Comparação Entre Métodos de Treinamento de Redes Neurias Artificiais em Problemas Práticos

Felipe Bartelt, *2016026841*

***Abstract*—This paper seeks to compare the performance of MLP-type artificial neural networks, with and without regularization, and RBF networks in data sets that represent practical problems. We sought to keep the MLP network without regularization as close as possible to the regularized MLP, taking for the former the same number of optimal neurons obtained for the regularized network, through the same 10-fold cross-validation used to determine the regularization parameter. Thus, the objective is to compare both the performance of the three networks and to analyze the difference in behavior induced by the addition of the regularization parameter in MLP networks. For the whole process, from data acquisition to training and testing, the Python library `scikit-learn` was used. At the end, the best performance of the regularized MLP and the difference in performance of the RBF network between classification and regression problems are shown.**

***Index Terms*—MLP, RBF, neural networks, regularization.**

***Resumo*—Esse artigo busca comparar o desempenho de redes neurais artificiais do tipo MLP, com e sem regularização, e redes RBF em conjuntos de dados que representam problemas práticos. Buscou-se manter a rede MLP sem regularização o mais próximo possível da MLP regularizada, tomando-se para a primeira o mesmo número de neurônios ótimo obtido para a rede regularizada, por meio da mesma validação cruzada em 10 dobras utilizada para determinação do parâmetro de regularização. Dessa forma, busca-se comparar tanto o desempenho das três redes, quanto analisar a diferença de comportamento induzida pela adição do parâmetro de regularização em redes MLP. Para todo o processo, desde a aquisição de dados ao treinamento e teste, utilizou-se a biblioteca de Python `scikit-learn`. Ao final, mostra-se o melhor desempenho da MLP regularizada e a diferença de desempenhos da rede RBF entre problemas de classificação e regressão.**

***Index Terms*—MLP, RBF, redes neurais, regularização.**

I. Introdução

ATUALMENTE, vive-se na era do Big Data, isso é, a cada segundo são transmitidas inúmeras informações, que são armazenadas em bancos de dados. Por exemplo, aproximadamente 500 horas de vídeo são carregados para o YouTube a cada minuto [1] e é estimado que seu banco de dados tenha no mínimo 45 petabytes de dados em formato de vídeo [2]. Dessa forma, não é difícil imaginar que a análise de dados se torna cada vez mais difícil. Apenas ter dados não significa nada, é necessário saber como utilizar toda essa informação. Continuando com o exemplo do YouTube, como a empresa poderia identificar automaticamente infringimento de direitos autorais e vídeos que não seguem a política da empresa senão utilizando métodos computacionais.

Para tratar essa enorme quantidade de dados, utiliza-se o aprendizado de máquina. O aprendizado de máquina pode ser dividido em três categorias [3]: aprendizado supervisionado, não supervisionado e aprendizado por reforço. No aprendizado supervisionado, tem-se acesso às respostas esperadas para o modelo em algumas amostras, podendo assim ser utilizado para problemas de classificação, diferenciar e-mails relevantes de *spam*, ou problemas de regressão, como estimar o comportamento de criptomoedas. Para o aprendizado não supervisionado, não há conhecimento das saídas esperadas para o modelo, dessa forma, é bastante utilizado para reconhecimento de padrões, permitindo agrupar dados similares e identificar anomalias, por exemplo, dado amostras de de um vírus, identificar automaticamente variantes. O aprendizado por reforço busca aprender como agir em certas situações por meio de um modelo de recompensa e punição, assim tipicamente se utiliza processos de decisão de Markov.

Infelizmente a aplicação de aprendizado de máquina requer vasto conhecimento, não sendo suficiente reaplicar modelos já treinados em uma aplicação em um outro tipo de processo, sendo necessário então que se compreenda a aplicabilidade dos diversos método de treinamento para cada tipo de aprendizado. Dessa forma, o intuito desse trabalho é comparar modelos de aprendizado supervisionado para problemas de classificação e regressão. Escolheu-se o multilayer perceptron, que é uma das redes mais utilizadas atualmente, e as redes de função de base radial, que permitem que dados sejam analisados por meio de funções como a distribuição gaussiana, para serem comparadas. Ainda, pretende-se treinar um multilayer perceptron com regularização, de forma a comparar a diferença de comportamento induzida por esse método em comparação com um multilayer perceptron de mesmo tamanho e com uma rede de função de base radial.

II. Revisão de Literatura

REDES neurais do tipo MLP (MultiLayer Perceptron) são bastante utilizadas na literatura, devido à sua grande versatilidade. Ao se utilizar mais de uma camada intermediária, permite-se tornar diversor problemas não lineares em problemas linearmente separáveis, dessa forma sendo bons classificadores e excelentes métodos de regressão e previsão. A atualização dos pesos dessa rede é feita pelo método de *backpropagation*, que é uma forma de corrigir os pesos da rede de forma retroativa, sendo os ”erros” de camadas posteriores levados em consideração para modificações de pesos de camadas anteriores. O método de otimização mais comum para o *backpropagation* é uma generalização do gradiente descendente, dada por (1).

$$\mathbf{w}_{li} = \mathbf{w}_{li} + \eta \left(\sum_{j=1}^{m} \delta_{kj} w_{ij} \right) h_i'(u_{ki}) x_l, \quad (1)$$

Onde $\mathbf{w}_{li}$ é o peso do neurônio i da camada l, η o passo do gradiente descendente, h_i' a derivada da função de ativação do neurônio atual i, x_l a variação de entrada l das amostras x e o termo entre parênteses corresponde ao termo de *back-propagation*, onde δ_{kj}, w_{ij} são, respectivamente, os erros das camadas posteriores e os respectivos pesos à ele associados. Dessa forma, a equação pode ser reduzida à uma forma mais simples (2).

$$\mathbf{w}_{li} = \mathbf{w}_{li} + \eta e_i h_i'(u_{ki}) x_l \quad (2)$$

As funções de ativação das camadas intermediárias dessas redes são tipicamente funções sigmoidais, unipolares ou bipolares. Sendo a função de ativação da camada saída dependente do objetivo da rede. Comumente se utiliza funções sigmoidais para problemas de classificação e a função identidade para problemas de regressão.

Uma técnica bastante utilizada para essas redes, mas também utilizada para as demais, é a regularização. Esse método consiste em uma forma de penalizar pesos grandes, forçando cada peso da rede a ter influência similar. Sabe-se que o problema de *overfitting* pode, em várias ocasiões, ser corrigido ao se selecionar variáveis relevantes para o problema, ignorando as com pouca relevância, porém, caso seja desejado utilizar todas as variáveis do conjunto, seja por não haver muitas ou então por se desejar que cada variável tenha uma importância para a previsão de um resultado final, utiliza-se a regularização.

A regularização então nada mais é que a adição de um novo hiperparâmetro λ, que permite restringir os peos da rede de uma forma que evite *overfitting*, porém, uma vez que ele penaliza os pesos, um valor muito alto de regularização pode ocasionar em *underfitting* da rede. Dessa forma, a seleção desse hiperparâmetro é feita com base em validação cruzada. Tomando-se como referência a Equação 2, pode-se tomar a equação de atualização dos pesos da rede com regularização por (3). O parâmetro de regularização também é levado em consideração pela função de custo.

$$\mathbf{w}_{li} = \mathbf{w}_{li} + \eta \left(e_i h_i'(u_{ki}) x_l + \lambda \mathbf{w}_{li} \right) \quad (3)$$

Redes neurais do tipo RBF (*Radial Basis Functions Neural Networks* - RBFNNs) [4] são redes de camada intermediária única, caracterizadas por funções radiais em seus neurônios, cujas respostas são combinadas linearmente para a obtenção da saída da rede. De maneira geral uma RBFNN pode ser descrita pela Equação 4

$$f(\mathbf{x}, \theta) = \sum_{i=1}^{p} w_i h_i(\mathbf{x}, \mathbf{z}_i) + w_0, \quad (4)$$

onde $\mathbf{w} = [w_0, w_1, \ldots, w_p]$ é o vetor de parâmetros do neurônio de saída, $\mathbf{z}_i$ é o vetor que contém todos os parâmetros da função de ativação radial do neurônio i da camada intermediária e $\theta = [\mathbf{w}, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_p]$ o vetor que contém a concatenação de todos os vetores de parâmetros da rede.

Normalmente, representa-se redes RBF por meio de funções Normais. Dessa forma, pode-se representar a resposta de um neurônio i da camada intermediária com base na matriz de covariância Σ_i e vetor de médias μ_i, conforme a Equação 5.

$$h_i(\mathbf{x}, \mathbf{z}_i) = \frac{1}{\sqrt{(2\pi)^n |\Sigma_i|}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1} (\mathbf{x}-\mu_i)\right)}, \quad (5)$$

onde n é o número de variáveis da entrada.

Considerando-se independência entre as variáveis, pode-se modificar a Equação 5 a forma da Equação 6.

$$h_i(\mathbf{x}, \mathbf{z}_i) = \frac{1}{\sqrt{(2\pi)^n |r^2 \mathbf{I}|}} e^{\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T (r^2\mathbf{I})^{-1} (\mathbf{x}-\mu_i)\right)}, \quad (6)$$

onde $\mathbf{I}$ é a matriz identidade de dimensão apropriada e r os desvios padrão das variáveis, chamado de raio da função.

Nota-se que os parâmetros de uma rede RBF são determinados, de certa forma, a priori. Necessita-se, além da definição do número de centros, de um método para alocação de centros e definição dos raios das funções radiais à ela pertencentes.

Um dos métodos mais comuns e, muitas vezes, menos custoso computacionalmente, é a alocação dos centros das RBFNNs por meio do método de *clustering* `k-means` [5], tomando-se os raios por meio da matriz de covariância das variáveis ou por meio de distâncias relativas entre os centros definidos ou distância entre centros e amostras.

III. Metodologia

BUSCANDO realizar uma comparação entre os modelos MLP e RBF e o comportamento obtido pela regularização de MLPs, testou-se as respostas desses modelos em dois conjuntos de regressão e dois conjuntos de classificação.

Todas as redes treinadas foram obtidas por meio do pacote de Python `scikit-learn`, assim como os conjuntos de dados. Um resumo das características dos *datasets* utilizados é apresentado na Tabela I. Os conjuntos de dados utilizados são bastante reconhecidos e utilizados para estudos, são esses: Boston Housing, cujo objetivo é aproximar preços medianos de casas, em mil dólares, por meio de variáveis obtidas em Boston; Diabetes, cujo obtivo é prever o progresso da diabetes por meio de medições de saúde dos pacientes, sendo essas normalizadas por meio de *standardization*; Breast Cancer Wisconsin (diagnostic), com objetivo de classificar se um tumor é benigno ou maligno baseado em 30 características do tumor, apresentando mais exemplos benignos do que malignos (145 amostras a mais); Iris, que fornece 4 caracterísitcas de plantas iris, de forma a se classificar se a planta pertence à espécie Setosa, Virginica ou Versicolour, apresentando assim um conjunto de dados com 33% de cada classe. Ao longo do texto, utilizar-se-á as abreviações apresentadas na Tabela I.

Para as comparações serem mais justas, realizou-se pré processamento mínimo. Utilizando-se normalização do tipo *standardization* para os conjuntos BH e BC, uma vez que os dados de DT já estão normalizados desta forma e os

Tabela I: Características dos *datasets*

Dataset	Abreviação	N° amostras	Dimensão	Tipo
Boston Housing	BH	506	13	Regressão
Diabetes	DT	442	10	Regressão
Breast Cancer	BC	569	30	Classificação
Iris	IR	150	4	Classificação

dados de IR não apresentam ordens de grandeza, ou limites, discrepantes.

Os aproximadores e regressores gaussianos do `scikit-learn` se otimizam automaticamente, ou seja, eles têm, de forma intrínseca, uma validação cruzada para a escolha dos melhores raios e centros da rede, que no caso é feito por RBFs, além do número ótimo de neurônios. Dessa forma, tomou-se como premissa encontrar, por meio de validação cruzada em 10 dobras, o melhor número de neurônios e parâmetro de regularização para a rede MLP. De forma a se ter uma comparação da eficácia da regularização, a rede MLP padrão, ou seja, sem regularização, teve como número de neurônios, para todos os conjuntos de dados, o mesmo número encontrado como ótimo para a rede regularizada. Apesar disso não representar o máximo de precisão de uma rede MLP sem regularização, isso permite com que o efeito desse método seja estudado muito mais facilmente.

Para todos os conjuntos de dados e todas as redes, utilizou-se as mesmas diretrizes de treinamento. A otimização da função de custo foi obtida pelo método LBFGS, ou seja, o método BFGS de memória limitada, que por ser um método quasi-Newton, remove a necessidade de escolha de passo do algoritmo; tomou-se como 2500 o número máximo de iterações do otimizador e os demais parâmetros foram deixados como os *default* da biblioteca, como por exemplo a tolerância de erro de 10^{-4}.

De forma a se comparar a eficácia dos métodos para cada conjunto de dados, utilizou-se o erro quadrático médio (MSE) como métrica para os conjuntos de regressão e a acurácia como métrica para os conjuntos de classificação. Para cada conjunto, realizou-se 50 iterações, de forma a se obter média e desvio padrão das métricas, assim como seu melhor resultado. O conjunto de dados foi separado em conjunto de treino e conjunto de teste, com uma proporção $70 - 30\%$, de forma aleatória a cada iteração. O mesmo conjunto de treino gerado foi utilizado para treinar todas as redes, assim como o conjunto de testes foi o mesmo utilizado para as três.

IV. Resultados

POR meio dos conjuntos de dados e métricas apresentadas, obteve-se como resultado comparativo os MSEs médios e desvios padrão, para os conjuntos de regressão, e as acurácias médias e desvios padrão, para os conjuntos de classificação. Obteve-se também os melhores resultados de cada rede, ou seja, o menor MSE médio e a maior acurácia média. Esses valores são apresentados por meio de diagramas de caixa, que incluem representação de média (linha pontilhada) e desvio padrão (losango pontilhado), além de tabelas que apresentam os resultados numéricos de forma concisa.

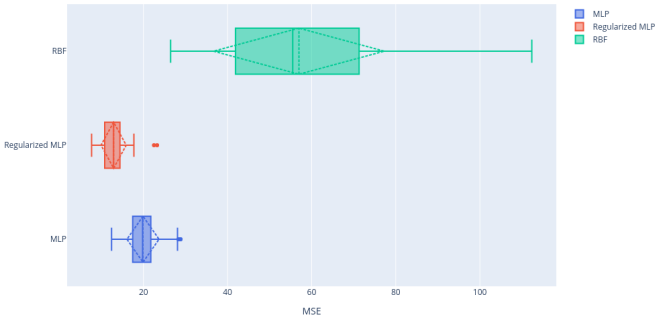

Figura 1: Distribuição de MSEs para o *dataset* Boston Housing

Figura 2: Últimas respostas obtidas por cada modelo e resposta esperada para o conjunto BH

Ainda, de forma a fornecer uma melhor visualização do comportamento das respostas das redes, plotou-se gráficos de dispersão das respostas obtidas pelas redes em comparação com as respostas esperadas, obtidos na última iteração do treinamento. Para os conjuntos de classificação, obteve-se as matrizes de confusão resultantes da última iteração de treinamento.

Como forma de abreviação, chamar-se-á MLP o perceptron de múltiplas camadas sem regularização, MLP-Reg o perceptron com regularização e RBF a rede de função de base radial.

Para o conjunto de dados BH, nota-se um desempenho muito superior das redes MLP em comparação com a rede RBF, Figura 1. O erro quadrático médio obtido pela RBF é muito maior e apresenta menor precisão, sendo o erro da MLP-Reg o menor dentre todos os outros. Os resultados médios e mínimos obtidos pelos modelos é apresentado na Tabela II.

É possível comparar o desempenho das redes ao se plotar a resposta esperada e as respostas obtidas por cada modelo, Figura 2, permitindo-se assim compreender as diferenças entre as aproximações dos métodos nesse conjunto de dados.

O pior desempenho da rede RBF é provavelmente devido à aproximação feita por essa rede. Pode-se supor que o comportamento da resposta esperada não é próximo gaussiano, sendo assim de difícil aproximação por essa rede.

Para o conjunto de dados DT, nota-se, na Figura 3, que todos os MSE obtidos tem valores elevados, porém, novamente, o desempenho da rede RBF foi inferior à ambas as redes MLP, apresentando ainda precisão muito menor. A MLP-Reg obteve,

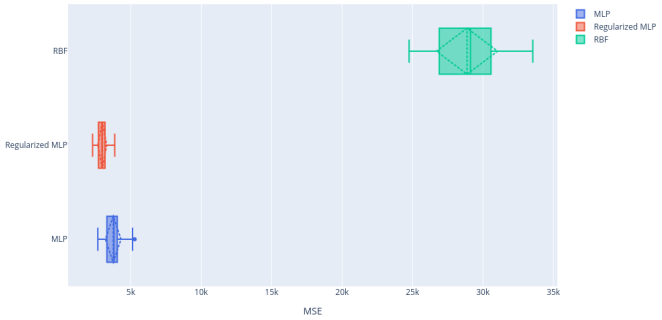

Figura 3: Distribuição de MSEs para o *dataset* Diabetes

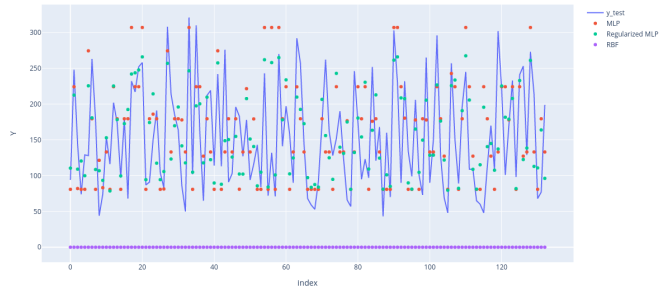

Figura 4: Últimas respostas obtidas por cada modelo e resposta esperada para o conjunto DT

Tabela II: Erros quadráticos médios obtidos pelos métodos avaliados dentre os *datasets* de regressão

Dataset	Método	MSE médio	MSE mínimo
BH	MLP	19.890 ± 3.810	12.421
	MLP-Reg	12.910 ± 2.992	7.667
	RBF	56.968 ± 20.304	26.461
DT	MLP	3751.998 ± 552.198	2655.670
	MLP-Reg	2975.952 ± 317.261	2286.383
	RBF	$28\,869.927 \pm 2170.585$	$24\,755.677$

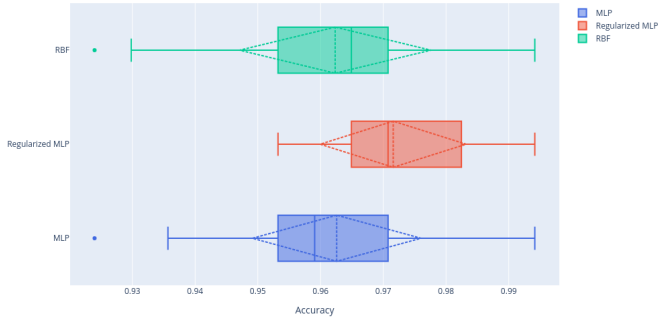

Figura 5: Distribuição de acurácias para o *dataset* Breast Cancer

novamente, o menor dos erros médios. Os resultados médios e mínimos obtidos pelos modelos é apresentado na Tabela II. Ao se avaliar o gráfico da Figura 4, percebe-se que a rede RBF convergiu a só prever uma estagnação da doença, a rede MLP-Reg consegue acompanhar os resultados de forma melhor e ainda se percebe que a rede MLP tende a subestimar ou superestimar bastante a evolução da diabetes, raramente tendo resposta coincidente à esperada.

Pode-se supor novamente que o desempenho da RBF é devido ao mesmo fenômeno, as respostas esperadas não seguem um comportamento próximo de uma gaussiana, dificultando a aproximação por meio dessa rede. Entretanto, visto que todos os desempenhos foram longe de ideias, espera-se que ao se realizar pré-processamento mais atencioso, seja possível melhorar bastante as respostas dos modelos, sendo este então um conjunto de difícil regressão sem um conhecimento maior da base de dados.

Nota-se para o conjunto de dados BC que a rede RBF teve desempenho muito melhor do que o obtido para os problemas de regressão, Figura 5, obtendo acurácia competitiva com a acurácia da MLP. Para todas as redes, obteve-se acurácia excelente, maior, em média, que 96%, sendo todas as máximas iguais à 99.415%, Tabela III. Ainda, percebe-se novamente que a MLP-Reg obteve desempenho superior às demais.

Ao se avaliar as matrizes de confusão, pode-se notar que a MLP, ao menos para essa última iteração, teve maior dificuldade em evitar falsos positivos, Figura 6, enquanto a RBF teve maior dificuldade em evitar falsos negativos, Figura 8. Isso pode indicar que uma rede MLP é mais recomendada que a RBF, ao menos da forma como foi treinada, para o problema de classificação de câncer de mama, uma vez que é preferível um falso positivo a um falso negativo nesse caso. A MLP-Reg apresentou apenas um falso positivo e um falso negativo, Figura 7, indicando que para essa iteração a rede não tem nenhum viés para classes. Pode-se supor que o excelente comportamento da rede RBF se deve à separabilidade gaussiana do problema.

O conjunto de dados IR é mais simples que o anterior, em questão de dimensionalidade, porém as acurácias obtidas por todas as redes foram inferiores às obtidas no BC, o que pode ser explicado pela difícil separação entre as classes Versicolor e Virginica. Nota-se por meio da Figura 9 e Tabela III que a rede RBF teve desempenho praticamente igual à MLP, apresentando apenas um desvio padrão pouco maior. Percebe-se novamente um melhor desempenho da MLP-Reg, que apesar de apresentar acurácia média pouco melhor que as demais, tem uma precisão maior de resultados. Todas as redes apresentaram acurácia máxima de 100%.

Ao se avaliar as matrizes de confusão Figuras 10, 11 e 12, nota-se exatamente o problema relatado, a maior dificuldade de se separar a classe Versicolor da classe Virginica. Entretanto essa dificuldade se traduz em apenas uma classificação errada de Virginica como Versicolor para as redes MLP e MLP-Reg e duas dessa mesma classificação errada para a rede RBF. De toda forma, o comportamento de todas as três redes foi excelente.

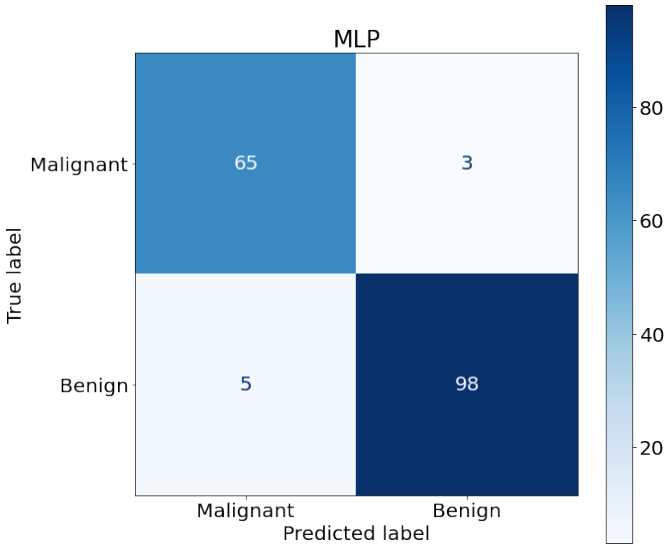

Figura 6: Matriz de confusão da rede MLP, na última iteração, para o *dataset* Breast Cancer

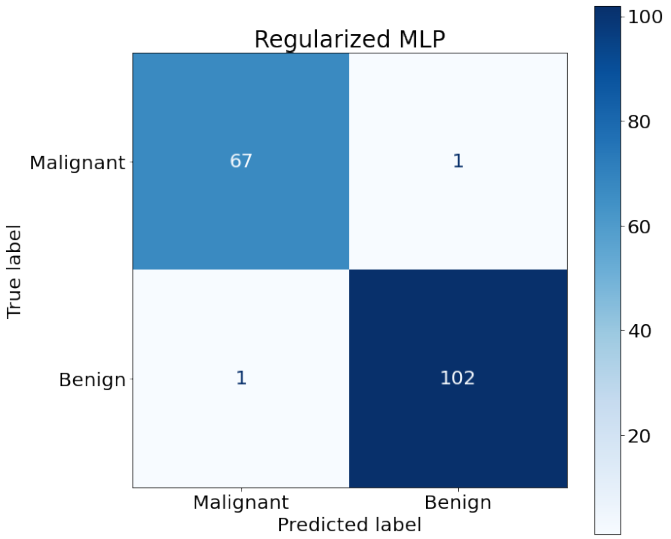

Figura 7: Matriz de confusão da rede MLP-Reg, na última iteração, para o *dataset* Breast Cancer

Tabela III: Acurácias obtidas pelos métodos avaliados dentre os *datasets* de classificação

Dataset	Método	Acurácia média	Acurácia máxima
BC	MLP	$(96.257 \pm 1.344)\%$	99.415%
	MLP-Reg	$(97.158 \pm 1.164)\%$	99.415%
	RBF	$(96.234 \pm 1.526)\%$	99.415%
IR	MLP	$(95.733 \pm 2.698)\%$	100%
	MLP-Reg	$(96.933 \pm 2.079)\%$	100%
	RBF	$(95.733 \pm 2.585)\%$	100%

V. Conclusão

De forma geral, percebeu-se que a rede RBF não é ideal para quaisquer problemas de regressão, tendo apresentado para os conjuntos analisados um desempenho muito infe-

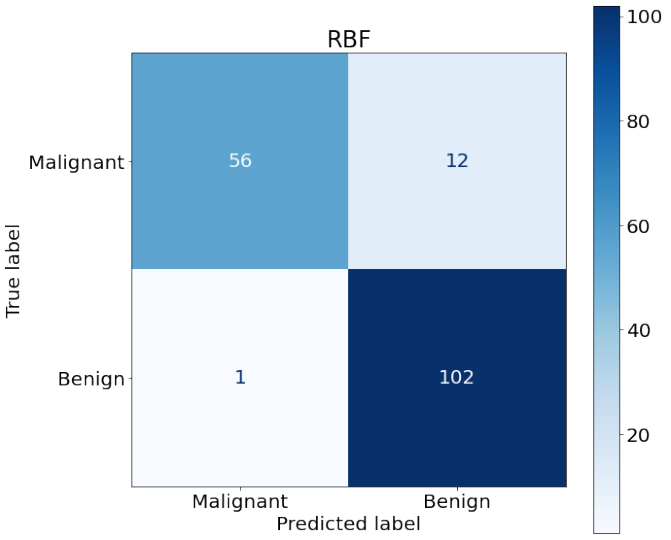

Figura 8: Matriz de confusão da rede RBF, na última iteração, para o *dataset* Breast Cancer

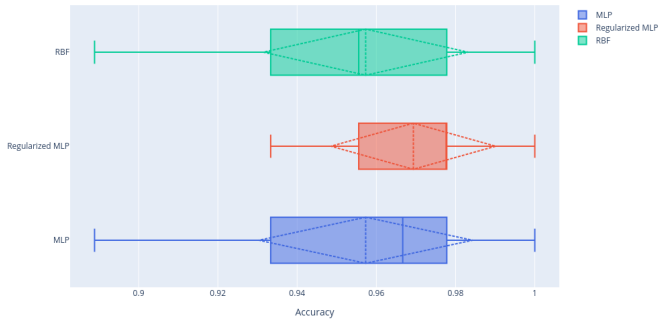

Figura 9: Distribuição de acurácias para o *dataset* Iris

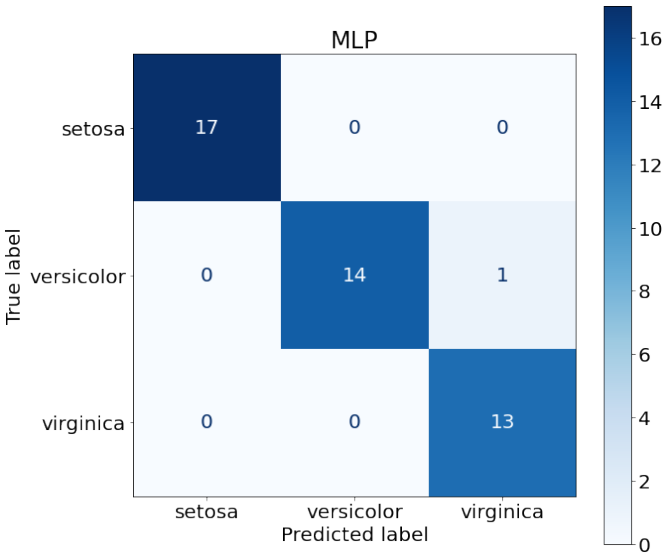

Figura 10: Matriz de confusão da rede MLP, na última iteração, para o *dataset* Iris

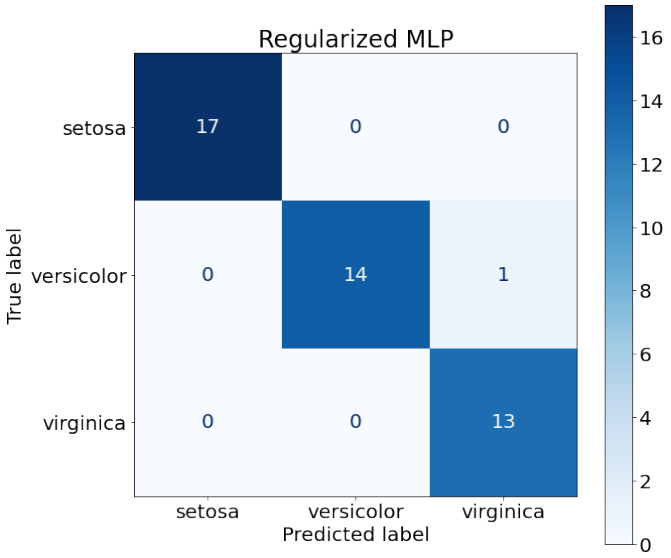

Figura 11: Matriz de confusão da rede MLP-Reg, na última iteração, para o *dataset* Iris

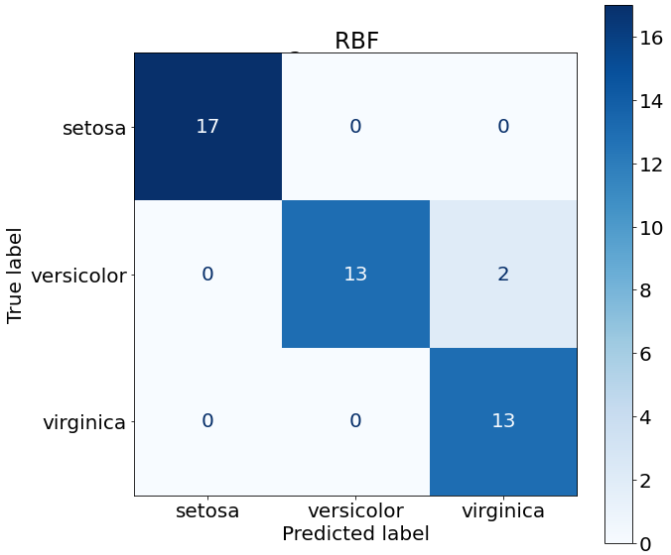

Figura 12: Matriz de confusão da rede RBF, na última iteração, para o *dataset* Iris

rior às respostas da MLP e MLP-Reg. Porém, para problemas de classificação a rede RBF se apresenta muito boa, obtendo acurácias tão boas quanto a rede MLP.

O uso de regularização se mostrou excelente, tornando o desempenho da MLP o melhor para todos os conjuntos avaliados. Ressalta-se que é possível uma MLP sem regularização atingir desempenho análogo, porém foi preferido manter o número de neurônios da MLP igual ao da MLP-Reg, com o intuito de ser possível tomar essa comparação de resultados. Assim, o uso de regularização é bastante interessante, uma vez que permite melhores resultados sendo apenas um hiperparâmetro a mais, podendo assim ser determinado junto aos demais por meio da validação cruzada.

Em suma, esse estudo permitiu a visualização do motivo de redes MLP serem tão utilizadas na literatura. Com a possibilidade de adicionar um parâmetro de regularização, essa rede se torna passível de ter desempenho excelente para conjuntos diversos de classificação, regressão ou previsão, exibindo grande versatilidade. Entretanto essa possibilidade não anula a possibilidade de uso de redes RBF. Problemas com distribuição gaussiana, ou próximo disso, são possíveis de serem aproximados por redes RBF de forma menos custosa, cabendo assim avaliar a arquitetura de rede ótima para um dado problema. Com isso, nota-se também a importância do conhecimento do conjunto de dados, que pode fornecer características importantes para decisões de projeto, como a importância de variáveis, necessidade de pré-processamento dos dados, arquitetura da rede, etc. Cabe assim ao projetista ter conhecimento dos dados antes de começar o projeto de redes neurais, evitando maior quantidade de tentativas falhas ao se tomar decisões genéricas demais para o problema em questão.

REFERÊNCIAS

[1] "Youtube at 15: My personal journey and the road ahead," blog.youtube. [Online]. Available: https://blog.youtube/news-and-events/youtube-at-15-my-personal-journey

[2] "Top 10 largest databases in the world," www.comparebusinessproducts.com. [Online]. Available: https://www.comparebusinessproducts.com/fyi/10-largest-databases-in-the-world

[3] K. P. Murphy, *Machine learning : a probabilistic perspective*. Mit Press, 2012.

[4] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[5] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 03 1982. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1056489

APÊNDICE

Algoritmo 1: Script utilizado para treinar as redes e gerar as imagens

```
"""
O seguinte script foi utilizado em um jupyter notebook, dessa forma, apenas concatenou-se todas as celulas
utilizadas em um codigo para que fosse adicionado ao trabalho.

"""

import numpy as np
import plotly.express as px
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.gaussian_process import GaussianProcessRegressor, GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, plot_confusion_matrix
from matplotlib import pyplot as plt
import plotly.graph_objects as go
from sklearn.datasets import (
    load_boston,
    load_diabetes,
    load_breast_cancer,
    load_iris
)
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV

# Boston Housing
X, y = load_boston(return_X_y=True)
standardize = StandardScaler()
X = standardize.fit_transform(X)
mlp_reg = MLPRegressor(hidden_layer_sizes = 13, activation='logistic', solver='lbfgs', max_iter=2500)
mlp_regR = MLPRegressor(hidden_layer_sizes = 13, activation='logistic', solver='lbfgs', max_iter=2500,
    ↪ alpha=10)
rbf = GaussianProcessRegressor(RBF())
mse_mlp, mse_mlpR, mse_RBF = [], [], []

for _ in range(50):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

    mlp_reg.fit(x_train, y_train)
    y_hat = mlp_reg.predict(x_test)
    mlp_regR.fit(x_train, y_train)
    y_hatR = mlp_regR.predict(x_test)
    rbf.fit(x_train, y_train)
    y_hatRBF = rbf.predict(x_test)

    mse_mlp.append(mean_squared_error(y_hat, y_test))
    mse_mlpR.append(mean_squared_error(y_hatR, y_test))
    mse_RBF.append(mean_squared_error(y_hatRBF, y_test))

fig = go.Figure()
fig.add_trace(go.Box(x=mse_mlp, name='MLP', marker_color='royalblue', boxmean='sd'))
fig.add_trace(go.Box(x=mse_mlpR, name='Regularized MLP', boxmean='sd'))
fig.add_trace(go.Box(x=mse_RBF, name='RBF', boxmean='sd'))
fig.update_layout(xaxis_title='MSE', width=1000, height=600)
fig.show(renderer = 'png', width=1000, height=600)

fig = px.line()
fig.add_scatter(x=np.arange(152), y=y_test, name='y_test')
fig.add_scatter(x=np.arange(152), y=y_hat, name='MLP', mode='markers')
fig.add_scatter(x=np.arange(152), y=y_hatR, name='Regularized MLP', mode='markers')
fig.add_scatter(x=np.arange(152), y=y_hatRBF, name='RBF', mode='markers')
fig.update_layout(xaxis_title = 'index', yaxis_title = 'Y')
fig.show(renderer = 'png', width=1400, height = 600)

print(np.min(mse_mlp), np.mean(mse_mlp), np.std(mse_mlp))
print(np.min(mse_mlpR), np.mean(mse_mlpR), np.std(mse_mlpR))
print(np.min(mse_RBF), np.mean(mse_RBF), np.std(mse_RBF))

#Diabetes
X, y = load_diabetes(return_X_y=True)
mlp_reg2 = MLPRegressor(activation='logistic', solver='lbfgs', max_iter=2500)
```

```
mlp_reg = MLPRegressor(hidden_layer_sizes = 9, activation='logistic', solver='lbfgs', max_iter=2500)
mlp_regR = MLPRegressor(hidden_layer_sizes = 9, activation='logistic', solver='lbfgs', max_iter=2500, alpha
    ↪ =10)
rbf = GaussianProcessRegressor(RBF())
mse_mlp, mse_mlpR, mse_RBF = [], [], []

for _ in range(50):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

    mlp_reg.fit(x_train, y_train)
    y_hat = mlp_reg.predict(x_test)
    mlp_regR.fit(x_train, y_train)
    y_hatR = mlp_regR.predict(x_test)
    rbf.fit(x_train, y_train)
    y_hatRBF = rbf.predict(x_test)

    mse_mlp.append(mean_squared_error(y_hat, y_test))
    mse_mlpR.append(mean_squared_error(y_hatR, y_test))
    mse_RBF.append(mean_squared_error(y_hatRBF, y_test))

fig = go.Figure()
fig.add_trace(go.Box(x=mse_mlp, name='MLP', marker_color='royalblue', boxmean='sd'))
fig.add_trace(go.Box(x=mse_mlpR, name='Regularized MLP', boxmean='sd'))
fig.add_trace(go.Box(x=mse_RBF, name='RBF', boxmean='sd'))
fig.update_layout(xaxis_title='MSE', width=1000, height=600)
fig.show(renderer = 'png', width=1000, height=600)

import pandas as pd
df = pd.DataFrame()
df['x'] = np.arange(133)
df['y_test'] = y_test
fig = px.line()
fig.add_scatter(x=np.arange(133), y=y_test, name='y_test')
fig.add_scatter(x=np.arange(133), y=y_hat, name='MLP', mode='markers')
fig.add_scatter(x=np.arange(133), y=y_hatR, name='Regularized MLP', mode='markers')
fig.add_scatter(x=np.arange(133), y=y_hatRBF, name='RBF', mode='markers')
fig.update_layout(xaxis_title = 'index', yaxis_title = 'Y')
fig.show(renderer = 'png', width=1200, height = 600)

print(np.mean(mse_mlp), np.std(mse_mlp), np.min(mse_mlp))
print(np.mean(mse_mlpR), np.std(mse_mlpR), np.min(mse_mlpR))
print(np.mean(mse_RBF), np.std(mse_RBF), np.min(mse_RBF))


#Breast Cancer

X, y = load_breast_cancer(return_X_y=True)
standardize = StandardScaler()
X = standardize.fit_transform(X)

mlp_clf = MLPClassifier(hidden_layer_sizes = 20, activation='logistic', solver='lbfgs', max_iter=2500)
mlp_clfR = MLPClassifier(hidden_layer_sizes = 20, activation='logistic', solver='lbfgs', max_iter=2500,
    ↪ alpha=0.1)
rbf = GaussianProcessClassifier(RBF())
mse_mlp, mse_mlpR, mse_RBF = [], [], []

for _ in range(50):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

    mlp_clf.fit(x_train, y_train)
    y_hat = mlp_clf.predict(x_test)
    mlp_clfR.fit(x_train, y_train)
    y_hatR = mlp_clfR.predict(x_test)
    rbf.fit(x_train, y_train)
    y_hatRBF = rbf.predict(x_test)

    mse_mlp.append(mlp_clf.score(x_test, y_test))
    mse_mlpR.append(mlp_clfR.score(x_test, y_test))
    mse_RBF.append(rbf.score(x_test, y_test))

fig = go.Figure()
fig.add_trace(go.Box(x=mse_mlp, name='MLP', marker_color='royalblue', boxmean='sd'))
fig.add_trace(go.Box(x=mse_mlpR, name='Regularized MLP', boxmean='sd'))
fig.add_trace(go.Box(x=mse_RBF, name='RBF', boxmean='sd'))
fig.update_layout(xaxis_title='Accuracy', width=1000, height=600)
fig.show(renderer = 'png', width=1000, height=600)
```

```
fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(10,10))
plot_confusion_matrix(rbf, x_test, y_test, cmap='Blues', display_labels = ['Malignant', 'Benign'], ax=axes)
plt.rcParams.update({'font.size': 20})
axes.title.set_text('RBF')
plt.show()

print(np.mean(mse_mlp), np.std(mse_mlp), np.max(mse_mlp))
print(np.mean(mse_mlpR), np.std(mse_mlpR), np.max(mse_mlpR))
print(np.mean(mse_RBF), np.std(mse_RBF), np.max(mse_RBF))

#Iris

X, y = load_iris(return_X_y=True)

mlp_clf = MLPClassifier(hidden_layer_sizes = 6, activation='logistic', solver='lbfgs', max_iter=2500)
mlp_clfR = MLPClassifier(hidden_layer_sizes = 6, activation='logistic', solver='lbfgs', max_iter=2500,
    ↪ alpha=1e-06)
rbf = GaussianProcessClassifier(RBF())
mse_mlp, mse_mlpR, mse_RBF = [], [], []

for _ in range(50):
    x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3)

    mlp_clf.fit(x_train, y_train)
    y_hat = mlp_clf.predict(x_test)
    mlp_clfR.fit(x_train, y_train)
    y_hatR = mlp_clfR.predict(x_test)
    rbf.fit(x_train, y_train)
    y_hatRBF = rbf.predict(x_test)

    mse_mlp.append(mlp_clf.score(x_test, y_test))
    mse_mlpR.append(mlp_clfR.score(x_test, y_test))
    mse_RBF.append(rbf.score(x_test, y_test))

fig = go.Figure()
fig.add_trace(go.Box(x=mse_mlp, name='MLP', marker_color='royalblue', boxmean='sd'))
fig.add_trace(go.Box(x=mse_mlpR, name='Regularized MLP', boxmean='sd'))
fig.add_trace(go.Box(x=mse_RBF, name='RBF', boxmean='sd'))
fig.update_layout(xaxis_title='Accuracy', width=1000, height=600)
fig.show(renderer = 'png', width=1000, height=600)

fig, axes = plt.subplots(nrows=1, ncols=1, figsize=(10,10))
plot_confusion_matrix(rbf, x_test, y_test, cmap='Blues', display_labels = data.target_names, ax=axes)
plt.rcParams.update({'font.size': 20})
axes.title.set_text('Regularized MLP')
plt.show()

print(np.mean(mse_mlp), np.std(mse_mlp), np.max(mse_mlp))
print(np.mean(mse_mlpR), np.std(mse_mlpR), np.max(mse_mlpR))
print(np.mean(mse_RBF), np.std(mse_RBF), np.max(mse_RBF))
```