# 背景

使用svm\_struct 模型，svm\_struct是svm\_light的泛化。后者是最早的svm模型，它的标记只有两个，-1或者1 。而svm\_struct可以处理复杂的分类问题，例如标记可以取多个类别或者标记是树状结构等。使用svm\_struct模型需要根据实际的问题设计一个自定义接口程序，将复杂空间的特征向量映射到一个特征向量上。

# 程序总体流程

## 程序目录结构

所有程序:



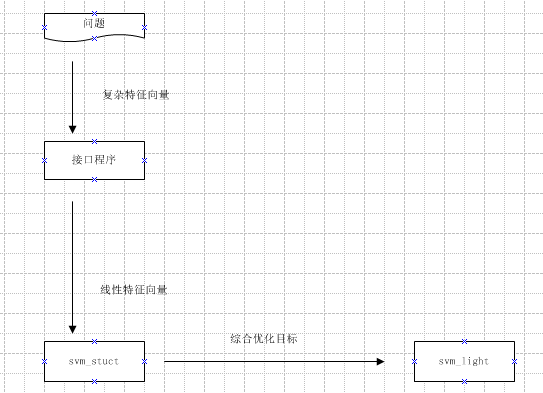
svm\_struct:



svm\_light:



## 程序模块关系图



接口程序对应程序：

svm\_struct\_api.c

svm\_struct\_api.h

svm\_struct\_api\_types.h

svm\_struct\_learn\_custom.c(一般用不到)

svm\_stuct文件夹下程序：

svm\_struct\_classify.c

svm\_struct\_common.c

svm\_struct\_common.h

svm\_struct\_learn.c

svm\_struct\_learn.h

svm\_struct\_main.c

svm\_light文件夹下程序：

svm\_classify.c

svm\_common.c

svm\_common.h

svm\_learn.c

svm\_learn.h

svm\_learn\_main.c

svm\_hideo.c

svm\_loqo.c

程序的调用次序如下：

训练的入口程序是svm\_struct/ svm\_struct\_main.c ,该程序调用svm\_struct\_api.c的 read\_struct\_examples方法读取格式化样本，再调用svm\_struct/svm\_struct\_learn.c 的

svm\_learn\_struct\_joint 方法计算出模型参数，即各个特征的权重。该方法将每行格式化样本得到的优化目标，即一个不等式，综合起来，形成一个不等式，然后调用svm\_light/svm\_learn.c的svm\_learn\_optimization方法得到各特征的权重，svm\_hideo.c和svm\_loqo.c 是对于目标函数是二次函数，约束条件是线性不等式的最优化问题的两种解法。

使用模型分类的入口程序是svm\_struct\_classify.c，该程序调用svm\_struct\_api.c的classify\_struct\_example方法对于未知样本给出预测分类。

# 针对特定问题的程序修改说明

对于一个实际问题，svm\_struct和svm\_light里的程序一般不需要修改。一般只需要修改 svm\_struct\_api.c， svm\_struct\_api.h，svm\_struct\_api\_types.h三个程序。

svm\_struct\_api\_types.h是对模型存储结构的定义和要注意的主要有三个结构体LABEL, STRUCTMODEL和STRUCT\_LEARN\_PARM。LABEL是对标记的定义，存储一行格式化样本的标记信息。STRUCTMODEL是对于svm\_struct模型的定义，主要存储特征向量，特征维度和一个二分类模型svm\_model存储实际问题转换成的二分类模型。STRUCT\_LEARN\_PARM存储STRUCTMODEL用到的一些变量。例如epsilon，C等。

以二级分类模型为例，修改如下：

//需要更改label的格式

typedef struct label {

/\* this defines the y-part (the label) of a training example,

e.g. the parse tree of the corresponding sentence. \*/

int first\_class; /\* 第一级类别 \*/

int second\_class; /\* 第二级类别 \*/

int num\_first\_classes; /\* 第一级不同类别个数 \*/

int num\_second\_classes; /\* 第二级不同类别个数\*/

//如果存储num\_first\_classes\*num\_second\_classes\*num\_third\_classes个score太占空间

//double \*scores; /\* value of linear function of each class \*/

double score;

} LABEL;

typedef struct structmodel {

double \*w; /\* pointer to the learned weights \*/

MODEL \*svm\_model; /\* the learned SVM model \*/

long sizePsi; /\* maximum number of weights in w \*/

double walpha;

/\* other information that is needed for the stuctural model can be

added here, e.g. the grammar rules for NLP parsing \*/

} STRUCTMODEL;

typedef struct struct\_learn\_parm {

double epsilon; /\* precision for which to solve

quadratic program \*/

double newconstretrain; /\* number of new constraints to

accumulate before recomputing the QP

solution \*/

int ccache\_size; /\* maximum number of constraints to

cache for each example (used in w=4

algorithm) \*/

double batch\_size; /\* size of the mini batches in percent

of training set size (used in w=4

algorithm) \*/

double C; /\* trade-off between margin and loss \*/

char custom\_argv[20][300]; /\* string set with the -u command line option \*/

int custom\_argc; /\* number of -u command line options \*/

int slack\_norm; /\* norm to use in objective function

for slack variables; 1 -> L1-norm,

2 -> L2-norm \*/

int loss\_type; /\* selected loss function from -r

command line option. Select between

slack rescaling (1) and margin

rescaling (2) \*/

int loss\_function; /\* select between different loss

functions via -l command line

option \*/

/\* further parameters that are passed to init\_struct\_model() \*/

int first\_num\_classes;

int second\_num\_classes;

int num\_features;

long top\_features;

} STRUCT\_LEARN\_PARM;

svm\_struct\_api.c里要修改一些定义的方法，下面以二级分类模型为例详细说明需要修改的各个方法：

### read\_struct\_examples

读取格式化训练样本

SAMPLE read\_struct\_examples(char \*file, STRUCT\_LEARN\_PARM \*sparm)

{

/\* Reads training examples and returns them in sample. The number of

examples must be written into sample.n \*/

read\_all\_labels();

SAMPLE sample; /\* sample \*/

EXAMPLE \*examples;

long n; /\* number of examples \*/

DOC \*\*docs; /\* examples in original SVM-light format \*/

LABEL \*target;

long totwords,i,first\_num\_classes=0,second\_num\_classes=0;

/\* Using the read\_documents function from SVM-light \*/

read\_multi\_documents(file,&docs,&target,&totwords,&n);

examples=(EXAMPLE \*)my\_malloc(sizeof(EXAMPLE)\*n);

for(i=0;i<n;i++) /\* find highest class label \*/

{

if(first\_num\_classes < (target[i].first\_class))

first\_num\_classes=target[i].first\_class;

if(second\_num\_classes < (target[i].second\_class))

second\_num\_classes=target[i].second\_class;

}

for(i=0;i<n;i++) /\* make sure all class labels are positive \*/

{

if((target[i].first\_class<1)||(target[i].second\_class<1)) {

printf("\nERROR: The class label of example number %ld is not greater than '1'!\n",i+1);

exit(1);

}

}

for(i=0;i<n;i++) { /\* copy docs over into new datastructure \*/

examples[i].x.doc=docs[i];

examples[i].y.first\_class=target[i].first\_class;

examples[i].y.second\_class=target[i].second\_class;

// examples[i].y.scores=NULL;

examples[i].y.num\_first\_classes=first\_num\_classes;

examples[i].y.num\_second\_classes=second\_num\_classes;

}

free(target);

free(docs);

sample.n=n;

sample.examples=examples;

if(struct\_verbosity>=0)

printf(" (%d examples) ",sample.n);

return(sample);

}

在svm\_multiclass中是调用svm\_light读取样本的函数read\_documents，在这里调用在svm\_struct\_api.c里重新定义的read\_multi\_documents方法，该方法调用在svm\_struct\_api.c里重新定义的parse\_multi\_document转换一行的格式化样本。

### init\_struct\_model

//设置struct mode 单词的个数，一级、二级类别的个数,和所有特征的个数

void init\_struct\_model(SAMPLE sample, STRUCTMODEL \*sm,

STRUCT\_LEARN\_PARM \*sparm, LEARN\_PARM \*lparm,

KERNEL\_PARM \*kparm)

{

/\* Initialize structmodel sm. The weight vector w does not need to be

initialized, but you need to provide the maximum size of the

feature space in sizePsi. This is the maximum number of different

weights that can be learned. Later, the weight vector w will

contain the learned weights for the model. \*/

long i,totwords=0;

WORD \*w;

sparm->first\_num\_classes=1;

sparm->second\_num\_classes=1;

for(i=0;i<sample.n;i++) /\* find highest class label \*/

{

if(sparm->first\_num\_classes < (sample.examples[i].y.first\_class))

sparm->first\_num\_classes=sample.examples[i].y.first\_class;

if(sparm->second\_num\_classes < (sample.examples[i].y.second\_class))

sparm->second\_num\_classes=sample.examples[i].y.second\_class;

}

for(i=0;i<sample.n;i++) /\* find highest feature number \*/

for(w=sample.examples[i].x.doc->fvec->words;w->wnum;w++)

if(totwords < w->wnum)

totwords=w->wnum;

sparm->num\_features=totwords;

if(struct\_verbosity>=0)

printf("Training set properties: %d features, %d first\_classes %d second\_classes \n",

sparm->num\_features,sparm->first\_num\_classes,sparm->second\_num\_classes);

sm->sizePsi=0;

sm->sizePsi+=sparm->num\_features\*sparm->first\_num\_classes;

sm->sizePsi+=sparm->num\_features\*sparm->second\_num\_classes;

if(struct\_verbosity>=2)

printf("Size of Phi: %ld\n",sm->sizePsi);

}

sm->sizePsi是复杂特征空间转换后的线性特征向量的维度。在二级分类里是将两级特征映射到特征向量的不通位置。第一级特征的数量是 num\_features\*first\_num\_classes，即所有训练样本中不同单词的个数和不同一级类别个数的乘积。第二级特征的数量是num\_features\* second\_num\_classes，即所有训练样本中不同单词的个数和不同二级类别个数的乘积。

### find\_most\_violated\_constraint\_marginrescaling

find\_most\_violated\_constraint\_slackrescaling和find\_most\_violated\_constraint\_marginrescaling完成类似的工作，svm\_struct\_main.c默认调用

svm\_learn\_struct\_joint(sample,&struct\_parm,&learn\_parm,&kernel\_parm,&structmodel,ONESLACK\_DUAL\_CACHE\_ALG)方法，该方法调用find\_most\_violated\_constraint\_marginrescaling方法。

LABEL find\_most\_violated\_constraint\_marginrescaling(PATTERN x, LABEL y,

STRUCTMODEL \*sm,

STRUCT\_LEARN\_PARM \*sparm)

{

/\* Finds the label ybar for pattern x that that is responsible for

the most violated constraint for the margin rescaling

formulation. It has to take into account the scoring function in

sm, especially the weights sm.w, as well as the loss

function. The weights in sm.w correspond to the features defined

by psi() and range from index 1 to index sm->sizePsi. Most simple

is the case of the zero/one loss function. For the zero/one loss,

this function should return the highest scoring label ybar, if

ybar is unequal y; if it is equal to the correct label y, then

the function shall return the second highest scoring label. If

the function cannot find a label, it shall return an empty label

as recognized by the function empty\_label(y). \*/

LABEL ybar;

DOC doc;

int first\_class,best\_first\_class=-1,first=1;

int second\_class,best\_second\_class=-1;

double score,bestscore=-1;

/\* NOTE: This function could be made much more efficient by not

always computing a new PSI vector. \*/

doc=\*(x.doc);

// ybar.scores=NULL;

ybar.num\_first\_classes=sparm->first\_num\_classes;

ybar.num\_second\_classes=sparm->second\_num\_classes;

char ll\_ss[512];

int temp\_c1;

int temp\_c2;

int li=0;

for(li=0;li<mla\_index;li++)

{

strcpy(ll\_ss,multi\_label\_arr[li]);

sscanf(ll\_ss,"%d\_%d",&temp\_c1,&temp\_c2);

ybar.first\_class=temp\_c1;

ybar.second\_class=temp\_c2;

doc.fvec=psi(x,ybar,sm,sparm);

score=classify\_example(sm->svm\_model,&doc);

free\_svector(doc.fvec);

score+=loss(y,ybar,sparm);

if((bestscore<score) || (first)) {

bestscore=score;

best\_first\_class=temp\_c1;

best\_second\_class=temp\_c2;

first=0;

}

}

if((best\_first\_class == -1)||(best\_second\_class==-1))

printf("ERROR: Only one class\n");

ybar.first\_class=best\_first\_class;

ybar.second\_class=best\_second\_class;

if(struct\_verbosity>=3)

printf(" [%d %d:%.2f] ",best\_first\_class,best\_second\_class,bestscore);

return(ybar);

}

该方法是svm\_learn\_struct\_joint用来将各行样本对应的不等式综合成一个不等式。该方法对于每行样本使用当前的模型参数计算出与正确标记y最不符合，即loss最大，的标记ybar。使用该ybar构造每行样本对应的不等式，即

f(y)\*w-f(ybar)\*w>= loss(y,ybar)- epsilon， (1)

f(y)是该行样本标记为y时对应的特征向量，f(ybar) 是该行样本标记为ybar时对应的特征向量, epsilon是该行样本对应的误差，w是总的特征向量的权重。svm\_learn\_struct\_joint中对不同行样本对应不等式的左边和右边进行加和，形成一个总的不等式。然后调用svm\_optimization得出优化后的w向量。

multi\_label\_arr存储形式为 一级类别\_二级类别 的字符串。它的大小是样本中所有不同的该形式的字符串的个数。

### psi

//返回样本(x,y)对应的特征

/\*特征格式：假设单词是x={(w1,f1),(w2,f2),...(wn,fn)},其中wi 是单词词典索引，fi是对应的频率

\* y={c1,c2},c1是第一级类别的索引(从1开始，连续)，c2是第二级类别的索引

\* 那么返回的特征是:

\* 第一级类别特征: {(w1+(c1-1)|V|,f1),(w2+(c1-1)|V|,f2),...(wn+(c1-1)|V|,fn)}

\* 第二级类别特征:

{(w1+|C\_1||V|+(c2-1)|V|,f1),(w2+|C\_1||V|+(c2-1)|V|,f2),...(wn+|C\_1||V|+(c2-1)|V|,fn)}

\*/

SVECTOR \*psi(PATTERN x, LABEL y, STRUCTMODEL \*sm,

STRUCT\_LEARN\_PARM \*sparm)

{

/\* Returns a feature vector describing the match between pattern x and

label y. The feature vector is returned as an SVECTOR

(i.e. pairs <featurenumber:featurevalue>), where the last pair has

featurenumber 0 as a terminator. Featurenumbers start with 1 and end with

sizePsi. This feature vector determines the linear evaluation

function that is used to score labels. There will be one weight in

sm.w for each feature. Note that psi has to match

find\_most\_violated\_constraint\_???(x, y, sm) and vice versa. In

particular, find\_most\_violated\_constraint\_???(x, y, sm) finds that

ybar!=y that maximizes psi(x,ybar,sm)\*sm.w (where \* is the inner

vector product) and the appropriate function of the loss. \*/

SVECTOR \*fvec;

register WORD \*sum,\*sumi;

register WORD \*ai;

long veclength;

long wveclength;

long c1,c2;

long C\_1=25;

long C\_2=136;

c1=y.first\_class;

c2=y.second\_class;

ai=x.doc->fvec->words;

veclength=0;

while(ai->wnum){

veclength++;

ai++;

}

veclength++;

wveclength=veclength\*(C\_1+C\_2);

wveclength++;

sum=(WORD \*)my\_malloc(sizeof(WORD)\*wveclength);

sumi=sum;

//第一级类别特征

ai=x.doc->fvec->words;

while(ai->wnum){

(\*sumi)=(\*ai);

sumi->wnum+=(c1-1)\*sparm->num\_features;

ai++;

sumi++;

}

//第二级类别特征

ai=x.doc->fvec->words;

while(ai->wnum){

(\*sumi)=(\*ai);

sumi->wnum=sumi->wnum+C\_1\*sparm->num\_features+(c2-1)\*sparm->num\_features;

ai++;

sumi++;

}

sumi->wnum=0;

char \*userdefined=NULL;

if(x.doc->fvec->userdefined)

{

userdefined=(char \*)my\_malloc(sizeof(char)\*(strlen(x.doc->fvec->userdefined)+1));

strcpy(userdefined,x.doc->fvec->userdefined);

}

fvec=create\_svector\_shallow(sum,userdefined,x.doc->fvec->factor);

/\* The following makes sure that the weight vectors for each class

are treated separately when kernels are used . \*/

fvec->kernel\_id=(c1-1)\*C\_2+c2;

return(fvec);

}

该方法是针对特定问题定义svm\_struct模型的核心函数，它的输入是样本为x,标记为y的样本，输出是转换后的一维特征向量。可以看到该函数是将一二级别的特征映射到特征向量fvec的不同位置。

### loss

double loss(LABEL y, LABEL ybar, STRUCT\_LEARN\_PARM \*sparm)

{

/\* loss for correct label y and predicted label ybar. The loss for

y==ybar has to be zero. sparm->loss\_function is set with the -l option. \*/

if(sparm->loss\_function == 0) { /\* type 0 loss: 0/1 loss \*/

if((y.first\_class==ybar.first\_class)&&(y.second\_class==ybar.second\_class))

{

return(0);

}

else

{

return(100);

}

}

if(sparm->loss\_function == 1) { /\* type 1 loss: squared difference \*/

if((y.first\_class==ybar.first\_class)&&(y.second\_class==ybar.second\_class))

{

return(0);

}

else

{

return(100);

}

}

else {

/\* Put your code for different loss functions here. But then

find\_most\_violated\_constraint\_???(x, y, sm) has to return the

highest scoring label with the largest loss. \*/

printf("Unkown loss function\n");

exit(1);

}

}

该函数定义公式(1)的loss(y,ybar)函数。

### classify\_struct\_example

//训练出权重w后，对未标记的样本进行类别预测,训练时不需要

LABEL classify\_struct\_example(PATTERN x, STRUCTMODEL \*sm,

STRUCT\_LEARN\_PARM \*sparm)

{

/\* Finds the label yhat for pattern x that scores the highest

according to the linear evaluation function in sm, especially the

weights sm.w. The returned label is taken as the prediction of sm

for the pattern x. The weights correspond to the features defined

by psi() and range from index 1 to index sm->sizePsi. If the

function cannot find a label, it shall return an empty label as

recognized by the function empty\_label(y). \*/

LABEL y;

DOC doc;

int first\_class,best\_first\_class=-1,first=1,j;

int second\_class,best\_second\_class=-1;

double score,bestscore=-1;

WORD \*words;

doc=\*(x.doc);

y.num\_first\_classes=sparm->first\_num\_classes;

y.num\_second\_classes=sparm->second\_num\_classes;

words=doc.fvec->words;

char ll\_ss[512];

int temp\_c1;

int temp\_c2;

int li=0;

long temp\_top=sparm->top\_features;

for(li=0;li<mla\_index;li++)

{

strcpy(ll\_ss,multi\_label\_arr[li]);

sscanf(ll\_ss,"%d\_%d",&temp\_c1,&temp\_c2);

printf("temp\_c1=%d temp\_c2=%d \n",temp\_c1,temp\_c2);

y.first\_class=temp\_c1;

y.second\_class=temp\_c2;

doc.fvec=psi(x,y,sm,sparm);

for(j=0;(doc.fvec->words[j]).wnum != 0;j++) { /\* Check if feature numbers \*/

if((doc.fvec->words[j]).wnum>(temp\_top-1)) /\* are not larger than in \*/

{

(doc.fvec->words[j]).wnum=0; /\* model. Remove feature if \*/

(doc.fvec->words[j]).weight=0;

}

}

score=classify\_example(sm->svm\_model,&doc);

free\_svector(doc.fvec);

if((bestscore<score) || (first)) {

bestscore=score;

best\_first\_class=temp\_c1;

best\_second\_class=temp\_c2;

first=0;

}

}

y.first\_class=best\_first\_class;

y.second\_class=best\_second\_class;

y.score=bestscore;

return(y);

}

该方法在训练模型时不使用，在对一个未知分类的样本预测分类时使用，它被svm\_struct/ svm\_struct\_classify.c调用。作用是使用训练好的参数找出得分最高的标记y。

其它需要该的方法还有write\_struct\_model和read\_struct\_model方法，这两个方法根据svm\_struct\_api\_types.h里的 STRUCT\_LEARN\_PARM改动而稍微做些修改，以将

STRUCT\_LEARN\_PARM完整地写入和读出。

# 分类模型的训练和使用

执行make编译后后生成classify和learn两个可执行文件，后者用于训练模型，前者使用训练好的模型对未知样本进行预测分类。

使用方法是：

./svm\_multiclass\_learn -c 5000 example4/sim\_train.txt example4/model

./svm\_multiclass\_classify example4/sim\_test.txt example4/model example4/predictions

example4/sim\_train.txt和example4/sim\_test.txt分别是训练样本和未分类样本，格式是

line .=. <first\_class> <second\_class> <wordindex:num>\*