Dr. Matthew Ellis
Prof. Eleni Vasilaki

Assignment 1

19th February 2021

Supervised learning of handwritten letters

This assignment is to implement a supervised learning algorithm that trains a multi-layer feed-forward network to classify a set of images, similar to the 2nd lab. You will then have to produce a report to present and discuss the results of your model. You will be working with the EMNIST data set[1], we have provided a version of this on the VLE. This data set is similar to the MNIST data set but contains handwritten letters (A-Z) instead of numbers. The full data set therefore contains 26 classes and each image has been scaled to a size of 28x28 pixels. Each pixel has an intensity in the range 0-255 (i.e not normalised). The original data set contains 5,600 samples per class in the training and testing portion of the data set. Here, we have restricted the size to only 1,000 per class for training and 250 per class for testing. The data set is given with each class in order so you should make sure you randomise it first. Please remember to cite the database source in your report.

Your task is to implement and train a multi-layer feed-forward neural network with the Mean Square Error (MSE), as described in class, but with the ReLU function instead of the logistic sigmoid used in the lab. The ReLU function is given as

$$f(h) = \max(h, 0), \tag{1}$$

where h is the total input for the neuron. You should also implement a form of minibatch updating so that the gradient is updated multiple times per epoch. For simplicity, use a minibatch size of 50 samples.

The provided data set has already been split into a training and testing part. It is standard practice to only use the testing part to give a final values for the performance of your trained model. Therefore, you should randomly split your training part into a third *validation* part (typically 20% of the original training size). This is not used for updating the weights but for optimising hyper-parameters, such as the learning rate. At each epoch, the validation part can be used to measure the error and accuracy to monitor how well training is proceeding.

In your report you should address the following points but are also encouraged to expand upon these where possible:

1. *Introduction* - Explain the process of training a classifier using supervised learning. What are the key ingredients and the objective?

2. *Methods* - Give a brief description of the algorithm you implemented. You should define the update rules for the weights including the gradient of the ReLU activation function. Discuss why ReLU might be more suitable for networks with many layers and if it has any drawbacks? Explain what the difference between online, mini-batch and full batch updating is? Why is it important to normalise data and initialise the weights well for the training to work properly?

3. *Results and Discussion* - Before training a multi-layer neural network it is important to have a baseline performance to compare to. Train a single layer network (a perceptron) on the data. When your model has converged the average weight changes should be close to zero. During the training calculate the average weight update using an exponential moving average. To do this the average matrix $\mathbf{A}_n$ at epoch n is updated using

$$\mathbf{A}_n = \begin{cases} \Delta \mathbf{w}_1 & \text{for } n = 1 \\ \mathbf{A}_{n-1}(1-\tau) + \tau \Delta \mathbf{w}_n & \text{for } n > 1 \end{cases} \tag{2}$$

where $\Delta \mathbf{w}_n$ is the weight update computed at epoch n, and τ is a small constant (such as 0.01). The first condition initialises the average to the first value of the weight update, i.e n is the first training iteration. **Plot the sum over all the elements of this average weight update matrix vs epochs** to show that your model has converged.

4. What classification performance do you achieve on the test data set with your single layer network? Read reference [1] (a copy will be available on the VLE), what accuracy for the EMNIST letters set do the authors achieve with a linear model?

5. When a model has many more parameters than the number of input patterns it is likely that it will start to overfit the data. This means that the model will start to learn the noise within the training data and generalises poorly to the test set. One method to avoid this is to add a penalty term to the error function that depends on the magnitude of the weights. Here, we will consider the L_1 regularisation penalty, which is given by

$$E_{L1} = \lambda_1 \sum_{k}^{N_{\text{layers}}} \sum_{ij} |w_{ij}^{(k)}|, \tag{3}$$

where λ_1 is a hyper-parameter that controls the strength of this penalty. The sum over k is over all the layers in the network while the sum over i and j is over the size of the weight matrix for that layer. This error term is added to the original mean squared error (MSE) term so that the total error is

$$E = E_{\text{MSE}} + E_{L1}, \tag{4}$$

where E_{MSE} is the MSE defined in class. This means when we calculate the derivative of the error function to find the update there will be an additional term

$$\frac{\partial E}{\partial w_{ij}^{(k)}} = \frac{\partial E_{\text{MSE}}}{\partial w_{ij}^{(k)}} + \frac{\partial E_{L1}}{\partial w_{ij}^{(k)}} \tag{5}$$

and likewise the L1 error will give an additional term in the weight update,

$$\Delta w_{ij,L1}^{(k)} = -\eta \frac{\partial E_{L1}}{\partial w_{ij}^{(k)}} \tag{6}$$

Derive this additional term and show your working in your report. It may help to define the absolute function as $|w| = \sqrt{w^2}$. Implement this extra term in your code but remember the weight update from the MSE will remain.

6. Train your model starting with a single hidden layer of 50 neurons with the ReLU activation function for a range of penalty strengths (λ_1). Make sure you allow enough time to reach convergence, e.g 250 epochs with a learning rate of $\eta = 0.05$. Values for λ_1 are typically quite small so I recommend varying logarithmically (i.e ..., $3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}, 10^{-3}$,...). **Plot the final validation error averaged over 5 different runs against the values of λ_1.** Does the penalty term aid the learning and, if it does, what is the optimal value you find? You should use this optimal value in the following questions. What are other methods to stop overfitting or overtraining? See reference [2] and the book it is contained in for further information.

7. Vary the number of neurons in the hidden layer and measure the percentage of correct classifications (accuracy) on the test data set. Perform statistics over at least 10 different initial conditions. **Plot the average testing accuracy (with error bars) against number of neurons and discuss your results.** Discuss your results, how do these compare to your perceptron results and those found in reference [1]?

8. Train your model with a second hidden layer of varying size and your first hidden layer fixed with a size of 100 neurons. **Again, plot the average testing accuracy (with error bars) against the number of neurons in the 2nd hidden layer.** How has the performance changed? Is a network with more smaller layers better than 1 larger layer?

Important Note: To give a benchmark for your run times on a 2.7 GHz Dual-Core Intel Core i5 laptop training with 1 layer of 400 hidden neurons for 100 epochs takes approximately 300 seconds. This benchmark uses matrix operations on the batch, you should consider optimising your code along theses lines.

Report

This is an individually written report of a scientific standard, i.e. in a journal-paper like format and style. It is recommended that you use the web to find relevant journal papers and mimic their structure (e.g. `https://www.ieee.org/conferences/publishing/templates.html`). Results should be clearly presented and justified. Figures should have captions and legends. Your report should **NOT exceed 5 pages** (excluding Appendix and Cover Page). It is important that your are concise and carefully consider the figures and discussion you are presenting. Additional pages will not be assessed. Two-column format is recommended, the minimum font size is 11pt and margins should be reasonable. Kindly note that the readability and clarity of your document plays a major role in the assessment.

In the report you should include:

1. A response to all points requested by the assignment (including graphs and explanations). It is suggested to adopt a similar ordering/numbering scheme to make clear that you have responded all questions.

2. An Appendix with snippets of your code referring to the algorithm implementations, with proper comments/explanations.

3. A sufficient information that your results could be reproduced, see "Important Note" below.

4. You should also provide references for sources that you have drawn upon; such as a reference for the database or for any methods you may implement from books or journal articles.

Important Note: Please make sure that your results are reproducible. Together with the assignment, please upload your code well commented and with sufficient information (e.g. Readme file) so that we can easily test how you produced the results. If your results are not reproducible by your code (or you have not provided sufficient information on how to run your code in order to reproduce the figures), the assessment will not receive full points.

Suggested language

The recommended programming languages are Python or Matlab. However, on your own responsibility, you may submit the project in any language you wish, but you should agree it beforehand with the Professor.

Marking

Assignments will be marked with the following breakdown: results and discussions contribute up to 70%, scientific presentation and code documentation up to 20% and originality in modelling the task for up to 10%.

A mark greater than 39% indicates an understanding of the basic concepts covered in this course. A mark greater than 69% indicates a deep knowledge of the concepts covered in this course, with evidence of independent thinking or engagement beyond the level of material directly covered during lectures/laboratory sessions.

To maximise your mark, make sure you explain your model and that results are supported by appropriate evidence (e.g. plots with captions, scientific arguments). Figures should be combined wherever appropriate to show a clear message. Any interesting additions you may wish to employ should be highlighted and well explained.

Submission

The **deadline** for submitting the report and the corresponding code is **Monday 15th March 2021, 15:00**. Please make sure you upload your report (single file only) in the "Assignment 1" slot. Your code should be uploaded on the "Assignment 1 - Code" slot and in case of multiple files please zip them and upload them as one zipped file. Please do not include the data set in the zip.

We remind that the results described in the report should be reproducible by your code (qualitatively). A dedicated appendix section on how to use your code to reproduce your results or a readme file along with your code is advisable.

Plagiarism, and Collusion

You are permitted to use Matlab or Python code developed for the lab as a basis for the code you produce for this assignment but with **appropriate acknowledgment**. You may discuss this assignment with other students, but the work you submit must be your own. Do not share any code or text you write with other students. Credit will not be given to material that is copied (either unchanged or minimally modified) from published sources, including web sites. Any sources of information that you use should be cited fully. Please refer to the guidance on "Collaborative work, plagiarism and collusion" in the Undergraduate or MSc Handbook.

References

[1] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: an extension of MNIST to handwritten letters." Retrieved from: `http://arxiv.org/abs/1702.05373`, (2017).

[2] L. Prechelt, *Early Stopping — But When?*, pp. 53–67. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.