附录 1：LIBSVM 的简单介绍

1．LIBSVM 软件包简介

LIBSVM 是台湾大学林智仁(Chih-Jen Lin)博士等开发设计的一个操作简单、易于使用、快速有效的通用 SVM 软件包，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。LIBSVM 是一个开源的软件包，需要者都可以免费的从作者的个人主页 http://www.csie.ntu.edu.tw/~cjlin/ 处获得。他不仅提供了 LIBSVM 的 C++语言的算法源代码，还提供了 Python、Java、R、MATLAB、Perl、Ruby、LabVIEW 以及 C#.net 等各种语言的接口，可以方便的在 Windows 或 UNIX 平台下使用，也便于科研工作者根据自己的需要进行改进（譬如设计使用符合自己特定问题需要的核函数等）。另外还提供了 WINDOWS 平台下的可视化操作工具 SVM-toy，并且在进行模型参数选择时可以绘制出交叉验证精度的等高线图。

2．LIBSVM 使用方法简介

LIBSVM 在给出源代码的同时还提供了 Windows 操作系统下的可执行文件，包括：进行支持向量机训练的 svmtrain.exe；根据已获得的支持向量机模型对数据集进行预测的 svmpredict.exe；以及对训练数据与测试数据进行简单缩放操作的 svmscale.exe。它们都可以直接在 DOS 环境中使用。如果下载的包中只有 C++的源代码，则也可以自己在 VC 等软件上编译生成可执行文件。

LIBSVM 使用的一般步骤是：

1） 按照 LIBSVM 软件包所要求的格式准备数据集；

2） 对数据进行简单的缩放操作；

3） 考虑选用 RBF 核函数 $K(x,y)=e^{-g\|x-y\|^2}$；

4） 采用交叉验证选择最佳参数C与g；

5） 采用最佳参数C与g对整个训练集进行训练获取支持向量机模型；

6） 利用获取的模型进行测试与预测。

一．LIBSVM 使用的数据格式

LIBSVM 使用的训练数据和测试数据文件格式如下：

<label> <index1>：<value1> <index2>：< value2> …

其中<label> 是训练数据集的目标值，对于分类，它是标识某类的整数(支持多个类)；对于回归，是任意实数。<index> 是以 1 开始的整数，表示特征的序号；<value>为实数，也就是我们常说的特征值或自变量。当特征值为 0 时，特征序号与特征值 value 都可以同时省略，即 index 可以是不连续的自然数。<label>与第一个特征序号、前一个特征值与后一个特征序号之间用空格隔开。测试数据文件中的 label 只用于计算准确度或误差，如果它是未知的，只需用任意一个数填写这一栏，也可以空着不填。例如：

+1 1:0.708 2:1 3:1 4:-0.320 5:-0.105 6:-1 8:1.21

为了使用的方便，可以编写小程序，将自己常用的数据格式按照这种数据格式要求转换成这种格式供 LIBSVM 直接使用。例如：笔者编写的在 MATLAB 中使用的格式转换函数 write4libsvm 如下：

```
function write4libsvm
% 为了使得数据满足libsvm的格式要求而进行的数据格式转换
% 原始数据保存格式为:
%              [标签 第一个属性值 第二个属性值...]
% 转换后文件格式为满足libsvm的格式要求，即:
%              [标签 1:第一个属性值 2:第二个属性值 3:第三个属性值 ...]
% JGRong@ustc
% 2004.6.16
[filename, pathname] = uigetfile( {'*.mat', ...
        '数据文件(*.mat)';  '*.*', '所有文件 (*.*)'},     '选择数据文件');
try
    S=load([pathname filename]);
    fieldName = fieldnames(S);
    str = cell2mat(fieldName);
    B = getfield(S,str);
    [m,n] = size(B);
    [filename, pathname] = uiputfile({'*.txt;*.dat' ,'数据文件
                        (*.txt;*.dat)';'*.*','所有文件 (*.*)'},'保存数据文件');
    fid = fopen([pathname filename],'w');
```

```
    if(fid~=-1)
        for k=1:m
            fprintf(fid,'%3d',B(k,1));
            for kk = 2:n
                fprintf(fid,'\t%d',(kk-1));
                fprintf(fid,':');
                fprintf(fid,'%d',B(k,kk));
            end
            fprintf(fid,'\n');
        end
        fclose(fid);
    else
        msgbox('无法保存文件!');
    end
catch
     msgbox('文件保存过程中出错!','出错了...','error');
end
```

二． svmscale 的用法

对数据集进行缩放的目的在于：1）避免一些特征值范围过大而另一些特征值范围过小；2）避免在训练时为了计算核函数而计算内积的时候引起数值计算的困难。因此，通常将数据缩放到[−1,1]或者是[0,1]之间。

用法：svmscale [-l lower] [-u upper] [-y y_lower y_upper]

[-s save_filename] [-r restore_filename] filename

（缺省值： lower = -1，upper = 1，没有对 y 进行缩放）

其中，

-l：数据下限标记；lower：缩放后数据下限；

-u：数据上限标记；upper：缩放后数据上限；

-y：是否对目标值同时进行缩放；y_lower 为下限值，y_upper 为上限值；

-s save_filename：表示将缩放的规则保存为文件 save_filename；

-r restore_filename：表示将缩放规则文件 restore_filename 载入后按此缩放；

filename：待缩放的数据文件（要求满足前面所述的格式）。

缩放规则文件可以用文本浏览器打开，看到其格式为：

lower upper

<index1> lval1 uval1

<index2> lval2 uval2

其中的 lower 与 upper 与使用时所设置的 lower 与 upper 含义相同；index 表示特征序号；lval 为该特征对应转换后下限 lower 的特征值；uval 为对应于转换后上限 upper 的特征值。

数据集的缩放结果在此情况下通过 DOS 窗口输出，当然也可以通过 DOS 的文件重定向符号“>”将结果另存为指定的文件。

使用实例：

1) svmscale –s train3.range train3>train3.scale

表示采用缺省值（即对属性值缩放到[–1,1]的范围，对目标值不进行缩放）对数据集 train3 进行缩放操作，其结果缩放规则文件保存为 train3.range，缩放集的缩放结果保存为 train3.scale。

2） svmscale –r train3.range test3>test3.scale

表示载入缩放规则 train3.range 后按照其上下限对应的特征值和上下限值线性的地对数据集 test3 进行缩放，结果保存为 test3.scale。

三. svmtrain 的用法

svmtrain 实现对训练数据集的训练，获得 SVM 模型。

用法： svmtrain [options] training_set_file [model_file]

其中，

options（操作参数）：可用的选项即表示的涵义如下所示

-s svm 类型：设置 SVM 类型，默认值为 0，可选类型有：

0 -- $C-SVC$

1 -- $n-SVC$

2 -- $one-class-SVM$

3 -- $e-SVR$

4 -- $n-SVR$

-t 核函数类型：设置核函数类型，默认值为 2，可选类型有：

0 -- 线性核：$u'*v$

1 -- 多项式核： $(g*u'*v+coef\,0)^{degree}$

2 -- RBF 核： $e^{(g\|u-v\|^2)}$

3 -- sigmoid 核： $\tanh(g*u'*v+coef\,0)$

-d degree：核函数中的 degree 设置，默认值为 3；

-g g：设置核函数中的g，默认值为$1/k$；

-r $coef0$：设置核函数中的$coef0$，默认值为 0；

-c $cost$：设置$C-SVC$、$e-SVR$、$n-SVR$中从惩罚系数C，默认值为 1；

-n n：设置$n-SVC$、$one-class-SVM$与$n-SVR$中参数n，默认值 0.5；

-p e：设置$n-SVR$的损失函数中的e，默认值为 0.1；

-m cachesize：设置 cache 内存大小，以 MB 为单位，默认值为 40；

-e e：设置终止准则中的可容忍偏差，默认值为 0.001；

-h shrinking：是否使用启发式，可选值为 0 或 1，默认值为 1；

-b 概率估计：是否计算 SVC 或 SVR 的概率估计，可选值 0 或 1，默认 0；

-wi weight：对各类样本的惩罚系数C加权，默认值为 1；

-v n：n折交叉验证模式。

其中-g 选项中的 k 是指输入数据中的属性数。操作参数 -v 随机地将数据剖分为 n 部分并计算交叉检验准确度和均方根误差。以上这些参数设置可以按照 SVM 的类型和核函数所支持的参数进行任意组合，如果设置的参数在函数或 SVM 类型中没有也不会产生影响，程序不会接受该参数；如果应有的参数设置不正确，参数将采用默认值。training_set_file 是要进行训练的数据集；model_file 是训练结束后产生的模型文件，该参数如果不设置将采用默认的文件名，也可以设置成自己惯用的文件名。

使用实例：

1）svmtrain train3.scale train3.model

训练 train3.scale，将模型保存于文件 train3.model，并在 dos 窗口中输出如下结果：

optimization finished, #iter = 1756

nu = 0.464223

obj = -551.002342, rho = -0.337784

nSV = 604, nBSV = 557

Total nSV = 604

其中，#iter 为迭代次数，nu 与前面的操作参数-n n 相同，obj 为 SVM 文件转换为的二次规划求解得到的最小值，rho 为判决函数的常数项b，nSV 为支持向量个数，nBSV 为边界上的支持向量个数，Total nSV 为支持向量总个数。

训练后的模型保存为文件 train3.model，用记事本等文本浏览器打开可以看

到其内容如下（其后“%”后内容为笔者所加注释）：

svm_type c_svc % 训练所采用的**svm**类型，此处为$C-SVC$

kernel_type rbf % 训练采用的核函数类型，此处为**RBF**核

gamma 0.047619 % 与操作参数设置中的g含义相同

nr_class 2 % 分类时的类别数，此处为两分类问题

total_sv 604 % 总共的支持向量个数

rho -0.337784 % 决策函数中的常数项b

label 0 1 % 类别标签

nr_sv 314 290 % 各类别标签对应的支持向量个数

SV % 以下为支持向量

1 1:-0.963808 2:0.906788 ... 19:-0.197706 20:-0.928853 21:-1

1 1:-0.885128 2:0.768219 ... 19:-0.452573 20:-0.980591 21:-1

... ...

1 1:-0.847359 2:0.485921 ... 19:-0.541457 20:-0.989077 21:-1

% 对于分类问题，上面的支持向量的各列含义与训练数据集相同；对于回归问题，略有不同，与训练数据中的标签 **label**（即 **y** 值）所对应的位置在模型文件的支持向量中现在存放的是 **Lagrange** 系数a 值，即为下面决策函数公式中的a 值：

$$f(x)=\sum_{i=1}^{k}(a_i-a_i^*)(\Phi(x_i)\cdot\Phi(x))+b=\sum_{i\in sv}(a_i-a_i^*)k(x_i,x)+b$$
$$=\sum_{i\in sv}a_i k(x_i,x)+b$$

四. svmpredict 的用法

svmpredict 是根据训练获得的模型，对数据集合进行预测。

用法：svmpredict [options] test_file model_file output_file

options（操作参数）：

-b probability_estimates：是否需要进行概率估计预测，可选值为 0 或者 1，默认值为 0。

model_file 是由 svmtrain 产生的模型文件；test_file 是要进行预测的数据文件；output_file 是 svmpredict 的输出文件，表示预测的结果值。svmpredict 没有其它的选项。