforme os resultados obtidos no anterior da disciplina, o uso do aprendizado Hebbiano é capaz de atingir a regularização desejada, porém seu desempenho é limitado a projeções que tornam os dados linearmente separáveis. Logo, podemos concluir que esta abordagem poderá ter desempenho satisfatório caso a projeção no espaço de verossimilhanças seja ortogonal.

D. RBF com Otimização de Largura

As redes RBF foram inicialmente introduzidas por [21] e são caracterizadas por um aprendizado que envolve duas etapas: (i) aplicar uma transformação aos padrões através de um kernel Gaussiano (ii) encontrar os pesos usando o estimador mínimos quadrados usado no Perceptron simples. Portanto, a qualidade do modelo obtido será diretamente afetado pela escolha dos centróides e larguras das funções radias de cada neurônios.

Esta definição é geralmente feita utilizando o algotimo *k-means* [11], porém podemos facilmente ver que a mesma abordagem de otimização da largura de SVMs com kernel RBF proposta por [2] pode ser aplicada. Se considerarmos que cada neurônio usará a largura ótima obtida pela técnica anterior, precisamos definir somente os centróides na sequência, os quais serão amostrados uniformemente no espaço de entradas, resultando em um menor esforço computacional e complexidade. Na etapa final de treinamento, também será feito o uso de regularização para um maior controle de sua generalização. Espera-se um desempenho similar ao SVM, porém sem a expectativa de maximização de margem durante o treinamento, a qual será controlada pelo coeficiente de regularização.

IV. Resultados

Os classificadores avaliados serão identificados por: **GK-H**, **GK-P**, **MLPK-H**, **MLPK-P**, **GK-SVM** e **GK-RBF**. Os prefixos **GK** e **MLPK** representam a projeção utilizada no espaço de verossimilhanças: kernel Gaussiano e MLP, respectivamente. Os sufixos **H**, **P**, **SVM** e **RBF** representam o modelo utilizado: Perceptron Hebbiano, Perceptron Simples, SVM e RBF respectivamente.

A. Análise de problemas bi-dimensionais

Considerando três conjuntos de dados gerados artificialmente para diferentes estruturas, a superfície de decisão para cada classificador pode ser vista nas Figuras 1, 2 e 3. Observe que à medida em que a complexidade do modelo é aumentada, maior é a dificuldade encontrada pelos modelos lineares.

Para o problema da Figura 1, é possível ver que todos os modelos estimaram com precisão a superfície de separação. Entretanto, note que para o problema da Figura 2 os modelos lineares não tiveram um bom desempenho, provavelmente devido à qualidade da projeção obtida. Isso é evidenciado pelo problema da Figura 3, para o qual os modelos lineares obtiveram desempenho superior quando utilizada a projeção MLP. Destaca-se também que o GK-RBF se ajustam muito bem aos dados e e de foram bem similar ao GK-SVM, indicando que a estratégia adotada é bastante promissora.

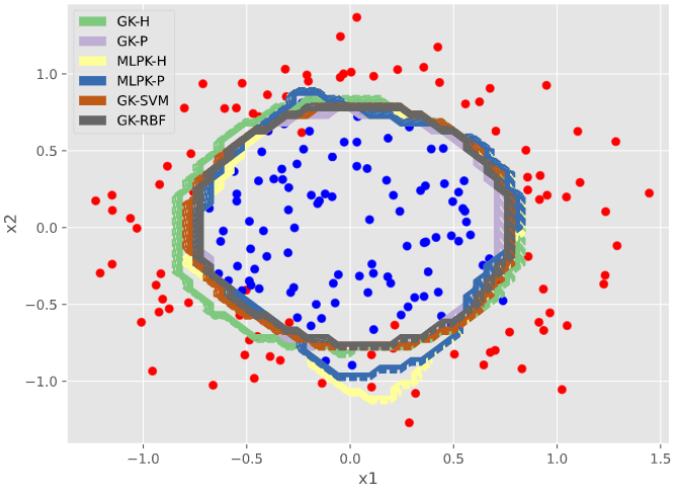

Fig. 1. Superfície de decisão para o problema de círculos concêntricos

B. Análise do experimento

Considerando os mesmos algoritmos da seção anterior, é incluído também o algoritmo ELM com regularização na sua versão original, ou seja, sem o uso da técnica de projeção no espaço de verossimilhanças. O número de neurônios das redes ELM e GK-RBF foram mantidos fixos em 100. Os valores de coeficiente de regularização utilizados para o GK-SVM, GK-RBF e ELM foram escolhidos no intervalo

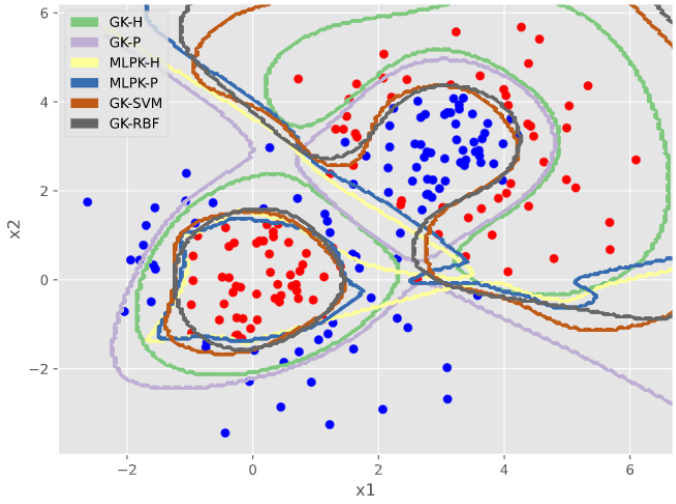

Fig. 2. Superfície de decisão para o problema de dois círculos concêntricos alternados

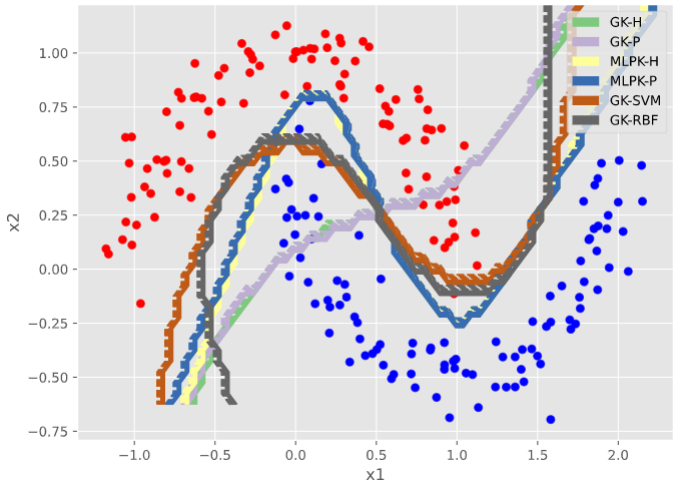

Fig. 3. Superfície de decisão para o problema das espirais

$[2^{-5}, 2^{-4}, \ldots, 2^{11}, 2^{12}]$. A acurácia média e seu intervalo de confiança obtido via bootstraping para o experimento estão consolidados na Tabela II. Comparando com os resultados reportados em [2], nota-se que todos estão coerentes, especialmente para a implementação do GK-SVM. Destaca-se o excelente desempenho do algoritmo ELM, o qual também está de acordo com o reportado na literatura [22].

Observando os demais métodos avaliados, é importante destacar também que o modelo GK-RBF apresentou desempenho equivalente ao GK-SVM em grande parte das bases de dados, porém ainda com ma acurácia elevada e compatível com o esperado para a base de dados. Portanto, podemos afirmar que esta é uma abordagem que, embora simples de entender e implementar, tem potencial para novas aplicações. Embora a versão clássica do treinamento de redes RBF não foi incluída para comparação, uma outra vantagem que ficou evidente durante a execução do experimento foi a redução drástica no tempo de treinamento, uma vez que foi eliminada a necessidade do cálculo dos centros e raios via clustering.

A grande surpresa destes testes está na baixa qualidade dos resultados do modelo com aprendizado Hebbiano: para todas as bases de dados avaliadas, o modelo GK-H apresentou acurácia inferior em relação aos demais para um intervalo de confiança de 95%. O mesmo não ocorreu quando foi utilizado o MLP-H, onde para algumas bases de dados a acurácia foi equivalente. Isso sugere que a projeção no espaço de verossimilhanças com o Kernel radial possa não ter atingido a ortogonalização esperada.

O cálculo do cross-talk sobre o conjunto de dados projetado permite obter uma ideia da qualidade desta ortogonalização, onde um valor próximo de zero indicará uma boa projeção. Os valores obtidos estão disponíveis na Tabela IV-B, os quais foram calculados considerando a média do valor de cada amostra em relação à todas as outras. É relevante citar que durante o cálculo, se considerássemos somente as amostras de classes diferentes classe, o valor era aproximadamente zero e foram omitidos da tabela. Portanto, os valores apresentados correspondem ao cross-talk intra-classe, o qual é naturalmente mais alto. Dessa forma podemos concluir que a ortogonalização obtida é de fato ortogonal, e outro fator pode estar afetando o aprendizado Hebbiano.

TABELA I

CROSS-TALK MÉDIO CALCULADO PARA AS PROJEÇÕES GAUSSIANA E MLP SOBRE TODO O CONJUNTO DE DADOS

Dataset	Gaussiana	MLP
ILPD	0.973675	1.299812
appendicitis	0.594261	-1.086522
australian	0.069242	0.029584
banknote	-0.212484	-0.224532
breastcancer	0.318198	0.575652
bupa	-0.262322	-1.124887
climate	-1.525012	-1.253363
fertility	1.445575	0.590236
glass	-0.216374	-1.345287
haberman	0.756672	0.193615
heart	-0.070371	-0.422381
ionosphere	0.048857	-0.723001
pima	0.408878	-0.415799
segmentation	1.377789	0.043600
sonar	-0.560352	0.396350

Se obervamos a proporção entre as classes de cada problema da Tabela III-A, podemos notar que há quando maior o desbalanceamento entre as classes, pior é o desempenho dos modelos com aprendizado Hebbiano. Isso é um resultado interessante, podendo significar que nestes casos a projeção calculada pode não ter sido ortogonal o suficiente para que o aprendizado Hebbiano fosse efetivo. Considerando a Tabela IV-B, pode-se notar também que estes datasets em específico possuem um desvio maior do zero em relação aos demais, reforçando esta observação.

A superfície de decisão estimada via aprendizado Hebbiano para um base com alta e baixa acurácia pode ser vista nas Figuras 4 e 5, respectivamente. É interessante observar que o desbalanceamento fez com que as projeções em cada dimensão

se tornassem desiguais. Além disso, o aprendizado Hebiano aparenta ter convergido para uma superfície que maximiza a separação dos dados, independente da classe. Isso sugere que uma classe dominou a outra durante o treinamento, e que o aprendizado Hebbiano possa não ser indica nestas situações.

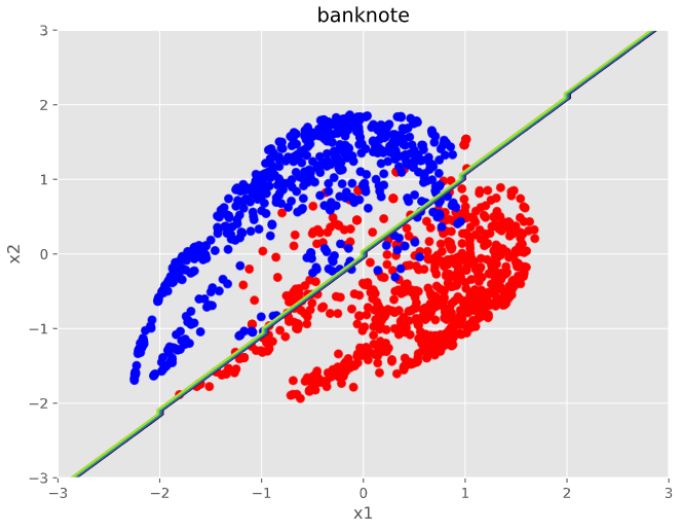

Fig. 4. Projeção para a base de dados *banknote* e a superfície de decisão estimado pelo aprendizado Hebbiano

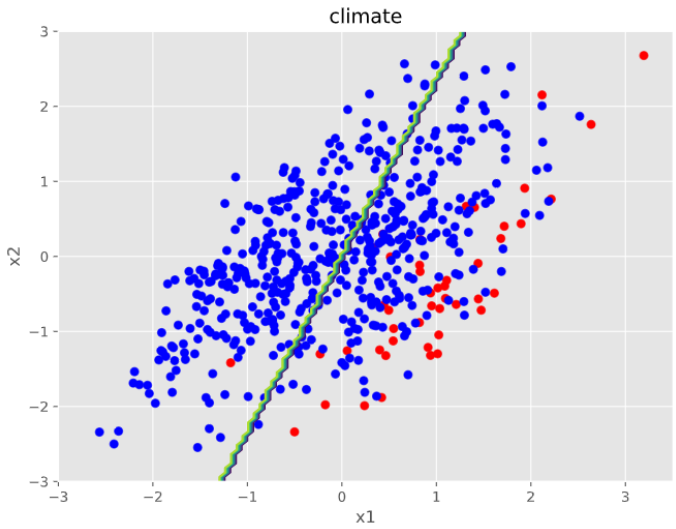

Fig. 5. Projeção para a base de dados *climate* e a superfície de decisão estimado pelo aprendizado Hebbiano

V. Conclusões

Neste trabalho foi feita uma avaliação do desempenho do ELM e modelos usando a abordagem por projeção no espaço de verossimilhanças sobre bases de dados de benchmark presentes na literatura. A partir de um experimento desenhado para tal finalidade, os resultados foram comparados estatísticamente utilizando a técnica de *bootstrapping* para a acurácia média, considerando um intervalo de confiança de 95%. Destaca-se o excelente desempenho do ELM em todos os conjuntos de dados, conforme observado nos trabalhos anteriores. É importante destacar também que resultados compatíveis com a literatura foram obtidos para o SVM com otimização da largura. Além disso, a aplicação deste mesmo conceito para o treinamento de redes RBF se mostrou bastante eficaz. Notou-se também que os modelos lineares avaliados obtiveram bons resultados dependendo da base de dados em questão, conseguindo atingir um desempenho estatisticamente equivalente aos demais.

Assim como no trabalho anterior, o desempeho do aprendizado Hebbiano deixou a desejar, apresentando um resultado muito inferior em relação aos demais modelos. Uma avaliação do cross-talk das bases de dados mostrou que não houve uma correlação clara entre o seu aumento e a baixa acurácia do modelo, de forma a indicar que outro fator esteja causando este problema. Para algumas bases de dados ele consegue atingir uma acurácia alta, embora estatisticamente tenha ficado inferior aos demais métodos para um intervalo de confiança de 95%.

Como trabalhos futuros, é sugerido uma investigação mais detalhada do que pode estar causando a baixa acurácia dos modelos com aprendizado Hebbiano. Além disso, seria interessante incluir o modelo RBF com aprendizdo via clustering e compará-lo com o novo método proposto.

Referências

[1] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[2] Murilo VF Menezes, Luiz CB Torres, and Antonio P Braga. Width optimization of rbf kernels for binary classification of support vector machines: A density estimation-based approach. *Pattern Recognition Letters*, 128:1–7, 2019.

[3] Shitong Wang, Zhaohong Deng, Fu-lai Chung, and Wenjun Hu. From gaussian kernel density estimation to kernel methods. *International Journal of Machine Learning and Cybernetics*, 4(2):119–137, 2013.

[4] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[5] Manuel Fernandez-Delgado, Jorge Ribeiro, Eva Cernadas, and Senén Barro Ameneiro. Direct parallel perceptrons (dpps): fast analytical calculation of the parallel perceptrons weights with margin control for classification tasks. *IEEE transactions on neural networks*, 22(11):1837–1848, 2011.

[6] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: a new learning scheme of feedforward neural networks. In *2004 IEEE international joint conference on neural networks (IEEE Cat. No. 04CH37541)*, volume 2, pages 985–990. IEEE, 2004.

[7] M Menezes, L Torres, and A Braga. Otimização da largura de kernels rbf para máquinas de vetores de suporte: Uma abordagem baseada em estimativa de densidades. In *XIII Congresso Brasileiro de Inteligência Computacional*, 2017.

[8] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[9] Richard Courant and David Hilbert. *Methods of Mathematical Physics*, volume 1. Wiley, New York, 1989.

[10] Thomas M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334, 1965.

[11] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice-Hall, Inc., 2007.

[12] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

TABELA II

ACURÁCIA MÉDIA DOS CLASSIFICADORES PARA INTERVALO DE CONFIANÇA DE 95%

Base	ELM	GK-H	GK-P	GK-RBF	GK-SVM	MLPK-H	MLPK-P
ILPD	**70.97 (69.82,72.20)**	52.95 (49.00,56.68)	66.29 (63.18,69.65)	**71.45 (71.01,71.92)**	**71.43 (70.94,71.96)**	60.77 (55.69,66.12)	**71.43 (70.90,72.02)**
appendicitis	**85.82 (81.09,90.00)**	76.36 (66.45,86.37)	**87.64 (82.00,93.46)**	**84.73 (78.00,90.91)**	**86.73 (80.36,92.55)**	61.55 (49.99,74.28)	75.36 (70.91,80.64)
australian	85.94 (83.04,89.13)	86.09 (83.33,88.99)	87.60 (85.35,89.70)	86.09 (83.62,89.13)	85.80 (82.90,88.84)	85.19 (82.93,87.60)	83.91 (81.88,85.95)
banknote	**97.96 (97.45,98.54)**	91.76 (90.44,93.08)	91.91 (90.38,93.30)	**99.78 (99.56,100.00)**	**100.00 (100.00,100.00)**	**100.00 (100.00,100.00)**	**100.00 (100.00,100.00)**
breastcancer	**96.48 (95.42,97.70)**	92.09 (89.99,94.02)	94.53 (93.74,95.31)	91.21 (90.33,92.09)	**98.82 (98.04,99.62)**	**96.88 (96.30,97.26)**	**96.31 (94.91,97.54)**
bupa	**70.45 (67.33,73.45)**	50.14 (45.03,55.31)	60.65 (58.71,62.75)	61.10 (57.97,63.86)	**70.81 (65.53,76.24)**	55.96 (51.24,60.89)	57.10 (53.28,60.34)
climate	**91.11 (90.37,91.67)**	57.59 (54.81,60.00)	**91.48 (90.93,92.04)**	**91.11 (90.37,91.85)**	**91.30 (90.74,91.85)**	78.15 (70.37,87.04)	**89.81 (88.52,91.11)**
fertility	**86.00 (83.00,89.00)**	52.22 (42.22,61.14)	**86.00 (83.00,89.00)**	**87.78 (85.56,90.00)**	**87.00 (84.00,90.00)**	60.00 (50.00,70.00)	**87.78 (85.56,91.11)**
glass	**96.26 (94.37,98.48)**	86.84 (79.99,94.32)	**97.08 (95.35,98.32)**	**97.68 (95.84,99.94)**	**96.26 (93.90,99.05)**	**95.30 (92.49,98.61)**	**96.26 (93.94,98.55)**
haberman	**76.24 (74.43,77.98)**	53.59 (50.04,57.03)	**74.69 (73.27,76.10)**	**72.23 (70.04,74.55)**	**72.56 (71.01,74.11)**	52.57 (44.63,60.08)	**72.54 (71.30,74.05)**
heart	**84.77 (83.54,86.01)**	78.19 (74.48,82.30)	**82.22 (78.52,85.56)**	**85.19 (82.59,87.78)**	79.63 (76.67,82.59)	**82.22 (78.52,85.93)**	**87.50 (84.25,90.28)**
ionosphere	**88.31 (85.98,91.17)**	70.08 (66.00,74.65)	**87.00 (84.11,90.20)**	**89.47 (85.14,93.22)**	**93.73 (91.17,96.32)**	**92.89 (90.63,95.14)**	93.17 (90.88,95.62)
pima	**76.42 (74.18,78.51)**	67.44 (65.56,69.13)	**72.52 (69.45,75.45)**	**75.91 (73.45,78.52)**	**77.74 (75.83,79.59)**	**75.52 (72.63,78.08)**	**76.77 (75.32,78.02)**
segmentation	**99.65 (99.48,99.83)**	57.06 (54.93,59.18)	**88.79 (87.97,89.65)**	**98.61 (98.14,99.09)**	**99.70 (99.48,99.91)**	78.16 (75.71,80.42)	85.71 (85.71,85.71)
sonar	77.14 (73.33,81.59)	69.52 (65.55,73.81)	75.06 (73.24,76.87)	73.97 (70.63,77.62)	**86.19 (83.33,89.21)**	**81.27 (78.89,83.81)**	**80.48 (77.46,83.65)**
Average	**85.87 (84.26,87.61)**	69.49 (67.11,72.01)	**82.55 (81.04,84.29)**	**84.94 (83.10,86.61)**	**86.66 (85.11,88.33)**	75.35 (72.56,78.27)	**83.76 (81.96,85.44)**

[13] Shifei Ding, Han Zhao, Yanan Zhang, Xinzheng Xu, and Ru Nie. Extreme learning machine: algorithm, theory and applications. *Artificial Intelligence Review*, 44(1):103–115, 2015.

[14] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: theory and applications. *Neurocomputing*, 70(1-3):489–501, 2006.

[15] Wan-Yu Deng, Qing-Hua Zheng, Lin Chen, and Xue-Bin Xu. Research on extreme learning of neural networks. *Chinese Journal of Computers*, 33(2):279–287, 2010.

[16] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[17] Steven L Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data mining and knowledge discovery*, 1(3):317–328, 1997.

[18] Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Euler Guimarães Horta. Aplicação de máquinas de aprendizado extremo ao problema de aprendizado ativo. 2015.

[21] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Syst.*, 2, 1988.

[22] Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, and Rui Zhang. Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529, 2011.