From macro to micro variables: underdetermination and network effects - how can machine learning techniques help

Alexander Denev

Joint work with:

Orazio Angelini

September 2016

The Context

- In certain practical applications we must assess the impact of a change of high level variables on a more granular structure of variables e.g.
  - > Calculating the effect of a shock to an index on a more granular portfolio
  - > Calculating the effect of a change of a macroeconomic variable on a network of companies
- We are facing the task of modelling both the exogenous shock effect on the network and the network endogenous effects

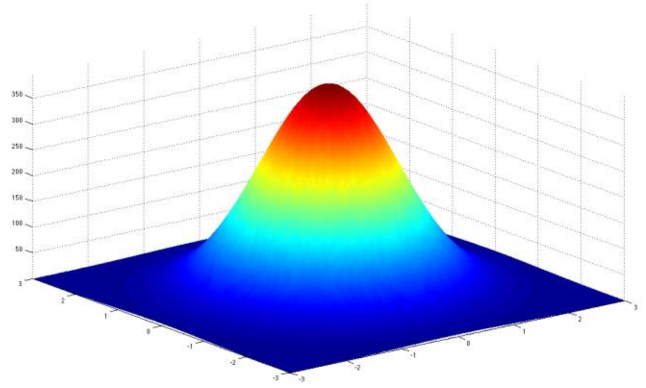
$$\begin{tikzcd}
\boxed{R_{S\&P500}} \arrow[r] & \boxed{\text{1,000 US Equities}}
\end{tikzcd}$$

Joint Distribution

- In several practical problems we have to deal with more than one variable
- We model the variables and their relationships through a *joint distribution*

Example: Bivariate Gaussian Distribution

$$p([X_{1,} \dots, X_n]) = \frac{1}{(2\pi)^{\frac{n}{2}}|\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu))$$



- Can we use a convenient visualisation to represent some of the properties of the joint distribution?

The precision matrix – what it is

- **Theorem** – Consider a Gaussian distribution $P(X_1, \ldots., X_n) = N(\mu, \Sigma)$, and let $Q = \Sigma^{-1}$ be the precision matrix. Then $Q_{i,j} = 0$ if and only if $P \Vdash (X_i \perp X_j | X_V - \{X_i, X_j\})$ where X_V is the set of all the variables in the graph
- Covariance matrix

$$\Sigma_{i,j} = 0 \;\Rightarrow\; X_i \perp X_j \; or \; p(X_i, X_j) = p(X_i)p(X_j)$$

- Precision matrix

$$Q_{ij} = 0 \;\Rightarrow\; X_i \perp X_j | X_{-ij} \; or \; p(X_i, X_j | X_{-ij}) = p(X_i | X_{-ij}) p(X_j | X_{-ij})$$

The precision matrix - example

$$\Sigma^{-1} = \begin{pmatrix} 1 & 6 & 0 & 0 & 0 \\ 6 & 2 & 7 & 0 & 0 \\ 0 & 7 & 3 & 8 & 0 \\ 0 & 0 & 8 & 4 & 9 \\ 0 & 0 & 0 & 9 & 5 \end{pmatrix} \Longleftrightarrow \Sigma = \begin{pmatrix} 0.10 & 0.15 & -0.13 & -0.08 & 0.15 \\ 0.15 & -0.03 & 0.02 & 0.01 & -0.03 \\ -0.13 & 0.02 & 0.10 & 0.07 & -0.12 \\ -0.08 & 0.01 & 0.07 & -0.04 & 0.07 \\ 0.15 & -0.03 & -0.12 & 0.07 & 0.08 \end{pmatrix}$$

$$\Sigma^{-1}_{15} = 0 \leftrightarrow X_1 \perp X_5 | X_2, X_3, X_4$$

$$X_1 \perp X_5 \overset{\nrightarrow}{\leftrightarrow} \Sigma_{15} = 0$$

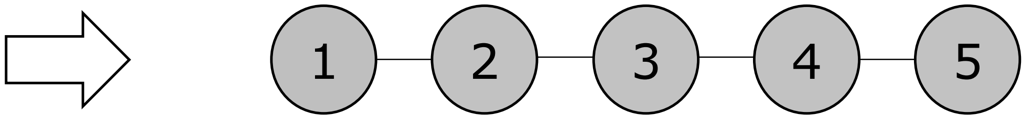
Gaussian Markov Networks

- If we start from a multivariate Gaussian we can cast in the form:

$$P(\boldsymbol{X}) \propto \prod_{i \neq j} \exp(-\frac{1}{2} X_i {\Sigma^{-1}}_{ij} X_j) \prod_{k} \exp(-\frac{1}{2} {\Sigma^{-1}}_{kk} X_k^2 + h_k X_k)$$

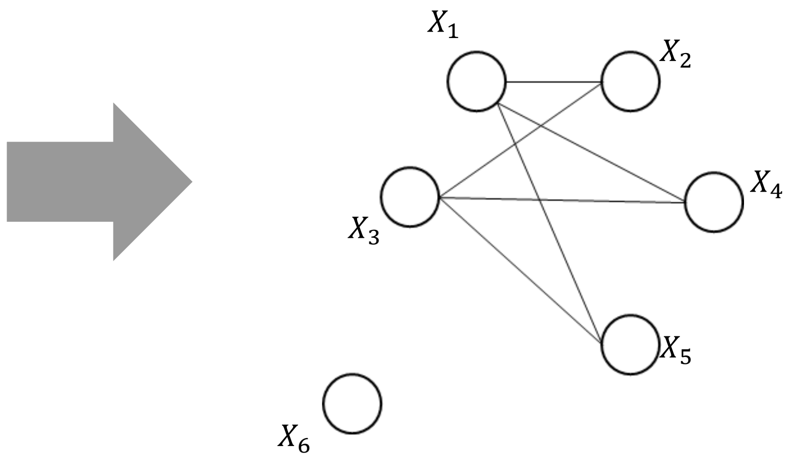
And associate a graphical model in which two nodes (variables) are not connected if the corresponding precision matrix element is 0

$$\Sigma^{-1} = \begin{pmatrix} 1 & 6 & 0 & 0 & 0 \\ 6 & 2 & 7 & 0 & 0 \\ 0 & 7 & 3 & 8 & 0 \\ 0 & 0 & 8 & 4 & 9 \\ 0 & 0 & 0 & 9 & 5 \end{pmatrix}$$



Gaussian Markov Networks - Example

$$\begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & 0 \\ * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$



Estimation

- Estimation methods
  - > **Covariance selection** – ill-posed when the covariance matrix is singular i.e. when the number of variables is larger than the number of samples $p \gg n$ i.e. **Big Data.**

  Ledoit (2004) and Ledoit (2012) propose 'shrinkage' methods for both the covariance and the precision matrices

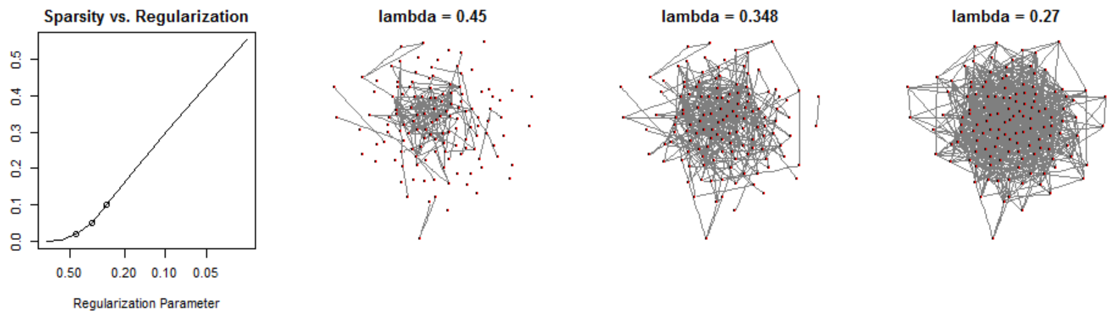
  - > L_1 **Regularization methods** – LASSO (Tibshirani (1996)), GLASSO (Banerjee (2007)) – applicable for $p \gg n$ by inducing sparsity

GLASSO - Introduction

- Idea: Consider a set of data with multivariate normality. We want to estimate a Sparse Precision Matrix Q that provides a Maximum Likelihood Estimate for

$$\frac{|Q|^{1/2}}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}(x-\mu)^T Q(x-\mu)\right) - \lambda Q$$

- The parameter λ is a constraining parameter that forces some coefficients to be zero thus enforcing sparsity.
- Friedman (2007) finds that it is computationally more efficient to estimate a Sparse Covariance matrix W using a three step iterative algorithm and then inverting it.
- Convergence is guaranteed based on the Coordinate Descent Methods of Tseng (2001).

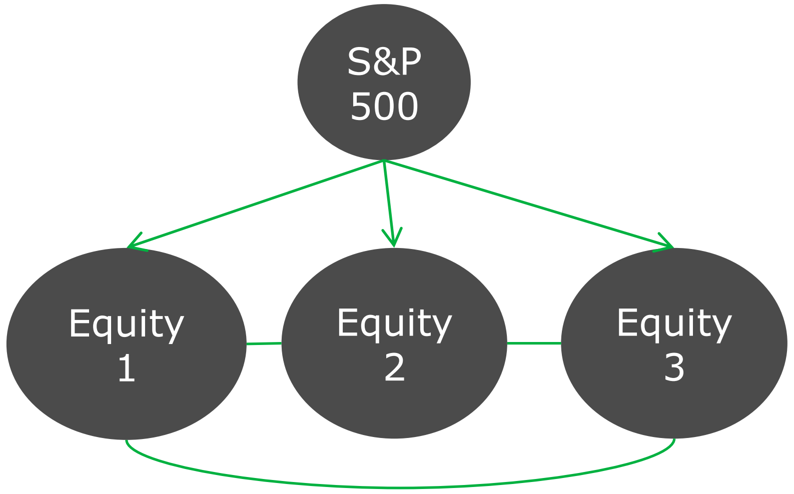


Network Effects – What are they?

- The presence of network links between variables may be due to:
  - > Omitted observable macro factors
  - > Omitted non-observable factors
  - > Idiosyncratic relationships

Chain Graphs

- Let G =(V,E) be a mixed graph with finite vertex set V and an edge set E that may contain two types of edges, namely directed (u→v) and undirected (u-v) edges
- The graph G is called a chain graph if it does not contain any semi-directed cycles, that is, it contains no path from v to v with at least one directed edge such that all directed edges have the same orientation



Chain Graphs

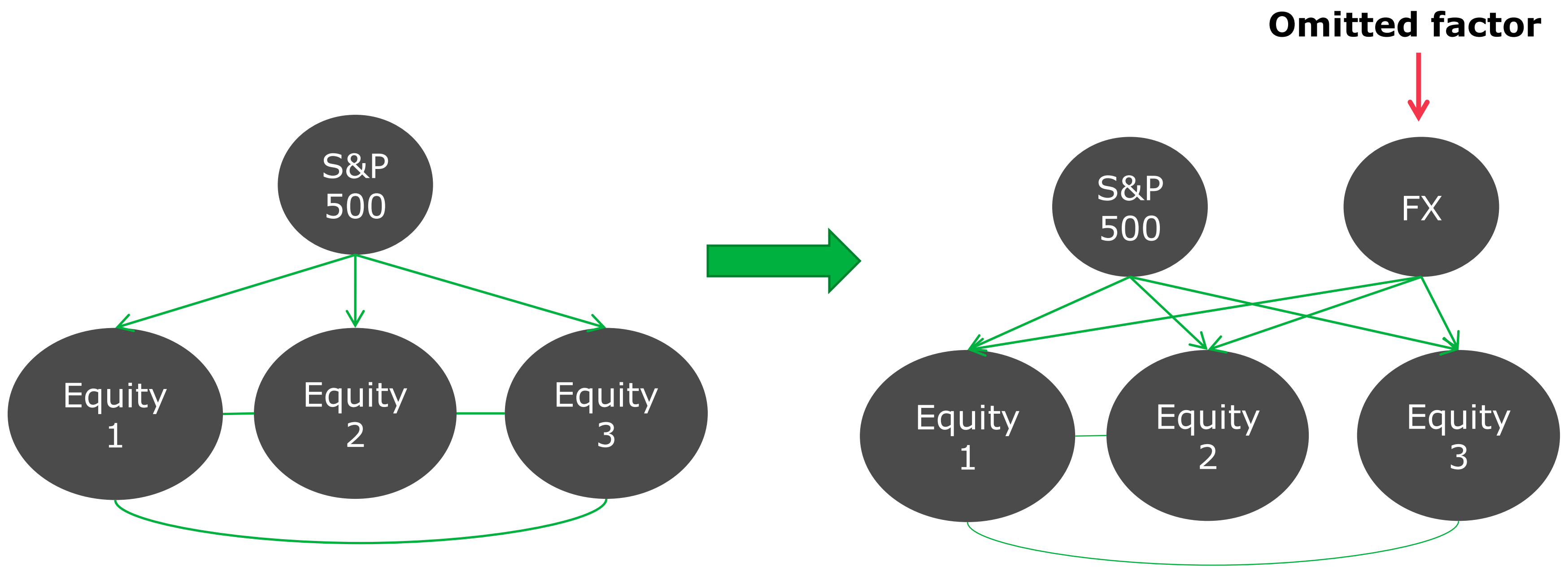
- A Chain Graph represents a Multivariate Gaussian which can be decomposed in recursive form.
- For example, for the chain graph of the previous slide

$$
\left\{
\begin{array}{l}
R_{S\&P500} = E[R_{S\&P500}] + \varepsilon_{S\&P500} \\
R_{Equity1} = \beta_{Equity1} R_{S\&P500} + \varepsilon_{Equity1} \\
R_{Equity2} = \beta_{Equity2} R_{S\&P500} + \varepsilon_{Equity2} \\
R_{Equity3} = \beta_{Equity3} R_{S\&P500} + \varepsilon_{Equity3} \\
\boxed{cov(\varepsilon_{Equityi}, \varepsilon_{Equityj}) \neq 0}
\end{array}
\right.
$$

Network effects

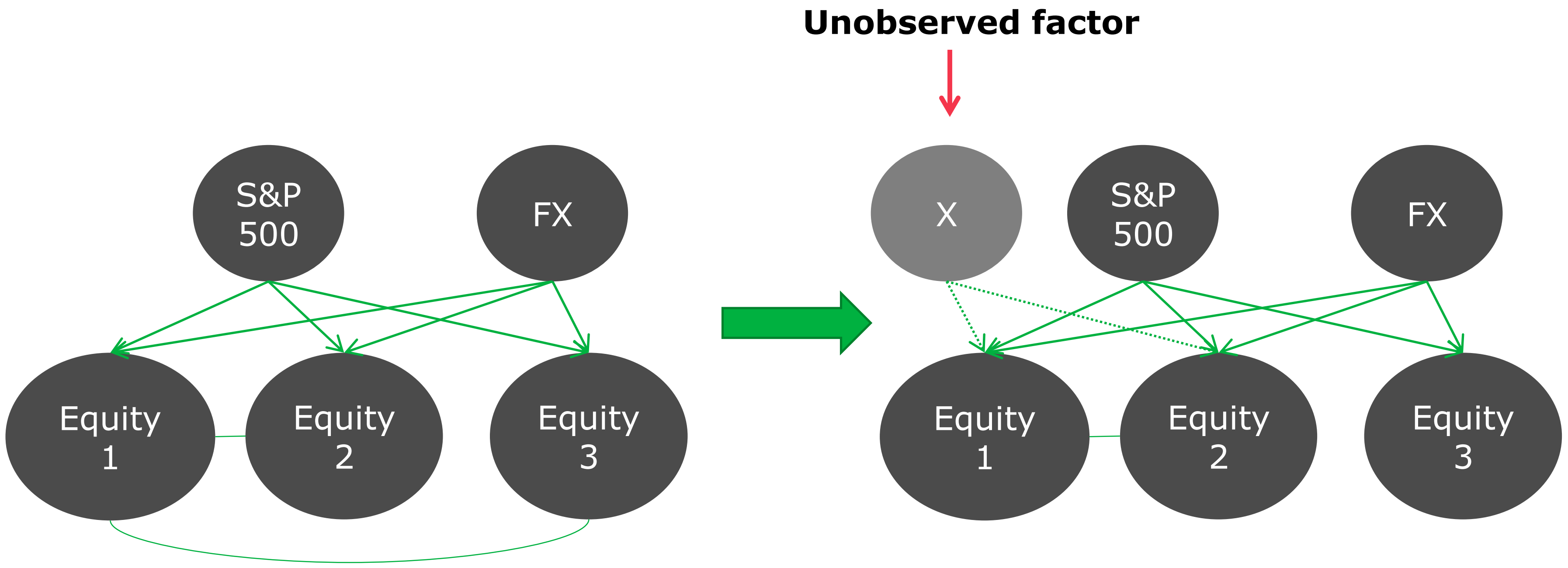
Network Effects – What are they?

- Inserting an extra factor can explain some of the links away



Network Effects – What are they?

- An unobserved factor can also remove links



Chain Graphs - Estimation

- We decompose the estimation of the Chain Graph in two steps
  1. Estimation of the loadings on the macro factor(s)
  2. Estimation of the network

- Two steps estimation procedure (Drton (2006), McCarter (2014))

The task

- **Task**: estimate the impact of a change of a variable on a balance sheet e.g. $R_{S\&P500} = -10\%$ over the next quarter

$$\begin{tikzcd}
\boldsymbol{\Omega} & \boldsymbol{\Omega_g} \\
\boxed{R_{S\&P500}} \arrow[r] & \boxed{\text{1,000 US Equities}}
\end{tikzcd}$$

In the end we want to obtain a distribution $P(\Omega_g|\Omega)$

Perturbations and their effect

- Perturbing a factor that feeds in the network and reading the results

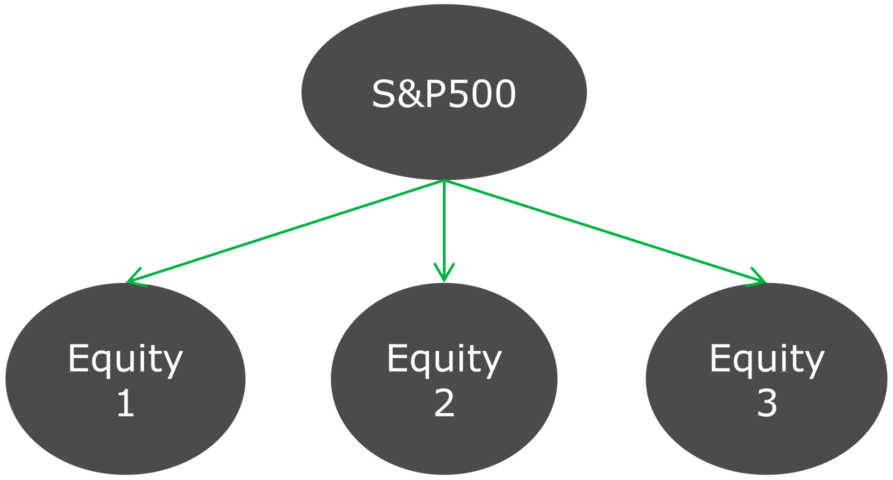
$$\left\{\begin{array}{l} R_{S\&P500} = x \quad \longleftarrow \quad \textbf{We fix this} \\ R_{Equity1} = \beta_{Equity1} x + \varepsilon_{Equity1} \\ R_{Equity2} = \beta_{Equity2} x + \varepsilon_{Equity2} \\ R_{Equity3} = \beta_{Equity3} x + \varepsilon_{Equity3} \\ \boxed{cov(\varepsilon_{Equityi}, \varepsilon_{Equityj}) \neq 0} \end{array}\right.$$

Under-determination of the task

- The distribution $P(\Omega_g|\Omega)$ will depend on the choices the modeller is faced with when structuring the task with regards to:
  - The *variables* to use
  - The *structure* of the relationships between the variables
  - The *parameters* behind the structure
- In the end different ways to structure the task will lead to a different distribution $P'(\Omega_g|\Omega)$, $P''(\Omega_g|\Omega)$….

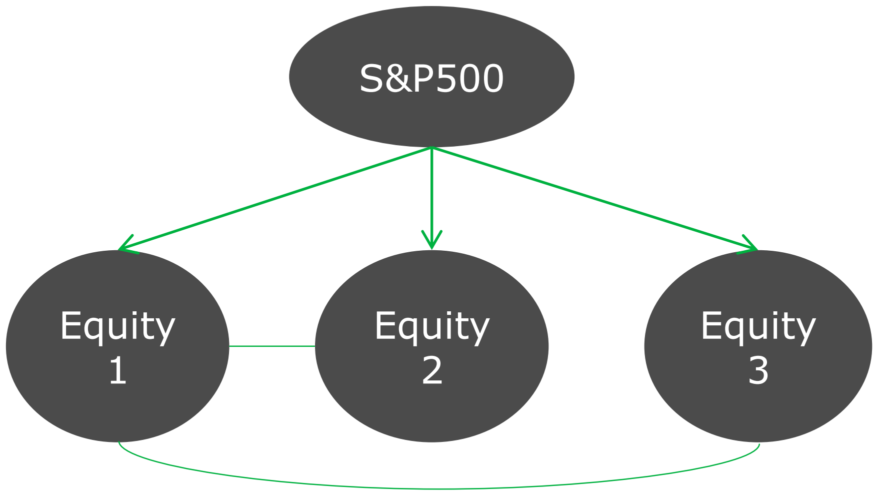
Under-determination of the task

- **First approach**: expand the shock directly to the stocks



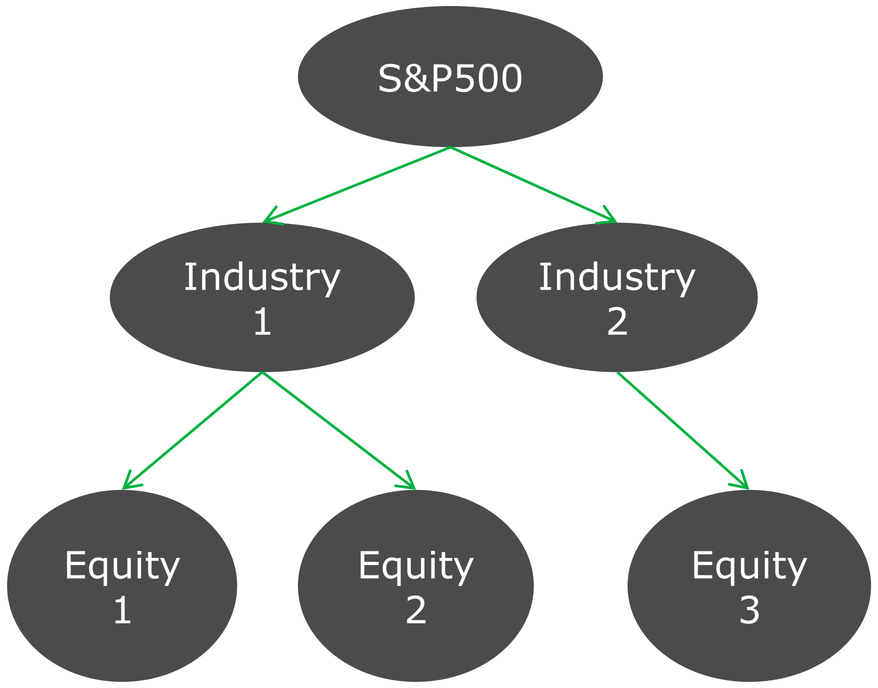
Under-determination of the task

- **Second possible approach**: expand the shock directly to the stocks by introducing network effects



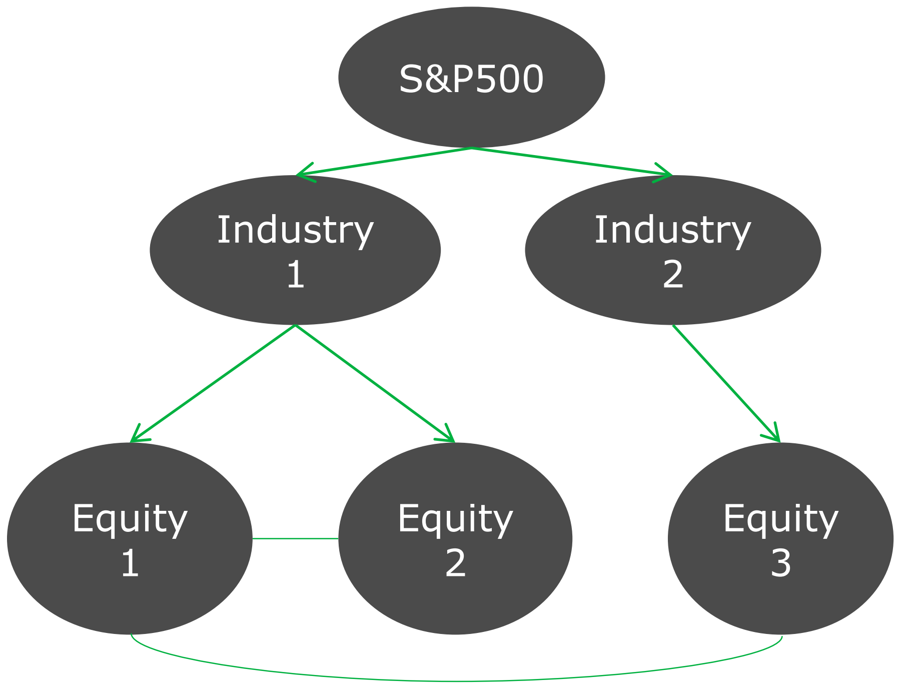
Under-determination of the task

- **Third possible approach**: expand the shock by passing through 1 intermediate layer of industry indices



Under-determination of the task

- **Forth possible approach**: expand the shock by passing through 1 intermediate layer of industry indices and by adding *network effects* in the last layer



Automatic selection

There are roughly speaking three approaches to automatic learning:

1. **Constrained based** – it views a structure as a set of independence relationships. The search algorithm tests for conditional dependencies and independencies in the data and hence learns the structure that best explains it.

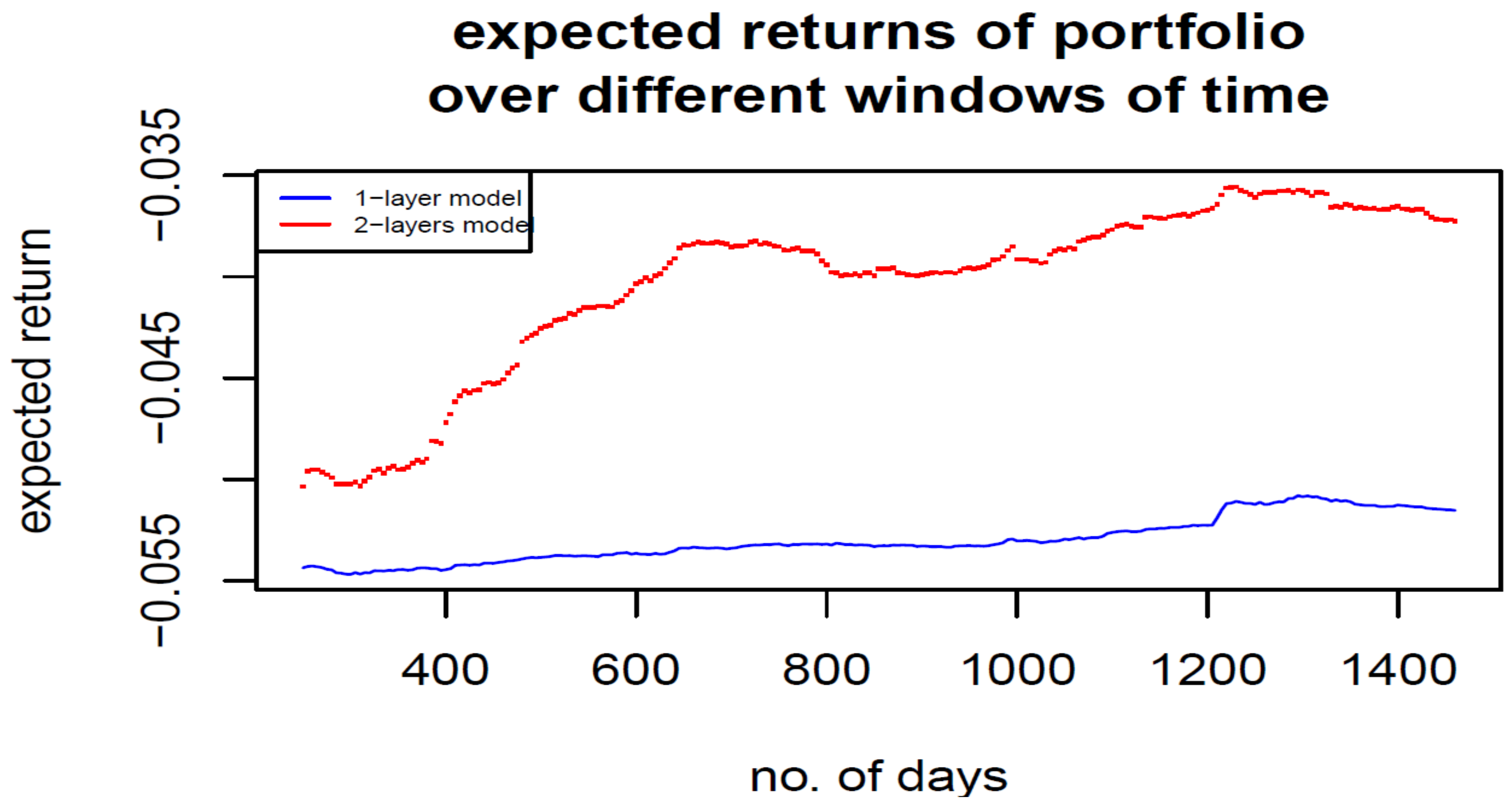
2. **Score based** – a hypothesis space is defined, that is a set of candidate structures, and a scoring function that measures how well the models fit the data. The learning is addressed as a model selection problem. The computational task is to find the highest-scoring structure.

3. **Bayesian model averaging** – it does not try to learn a single structure but an ensemble of them and averages their predictions i.e.

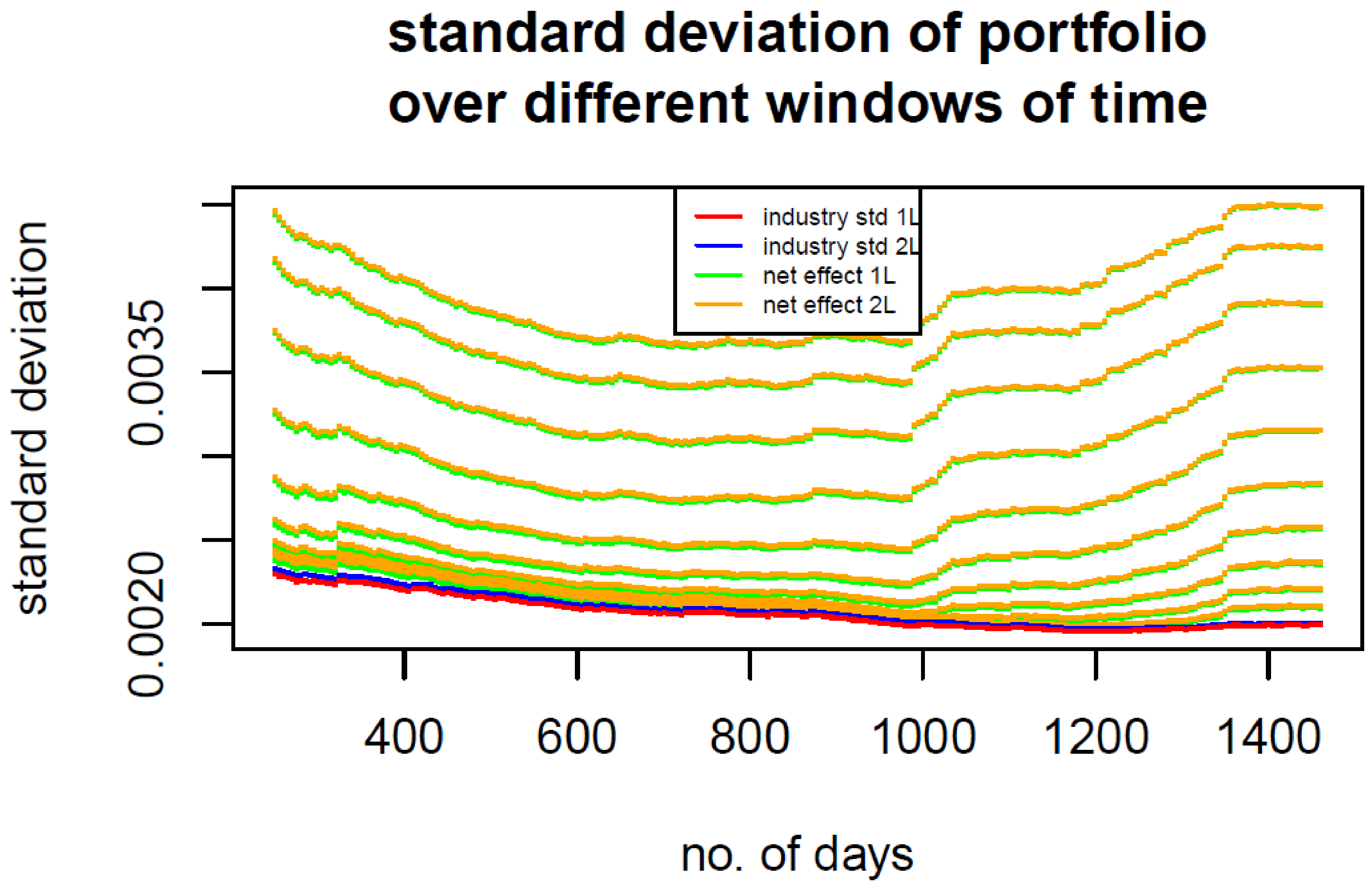
$$P(\Omega_g|\Omega_T) \propto \sum_i \int P(\Omega_g|\Omega_T, G_i, \Theta_i)P(\Theta_i|G_i, \Omega_T)c^{|G_i|}\, d\Theta_i$$

with $0 < c < 1$ and $|G_i|$ the number of edges in the i-th graph G_i

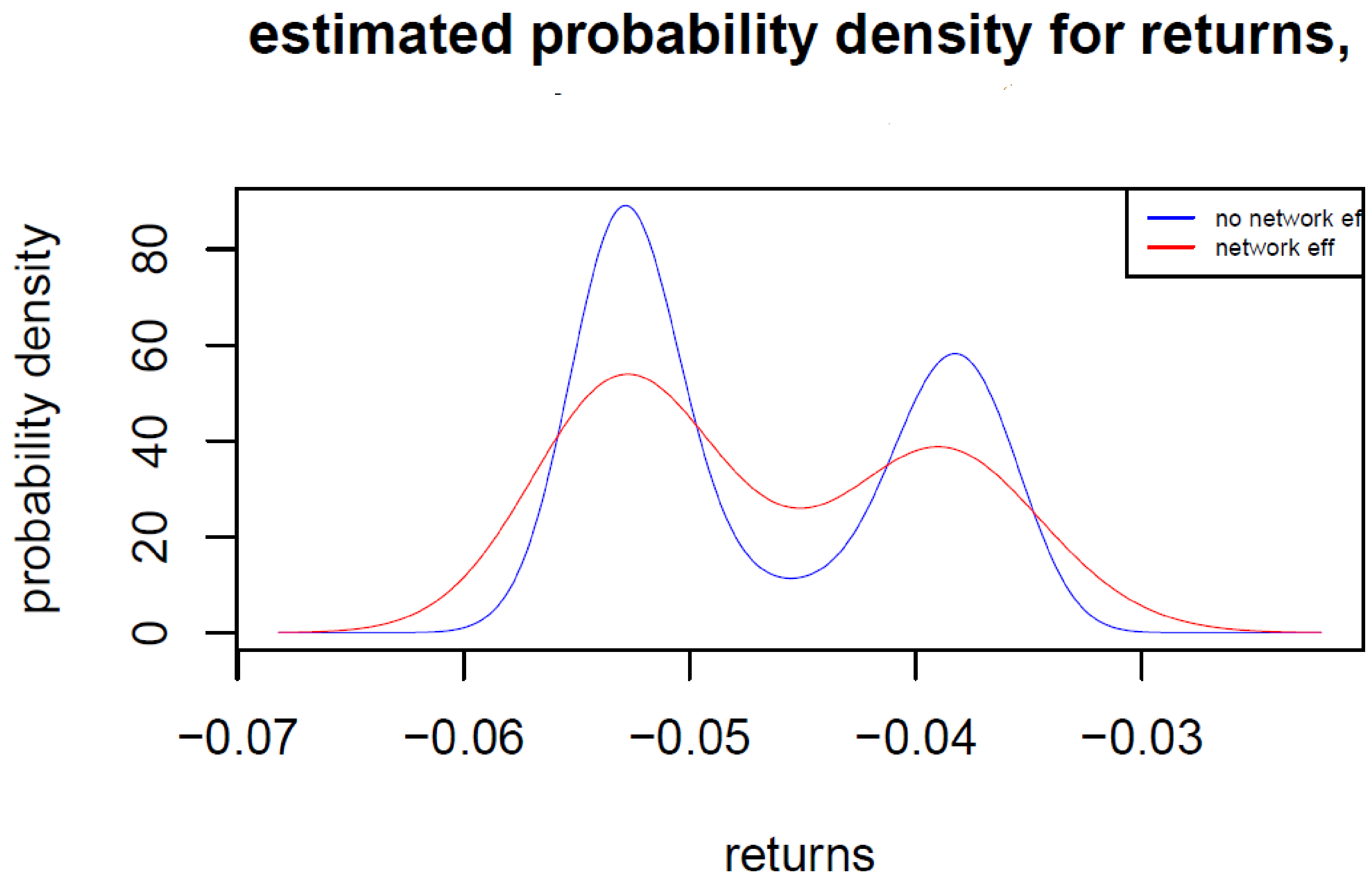
Results


expected returns of portfolio
over different windows of time
1−layer model
2−layers model
expected return
−0.055
−0.045
−0.035
400
600
800
1000
1200
1400
no. of days

Results


standard deviation of portfolio over different windows of time
standard deviation
0.0020
0.0035
400
600
800
1000
1200
1400
no. of days
industry std 1L
industry std 2L
net effect 1L
net effect 2L

Results


estimated probability density for returns,
no network ef
network eff
probability density
0
20
40
60
80
−0.07
−0.06
−0.05
−0.04
−0.03
returns