模式识别第一次课程作业报告

57119101王晨阳

# 问题描述

编写感知器算法程序，求下列模式分类的解向量：

设

# 算法介绍

我们首先定义 Vector 结构用来存储向量：

struct Vector{

    int row;

    vector<int> vec;

};

然后定义 TrainingSample 结构来存储训练样本：

struct TrainingSample{

    Vector vec;

    int tag;

};

然后编写函数用于矩阵运算。本题中需要用到向量乘法、向量数乘、向量加法。下面是它们对应的程序：

auto Multiplication(Vector W, Vector X){

    int ret = 0;

    for (int i = 0; i < W.row; i++){

        ret += (W.vec[i] \* X.vec[i]);

    }

    return ret;

}

auto ScalarMultiplication(int c, Vector X){

    Vector ret;

    for (int i = 0; i < X.row; i++){

        ret.vec.push\_back(c \* X.vec[i]);

    }

    return ret;

}

auto Addition(Vector W, Vector X){

    Vector ret;

    ret.row = W.row;

    for (int i = 0; i < W.row; i++){

        ret.vec.push\_back(W.vec[i] + X.vec[i]);

    }

    return ret;

}

对于感知器算法，我们需要知道：训练样本、权向量初始值、校正增量系数。

算法的流程为：

* 选择 个分属于 和 的模式识别样本构成训练样本集，对样本规范化处理，对 的所有样本乘以 。编号为 ，任取权向量初始值 开始迭代
* 第 次迭代时，输入一个样本
* 为校正增量系数，
* 若上一轮迭代未发生错误，则迭代结束

简单模拟即可。程序如下：

auto Perception(vector<TrainingSample> Ts, Vector W, int c){

    bool flag = false;

    Vector w\_pre = W;

    Vector w\_tmp;

    int k = 0;

    w\_pre.row = w\_tmp.row = W.row;

    while (!flag){

        k++;

        flag = true;

        for (int i = 0; i < Ts.size(); i++){

            if (Multiplication(w\_pre, Ts[i].vec) > 0){

                w\_tmp = w\_pre;

            }

            else{

                w\_tmp = Addition(w\_pre, ScalarMultiplication(c, Ts[i].vec));

                w\_pre = w\_tmp;

                flag = false;

            }

        }

    }

    return w\_tmp;

}

# 结果分析

我们在 Ubuntu 20.04.3 LTS, 5.10.60.1-microsoft-standard-WSL2, gcc 9.3.0 上测试程序。

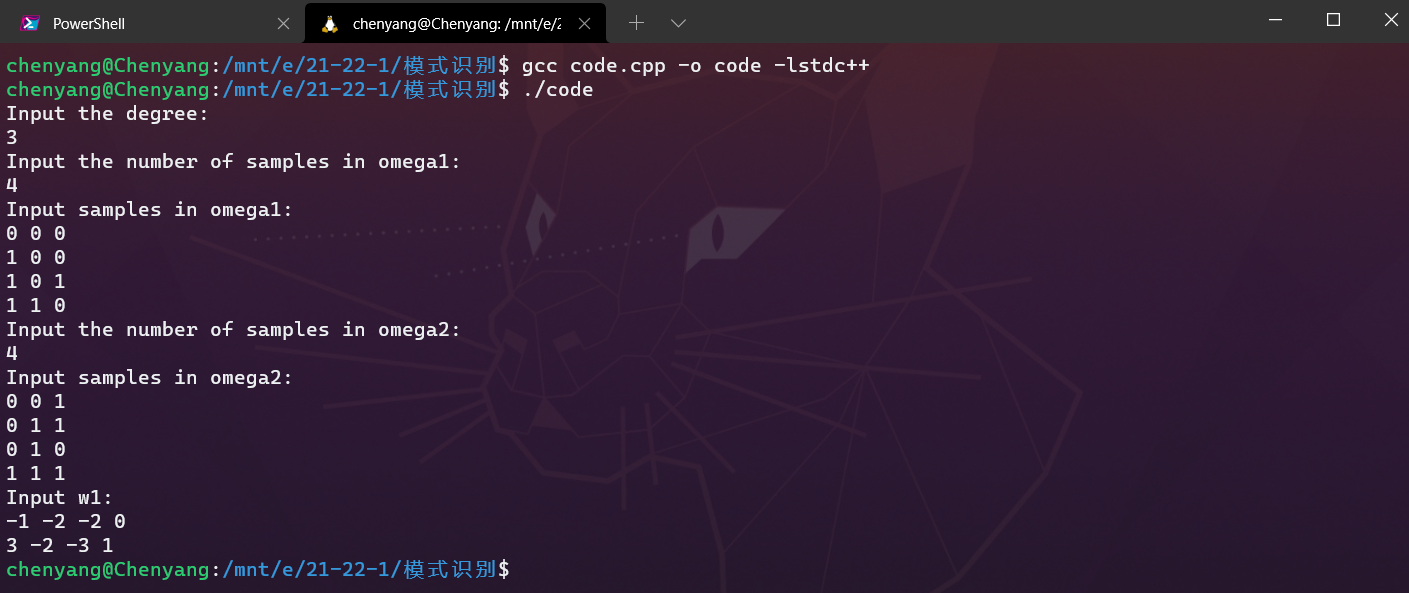


图1 Windows terminal 截图

输入题目所给数据，输出结果为

经验证，结果无误。

# 附录

全部程序：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

 \* @author Chenyang

 \* @date 2021-10-28

 \* @description:

 \*   Perception algorithm

 \* @example

 \*   input:

-----------------------------------------------------------------------

Input the degree:

3

Input the number of samples in omega1:

4

Input samples in omega1:

0 0 0

1 0 0

1 0 1

1 1 0

Input the number of samples in omega2:

4

Input samples in omega2:

0 0 1

0 1 1

0 1 0

1 1 1

Input w1:

-1 -2 -2 0

-----------------------------------------------------------------------

 \*   output:

-----------------------------------------------------------------------

3 -2 -3 1

-----------------------------------------------------------------------

 \* @test passed on Ubuntu 20.04.3 LTS, gcc 9.3.0

 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <vector>

using namespace std;

struct Vector{

    int row;

    vector<int> vec;

};

struct TrainingSample{

    Vector vec;

    int tag;

};

auto Multiplication(Vector W, Vector X){

    int ret = 0;

    for (int i = 0; i < W.row; i++){

        ret += (W.vec[i] \* X.vec[i]);

    }

    return ret;

}

auto ScalarMultiplication(int c, Vector X){

    Vector ret;

    for (int i = 0; i < X.row; i++){

        ret.vec.push\_back(c \* X.vec[i]);

    }

    return ret;

}

auto Addition(Vector W, Vector X){

    Vector ret;

    ret.row = W.row;

    for (int i = 0; i < W.row; i++){

        ret.vec.push\_back(W.vec[i] + X.vec[i]);

    }

    return ret;

}

auto Perception(vector<TrainingSample> Ts, Vector W, int c){

    bool flag = false;

    Vector w\_pre = W;

    Vector w\_tmp;

    int k = 0;

    w\_pre.row = w\_tmp.row = W.row;

    while (!flag){

        k++;

        flag = true;

        for (int i = 0; i < Ts.size(); i++){

            if (Multiplication(w\_pre, Ts[i].vec) > 0){

                w\_tmp = w\_pre;

            }

            else{

                w\_tmp = Addition(w\_pre, ScalarMultiplication(c, Ts[i].vec));

                w\_pre = w\_tmp;

                flag = false;

            }

        }

    }

    return w\_tmp;

}

int main(){

    /\*\*

     \* @description: input

     \*/

    vector<TrainingSample> Ts;

    Vector w1;

    int degree, numOfOmega1, numOfOmega2;

    cout << "Input the degree:\n";cin >> degree;

    cout << "Input the number of samples in omega1:\n";cin >> numOfOmega1;

    cout << "Input samples in omega1:\n";

    for (auto i = 0; i < numOfOmega1; i++){

        int x;TrainingSample tmpTrainingSample;

        tmpTrainingSample.vec.row = degree + 1;

        for (auto j = 0; j < degree; j++){cin >> x;tmpTrainingSample.vec.vec.push\_back(x);}

        tmpTrainingSample.vec.vec.push\_back(1);tmpTrainingSample.tag = 1;

        Ts.push\_back(tmpTrainingSample);

    }

    cout << "Input the number of samples in omega2:\n";cin >> numOfOmega2;

    cout << "Input samples in omega2:\n";

    for (auto i = 0; i < numOfOmega2; i++){

        int x;TrainingSample tmpTrainingSample;

        tmpTrainingSample.vec.row = degree + 1;

        for (auto j = 0; j < degree; j++){cin >> x;tmpTrainingSample.vec.vec.push\_back(-x); }

        tmpTrainingSample.vec.vec.push\_back(-1);tmpTrainingSample.tag = 2;

        Ts.push\_back(tmpTrainingSample);

    }

    cout << "Input w1:\n";

    int x;

    for (auto i = 0; i < degree + 1; i++){

        cin >> x;w1.vec.push\_back(x);

        w1.row = degree + 1;

    }

    /\*\*

     \* @description: main function

     \*/

    Vector ans = Perception(Ts, w1, 1);

    /\*\*

     \* @description: output

     \*/

    for (auto i = 0; i < ans.row; i++){

        cout << ans.vec[i] << ' ';

    }

    cout << endl;

    return 0;

}