Homework 3

5130309194 Shen Jiyuan

---

1 Implement traditional one-versus-one , one-versus-rest task and part-versus-part task decomposition methods to solve a multi-class problem mentioned below. Required: Use two different kernel functions, namely linear and RBF, in all classifiers.

Multi-class Problem: The dataset (train.txt, test.txt) contains protein sequences from 12 subcellular locations.20 dimensions stand for 20 amino acid composition of the sequences.

No. of proteins: 7579 (6065 training samples, 1514 test samples).

No. of classes: 12 (0~11).

The data file format:

label	*dim1 : value*	*dim2 : value*	*...*	*dim20 : value*
0	*1 : 0.095861*	*2 : 0.010893*	*...*	*20 : 0.032680*

Solution:

In this experiment, Libsvm is introduced as a package tool to implement task decomposition methods for multi-class problem.

● matlab-based input processing

We should first mine all data and labels from files (train.txt, test.txt). Here, limited by the special format of the data files, we will use fopen and fscanf to get data and initialize our matrix. Take the file ‘train.txt’ for instance:

```
fid=fopen('train.txt','r');
[f,count]=fscanf(fid,'%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f
 %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f %f:%f',[41,6065]);
fclose(fid);
f=f';
data_tr=f(:,[3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41]);
label_tr=f(:,1);
```

● one-versus-one

Definition: For k-class problems, (k-1)k/2 classifiers are needed to classify all test samples. And finally we will choose the mode as its class for each test sample.

Since Libsvm itself is coded in the method one-versus-one, it is convenient for us to directly call related functions. Here, two functions are called: svmtrain and svmpredict.

```
%train one-versus-one linear
model=svmtrain(label_tr,data_tr,'-c 2048 -g 0.2 -t 0 -b 1');
%predict for test set
[ptest,acctest,probtest]=svmpredict(label_te,data_te,model,'-b 1');
```

● one-versus-rest

Definition: For k-class problems, (k-1)k/2 classifiers are needed to classify all test samples. Since we have exactly 12 classes, we should establish 12 classifiers. Classifier i is used to determine whether test sample belongs to class i.

```
%train one-versus-rest linear
model=cell(num_label,1);
for k=1:num_label
   model{k}=svmtrain(double(label_tr==k-1),data_tr,'-c 2048 -g 0.2 -t 0 -b 1');
end
%get probability estimates of test instances using each model
prob=zeros(num_te,num_label);
for k=1:num_label
   [~,~,p]=svmpredict(double(label_te==k-1),data_te,model{k},'-b 1');
   prob(:,k)=p(:,model{k}.Label==1); %probability of class==k
end
%predict test instances with the highest probability
prob=prob';
[~,pred]=max(prob); %pred is the label prediction of each test
pred=pred';
pred=pred-1;
acctest=sum(pred==label_te) ./ numel(label_te);
C=confusionmat(label_te,pred);
```

Notice here cell and zeros are functions called for pre-allocation.

● part-versus-part

Definition: For k-class problems, divide all samples into small subsets, and take one-versus-one decomposition method on each subsets. Then apply Min-Max Algorithm on these predictions. Finally, as same as one-versus-one method, it chooses all the mode.

In my implementation of part-versus-part method, a large amount of logic sentences are used to make a good play on matrix. And I extract all data and label for one same label to new matrix. Then apply a (k-1)k/2 runs for each Min-Max Algorithm on one part, where we select M= ceil(min(xn,yn)/2). This means that we will have exactly (xn/M+1)*(yn/M+1) runs for trainings.

```
for k1=0:m:xn
          for k2=0:m:yn
             ll=k1+m;lr=k2+m;
             if ll>xn
                ll=xn;
             end
             if lr>yn
                lr=yn;
             end
             data=[x(k1:1:ll);y(k2:1:lr)];label=[lx(k1:1:ll);ly(k2:1:lr)];
             model=svmtrain(label,data,'-c 2048 -g 0.2 -t 0 -b 1');
             [p,~,probtest]=svmpredict(label_te,data_te,model,'-b 1');
             p=p';
```

```
        if k==0
            ptmp=p;k=k+1;
        end
        if k==1
           ptmp=ptmp|p;k=0;  (here extend for multi-class)
           if s==0
               pulti=ptmp;s=s+1;
           else
               pulti=pulti&ptmp;  (here extend for multi-class)
           end
        end
    end
end
```

Principles for Min-Max Algorithm implementations have been discussed last time(each time produce an useful matrix information, then apply a min or a max.)

- Linear and RBF (kernel function)

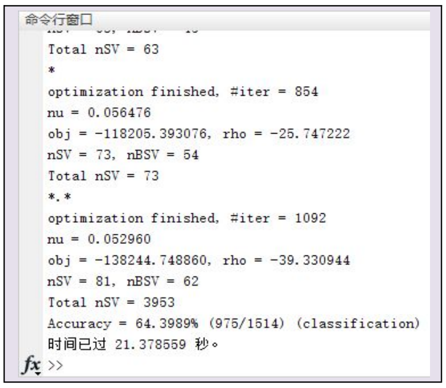
  Since there is a parameter in svmtrain that allow us to select between kernel functions: -t. Set ‘-t 0’ for linear kernel function, and ‘-t 2’ for RBF kernel function.

---

2 Compare the advantages and disadvantages of these three tack decomposition methods.

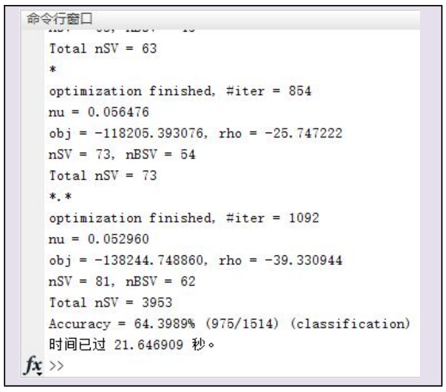
Solution:

- Running one-versus-one: (linear)



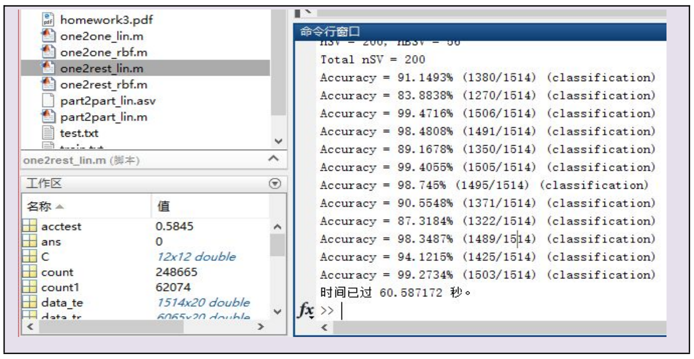
```
命令行窗口
Total nSV = 63
*
optimization finished, #iter = 854
nu = 0.056476
obj = -118205.393076, rho = -25.747222
nSV = 73, nBSV = 54
Total nSV = 73
*.*
optimization finished, #iter = 1092
nu = 0.052960
obj = -138244.748860, rho = -39.330944
nSV = 81, nBSV = 62
Total nSV = 3953
Accuracy = 64.3989% (975/1514) (classification)
时间已过 21.378559 秒。
fx >>
```

- Running one-versus-one: (rbf)



```
命令行窗口
Total nSV = 63
*
optimization finished, #iter = 854
nu = 0.056476
obj = -118205.393076, rho = -25.747222
nSV = 73, nBSV = 54
Total nSV = 73
*.*
optimization finished, #iter = 1092
nu = 0.052960
obj = -138244.748860, rho = -39.330944
nSV = 81, nBSV = 62
Total nSV = 3953
Accuracy = 64.3989% (975/1514) (classification)
时间已过 21.646909 秒。
fx >>
```

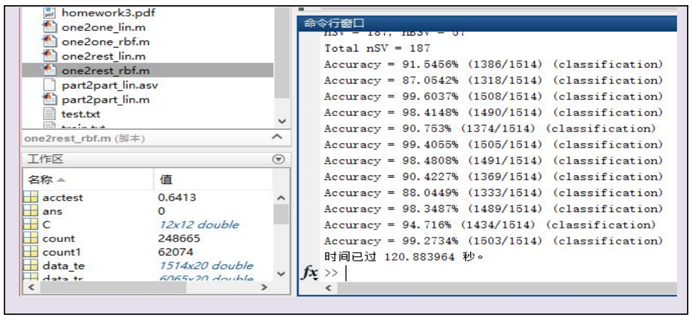
- Running one-versus-rest: (linear)



名称	值
acctest	0.5845
ans	0
C	12x12 double
count	248665
count1	62074
data_te	1514x20 double

```
命令行窗口
Total nSV = 200
Accuracy = 91.1493% (1380/1514) (classification)
Accuracy = 83.8838% (1270/1514) (classification)
Accuracy = 99.4716% (1506/1514) (classification)
Accuracy = 98.4808% (1491/1514) (classification)
Accuracy = 89.1678% (1350/1514) (classification)
Accuracy = 99.4055% (1505/1514) (classification)
Accuracy = 98.745% (1495/1514) (classification)
Accuracy = 90.5548% (1371/1514) (classification)
Accuracy = 87.3184% (1322/1514) (classification)
Accuracy = 98.3487% (1489/1514) (classification)
Accuracy = 94.1215% (1425/1514) (classification)
Accuracy = 99.2734% (1503/1514) (classification)
时间已过 60.587172 秒。
fx >>
```

● Running one-versus-rest: (rbf)



homework3.pdf
one2one_lin.m
one2one_rbf.m
one2rest_lin.m
one2rest_rbf.m
part2part_lin.asv
part2part_lin.m
test.txt

one2rest_rbf.m (脚本)

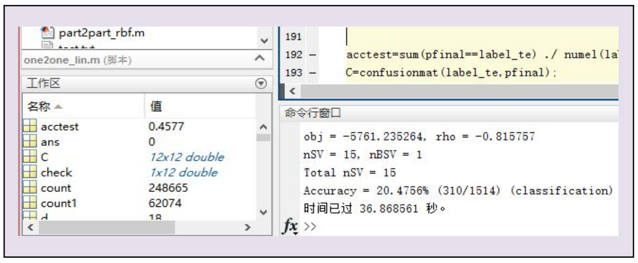
工作区

名称	值
acctest	0.6413
ans	0
C	12x12 double
count	248665
count1	62074
data_te	1514x20 double

命令行窗口

```
Total nSV = 187
Accuracy = 91.5456% (1386/1514) (classification)
Accuracy = 87.0542% (1318/1514) (classification)
Accuracy = 99.6037% (1508/1514) (classification)
Accuracy = 98.4148% (1490/1514) (classification)
Accuracy = 90.753% (1374/1514) (classification)
Accuracy = 99.4055% (1505/1514) (classification)
Accuracy = 98.4808% (1491/1514) (classification)
Accuracy = 90.4227% (1369/1514) (classification)
Accuracy = 88.0449% (1333/1514) (classification)
Accuracy = 98.3487% (1489/1514) (classification)
Accuracy = 94.716% (1434/1514) (classification)
Accuracy = 99.2734% (1503/1514) (classification)
时间已过 120.883964 秒。
>>
```

● Running part-versus-part: (linear)



part2part_rbf.m

one2one_lin.m (脚本)

工作区

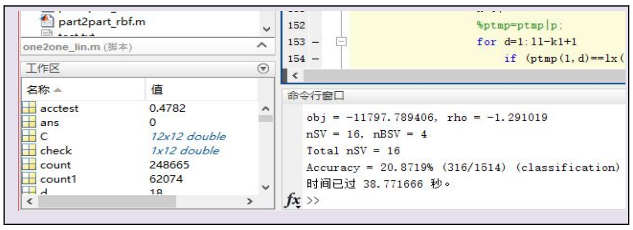
名称	值
acctest	0.4577
ans	0
C	12x12 double
check	1x12 double
count	248665
count1	62074

```
acctest=sum(pfinal==label_te) ./ numel(la
C=confusionmat(label_te,pfinal);
```

命令行窗口

```
obj = -5761.235264, rho = -0.815757
nSV = 15, nBSV = 1
Total nSV = 15
Accuracy = 20.4756% (310/1514) (classification)
时间已过 36.868561 秒。
>>
```

● Running part-versus-part: (rbf)



part2part_rbf.m

one2one_lin.m (脚本)

工作区

名称	值
acctest	0.4782
ans	0
C	12x12 double
check	1x12 double
count	248665
count1	62074

```
            %ptmp=ptmp|p;
            for d=1:11-k1+1
                if (ptmp(1,d)==1x(
```

命令行窗口

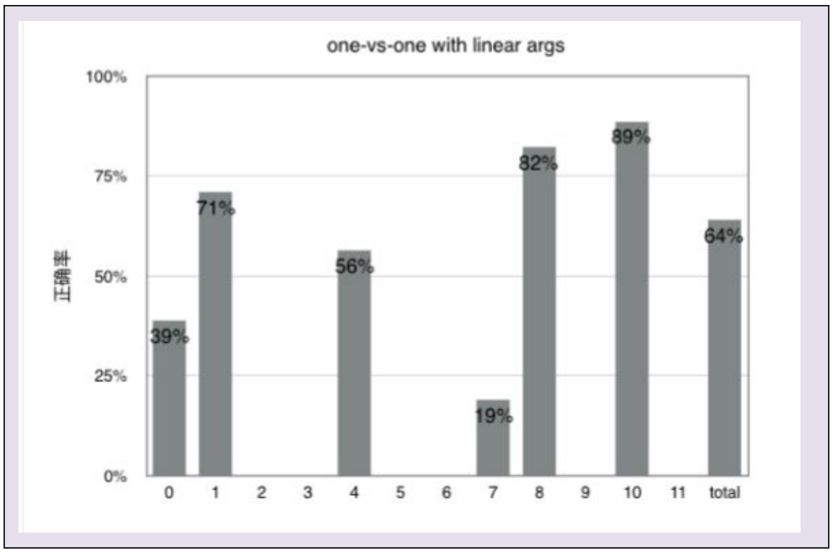
```
obj = -11797.789406, rho = -1.291019
nSV = 16, nBSV = 4
Total nSV = 16
Accuracy = 20.8719% (316/1514) (classification)
时间已过 38.771666 秒。
>>
```

● Analysis over Results

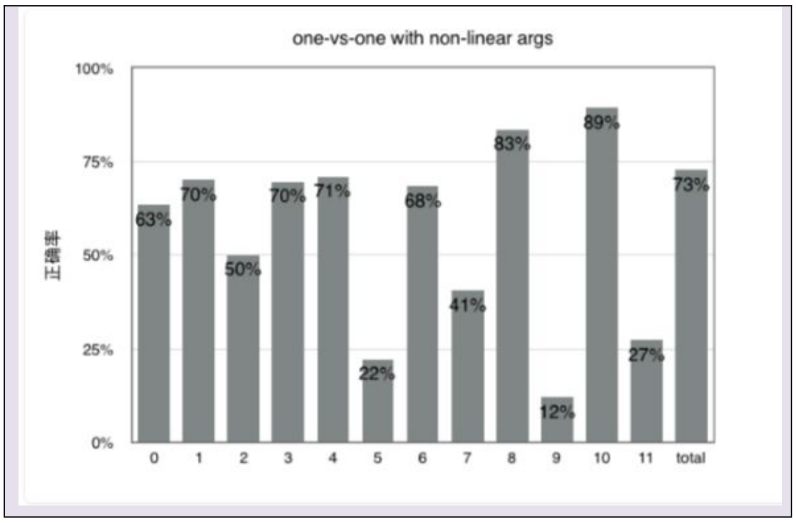
By using Confusionmat(label_test, label_predict) function, we can have all accuracy values.

And we can figure them as follows:

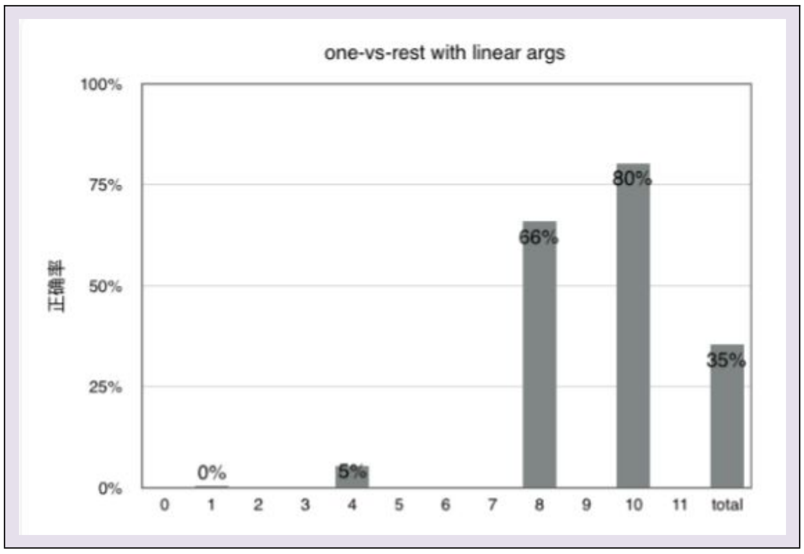
→ Running one-versus-one: (linear)



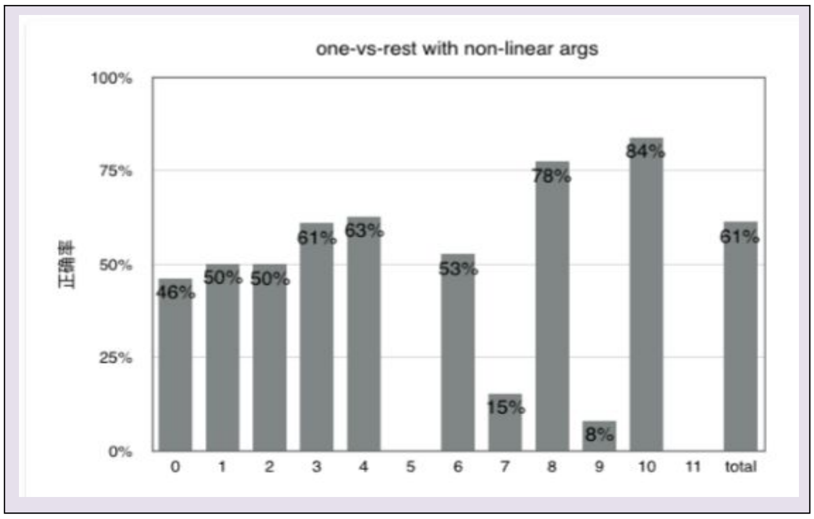
→ Running one-versus-one: (rbf)



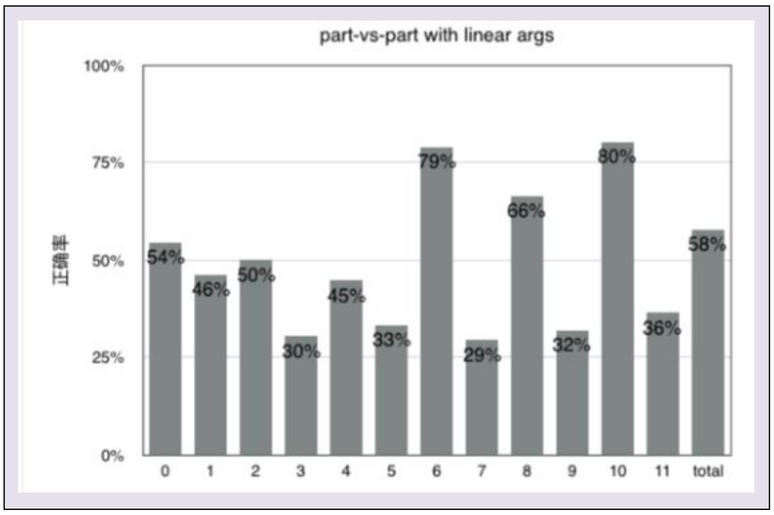
→ Running one-versus-rest: (linear)



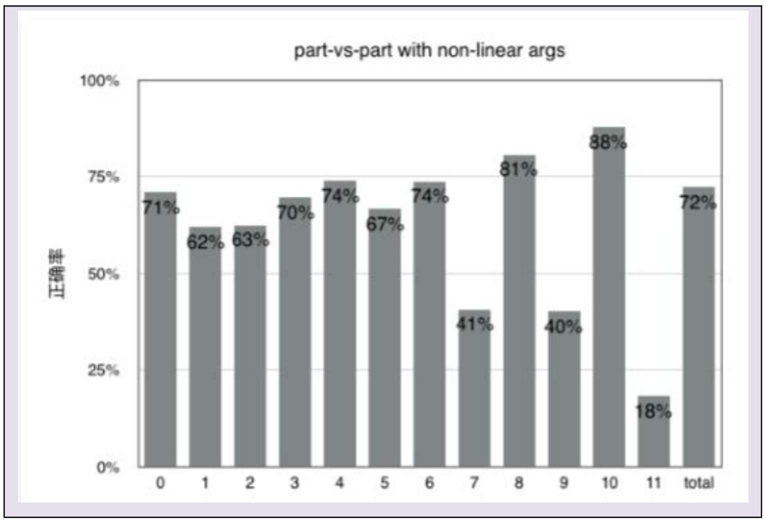
→Running one-versus-rest: (rbf)



→ Running part-versus-part: (linear)



→ Running part-versus-part: (rbf)



<u>Time Analysis:</u> In theory, since decision plain number has largely reduced, one-verses-rest decomposition method can take a faster predicting speed compared to one-verses-one decomposition method. However, considering each determination one-verses-rest method will use all of the samples and it is implemented in libsvm where one-verses-one coded as a ground, we can observe directly from the results that training time of one-verses-rest can be larger than one-verses-one.

<u>Accuracy Analysis</u>: One-versus-rest each time takes a two classes decision for all training Samples, then it will lead to significant imbalance and we can see from the results that it apparently affects the prediction accuracy. While one-versus-one can also exist such a imbalance, which inspired the part-versus-part decomposition method. Part-versus-part makes use of Min-Max to undermine such class-sample imbalance. And we can observe from the results that part-versus-part actually works for multi-classes problems.

● Summary

► Comparing RBF with Linear & Considering one-versus-one

The comparison can be placed with results in one-versus-one decomposition method. And here RBF as a nonlinear kernel function can exactly have a better model after training than linear.

Especially when the data volume is small, linear one will directly distract them. So we know
In one-versus-one, non-linear one is more satisfying.

► Considering one-versus-rest

The one-versus-rest method is not satisfying!No matter in a linear or a RBF kernel function, the accuracy is smaller than the other two method.

► Considering part-versus-part

The part-versus-part may exhibit no obvious advantage compared to original one-versus-one if we consider the accuracy. However, such a mesh may perform really well when the data scale is smaller since it can reserve most points.

► Considering comparisons

	Advantage	Disadvantage
One-versus-one	Good accuracy; Good speed; Good performance	Hard to deal with small-volume data
One-versus-rest	Good speed	Bad accuracy
Part-versus-part	Good performance; Deal with small-volume data	Bad speed