COIT20277 Introduction to Artificial Intelligence

Week 6

Deep Learning

- AI, Machine Learning and Deep Learning
- Artificial Neural Networks

CQUniversity AUSTRALIA

BE WHAT YOU WANT TO BE
cqu.edu.au

Acknowledgement of Country

I respectfully acknowledge the Traditional Custodians of the land on which we live, work and learn. I pay my respects to the First Nations people and their Elders, past, present and future

CQUniversity AUSTRALIA

BE WHAT YOU WANT TO BE
cqu.edu.au

The Perceptron Program

- **TASK:** Modify this program so that it performs a **logical OR** of the inputs instead.

jupyter logical_AND_perceptron Last Checkpoint: 4 days ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[ ]:
```python
# A perceptron for the logcal AND operation of its input
import numpy as np

def activation_function(x):
    # Step activation function
    return 1 if x >= 0 else 0

def predict(inputs, weights, bias):
    # Calculate weighted sum of inputs and apply activation function
    weighted_sum = np.dot(inputs, weights) + bias
    return activation_function(weighted_sum)

# Initialize weights with random values
weights = np.random.rand(2)
bias = np.random.rand()

# Test the perceptron with logical AND operation
def test_logical_and():
    # Logical AND truth table
    test_cases = [[0, 0], [0, 1], [1, 0], [1, 1]]
    expected_outputs = [0, 0, 0, 1]

    # Test each case
    for inputs, expected_output in zip(test_cases, expected_outputs):
        output = predict(inputs, weights, bias)
        print(f"Inputs: {inputs}, Predicted Output: {output}, Expected Output: {expected_output}")

if __name__ == "__main__":
    test_logical_and()
```

The Multiple Layer Neural Network

- **TASK:** Modify this program so that it performs a **logical XOR** of the inputs instead.

jupyter Untitled Last Checkpoint: 5 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [13]:

```python
from sklearn.neural_network import MLPClassifier
import numpy as np

# Define the logical AND truth table
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]])
y = np.array([0, 0, 0, 1])  # Output for logical AND

# Create and configure the MLPClassifier
clf = MLPClassifier(hidden_layer_sizes=(5, 2), activation='relu', solver='adam', max_iter=10000, random_state=1)

# Train the classifier
clf.fit(X, y)

# Test the trained classifier
test_cases = [[0, 0], [0, 1], [1, 0], [1, 1]]
for inputs in test_cases:
    output = clf.predict([inputs])[0]
    print(f"Inputs: {inputs}, Predicted Output: {output}")
```

```
Inputs: [0, 0], Predicted Output: 0
Inputs: [0, 1], Predicted Output: 0
Inputs: [1, 0], Predicted Output: 0
Inputs: [1, 1], Predicted Output: 1
```

THANK YOU

TIME FOR DISCUSSION & QUESTIONS