Analysis of SVM In R

linear and non linear SVM

Sunil Srinivas Gummadam
Computer Science
Wayne State University
Detroit, USA
fb8563@wayne.edu

***Abstract*—This paper explains the performance analysis of various popular data sets in R using svm and explains the various kernel functions that we can use for performing svm.**

I. Background Introduction

SVM is a classifier derived from statistical learning theory by Vapnik and Cherovonenkis. Support Vector Machines apply a linear method to the data in a high dimension feature space. This high dimension feature space is non linearly related to the input space. SVM implicit mapping of input data into high dimension feature space is defined by the kernel function. The kernel function returns the inner product between the images of two data points. Learning takes place in the high dimension feature space. The kernel trick is very simple to apply than to project x and x' into feature space. There are two primary advantages with svm they are: Once a kernel function is selected we can practically work with data in any dimension without much significant computational cost. Another advantage is that we can design our own kernel function and that can be applied directly to data with out any loss from the feature extraction process.

II. Classification Methods

A. Common Classification Technique

In classification support vector machines uses hyperplane to distinguish between different classes of data. Optimal performance of hyperplane depends on the maximal margin of separation between the two classes of data. We can solve the quadratic optimization problem and obtain solution in terms of support vectors, they have all the relevant information about the classification problem.

B. Multi-Class Classification Schemes

There are two schemes that we can apply, they are one-against-one and one-against-all. In one against all we train K different classifiers, each classifier is trained to separate one class from the rest. We then decide the classification based on the classifier when combined has the highest decision value. In one-against-one classification method we choose K choose 2(kc2) classifiers, each classifier is used for training on data for 2 classes. Then we classify the data based on the classifier that has the highest vote. Cost estimation for svm

When we use svm for our classification, R allows the users to choose how complex or simple we want our prediction function to be.

C. C- SVM

All the svm functions in R takes an input parameter called C which stands for cost. This parameter explains how much penalty is paid by svm for miss classifying the data. Based on the C value we choose, svm formulates the prediction function. If a higher value for C is chosen, svm uses a complex prediction function to avoid the high penalty. On the other hand if we choose a lower value of C, svm uses a simple prediction function.

D. v-SVM

The running time or training the data set using svm depends on another more intuitive parameter called as v. The v parameter that svm uses has a very interesting property and it acts as limit or cut off error rate for svm. It specifies the upper bound on the training error and also acts a lower bound on the fraction or the number of the support vectors found in the data set.

E. Kernel Functions

Another very important factor which controls the training time for svm is the kernel function which we use. Different packages in R allows the usage of several kernel functions. Some packages in R such as ksvm also allows users to define and use their own kernel function also called as custom kernel function. Though every package does not support using a custom kernel function, it provides alternate advantages to it.

As seen before we know that kernel functions return the inner product between the images of two data points in a high dimension feature space. This defines the concept of similarity with very little computational cost in the high dimension feature space.

Some of the most widely used kernel functions and its implementation are

- linear kernel: The simplest of all kernel functions. Simply we need to return the vectorized sum between the two vectors. In R, its a single line of code :print (sum(x * y)))
- Gaussian Radial Basis Function(RBF) kernel: Its mostly used when we do not have any kind of prior knowledge about the data. It takes into account the variance of data distribution. In R the code is print (exp(-0.001 * sum((x -y)^2)))

- Polynomial Kernel: The most popular in image processing techniques is the polynomial kernel. Uses the cost parameter as its input and also takes into account the dimension that we project our data set into. Does all this computational cost to classify the data. In R: print (sum (t(x) * y) + c)^ d, where c is the cost and d is the dimension.
- Laplace Radial Kernel is very similar to RBF kernel including all its functionality
- Sigmoid kernel is used in Neural Networks.
- Anova Radial Kernel: Uses Diagonal projection, we need a recursive function to implement it. Both the spline and Anova Radial kernel typically are used in regression problems.

III. Results and Discussions

When using svm for mock data with very few observations and less features, we can observe that any kernel function and any package will use the same amount of CPU time.

A. *Analysis on mock data*

Data would be provided in the next sections, the data has output classification into 1 and -1, we observe here that the more complex the kernel function the more time it would consume for its training. When using kernlab package and setting the cost to 250, the various outputs for different kernel functions are

a) tandot : elapsed time is 0.01
b) laplacedot: elapsed time is 0.08
c) Besseldot : elapsed time is 0.01
d) Anovadot : elapsed time is 0.04
e) Splinedot : elapsed time is 0.06

B. *Analysis on Spam Data set:*

Iris Data consists of classifying email data into spam vs non spam, consists of 4601 email observations. The various outputs for different kernel functions corresponding to cost of 10 are

a) rbfdot: elapsed time is 11.69
b) tanhdot: elapsed time is 4.35
c) polydot : elapsed time is 23.85
d) Vanilladot : elapsed time is 20.51

We can extend our analysis to several other data sets and compare the time each kernel function takes for training, but it is necessary to draw some kind of conclusion every time we run svm for classification.

We observe as the number of observations in our data sets increase svm will see an increase in its elapsed time set. In this case we need to remember several things before performing svm, firstly we need to decide on the cost parameter, the more the c value the more svm will use complex prediction function and thus we see increase in elapsed time. For some complex kernel function like the polydot, setting high cost function will significantly use up much of system time and it comes to be as high as 164.33. And the other complex kernel function such as vanilla dot uses 165.39.

IV. Conclusions

Not all data can be classified using linear svm and some times there may be not be any option except to use high cost and complex prediction function. We may just need to know which kernel function to use in which situation.

A. *Packages:*

1) Kernlab: Use this package when you really have a very complex data to classify, it provides support to all the known kernel functions. The model selection it uses is hyper-parameter estimation. When you really want t experiment on different results for different kernel functions and using your custom kernel function use this package. This package is the only package in R that supports user to use custom kernel functions.

2) e1071: Use this package for classifying binary data, categorical data. This package uses comparatively less time than the kernlab package, though it does not provide support to all the available kernel functions, it is a very useful package for many data sets in R. The model selection it generally uses is grid-search function. Trade off with Kernlab is that though it does not provide kernel custom selection, it allows to choose your own model.

Table showing the Analysis of some data sets:

Dataset	Ksvm() Kernlab	Svm() e1071	Svmlight() klaR
Spam	10.07	6.71	34
musk	0.16	0.12	4.65
Breast Cancer	0.05	2.53	1.32
MockData	0.1	0.1	0.1

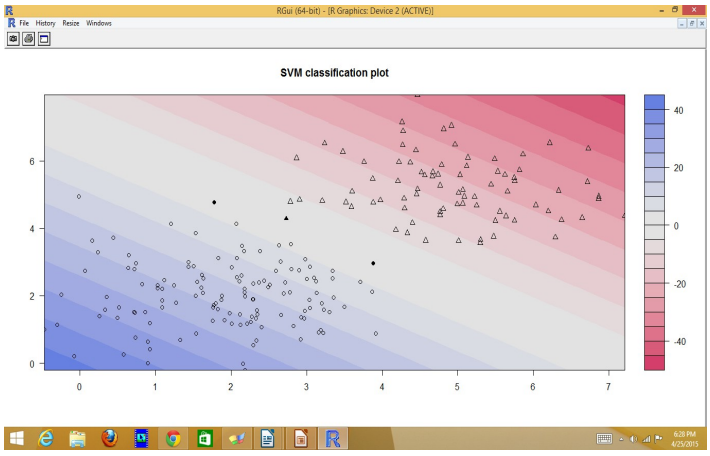

Fig. 1. SVM LINEAR BOUNDARY CLASSIFICATION PLOT

SVM is a very powerful tool that we can use for classification, and regression. Proper usage of the kernel function and model selection will further improve the performance of svm.

References:

1.) Journal of Statistical Software April 2006, Volume 15, Issue 9. http://www.jstatsoft.org/

Alexandros Karatzoglou Technische Universit¨at Wien David Meyer Wirtschaftsuniversit¨at Wien Kurt Hornik Wirtschaftsuniversit¨at Wien

2.) Class slides in SVM by Dr.DongXiaoZhu, Wayne State University