COMP30230 Connectionist Computing

Jiayi Yang22207268

December 2025

1 Coding Choice

```
This project follows the given suggestions on implementation, the programming language selected is java.
we have a MLP object with
     * Constructor for MLP
     * @param numInputs Number of input neurons
     * @param numHidden Number of hidden neurons
     * @param numOutputs Number of output neurons
     * @param useTanhHidden true for tanh hidden activation, false for sigmoid
     * @param useLinearOutput true for linear output, false for sigmoid

we have a training object
    /**
     * Constructor for Training
     * @param network The MLP to train
     * @param maxEpochs Maximum number of training epochs
     * @param batchSize Number of examples between weight updates (1 = online, numExamples = batch)
     * @param learningRate Learning rate for gradient descent
     */

then the 3 tests are conducted seperately, logs are written into files
```

2 Test1

```
===========================================
Test1: XOR Function Learning
===========================================

Network Configuration:
- Inputs: 2
- Hidden units: 4
- Outputs: 1
- Hidden activation: sigmoid
- Output activation: sigmoid
```

```
Training Parameters:
- Max epochs: 10000
- Batch size: 4
- Learning rate: 0.5

--- Training Started ---

Error at epoch 0 is 0.5041603383266205
Error at epoch 1000 is 0.4999766039839972
Error at epoch 2000 is 0.49971695443414715
Error at epoch 3000 is 0.15468614761818042
Error at epoch 4000 is 0.008318000284138975
Error at epoch 5000 is 0.0034119256969117637
Error at epoch 6000 is 0.002034817624153567
Error at epoch 7000 is 0.0014146003866927167
Error at epoch 8000 is 0.0010690848156050558
Error at epoch 9000 is 8.515514928271106E-4
Error at epoch 9999 is 7.033376043674808E-4

--- Training Completed ---
Final error: 7.033376043674808E-4

Predictions:
============
Input: [0.0, 0.0] -> Target: [0.0] -> Predicted: [0.0104]
Input: [0.0, 1.0] -> Target: [1.0] -> Predicted: [0.9804]
Input: [1.0, 0.0] -> Target: [1.0] -> Predicted: [0.9812]
Input: [1.0, 1.0] -> Target: [0.0] -> Predicted: [0.0237]

--- Verification ---
Input: (0, 0) -> Expected: 0, Predicted: 0 (0.0104) ?
Input: (0, 1) -> Expected: 1, Predicted: 1 (0.9804) ?
Input: (1, 0) -> Expected: 1, Predicted: 1 (0.9812) ?
Input: (1, 1) -> Expected: 0, Predicted: 0 (0.0237) ?

SUCCESS: All predictions are correct!
```

3 Test2

```
===========================================
Test2: Sin Function Approximation
===========================================

Generating 500 random examples...
Training set: 400 examples
```

```
Test set: 100 examples

Network Configuration:
- Inputs: 4
- Hidden units: 5
- Outputs: 1
- Hidden activation: tanh
- Output activation: linear

Training Parameters:
- Max epochs: 5000
- Batch size: 20
- Learning rate: 0.01

--- Training Started ---

Error at epoch 0 is 62.532136655999466
Error at epoch 500 is 0.12839401786775392
Error at epoch 1000 is 0.07455584785614716
Error at epoch 1500 is 0.05080225344616496
Error at epoch 2000 is 0.03508940540527945
Error at epoch 2500 is 0.024186093332069246
Error at epoch 3000 is 0.01684589822258476
Error at epoch 3500 is 0.012140247988349055
Error at epoch 4000 is 0.009182500293434563
Error at epoch 4500 is 0.0073118003845650745
Error at epoch 4999 is 0.006104545301484009

--- Training Completed ---

===========================================
RESULTS
===========================================

Total Training Error: 0.00519531902222207
Total Test Error: 0.0017417386211084451

Average Training Error per example: 1.2988297555555175E-5
Average Test Error per example: 1.741738621108445E-5

--- Sample Test Predictions (first 10) ---
Input: [-0.605, -0.361, 0.597, 0.163] -> Target: 0.1886, Predicted: 0.1880, Error: 0.0006
Input: [-0.114, -0.936, -0.774, -0.011] -> Target: 0.0591, Predicted: 0.0577, Error: 0.0014
Input: [-0.885, 0.400, -0.235, 0.666] -> Target: -0.8165, Predicted: -0.8128, Error: 0.0037
Input: [0.288, -0.420, -0.107, -0.038] -> Target: 0.5959, Predicted: 0.5930, Error: 0.0029
Input: [-0.489, -0.058, -0.928, 0.020] -> Target: -0.9816, Predicted: -0.9881, Error: 0.0065
```

```
Input: [-0.415, 0.938, 0.558, -0.351] -> Target: -0.4289, Predicted: -0.4236, Error: 0.0052
Input: [0.409, 0.098, -0.781, 0.109] -> Target: -0.5470, Predicted: -0.5491, Error: 0.0022
Input: [-0.658, -0.735, -0.559, 0.363] -> Target: -0.7483, Predicted: -0.7547, Error: 0.0063
Input: [0.858, 0.818, -0.128, 0.199] -> Target: -0.2838, Predicted: -0.2816, Error: 0.0022
Input: [0.315, 0.256, 0.776, -0.875] -> Target: 0.9902, Predicted: 0.9909, Error: 0.0007

===========================================
ANALYSIS
===========================================

Q: What is the error on training at the end?
A: Total training error: 0.005195 (Average: 0.000013 per example)

Q: How does it compare with the error on the test set?
A: Test error is 0.34x the training error
   The errors are very similar, indicating good generalization.

Q: Do you think you have learned satisfactorily?
A: YES - The network has learned the function very well!
   Average error per example is very small.
```

4 Test3

```
===========================================
Test3: Letter Recognition
===========================================

Loading dataset from letter-recognition.csv...
Loaded 20000 examples
Training set: 16000 examples
Test set: 4000 examples

Network Configuration:
- Inputs: 16
- Hidden units: 30
- Outputs: 26
- Hidden activation: tanh
- Hidden units: 30
- Outputs: 26
- Hidden activation: tanh
- Output activation: sigmoid

Training Parameters:
- Output activation: sigmoid

Training Parameters:
```

```
- Max epochs: 5000
- Batch size: 100
- Learning rate: 0.01

--- Training Started ---


Error at epoch 0 is 8265.21436442745

Error at epoch 0 is 8265.21436442745
Error at epoch 500 is 2112.6050072962125
Error at epoch 1000 is 1982.4466366492686

Error at epoch 0 is 8265.21436442745
Error at epoch 500 is 2112.6050072962125

Error at epoch 0 is 8265.21436442745

Error at epoch 0 is 8265.21436442745
Error at epoch 500 is 2112.6050072962125
Error at epoch 1000 is 1982.4466366492686
Error at epoch 1500 is 1936.8473632489172
Error at epoch 2000 is 1905.514412464231
Error at epoch 2500 is 1873.1920323126003
Error at epoch 3000 is 1844.036526235373
Error at epoch 3500 is 1811.4258809736314
Error at epoch 4000 is 1788.2642524779033
Error at epoch 4500 is 1771.3316637378655
Error at epoch 4999 is 1756.4554063836613

--- Training Completed in 824.879 seconds ---

===========================================
RESULTS
===========================================

Final Training Error: 1756.4554

Training Accuracy: 13276/16000 (82.98%)
Test Accuracy: 3277/4000 (81.93%)

--- Per-Letter Test Accuracy ---
A: 140/158 (88.6%)
B: 145/155 (93.5%)
C: 112/148 (75.7%)
D: 142/159 (89.3%)
```

```
E: 112/148 (75.7%)
F: 108/139 (77.7%)
G: 114/161 (70.8%)
H: 44/165 (26.7%)
I: 116/139 (83.5%)
J: 129/170 (75.9%)
K: 134/150 (89.3%)
L: 124/147 (84.4%)
M: 178/192 (92.7%)
N: 138/167 (82.6%)
O: 107/130 (82.3%)
P: 146/162 (90.1%)
Q: 115/135 (85.2%)
R: 147/167 (88.0%)
S: 112/142 (78.9%)
T: 122/152 (80.3%)
U: 132/145 (91.0%)
V: 144/158 (91.1%)
W: 132/155 (85.2%)
X: 117/161 (72.7%)
Y: 131/142 (92.3%)
Z: 136/153 (88.9%)

--- Sample Test Predictions (first 20) ---
Actual: C, Predicted: K (confidence: 0.014) ?
Actual: F, Predicted: F (confidence: 0.991) ?
Actual: Z, Predicted: Z (confidence: 1.000) ?
Actual: R, Predicted: R (confidence: 0.632) ?
Actual: M, Predicted: M (confidence: 1.000) ?
Actual: T, Predicted: T (confidence: 1.000) ?
Actual: G, Predicted: O (confidence: 1.000) ?
Actual: S, Predicted: S (confidence: 0.999) ?
Actual: U, Predicted: U (confidence: 0.972) ?
Actual: N, Predicted: N (confidence: 1.000) ?
Actual: Z, Predicted: Z (confidence: 0.984) ?
Actual: G, Predicted: O (confidence: 1.000) ?
Actual: S, Predicted: S (confidence: 0.999) ?
Actual: U, Predicted: U (confidence: 0.972) ?
Actual: N, Predicted: N (confidence: 1.000) ?
Actual: Z, Predicted: Z (confidence: 0.984) ?
Actual: U, Predicted: U (confidence: 0.972) ?
Actual: N, Predicted: N (confidence: 1.000) ?
Actual: Z, Predicted: Z (confidence: 0.984) ?
Actual: F, Predicted: F (confidence: 0.983) ?
Actual: D, Predicted: D (confidence: 0.654) ?
Actual: E, Predicted: E (confidence: 0.988) ?
```

```
Actual: D, Predicted: D (confidence: 0.749) ?
Actual: D, Predicted: D (confidence: 0.749) ?
Actual: O, Predicted: O (confidence: 1.000) ?
Actual: O, Predicted: O (confidence: 1.000) ?
Actual: R, Predicted: R (confidence: 0.134) ?
Actual: R, Predicted: R (confidence: 0.134) ?
Actual: U, Predicted: U (confidence: 1.000) ?
Actual: H, Predicted: K (confidence: 0.266) ?
Actual: Z, Predicted: Z (confidence: 0.855) ?

===========================================
ANALYSIS
===========================================

Q: How well can you classify the test data?
A: The network achieved 81.93% accuracy on the test set.
   This is excellent performance for letter recognition!

Note: Random baseline would be ~3.85% (1/26 chance)
The original research achieved ~80% accuracy.
```

5 Conclusion

What I have learned: Three main parameters for training

Epoch: This is one full cycle through the entire dataset. If I study a textbook from start to finish once, that's one epoch. In my code, this represents how many times the network sees every example in the training data.

Batch Size: Instead of updating the model after every single example, we group them. The batch size is how many examples the model looks at together before it decides to update its internal numbers.

Learning Rate: This controls how big of a step the model takes when it tries to fix an error. If the step is too big, it overshoots the target; if it's too small, it barely moves.

How the MLP works: The actual training process is a cycle. First, in the forward pass, the MLP takes the inputs (like 0 and 1) and pushes them through the hidden layers to make a guess. It's usually wrong at first. We then calculate the error and do a backward pass (backpropagation). This is where the math happens—we look at the error and trace it back through the network to see which connections were responsible. Finally, we do the weight update, slightly tweaking the connections to reduce that error for the next time.

Test Results Analysis:

Test 1: Why did a Learning Rate ¡ 0.5 (not exactly, but smaller than some value between 0.5 and 0.1) fail? I found that when the learning rate was too low, the model performed poorly. For XOR problem, if the learning rate is too small, the model takes tiny steps and often gets stuck in a local minimum.

Test 2: The test showed that 5 hidden units worked better than 20. This might be caused by overfitting due to the complexity of hidden layer, 5 units can generalize on the sin function.

Test 3: Precision at 5000 epochs versus 10000 epochs are similar. This suggests that the model had already converged by around epoch 5000. Additional 5000 epoch does not reduce the loss. Also, letter H is not well recognized, probably due to its shape similar to other letters.