IA353A - Neural Networks
EFC2 - Question 2

Rafael Claro Ito
(R.A.: 118430)

May 2020

1 Source files

The Jupyter notebook with the code used to generate the plots and results presented in this report, all figures showed here and even the LaTeX source code used to generate this PDF can be found at the following GitHub repository:

`https://github.com/ito-rafael/IA353A-NeuralNetworks-1s2020`

2 Extreme Learning Machine

2.0 Regularization coefficient (Ridge Regression)

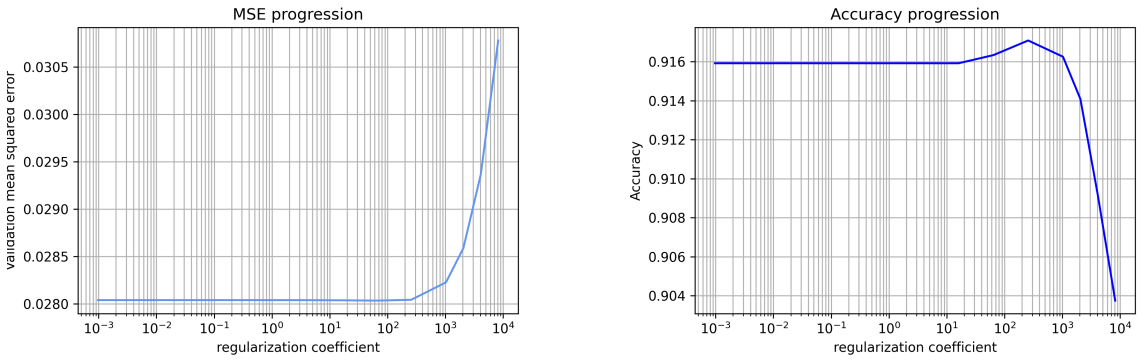
	λ optimum	
	MSE	Accuracy
coarse search	64	256
fine search	102.7	662.7

Table 1: Values of regularization coefficient found in coarse and fine searches

2.0.1 Coarse search

While performing the coarse search for the best regularization coefficient, 3 more values of lambda were added. This was done in order to see the falling of the accuracy curve, even though the best accuracy was in 2^8. The final values of lambda tested were:

$$\text{alpha\_interval} = [2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}, 2^{0}, 2^{2}, 2^{4}, 2^{6}, 2^{8}, 2^{10}, \mathbf{2^{11}, 2^{12}, 2^{13}}]$$

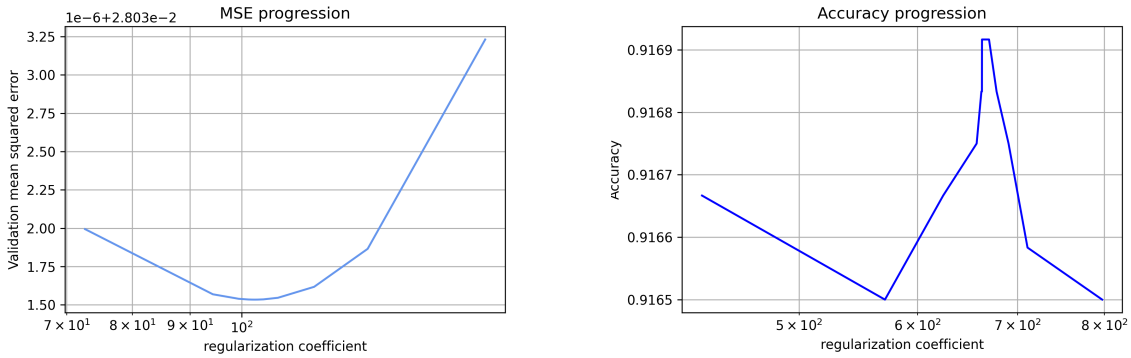


(a) MSE progression

(b) Accuracy progression

Figure 2: Progression of MSE and accuracy in validation set for different values of the regularization coefficient (coarse search)

2.0.2 Fine search



(a) MSE progression)

(b) Accuracy progression)

Figure 3: Progression of MSE and accuracy in validation set for different values of the regularization coefficient (fine search)

In order to perform the fine search of the regularization coefficient, a golden-section one dimensional search algorithm was coded. Among the function parameters, the most important ones are the intervals of the search, precision desired and the loss function. The code can be found in:
`https://github.com/ito-rafael/machine-learning/blob/master/snippets/golden_section_search_valid.py`

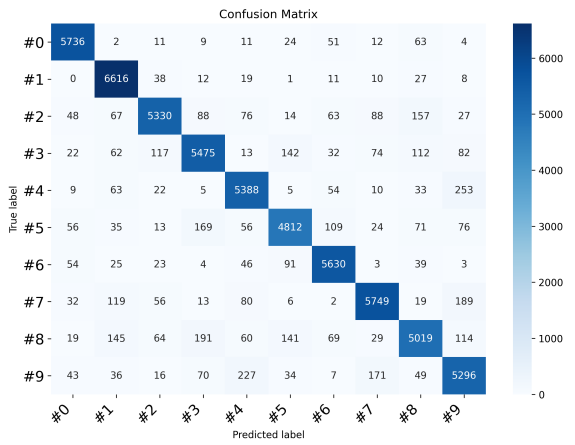
2.1 Confusion Matrix and Misclassification

Considering the training set, plot the confusion matrix and a few examples of misclassified digits from at least three different classes.

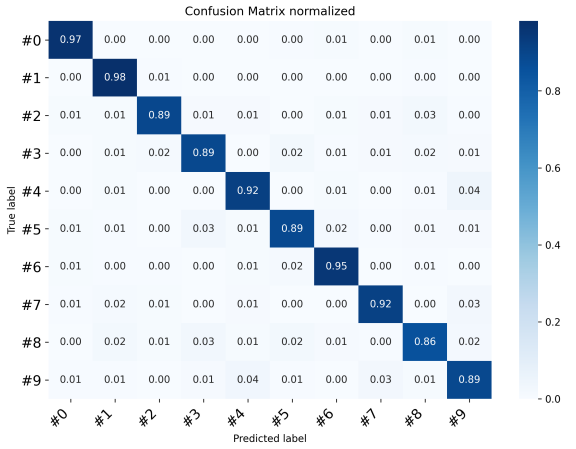
2.1.1 Confusion Matrix

Digit	nº of samples
0	5923
1	6742
2	5958
3	6131
4	5842
5	5421
6	5918
7	6265
8	5851
9	5949

Table 2: Number of samples for each class in the training set



(a) Confusion matrix with raw values

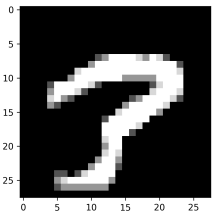


(b) Confusion matrix with normalized values

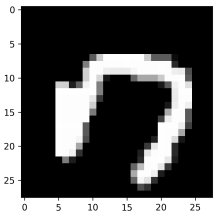
Figure 4: Confusion matrix with normalized and raw values

The values displayed in the confusion matrix were obtained with the extreme learning machine applied to the training set (training plus validation). The training set is somewhat balanced, containing the number of samples for each class as illustrated in Figure 2

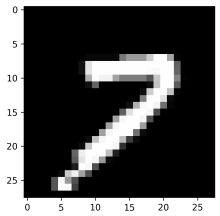
2.1.2 Misclassified data



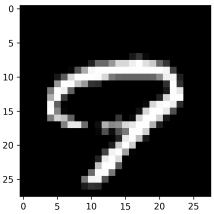
(a) Real class: 3 (0.02)
Predicted class: 7 (1.25)



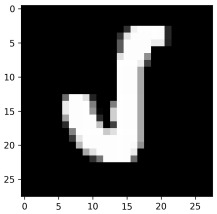
(b) Real class: 7 (0.10)
Predicted class: 0 (1.06)



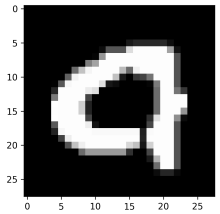
(c) Real class: 4 (0.02)
Predicted class: 7 (0.90)



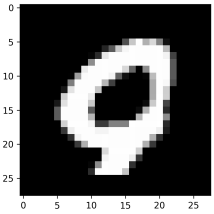
(d) Real class: 9 (-0.12)
Predicted class: 7 (0.73)



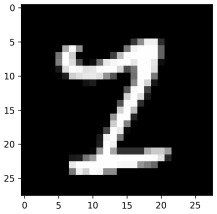
(e) Real class: 5 (-0.07)
Predicted class: 6 (0.78)



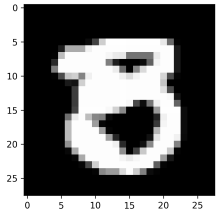
(f) Real class: 4 (0.14)
Predicted class: 0 (0.97)



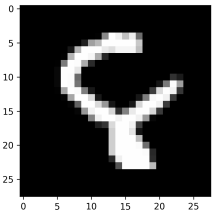
(g) Real class: 9 (-0.04)
Predicted class: 0 (0.77)



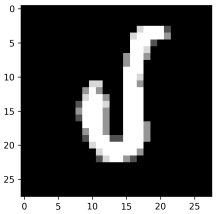
(h) Real class: 1 (-0.14)
Predicted class: 7 (0.66)



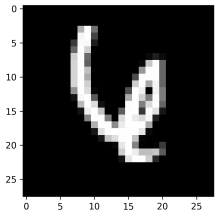
(i) Real class: 3 (-0.14)
Predicted class: 8 (0.66)



(j) Real class: 8 (-0.03)
Predicted class: 4 (0.74)



(k) Real class: 5 (-0.11)
Predicted class: 6 (0.65)



(l) Real class: 6 (0.09)
Predicted class: 4 (0.82)

Figure 5: Misclassified digits

The digits shown in Figure 5 are in the top 30 misclassified digits from which the difference between the output for the real class and the output for the predicted class are the highest. Both real and predicted class and its outputs values associated are indicated. The output is shown in parenthesis, in front of the correspondent class.

2.2 Performance and computational resources improvements

Question:
Seguindo a sugestão de empregar 500 unidades na camada intermediária da rede neural, apresente argumentos para sustentar o ganho de desempenho verificado e uma execução em menor tempo computacional, quando comparado com o classificador linear da Q1.

Answer:

- Performance improvement:

  We can see better results using the ELM comparing to the linear model of Q1 because now we use a non-linearity activate function at the hidden layer (here we used the rectifier activation function, i.e., the units of the hidden layer are ReLUs). The universal approximation theorem states that a feed-forward network with a single hidden layer with a finite number of neurons with non-linear activation function is capable of approximate any continuous function. Even though we are not training a neural network, but using "ready-to-use" hidden layer weights instead, the intuition behind the non-linear layer followed by a linear layer is the same. We can also see this performance improvement at the perspective that instead of finding a classifier directly from the values of the pixels in the image, now we have a new problem to solve. Now we have to find a classifier using the outputs of the neurons in the hidden layer, which is an easier problem than the previous one.

- Computational resources improvement:

  The reason that the code from Q2 runs faster than the code from Q1 is that the operation with the highest computational cost is by far calculating the inverse matrix of $(X^T X + \lambda I')$ when computing the weights, as can be seen in the following equation:

  $$\mathbf{W} = (X^T X + \lambda I')^{-1} X^T \mathbf{y}$$

  In Q1 we need to invert the matrix $(X^T X + \lambda I')$ of size 785 x 785, while in Q2 we have to invert the matrix $(H^T H + \lambda I')$ of size 501 x 501. This difference of shapes considering the matrix inversion is decisive to lower the computational time, even though in Q2 there is an extra product $H = XW'$ happening before the calculus of the $\mathbf{W}$ matrix, where W' is the matrix with the weights of the hidden layers chosen randomly.
  When using the rectifier activation function (commonly called ReLU, although ReLU is the unit - neuron), some of the outputs of the hidden layer will be zero. This might make the product of $H^T H$ a bit easier since a portion of elements will be zero, but the dominant term where most of the complexity resides is definetely at the computation of the inverse matrix.

2.3 Regularization coefficients comparison (Linear and ELM)

Question:
Compare os coeficientes de regularização obtidos nessas duas primeiras atividades (classificador linear da Q1 e ELM da Q2), os apresentando numa única tabela para fácil visualização, e procure justificar a diferença.

Answer:
The regularization coefficients for both Q1 and Q2 (coarse and fine search) can be seen in Table 3.

	λ optimum			
	Q1		Q2	
	MSE	Accuracy	MSE	Accuracy
coarse search	64	1024	64	256
fine search	51.5	1091.8	102.7	662.7

Table 3: Values of regularization coefficient found in coarse and fine searches in both Q1 and Q2

From now on the regularization coefficient will be just called "alpha" to simplify the text.

The results shown in Table 3 are curious, since the optimum alpha increased for the MSE criterion, but decreased for the accuracy metric. A higher value of alpha indicates that the weights are being more regularized, suggesting that the model is more flexible.

Talking about the ELM in specific, although the results obtained are much better than the linear approach (95% against 85% of accuracy), the machine is not necessarily "more flexible" in the sense stated in the previous paragraph. The improvements are achieved mainly because of the non-linearity activation function (tanh, ReLU) than due to flexibility, since the weights of the hidden layer were randomly chosen.

The smaller optimum alpha for the accuracy metric suggests a less flexible model for the ELM approach when comparing to the linear model. This can be explained due to the reduced values reaching the output layer (501 vs 785) and also the quality of these signals to perform the linear classification.

However, the bigger optimum alpha for the MSE criterion suggests that for these randomly generated weights, the ELM ended with a more flexible model than the linear one, even for a reduced number of signals reaching the linear output classifier layer.

2.4 Regularization coefficient for a different initialization

Question:
Mantendo a mesma partição entre conjuntos de treinamento e validação, o que você espera que ocorra com o coeficiente de regularização caso os neurônios da camada intermediária sejam inicializados com pesos sinápticos distintos a cada execução?

Answer:

As already explained at the answer of the previous question, the flexibility of the model strongly depends on the weights of the hidden layer. By randomly choosing the weights and knowing that the role of the regularization coefficient is to prevent overfitting in a model that is too flexible, the alpha will likely change every time, since for each weights configuration we are dealing with a new problem to solve, looking at the perspective of the output linear layer. The "ReLU" activation function can also perform a role here, since a portion of the outputs of the hidden layer will be zero depending on the weights generated (and also the input data).

In summary, for each randomly generated weights, a new model with a new flexibility will be generated, and depending on the flexibility, more or less regularization will be necessary. Furthermore, as stated at the answer of the previous question, the necessity of regularize the weights of the linear output layer more or less can also change when looking at the accuracy metric or the MSE criterion.

2.5 Modifications aiming better results

Question:
Promova algum tipo de alteração nas especificações da ELM e/ou de seu treinamento de modo a produzir resultados superiores àqueles conquistados ao se seguir o roteiro desta questão. Descreva adequadamente as alterações realizadas.

Answer:

	Accuracy	
	Training set	Test set
500 neurons	0.9175	0.9194
1000 neurons	0.9451	0.9448

Table 4: Results comparison for 500 and 1000 neurons at the hidden layer

Changing the number of neurons at the hidden layer from 500 to 1000 improved the results, as shown in Table 4.

This indicates that the model with 500 neurons at the hidden layer was not flexible enough, operating in underfitting. After increasing to 1000 neurons, more relevant signals are passed to the output layer and the model can make a better classification.