Computer Intelligence Lab - CS354N

Weekly Assignment Report

Question 1:

```C/C++
#include <iostream>
#include <vector>
#include <fstream>
#include <sstream>
#include <ctime>
#include <cstdlib>


using namespace std;


struct Perceptron {
    vector<double> weights;
    double eta;  // Learning rate
    int theta;   // Threshold
    int bias_ip; // Bias input


    Perceptron(double eta, int theta, int input_size) {
        this->eta = eta;
        this->theta = theta;
        this->bias_ip = 1;
```

```
        weights = vector<double>(input_size + 1);
        for (auto &w : weights) {
            w = (rand() % 200 - 100) / 100.0;  // Random weights between -1 and
1
        }
    }

    void train(const vector<pair<vector<int>, int>> &data, int max_epochs) {
        bool flag = true;
        int epoch = 0;

        while (flag && epoch < max_epochs) {
            flag = false;
            cout << "Epoch " << epoch + 1 << ":\n";
            for (const auto &sample : data) {
                const vector<int> &inputs = sample.first;
                int target = sample.second;
                double net = weights[0] * bias_ip;
                for (size_t i = 0; i < inputs.size(); ++i) {
                    net += weights[i + 1] * inputs[i];
                }
                int output = (net >= theta) ? 1 : 0;
```

```
            int error = target - output;
            if (error != 0) {
                flag = true;
                weights[0] += eta * error * bias_ip;
                for (size_t i = 0; i < inputs.size(); ++i) {
                    weights[i + 1] += eta * error * inputs[i];
                }
                cout << "Updated weights: ";
                for (double weight : weights) {
                    cout << weight << " ";
                }
                cout << endl;
            }
        }

        // Testing after each epoch
        cout << "Testing after Epoch " << epoch + 1 << ":\n";
        for (const auto &sample : data) {
            const vector<int> &inputs = sample.first;
            int target = sample.second;
            int output = predict(inputs);
            cout << "Inputs: ";
            for (int input : inputs) {
```

```
                    cout << input << " ";
                }
                cout << "Output: " << output << " Target: " << target << endl;
            }
            cout << "------------------------------\n";

            epoch++;
        }

        cout << "Training completed in " << epoch << " epochs." << endl;
    }

    int predict(const vector<int> &inputs) {
        double net = weights[0] * bias_ip;
        for (size_t i = 0; i < inputs.size(); ++i) {
            net += weights[i + 1] * inputs[i];
        }
        return (net >= theta) ? 1 : 0;
    }
};

vector<pair<vector<int>, int>> load_data(const string &filename) {
    vector<pair<vector<int>, int>> data;
```

```
    ifstream file(filename);
    string line;

    while (getline(file, line)) {
        stringstream ss(line);
        vector<int> inputs;
        int input, target;

        while (ss >> input) {
            inputs.push_back(input);
        }
        target = inputs.back();
        inputs.pop_back();
        data.push_back({inputs, target});
    }

    return data;
}

int main() {
    srand(time(0));

    int choice;
```

```
cout << "Select the gate to train the perceptron model:\n";
cout << "1. AND\n";
cout << "2. NAND\n";
cout << "3. OR\n";
cout << "4. NOR\n";
cout << "Enter your choice: ";
cin >> choice;

string filename;
switch (choice) {
    case 1:
        filename = "example_and.txt";
        break;
    case 2:
        filename = "example_nand.txt";
        break;
    case 3:
        filename = "example_or.txt";
        break;
    case 4:
        filename = "example_nor.txt";
        break;
    default:
```

```
            cout << "Invalid choice!" << endl;
            return 1;
    }

    vector<pair<vector<int>, int>> data = load_data(filename);
    if (data.empty()) {
        cout << "Failed to load data from " << filename << endl;
        return 1;
    }

    int input_size = data[0].first.size();
    double eta = (rand() % 100 + 1) / 200.0; // Random learning rate between
0.005 and 0.5
    int max_epochs = rand() % 100 + 1;       // Random number of epochs between
1 and 100

    Perceptron perceptron(eta, 0, input_size);
    perceptron.train(data, max_epochs);
    cout << "Trained weights: ";
    for (double weight : perceptron.weights) {
        cout << weight << " ";
    }
    cout << endl;
```

```
    cout << "Final Predictions:" << endl;
    bool converged = true;
    for (const auto &sample : data) {
        const vector<int> &inputs = sample.first;
        int target = sample.second;
        int output = perceptron.predict(inputs);
        cout << "Inputs: ";
        for (int input : inputs) {
            cout << input << " ";
        }
        cout << "Output: " << output << " Target: " << target << endl;
        if (output != target) {
            converged = false;
        }
    }
    if (converged) {
        cout << "Training has converged (target = predicted value) for every
input." << endl;
    } else {
        cout << "Training has not converged for some inputs." << endl;
    }
    return 0;
}
```

Output:

```
┌──(thenetherwatcher㉿kali)-[~/Documents/CS354-Lab/Lab-5]
└─$ g++ q1.cpp

┌──(thenetherwatcher㉿kali)-[~/Documents/CS354-Lab/Lab-5]
└─$ ./a.out
Select the gate to train the perceptron model:
1. AND
2. NAND
3. OR
4. NOR
Enter your choice: 1
Epoch 1:
Updated weights: -0.445 0.76 0.23 0.085
Updated weights: -0.65 0.555 0.23 0.085
Updated weights: -0.855 0.35 0.025 0.085
Updated weights: -0.65 0.555 0.23 0.29
Testing after Epoch 1:
Inputs: 0 0 0 Output: 0 Target: 0
Inputs: 0 0 1 Output: 0 Target: 0
Inputs: 0 1 0 Output: 0 Target: 0
Inputs: 0 1 1 Output: 0 Target: 0
Inputs: 1 0 0 Output: 0 Target: 0
Inputs: 1 0 1 Output: 1 Target: 0
Inputs: 1 1 0 Output: 1 Target: 0
Inputs: 1 1 1 Output: 1 Target: 1
--------------------------------
Epoch 2:
Updated weights: -0.855 0.35 0.23 0.085
Updated weights: -0.65 0.555 0.435 0.29
Testing after Epoch 2:
Inputs: 0 0 0 Output: 0 Target: 0
Inputs: 0 0 1 Output: 0 Target: 0
Inputs: 0 1 0 Output: 0 Target: 0
Inputs: 0 1 1 Output: 1 Target: 0
Inputs: 1 0 0 Output: 0 Target: 0
Inputs: 1 0 1 Output: 1 Target: 0
Inputs: 1 1 0 Output: 1 Target: 0
Inputs: 1 1 1 Output: 1 Target: 1
--------------------------------
Epoch 3:
Updated weights: -0.855 0.555 0.23 0.085
Testing after Epoch 3:
Inputs: 0 0 0 Output: 0 Target: 0
Inputs: 0 0 1 Output: 0 Target: 0
Inputs: 0 1 0 Output: 0 Target: 0
Inputs: 0 1 1 Output: 0 Target: 0
Inputs: 1 0 0 Output: 0 Target: 0
Inputs: 1 0 1 Output: 0 Target: 0
Inputs: 1 1 0 Output: 0 Target: 0
Inputs: 1 1 1 Output: 1 Target: 1
--------------------------------
Epoch 4:
Testing after Epoch 4:
Inputs: 0 0 0 Output: 0 Target: 0
Inputs: 0 0 1 Output: 0 Target: 0
Inputs: 0 1 0 Output: 0 Target: 0
Inputs: 0 1 1 Output: 0 Target: 0
Inputs: 1 0 0 Output: 0 Target: 0
Inputs: 1 0 1 Output: 0 Target: 0
Inputs: 1 1 0 Output: 0 Target: 0
Inputs: 1 1 1 Output: 1 Target: 1
--------------------------------
Training completed in 4 epochs.
Trained weights: -0.855 0.555 0.23 0.085
Final Predictions:
Inputs: 0 0 0 Output: 0 Target: 0
Inputs: 0 0 1 Output: 0 Target: 0
Inputs: 0 1 0 Output: 0 Target: 0
Inputs: 0 1 1 Output: 0 Target: 0
Inputs: 1 0 0 Output: 0 Target: 0
Inputs: 1 0 1 Output: 0 Target: 0
Inputs: 1 1 0 Output: 0 Target: 0
Inputs: 1 1 1 Output: 1 Target: 1
Training has converged (target = predicted value) for every input.
```

Here, if the weights are properly initialized, then the model will converge quite early(~3-5 epochs), but if they are poorly initialized or small learning rate is taken, then it takes some time for the training to converge, but it ultimately does.

Question 2:

```
C/C++

#include <iostream>

#include <vector>

#include <cstdlib>


using namespace std;


struct Perceptron {

    vector<double> weights;

    double eta;  // Learning rate

    int bias_ip; // Bias input


    Perceptron(double eta, int input_size) {

        this->eta = eta;

        this->bias_ip = 1;

        weights = vector<double>(input_size + 1, 0); // Including bias weight

    }


    void train(const vector<pair<vector<double>, int>> &data, int max_epochs) {

        bool flag = true;
```

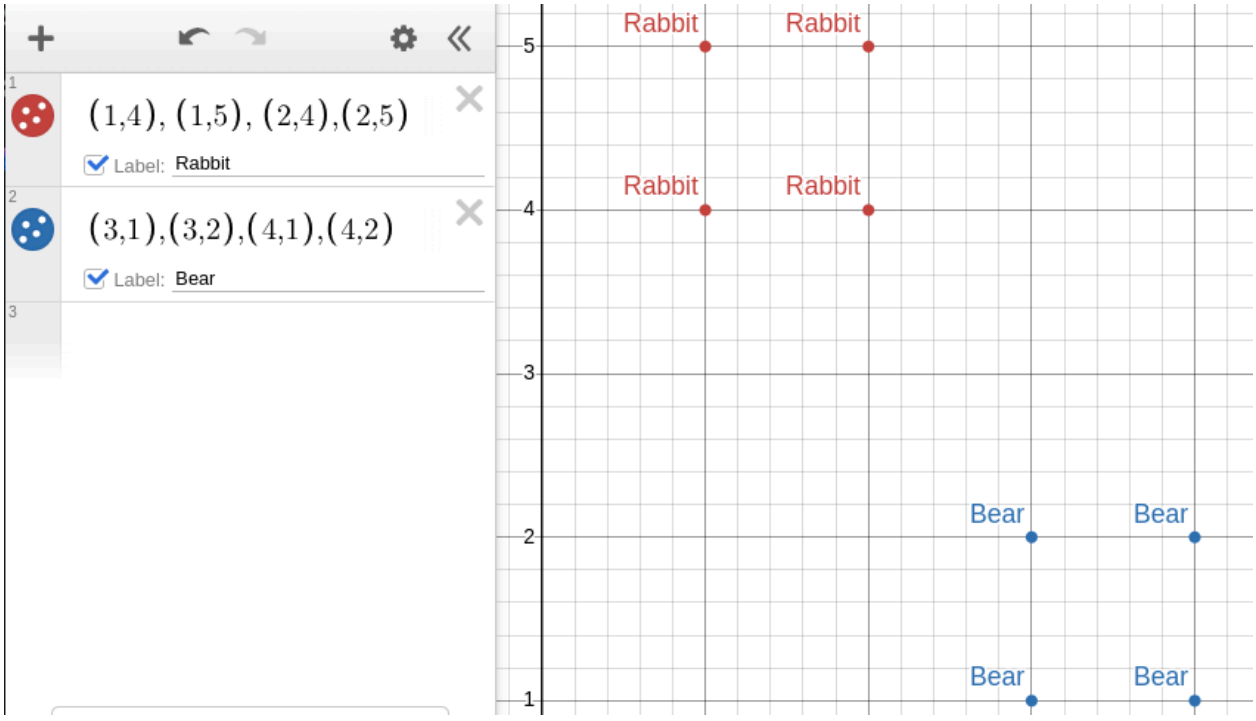
```
int epoch = 0;


while (flag && epoch < max_epochs) {
    flag = false;
    cout << "Epoch " << epoch + 1 << ":\n";
    for (const auto &sample : data) {
        const vector<double> &inputs = sample.first;
        int target = sample.second;


        double net = weights[0] * bias_ip;
        for (size_t i = 0; i < inputs.size(); ++i) {
            net += weights[i + 1] * inputs[i];
        }
        int output = (net >= 0) ? 1 : 0;


        int error = target - output;
        if (error != 0) {
            flag = true;
            weights[0] += eta * error * bias_ip;
            for (size_t i = 0; i < inputs.size(); ++i) {
                weights[i + 1] += eta * error * inputs[i];
            }
            cout << "Updated weights: ";
```

```
                    for (double weight : weights) {
                        cout << weight << " ";
                    }
                    cout << endl;
                }
            }
            epoch++;
            cout << "------------------------------\n";
        }

        cout << "Training completed in " << epoch << " epochs." << endl;
    }

    int predict(const vector<double> &inputs) {
        double net = weights[0] * bias_ip;
        for (size_t i = 0; i < inputs.size(); ++i) {
            net += weights[i + 1] * inputs[i];
        }
        return (net >= 0) ? 1 : 0;
    }
};

int main() {
```

```
    vector<pair<vector<double>, int>> data = {
        {{1.0, 2.0}, 0},  // Rabbit
        {{1.5, 2.5}, 0},  // Rabbit
        {{2.0, 3.0}, 0},  // Rabbit
        {{4.0, 1.0}, 1},  // Bear
        {{4.5, 1.5}, 1},  // Bear
        {{5.0, 2.0}, 1}   // Bear
    };

    Perceptron perceptron(0.1, 2); // Learning rate = 0.1, 2 features (weight, ear length)
    perceptron.train(data, 100);   // Train for up to 100 epochs

    cout << "Trained weights: ";
    for (double weight : perceptron.weights) {
        cout << weight << " ";
    }
    cout << endl;

    cout << "Predictions:\n";
    for (const auto &sample : data) {
        const vector<double> &inputs = sample.first;
        int target = sample.second;
```

```
        int output = perceptron.predict(inputs);

        cout << "Inputs (Weight, Ear Length): (" << inputs[0] << ", " <<
inputs[1] << ") ";

        cout << "Predicted: " << (output == 0 ? "Rabbit" : "Bear") << ", Target:
" << (target == 0 ? "Rabbit" : "Bear") << endl;

    }


    return 0;

}
```

Data Visualisation



Output:

```
┌──(thenetherwatcher㉿kali)-[~/Documents/CS354-Lab/Lab-5]
└─$ g++ q2.cpp

┌──(thenetherwatcher㉿kali)-[~/Documents/CS354-Lab/Lab-5]
└─$ ./a.out
Epoch 1:
Updated weights: -0.1 -0.1 -0.2
Updated weights: 0 0.3 -0.1
--------------------------------
Epoch 2:
Updated weights: -0.1 0.2 -0.3
--------------------------------
Epoch 3:
--------------------------------
Training completed in 3 epochs.
Trained weights: -0.1 0.2 -0.3
Predictions:
Inputs (Weight, Ear Length): (1, 2) Predicted: Rabbit, Target: Rabbit
Inputs (Weight, Ear Length): (1.5, 2.5) Predicted: Rabbit, Target: Rabbit
Inputs (Weight, Ear Length): (2, 3) Predicted: Rabbit, Target: Rabbit
Inputs (Weight, Ear Length): (4, 1) Predicted: Bear, Target: Bear
Inputs (Weight, Ear Length): (4.5, 1.5) Predicted: Bear, Target: Bear
Inputs (Weight, Ear Length): (5, 2) Predicted: Bear, Target: Bear
```