模式识别与机器学习

081203M04004H

Chap 3 课程作业解答

2022 年 10 月 11 号

Professor: 黄庆明

学生: 周胤昌
学号: 202228018670052
学院: 网络空间安全学院
所属专业: 网络空间安全
方向: 安全协议理论与技术

Problem 1

在一个 10 类的模式识别问题中, 有 3 类单独满足多类情况 1, 其余的类别满足多类情况 2. 问该模式识别问题所需判别函数的最少数目是多少?

Solution: 将 10 类问题可看作 4 类满足多类情况 1 的问题, 可将 3 类单独满足多类情况 1 的类找出来, 剩下的 7 类全部划到 4 类中剩下的一个子类中. 再在此子类中, 运用多类情况 2 的判别法则进行分类, 此时需要 $7*(7-1)/2=21$ 个判别函数. 故共需要 4+21=25 个判别函数.

Problem 2

一个三类问题, 其判别函数如下:

$$d_1(\boldsymbol{x})=-x_1,\ d_2(\boldsymbol{x})=x_1+x_2-1,\ d_3(\boldsymbol{x})=x_1-x_2-1$$

(1). 设这些函数是在多类情况 1 条件下确定的, 绘出其判别界面和每一个模式类别的区域.

(2). 设为多类情况 2, 并使: $d_{12}(\boldsymbol{x})=d_1(\boldsymbol{x}), d_{13}(\boldsymbol{x})=d_2(\boldsymbol{x}), d_{23}(\boldsymbol{x})=d_3(\boldsymbol{x})$. 绘出其判别界面和多类情况 2 的区域.

(3). 设 $d_1(\boldsymbol{x}), d_2(\boldsymbol{x})$ 和 $d_3(\boldsymbol{x})$ 是在多类情况 3 的条件下确定的, 绘出其判别界面和每类的区域.

Solution: 三种情况分别如下图1中所示:

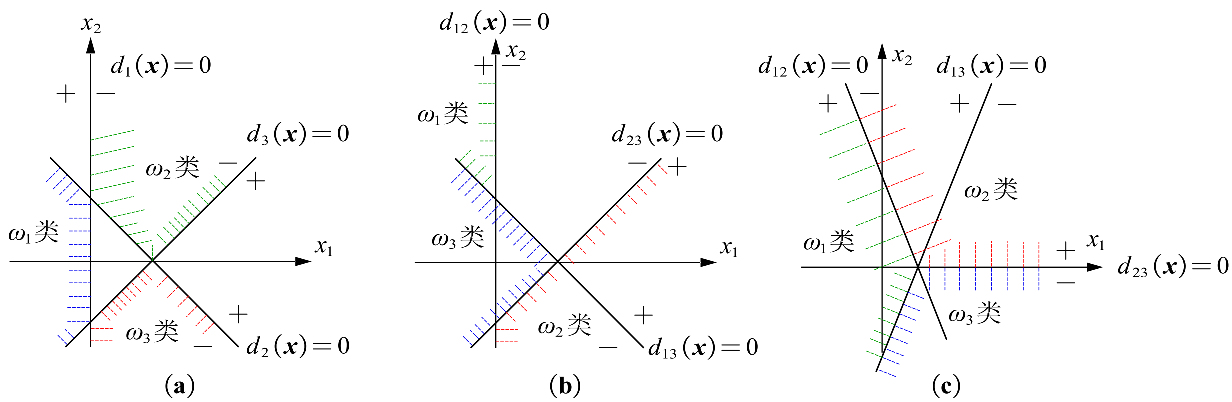

图 1: 判别界面与区域

Problem 3

两类模式, 每类包括 5 个 3 维不同的模式, 且良好分布. 如果它们是线性可分的, 问权向量至少需要几个系数分量? 假如要建立二次的多项式判别函数, 又至少需要几个系数分量? (设模式的良好分布不因模式变化而改变)

Solution: (1). 若是线性可分的, 则权向量需要至少 $n+1=4$ 个系数分量;

(2). 根据公式易知此情形下的系数分量个数至少为 $N_\omega=C_{n+r}^r=\dfrac{(n+r)!}{r!n!}=\dfrac{5!}{2!3!}=10.$

Problem 4

1. 用感知器算法求下列模式分类的解向量 $\boldsymbol{w}$:

$$\begin{aligned}\omega_1 &= \left\{(0,0,0)^{\mathrm{T}}, (1,0,0)^{\mathrm{T}}, (1,0,1)^{\mathrm{T}}, (1,1,0)^{\mathrm{T}}\right\} \\ \omega_2 &= \left\{(0,0,1)^{\mathrm{T}}, (0,1,1)^{\mathrm{T}}, (0,1,0)^{\mathrm{T}}, (1,1,1)^{\mathrm{T}}\right\}\end{aligned}$$

Solution: 将属于 ω_2 的训练样本乘以 (-1), 并写成增广向量的形式:

$$\begin{aligned}&x_1 = (0,0,0,1)^{\mathrm{T}}, x_2 = (1,0,0,1)^{\mathrm{T}}, x_3 = (1,0,1,1)^{\mathrm{T}}, x_4 = (1,1,0,1)^{\mathrm{T}} \\ &x_5 = (0,0,-1,-1)^{\mathrm{T}}, x_6 = (0,-1,-1,-1)^{\mathrm{T}}, x_7 = (0,-1,0,-1)^{\mathrm{T}}, x_8 = (-1,-1,-1,-1)^{\mathrm{T}}\end{aligned}$$

迭代选取 $C = 1, w_1 = (0,0,0,0)^{\mathrm{T}}$, 则迭代过程中权向量变换如下:

$$\begin{aligned}w(2) &= (0,0,0,1)^{\mathrm{T}}, w(3) = (0,0,-1,0), w(4) = (0,-1,-1,-1)^{\mathrm{T}}, w(5) = (0,-1,-1,0)^{\mathrm{T}} \\ w(6) &= (1,-1,-1,1)^{\mathrm{T}}, w(7) = (1,-1,-2,0)^{\mathrm{T}}, w(8) = (1,-1,-2,1)^{\mathrm{T}}, w(9) = (2,-1,-1,2)^{\mathrm{T}} \\ w(10) &= (2,-1,-2,1)^{\mathrm{T}}, w(11) = (2,-2,-2,0)^{\mathrm{T}}, w(12) = (2,-2,-2,1)^{\mathrm{T}} \,(\text{此时已收敛})\end{aligned}$$

所以最终得到解向量 $\boldsymbol{w} = (2,-2,-2,1)^{\mathrm{T}}$, 相应的判别函数为 $d(\boldsymbol{x}) = 2x_1 - 2x_2 - 2x_3 + 1$.

2. 编写求解上述问题的感知器算法程序.

Solution: C++ 代码如下:

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <numeric>
using namespace std;

int main() {
    int d, n1, n2, C;
    cout << " 请依次输入样本维数 d, w1 类的样本数 n1, w2 类的样本数 n2, 迭代步长 C" << endl;
    cin >> d >> n1 >> n2 >> C;
    vector<vector<float>> omega_1(n1, vector<float>(d));
    vector<vector<float>> omega_2(n2, vector<float>(d));
    cout << " 请依次输入 w1 类的模式样本" << endl;
    for(int i = 0; i < n1; i++) {
        for(int j = 0; j < d; j++) {
            cout << " omega_1[" << i << "][" << j << "] : ";
            cin >> omega_1[i][j];
        }
    }
    cout << " 请依次输入 w2 类的模式样本" << endl;
    for(int i = 0; i < n2; i++) {
        for(int j = 0; j < d; j++) {
            cout << " omega_2[" << i << "][" << j << "] : ";
            cin >> omega_2[i][j];
        }
    }

```

```
    vector<vector<float>> sample_1(n1, vector<float>(d + 1));
    vector<vector<float>> sample_2(n2, vector<float>(d + 1));
    /* 增广后的 ω1 类的模式样本 */
    for(int i = 0; i < n1; i++) {
        copy(omega_1[i].begin(), omega_1[i].end(), sample_1[i].begin());
        sample_1[i][d] = 1;
    }
    /* 增广后的 ω2 类的模式样本 */
    for(int i = 0; i < n2; i++) {
        copy(omega_2[i].begin(), omega_2[i].end(), sample_2[i].begin());
        sample_2[i][d] = 1;
    }
    for(int i = 0; i < n2; i++) {
        for(int j = 0; j < d + 1; j++) {
            sample_2[i][j] = (-1.0) * sample_2[i][j];  //增广的 ω2 训练样本乘以-1
        }
    }
    int n = n1 + n2;
    vector<vector<float>> sample(n, vector<float>(d + 1));
    copy(sample_1.begin(), sample_1.end(), sample.begin());
    copy(sample_2.begin(), sample_2.end(), sample.begin() + n1);
    vector<float> w(d + 1, 0);
    int cnt = 0;
    while (cnt != n) { //当被正确分类的样本数 cnt 等于总样本数 n 时, 结束循环
        cnt = 0; //计数变量清零
        for(int i = 0; i < n; i++) {
            if(inner_product(w.begin(), w.end(), sample[i].begin(), 0.0) > 0) {
                cnt++;  //若被正确分类, 则权向量不变且对应的样本数自增一
            }
            else {
                for(int j = 0; j < d + 1; j++) {  //若 x(k) 被错误分类,
                    w[j] = w[j] + C * sample[i][j];  //则 w(k+1) = w(k) + C * x(k)
                }
            }
        }
    }
    cout << " 解向量 w 的分量分别为" << endl;
    for(int i = 0; i < d + 1; i++) {
        cout << " w[" << i << "]"  << " = " << w[i];
    }
    return 0;
}
```

程序执行结果如下图2所示:

```
Output  Build Log

请依次输入样本维数d, w1类的样本数n1, w2类的样本数n2, 迭代步长c
3 4 4 1
请依次输入w1类的模式样本
 omega_1[0][0] : 0
 omega_1[0][1] : 0
 omega_1[0][2] : 0
 omega_1[1][0] : 1
 omega_1[1][1] : 0
 omega_1[1][2] : 0
 omega_1[2][0] : 1
 omega_1[2][1] : 0
 omega_1[2][2] : 1
 omega_1[3][0] : 1
 omega_1[3][1] : 1
 omega_1[3][2] : 0
请依次输入w2类的模式样本
 omega_2[0][0] : 0
 omega_2[0][1] : 0
 omega_2[0][2] : 1
 omega_2[1][0] : 0
 omega_2[1][1] : 1
 omega_2[1][2] : 1
 omega_2[2][0] : 0
 omega_2[2][1] : 1
 omega_2[2][2] : 0
 omega_2[3][0] : 1
 omega_2[3][1] : 1
 omega_2[3][2] : 1
解向量w的分量分别为
 w[0] = 2 w[1] = -2 w[2] = -2 w[3] = 1
                                        运行结束。
```

图 2: Problem 4 程序运行结果

Problem 5

用多类感知器算法求下列模式的判别函数:

$$\omega_1 : (-1,-1)^{\mathrm{T}}, \omega_2 : (0,0)^{\mathrm{T}}, \omega_3 : (1,1)^{\mathrm{T}}$$

Solution: 采用一般化的感知器算法, 将模式样本写成增广形式, 即

$$x_1 = (-1,-1,1)^{\mathrm{T}}, x_2 = (0,0,1)^{\mathrm{T}}, x_3 = (1,1,1)^{\mathrm{T}}$$

取初始值 $w_1 = w_2 = w_3 = (0,0,0)^{\mathrm{T}}$, 取 $C = 1$.

以 x_1 为训练样本来进行第一次迭代, 计算内积得到 $d_1(1) = d_2(1) = d_3(1) = 0$, 故

$$w_1(2) = (-1,-1,1)^{\mathrm{T}}, w_2(2) = (1,1,-1)^{\mathrm{T}}, w_3(2) = (1,1,-1)^{\mathrm{T}}$$

以 x_2 为训练样本来进行第 2 次迭代, 计算内积得到 $d_1(2) = 1, d_2(2) = -1, d_3(2) = -1$, 故

$$w_1(3) = (-1,-1,0)^{\mathrm{T}}, w_2(3) = (1,1,0)^{\mathrm{T}}, w_3(3) = (1,1,-2)^{\mathrm{T}}$$

以 x_3 为训练样本来进行第 3 次迭代, 计算内积得到 $d_1(3) = -2, d_2(3) = 2, d_3(3) = 0$, 故

$$w_1(4) = (-1,-1,0)^{\mathrm{T}}, w_2(4) = (0,0,-1)^{\mathrm{T}}, w_3(4) = (2,2,-1)^{\mathrm{T}}$$

以 x_1 为训练样本来进行第 4 次迭代, 计算内积得到 $d_1(4) = 2, d_2(4) = -1, d_3(4) = -5$, 显然 x_1 被正确分类了, 故权向量此时不用更新;

以 x_2 为训练样本来进行第 5 次迭代, 计算内积得到 $d_1(5) = 0, d_2(5) = -1, d_3(5) = -1$, 故

$$w_1(6) = (-1,-1,-1)^{\mathrm{T}}, w_2(6) = (0,0,0)^{\mathrm{T}}, w_3(6) = (2,2,-2)^{\mathrm{T}}$$

以 x_3 为训练样本来进行第 6 次迭代, 计算内积得到 $d_1(6) = -3, d_2(6) = 0, d_3(6) = 2$, 显然 x_3 被正确分类了, 故权向量此时不用更新;

以 x_1 为训练样本来进行第 7 次迭代, 计算内积得到 $d_1(7) = 1, d_2(7) = 0, d_3(7) = -6$, 显然 x_1 被正确分类了, 故权向量此时不用更新;

以 x_2 为训练样本来进行第 8 次迭代, 计算内积得到 $d_1(8) = -1, d_2(8) = 0, d_3(8) = -2$, 显然 x_2 被正确分类了, 故权向量此时不用更新. 由于第 6,7,8 次迭代中对 x_1, x_2, x_3 均以正确分类, 故权向量的解为 $w_1 = (-1,-1,-1)^{\mathrm{T}}, w_2 = (0,0,0)^{\mathrm{T}}, w_3 = (2,2,-2)^{\mathrm{T}}$ 且判别函数为 $d_1(\boldsymbol{x}) = -x_1 - x_2 - 1, d_2(\boldsymbol{x}) = 0, d_3(\boldsymbol{x}) = 2x_1 + 2x_2 - 2$. 类似 Problem4, 可以编写出多类感知器算法的 C++ 程序 (见后页), 程序执行结果如下图3中所示:

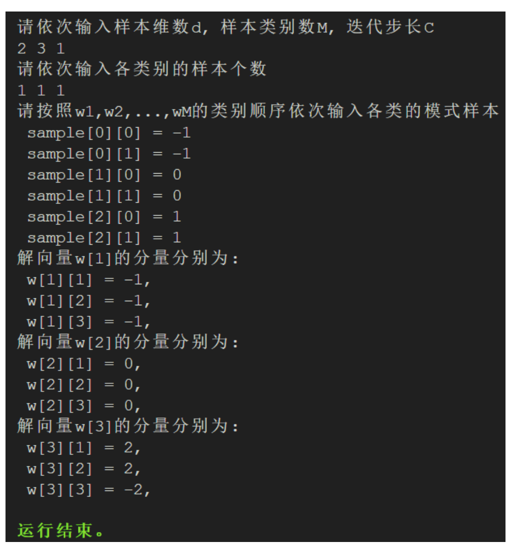

图 3: Problem 5 程序运行结果

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <numeric>
using namespace std;

int main() {
    int d, M, C;
    cout << " 请依次输入样本维数 d, 样本类别数 M, 迭代步长 C" << endl;
    cin >> d >> M >> C;
    vector<int> N(M);

    cout << " 请依次输入各类别的样本个数" << endl;
    for(int i = 0; i < M; i++) {
        cin >> N[i];
    }

    int n = accumulate(N.begin(), N.end(), 0);
    vector<vector<float>> sample_init(n, vector<float>(d));
    cout << " 请按照 w1,w2,...,wM 的类别顺序依次输入各类的模式样本" << endl;
    for(int i = 0; i < n; i++) {
        for(int j = 0; j < d; j++) {
            cout << " sample[" << i << "][" << j << "] = ";
            cin >> sample_init[i][j];
        }
    }

    vector<vector<float>> sample(n, vector<float>(d + 1));
    for(int i = 0; i < n; i++) {
        copy(sample_init[i].begin(), sample_init[i].end(), sample[i].begin());
        sample[i][d] = 1;
    }
    vector<vector<float>> w(M, vector<float>(d + 1, 0));
    vector<float> D(M);
    int cnt = 0;
    while (cnt != n) { //当被正确分类的样本数 cnt 等于总样本数 n 时, 结束循环
        cnt = 0; //每一轮迭代都需要把计数变量 cnt 清零
        for(int i = 0; i < n; i++) {
            for(int m = 0; m < M; m++) {
                D[m] = inner_product(w[m].begin(), w[m].end(), sample[i].begin(), 0.0);
            }
            int index = max_element(D.begin(), D.end()) - D.begin();
            int flag = count(D.begin(), D.end(), D[index]);
            if(i <= N[0] - 1 && i >= 0) { //第 1 类需要单独判断, 否则会越界
                if(index + 1 == 1 && flag == 1) {
                    cnt++;  //若当前样本被正确分类, 则权向量不变且对应的样本数自增一
                }
                else {  //若当前样本没被正确分类, 则按算法规则调整权向量
                    for(int m = 0; m < M; m++) {
                        if(D[m] >= D[0] && m != 0) {
                            for(int j = 0; j < d + 1; j++) {
                                w[m][j] = w[m][j] - C * sample[i][j];
                            }
                        }
```

```
                        else if(D[m] < D[0]) {
                            ;
                        }
                        else if(m == 0) {
                            for(int j = 0; j < d + 1; j++) {
                                w[m][j] = w[m][j] + C * sample[i][j];
                            }
                        }
                    }
                }
            }
            else {
                int Class = 2;
                for(int k = 2; k <= M; k++) {
                    int left = accumulate(N.begin(), N.begin() + k - 1, 0);
                    int right = accumulate(N.begin(), N.begin() + k, 0) - 1;
                    if(i >= left && i <= right) {
                        Class = k;  //求出当前样本所在的真实类别 (即第 k 类)
                        break;
                    }
                }

                if(index + 1 == Class && flag == 1) {
                    cnt++;  //若当前样本被正确分类, 则权向量不变且对应的样本数自增一
                }

                else {  //若当前样本没被正确分类, 则按规则调整权向量
                    for(int m = 0; m < M; m++) {
                        if(D[m] >= D[Class - 1] && m != Class - 1) {
                            for(int j = 0; j < d + 1; j++) {
                                w[m][j] = w[m][j] - C * sample[i][j];
                            }
                        }
                        else if(D[m] < D[Class - 1]) {
                            ;
                        }
                        else if(m == Class - 1) {
                            for(int j = 0; j < d + 1; j++) {
                                w[m][j] = w[m][j] + C * sample[i][j];
                            }
                        }
                    }
                }
            }
        }
    }
    for(int i = 0; i < M; i++) {
        cout << " 解向量 w[" << i + 1 << "] 的分量分别为: " << endl;
        for(int j = 0; j < d + 1; j++) {
            cout << " w[" << i + 1 << "][" << j + 1 << "] = " << w[i][j] << "," << endl;
        }
    }
    return 0;
}
```

Problem 6

采用梯度法和准则函数 $J\left(w,x,b\right)=\dfrac{1}{8\left\|x\right\|^{2}}\left[\left(w^{\mathrm{T}}x-b\right)-\left|w^{\mathrm{T}}x-b\right|\right]^{2}$(其中 $b>0$), 试导出两类模式的分类算法.

Solution: 上述式子对 w 求导可得

$$\frac{\partial J}{\partial w}=\frac{1}{4\left\|x\right\|^{2}}\left[\left(w^{\mathrm{T}}x-b\right)-\left|w^{\mathrm{T}}x-b\right|\right]\cdot\left[x-x\cdot\operatorname{sgn}\left(w^{\mathrm{T}}x-b\right)\right]$$

其中 sgn 表示符号函数, 即 $\operatorname{sgn}\left(w^{\mathrm{T}}x-b\right)=\begin{cases}1, w^{\mathrm{T}}x-b>0\\-1, w^{\mathrm{T}}x-b\leq 0\end{cases}$.

于是可得到迭代式

$$w\left(k+1\right)=w\left(k\right)+C\frac{\partial J}{\partial w\left(k\right)}=w\left(k\right)+\begin{cases}0, w^{\mathrm{T}}\left(k\right)x-b>0\\\frac{w^{\mathrm{T}}(k)x-b}{\|x\|^{2}}Cx, w^{\mathrm{T}}\left(k\right)x-b\leq 0\end{cases}$$

至此, Chap 3 的作业解答完毕.