COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS
DEPARTMENT OF APPLIED INFORMATICS

Neural Networks
Project 1 – Multilayer perceptron

March 25, 2025

OVERVIEW

Task: Implement a general multilayer perceptron classifier (with at least one hidden layer), trained by the backpropagation algorithm. Employ this model on a task of classifying points on a plane into three categories. Use a validation technique to select the best performing model, then perform final testing.

Tools: You are free to use all the codes from the classes. Use bravely `numpy` and `matplotlib`. What is not allowed: Deep learning toolboxes (`tensorflow`, `pytorch`, etc). The backpropagation algorithm must be implemented by you!

Deadline: April 20, 23:55
Late submissions are penalized by -2 points each day. **It is not possible to submit a project more than 5 days after the deadline.**

SPECIFICS

Model:

- Multi-Layer Perceptron, having at least one *non-linear* hidden layer
- (Stochastic) Gradient Descent via backpropagation (online, mini-batch or batch)

Data:

- one-line header, then one sample per line
- points in a 2D plane (2 real-valued inputs)
- three output classes (`A`, `B` and `C`)
- train set -– `2d.trn.dat` , 8000 samples — training data (estimation and validation)
- test set — `2d.tst.dat` , 2000 samples — testing data

Training: Split the training data set into a bigger *estimation* subset and a smaller *validation* subset.[1] Use this split to perform model selection, i.e. find the best-performing combination of hyper-parameters:

- train the model on the *estimation* subset
- test the model on the *validation* subset (not the *test* set! don't touch that yet!)
- remember the hyper-parameters of the best performing model

Sanity check: a properly working network should reach a classification accuracy of ≥95%

Also try experimenting with some of the following (at least 4 for maximum points):

- input preprocessing (e.g. normalization/rescaling)
- activation functions (logsig, tanh, softmax, ...)
- early-stopping
- learning rate schedule
- weight initialization type (uniform, Gaussian, sparse, ...) and scale
- momentum type (none, classic, Nesterov's accelerated gradient) and strength
- regularization (weight decay, L_2 penalty, ...)

Testing:
Using the best performing set of hyper-parameters (on the *validation* set), train a new model on the full *training* set, then perform final testing on the *test* set. Report classification accuracy, regression error and calculate a confusion matrix.

Bonus [2 points]:
Train the model using a more sophisticated training method, such as Adaptive moment estimation (Adam).

SUMBISSION

Submit your code and report using the moodle as a single archive (e.g. `.zip`).
Code (9 p.)
Projects should be written in Python; use of previously finished labs is strongly encouraged. All the "interesting" bits should be identifiable in the code (especially all the relevant equations). You'll probably need no additional libraries other than the standard `numpy` / `scipy` / `matplotlib` combo. Don't reimplement the wheel, use `np.loadtxt` / `np.savetxt` / `np.load` / `np.save` and `plt.savefig` where necessary.
Model selection should not be performed by hand, but the project should rather include a runnable program that goes through the various combinations of parameters, selects the best performing model and tests and produces final outputs.
Points will be awarded for:

- Performance (up to 2 pt.)
- Code – it should be modular and easy to read (up to 1 pt.)
- Efficient implementation (up to 2 pt.)
- Number of tested hyper-parameters – the more, the better (up to 2 pt.)
- Best model selection (up to 2 pt.)

[1]Nomenclatures may differ according to the source – *train/test* and *estimation/validation* split represent the same idea, but at different levels and for different reason.

Report (6 p.)
Create a report briefly describing model selection, training and testing in `.pdf` format. The report should be sufficiently detailed so that one can read the description, reimplement your project and using the provided parameters arrive at the same results (reproducibility). Assume previous knowledge of neural network algorithms, so for example don't explain how backpropagation works. But include details such as if the training was online/minibatch/batch and whether and what strength of momentum was used.

- for each examined model (hyper-parameter combination), report estimation and validation error [**table**]
- for the best model:
  - error vs. time (at least one instance) [**plot**]
  - outputs in 2D [**plot**]
  - confusion matrix [**table**]
    - rows = actual classes
    - columns = predicted classes
    - sum of each row = 100%