# *COSNet*: An R Package for Predicting Binary Labels in Partially-Labeled Graphs

Marco Frasca and Giorgio Valentini

October 26, 2021

Dipartimento di Informatica
Università degli Studi di Milano
Via Comelico 39/41, 20135 Milano, Italy
frasca@di.unimi.it

## Scope and Purpose of this Document

This document is a user manual for *COSNet*, the software implementing the model developed by Bertoni *et al.* (2011), Frasca *et al.* (2013). It provides an introduction into how to use COSNet. Not all features of the R package are described in full detail. Such details can be obtained from the documentation enclosed in the R package.

## Contents

# 1 Introduction

This package implements the algorithm *COSNet* (Bertoni *et al. 2011*, Frasca *et al. 2013*), which has been proposed for predicting node labels in partially labeled graphs, especially when labelings are highly unbalanced. In this context, nodes represent the instances of the problem, whose labels can be positive (+) or negative (-). Unbalanced labeling means that one class (usually the negative class) considerably outnumbers the other one. Many real world problems are characterized by few positives and much more negatives, such as the gene function prediction, where genes having the most specific biomolecular functions are very few (Ashburner *et al. 2000*), or in medical diagnosys of cancer, where patients having a certain cancer (positive class) are the large minority. The instances are only partially labeled, and the aim is to extend the labeling to all the instances. In this context, imbalance-unaware algorithms may suffer high decay in performance when classifying new instances (Ling and Sheng *2007*). *COSNet* automaticly learns from the input data the model parameters able in dealing with the label imbalance, and efficiently infers binary labels for the unlabeled instances in the graph.

Formally, the input of *COSNet* is represented by a weighted graph $G = (V, \boldsymbol{W})$, where $V$ is the set of nodes and $\boldsymbol{W} = (w_{ij}|_{i,j=1}^{n})$ is the symmetric weight matrix with null diagonal: the weight $w_{ij} \in [0, 1]$ denotes a similarity index of node $i$ with respect to node $j$. The labeling of $V$ in positive $V_+$ and negative $V_-$ nodes is known only for a subset $S \subset V$, while is unknown for $U = V \setminus S$. The aim is to extend the labeling to nodes in $U$, that is inferring a bipartition of $U$ in positive $U_+$ and negative $U_-$ instances. The labeling imbalance can be represented through a coefficient $\epsilon = |S_+|/|S_-|$, where $S_+$ and $S_-$ are the sets of positive and negative examples, respectively. The labeling is considered highly unbalanced when $\epsilon << 1$.

# 2 COSNet

*COSNet* is a binary classifier based on a parametric Hopfield network which embeds the partial labeling and node similarities $\boldsymbol{W}$ and predict the binary labels for the unlabeled nodes through an asynchronous dynamics, which updates only the unlabeled neurons. In order to deal with label imbalance,

the model introduces two parameters, $\alpha \in [0, \frac{\pi}{2}[$ and $\gamma \in \mathbb{R}$, determining respectively the neuron activation values $(\sin\alpha,\ -\cos\alpha)$ and the neuron activation thresholds. The parameters are automatically learned by an efficient supervised procedure on the basis of the input data. The initial state for neuron $v \in V$ is set as follows:

$$x_v(0) = \begin{cases} \sin\alpha & \text{if } v \text{ is positive labeled} \\ -\cos\alpha & \text{if } v \text{ is negative labeled} \\ 0 & \text{if } v \text{ is unlabeled} \end{cases}$$

For each unlabeled node $i$, the initial state $x_i = 0$ is changed to $-\cos\alpha$ or to $\sin\alpha$ according to the following asynchronous dynamics:

$$x_i(t) = \begin{cases} \sin\alpha & \text{if } \sum_{j=1}^{i-1} W_{ij}x_j(t) + \sum_{k=i+1}^{n} W_{ik}x_k(t-1) - \gamma > 0 \\ -\cos\alpha & \text{if } \sum_{j=1}^{i-1} W_{ij}x_j(t) + \sum_{k=i+1}^{n} W_{ik}x_k(t-1) - \gamma \leq 0 \end{cases}$$

where $n = |V|$ and $t$ is the current time. At each time $t$, the state of the network is $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_n(t))$, and a Lyapunov state function named *energy function* is associated to the network:

$$E(\boldsymbol{x}) = -\frac{1}{2}\sum_{\substack{i,j=1 \\ j\neq i}}^{n} W_{ij}x_i x_j + \sum_{i=1}^{n} x_i\gamma$$

The dynamics converges to an equilibrium state $\hat{\boldsymbol{x}}$ corresponding to a minimum of $E$ (Frasca *et.al 2013*), which is used to infer the bipartiton $(U_+, U_-)$ of $U$: $U_+ = \{i \in U, \hat{x}_i = \sin\alpha\}$ and $U_- = \{i \in U, \hat{x}_i = -\cos\alpha\}$. Furthermore, *COSNet* can been adopted as ranker by assigning to each neuron a score related to its internal energy at equilibrium. More precisely, the score assigned to neuron $i \in U$ is the following:

$$s(i) = \sum_{j\neq i}(W_{ij}\hat{x}_j - \gamma) \tag{1}$$

# 3 An Example of the Usage of *COSNet* for the Functional Classification of Yeast Genes with the Functional Catalogue

In this section we apply *COSNet* to predict the functions of yeast proteins. We adopt a binary protein-protein interactions data set of 2338 yeast proteins from the STRING data base (von Mering et al. 2002), contained in the R package *bionetdata*[1] and the corresponding Functional Catalogue (FunCat) annotations.

[1]The package can be downloaded at http://cran.r-project.org/web/packages/bionetdata/index.html.

### 3.1 Data Loading

First, let us load the library and check the data:

```
> library(COSNet);

COSNet: Cost-Sensitive algorithm for binary classification in graphs.

> library(bionetdata)
> data(Yeast.STRING.data)
> dim(Yeast.STRING.data)

[1] 2338 2338

> rownames(Yeast.STRING.data)[1:10]

 [1] "YJR121W" "YAL009W" "YGR165W" "YLR298C" "YOL052C" "YOL051W" "YPR083W"
 [8] "YOR110W" "YGR032W" "YML015C"
```

The named squared matrix *Yeast.STRING.data* contains 1 in position ($i$, $j$) if the corresponding proteins interact, 0 otherwise. In the same package, in order to define the binary labels, we load the annotations of 176 FunCat classes for the proteins included in *Yeast.STRING.data*. Annotations refer the funcat-2.1 scheme, and funcat-2.1 data 20070316 data, available from the MIPS web site[2].

```
> data(Yeast.STRING.FunCat)
> dim(Yeast.STRING.FunCat)

[1] 2338  177

> rownames(Yeast.STRING.FunCat)[1:10]

 [1] "YJR121W" "YAL009W" "YGR165W" "YLR298C" "YOL052C" "YOL051W" "YPR083W"
 [8] "YOR110W" "YGR032W" "YML015C"

> colnames(Yeast.STRING.FunCat)[1:10]

 [1] "00"          "01"          "01.01"       "01.01.03"    "01.01.06"
 [6] "01.01.06.05" "01.01.09"    "01.02"       "01.03"       "01.03.01"
```

The number of columns is 177 because the authors added a dummy class "00". Even in this case, *Yeast.STRING.FunCat* is a binary matrix whose $i, j - th$ component is 1 if protein $i$ is annotated with class $j$, 0 otherwise. Note that the row names of both *Yeast.STRING.FunCat* and *Yeast.STRING.data* are identical. We first exclude the dummy class, that is useful only when hierarchical computations are performed, and then we select some classes and, since *COSNet* needs $\{1, -1\}$-labels, we change each 0-component of *Yeast.STRING.FunCat* to -1.

[2] http://mips.gsf.de/projects/funcat

```
> ## excluding the dummy "00" root
> to.be.excl <- which(colnames(Yeast.STRING.FunCat) == "00")
> Yeast.STRING.FunCat <- Yeast.STRING.FunCat[, -to.be.excl]
> ## choosing the first 35 classes
> labeling <- Yeast.STRING.FunCat[, 1:35]
> ## number of positive labels
> colSums(labeling)
```

```
         01       01.01    01.01.03    01.01.06 01.01.06.05    01.01.09
        859         170          32          45          27          56
      01.02       01.03    01.03.01 01.03.01.03    01.03.04    01.03.16
         49         161          50          26          38          57
01.03.16.01       01.04       01.05    01.05.02 01.05.02.04 01.05.02.07
         36         239         277          56          26          54
   01.05.03    01.05.25       01.06    01.06.02 01.06.02.01    01.06.06
         35          78         122          33          25          21
      01.07    01.07.01       01.20          02       02.01       02.07
        119          87          39         196          27          23
      02.10       02.11       02.13    02.13.03       02.19
         24          24          65          34          35
```

```
> Yeast.STRING.FunCat[Yeast.STRING.FunCat == 0] <- -1
```

### 3.2 Predicting Labels for Unlabeled Nodes with *COSNet*

Now we predict the node labels through a 5-fold cross validation procedure implemented by the function `cosnet.cross.validation` provided by the `COSNet` package. This procedure at each time hides the labels in a fold and predicts them with *COSNet*.

```
> out <- cosnet.cross.validation(labeling, Yeast.STRING.data, 5, cost=0)
```

Note that the `cost` parameter of *COSNet* is set to 0: this means that the unregularized version is adopted. Now we test the regularized version of *COSNet* on the same data:

```
> out.r <- cosnet.cross.validation(labeling, Yeast.STRING.data, 5, cost=0.0001)
```

### 3.3 Processing the Results

The output of the `cosnet.cross.validation` function is a list whose fields are: ”labels”, named matrix containing the input labels; ”predictions”, named matrix containing the binary predictions; ”scores”, named matrix containing the predicted scores according to Eq. (1).

```
> predictions <- out$predictions
> scores <- out$scores;
> labels <- out$labels;
> predictions.r <- out.r$predictions
> scores.r <- out.r$scores;
> labels.r <- out.r$labels;
```

### 3.4 Assessing *COSNet* Performance

We now evaluate the performance of *COSNet* in terms of F-score, Area under the ROC Curve (AUC) and in terms of Precision at $x$ Recall level (PxR). We use the R package *PerfMeas*, which provides functions to compute the performance measures we need:

```
> library(PerfMeas);
> ## computing F-score
> Fs <- F.measure.single.over.classes(labels, predictions);
> ## Average F-score
> Fs$average[4]

        F
0.3422293

> Fs.r <- F.measure.single.over.classes(labels.r, predictions.r);
> # Average F-score for the regularized version of COSNet
> Fs.r$average[4]

       F
0.369015

> ## Computing AUC
> labels[labels <= 0] <- 0;
> labels.r[labels.r <= 0] <- 0;
> auc <- AUC.single.over.classes(labels, scores);
> ## AUC averaged across classes
> auc$average

[1] 0.723827

> auc.r <- AUC.single.over.classes(labels.r, scores.r);
> ## AUC averaged across classes for the regularized version of COSNet
> auc.r$average

[1] 0.723248
```

```
> ## Computing precision at different recall levels
> PXR <- precision.at.multiple.recall.level.over.classes(labels,
+ scores, seq(from=0.1, to=1, by=0.1));
> ## average PxR
> PXR$avgPXR

       0.1        0.2        0.3        0.4        0.5        0.6        0.7
0.59117243 0.49228825 0.38739390 0.30099732 0.19900342 0.11903425 0.07386392
       0.8        0.9          1
0.06088447 0.05229085 0.04187044

> PXR.r <- precision.at.multiple.recall.level.over.classes(labels.r,
+         scores.r, seq(from=0.1, to=1, by=0.1));
> ## average PxR for the regularized version of COSNet
> PXR.r$avgPXR

       0.1        0.2        0.3        0.4        0.5        0.6        0.7
0.62406194 0.50142439 0.41747138 0.31921779 0.19777373 0.12346457 0.06865577
       0.8        0.9          1
0.06002590 0.05173550 0.04184401
```

# 4 An Example of the Usage of *COSNet* for Inferring Gene Ontology Labels for Fly Genes

In this section we show an application of *COSNet* in predicting the Gene Ontology (GO)(Ashburner *et al. 2000*) labels for 9361 Drosophila melanogaster genes. The input similarity matrix is obtained by integrating several types of data, including co-expression, genetic interactions, protein ontologies and physical interactions. The Gene Ontology annotations (release 15-5-13) with 3-300 annotated genes have been considered. Both data and annotations can be downloaded at *http://frasca.di.unimi.it/cosnetdata/*.

## 4.1 Data Loading

```
> ## reading similarity network W
> W <-
+ as.matrix(read.table(file=paste(sep="", "http://frasca.di.unimi.it/",
+  "cosnetdata/u.sum.fly.txt"), sep=" "))
> ## reading GO annotations
> GO.ann.sel <-
+ as.matrix(read.table(file=paste(sep="", "http://frasca.di.unimi.it/",
+ "cosnetdata/GO.ann.fly.15.5.13.3_300.txt"), sep = " ",))
> GO.classes <- colnames(GO.ann.sel)
> ## changing "." to ":"
```

```
> GO.classes <- unlist(lapply(GO.classes, function(x){
+                 substr(x, 3, 3) <- ":"; return(x)}))
> colnames(GO.ann.sel) <- GO.classes;
```

### 4.2 Predicting GO Labels with *COSNet*

Now we determine a random partition in 3 folds of the input data, hide the labels of the genes in one of these folds (test set), and use the labels in the other 2 folds as training set for *COSNet*.

```
> n<-nrow(W);
> ## selecting some classes to be predicted
> classes <- c("GO:0009605", "GO:0022414", "GO:0032504",
+              "GO:0002376", "GO:0009888", "GO:0065003");
> labels <- GO.ann.sel[, classes]
> ## for COSNet negative labels must be -1
> labels[labels <= 0] <- -1;
> ## Determining a random partition for the class GO:0009605 in 3 folds
> ##    ensuring that each fold has a similar proportion of positives
> folds <- find.division.strat(labels[, 1], 1:n, 3)
> ## hiding the labels of the test set (the fold of index 1)
> labels[folds[[1]], ] <- 0;
> ## predicting the hidden labels for each class with COSNet
> res <- apply(labels, 2, function(x, W, cost){
+         return(COSNet(W, x, cost))},
+         W = W, cost = 0.0001);
```

### 4.3 Result Evaluation

The function `COSNet` returns a list with five members, including binary predictions, ranking scores, and learned parameters. We now show how to compute, for instance, the AUC and the P10R achieved for the first GO term. Moreover, we show the value learned for the model parameters.

```
> library(PerfMeas);
> ## last predicted term
> term.ind <- 6;
> scores <- res[[term.ind]]$scores;
> test.genes <- names(scores);
> test.labels <- as.vector(GO.ann.sel[test.genes, term.ind]);
> pos.labels <- sum(test.labels > 0)
> pos.labels

[1] 102
```

```
> alpha <- res[[term.ind]]$alpha
> gamma <- res[[term.ind]]$c
> alpha
```

```
[1] 1.489928
```

```
> gamma
```

```
[1] 0.005507229
```

```
> AUC <- AUC.single(scores, test.labels)
> AUC
```

```
[1] 0.801313
```

```
> P10R <- precision.at.recall.level(scores, test.labels,
+ rec.level = 0.1)
> P10R
```

```
[1] 0.6470588
```

# 5 Usage of *COSNet* for Predicting Therapeutical Categories of Drugs

This section shows an application of *COSNet* in predicting drugs categories from DrugBank for 1253 DrugBank drugs. The drug similarity matrix and the corresponding labels are available in the R package *bionetdata*, and contain respectively the Tanimoto chemical structure similarity scores among the considered drugs and the 0/1 labels for 45 drug categories, where in postition $i, j$-th we have the value 1 whether the drug $i$ is associated with the drug category $j$, 0 otherwise.

## 5.1 Loading the data

```
> library(bionetdata);
> ## similarity matrix DD.chem.data
> data(DD.chem.data);
> ## label matrix DrugBank.Cat
> data(DrugBank.Cat);
```

## 5.2 Learning the Model Parameters

We show now how to learn the *COSNet* parameters using the package function `optimizep`. Fixed one drug category, first of all, we determine a random stratified partition of the data using the function `find.division.strat` provided by the *COSNet* package, and then hide the labels of one fold and use the other folds as training set.

```
> n <- nrow(DD.chem.data);
> drugs <- rownames(DD.chem.data);
> drug.category <- c("Cephalosporins");
> labels <- as.vector(DrugBank.Cat[, drug.category]);
> names(labels) <- rownames(DrugBank.Cat);
> ## Determining a random partition in 5 folds ensuring that each
> ##    fold has a similar proportion of positives
> folds <- find.division.strat(labels, 1:n, 5)
> labels[labels <= 0] <- -1;
> ## hiding the test labels (the fold of index 1)
> test.drugs <- folds[[1]];
> training.drugs <- setdiff(1:n, test.drugs);
> labels[test.drugs] <- 0;
```

Now we need to project the training nodes into a bidimensional space, step necessary for the learning procedure (Frasca *et al. 2013*), and we can do it using the package function `generate_points`, which returns a list with two fields: *pos_vect*, the named vector of abscissae of projected points and *neg_vect*, the named vector of ordinates of projected points. Then we can call the function `optimizep` to learn the parameters of the model.

```
> points <- generate_points(DD.chem.data, test.drugs, labels);
> str(points)

List of 2
 $ pos_vect: num [1:1253, 1] 6.71 7.41 3.97 6.22 1.34 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1253] "DB00115" "DB00116" "DB00117" "DB00118" ...
  .. ..$ : NULL
 $ neg_vect: num [1:1253, 1] 211 246 190 188 106 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1253] "DB00115" "DB00116" "DB00117" "DB00118" ...
  .. ..$ : NULL

> opt_parameters <- optimizep(points$pos_vect[training.drugs],
+                 points$neg_vect[training.drugs], labels[training.drugs]);
```

### 5.3 Running the Sub-network of Unlabeled Nodes to Infer Labels

We now extend the learned parameters to the sub-network of unlabeled nodes and run it with the package function `runSubnet`.

```
> ## alpha parameter
> alpha <- opt_parameters$alpha;
> ## gamma parameter
> gamma <- opt_parameters$c;
```

```
> ## optimal F-score achieved during learning phase
>     ## procedure (see Frasca et al. 2013)
> Fscore <- opt_parameters$Fscore;
> res <- runSubnet(DD.chem.data, labels, alpha, gamma, cost=0.035);
```

### 5.4 Extracting Predictions

The output of the `runSubnet` function is a list with three fields: "state" which is a named vector containing the binary predictions; "scores", named vector containing the scores described in Eq. (1); "iter", integer representing the network iterations needed to reach the fixed state. By means of this vectors, we can now compute the prediction performance.

```
> library(PerfMeas)
> str(res)

List of 3
 $ state : Named num [1:251] -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
  ..- attr(*, "names")= chr [1:251] "DB00122" "DB00128" "DB00142" "DB00148" ...
 $ scores: Named num [1:251] -6.18 -8.47 -9.22 -6.53 -9.23 ...
  ..- attr(*, "names")= chr [1:251] "DB00122" "DB00128" "DB00142" "DB00148" ...
 $ iter  : num 3

> res$iter

[1] 3

> labels <- as.vector(DrugBank.Cat[, drug.category]);
> names(labels) <- rownames(DrugBank.Cat);
> test.names <- names(res$scores);
> AUC <- AUC.single(res$scores, labels[test.names]);
> AUC;

[1] 1

> P10R <- precision.at.recall.level(res$scores,
+         labels[test.names], rec.level=0.1);
> P10R;

[1] 1

> Fs <- F.measure.single(res$state, labels[test.names]);
> Fs

        P         R         S         F         A      Pos.
0.8571429 1.0000000 0.9959184 0.9230769 0.9960159 6.0000000
```

### 5.5 Cross Validating *COSNet* in Predicting Drug Therapeutical Categories

```
> library(bionetdata);
> data(DD.chem.data);
> data(DrugBank.Cat);
> labels <- DrugBank.Cat;
> labels[labels <= 0] <- -1;
> out <- cosnet.cross.validation(labels, DD.chem.data,
+         5, cost=0.035);
> Fs <- F.measure.single.over.classes(labels, out$predictions);
> Fs$average[4];

        F
0.3767213

> labels[labels <= 0] <- 0;
> auc <- AUC.single.over.classes(labels, out$scores);
> auc$average

[1] 0.8166223

> PXR <- precision.at.multiple.recall.level.over.classes(labels,
+         out$scores, seq(from=0.1, to=1, by=0.1));
> PXR$avgPXR

       0.1        0.2        0.3        0.4        0.5        0.6        0.7
0.60564369 0.58635681 0.51374627 0.44636801 0.35913316 0.24047006 0.18846460
       0.8        0.9          1
0.16449520 0.11383054 0.05890262
```

## References


Bertoni, A., Frasca, M., Valentini, G. (2011). COSNet: A Cost Sensitive Neural Network for Semi-supervised Learning in Graphs. *ECML/PKDD (1)* **6911**, 219-234.

Frasca, M., Bertoni, A., and Valentini, G. (2013). A neural network algorithm for semi-supervised node label learning from unbalanced data. *Neural Networks*, **43**(0), 84 – 98.

Von Mering, C., Krause, R., Snel, B., Cornell, M., Oliver, S., Fields, S., and Bork, P. (2002). Comparative assessment of large-scale data sets of protein-protein interactions. *Nature* **417** 399-403.

Ashburner, M. et al. (2000), Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics* **25** (1) 25–29.

Ling, C. X. and Sheng, V. S. (2007). Cost-sensitive Learning and the Class Imbalanced Problem.