Data Mining Lab 7: Introduction to Support Vector Machines (SVMS)

1 Introduction

This lab will present a very basic introduction to Support Vector Machines. The package used by R is called `e1071` . We will use the churn data.

1.1 Getting Setup

- Download the churn data from the course website.
- Download the library e1071 together with the pdf help file. There is also another file under the Vignettes section. Download this file as well. It gives a further explanation of the package which you may find helpful. .
- Load the churn data into a dataframe
- Load the `e1017` package, which contains the functions to build SVMS in `R`. The package contains an amazing number of different functions. Unbelievable range !!!
- Split data into training and test sets as before.

2 To run a SVM Model

Before you start you need to make sure that the target variable is categorical. This needs to be done to make sure you run the correct type of model. There is an option for specifying the type of model but the package will default if you do not specify the type. See section 5 for more information. To run a very basic model do the following :

```
svm.model <- svm(churn ~ ., data = trainset, cost = 100, gamma = 1)
```

- Training data is in a data frame called `trainset`
- Model uses all the columns (apart from the target variable churn) in the dataframe as we used $'.'$ in the definition of the model
- if you want to select out some variables use the following convention

```
data=trainset[3:10]  -  uses variables 3 to 10
data=trainset[c(1,4,6)] - uses variables 1,4,and 6.
data=trainset[-10] - uses all variables except 10
```

- The cost parameter is the C in the notes - max absolute value for h_i
- gamma is the parameter for the kernel

This creates an object of type `svm`. Again use the function `str` or `names` to see what it contains. By typing

```
svm.model
```

We get the following:

```
Call:
svm(formula = churn ~ ., data = trainset, cost = 100, gamma = 1)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100
      gamma:  1


Number of Support Vectors:  598
```

The command `summary(svm.model)` gives additional information.

3 Assessing Model

To evaluate the model we need to calculate predicted values for the test set and compare them to the target values.

```
svm.pred <- predict(svm.model, testset[, -1])
```

The above calculates predicted classes (see below for how to calculate probabilities). The [,-1] means ignore variable 1 with ', ' indicating take all rows. Variable 1 in this situation is the target variable.

To compare the target values with the predicted values use

```
svmt=table(pred = svm.pred, true = testset[, 1])
svmt
```

The function `classAgreement` computes various agreement measures for the table (something a little different which you can use for other models as well). You should have covered these last year.

3.1 Plot the Results

We can view the result in the original input space. To do this we have to specify two dimensions. You can also slice by a third variable if you so wish. See e1071.pdf file for how to do this. I have to admit 2 dimensions is plenty for me. The support vectors are indicated on this plot by an 'x' with the other points indicated by '0'. The red and black correspond to the two different classes. The command is

```
plot(svm.model,data=trainset,day.mins~intl.charge)
```

3.2 Examine the Support Vectors

The summary of the model will have given the number of support vectors in this case 598. We can look at the value of the support vectors

```
svs=svm.model$SV
```

The object `svs` contains 4 vectors

1. index of case in original data set
2. coefficients of the input variables for the support vectors - location in the original scaled space

The model also prints out what it calls `coefs` which are the $h_i y_i$ of the notes where y_i are ± 1 Again you have to decide which are the "No's" and which are the "yeses". The decision is made alphabetically with +1 for N and -1 for Y. We see the support vectors range from -100 to 100. You can link these data back to the original data using the index available in variable `svm.model$index` . The cases with values ± 100 are cases which are difficult to classify. To create a variable indicating whether a case is a support vector or not you can use the following code (with luck!!)

```
supp=vector(mode="character",length=length(trainset$churn))
cindex=svm.model$index
supp[cindex]="Yes"
supp[-cindex]="No"
```

4 Choice of kernels

The default kernel is the radial basis. See following table for other kernels and the parameters which need to be set - the same table as in the slides with the `R` name for the parameters.

Kernel	Formula	Parameters	`R` name
Linear	$\mathbf{u^T v}$	none	
Polynomial	$\gamma(\mathbf{u^T v} + c_0)^d$	γ, d, c_0	gamma=γ coef0=c_0 degree=d
Gaussian Radial basis fct.	$exp[-\gamma\|\mathbf{u} - \mathbf{v}\|^2]$	γ	gamma=γ
Sigmoid	tanh[$\gamma[\mathbf{u^T v} + c_0]$]	γ, c_0	gamma=γ coef0=c_0

See e1071.pdf for more info.

5 Other Parameters

The `type` parameter should be set to C-classification. if f your target variable is a factor you do not have to worry about this.

You can carry out a k-fold crossvalidation on the training by specifying a value for the parameter `cross`. This gives a range of accuracy figures.

To output probabilities of belonging to the classes include the parameter `probability=TRUE` both in the `svm` function and `predict`. It looks like this

```
svm.model =svm(churn ~ ., data = trainset, cost = 100, gamma =1
  probability=TRUE)
svm.pred = predict(svm.model, testset[, -1], probability=TRUE)
```

To access the probabilites of "Yes" type ;

```
 py=attr(svm.pred, "probabilities")[,2]
```

For p(No) use 1 instead of 2. You can then use all of the other techniques now to evaluate the model. Try plotting whether a case is a support vector or not vs P(Yes).

6 Tuning a Support Vector machine

In the above model we used 1 value for gamma and C. It is suggested that we experiment with different values. We can do this using the `tune` command as follows:

```
obj <- tune.svm(churn~., data = trainset, gamma =
 seq(.5, .9, by = .1), cost = seq(100,1000, by = 100))
```

This can take a very long time. Go do something else. It will return the best values to use for the parameters `gamma` and `cost` . These are onlu suggested values for C and gamma. Experiment!! Just as a matter of interest the package `e1071` contains functions for tuning other classification techniques.