SVM2CRM: support vector machine for the cis-regulatory elements detections

Guidantonio Malagoli Tagliazucchi, Silvio Bicciato
Center for genome research
University of Modena and Reggio Emilia, Modena

October 17, 2016

October 17, 2016

Contents

1 Introduction

The genome-wide identification of cis-regulatory elements (CRE), i.e., those DNA sequences, as promoters, enhancers, insulators, and silencers required to regulate gene expression, still represents a major challenge for computational biology. Recently, the question has been addressed through the analysis of ChIP-seq data with algorithms based on, e.g., Hidden Markov Models ([**?**]). All these approaches require ChIP-seq data for a variety of histone marks, but a consensus on the optimal set of histone modifications for an efficient identification of CRE has not been reached yet. Here we report a computational procedure based on *LibLinear*([**?**]) and ChIP-seq data to determine the optimal number of histone marks necessary for the genome-wide identification of cis-regulatory elements. The methods first performed a step of feature selection to reduce the number of histone marks to use during the analysis. In SVM2CRM we implemented a method based on k-means algorithms followed by the estimation of the index coverage of regualtory regions (ICRR). This value compares the coverage and the signals of the histone marks associated with enhancers and background genomic regions. Then we implemented a function that generate all combinations of histone marksmodels using from n (number minimum of histone marks) to M (number maximum of histone marks). After this step the user can

select the best combination of histone marks to predict enhancers.In this vignette we used SVM2CRM on a reduce public ChIP-seq dataset from CD4+ T-cells([**?**]). Questions, comments or bug reports about this document or about the SVM2CRM functions are welcome. Please address them to the author (guidantonio.malagolitagliazucchi@unimore.it).

2 Input data and pre-processing

The first step of the analysis with SVM2CRM is the preprocessing of the data. Basically, SVM2CRM allows to import bed files in the working directory using cisREfindbed function. This bed files to import should be contain the signals of each histone marks considering the windows size defined by the user. For details about this object see documentation of *SVM2CRMdata*. Specially this function requires diverse parameters. A vector with the list of bed files that the user want use. The bin.size and the window size that respectivelly represent the size of bin used to normalized the data (e.g. 100bp) and windows size that the user want to use to describe the signal of each histone marks (e.g. 1000bp). Finally, the user can set a function to smooth the signal of the histone marks inside a particular window (e.g. median). During this step cisREfindbed generate a data.frame that contains in the columns the histone marks, and in the rows the corresponding signals genome-wide. In the follow example we used cisREfindbed on chromosome 1 and using a windows size of 1000bp without provide a function to model the signals. To reduce the computational cost we considered only one histone modification, but the user can re-run the analysis using all histone modification provide in this package in the external dataset folder.

```
> library(SVM2CRM)
> library(SVM2CRMdata)
> library(GenomicRanges)
> library(rtracklayer)
> library(zoo)
> library(squash)
> library(pls)

> setwd(system.file("extdata", package = "SVM2CRMdata"))
> chr<-"chr1"
> bin.size<-100
> windows<-1000
> smoothing<-"FALSE"
```

```
> function.smoothing<-"median"
> list_file<-grep(dir(),pattern=".bz2",value=TRUE)
>
> #completeTABLE_example<-cisREfindbed(list_file=list_file[1],chr=chr,bin.size=bin.si
>
> #str(completeTABLE_example)
```

3 Generation of training set

In the second step SVM2CRM requires to generate the training set. The user can create this set using getSignal function. getSignal function as for cisREfindbed require to define the list of bed files of the histone modifications to use during the analysis and two tables with the genomics coordinates of the "features", e.g. the p300 binding sites that describe enhancers regions and random regions of the genome. This function simply models the signals of each histone marks around the features used in the input files and considering the bin.size and windows size defined during the step of preprocessing. Here we used two training set already generated using getSignal function.The first one contain the signal of the histone marks in correspondence of p300 binding sites (or other genomic-regions of true enhancers). The second matrix contains the signals of the histone marks in correspondence of genomic-random regions (background genomic regions). In the follow comment lines we reported two examples of the usage of getSignal function. Finally we merged together the results of getSignal.

```
> setwd(system.file("data",package="SVM2CRMdata"))
> load("CD4_matrixInputSVMbin100window1000.rda")
> completeTABLE<-CD4_matrixInputSVMbin100window1000
> new.strings<-gsub(x=colnames(completeTABLE[,c(6:ncol(completeTABLE))]),pattern="CD4
> new.strings<-gsub(new.strings,pattern=".norm.w100.bed",replacement="")
> colnames(completeTABLE)[c(6:ncol(completeTABLE))]<-new.strings
> #list_file<-grep(dir(),pattern=".sort.txt",value=T)
>
> setwd(system.file("data",package="SVM2CRMdata"))
> load("train_positive.rda")
> load("train_negative.rda")
> #train_positive<-getSignal(list_file,chr="chr1",reference="p300.distal.fromTSS.txt"
> #train_negative<-getSignal(list_file,chr="chr1",reference="random.region.hg18.nop30
> #training_set<-rbind(train_positive,train_negative)
>
```

```
> training_set<-rbind(train_positive,train_negative)
> colnames(training_set)[c(5:ncol(training_set))]<-gsub(x=gsub(x=colnames(training_se
```

4 Integration of the annotation with the signals

Using createSVMinput we can integrate the results of getSignal with an annotation of cis-regulatory elements already characterized and then add two labels that describe for example if a particular genomic region is an enhancer region or not. The user can set specific two labels to indicate the functional role of the chromatin. (e.g. "enhancers", "not enhancers")

```
> setwd(system.file("extdata", package = "SVM2CRMdata"))
> data_level2 <- read.table(file = "GSM393946.distal.p300fromTSS.txt",sep = "\t", str
> data_level2<-data_level2[data_level2[,1]=="chr1",]
> DB <- data_level2[, c(1:3)]
> colnames(DB)<-c("chromosome","start","end")
> label <- "p300"
> table.final.overlap<-findFeatureOverlap(query=completeTABLE,subject=DB,select="all"
> data_enhancer_svm<-createSVMinput(inputpos=table.final.overlap,inputfull=completeTA
> colnames(data_enhancer_svm)[c(5:ncol(data_enhancer_svm))]<-gsub(gsub(x=colnames(dat
>
```

5 Variables selection

When the users want to perform the research genome-wide of enhancers regions can handle ChIP-seq dataset containing several experiments of different histone marks. The number of histone modifications is certainly a variable that impacts on the prediction of enhancers. Different works tried to define which is the best combination of histone marks to predict enhancers. However it seems that until now there is not a consensus about the optimal number of histone marks for the prediction of this class of cis-regulatory elements. Moreover, in the genome the number of histone marks is greater than of 50 and the functional and biological roles of each of those are not yet clear. From the computationl aspect, consider a large number of variables (histone marks) can introduce biases. In particular redundant signals between histone marks can be considered. Source of error on the prediction of enhancers is the usage of histone marks that are slightly associated with enhancers regions. For that reason, before the perfoming the prediction of enhancers it is necessary to filter the initial dataset from the variables that

could influence the analysis of prediction. Therefore we implemented a step of features selection in SVM2CRM. In particular we developed a two step based procedure of variable filtering encoded in smoothInputFS and featSelectionWithKmeans functions based on k-means algorithm and the index of coverage of the regulatory regions ICRR. In the first step smoothInputFS allow to perfom the signals smoothing of each histone mark. For example, if the data were binned of 100 bp, a windows size of smoothing equal to 2 means that the signal of the histone marks is smooth every 200 bp. Next using k-means algorithm with a nk number of histone marks defines by the user it is possible clusterized the signals of the histone marks and estimate the means of these inside each group with featSelectionWithKmeans. In the second step the index coverage of the regulatory regions (ICRR) is estimated (for details see. the documentation of featSelectionWithKmeans function). Briefly, this is an index that is computed from the total coverage of the positive features (e.g. p300 binding sites) and background genomic regions and associates the signals of eachclass with the coverages. The results of featSelectionWithKmeans can be filtered using two thresholds based on the signals median of histone marks and considering the a ICRR values maximum. This values assume values from 0 to 1. A value close to 0 means that the difference between the coverage of the two classes of enhancers is little, in contrast, if this value is close to 1 this means that there is a diversity between the two fraction of the cis-regulatory elements. featSelectionWithKmeans is a function that returns a object (list) with different elements. The first one contain a matrix with the results of the feature selection analysis, the parameters defined by the user and the histone marks recovered after feature selection. Finally the user can visualize the results of the analysis from two plot: in thefirst the x-axis contain the name of the histone marks while in y-axis the median of mean of each group.The second plot contains the index of coverage between enhancers and not enhancers regions.

Here we created a vector with the list of histone marks to use during the step of features selection.

```
> listcolnames<-c("H2AK5ac","H2AK9ac","H3K23ac","H3K27ac","H3K4me1","H3K4me2","H3K4me
```

The first step of feature selection perfoms the smoothing of the signals using a windows size of k, here for e.g. we set k equal to 20 that correspond to 20*100=2000

```
> dftotann<-smoothInputFS(train_positive[,c(6:ncol(train_positive))],listcolnames,k=2

[1] "H2AK5ac"
[1] "rename"
```

```
[1] "H2AK9ac"
[1] "rename"
[1] "H3K23ac"
[1] "rename"
[1] "H3K27ac"
[1] "rename"
[1] "H3K4me1"
[1] "rename"
[1] "H3K4me2"
[1] "rename"
[1] "H3K4me3"
[1] "rename"
```

The second step allows to run the feature selection step. The only parameter required is the number of cluster for k-means algorihtm (nk). featSelectionWithKmeans implement also the automatically investigation of nk clusters using a Bayesian Information Criterion for EM initialized.

```
> results<-featSelectionWithKmeans(dftotann,5)
```

```
[1] "H2AK5ac_1"
[1] "H2AK9ac_1"
[1] "H3K23ac_1"
[1] "H3K27ac_1"
[1] "H3K4me1_1"
[1] "H3K4me2_1"
[1] "H3K4me3_1"
[1] "H2AK5ac_1"
[1] "H2AK9ac_1"
[1] "H3K23ac_1"
[1] "H3K27ac_1"
[1] "H3K4me1_1"
[1] "H3K4me2_1"
[1] "H3K4me3_1"
```

The "results" object is a list that containg 7 elements. Here we save a data.frame that contain in the first column all histone marks, in the second column the signals of the histone modifications smooth. In the last column are the ICRR values.

```
>     resultsFS<-results[[7]]
```

A second threshold based on the ICRR values was used. In fact in the plot generated with featSelectionWithKmeans we observed that for some histone marks with high signals they have also high values of ICRR. This means that these features have high signals for few cis-regulatory elements. Here we filtered using a value of "0.5".

```
> resultsFSfilterICRR<-resultsFS[which(resultsFS[,3]<0.26),]
```

The list of histone marks after feature selection analysis is saved.

```
> listHM<-resultsFSfilterICRR[,1]
> listHM<-gsub(gsub(listHM,pattern="_.",replacement=""),pattern="CD4.",replacement=""
```

Finally we saved the list of histone marks after feature selection analysis.

```
> selectFeature<-grep(x=colnames(training_set[,c(6:ncol(training_set))]),pattern=past
> colSelect<-c("chromosome","start","end","label",selectFeature)
> training_set<-training_set[,colSelect]
>
```

6 Tuning parameters for SVM

SVM2CRM implements function from *LibLinear* package. Before to proceeding with to prediction it is necessary performs the tuning of parameters for svm predictions. Different parameters can be tuned by the user to predict the different classes correctly. The first parameter to define is the the weight of each class. If you generated a training set with a different number of positive and negative examples (e.g. the binding sites of p300 or random regions) you need to set the weight of each class (e.g. enhancers/not enhancers) to investigate the effect of the weights on the prediction. The second important parameter to define is the type of kernel "typeSVM". SVM2CRM is based on *LibLinear*. This package implement different kinds of kernel functions: 0 -L2- regularized logistic regression, 1 -L2-regularized L2-loss support vector classification (dual), 2 -L2-regularized L2-loss support vector classification (primal), 3 -L2-regularized L1-loss support vector classification (dual), 4 -multi-class support vector classification by Crammer and Singer, 5 -L1-regularized L2-loss support vector classification, 6 -L1-regularized logistic regression, 7-L2-regularized logistic regression (dual). The last parameter to set is "costV", the cost of constraints violation. The default value is 1. Finally, if you have more than 2 histone marks, you need to find which is the best combination of histone modification to predict cis-regulatory elements.

SVM2CRM allows to perform this analysis using tuningParametersCombROC function. tuningParametersCombROC implements performanceSVM function that estimate the performance of prediction during each iteration of tuningParametersCombROC (for details ?perfomanceSVM). The function tuningParametersCombROC allow high flexibility: the user can initially set the type of kernel, the cost, the number of histone marks. The output of tuningParametersCombROC is a data.frame where for each model there are the parameters compute with performanceSVM. To help the user to discriminate how to choose the best model SVM2CRM implement several functions to plot the performance obtained with performanceSVM during the use of tuningParametersCombROC. In the follow code we wanted asses the perfomance of prediction using all histone marks avaible after the step of feature selection. Then, we created several vectors containing different parameters of typeSVM and costV, but we run the analysis only using typeSVM and costV equal to 0.

```
> vecS <- c(2:length(listHM))
> typeSVM <- c(0, 6, 7)[1]
> costV <- c(0.001, 0.01, 0.1, 1, 10, 100, 1000)[6]
> infofile<-data.frame(a=c(paste(listHM,"signal",sep=".")))
> infofile[,1]<-gsub(gsub(x=infofile[,1],pattern="CD4.",replacement=""),pattern=".sor
```

In this example we test the perfomance of all models considerig a cost 0, a L2- regularized logistic regression and we set a number of positive and negative classes respectivelly of 100 and 400.

```
> tuningTAB <- tuningParametersCombROC(training_set = training_set, typeSVM = typeSVM
```

```
[1] "last comparison"
[1] 6 2
[1] 6 2
[1] "k-fold-validation in all combination"
[1] "H2AK5ac.signal" "H3K23ac.signal"
[1] "H2AK5ac.signal" "H3K4me1.signal"
[1] "H2AK5ac.signal" "H3K4me2.signal"
[1] "H3K23ac.signal" "H3K4me1.signal"
[1] "H3K23ac.signal" "H3K4me2.signal"
[1] "H3K4me1.signal" "H3K4me2.signal"
```

Because tuningParametersCombROC estimate the performance of prediction for each model the user can filter the results using several criteria

(e.g. sensitivity, specificity etc). Here we used the F-score asparameter to reduce the total number of generated models. For each dataset and cell type it is necessary to manually set the correct values of F-score adequate to exclude the models that risk to overfit. In particular we set a thresholds of F-score <= to 0.95. Next the user can use a further filter using two approaches: in the first one perform the manually selection of the histone marks with the highest frequencies. Secondly the user can use the top models of tuningComParameterROC.

Here we used a F-score of 0.95 such threshold and selected all models with the number of histone marks that is greater than 2 and with maximimum F-score.

```
> tuningTABfilter<-tuningTAB[(tuningTAB$fscore<0.95),]
> row_max_fscore<-which.max(tuningTABfilter[,"fscore"])
> listHM_prediction<-gsub(tuningTABfilter[row_max_fscore,4],pattern="//",replacement=
```

7 Prediction genome wide

In the last step, the user can perfom the prediction of enhancers genome-wide using the histone marks defined during the step of feature selection and the best parameters selected using tuningParametersCombROC. The prediction of cis-regulatory elements genome-wide is possibile using the function predictionGW. This function return a bed file with the genomic coordinate of putative-enhancers.

```
> columnPR<-grep(colnames(training_set),pattern=paste(listHM_prediction,collapse="|")
> predictionGW(training_set=training_set,data_enhancer_svm=data_enhancer_svm, listHM=

[[1]]
An object of class "performance"
Slot "x.name":
[1] "None"

Slot "y.name":
[1] "Area under the ROC curve"

Slot "alpha.name":
[1] "none"

Slot "x.values":
```

```
list()

Slot "y.values":
[[1]]
[1] 0.9782388

[[2]]
[1] 0.9763366

[[3]]
[1] 0.9745577


Slot "alpha.values":
list()


[[2]]
     tpr.sensitivity        fpr       acc    fscore spc.specificity       ppv
res        0.9349112 0.08719852 0.9157979 0.9237241       0.9128015 0.6269841
res1       0.9329897 0.09211776 0.9117883 0.9202647       0.9078822 0.6510791
res2       0.9480519 0.06770357 0.9342422 0.9401082       0.9322964 0.6636364
           npv       fdr
res  0.9889447 0.3730159
res1 0.9865841 0.3489209
res2 0.9922103 0.3363636

[[3]]
[1] -0.006092356 -0.010539233  0.005505496

[[4]]
[1] 1.771238e-89 2.869043e-99 1.382200e-81
```

8 References

9 Session information

The output in this vignette was produced under the following conditions:

```
> sessionInfo()
```

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] parallel  stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] pls_2.5-0            squash_1.0.7         zoo_1.7-13
 [4] rtracklayer_1.34.0   GenomicRanges_1.26.0 GenomeInfoDb_1.10.0
 [7] IRanges_2.8.0        S4Vectors_0.12.0     BiocGenerics_0.20.0
[10] SVM2CRM_1.6.0        SVM2CRMdata_1.5.0    LiblineaR_1.94-2

loaded via a namespace (and not attached):
 [1] XVector_0.14.0             bitops_1.0-6
 [3] tools_3.3.1                zlibbioc_1.20.0
 [5] mclust_5.2                 boot_1.3-18
 [7] RSQLite_1.0.0              lattice_0.20-34
 [9] Matrix_1.2-7.1             DBI_0.5-1
[11] spam_1.4-0                 Biostrings_2.42.0
[13] gtools_3.5.0               caTools_1.17.1
[15] fields_8.4-1               maps_3.1.1
[17] grid_3.3.1                 Biobase_2.34.0
[19] dtw_1.18-1                 AnnotationDbi_1.36.0
[21] XML_3.98-1.4               BiocParallel_1.8.0
[23] gdata_2.17.0               ROCR_1.0-7
[25] Rsamtools_1.26.0           gplots_3.0.1
[27] CircStats_0.2-4            GenomicAlignments_1.10.0
[29] MASS_7.3-45                SummarizedExperiment_1.4.0
[31] KernSmooth_2.23-15         proxy_0.4-16
[33] RCurl_1.95-4.8             verification_1.42
```