Using the OncoSig Classifiers to Discover Novel Oncoprotein network Dependencies

Joshua Broyde, Diana Murray, Barry Honig, Andrea Califano
Columbia University, New York, USA

December 13, 2018

Introduction

OncoSig comprises a set of machine learning approaches for determinging novel sets of gene products (i.e. genes or proteins) that support the activity of an oncogene or tumor suppressor (i.e. Oncoprotein-Centric Map, or OC-map). This is relevant for determining which genes/proteins are involved in an oncoprotein's functional network. OncoSig queries a molecular interaction network or other features regarding a protein's function to predict novel members of the OC-map. This molecular interaction network could contain features such as protein-protein interactions, or gene regulatory networks. OncoSig can be used primarily in two ways, in a supervised or unsupervised fashion. In the supervised fashion OncoSig uses either a Naive Bayes or Random Forest classifier to train on the molecular interaction network and a gold standard of known members of a particular OC-Map (for example, the members of the KRAS signaling pathway). This approach is appropriate for cases where some members of an OC-Map are known and one wants to leverage the known ones to predict other OC-Map members.

In a cases where a gold standard is not known, OnconSig can be used in an unsupervised fashion, where an OC-Map trained on a well characterized Oncoprotein is applied to one that is poorly characterized. This usage is appropriate where there is no gold standard for a particular Oncoprotein.

Installation and loading

After first installing R (http://www.r-project.org) and the OncoSig library, load OncoSig.

```
> library("OncoSig")
```

OncoSig Naive Bayes Classifier

The OncoSig Naive Bayes (OncoSigNB) Classifier is a supervised learning approach that is well suited to discovering OC-Map members when there are a few number of features describing each gene product and when the features have no or low statistical dependence. To run OncoSigNB, we create dataframes that correspond to the training and testing sets (which are labeled as 1 or 0, if they are in the gold standard OC-Map or not, respectivley). For example:

```
> df_1=read.delim("~/OncoSig/Input_data_files/Naive_Bayes_evidences_set_1.txt",header=TRUE)
> df_2=read.delim("~/OncoSig/Input_data_files/Naive_Bayes_evidences_set_2.txt",header=TRUE)
> the_bins=list(c(0,40,200,1200),c(0,.1),c(-2,-0.15,-0.02,0.0925),c(1,2,6),
+              c(0,0.25),c(1,3,20),c(1,4,20),c(1,4,20),c(0,0.0001,0.9999),
+              c(0,0.01,0.05))
> predictions=OncoSigNB(training_set = df_1,testing_set = df_2,
+                       the_bins=the_bins,correlated_features =list())
```

In this example, we specified the training and testing sets, how to bin the data for each feature, and passed an empty list to indicate that there are no correlated features.

The input training and testings should be formatted like so.

```
> df_1=read.delim("~/OncoSig/Input_data_files/Naive_Bayes_evidences_set_1.txt",header=TRUE)
> df_1[1:5,]

      V1 df_labels PREP_LR luad_vip_pv  luad_lincs MS_RALGDS
1 Q16539         1  676.84    1.00e+00  0.13985800        NA
2 P78383         1      NA          NA          NA        NA
3 P30281         1      NA    1.00e+00 -0.02819036        NA
4 P28799         1   30.83    7.97e-05  0.34413929        NA
5 Q96HE7         1      NA    1.00e+00 -0.33786435        NA
  Luad_vip_up MS_TBK1 MS_RALA MS_RALB Demand_pv mindy_overlap_lung
1          NA      NA      NA      NA         1       1.0000000000
2          NA      NA      NA      NA         1                 NA
3          NA      NA      NA      NA         1       0.0682243033
4          NA      NA      NA      NA         1       0.0002509032
5          NA      NA      NA      NA         1       0.0771524132
```

Note that when a feature is missing, it is simply coded as NA. The first column is the name (e.g. a gene name), the second column is the label, and all later columns are features.

OncoSigRF

Supervised classification can also be run using the OncoSig Random Forest (OncoSig RF) classifier. Random Forests can be used when the features are statistically dependent, and can more easily be used when the number of features number in the thousands or more. We recommend OncoSigRF when integrating a molecular interaction network that may contain many tens of thousands (note that when using a network of more than a few hundred thousand interactions, OncoSig RF will require a few hours and a prohibitive amount of memory to run).

```
> library (randomForest)
```

First we read in the molecular interaction network and convert it to a matrix

```
> Network_location="~/OncoSig/Input_data_files/LUAD/original_network_sample.txt"
> Network=read.delim(Network_location,header=F)
> Network$V1=as.character(Network$V1)
> Network$V2=as.character(Network$V2)
> Network$V3=as.numeric(Network$V3)
> Network=as.matrix(Network)
> Network[,3]=as.numeric(Network[,3])
> Network_matrix=listToMatrix(Network)
```

Note the format of the input network. The first column is the feature name, the second column is the gene product (i.e. the row of data), and the third column is the score.

```
> Network[1:5,]

     V1                 V2       V3
[1,] "Q9Y5P4_CINDY_SIG" "Q8N653" "79"
[2,] "A6NF89_PREPPI"    "P47881" "2109.56"
[3,] "P61586_PREPPI"    "Q6ZUM4" "6546.191"
[4,] "Q06124_CINDY_SIG" "O14647" "50"
[5,] "Q9Y606_ARACNE"    "Q00613" "0.3334542"
```

Now read in and process the gold standard

```
> Gold_Standard_location=
+   "~/OncoSig/Input_data_files/LUAD/10_oncogene_pathways/KRAS/total.txt"
> Gold_Standard=read.delim(Gold_Standard_location,header=F)
> Gold_Standard$V1=as.character(Gold_Standard$V1)
```

preprocess the data

```
> #Convert Matrix to Dataframe for future steps
> Network_matrix_df=as.data.frame(Network_matrix)
> #Remove Members of Gold Standard Not in the Network:
> Gold_Standard_in_Network_names=intersect(rownames(Network_matrix_df),
+                                Gold_Standard$V1)
> Negative_Set_names=setdiff(rownames(Network_matrix_df),
+                                 Gold_Standard_in_Network_names)
> remove(Network_matrix) #delete Matrix.
```

Run the OncoSigRF Classifier

```
> Query_output_results=OncoSigRF(Network_matrix_df,
+              Gold_Standard_in_Network_names, max_iterations=5)
> Query_output_results_scores=as.data.frame(Query_output_results[[1]])
```

Unsupervised OncoSig

Supervised classification is only applicable when a gold standard suitable for training can be found. However, some Oncogenes/Tumor Suppressors may have no known gene product dependencies. In this case, we can apply a forest created specifically using one Oncogene/Tumor Suppressor and apply it to another. See the documentation for the OncoSigUnsup function for further details. In this example, we read in a random forest created using features for the EGFR oncogene and apply it to the KRAS oncogene.

```
> KRAS_features=
+   "~/OncoSig/Input_data_files/LUAD/OncoSigUnsup/feature_list_KRAS.txt"
> EGFR_forest=
+   "~/OncoSig/Input_data_files/LUAD/OncoSigUnsup/All_forests_EGFR.r"
> results=OncoSigUnsup(KRAS_features,EGFR_forest)
```