# Comp258

# Neural Networks

# Lab #1 Foundations of Neural Networks

Submitted by: Cody Barker

Submitted for: Ilia Nika

Date: Sept 24th, 2025

Exercise 1:

The purpose of this exercise is to compare the performance of two linear classifiers, Perceptron and ADALINE. Both are trained to recognize the letter “A” in various fonts. Both models were trained and tested using 70/30 spilt and evaluated under conditions of noise and missing data. With the goal to examine how robust each algorithm is to imperfect data.

As a base line trained with only one “A” to train on and the rest were not A, the results found that perceptron was able to correctly identify A and not A more regularly than ADALINE.

Baseline (No Distortions):

Perceptron Accuracy: 100%

ADALINE Accuracy: 71.43%

The Perceptron perfectly classified all test samples, while ADALINE misclassified some distractors (e.g., “K” as “A”).

With Noise at various levels over 10000 trials:

| Noise Level (pixels) | Perceptron Correct (%) | ADALINE Correct (%) |
|---|---|---|
| 5 | 99.99 | 78.50 |
| 10 | 98.10 | 64.40 |
| 15 | 91.10 | 55.10 |
| 20 | 79.1o | 48.80 |

We can see that Perceptron maintains high accuracy until noise becomes too much. ADALINE can be seen struggling early on with even low levels of noise.

With Missing Data at various levels over 10000 trials:

| Missing Data (pixels) | Perceptron Correct (%) | ADALINE Correct (%) |
|---|---|---|
| 5 | 100.00 | 97.50 |
| 10 | 100.00 | 88.50 |
| 15 | 99.90 | 80.80 |
| 20 | 99.95 | 73.40 |
| | | |

Perception is almost unaffected by missing data, while ADALINE while not as steep as noise did struggle compared to Perceptron.

The Perceptron is effective even when noise or missing data is added. Its binary thresholding mechanism makes it tolerant of distortions that do not drastically shift the across the decision boundary. While ADALINE is more sensitive to both noise and missing data because it minimizes squared error on continuous outputs, small changes in input can impact its prediction. Therefore, we can take away that Perceptron is better suited when data is noisy or incomplete, while ADALINE may be better if the data is clean.

Exercise 2:

The purpose of this exercise is to use the Network class from lecture 3 to train a simple MLP for binary classification on Hepatitis dataset. Various architectures and learning parameters were tuned to evaluate how the model performance changes with different hyperparameter settings.

Baseline Results (Network of 19 input neurons, 30 hidden layer 1, 15 hidden layer 2, and 2 output neurons):

Test Accuracy: 90.32%

Confusion Matrix:

$$\begin{bmatrix} 5 & 1 \\ 2 & 23 \end{bmatrix}$$

**Classification Report:**

- Die → Precision: 0.71, Recall: 0.83, F1: 0.77
- Live → Precision: 0.96, Recall: 0.92, F1: 0.94

The hyperparameters the classifier was tested on:

- epoch_options = [10, 20, 50]
- batch_options = [5, 10, 20]
- eta_options = [0.5, 0.1, 0.05]
- architectures = [[19,10,2], [19, 20, 2], [19, 30, 15, 2], [19, 50, 25, 2], [19, 64, 32, 16, 2]]

Hyperparameter Results:

Various Hyperparameter outputs:

Summary of results (some outputs):

- arch=[19, 10, 2], epochs=10, batch=5, eta=0.5 → acc=90.32%
- arch=[19, 10, 2], epochs=10, batch=10, eta=0.5 → acc=77.42%
- arch=[19, 10, 2], epochs=10, batch=10, eta=0.1 → acc=61.29%
- arch=[19, 10, 2], epochs=10, batch=10, eta=0.05 → acc=77.42%
- arch=[19, 10, 2], epochs=10, batch=20, eta=0.5 → acc=83.87%
- arch=[19, 10, 2], epochs=10, batch=20, eta=0.1 → acc=19.35%
- arch=[19, 10, 2], epochs=10, batch=20, eta=0.05 → acc=38.71%
- arch=[19, 10, 2], epochs=20, batch=5, eta=0.5 → acc=90.32%
- arch=[19, 10, 2], epochs=50, batch=5, eta=0.5 → acc=93.55%
- arch=[19, 10, 2], epochs=50, batch=5, eta=0.1 → acc=87.10%
- arch=[19, 10, 2], epochs=50, batch=20, eta=0.1 → acc=77.42%
- arch=[19, 20, 2], epochs=50, batch=10, eta=0.05 → acc=83.87%
- arch=[19, 20, 2], epochs=50, batch=20, eta=0.5 → acc=80.65%
- arch=[19, 20, 2], epochs=50, batch=20, eta=0.1 → acc=19.35%
- arch=[19, 20, 2], epochs=50, batch=20, eta=0.05 → acc=19.35%

Best Result:

- Architecture: [19, 10, 2]
- Epochs : 20
- Batch Size : 5
- Learning Rate: 0.05
- Accuracy : 96.77%

From the output we can say that High n around n=0.5 performed consistently well, while n=0.05 needed more epochs but achieved the best accuracy at 96.77%. Smaller batch sizes gave the better results while larger batches sometimes led to unstable training. Adding more layers did not consistently improve results. Simple architectures performed as well or better than deep ones for this dataset.