Redes Neurais Artificiais: Artigo 03

Leonam R. S. Miranda, Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais

***Resumo*—Neste trabalho foi feito um estudo comparativo entre diferentes modelos de redes neurais, em diferentes conjuntos de dados.**

***Palavras-chave*—Redes Neurais, Hebbiano, Valição Cruzada, Kernel, KDE,**

I. Introdução

Neste trabalho será feito um estudo comparativo entre diferentes modelos de redes neurais, Perceptron, Adaline, ELM, RBF, ELM Hebbiano (Euler) [1], redes MLP e Hebbiano utilização kernels previamente para gerar uma projeção dos Dados no Espaço de Verossimelhanças. A escolha dos conjuntos de dados foi feita visando obter alguma diversidade quanto ao número de variáveis, número de amostras, dados desbalanceados, etc. A maior parte das bases de dados utilizados neste trabalho se encontram no reposiório de aprendizagem de máquina da UCI [2]. Nesse trabalho serão avaliados apenas problemas de classificação, portanto a acurácia e a área sobre a curva ROC (receiver operating characteristic) AUC (area under curve) [3], devido ao desbalanceamento de algumas das bases utilizadas, foram adotadas como metricas de performance. Para cada modelo, em cada uma das bases de dados, será utilizado validação cruzada para selecionar os hiperparâmetros, visando encontrar o modelos onde o erro esperado seja baixo, sem sobreajuste [4].

II. Aprendizagem Hebbiana Utilizando Kernels

Antes de explicar o modelo que aplica aprendizado hebbiano sobre o espaço de verossimilhanças fornecidas pelo estimador de densidade de kernel primeiramente será explicado separadamente o aprendizado Hebbiano e a metodologia para se obter os estimadores de densidade de Kernel.

A. Aprendizado Hebbiano

O aprendizado Hebbiano é uma forma de aprendizado não supervisionado cujo resultado é proporcional ao produto cruzado dos padrões pelos seus rótulos. Essa forma de aprendizado não se baseia em correção do erro, não exigindo, portanto, a comparação do rótulo desejado pelo obtido. Um termo residual devido a não ortogonalidade dos dados estará presente na maioria das aplicações

No aprendizado de redes neurais artificiais, a atualização dos pesos w_{ij} que conectam neurônios i e j de acordo com a regra de Hebb [5] é proporcional ao produto cruzado de seus valores de ativação para cada associação entrada e saída k ou, em outras palavras, $w_{ij} \propto x_{ik}y_{jk}$. A regra pode ser escrita na seguinte forma matricial:

$$w = X^T y \tag{1}$$

Conforme [1], como $\hat{y} = XW$. Ao substituir na equação 1 tem-se que $\hat{y} = XX^T y$. Para recuperar perfeitamente y deve-se ter $XX^T = I$. Se essa condição é obtida o erro de treinamento é zero, já que $\hat{y} = y$, e a regra de aprendizado seria equivalente ao método da pseudoinversa. Nessa situação X é uma base ortonormal e $X^T = X^{-1}$ [6].. Entretanto, na maioria das situações reais X não será uma base ortonormal e um termo residual devido ao produto XX^T causará um deslocamento de $\hat{y}$ em relação a y.

Como $\hat{y}_k = \sum_{j=1}^{N} x_k^T x_j y_j$, o termo correspondente a $j = k$ pode ser separado da soma o que leva a equação 2 . Na situação particular em que os dados de entrada são normalizados, ou seja $x_k^T xk = 1$, a Equação 2 pode ser simplificada na Equação 3 [7]:

$$\hat{y}_k = x_k^T x_k y_k + \sum_{j=1, j\neq k}^{N} x_k^T x_j y_j \tag{2}$$

$$\hat{y}_k = y_k + \sum_{j=1, j\neq k}^{N} x_k^T x_j y_j \tag{3}$$

O somatório $\sum_{j=1,j\neq k}^{N} x_k^T x_j y_j$ é chamado de ***crosstalk***. O ***crosstalk*** é ineente ao aprendizado Hebbiano e é devido à não ortogonalidade dos dados de entrada; Ao realizar uma seleção dos padrões mais relevantes a serem aprendidos gera um efeito de regularização, diminuindo o sobreajuste. Na tese do EULer [1] é proposto estratégias de aprendizado ativo para seleção de padrões para serem aplicados no aprendizado Hebbiano.

B. Estimador de Densidade de Kernel

O KDE é um estimador de densidade não paramétrico, que se baseia em uma soma de funções do kernel centradas em cada amostra de dados d-dimensional [8]. As funções do kernel, tipicamente gaussianas, têm apenas um parâmetro, a largura, uma vez que seus centros são as próprias amostras de dados. Assumindo a independência entre as variáveis de entrada d e o mesmo raio σ para todas as dimensões das funções gaussianas do KDE, a estimativa da densidade em um ponto arbitrário x_i pode ser obtida pela soma dos produtos acumulados em todas as dimensões, para todos os padrões no conjunto de amostras:

$$\hat{f}_h(x_i) = \frac{1}{N}\sum_{k=1}^{N}\left[\prod_{j=1}^{d}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{1}{2}\left(\frac{x_{ij}-x_{kj}}{\sigma}\right)}\right] \tag{4}$$

sendo $\hat{f}_h(x_i)$ a função de densidade no ponto x_i, N é o tamanho da mostra e d é a dimensão do espaço de entrada.

Já que os produtos da Equação 4 são equivalentes a $\frac{1}{(\sqrt{2\pi}\sigma)^d} e^{-\frac{(x_{ij}-x_{kj})^2}{2\sigma^2}}$, pode ser reescrita como:

$$\hat{f}_h(x_i) = \frac{1}{N(\sqrt{2\pi}\sigma)^d} \sum_{k=1}^{N} e^{-\frac{(x_i-x_k)^2}{2\sigma^2}} \tag{5}$$

A soma da Eq. 5 representa a soma de todos os elementos de uma linha (ou coluna) da matriz de kernel gaussiana $\mathbf{K}$, $[k_{ij} = K(x_i, x_j)]$, com raio σ, sendo k_{ij} o valor do kenerl avalaiado com x_i e x_j, e pode ser reescrita como a Eq. 6.

$$\hat{f}_h(x_i) = \frac{1}{N(\sqrt{2\pi}\sigma)^d} \sum_{k=1}^{N} K(x_i, x_k) \tag{6}$$

Assumindo independência entre as variáveis de entrada e um único raio para cada dimensão, a densidade estimada para um padrão arbitrário x_i pode ser obtida diretamente da matriz kernel $\mathbf{K}$, como mostrado esquematicamente na Fig. 1. A representação esquemática da figura considera que o problema de classificação é binário e que as amostras das duas classes são ordenadas, sendo as amostras da classe C_1 seguidas das da classe C_2. Classificando os elementos das duas classes, resulta na forma bloco-diagonal da Fig. 1. As submatrizes K_{11} e K_{22} contêm elementos do kernel dentro intra-classe e as submatrizes simétricas K_{12} e K_{21} contêm os valores do kernel entre as classes.

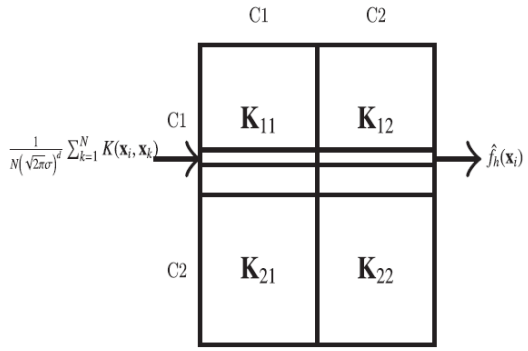
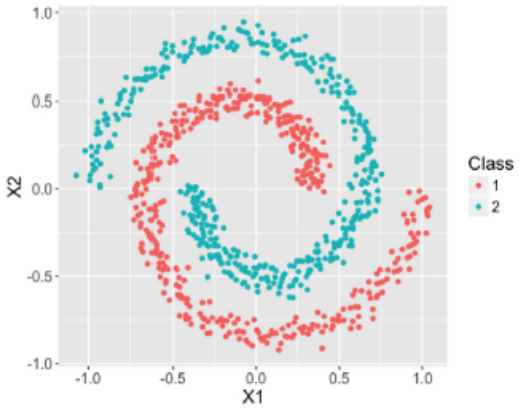


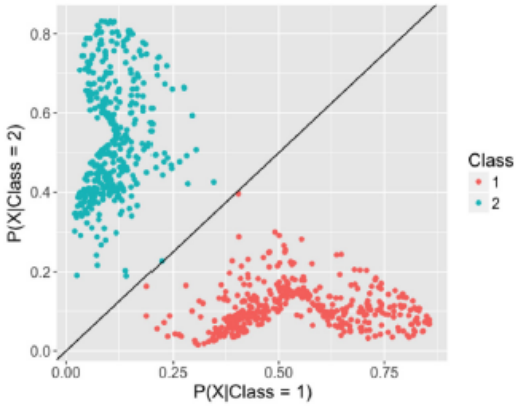
Figura 1. Representação esquemática da estimativa da densidade do kernel de um padrão de conjunto de aprendizagem x_i. De acordo com a Eq. 6, a soma é realizada para todos os elementos da linha i da matriz do kernel [9].

Como um exemplo [9], foram obtidas as projeções no espaço de verossimilhanças utilizando KDE no problema bem conhecido não linear de espirais, apresentado na 2(a). As duas verossimilhanças $P(x|C_1)$ e $P(x|C_2)$ foram estimadas usando KDE, resultando no mapeamento 2(b) para cada amostra. No espaço de verossimilhanças uma separação linear das amostras pode ser obtida.

Assim, a ideia do classificador que utiliza **Aprendizagem Hebbiana com Kernels RBF** é aplicar aprendizado hebbiano sobre as projeções das amostras no espaço de verossimilhanhças obtidas com kernels MLP ou gaussianos.



(a) Non-linearly separable problem at input space



(b) Projection of data at likelihood space, using KDE with $\sigma = 0.22$. Here, $k = 1$, as the number of samples is the same in both classes. For a new pattern, it is classified as belonging to Class 2 if it is mapped to the left of the black line, and classified as belonging to Class 1 otherwise.

Figura 2. Exemplo de dados separáveis não linearmente mapeados para o espaço de verossimilhanças [9].

III. Redes MLP

As redes MLP são caracterizadas por uma estrutura de neurônios com duas ou mais camadas alimentadas para frente (*feed-forward*). Os neurônios das camadas intermediárias possuem tipicamente funções de ativação sigmoidais enquanto que os neurônios da camada de saída podem ser implementados também com funções de ativação lineares, dependendo do problema a ser resolvido. Usualmente as saídas sigmoidais são utilizadas em problemas de classificação, enquanto que as saídas lineares são utilizadas em problemas de aproximção de funções ou de previsão. O esquema geral da topologia de uma rede MLP é apresentado na Fig. 3.

Redes neurais do tipo MLP são aproximadores universais de funções [10] e são tipicamente aplicadas a problemas de classificação, regressão ou previsão. A aproximação de funções com redes MLP requer basicamente a definição de uma arquitetura de rede, de um método de indução e de um algoritmo que implemente este método. O método de indução mais usual é a minimização do erro quadrático.

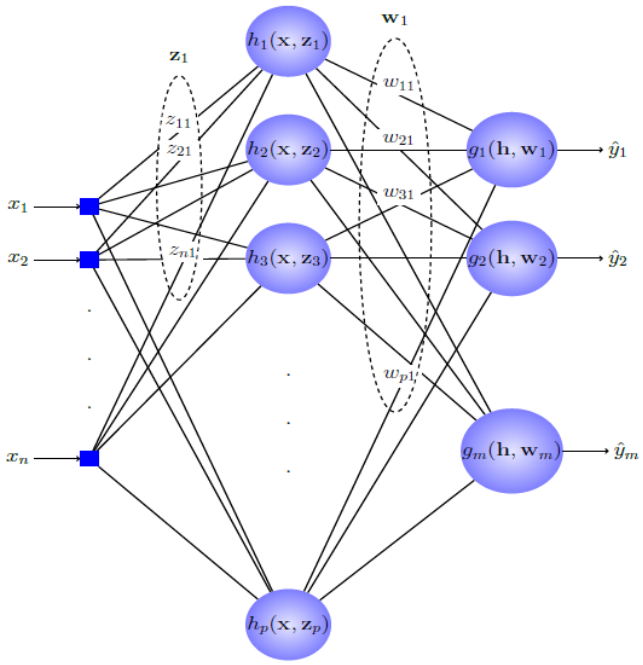


Figura 3. Rede neural de duas camadas do tipo MLP.

Estes modelos, no entanto, utilizaram diferentes algoritmos para a implementaçãoo da mininização do erro quadrático. O método de indução tipicamente utilizado para redes MLP é o gradiente descendente, cuja exemplo mais conhecido é o algoritmo backpropagation [11].

A estrutura do modelo, sua topologia e o número de neurônios em cada camada terão influencia na capacidade de o modelo resolver determinado tipo de problema. A escolha de uma dada topologia determinará a capacidade do modelo em se ajustar aos dados e também as incertezas inerentes ao processo de amostragem. Neste trabalho serão utilizadas redes MLP com apenas uma camada intermediária. Para cada um dos conjuntos de dados avaliados, será escolhido realizando validação cruzada o número de neurônios na camada intermediária e a sua função de ativação.

IV. Seleção de modelos

Para cada um dos modelos há hiperparâmetros que devem ser escolhidos antes de treiná-los. Para cada um dos conjuntos de dados ao alterar os hipeparâmetros dos modelos são obtidas métricas diferentes de performance. Assim, a escolha dos conjunto de hiperparâmetros para cada modelo em cada um dos conjuntos de dados foi feita através de busca intensiva em grade [12].

Aprender os parâmetros de uma função de previsão e testá-la com os mesmos dados é um erro metodológico: um modelo que apenas repetisse os rótulos das amostras que acabou de ver teria uma pontuação perfeita, mas não conseguiria prever nada de útil para dados ainda não vistos. Essa situação é chamada de overfitting. Para evitá-lo, é prática comum, ao realizar um experimento de aprendizado de máquina (supervisionado), manter parte dos dados disponíveis como um conjunto de testes X_test, y_test. Observe que a palavra “experimento” não tem a intenção de denotar apenas uso acadêmico, porque mesmo em ambientes comerciais, o aprendizado de máquina geralmente começa experimentalmente.

Ao avaliar diferentes configurações (”hiperparâmetros”) para estimadores ainda há o risco de overfitting no conjunto de teste, porque os parâmetros podem ser ajustados até que o estimador tenha um desempenho ideal. Dessa forma, o conhecimento sobre o conjunto de teste pode “vazar” para o modelo e as métricas de avaliação não informam mais sobre o desempenho de generalização. Para resolver este problema, outra parte do conjunto de dados pode ser realizada como um chamado ”conjunto de validação”: o treinamento continua no conjunto de treinamento, após a avaliação feita no conjunto de validação, e quando o experimento parece ser bem sucedido, a avaliação final pode ser feita no conjunto de teste.

No entanto, ao particionar os dados disponíveis em três conjuntos, reduzimos drasticamente o número de amostras que podem ser usadas para aprender o modelo, e os resultados podem depender de uma escolha aleatória particular para o par de conjuntos de treinamento e validação.

Uma solução para este problema é um procedimento denominado validação cruzada (abreviatura de CV). Um conjunto de teste ainda deve ser apresentado para avaliação final, mas o conjunto de validação não é mais necessário ao fazer o CV. Na abordagem básica, chamada k-fold CV, o conjunto de treinamento é dividido em k conjuntos menores (outras abordagens são descritas abaixo, mas geralmente seguem os mesmos princípios). O seguinte procedimento é seguido para cada uma dos k conjuntos (*”folds”*):

- Um modelo é treinado usando $k - 1$ folds como conjunto de treinamento;
- O modelo resultante é validado na parte restante dos dados (ou seja, é usado como um conjunto de teste para calcular uma medida de desempenho, como acurácia, erro quadrático médio, AUC, etc).

A medida de desempenho relatada pela validação cruzada k-fold é, então, a média dos valores calculados no loop. Esta abordagem pode ser computacionalmente cara, mas não desperdiça muitos dados (como é o caso ao corrigir um conjunto de validação arbitrário), o que é uma grande vantagem em problemas como inferência inversa, onde o número de amostras é muito pequeno.

Para este trabalho seão utilizadas as métricas AUC, para os problemas de classificação e e o coeficiente de determinação R^2 para os probelmas de regressão, que serão explicados nas subceções a seguir.

A. Curva ROC e Área sob a Curva (AUC)

Uma curva**ROC** [14] é construída traçando a taxa de verdadeiro positivo (TPR) contra a taxa de falso positivo (FPR). A taxa positiva verdadeira é a proporção de observações que foram corretamente previstas como positivas de todas as observações positivas ($TP/(TP + FN)$). Da

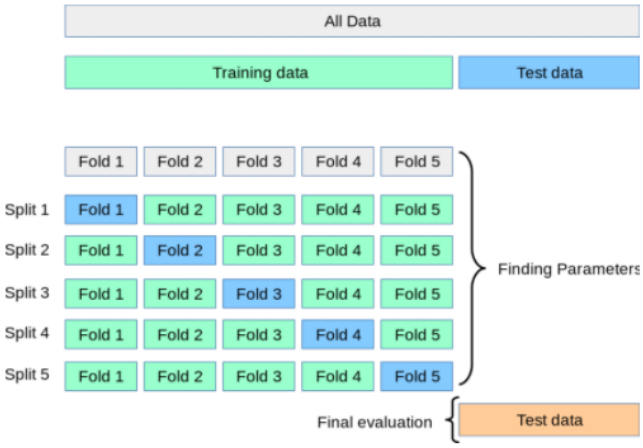


Figura 4. 5 folds CV [13]

mesma forma, a taxa de falsos positivos é a proporção de observações que são incorretamente previstas como positivas de todas as observações negativas ($FP/(TN+FP)$). Por exemplo, em testes médicos, a verdadeira taxa positiva é a taxa em que as pessoas são corretamente identificadas para teste positivo para a doença em questão. Sendo:

- TP: Verdadeiro positivo (quantidade de amostras classificadas como positivas sendo positivas);
- TN: Verdadeiro negativo (quantidade de amostras classificadas como negativas sendo negativas);
- FP: Falso positivo (quantidade de amostras classificadas como positivas sendo negativas);
- FN: Falso negativo (quantidade de amostras classificadas como negativas sendo positivas);

Um classificador discreto que retorna apenas a classe prevista fornece um único ponto no espaço ROC. Mas para classificadores probabilísticos, que fornecem uma probabilidade ou pontuação que reflete o grau em que uma instância pertence a uma classe em vez de outra, podemos criar uma curva variando o *threshold* para a pontuação. A Figura 5, exemplifica uma curva ROC.

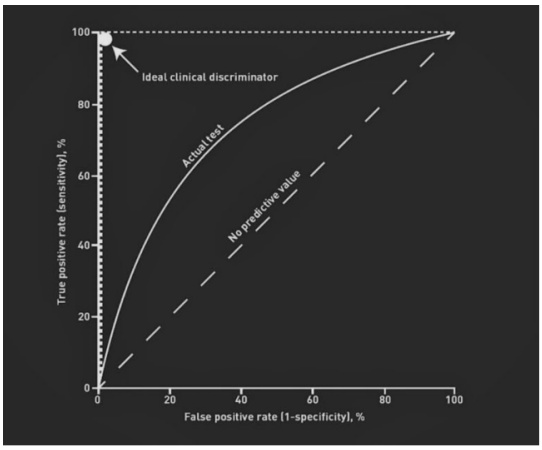


Figura 5. Exemplo de curva ROC

A curva ROC mostra o trade-off entre sensibilidade (ou TPR) e especificidade ($1-FPR$). Classificadores que fornecem curvas mais próximas do canto superior esquerdo indicam um melhor desempenho. Como linha de base, espera-se que um classificador aleatório forneça pontos situados ao longo da diagonal ($FPR = TPR$). Quanto mais perto a curva chega da diagonal de 45 graus do espaço ROC, menos preciso é o teste.

Observe que o ROC não depende da distribuição da classe. Isso o torna útil para avaliar classificadores que prevêem eventos raros (dados desbalanceados), como doenças ou desastres. Em contraste, avaliar o desempenho usando acurácia $(TP+TN)/(TP+TN+FN+FP)$ favoreceria classificadores que sempre predizem um resultado negativo para eventos raros.

Para comparar diferentes classificadores, pode ser útil resumir o desempenho de cada classificador em uma única medida. Uma abordagem comum é calcular a área sob a curva ROC, que é abreviada para **AUC** [15]. É equivalente à probabilidade de que uma instância positiva escolhida aleatoriamente seja classificada mais alta do que uma instância negativa escolhida aleatoriamente, ou seja, é equivalente à estatística de soma de classificação de Wilcoxon de duas amostras [16].

Um classificador com **AUC** alta pode ocasionalmente ter uma pontuação pior em uma região específica do que outro classificador com AUC mais baixa. Mas, na prática, o AUC tem um bom desempenho como uma medida geral de precisão preditiva.

V. Experimentos e Resultados

Experimentos foram performados com 6 conjunto de dados, sendo 5 do repositório da UCI [2] e um conjunto de dados de expressões gênicas [17].

Todos os conjuntos de dados foram normalizados com média $\bar{x} = 0$ e desvio padrão $\sigma = 1$, removendo amostras com valores ausentes. Em todas as bases de dados as amostras foram separadas entre treinamento e teste, sendo 75% para treinamento e 25% para testes.

Para a base de dados de Golub [17] especificamente, foi utilizado o algoritmo PCA [18] para selecionar as características que correspondem por 90% da variância, reduzindo de 7129 expressões gênicas para 22.

Para cada conjunto de dados há hiperparâmetros a serem definitos para cada modelo. Mo modelo que utiliza aprendizagem hebbiana com Kernels, deve-se defirnir o kenerl utilizado *mlp* ou *gaussiano*. Para as redes MLP deve-se dfinir o número de neurônios da camada intermediária, avaliados entre $2^i \ \forall i \in [1, 11]$, e a função de ativação da camada intermediária. As funções de ativação da camada intermediária que serão avaliadas estão definidas a seguir:

- identidade, sem operação de ativação, útil para implementar regressão linear, retorna $f(x) = x$
- logistica, a função sigmóide logística, retorna $f(x) = \frac{1}{(1+exp(-x))}$.
- tanh, a função tangente hiperbólica, retorna $f(x) = tanh(x)$.
- relu, a função de unidade linear retificada, returna $f(x) = max(0, x)$

Assim foi executado uma busca em grid avaliando o resultado da validação cruzada com 10 *folds* sobre o conjunto de treinamento, mensurando a performance para cada combinação de hiperparâmetros. Assim, os melhores hiperparâmetros econtrados, após busca em grid executando validação cruzada com 10 *folds*, se encontram na Tabela I.

Em todos os conjuntos de dados o melhor kernel encontrado por validação cruzada foi o *'mlp'*. As seguir se encontram imagens (Figs. [6, 11]) dos espaços de verossimilhanças obtidos aplicando kernel *'mlp'* em cada um dos conjuntos de dados.

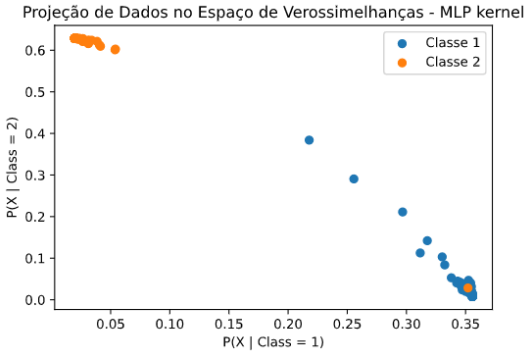


Figura 6. Base de Dados: Ionosphere

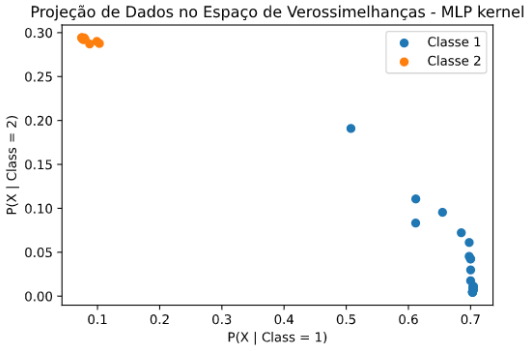


Figura 7. Base de Dados: Golub

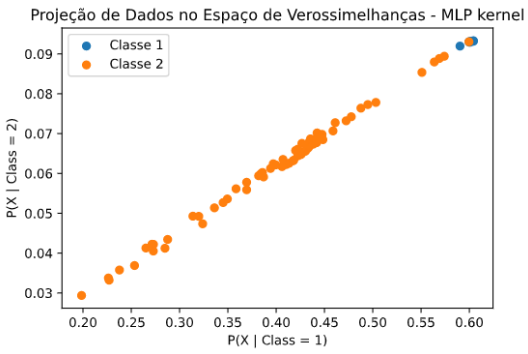


Figura 8. Base de Dados: Fertility

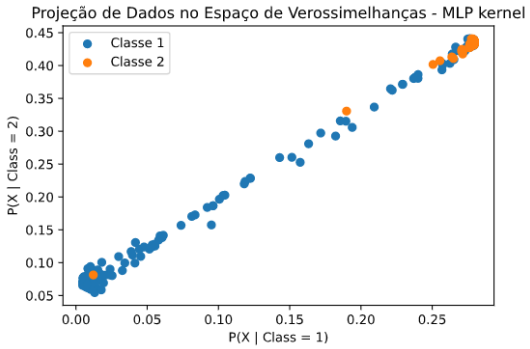


Figura 9. Base de Dados: Indian Liver

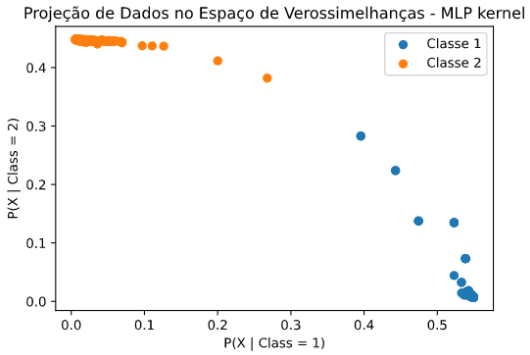


Figura 10. Base de Dados: Banknote

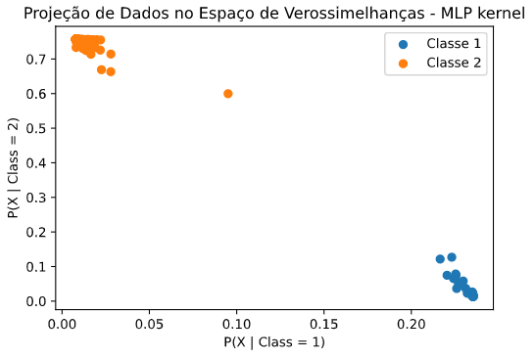


Figura 11. Base de Dados: parkisons

É possivel observar que a partir das figuras 6 a 11 que não foram obtidas boas separações das bases *Fertility* e *indian Liver* ao fazer a projeção das mesma sobre o espaço de verossimilhanças utilizando kernel *'mlp'*. Nessas mesmas bases foram as únicas que foi esolhido a função de ativação *identity* ao realizar validação cruzada sobre as redes MLP, sendo que usualmente em redes neurais de multiplas camadas, o mais comum na literatura é a utilização da função de ativação $ReLU$, já que a função de ativação $ReLU$ facilita a convergência.

Após obter os melhores hiperparâmetros para cada modelo e para cada conjunto de dados, realizou-se o treinamento dos modelos e foi avaliado a performance dos mesmos sobre os conjuntos de testes. Os resultados obtidos

Tabela I
MELHORES HIPERPARÂMETROS ECONTRADOS APÓS BUSCA EM GRID EXECUTANDO VALIDAÇÃO CRUZADA COM 10 *FOLDS*

	Hebbiano	MLP	
Conjunto de Dados	kernel	p	função de ativação
Ionosphere	mlp	128	ReLU
Golub	mlp	128	ReLU
Fertility Diagnosis	mlp	32	identity
Indian Liver Patient	mlp	16	identity
Banknote Authentication	mlp	32	ReLU
Parkisons	mlp	256	ReLU

Tabela II
VALORES MÉDIOS DE AUC E CARACTERÍSTICAS DOS CONJUNTOS DE DADOS

	Hebbiano	MLP	N_d	N	N^+	N^-
Conjunto de Dados						
Ionosphere	0.803 ± 0.017	0.877 ± 0.014	33	351	225	126
Golub	0.746 ± 0.165	0.938 ± 0.029	7129	72	25	47
Fertility Diagnosis	0.495 ± 0.011	0.500 ± 0.000	9	100	12	81
Indian Liver Patient	0.500 ± 0.000	0.715 ± 0.013	10	579	414	165
Banknote Authentication	1.000 ± 0.000	1.000 ± 0.000	4	1372	610	762
Parkisons	0.821 ± 0.108	0.929 ± 0.010	4	1372	610	762

para cada conjunto de dados estão listado na Tabela II, onde N e N_d correspondem à quantidade total de amostras e atributos, respectivamente, N^+ é o número de amostras positivas e N^- o número de amostras negativas. Na tabela se encontram as performances médias (AUC) [3] e desvios padrões após dez ensaios.

Os resultados, principalemte do MLP estão muito proximos de resultados obtidos em trabalhos como: [9] e [19].

Observa-se que os piores resultados foram obtdos com as bases mais desbalanceadas (Fertility Diagnosis e Indian Liver Patient), onde o Hebbiano com Kernel e o MLP não alcaçaram resultados satisfatórios. Em todos os outros conjuntos de dados o algoritmo MLP obteve performance maior ou igual as performances obtidas pelo Hebbiano com kernel.

VI. ANÁLISE DE CONCLUSÃO

A partir desse trabalho pode-se observar a importância na seleção de hiperparâmetros nos modelos, pois impacta bastante em seus desempenhos, e como a escolha dos melhores hiperparâmetros varia bastante em função da base de dados. Entretanto, deve-se levar em consideração que a busca em grid realizando validação cruzada para seleção de modelos é um processo muito custoso, dependedo da quantidade de valores dos hiperparâmetros a serem avaliados.

Tealvez fossem obtido resultados ainda melhores se fossem utilizadas redes MLP com mais camadas, mas tendo em mente que ao aumentar a capacidade do modelo, tende-se a ocorrer o fenômeno de overfitting.

Comparando com os resultados com os obtidos no artigo anterior onde foram avaliados os algoritmos ELM, RBF, ELM Hebbiano, e Perceptron o algoritmo MLP obteve melhores resultados nos conjuntos de dados *Ionosphere*, *golub* e *Parkisons*. O algoritmo Perceptron alcançou os melhores resultados nos conjuntos de dados *Indian Liver Patient* e *Fertility Diagnosis*. Varios modelos conseguiram separar o conjunto de dados *Banknote* com 100% de acurácia, demonstrando que esse conjunto de dados é facilmente separável. O modelo Hebbiano utilizando Kernels alcançou melhores resultados do que o ELM Hebbiano na maioria dos conjuntos de dados, demonstrando que essa abordagem é promissora.

REFERÊNCIAS

[1] E. G. Horta, "Aplicação de máquinas de aprendizado extremo ao problema de aprendizado ativo," 2015.

[2] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[3] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.

[4] D. Krstajic, L. J. Buturovic, D. E. Leahy, and S. Thomas, "Cross-validation pitfalls when selecting and assessing regression and classification models," *Journal of cheminformatics*, vol. 6, no. 1, pp. 1–15, 2014.

[5] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[6] H. Anton and R. C. Busby, *Contemporary linear algebra*. Wiley, 2003.

[7] J. A. Hertz, *Introduction to the theory of neural computation*. CRC Press, 2018.

[8] B. W. Silverman, *Density estimation for statistics and data analysis*, vol. 26. CRC press, 1986.

[9] M. V. Menezes, L. C. Torres, and A. P. Braga, "Width optimization of rbf kernels for binary classification of support vector machines: A density estimation-based approach," *Pattern Recognition Letters*, vol. 128, pp. 1–7, 2019.

[10] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[12] M. M. RAMADHAN, I. S. SITANGGANG, F. R. NASUTION, and A. GHIFARI, "Parameter tuning in random forest based on grid search method for gender classification based on voice frequency," *DEStech Transactions on Computer Science and Engineering*, no. cece, 2017.

[13] "Cross-validation: evaluating estimator performance."

[14] J. Fan, S. Upadhye, and A. Worster, "Understanding receiver operating characteristic (roc) curves," *Canadian Journal of Emergency Medicine*, vol. 8, no. 1, pp. 19–20, 2006.

[15] C. X. Ling, J. Huang, H. Zhang, *et al.*, "Auc: a statistically consistent and more discriminating measure than accuracy," in *Ijcai*, vol. 3, pp. 519–524, 2003.

[16] P. D. Bridge and S. S. Sawilowsky, "Increasing physicians' awareness of the impact of statistics on research outcomes: comparative power of the t-test and wilcoxon rank-sum test in small samples applied research," *Journal of clinical epidemiology*, vol. 52, no. 3, pp. 229–235, 1999.
[17] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, *et al.*, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *science*, vol. 286, no. 5439, pp. 531–537, 1999.
[18] S. Roweis, "Em algorithms for pca and spca," *Advances in neural information processing systems*, pp. 626–632, 1998.
[19] L. C. Torres, C. L. Castro, F. Coelho, and A. P. Braga, "Large margin gaussian mixture classifier with a gabriel graph geometric representation of data set structure," *IEEE transactions on neural networks and learning systems*, 2020.