Use Case 2

Štefan Lyócsa

Masaryk University

Goal

The goal is to improve credit model by using network variables. Instead of the dissimilarity matrix, we create a matrix using a **Factor Network** model approach.

- We create an adjacency matrix.
- We calculate vertex level network variables.
- We augment credit models with new variables.
- We compare the forecasting accuracy.

We **assume** that loan characteristics (size, duration, ...) are driven by some unobserved (**latent**) factors. We can think of these factors are variables that more accurately (with less noise) loan characteristics. Let n by the number of loans, p number of loan characteristics. $\boldsymbol{X}$ be a $n \times p$ matrix, and k an arbitrary number of factors. The factor representation of $\boldsymbol{X}$ is:

$$\boldsymbol{X} = \boldsymbol{F}\boldsymbol{W} + \boldsymbol{\epsilon}$$

Here $\boldsymbol{F}$ is a $n \times k$ matrix of factors, $\boldsymbol{W}$ a $k \times p$ matrix of coefficients and $\boldsymbol{\epsilon}$ a $n \times p$ matrix of residuals. Factors and residuals are mutually independent and normally distributes.

Example

Let's have **three** loan characteristics, loan size, loan duration, interest rate ($p = 3$), and **two** underlying (unobserved) factors ($k = 2$). A loan with values of 1000 (size), 36 (duration), 15.5 (interest rate) could be decomposed into:

$$1000 = f_{1,1}w_{1,1} + f_{1,2}w_{2,1} + \epsilon_{1,1}$$
$$36 = f_{1,1}w_{1,2} + f_{1,2}w_{2,2} + \epsilon_{1,2}$$
$$15.5 = f_{1,1}w_{1,3} + f_{1,2}w_{2,3} + \epsilon_{1,3}$$

Example

$$1000 = f_{1,1}w_{1,1} + f_{1,2}w_{2,1} + \epsilon_{1,1}$$
$$36 = f_{1,1}w_{1,2} + f_{1,2}w_{2,2} + \epsilon_{1,2}$$
$$15.5 = f_{1,1}w_{1,3} + f_{1,2}w_{2,3} + \epsilon_{1,3}$$

Instead of three variables, we have two (Factor 1 and 2). The factors are unique to each loan, while coefficients are common to each loan. We use the two factors to *re-construct* the three values of loan characteristics. **The more** we can re-construct with **as few** factors as possible, the better. Factors are estimated using *singular value decomposition* approach.

How to create a network? We are going to use previously estimated factors - instead of variables. Only if two loans have similar factor values they should be connected. Again, we hope that in the network **bad loans** are going to be clustered together.

Let's have a binary adjacency matrix $\boldsymbol{G} \in \{0,1\}^{n \times n}$, where:

$$g_{i,j} = 1 \text{ if } \left(\gamma_{i,j} = \Phi\left[\theta + (\boldsymbol{F}\boldsymbol{F}^T)_{i,j}\right] > \gamma\right) \text{ and } g_{i,j} = 0 \text{ otherwise}$$

where Φ is the cumulative distribution function of the normal distribution. We follow the previous literature and choose $\gamma = 0.10$ and $\theta = \Phi^{-1}(\frac{2}{N-1})$.

Create network in R

Import data again (but do not delete anything from previous Use Case). We arbitrarily select variables that we think might identify a bad loan:

- `X = DT[,c('int','durm','linctot','noliab')]`

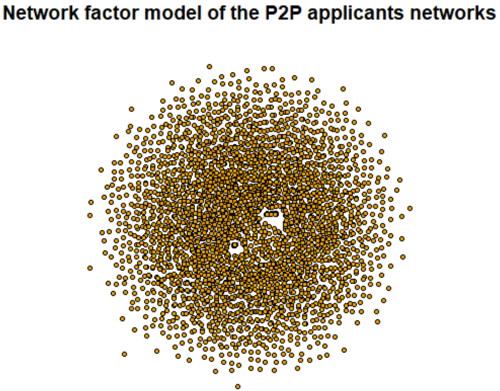
Run the following function:

- `AM = FN_SVD(X,p=0.75,gam=0.10)`
- `g = graph_from_adjacency_matrix(AM, mode = 'undirected', weighted = TRUE)`

We can visualize the Network Factor Model:

- `plot(g, graph = 'NFM', vertex.label=NA,vertex.size = 3, main = 'Network factor model of the P2P applicants networks')`

Create the network in R



- vertex degree,
- harmonic centrality,
- Community detection - Louvain method.

To address the issue of isolated vertices, one can assume that the shortest distance between vertex i and an isolated vertex j is ∞, while conveniently assuming that $1/\infty = 0$. Harmonic centrality is therefore:

$$H_(i) = \sum_{d_(i,j)<\infty, i\neq j} \frac{1}{d_(i,j)}$$

where $d_(i,j)$ is the shortest path from vertex i to vertex j in the network.

Estimating vertex level attributes in R

The following function calculates centrality and community:

- `NetDscr=BVC(g)`

Now add variable into the model:

- `DT$Deg = NetDscr$VCentrality[,1]`
- `DT$Hac = NetDscr$VCentrality[,2]`
- `DT = data.frame(DT,NetDscr$Community)`

Preparing data

Define the matrix of input and output variables:

- ```
  indep = as.matrix(DT[1:(N-NF),c('new','ver3','ver4','lf
  'undG','female','lamt','int','durm','educprim','educbasic
  'educvocat','educsec','msmar','msco','mssi','msdi','nrode
  'espem','esfue','essem','esent','esret','dures','exper',
  'linctot','noliab','lliatot','norli','noplo','lamountplo'
  'lamntplr','lamteprl','nopearlyrep','Deg','Hac',paste('g'
  ```
- ```
  dep = DT[1:(N-NF),'RR2']
  ```

Preparing data

- pred = as.matrix(DT[(N-NF+1):N,c('new','ver3','ver4','l
'undG','female','lamt','int','durm','educprim','educbasic
'educvocat','educsec','msmar','msco','mssi','msdi','nrode
'espem','esfue','essem','esent','esret','dures','exper',
'linctot','noliab','lliatot','norli','noplo','lamountplo'
'lamntplr','lamteprl','nopearlyrep','Deg','Hac',paste('g'
- ytrue = DT[(N-NF+1):N,'RR2']

LASSO model

Model estimation:

- `m3_L = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=1)`
- `coef(m3_L,s='lambda.1se')`

Forecast loan returns:

- `yhat = predict(m3_L,newx=pred,s=m3_L$lambda.1se)`

Calculate mean squared error:

- `LASSO_FN = mean((yhat-ytrue)`^{2}`)`

RIDGE model

Model estimation:

- `m3_R = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=0)`
- `coef(m3_R,s='lambda.1se')`

Forecast loan returns:

- `yhat = predict(m3_R,newx=pred,s=m3_R$lambda.1se)`

Calculate mean squared error:

- `RIDGE_FN = mean((yhat-ytrue)^2)`

Elastic net model

- `m3_E25 = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=0.25)`
- `m3_E50 = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=0.50)`
- `m3_E75 = cv.glmnet(x=indep,y=dep,nfolds=30,alpha=0.75)`

- `yhat = predict(m3_E25 ,newx=pred,s=m3_E25$lambda.1se)`
- `yhat = predict(m3_E50 ,newx=pred,s=m3_E50$lambda.1se)`
- `yhat = predict(m3_E75 ,newx=pred,s=m3_E75$lambda.1se)`

- EN25FN = mean((yhat-ytrue)^2)
- EN50FN = mean((yhat-ytrue)^2)
- EN75FN = mean((yhat-ytrue)^2)

Comparing forecast accuracy

Is network approach worth the strugle?

- `MSEs = c(OLS, LASSO, RIDGE, EN25, EN50, EN75, LASSO_N, RIDGE_N, EN25N, EN50N, EN75N, LASSO_FN, RIDGE_FN, EN25FN, EN50FN, EN75FN)`
- `names(MSEs) = c('OLS', 'LASSO', 'RIDGE', 'EN25', 'EN50', 'EN75', 'LASSO_N', 'RIDGE_N', 'EN25N', 'EN50N', 'EN75N', 'LASSO_FN','RIDGE_FN', 'EN25FN', 'EN50FN', 'EN75FN')`
- `MSEs = sort(MSEs)`
- `cbind(MSEs)`

Comparing forecast accuracy

Is the factor network approach worth the struggle?

	MSEs
EN75FN	848.6278
EN50FN	849.5642
RIDGE_FN	850.8609
LASSO_N	856.3854
EN25N	858.0379
LASSO_FN	859.8096
RIDGE_N	860.3076
EN50N	862.0838
EN75N	864.3380
EN25FN	864.3384
EN50	870.3685
LASSO	871.5219
EN75	874.5317
EN25	874.8249
RIDGE	929.6943
OLS	995.6180

Use Case 2

Štefan Lyócsa

Masaryk University