合肥工業大學

机器学习工程报告

课　　程：机器学习基础

姓　　名：黄满宗

学　　号：2018217257

完成时间：2020/12/30

概要：

在 libsvm 的数据集合中，选取一个多分类数据集合，利用不同的 libsvm 核函数、和参数进行训练 libsvm，并比较实验结果。至少要验证 3 个以上的核函数，并尝试 5 组以上的参数

每人分工：独立完成

研究背景与意义

通过有监督的学习，对历史数据进行分类，得到目标函数，通过目标函数可以预测未来某时刻在某种状态下的结果

如现实生活中可以通过记录历史数据，比如某个商场的人流量，天气和日期等，通过机器学习算法预测明天商场的人流量。

模型方法

支持向量机（SVM）是一种二分类模型，它的基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使它有别于感知机；SVM 还包括核技巧，这使它成为实质上的非线性分类器。SVM 的学习策略就是间隔最大化，可形式化为一个求解凸二次规划的问题，也等价于正则化的合页损失函数的最小化问题。SVM 的学习算法就是求解凸二次规划的最优化算法。

作业使用的实现工具是 libsvm 工具包，LIBSVM 软件包是台湾大学林智仁(Chih-Jen Lin)博士等用 C++实现的 SVM 库，并且拥有 matlab,perl 等工具箱或者代码，移植和使用都比较方便.它可以解决分类问题(包括 C-SVC、n-SVC)、回归问题(包括 e-SVR、n-SVR)以及分布估计(one-class-SVM)等问题，提供了线性、多项式、径向基和 S 形函数四种常用的核函数供选择，可以有效地解决多类问题、交叉验证选择参数、对不平衡样本加权、多类问题的概率估计等

系统设计

训练数据

TICDATA2000.txt: 这个数据集用来训练和检验预测模型，并且建立了一个 5822 个客户的记录的描述。每个记录由 86 个属性组成，包含社会人口数据（属性 1-43）和产品的所有关系（属性 44-86 ）。社会人口数据是由派生邮政编码派生而来的，生活在具有相同邮政编码地区的所有客户都具有相同的社会人口属性。第 86 个属性：“大篷车：家庭移动政策” ，是我们的目标变量。

TICEVAL2000.txt: 这个数据集是需要预测（ 4000 个客户记录）的数据集。它和 TICDATA2000.txt 它具有相同的格式，只是没有最后一列的目标记录。我们只

希望返回预测目标的列表集，所有数据集都用制表符进行分隔。

TICTGTS2000.txt：最终的目标评估数据。这是一个实际情况下的目标数据，将与我们预测的结果进行校验。我们的预测结果将放在 result.txt 文件中。

本实验任务可以理解为分类问题，即分为 2 类，也就是数据源的第 86 列，可以分为 0、1 两类。我们首先需要对 TICDATA2000.txt 进行训练，生成 model，再根据 model 进行预测。

实现过程

在源程序里面，主要由以下 2 个函数来实现：

(1) struct svm_model *svm_train(const struct svm_problem *prob, const struct svm_parameter *param);

该函数用来做训练，参数 prob，是 svm_problem 类型数据，具体结构定义如下：

```
struct svm_problem //存储本次参加运算的所有样本(数据集)，及其所属类别。
{
int n; //记录样本总数
double *y; //指向样本所属类别的数组
struct svm_node **x; //指向一个存储内容为指针的数组
};
```

其中 svm_node 的结构体定义如下：

```
struct svm_node //用来存储输入空间中的单个特征
{
int index; //输入空间序号，假设输入空间数为 m
double value; //该输入空间的值
};
```

所以，prob 也可以说是问题的指针，它指向样本数据的类别和输入向量，在内存中的具体结构图如下：

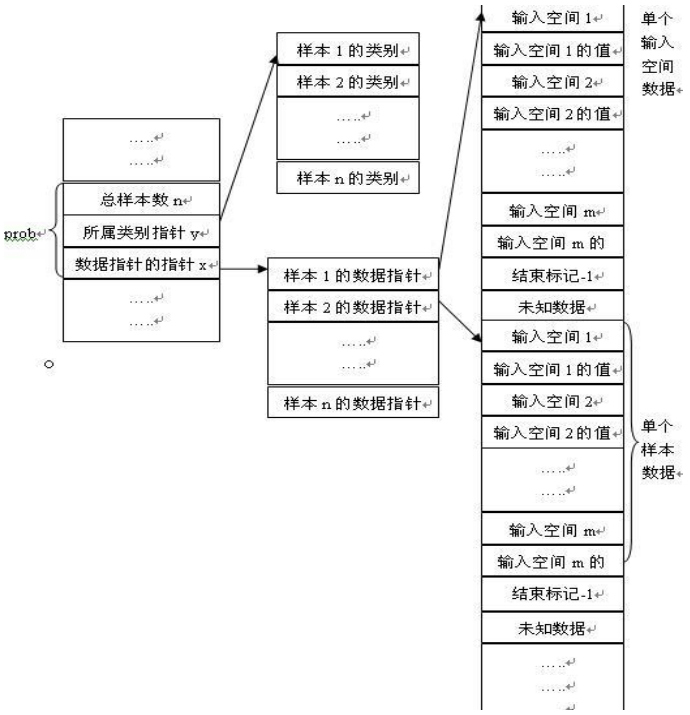

图 1.1LIBSVM 训练时，样本数据在内存中的存放结构

参数 param，是 svm_parameter 数据结构，具体结构定义如下：

```
struct svm_parameter // 训练参数
{
int svm_type; //SVM 类型，
int kernel_type; //核函数类型
int degree; /* for poly */
double gamma; /* for poly/rbf/sigmoid */
double coef0; /* for poly/sigmoid */
/* these are for training only */
double cache_size; /* in MB 制定训练所需要的内存*/
double eps; /* stopping criteria */
double C; /* for C_SVC, EPSILON_SVR and NU_SVR , 惩罚因子*/
int nr_weight; /* for C_SVC 权重的数目*/
int *weight_label; /* for C_SVC 权重，元素个数由 nr_weight 决定*/
double* weight; /* for C_SVC */
double nu; /* for NU_SVC, ONE_CLASS, and NU_SVR */
double p; /* for EPSILON_SVR */
int shrinking; /* use the shrinking heuristics 指明训练过程是否使用压缩*/
int probability; /* do probability estimates 指明是否要做概率估计*/
}
```

其中，SVM 类型和核函数类型如下：

```
enum { C_SVC, NU_SVC, ONE_CLASS, EPSILON_SVR, NU_SVR }; /* svm_type */
enum { LINEAR, POLY, RBF, SIGMOID, PRECOMPUTED }; /* kernel_type */
```

只需申请一个 svm_parameter 结构体，并按实际需要设定 SVM 类型、核函数和各种参数的值即可完成参数 param 的设置。

设定完这两个参数，就可以直接在程序中调用训练函数进行训练了，该其函数返回一个 struct svm_model *SVM 模型的指针，可以使用 svm_save_model(const char *model_file_name, const struct svm_model *model)函数，把这个模型保存在磁盘中。至此，训练函数的移植已经完成。

(2) double svm_predict(const struct svm_model *model, const struct svm_node *x);

参数 model，是一个 SVM 模型的指针，可以使用函数 struct svm_model *svm_load_model(const char *model_file_name)，导入训练时保存好的 SVM 模型，此函数返回一个 SVM 模型的指针，可以直接赋值给变量 model。

参数 x，是 const struct svm_node 结构体的指针，本意是一个输入空间的指针，但实际上，该函数执行的时候，是从参数 x 处计算输入空间，直到遇到单个样本数据结束标记-1 才结束，也就是说，该函数运算了单个样本中的所有输入空间数据。因此，在调用此函数时，必须先把预测样本的数据按图中的固定格式写入内存中。另外，该函数只能预测一个样本的，本文需要对图像中的所有像数点预测，就要使用 for 循环反复调用。

该函数返回一个 double 类型，指明被预测数据属于哪个类。面对两分类问题的时候，通常使用+1 代表正样本，即类 1；-1 代表负样本，即类 2。最后根据返回的 double 值就可以知道预测数据的类别了。

实验结果

以下是使用不同的核函数训练的结果

RBF 核函数，预测正确率：90.1574%

POLY 核函数，预测正确率：87.8341%

```
训练数据共:5822条记录
测试数据共:4003条记录
训练的时间:13599.5ms
预测的时间:262.8ms
测试正确率为:87.8341%
```

LINEAR 核函数，预测正确率：93.9795%

```
训练数据共:5822条记录
测试数据共:4003条记录
训练的时间:158572ms
预测的时间:196.3ms
测试正确率为:93.9795%
```

SIGMOID 核函数 预测正确率：89.2081%

```
训练数据共:5822条记录
测试数据共:4003条记录
训练的时间:227.6ms
预测的时间:228.7ms
测试正确率为:89.2081%
```

部分代码

Machine-learning.cpp

```
#include "SVM.h"
#include <iostream>
#include <list>
#include <iterator>
#include <vector>
#include <string>
#include <ctime>

using namespace std;

svm_parameter param;
svm_problem prob;
svm_model* svmModel;
list<svm_node*> xList;
list<double>  yList;
const int MAX = 10;
```

```
const int nTstTimes = 10;
vector<int> predictvalue;
vector<int> realvalue;
int trainNum = 0;


void setParam()
{
    param.svm_type = C_SVC;
    param.kernel_type = RBF;
    param.degree = 3;
    param.gamma = 0.5;
    param.coef0 = 0;
    param.nu = 0.5;
    param.cache_size = 40;
    param.C = 500;
    param.eps = 1e-3;
    param.p = 0.1;
    param.shrinking = 1;
    // param.probability = 0;
    param.nr_weight = 0;
    param.weight = NULL;
    param.weight_label = NULL;
}

void train(char* filePath)
{

    FILE* fp;
    int k;
    int line = 0;
    int temp;

    if ((fp = fopen(filePath, "rt")) == NULL)
        return;
    while (1)
    {
        svm_node* features = new svm_node[85 + 1];

        for (k = 0; k < 85; k++)
        {
            fscanf(fp, "%d", &temp);

            features[k].index = k + 1;
```

```
            features[k].value = temp / (MAX * 1.0);
        }

        features[85].index = -1;


        fscanf(fp, "%d", &temp);
        xList.push_back(features);
        yList.push_back(temp);

        line++;
        trainNum = line;
        if (feof(fp))
            break;
    }




    setParam();
    prob.l = line;
    prob.x = new svm_node * [prob.l];  //对应的特征向量
    prob.y = new double[prob.l];    //放的是值
    int index = 0;
    while (!xList.empty())
    {
        prob.x[index] = xList.front();
        prob.y[index] = yList.front();
        xList.pop_front();
        yList.pop_front();
        index++;
    }
    //std::cout<<prob.l<<"list end\n";
    svmModel = svm_train(&prob, &param);

    //std::cout<<"\n"<<"over\n";
    //保存model
    svm_save_model("model.txt", svmModel);

    //释放空间
    delete  prob.y;
    delete[] prob.x;
    svm_free_and_destroy_model(&svmModel);
}
```

```
void predict(char* filePath)
{
    svm_model* svmModel = svm_load_model("model.txt");

    FILE* fp;
    int line = 0;
    int temp;

    if ((fp = fopen(filePath, "rt")) == NULL)
        return;


    while (1)
    {
        svm_node* input = new svm_node[85 + 1];
        for (int k = 0; k < 85; k++)
        {
            fscanf(fp, "%d", &temp);
            input[k].index = k + 1;
            input[k].value = temp / (MAX * 1.0);
        }
        input[85].index = -1;



        int predictValue = svm_predict(svmModel, input);
        predictvalue.push_back(predictValue);

        cout << predictValue << endl;
        if (feof(fp))
            break;
    }

}

void writeValue(vector<int>& v, char* filePath)
{

    FILE* pfile = fopen(filePath, "wb");

    vector<int>::iterator iter = v.begin();
    char* c = new char[2];
    for (; iter != v.end(); ++iter)
```

```
    {

        c[1] = '\n';

        if (*iter == 0)
            c[0] = '0';
        else
            c[0] = '1';
        fwrite(c, 1, 2, pfile);
    }
    fclose(pfile);
    delete c;
}

bool getRealValue()
{
    FILE* fp;
    int temp;

    if ((fp = fopen("tictgts2000.txt", "rt")) == NULL)
        return false;
    while (1)
    {

        fscanf(fp, "%d", &temp);
        realvalue.push_back(temp);
        if (feof(fp))
            break;
    }
    return true;
}

double getAccuracy()
{
    if (!getRealValue())
        return 0.0;
    int counter = 0;
    int counter1 = 0;
    for (int i = 0; i < realvalue.size(); i++)
    {
        if (realvalue.at(i) == predictvalue.at(i))
        {
            counter++;
            if (realvalue.at(i) == 1)
```

```
                counter1++;
            }
        }

        return counter * 1.0 / realvalue.size();
}

int main()
{
        clock_t t1, t2, t3;

        cout << "请稍等待..." << endl;
        t1 = clock();
        char data1[] = "ticdata2000.txt";
        train(data1);   //训练
        t2 = clock();

        char data2[] = "ticeval2000.txt";
        predict(data2);         //预测
        t3 = clock();

        char result[] = "result.txt";
        writeValue(predictvalue, result);  //将预测值写入到文件
        double accuracy = getAccuracy();          //得到正确率
        cout << "训练数据共:" << trainNum << "条记录" << endl;
        cout << "测试数据共:" << realvalue.size() << "条记录" << endl;
        cout << "训练的时间:" << 1.0 * (t2 - t1) / nTstTimes << "ms" << endl;
        cout << "预测的时间:" << 1.0 * (t3 - t2) / nTstTimes << "ms" << endl;
        cout << "测试正确率为:" << accuracy * 100 << "%" << endl;
        return 0;
}
```

六．对本门课的感想、意见和建议

通过学习机器学习，我学习到了机器学习相关知识，感受到了机器学习的魅力，并通过大作业将所学知识巩固和应用，可以通过机器学习算法解决生活中的一些问题。尽管学的还没有特别深入，但也为日后自我学习或深入研究打下了良好基础。通过老师们的言传身教，我们也了解到了有关机器学习的一些前沿研究，了解到机器学习目前的发展方向。

感谢刘老师、杨老师和孙老师悉心教导，通过这门课，我们确实学到了很多有用的知识，

为我们以后的发展道路打下了坚实的基础。