Assignment (1)

February 22, 2025

[1]:
```
import numpy as np
```

[2]:
```
# Define tanh activation function
def tanh(x):
    return np.tanh(x)
```

[3]:
```
# Initialize weights randomly in the range [-0.5, 0.5]
def init_weight():
    return np.random.uniform(-0.5, 0.5)
```

[4]:
```
# Network structure
network = {
    'inputs': {'i1': 0.05, 'i2': 0.10},
    'biases': {'b1': 0.5, 'b2': 0.7},
    'weights': {
        'w1': init_weight(), 'w2': init_weight(),
        'w3': init_weight(), 'w4': init_weight(),
        'w5': init_weight(), 'w6': init_weight(),
        'w7': init_weight(), 'w8': init_weight()
    }
}
```

[5]:
```
def forward_pass(network):
    # Hidden layer computations
    h1_input = network['inputs']['i1'] * network['weights']['w1'] +␣
 ↪network['inputs']['i2'] * network['weights']['w3'] + network['biases']['b1']
    h2_input = network['inputs']['i1'] * network['weights']['w2'] +␣
 ↪network['inputs']['i2'] * network['weights']['w4'] + network['biases']['b1']
    h1_output = tanh(h1_input)
    h2_output = tanh(h2_input)

    # Output layer computations
    o1_input = h1_output * network['weights']['w5'] + h2_output *␣
 ↪network['weights']['w7'] + network['biases']['b2']
    o2_input = h1_output * network['weights']['w6'] + h2_output *␣
 ↪network['weights']['w8'] + network['biases']['b2']
```

```
    o1_output = tanh(o1_input)
    o2_output = tanh(o2_input)

    return {
        'h1_input': h1_input, 'h2_input': h2_input,
        'h1_output': h1_output, 'h2_output': h2_output,
        'o1_input': o1_input, 'o2_input': o2_input,
        'o1_output': o1_output, 'o2_output': o2_output
    }
```

[6]:
```
def compute_error(results):
    target_o1 = 0.01
    target_o2 = 0.99
    error_o1 = 0.5 * (target_o1 - results['o1_output']) ** 2
    error_o2 = 0.5 * (target_o2 - results['o2_output']) ** 2
    return error_o1 + error_o2
```

[7]:
```
# Perform forward pass
results = forward_pass(network)
print(f"Output o1: {results['o1_output']}")
print(f"Output o2: {results['o2_output']}")
```

```
Output o1: 0.5850921382371571
Output o2: 0.5763709028886127
```

[8]:
```
# Compute error
error = compute_error(results)
print(f"Error: {error}")
```

```
Error: 0.25090999871968345
```

[9]:
```
print("")
```