Inferring mutual information networks using the *minet* package

March 3, 2008

1 Introduction

The *minet* package allows the user to infer mutual information networks from data. Each node in the network corresponds to a feature and the higher the weight of a link between two nodes, the higher the confidence in the existence of an interaction between the two features. In order to evaluate the interaction between two features, the mutual information is used as a dependence measure. The inference proceeds in two steps. First we compute the mutual information matrix (MIM), a square matrix where MIM_{ij} is the mutual information between variables X_i and X_j. In the second step, we apply an algorithm to the MIM in order to compute a score that will be used as the weight between the network nodes. The following sections give examples on how to use the package in order to:

- discretize data.
- estimate the mutual information matrix.
- infer a network.
- validate infered networks.

2 Inference

2.1 Mutual Information Matrix Estimation

```
> library(minet)
> data(syn.data)
> disc.method <- "equalwidth"
> nbins <- sqrt(nrow(syn.data))
> data <- disc(syn.data, disc.method, nbins)
> estimator = "empirical"
> mim <- build.mim(data, estimator)
> mim[1:5, 1:5]
```

```
             CDC11     SWI4    CDC10     SPT16 SWI4_SWI6
CDC11     0.000000 1.537586 1.486108 0.5729100 1.5197844
SWI4      1.537586 0.000000 1.548425 0.6907569 1.6579561
CDC10     1.486108 1.548425 0.000000 0.6252061 1.4391632
SPT16     0.572910 0.690757 0.625206 0.0000000 0.6665617
SWI4_SWI6 1.519784 1.657956 1.439163 0.6665617 0.0000000
```

In the above code, the mutual information matrix is built using the function `build.mim`. This function takes as argument the dataset and the mutual information estimator[1]. Depending on the estimator used, the data must contain discrete values. In order to discretize the data the function `disc` can be used. This function allows the user to choose between two binning algorithms[2].

2.2 Obtaining The Network

Once the MIM is computed, the network is obtained using one of the following functions : `clr.net`, `aracne.net`, `mr.net`, which all take as argument the mutual information matrix :

```
> net <- mr.net(mim)
> net[1:5, 1:5]
```

```
              CDC11       SWI4      CDC10      SPT16  SWI4_SWI6
CDC11     0.0000000 0.17608455 0.28114680 0.00000000 0.08705329
SWI4      0.1760845 0.00000000 1.54842535 0.07968971 1.65795608
CDC10     0.2811468 1.54842535 0.00000000 0.02801786 0.44685669
SPT16     0.0000000 0.07968971 0.02801786 0.00000000 0.02280844
SWI4_SWI6 0.0870533 1.65795608 0.44685669 0.02280844 0.00000000
```

The returned value is the weighted adjacency matrix of the network.

2.3 The `minet` function

All the above code can be summarized with the `minet` function:

```
> library(minet)
> data(syn.data)
> net <- minet(syn.data, method = "mrnet", estimator = "empirical",
+     disc.method = "equalwidth", nbins = sqrt(nrow(syn.data)))
> net[1:5, 1:5]
```

```
               CDC11       SWI4      CDC10      SPT16  SWI4_SWI6
CDC11     0.00000000 0.07783305 0.12427277 0.00000000 0.03847938
SWI4      0.07783305 0.00000000 0.68443641 0.03522452 0.73285129
```

[1]Four estimators (`"empirical"`, `"gaussian"`, `"millermadow"` and `"shrink"`) are implemented

[2]Equal frequencies and equal width algorithms

```
CDC10     0.12427277 0.68443641 0.00000000 0.01238448 0.19752001
SPT16     0.00000000 0.03522452 0.01238448 0.00000000 0.01008181
SWI4_SWI6 0.03847938 0.73285129 0.19752001 0.01008181 0.00000000
```

Note that in this case the returned object is the *normalized* weighted adjacency matrix of the network (i.e. the values range from 0 to 1).

3 Validation

3.1 Obtaining Confusion Matrices

```
> library(minet)
> data(syn.data)
> data(syn.net)
> net <- minet(syn.data)
> table <- validate(net, syn.net, steps = 20)
> table[1:10, ]
```

```
   thrsh  tp   fp   tn fn
1   0.00 130 2370    0  0
2   0.05  76  212 2158 54
3   0.10  62  146 2224 68
4   0.15  62  112 2258 68
5   0.20  58   82 2288 72
6   0.25  58   70 2300 72
7   0.30  54   58 2312 76
8   0.35  50   40 2330 80
9   0.40  42   24 2346 88
10  0.45  42   22 2348 88
```

In the above code, the `validate` function compares the infered network `net` to `syn.net`, the network underlying `syn.data`. Note that the true underlying network has to be a boolean matrix. For each of the `steps` threshold values T, the edges whose weight are (strictly) below T are eliminated. All the other edges will have a weight 1. Each of the `steps` resulting graph is compared to the true underlying network in order to get `steps` confusion matrices. The obtained confusion matrices are appended to a data frame which will be the returned object of the `validate` function.

Note that the `validate` function distinguishes the following cases:

- Both networks are oriented
- Both networks are unoriented
- One of the network is oriented and the other unoriented

In the third case, the oriented network will be considered unoriented.

3.2 Processing Confusion Matrices

```
> library(minet)
> data(syn.data)
> data(syn.net)
> net1 <- minet(syn.data, method = "mrnet")
> net2 <- minet(syn.data, method = "clr")
> table1 <- validate(net1, syn.net, steps = 50)
> table2 <- validate(net2, syn.net, steps = 50)
```

Once the dataframes `table1` and `table2` are computed, we can use the function

- `pr(table)` to obtain precisions and recalls.
- `rates(table)` to obtain true positive rates and false positive rates.
- `fscores(table,beta)` to obtain $F_\beta - scores$.

The functions `show.pr` and `show.roc` allow the user to plot PR-curves and ROC-curves respectively. Both functions return the device associated to the plotting window used:

```
> dev <- show.pr(table1, pch = 2, type = "b", col = "green")
> show.pr(table2, device = dev, pch = 1, type = "b", col = "blue")

postscript
         2

> dev <- show.roc(table1, type = "b", col = "green")
> show.roc(table2, device = dev, type = "b", col = "blue")

postscript
         3
```

In the above code, the functions take as argument the dataframe `table` returned by `validate`. The user can use the returned device to display several curves on the same plotting window.