# 1 SVM module

## 1.1 Brief introduction to LRM and SVM

Firstly, logistic regression and SVM with kernel function linear. are totally different. Statistic inferior can be made and further topics of asymptotic statistics can be discussed in the framework of logistic regression model. But we can not do this in a SVM model.
Second of all, logistic regression require the variable obeys such rules below:

1. $X|_{Y=1} \sim N(\mu_1, \Sigma), X|_{Y=0} \sim N(\mu_0, \Sigma)$
2. $\mu_1 \neq \mu_2$, but $X|_{Y=1}, X|_{Y=0}$ share the same correlation matrix

Although we could prove, the $X_1|_{Y=1} \sim N(.,.)$with 95% confidence, other variables may not be so. Use some techniques, we can transform other variables normal, such as Box-Cox transformations(Box and Cox,1964 ):

$$y_i^{(\lambda)} = \begin{cases} \frac{(y_i+\lambda_2)^{\lambda_1}-1}{\lambda}, & \text{if } \lambda_1 \neq 0 \\ \log(y_i+\lambda_2), & \text{if } \lambda_1 = 0 \end{cases}$$

where

$$\lambda_2 + y_i > 0 \forall i$$

Here may be some potential problems. One is that the condition probabilities can not determine the joint distributions, even when they are normal. The second is that this method is not natural at all and may cause a transform abuse. So in data clean step, we only do a simple scale.

If the training data set in this exercise is highly imbalanced, or rare risk as someone calls. There are lots of papers handling such problem (King and Zeng, 2001 and Owen 2007).
There are many interesting results. In King and Zeng's work, for example, they demonstrates that the MLE of coefficients of variables $\hat{\beta}$ need not be changed, but the constant term should be adjusted by subtracting out the bias factor $\ln \frac{\frac{1-\tau}{\tau}}{\frac{\bar{y}}{1-\bar{y}}}$, where $\tau = Pr(Y=1), \bar{y} = Pr(y=1) = mean(y)$.
Actually, they give the general results for more generalized linear regression and others.

This is the kernel we use in this study. We use a test-and-try strategy to choose the best one.

Table 1: Different SVM kernel function

| Kernel function | Formular |
|---|---|
| Linear | $u^T v$ |
| Polynomial | $(\gamma u^T v + coef_0)^{degree}$ |
| radial basis | $exp^(-\gamma\|u-v\|^2)$ |
| sigmoid: | $tanh(\gamma u^T v + coef_0)$ |

## 1.2 Data Clean

### 1.2.1 Categorical Data

Before use e1071/libsvm, the data needed to be scaled to the range near $[-1, 1]$. Class/Factor variables should be transformed into 0-1 matrix instead of numeric vector. For example, a feature $(a, b, a, c, b)^T$ should be transformed into

$$\begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}$$

One must pay attenion that, the rank of such matrix should be number of levels$-$1. Otherwise, iteraction terms should not be included. Foexample

```
> library(biostacs)
> mat<-data.frame(X=1:10,Y=factor(sample(1:5,10,replace=T)),
+                                 Z=sample(letters[1:3],10,replace=T))
> a=model.matrix(~mat$Y);
> b=model.matrix(~mat$Y-1);
> cat('a is WRONG!')

a is WRONG!

> a

   (Intercept) mat$Y3 mat$Y4 mat$Y5
1            1      1      0      0
2            1      0      0      1
3            1      1      0      0
4            1      0      0      0
5            1      0      1      0
6            1      0      1      0
7            1      0      0      1
8            1      1      0      0
9            1      0      0      1
10           1      0      0      1
attr(,"assign")
[1] 0 1 1 1
attr(,"contrasts")
attr(,"contrasts")$`mat$Y`
[1] "contr.treatment"

> cat('b is WRONG!');

b is WRONG!

> b

   mat$Y2 mat$Y3 mat$Y4 mat$Y5
1       0      1      0      0
2       0      0      0      1
```

```
3       0      1      0      0
4       1      0      0      0
5       0      0      1      0
6       0      0      1      0
7       0      0      0      1
8       0      1      0      0
9       0      0      0      1
10      0      0      0      1
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$`mat$Y`
[1] "contr.treatment"
```

```
> cat('This is the RIGHT one:')
```

```
This is the RIGHT one:
```

```
> a[,-1]
```

```
   mat$Y3 mat$Y4 mat$Y5
1       1      0      0
2       0      0      1
3       1      0      0
4       0      0      0
5       0      1      0
6       0      1      0
7       0      0      1
8       1      0      0
9       0      0      1
10      0      0      1
```

```
> example<-svm.prehandle(mat,att.biostacs=T)
```

```
2 th feature may cause error!!!Transform Factor into numeric!!! Using expand method here.
3 th feature may cause error!!!Transform Factor into numeric!!! Using expand method here.
Usage: return value include the scale parameters and train matrix!
```

```
> example
```

```
$mat
              X         Y        Z
 [1,] 0.1611646 0.6488857 0.559017
 [2,] 0.3223292 1.2977714 0.559017
 [3,] 0.4834938 0.6488857 0.559017
 [4,] 0.6446584 0.3244428 1.118034
 [5,] 0.8058230 0.9733285 0.559017
 [6,] 0.9669876 0.9733285 1.118034
 [7,] 1.1281521 1.2977714 1.677051
 [8,] 1.2893167 0.6488857 0.559017
 [9,] 1.4504813 1.2977714 0.559017
[10,] 1.6116459 1.2977714 1.677051

$scale
       X        Y        Z
6.204837 3.082207 1.788854

$factor.nu
[1] 2 3
```

The trick is, whenever one tries to transform the test data with categorical features, he could comine the test data and train data together and do the data clean. One thing I would remind you is that, one can not intoduce new levels for particular class variable in predicted model.

#### 1.2.2 Missing Data

Use mean/mode. If there are several mode for one numeric array, we only randomly choose one of them ([1]).

### 1.3 Model Tuning

There are

### 1.4 Penalized SVM

## 2 lars

## 3 GAS

GAS is designed to solve a K-sparse problem, and a greedy algorithm that is able to maximize the AUC in training model. Unlike LASSO/LARS ([2]and such penalized method, GAS uses a special strategy (figure) to find out the solutions. AUC is similar to forward selection, which only adds one variable that is not already in the model and increases the value of AUC. If GAS fails to find out the solution with k variables, it will output the model that generates the maximum AUC instead.

**Algorithm 1** Greed AUC Stepwise

```
Require: initialize selected variable stack $V$, and AUC stack $A$.

Ensure: Generating proper logistic model
  $i \leftarrow 0$
  compute NULL model: $AUC \leftarrow auc_0$, A.push($auc_0$)
  STEP 1
  $i++$
  $auc_i = max^*_j\{auc_{ij}\}, \quad \forall j \in I/V$; where $auc_{ij}$ is the AUC of LRM with
  $V \cup j$ variables
  A.push($auc_i$);
  V.push($argmax^*_{j\in I/V}\{auc_{ij}\}$);
  STEP 2
  if $auc_i < auc_{i-1}$ then
      $i--$
      A.pull(), V.pull()
      Goto Step 1 but change the definition of $max^*$ as the second (or third,
  fourth, etc ) largest number.
      if $max^*$ can not be well-defined then
          Break
      end if
  else
      Goto Step 1
  end if
  OUTPUT final model
```

## References

[1] C. Xiru. Nonparametric two-sample problem with nuisance location and scale parameters. *Acta Mathematica Sinica*, page 05, 1979.

[2] J. Zhu and T. Hastie. Classification of gene microarrays by penalized logistic regression. *Biostatistics*, 5(3):427–443, 2004.