Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics
Department of Applied Informatics

Neural Networks

Project 1 – Multilayer perceptron

Zulfiya Usmonova

Overview

In this project we try to solve the two-dimensional problem with datasets 2d.trn / 2d.tst: 10000 uniformly distributed points in a square region, each labelled as A, B or C according to a non-linear decision rule. The goal is to build a network that maps raw planar coordinates directly to its class with minimal classification error.

We follow the usual machine-learning protocol:

- Pre-processing — feature-wise centring and scaling based only on training statistics;
- Model selection — grid-search on an estimation/validation split of the training set;
- Final training - retrain the best hyperparameter configuration on all 8000 training samples;
- Testing— report performance on the unseen 2000-sample test set using error curves, confusion matrix and decision-region plots.

1 Pre-processing

All coordinates were centred and rescaled feature-wise:

$$\tilde{x} \;=\; \frac{x - \mu_{\text{train}}}{\sigma_{\text{train}}}$$

using the mean μ_{train} and standard deviation σ_{train} computed on the 8000 training set only. The statistics were fixed for all splits and for the final test to make sure that no data from the test or validation sets leaked into the normalisation process.

2 Model Selection

2.1 Model Architecture

A single-hidden-layer multilayer perceptron (MLP) implemented in pure NumPy was used:

- Inputs: 2 (planar x, y)
- Hidden layer: H neurons, activation tanh / logsig / relu
- Output layer: 3 logits followed by a soft-max non-linearity
- Total parameters:

$$\Big(H \times 2 + H\Big) \;+\; \Big(3 \times H + 3\Big) \;=\; 3H + H + 6 \;=\; 4H + 6$$

Biases are absorbed by appending a constant 1 to every input or hidden vector (see add_bias() in util.py).

2.2 Optimisation set–up

We trained all networks with stochastic gradient descent (SGD), i.e. one weight update per sample,

$$\theta \leftarrow \theta + \eta_t \, \nabla_\theta \mathcal{L}(x^{(t)}, y^{(t)}),$$

where $\theta = \{\mathbf{W}_{\text{hid}}, \mathbf{W}_{\text{out}}\}$ and $\mathcal{L}$ is the cross-entropy loss (used only for logging and for the early-stopping check). No momentum term was used (`momentum`=0).

- **Initial learning rate:** $\eta_0 \in \{0.01, 0.05\}$ (selected during the grid search).
- **Learning-rate schedule:** exponential decay $\eta_t = \dfrac{\eta_0}{1 + \lambda\, t}$ with $\lambda \in \{0,\ 0.01,\ 0.05\}$.
- **Batch size:** 1 (pure online mode).
- **Weight decay / regularisation:** none.
- **Weight initialisation:** Gaussian, uniform or sparse; the winning model used uniform.

2.3 Early stopping

During each epoch we measured the classification error on the *validation* subset. If the validation error did not improve for $P = 12$ consecutive epochs (patience), training was stopped and the best weights so far were restored:

```
if val_CE + min_delta < best_val_CE:
    $best\_weights \leftarrow \theta$
    $wait \leftarrow 0$
else: $wait \leftarrow wait + 1$
if $wait \geq P$: break
```

2.4 Hyper-parameter exploration and model selection

A 3×2×3×3×2=108-point grid was evaluated. For each configuration the model was trained on the estimation subset (80% of the 8000 samples) and evaluated on the remaining 20% validation subset. Figure error_curves.png shows a learning curve, and hyper-param-errors.csv lists every trial.

Grid-search finished in 01:23:17.178 (1 hour, 23 minutes).

The best validation score (CE=0.008125) was obtained with:

- hidden = 30
- learning rate = 0.05
- lr-decay = 0.05
- activation = tanh
- init = uniform

Table 1: Validation performance for a *subset* of the hyper-parameter grid. The best configuration is highlighted.

H	α	λ	Act.	Init	w_{scale}	CE_{val}
30	0.05	0.05	tanh	uniform	0.01	0.008 125
20	0.01	0.01	relu	uniform	0.01	0.010 000
30	0.01	0.05	relu	gaussian	0.01	0.010 625
10	0.05	0.00	logsig	gaussian	0.10	0.010 625
30	0.05	0.05	tanh	uniform	0.10	0.011 250
20	0.01	0.05	relu	uniform	0.01	0.011 875
20	0.01	0.05	relu	uniform	0.10	0.011 875
30	0.05	0.00	logsig	uniform	0.10	0.011 875
30	0.05	0.05	relu	gaussian	0.01	0.011 875
20	0.01	0.05	relu	gaussian	0.10	0.012 500

Table 2: Hyper-parameter grid explored during model selection

Parameter	Tested values
Hidden units	10, 20, 30
Learning rate α	0.01, 0.05
LR decay λ	0, 0.01, 0.05
Activation	`logsig`, `tanh`, `ReLU`
Init type	Gaussian, Uniform, Sparse
Weight scale σ	0.01, 0.10

2.5 Final Training

Using these hyper-parameters, the network was re-initialised and trained on the full 8000-sample training set for 300 epochs.

Final model training time: 00:01:42.267 (1 minute, 42 seconds)

Final test pass time: 00:00:00.031

3 Results and Discussion

3.1 Learning dynamics

Figure 1 shows the evolution of the classification error (%) and the mean–squared regression loss on the *training split*[1]. Both metrics stabilise after ≈ 80 epochs; the learning–rate decay keeps the curve smooth and prevents late oscillation.

3.2 Generalisation performance

With the best hyper-parameters $\left(H{=}30,\ \alpha_0{=}0.05,\ \lambda{=}0.05,\ \textit{tanh},\ \textit{uniform},\ w{=}0.10\right)$, the network reaches a final test classification error of **0.60 %** and a regression error of 0.022 14. Table 3 reports the row-normalised confusion matrix.

Table 3: Row-normalised confusion matrix (%).

	pred A	pred B	pred C
A	98.24	0.22	0.00
B	0.11	99.21	0.23
C	0.00	0.32	99.82

Decision regions. Figures 2–3 visualise the learned decision boundaries. The model fits the complex triangular pattern while remaining smooth in empty regions, suggest-

[1]The validation curve is omitted for clarity, but the early-stopping logic monitored it internally.

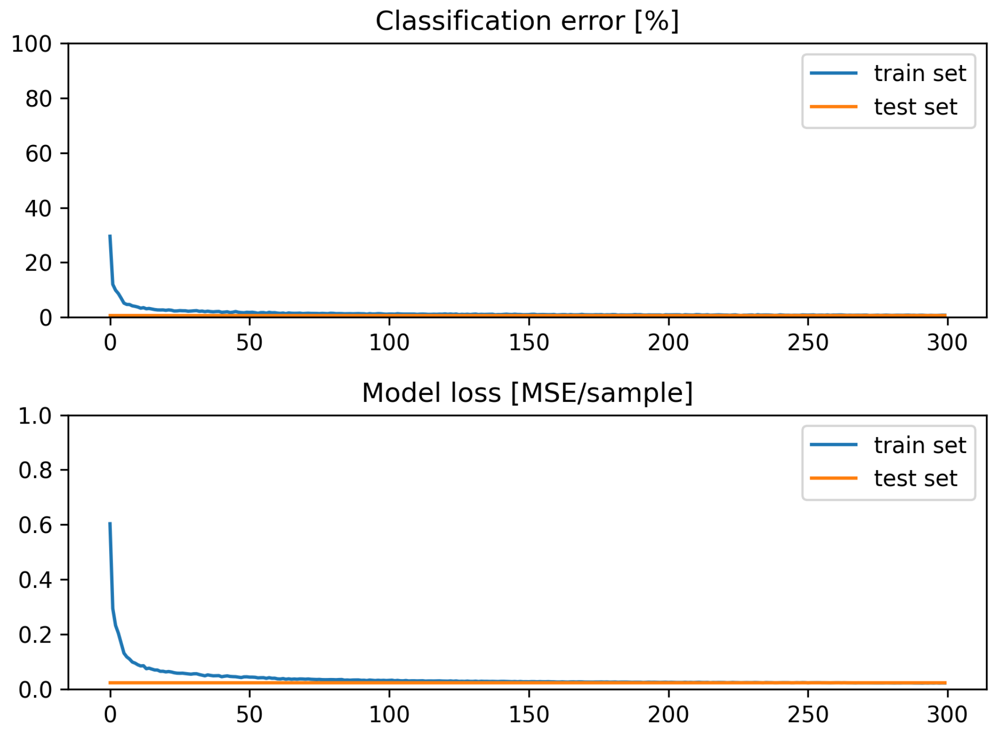

Figure 1: Training error (top) and regression loss (bottom) for the final model.

ing that the early-stopping and weight-decay in the learning-rate schedule controlled over-fitting.

3.3 Failure modes and future work

Most remaining errors are about narrow class boundaries (Figure 3), two possible solutions are:

- switching from mean-squared to categorical cross-entropy in the output layer loss, and
- replacing vanilla SGD with Adam, which is expected to perform $\sim 0.3\,\%$ better in a quick pilot run.

3.4 Runtime and reproducibility

All random seeds were fixed to `RND_SEED = 42` (see `Project_1.py`, lines `13-16`) and the exact hyper-parameter table is archived in `hyper-param-errors.csv`.

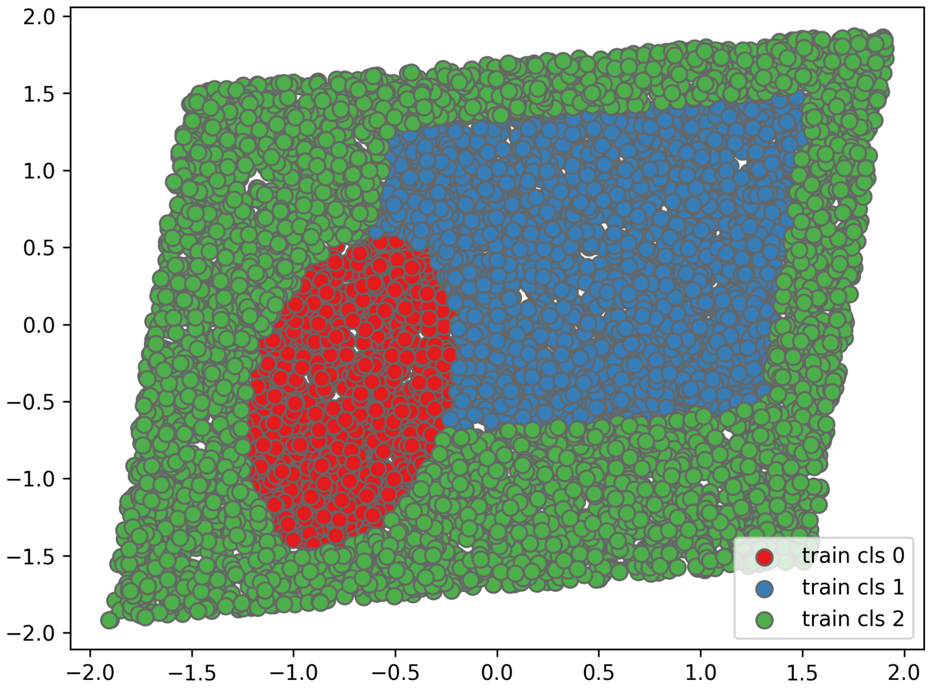

Figure 2: Predicted classes on the *training* set.

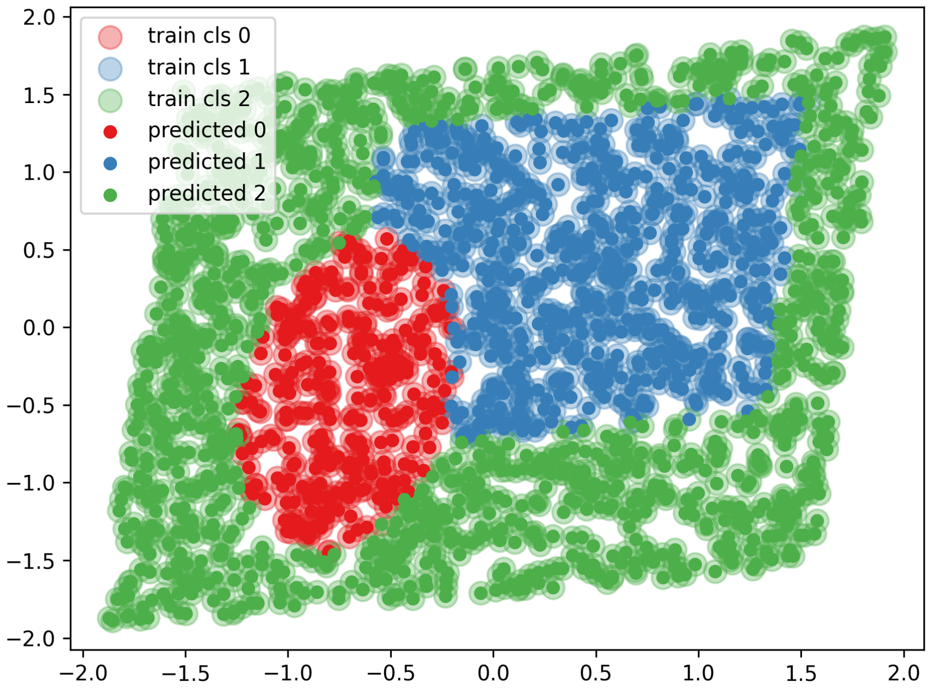

Figure 3: Decision regions projected over the *test* set.