Projeto 1 - Desenvolvimento do Multilayer Perceptron(MLP) utilizando múltiplas camadas

Thauan Leandro Gonçalves

26 de setembro de 2018

1 Introdução

Neste trabalho iremos utilizar uma MLP contendo duas camadas intermediárias(Hidden Layers), sendo que a camada de entrada(Input Layer) possui L neurônios; as duas camadas intermediárias possuem K e J neurônios, respectivamente; e a camada de saída(Output Layer) possui I neurônios. Apesar de utilizarmos duas camadas intermediárias, os cálculos funcionarão para números diferentes.

A Figura 1 mostra a estrutura da MLP que iremos utilizar.

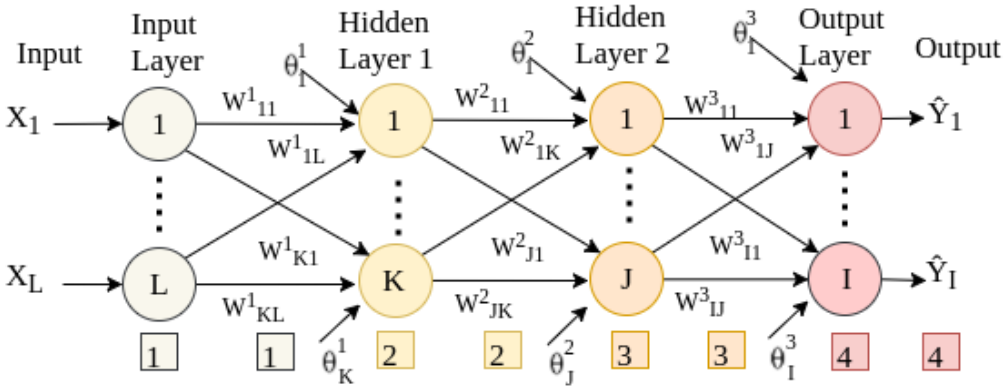

Figura 1: Estrutura da MLP com duas camadas intermediárias e números variáveis de neurônios.

Suponha que tenhamos um conjunto de dados representados por *Dataset* que contém n instâncias. Cada instância contém os dados de entrada, representados por $X = \{x_1, ..., x_L\}$, e os dados de saída que são representados por $Y = \{y_1, ..., y_I\}$. Reparamos que Y é diferente de $\hat{Y} = \{\hat{y}_1, ..., \hat{y}_I\}$, uma vez que os primeiros são os dados contidos no *Dataset* e os segundos são os valores obtidos pelo MLP, sendo portanto estimadores.

Os valores de w e θ representados na Figura 1 são os parâmetros do MLP. O principal objetivo do algoritmo é obter os valores adequados desses parâmetros de forma com que, a partir de um X de entrada, a estrutura possa fornecer um $\hat{Y}$ o mais próximo possível do valor esperado Y. Esse processo de se obter os valores corretos dos parâmetros é conhecido como treinamento do MLP, sendo um dos processos mais importantes junto com a fase de teste e validação do modelo.

2 Funcionamento do MLP

O MLP possui duas fases: a *Forward* e a *Backpropagation*. Essas fases serão descritas nas próximas duas seções.

2.1 Forward

O principal objetivo desta fase é propagar os valores de X, recebidos pela camada de entrada, até a última camada do MLP, obtendo assim os valores de $\hat{Y}$ que sejam os mais próximos possíveis dos

valores esperados Y. Essa propagação é realizada através da aplicação de duas funções que estão contidas nos neurônios de todas as camadas, exceto a camada de entrada. Cada neurônio destas camadas recebe um valor de entrada, que são provindos pelos neurônios que estão nas camadas anterios, aplica as duas funções e a propaga para as camadas da frente.

Essas funções são influenciadas diretamente pelos valores que chegam a ela, pelas camadas anteriores, quanto pelos seus parâmetros w e θ, de forma que é possível modificar esses parâmetros de forma que se propague um valor que melhore a aproximação de $\hat{Y}$ para Y.

Por exemplo, suponhamos que queiramos calcular os valores que irão ser propagados pelo neurônio K, na primeira camada escondida. A primeira função que este neurônio realizará será dada por 1, onde será obtido o valor de net_K^1[1] que é uma combinação linear dos dados provindos da camada anterior e os parâmetros $w_{K1}, ..., w_{KL}$ e θ_K^1. A segunda função será dada por 2, onde será aplicada a função sigmóide sobre o valor obtido pela função anterior. Esta função é conhecida como função de ativação, uma vez que ela mapea valores do $\mathbb{R}$ no intervalo de $(-1, 1)$.

$$net_K^1 = \sum_{l=1}^{L}(w_{Kl}^1 x_l) + \theta_K^1 \tag{1}$$

$$fnet_K^1 = \frac{1}{1 + e^{-net_K^1}} \tag{2}$$

O valor produzido pela função 2 será propagado para as próxima camadas, sendo realizadas as mesmas operações sobre os dados de entrada, mudando somente os valores que chegam ao neurônio e os parâmetros que cada um possui.

Da mesma forma como mostrado acima, as funções 3 e 4 mostram como obter o valor de $\hat{Y}_I$ no neurônio I, camada de saída.

$$net_I^3 = \sum_{j=1}^{J}(w_{Ij}^3 * fnet_j^2) + \theta_I^3 \tag{3}$$

$$\hat{Y}_I = fnet_I^3 = \frac{1}{1 + e^{-net_I^3}} \tag{4}$$

2.2 Backpropagation

A segunda parte do MLP é responsável pelas modificações dos parâmetros w e θ com base nos erros de aproximação obtidos de Y por $\hat{Y}$, realizando o caminho inverso em relação a fase *forward*. Para isso, definimos E^2 como a soma dos erros quadrados da diferença entre os valores obtidos $\hat{Y}$ e os valores esperados Y, representado em 5.

$$E^2 = \sum_{m=1}^{I}(y_m - \hat{y}_m)^2 \tag{5}$$

O objetivo será reduzir o valor de E^2 através da variação dos parâmetros. Desta forma, podemos utilizar a derivada E^2 em relação a todos os parâmetros do modelo, alterando-os no sentido inverso ao crescimento desta função, alterando os parâmetros de forma a reduzir o erro. Assim, podemos alterar os valores dos w e θ de acordo com 6 e 7, de forma com que em cada iteração do algoritmo o erro de aproximação diminua.

Dado as equações de variação de peso,o próximo passo será obter as derivadas do E^2 em relação aos parâmetros, o que será feito na próxima seção.

$$w^{t+1} = w^t - \eta\frac{\partial E^2}{\partial w} \tag{6}$$

$$\theta^{t+1} = \theta^t - \eta\frac{\partial E^2}{\partial \theta} \tag{7}$$

[1] Os valores subescritos e sobrescritos são somente para facilitar a compreensão, não sendo exponencias. O valor subescrito indica o número do neurônio na sua camada e o sobrescrito indica o número da camada.

3 Formulações matemáticas

3.1 Formulação das derivadas em relação aos parâmetros da camada de saída

O que vamos calcular nesta seção são as derivadas do E^2 em relação aos W^3_{ab} e θ^3_a, onde W^3_{ab} representa o parâmetro w que liga o neurônio a da camada de saída e o neurônio b da segunda camada escondida, e θ^3_a representa o parâmetro θ do neurônio a da camada de saída. Para isso, vamos usar o fato de que a derivada de $fnet$ em relação a net é dado pela função 8.

$$\frac{\partial fnet}{\partial net} = fnet(1 - fnet)^2 \tag{8}$$

Com base nas funções em 3, 4, 5 e 8,temos que:

$$\begin{aligned}
\frac{\partial E^2}{\partial w^3_{ab}} &= \frac{\partial \sum_{i=1}^{I}(y_i - \hat{y}_i)^2}{\partial w^3_{ab}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - \hat{y}_i)^2}{\partial w^3_{ab}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(net^3_i))^2}{\partial w^3_{ab}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j) + \theta^3_i))^2}{\partial w^3_{ab}} \\
&= \sum_{i=1}^{I} -2(y_i - \hat{y}_i) \frac{\partial fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j) + \theta^3_i)}{\partial w^3_{ab}} \text{[3]} \\
&= -2 \sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) \sum_{j=1}^{J} \frac{\partial (w^3_{ij} * fnet^2_j)}{\partial w^3_{ab}} \text{[4]} \\
&= -2(y_a - \hat{y}_a) fnet^3_a(1 - fnet^3_a) fnet^2_b
\end{aligned}$$

$$\frac{\partial E^2}{\partial w^3_{ab}} = -2(y_a - \hat{y}_a) fnet^3_a(1 - fnet^3_a) fnet^2_b \tag{9}$$

$$\frac{\partial E^2}{\partial \theta^3_a} = -2(y_a - \hat{y}_a) fnet^3_a(1 - fnet^3_a) 1 \tag{10}$$

A derivada do E^2 em relação ao θ^3_a é parecida com a mostrada em 9 exceto pelo último termo que invés de ser $fnet^2_b$ será 1, sendo mostrada em 10.

3.2 Formulação das derivadas em relação aos parâmetros da segunda camada intermediária

Inicialmente iremos obter as derivadas do E^2 em relação aos W^2_{bc} e θ^2_b, onde o W^2_{bc} representa o parâmetros w que liga o neurônio b da segunda camada intermediária e o neurônio c da primeira camada intermediária, e θ^2_b é o parâmetro θ do neurônio b da segunda camada intermediária.

Da mesma forma feita na formulação das derivadas em relação aos parâmetros da camada de saída, temos que:

[2] $fnet(1 - fnet)$ indica o valor de $fnet$ multiplicado por $(1 - fnet)$.

[3] O valor -2, multiplicando o somatório, vem da derivada do termo ao quadrado e da derivada de $-fnet^3_i$.

[4] Como a $\frac{\partial (w^3_{ij} * fnet^2_j)}{\partial w^3_{ab}}$ é 0 para todo ij diferente de ab, temos que o primeiro somatório desaparece, ficando apenas os elementos com $i = a$, e o segundo somatório também desaparece, ficando apenas os alementos com $j = b$. Para ij igual a ab o valor desta derivada é $fnet^2_b$, o que explica o resultado final da $\frac{\partial E^2}{\partial w^3_{ab}}$.

$$
\begin{aligned}
\frac{\partial E^2}{\partial w^2_{bc}} &= \frac{\partial \sum_{i=1}^{I}(y_i - \hat{y}_i)^2}{\partial w^2_{bc}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - \hat{y}_i)^2}{\partial w^2_{bc}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(net^3_i))^2}{\partial w^2_{bc}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j) + \theta^3_i))^2}{\partial w^2_{bc}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j(net^2_k)) + \theta^3_i))^2}{\partial w^2_{bc}} \\
&= \sum_{i=1}^{I} \frac{\partial (y_i - fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j(\sum_{k=1}^{K}(w^2_{jk} * fnet^1_k) + \theta^2_j))) + \theta^3_i))^2}{\partial w^2_{bc}} \\
&= -2\sum_{i=1}^{I}(y_i - \hat{y}_i)\frac{\partial fnet^3_i(\sum_{j=1}^{J}(w^3_{ij} * fnet^2_j(\sum_{k=1}^{K}(w^2_{jk} * fnet^1_k) + \theta^2_j))) + \theta^3_i)}{\partial w^2_{bc}} \\
&= -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i)\sum_{j=1}^{J}\frac{\partial (w^3_{ij} * fnet^2_j(\sum_{k=1}^{K}(w^2_{jk} * fnet^1_k) + \theta^2_j) + \theta^3_i)}{\partial w^2_{bc}} \text{[5]} \\
&= -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) w^3_{ib}\frac{\partial fnet^2_b(\sum_{k=1}^{K}(w^2_{bk} * fnet^1_k) + \theta^2_b)}{\partial w^2_{bc}} \text{[6]} \\
&= -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) w^3_{ib} fnet^2_b(1 - fnet^2_b)\frac{\partial (\sum_{k=1}^{K}(w^2_{bk} * fnet^1_k) + \theta^2_b)}{\partial w^2_{bc}} \\
&= -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) w^3_{ib} fnet^2_b(1 - fnet^2_b) fnet^1_c
\end{aligned}
$$

$$
\frac{\partial E^2}{\partial w^2_{bc}} = -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) w^3_{ib} fnet^2_b(1 - fnet^2_b) fnet^1_c \tag{11}
$$

$$
\frac{\partial E^2}{\partial \theta^2_b} = -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i) w^3_{ib} fnet^2_b(1 - fnet^2_b) 1 \tag{12}
$$

A derivada do E^2 em relação a θ^2_b será parecida com a derivada em relação ao parâmetro w^2_{bc}, sendo que a diferença é dada pela substituições do valor de $fnet^1_c$ por 1. As funções em 11 e 12 mostram as derivadas destes parâmetros.

3.3 Formulação das derivadas em relação aos parâmetros da primeira camada intermediária e generalização

A formulação matemática das outras camadas irá seguir os mesmos pricípios utilizados até aqui, principalmente o cancelamento dos somatório devido as derivadas iguais a 0. Por exemplo, as

[5] Dado que o somatório em j percorre todos os neurônios da segunda camada escondida, e sendo b o índice que indica qual é o neurônio desta camada a qual estamos derivando, temos que a $\frac{\partial (w^3_{ij} * fnet^2_j(net^2_j)}{\partial w^2_{bc}}$ será 0 para todos o w^3_{ij} diferente de w_{ib^3}. Assim podemos tirar o soomatório em j e usar apenas os valores que possuem o índice $j = b$.

[6] Dado que o somatório em k percorre todos os neurônios da primeira camada escondida, e sendo c o índice que indica qual é o neurônio desta camada a qual estamos derivando, temos que a $\frac{\partial (w^2_{bk} * fnet^1_k(net^1_k)}{\partial w^2_{bc}}$ será 0 para todos o w^2_{bk} diferente de w_{bc^2}. Assim podemos tirar o soomatório em k e usar apenas os valores que possuem o índice $k = c$.

funções 11 e 14 mostram as formulações para a variação dos parâmetros da primeira camada intermediária.

$$\frac{\partial E^2}{\partial w^1_{cd}} = -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i)\sum_{j=1}^{J} w^3_{ij} fnet^2_j(1 - fnet^2_j) w^2_{jc} fnet^2_c(1 - fnet^2_c) x_d \quad (13)$$

[7]

$$\frac{\partial E^2}{\partial \theta^1_c} = -2\sum_{i=1}^{I}(y_i - \hat{y}_i) fnet^3_i(1 - fnet^3_i)\sum_{j=1}^{J} w^3_{ij} fnet^2_j(1 - fnet^2_j) w^2_{jc} fnet^2_c(1 - fnet^2_c) 1 \quad (14)$$

3.4 Variação dos parâmetros

A última etapa de execução do algoritmo MLP será variar os parâmetros w e θ com base nos valores obtidos nos cálculos das derivadas calculadas na seções anterior. As funções 6 e 7 mostram que os valores dos parâmetros para a nova iteração do algoritmo é igual aos valores antigos menos a constante η multiplicado derivada do E^2 em relações aos parâmetros. Aqui o valor de η(momentum) indica o quanto a derivada vai interferir na variação dos parâmetros, sendo responsável pela velocidade de convergência do algorimo.

A próxima seção irá descrever a maneira como o MLP foi implementado para este trabalho e quais são os parâmetros que este algoritmo aceita, parâmetros estes que indicam como ele irá executar.

4 Implementação

Para este trabalho foi realizada uma implementação baseada quase totalmente em matrizes. Todos os somatórios contidos nas formulações matemáticas descritas nas seções anteriores foram tranformadas de forma que permitice uma equivalência realizando multiplicações de matrizes.

O algorimo permite que se especifique qual o número de camadas intermediárias, assim como o número de neurônios que cada uma das camadas irão possuir, inclusive sobre os neurônios das camadas de entrada e saída. Além disso, permite que varie o parâmetro η, que interfere na velocidade de conversão; no parâmetro do número de iterações, que especifica o número máximo de iterações que o algorimo deve executar para a convergência; e a a precisão aceitável para a convergência, que indica o quanto a saída do algoritmo tem que estar próxima do valor esperado, para que o algoritmo pare de executar.

Para detalhes sobre a implementação, peço que deêm uma olhada no código fonte, implementado usando a linguagem python. A seguir, nas próximas seções, serão descritas a avaliação do algoritmo com base na variação dos seus parâmetros, utilizando para isso dois conjuntos de dados usados para classificação/regressão de algorimos de redes neurais.

5 Avaliação

Para avaliar o algoritmo frente a variação de seus parâmetros, foi realizados uma série de experimentos, utilizando dois conjuntos de dados: Wine Dataset e Geographical Original of Music Data Set(GOM)[1]. O wine possui 14 atributos (13 atributos exploratórios e 1 atributo de classe) e 178 instâncias. O GOM possui 70 atributos (68 exploratórios e 2 atributos de classe) e 1059 instâncias.

Para tentar padronizar as medições em relação aos dois conjuntos de dados, foi utilizado uma representação semelhante para as classes tanto do wine quanto do GOM. Para tal, foi utilizada uma representação do pacote nnet do R[2], que cria atributos artificialmente para representar as classes dos datasets, seguindo o seguinte critério: são criado um atributo para cada classe diferente, de forma que quando uma instância do dataset possui uma determinada classe, é atributo 1 ao atributo daquela classe e 0 aos atributos que representam as outras classes. Assim, usando o GOM que possui 32 classes diferentes, são criadas artificialmente 32 atributos para representar as classes. Esse método só pode ser utilizado para classificação, sendo que o caso GOM foi considerado, por nós, classificação e não uma regressão.

[7] O valor de x_d é o valor que sai do neurônio d da camada de entrada. Se houvesse mais camadas intermediárias, substituiríamos x_d pelo $fnet_d$.

5.1 Avaliação com base na variação do número de instâncias utilizadas para treinamento e teste

Para avaliar a variação do erro médio da classificação com base na variação da proporção de instâncias utilizadas para treinamento e teste, foi utilizado a técnica *k-Fold Cross Validation*. Para o wine foi utilizado 7 valores de K diferentes, a saber: 2, 4, 6, 8, 10, 12 e 14. Por exemplo, quando utilizado K valendo 14, foi criado 14 subconjuntos de instâncias, de forma que a proporção de classe em cada subconjunto fosse iguais, ou seja, havia o mesmo número de instância de cada classe em todos os subconjuntos. Depois, foi utilizado um dos subconjunto para teste e todos os outros para treino. Assim, para este K, foi executado 14 vezes o MLP, cada uma delas utilizando um dos subconjuntos diferentes para teste. O resultado final do erro era a média dos erros que cada uma das 14 execuções proporcionava. A figura 2 mostra a variação do erro médio para os diferentes valores de K.

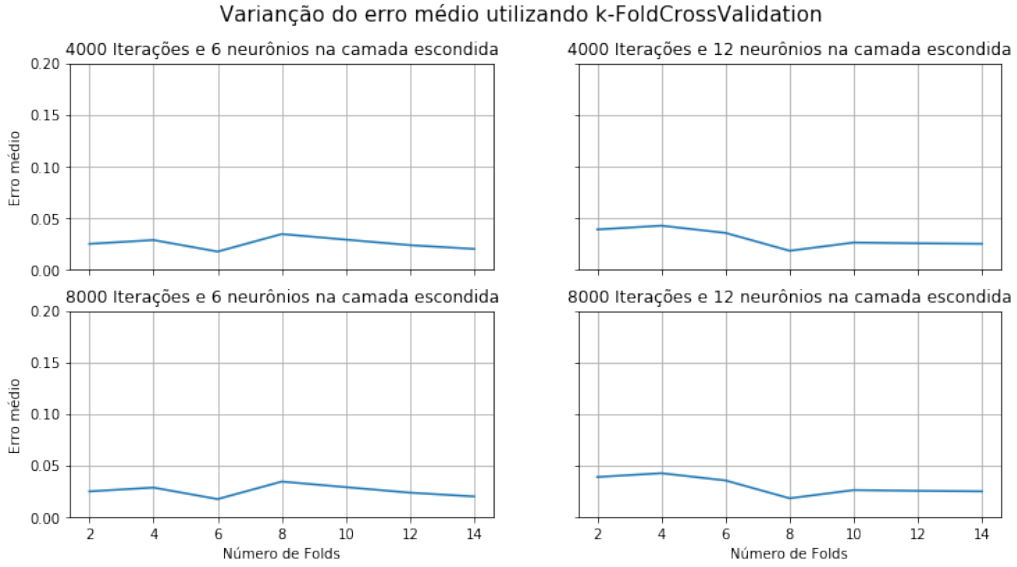

Figura 2: Variação do erro médio do MLP em relação ao número de folds utilizados pelo método K-Fold Cross Validation. Os gráficos mostram para diferentes número de iterações e diferentes número de neurônios na camada intermediária. Neste caso só há uma camada intermediária.

Da mesma forma foi avaliado a variação do erro médio a partir da variação do número de folds para GOM. Porém, os números utilizados para K foram apenas: 2, 6 e 10. Diminuimos o número de testes porque o algoritmo demorava muito mais para rodar com o GOM do que com o wine. A figura 3 mostra a variação do erro médio a partir da variação de K.

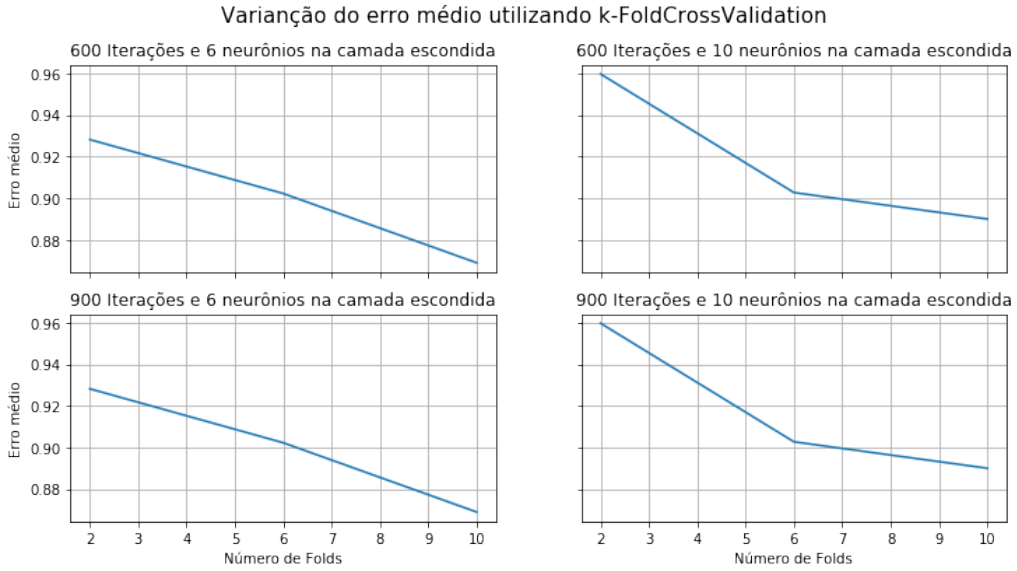

Figura 3: Variação do erro médio do MLP em relação ao número de folds utilizados pelo método K-Fold Cross Validation. Os gráficos mostram para diferentes número de iterações e diferentes número de neurônios na camada intermediária. Neste caso só há uma camada intermediária.

Como pode ser percebido pelos gráficos nas figuras 2 e 3, o erro médio calculado na segunda figura foi bem maior. Isso se deve, dentre outras coisas, ao fato de que o número de instâncias e de atributos do GOM, em relação ao wine, ser bem maior, o que impossibilitou realizar experimentos com um número grande de iterações. A figura 4 mostra um experimento usando GOM que rodou por mais de 3000 iterações, e é possível perceber a tendência de diminuição do erro a medida que as iterações aumentam.

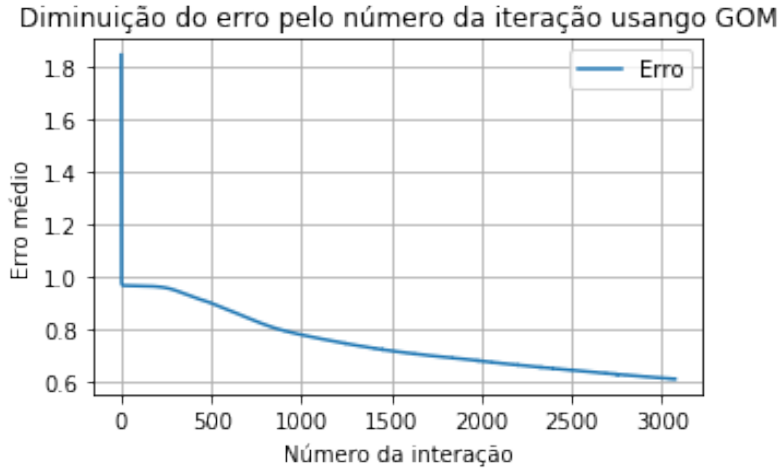

Figura 4: Diminuição do erro médio em GOM a medida que as iterações aumentam.

5.2 Avaliação com base na variação do número de neurônios na camada intermediária

Para avaliar o erro médio cometido a medida que variávamos o número de neurônios na camada intermediária, calculamos a média e o desvio padrão para o número de neurônios utilizados nos experimentos, a saber: para o wine foram utilizados 2, 4, 6, 8, 10, 12 e 14 neurônios; para o GOM foram usados 2, 6 e 10 neurõnios, sendo que para o cálculo da média e desvio padrão foram utilizados 35 observações para o wine e para 9 para o GOM.

As figuras 5 e 6 mostram a média e a variância do erro médio em relação a variação do número de neurônios da camada intermediária para o wine e o GOM, respectivamente.

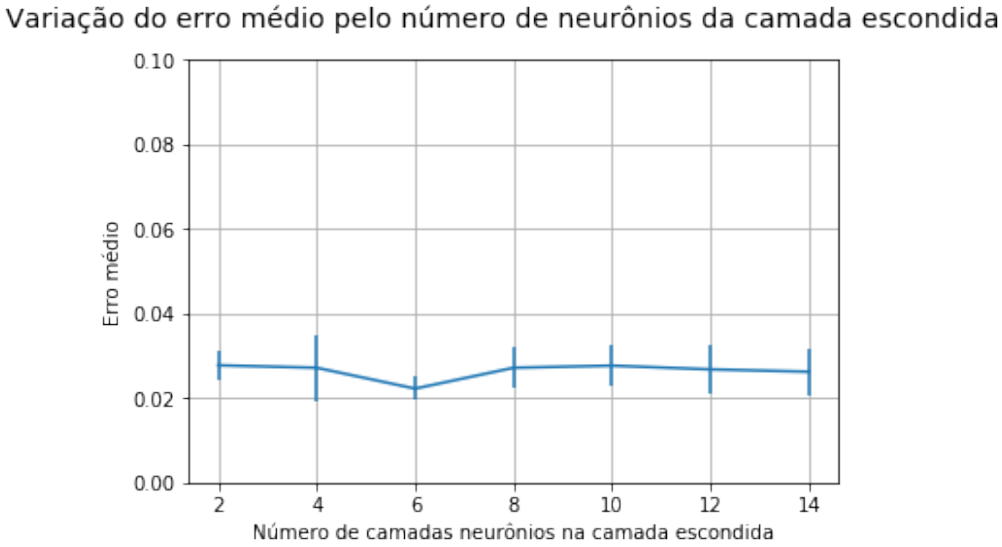

Figura 5: Média e desvio padrão do erro médio em relação a variação do número de neurônios na camada intermediária, usando o wine.

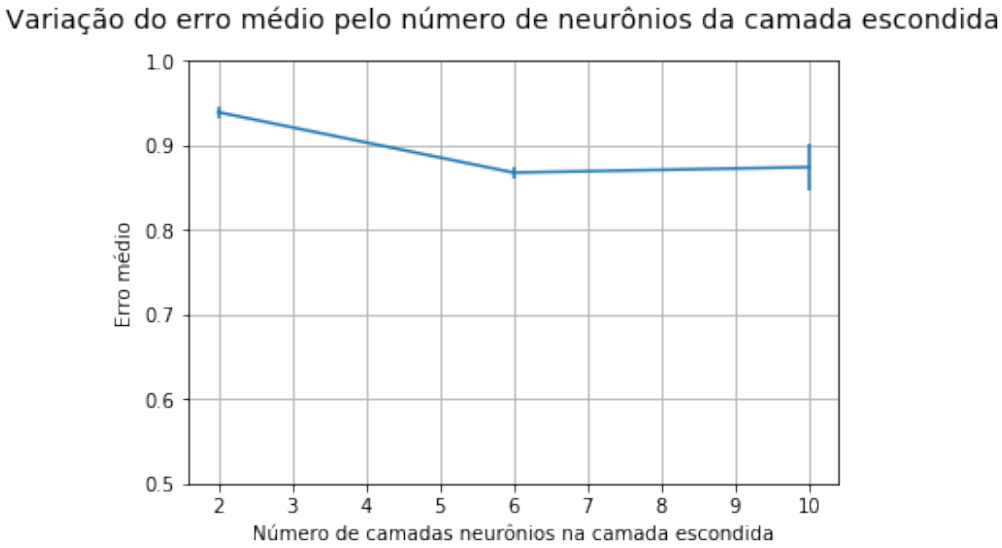

Figura 6: Média e desvio padrão do erro médio em relação a variação do número de neurônios na camada intermediária, usando o GOM.

5.3 Avaliação com base na variação do número de neurônios das duas camadas intermediárias

Para avaliar a variação do erro médio em função da variação do número de neurônios das duas camadas intermediárias, considerando o uso de duas camadas, foi considerado os número de 2, 6 e 10 neurônios para ambas as camadas e conjunto de dados. Para ambos os casos foi calculado a média e o desvio padrão considerando 9 observações.

As figuras 7 e 8 mostram a média e a variação do erro médio em relação a variação do número de neurônios nas duas camadas intermediárias para o wine e GOM, respectivamente.

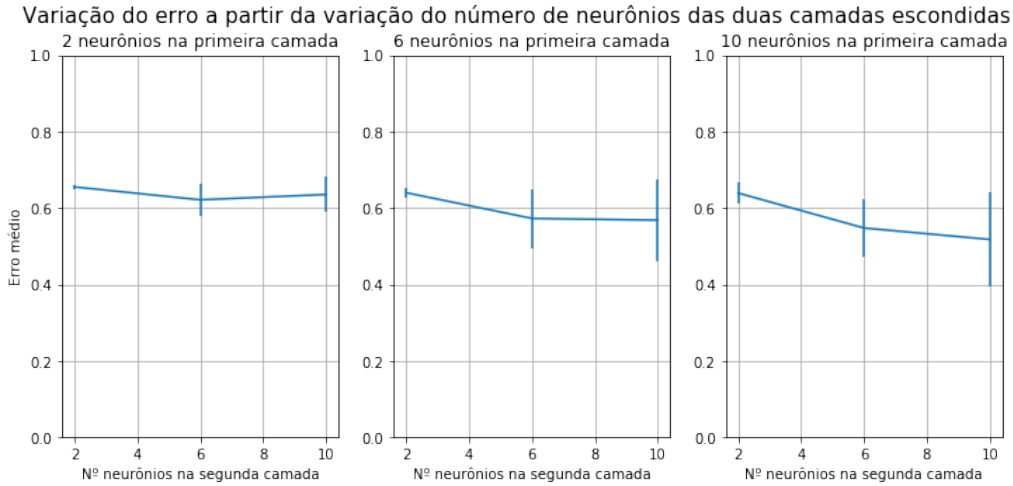

Figura 7: Média e desvio padrão do erro médio em relação a variação do número de neurônios na primeira camada intermediária (representado por cada um dos 3 gráficos) e segunda camada intermediária para o wine.

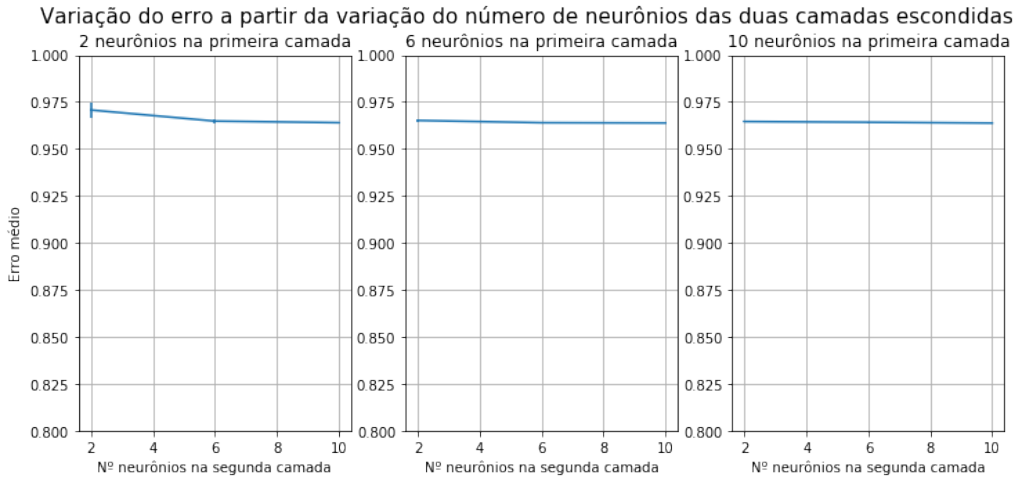

Figura 8: Média e desvio padrão do erro médio em relação a variação do número de neurônios na primeira camada intermediária (representado por cada um dos 3 gráficos) e segunda camada intermediária para o GOM.

5.4 Avaliação da velocidade da convergência com base no parâmetro momentum

O parâmetro momentum, representado nas equações pela letra η, indica o quanto as derivadas dos parâmetros irão interferir nos valores antigos dos parâmetros. Se o η for muito pequeno, o algoritmo irá precisar de muitas iterações para convergir; se *eta* for muito grande o algoritmo poderá ter problemas para convergir.

Analisando os gráficos representados nas figuras 9 e 10 podemos verificar que o algoritmo está convergindo para η baixos, porém lentamente. Para η altos podemos verificar também a dificuldade de convergência, sendo que o erro fica praticamente constante. Para *eta* valendo 0.1 e 1 verificamos a melhor convergência, mesmo para o caso utilizando o GOM.

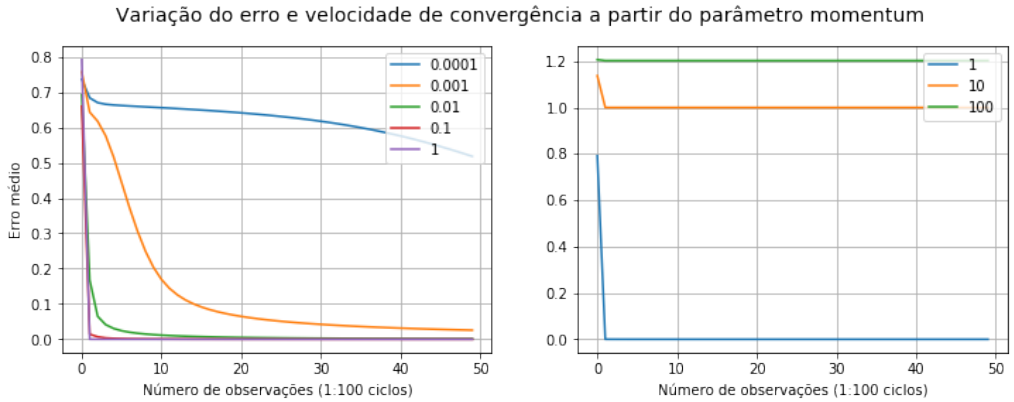

Figura 9: Velocidade de convergência para diferentes valores do parâmetro momentum(η) utilizando o wine.

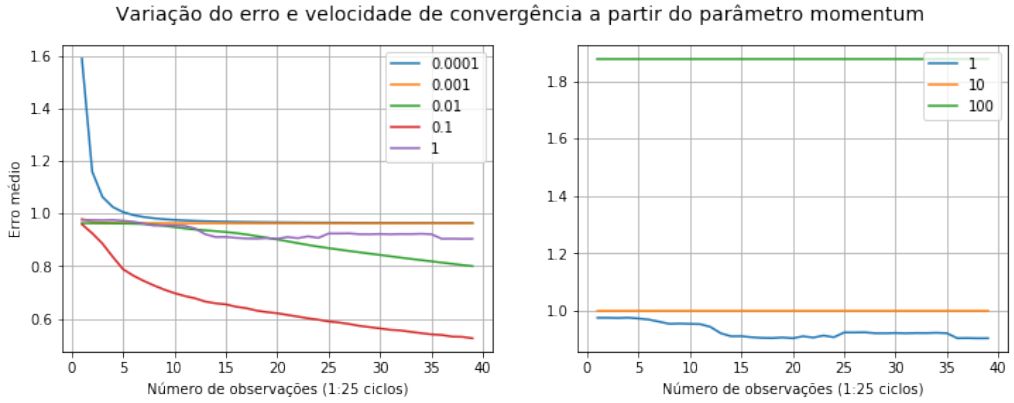

Figura 10: Velocidade de convergência para diferentes valores do parâmetro momentum(η) utilizando o GOM.

Referências

[1] Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017.

[2] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. ISBN 0-387-95457-0.