Fabrizio Lillo
Università di bologna & Scuola Normale Superiore
https://fabriziolillo.wordpress.com

DEEP NEURAL NETWORKS FOR THE ESTIMATION OF TIME SERIES MODELS

AlmaHAI: Hard Sciences, 28.4.20

Motivation

Motivation

- Neural Networks have been proved to be quite effective in forecasting complicated time series

Motivation

- Neural Networks have been proved to be quite effective in forecasting complicated time series
- Recurrent Neural Networks, Long Short Term Memory networks, Gated Recurrent Unit, Echo State Networks, etc.

Motivation

- Neural Networks have been proved to be quite effective in forecasting complicated time series
- Recurrent Neural Networks, Long Short Term Memory networks, Gated Recurrent Unit, Echo State Networks, etc.
- The idea is to have stored states and feedback loops (or time delays) to keep memory of the past

Motivation

- Neural Networks have been proved to be quite effective in forecasting complicated time series
- Recurrent Neural Networks, Long Short Term Memory networks, Gated Recurrent Unit, Echo State Networks, etc.
- The idea is to have stored states and feedback loops (or time delays) to keep memory of the past
- The advantage is to have a more flexible and less parametrised model wrt traditional time series models (e.g. ARMA)

Motivation

- Neural Networks have been proved to be quite effective in forecasting complicated time series
- Recurrent Neural Networks, Long Short Term Memory networks, Gated Recurrent Unit, Echo State Networks, etc.
- The idea is to have stored states and feedback loops (or time delays) to keep memory of the past
- The advantage is to have a more flexible and less parametrised model wrt traditional time series models (e.g. ARMA)
- However sometimes one is interested in estimating parameters of deterministic or stochastic time series models

Examples

Examples

- In Economics, Finance, Social Sciences, etc. one builds a structural model of a given system, which describes the choices of the agents and the interactions between them.

Examples

- In Economics, Finance, Social Sciences, etc. one builds a structural model of a given system, which describes the choices of the agents and the interactions between them.

- For example Agent Based Models are microscale models depending on a large number of parameters

Examples

- In Economics, Finance, Social Sciences, etc. one builds a structural model of a given system, which describes the choices of the agents and the interactions between them.

- For example Agent Based Models are microscale models depending on a large number of parameters

- These parameters are unobservable or latent: risk aversion, memory scale of agents, interaction between agents, expectations of agents, etc

Examples

- In Economics, Finance, Social Sciences, etc. one builds a structural model of a given system, which describes the choices of the agents and the interactions between them.

- For example Agent Based Models are microscale models depending on a large number of parameters

- These parameters are unobservable or latent: risk aversion, memory scale of agents, interaction between agents, expectations of agents, etc

- Inferring these parameters (for example from the time series generated by the model) from empirical data is important, also for obtaining calibrated models on which policy experiments are run

Examples

- In Economics, Finance, Social Sciences, etc. one builds a structural model of a given system, which describes the choices of the agents and the interactions between them.
- For example Agent Based Models are microscale models depending on a large number of parameters
- These parameters are unobservable or latent: risk aversion, memory scale of agents, interaction between agents, expectations of agents, etc
- Inferring these parameters (for example from the time series generated by the model) from empirical data is important, also for obtaining calibrated models on which policy experiments are run
- More generally, inferring interactions from correlation (physics, neuroscience, social science, etc).

Motivation

Motivation

- Suppose data X are generated by a model $\mathscr{M}$ with parameter θ whose prior is $\pi(\theta)$

Motivation

- Suppose data X are generated by a model $\mathscr{M}$ with parameter θ whose prior is $\pi(\theta)$
- If the likelihood function $p(X|\theta)$ is available, the posterior distribution of θ given observed data x_{obs} can be computed via Bayes' rule

$$\pi(\theta|x_{obs}) = \frac{\pi(\theta)p(x_{obs}|\theta)}{p(x_{obs})}$$

Motivation

- Suppose data X are generated by a model $\mathscr{M}$ with parameter θ whose prior is $\pi(\theta)$
- If the likelihood function $p(X|\theta)$ is available, the posterior distribution of θ given observed data x_{obs} can be computed via Bayes' rule

$$\pi(\theta|x_{obs}) = \frac{\pi(\theta)p(x_{obs}|\theta)}{p(x_{obs})}$$

- From the knowledge of the posterior $\pi(\theta|x_{obs})$ we can estimate θ via argmax or expectation $\mathbb{E}_\pi[\theta|X]$

Motivation

- Suppose data X are generated by a model $\mathscr{M}$ with parameter θ whose prior is $\pi(\theta)$

- If the likelihood function $p(X|\theta)$ is available, the posterior distribution of θ given observed data x_{obs} can be computed via Bayes' rule

$$\pi(\theta|x_{obs}) = \frac{\pi(\theta)p(x_{obs}|\theta)}{p(x_{obs})}$$

- From the knowledge of the posterior $\pi(\theta|x_{obs})$ we can estimate θ via argmax or expectation $\mathbb{E}_{\pi}[\theta|X]$

- The likelihood function must be known in closed form or sampled in MCMC schemes

Motivation

- Suppose data X are generated by a model $\mathscr{M}$ with parameter θ whose prior is $\pi(\theta)$
- If the likelihood function $p(X|\theta)$ is available, the posterior distribution of θ given observed data x_{obs} can be computed via Bayes' rule

$$\pi(\theta|x_{obs}) = \frac{\pi(\theta)p(x_{obs}|\theta)}{p(x_{obs})}$$

- From the knowledge of the posterior $\pi(\theta|x_{obs})$ we can estimate θ via argmax or expectation $\mathbb{E}_\pi[\theta|X]$
- The likelihood function must be known in closed form or sampled in MCMC schemes
- What can we do if the likelihood is not known in closed form, but we can simulate the model $\mathscr{M}$ given θ?

Approximate Bayesian Computation

Jiang et al. 2017

Approximate Bayesian Computation

Algorithm 1 ABC rejection sampling 1

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π(θ)
        Draw X' ~ M given θ'
    until X' = x_obs (acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

Jiang et al. 2017

Approximate Bayesian Computation

Algorithm 1 ABC rejection sampling 1

---

for $i = 1, \ldots, n$ **do**
  repeat
    Propose $\theta' \sim \pi(\theta)$
    Draw $X' \sim \mathcal{M}$ given θ'
  until $X' = x_{obs}$ (**acceptance criterion**)
  Accept θ' and let $\theta^{(i)} = \theta'$
end for

- **However when $x_{obs} \in \mathbb{R}^p$, the event $X' = x_{obs}$ has probability 0, and hence Algorithm 1 is unable to produce any draws.**

Jiang et al. 2017

Approximate Bayesian Computation

Algorithm 1 ABC rejection sampling 1

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π(θ)
        Draw X' ~ M given θ'
    until X' = x_obs (acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

- **However when $x_{obs} \in \mathbb{R}^p$, the event $X' = x_{obs}$ has probability 0, and hence Algorithm 1 is unable to produce any draws.**
- **One can introduce a low-dimensional summary statistic S and use the following**

Jiang et al. 2017

Approximate Bayesian Computation

Algorithm 1 ABC rejection sampling 1

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π(θ)
        Draw X' ~ M given θ'
    until X' = x_obs (acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

- However when $x_{obs} \in \mathbb{R}^p$, the event $X' = x_{obs}$ has probability 0, and hence Algorithm 1 is unable to produce any draws.
- One can introduce a low-dimensional summary statistic S and use the following

Algorithm 2 ABC rejection sampling 2

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π
        Draw X' ~ M with θ'
    until ||S(X') − S(x_obs)|| < ε (relaxed acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

Jiang et al. 2017

Approximate Bayesian Computation

Algorithm 1 ABC rejection sampling 1

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π(θ)
        Draw X' ~ M given θ'
    until X' = x_obs (acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

- **However when $x_{obs} \in \mathbb{R}^p$, the event $X' = x_{obs}$ has probability 0, and hence Algorithm 1 is unable to produce any draws.**
- **One can introduce a low-dimensional summary statistic S and use the following**

Algorithm 2 ABC rejection sampling 2

```
for i = 1, ..., n do
    repeat
        Propose θ' ~ π
        Draw X' ~ M with θ'
    until ||S(X') − S(x_obs)|| < ϵ (relaxed acceptance criterion)
    Accept θ' and let θ^(i) = θ'
end for
```

$$\pi(\theta \,|\, \|S(X') - S(x_{obs})\| < \epsilon) \approx \pi(\theta \,|\, x_{obs})$$

Jiang et al. 2017

Deep Neural Networks for Approximate Bayesian Computation

Jiang et al. 2017

Deep Neural Networks for Approximate Bayesian Computation

- **Idea**: Automatically learn summary statistics for high-dimensional X by using **Deep Neural Networks** (DNN) which is expected to effectively learn a good approximation to the posterior $\mathbb{E}_{\pi}[\theta \mid X]$

Jiang et al. 2017

Deep Neural Networks for Approximate Bayesian Computation

- **Idea**: Automatically learn summary statistics for high-dimensional X by using **Deep Neural Networks** (DNN) which is expected to effectively learn a good approximation to the posterior $\mathbb{E}_\pi[\theta \mid X]$

- The minimization problem is

$$\min_\beta \frac{1}{N} \sum_{i=1}^{N} \|f_\beta(X^{(i)}) - \theta^{(i)}\|_2^2$$

where f_β denotes a DNN with parameter β.

Jiang et al. 2017

Deep Neural Networks for Approximate Bayesian Computation

- **Idea**: Automatically learn summary statistics for high-dimensional X by using **Deep Neural Networks** (DNN) which is expected to effectively learn a good approximation to the posterior $\mathbb{E}_\pi[\theta \mid X]$

- The minimization problem is

$$\min_\beta \frac{1}{N} \sum_{i=1}^{N} \|f_\beta(X^{(i)}) - \theta^{(i)}\|_2^2$$

  where f_β denotes a DNN with parameter β.

- The resulting estimator $\hat{\theta}(X) := f_{\hat{\beta}}(X)$ approximates $\mathbb{E}_\pi[\theta \mid X]$ and serves as the summary statistic for ABC.

Jiang et al. 2017

(Toy) Example: MA(2)

Jiang et al. 2017

(Toy) Example: MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Jiang et al. 2017

(Toy) Example: MA(2)

$\epsilon_t \sim iid(0, \sigma^2)$

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Jiang et al. 2017

(Toy) Example: MA(2)

$\theta_1 \in [-2,2], \quad \theta_2 \in [-1,1], \quad \theta_2 \pm \theta_1 \geq -1$

$\epsilon_t \sim iid(0,\sigma^2)$

$$X_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2}$$

Jiang et al. 2017

(Toy) Example: MA(2)

$\theta_1 \in [-2,2], \quad \theta_2 \in [-1,1], \quad \theta_2 \pm \theta_1 \geq -1$

$\epsilon_t \sim iid(0,\sigma^2)$

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Input: Time series of length 100

Uniform prior on $\theta_1,\ \theta_2$

Jiang et al. 2017

(Toy) Example: MA(2)

$\theta_1 \in [-2,2], \quad \theta_2 \in [-1,1], \quad \theta_2 \pm \theta_1 \geq -1$

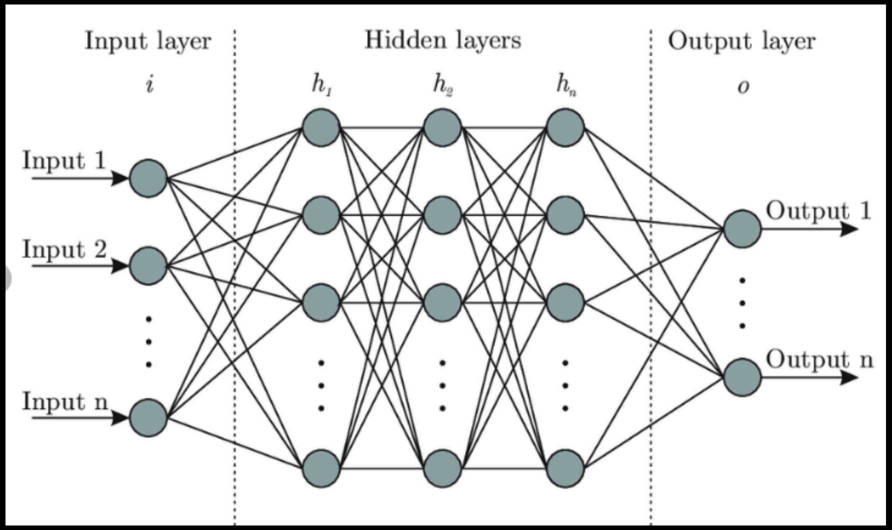
$\epsilon_t \sim iid(0,\sigma^2)$

$$X_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2}$$

3-layer DNN with 100 neurons on each hidden layer

Input: Time series of length 100

Uniform prior on $\theta_1,\ \theta_2$



Jiang et al. 2017

(Toy) Example: MA(2)

$\theta_1 \in [-2,2], \quad \theta_2 \in [-1,1], \quad \theta_2 \pm \theta_1 \geq -1$

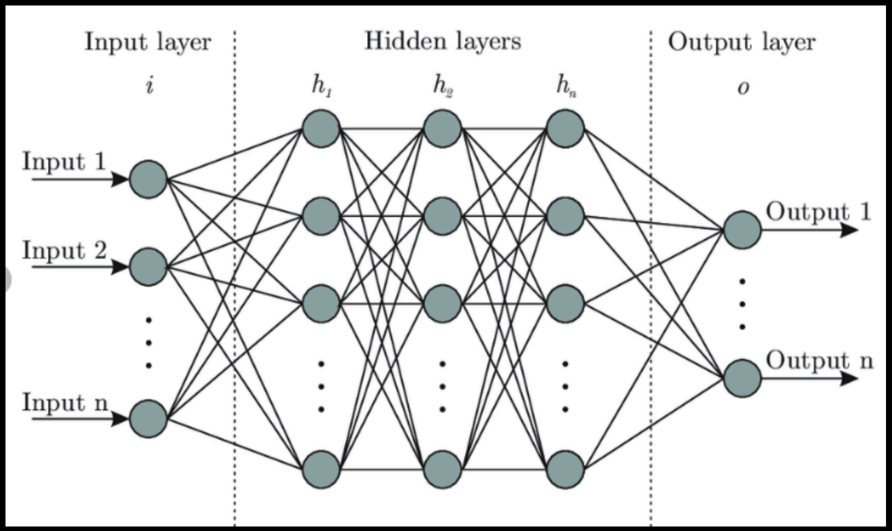
$\epsilon_t \sim iid(0,\sigma^2)$

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

3-layer DNN with 100 neurons on each hidden layer

Input: Time series of length 100

Uniform prior on $\theta_1,\ \theta_2$



Output: parameters $\theta_1,\ \theta_2$

Jiang et al. 2017

(Toy) Example: MA(2)

$\theta_1 \in [-2,2], \quad \theta_2 \in [-1,1], \quad \theta_2 \pm \theta_1 \geq -1$

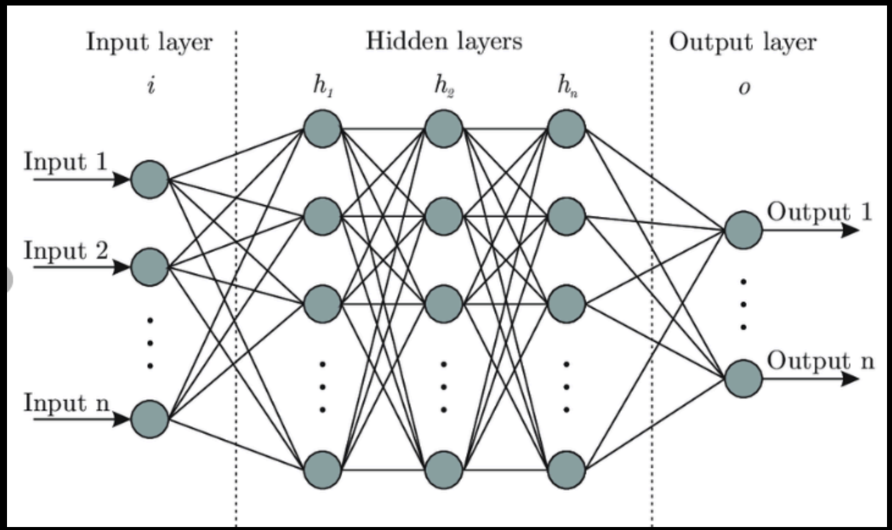
$\epsilon_t \sim iid(0,\sigma^2)$

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

3-layer DNN with 100 neurons on each hidden layer

Input: Time series of length 100

Uniform prior on $\theta_1,\ \theta_2$



Output: parameters $\theta_1,\ \theta_2$

$$\min_\beta \frac{1}{N} \sum_{i=1}^{N} \| f_\beta(X^{(i)}) - \theta^{(i)} \|_2^2 \qquad N = 10^5$$

Jiang et al. 2017

Results

Jiang et al. 2017

Results

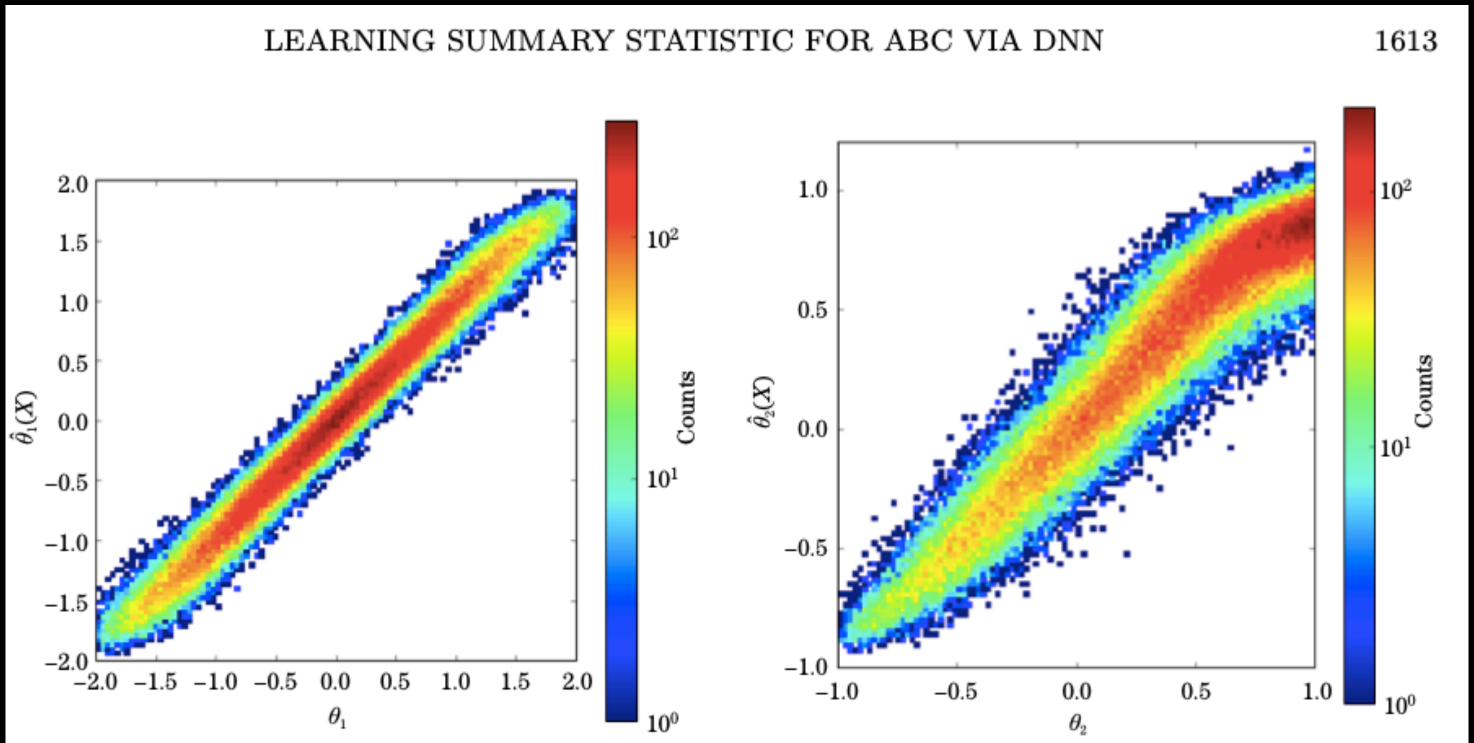


Figure 8. DNN predicting θ_1, θ_2 on the test dataset of 10^5 instances.

Jiang et al. 2017

Results

LEARNING SUMMARY STATISTIC FOR ABC VIA DNN 1613

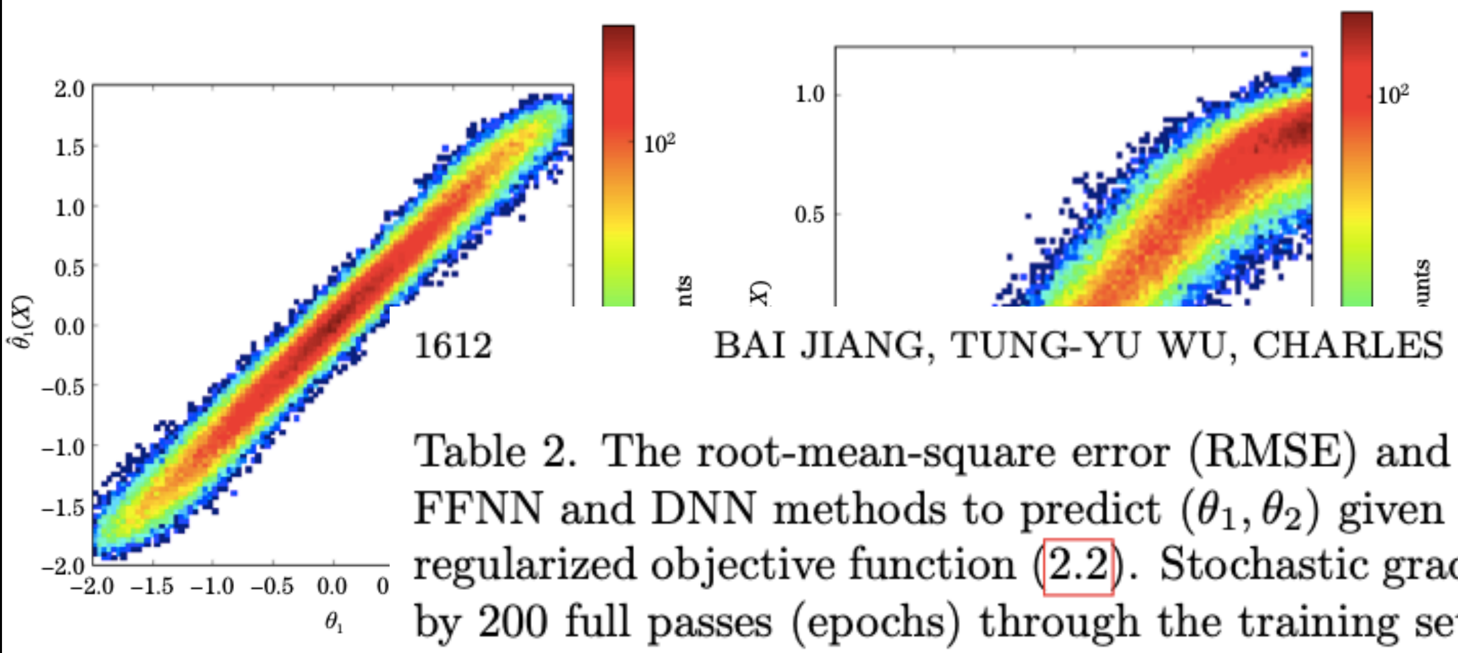


Figure 8. DN

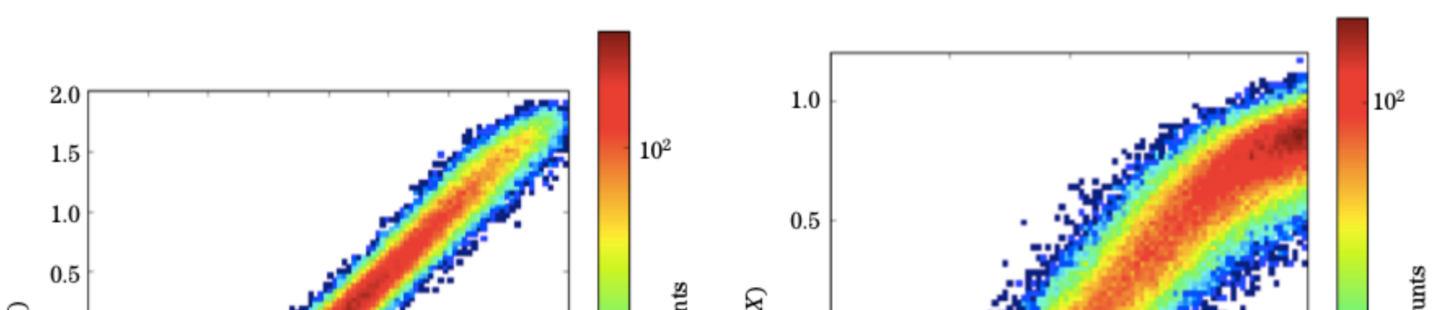
1612 BAI JIANG, TUNG-YU WU, CHARLES ZHENG AND WING WONG

Table 2. The root-mean-square error (RMSE) and training time of the semi-automatic, FFNN and DNN methods to predict (θ_1, θ_2) given X. λ is the penalty coefficient in the regularized objective function (2.2). Stochastic gradient descent fits each FFNN or DNN by 200 full passes (epochs) through the training set.

	Training RMSE		Testing RMSE		Time (s)
Method	θ_1	θ_2	θ_1	θ_2	
Semi-automatic	0.8150	0.3867	0.8174	0.3857	45.63
FFNN, $\lambda = 0$	0.1857	0.2091	0.1884	0.2115	543.42
DNN, $\lambda = 0$	**0.1272**	**0.1355**	**0.1293**	**0.1378**	1,402.02
FFNN, $\lambda = 0.001$	0.2642	0.2522	0.2679	0.2546	432.27
DNN, $\lambda = 0.001$	0.1958	0.1939	0.1980	0.1956	1,282.66

Jiang et al. 2017

Results



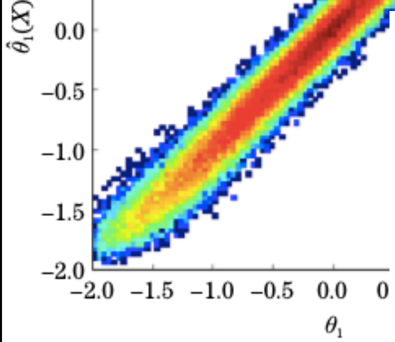


Figure 8. DNN

Table 2. The root-mean-square error (RMSE) and training time of the semi-automatic, FFNN and DNN methods to predict (θ_1, θ_2) given X. λ is the penalty coefficient in the regularized objective function (2.2). Stochastic gradient descent fits each FFNN or DNN by 200 full passes (epochs) through the training set.

Method	Training RMSE θ_1	Training RMSE θ_2	Testing RMSE θ_1	Testing RMSE θ_2	Time (s)
Semi-automatic	0.8150	0.3867	0.8174	0.3857	45.63
FFNN, $\lambda = 0$	0.1857	0.2091	0.1884	0.2115	543.42
DNN, $\lambda = 0$	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
FFN					
DNN					

Table 3. Mean and covariance of exact/ABC posterior distributions for observed data x_{obs} generated with $\theta = (0.6, 0.2)$ in Figure 10.

Posterior	mean(θ_1)	mean(θ_2)	std(θ_1)	std(θ_2)	cor(θ_1, θ_2)
Exact	0.6418	0.2399	0.1046	0.1100	0.6995
ABC (DNN)	0.6230	0.2300	0.1210	0.1410	0.4776
ABC (auto-cov)	0.7033	0.1402	0.1218	0.2111	0.2606
ABC (semi-auto)	0.0442	0.1159	0.5160	0.4616	−0.0645

Jiang et al. 2017

Ising model

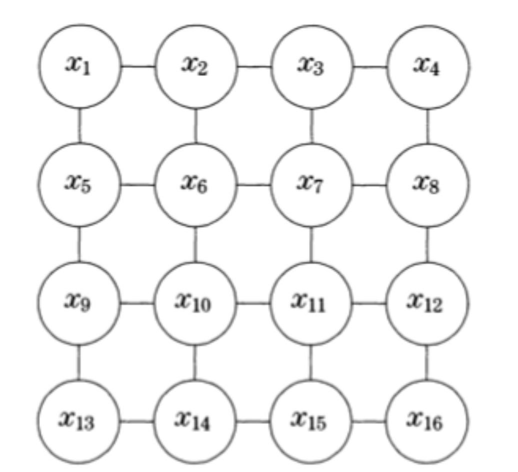


Figure 3. Ising model on 4 × 4 lattice.

$X_j \in \{-1, 1\}$

$$p(X \mid \theta) = \frac{exp\left(\theta \sum_{j\sim k} X_j X_k\right)}{Z(\theta)}$$

Jiang et al. 2017

Ising model

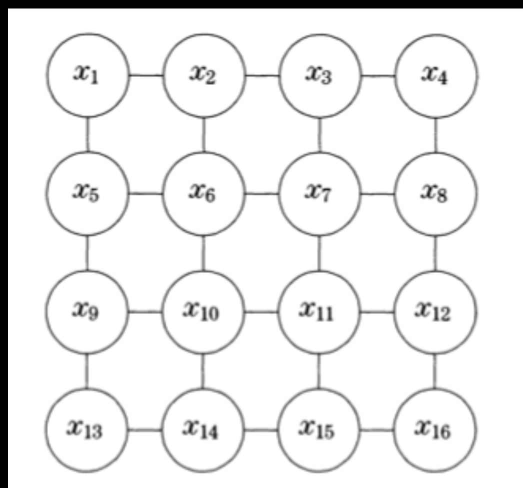


Figure 3. Ising model on 4×4 lattice.

$X_j \in \{-1, 1\}$

$$p(X|\theta) = \frac{exp\left(\theta \sum_{j\sim k} X_j X_k\right)}{Z(\theta)}$$

- **Even if the pdf is unavailable, data X can be simulated given θ using Monte Carlo methods such as the Metropolis algorithm**

Jiang et al. 2017

Ising model

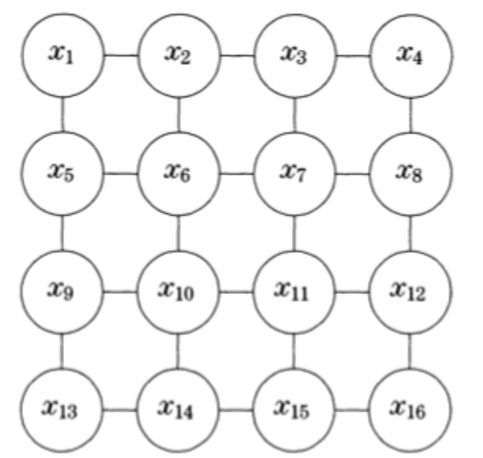


Figure 3. Ising model on 4×4 lattice.

$X_j \in \{-1,1\}$

$$p(X|\theta) = \frac{exp\left(\theta \sum_{j\sim k} X_j X_k\right)}{Z(\theta)}$$

- **Even if the pdf is unavailable, data X can be simulated given θ using Monte Carlo methods such as the Metropolis algorithm**
- **ABC inference can be performed by using the sufficient statistic** $S^*(X) = \sum_{j\sim k} X_j X_k$

Jiang et al. 2017

Ising model

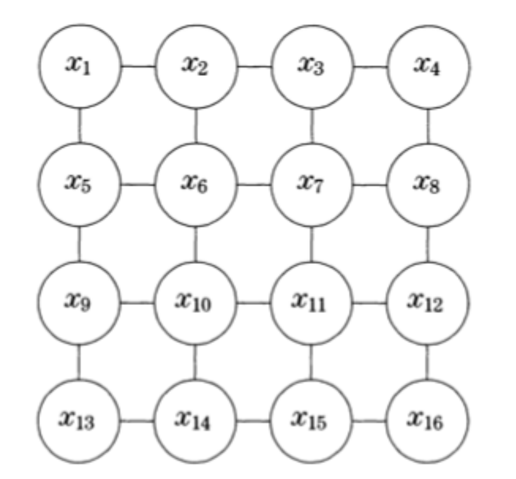


Figure 3. Ising model on 4×4 lattice.

$X_j \in \{-1,1\}$

$$p(X|\theta) = \frac{exp\left(\theta \sum_{j\sim k} X_j X_k\right)}{Z(\theta)}$$

- **Even if the pdf is unavailable, data X can be simulated given θ using Monte Carlo methods such as the Metropolis algorithm**
- **ABC inference can be performed by using the sufficient statistic** $S^*(X) = \sum_{j\sim k} X_j X_k$
- **Alternatively one can use a DNN to estimate θ**

Jiang et al. 2017

Ising model

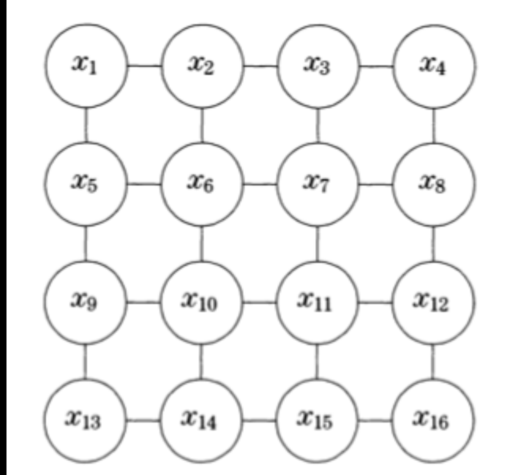
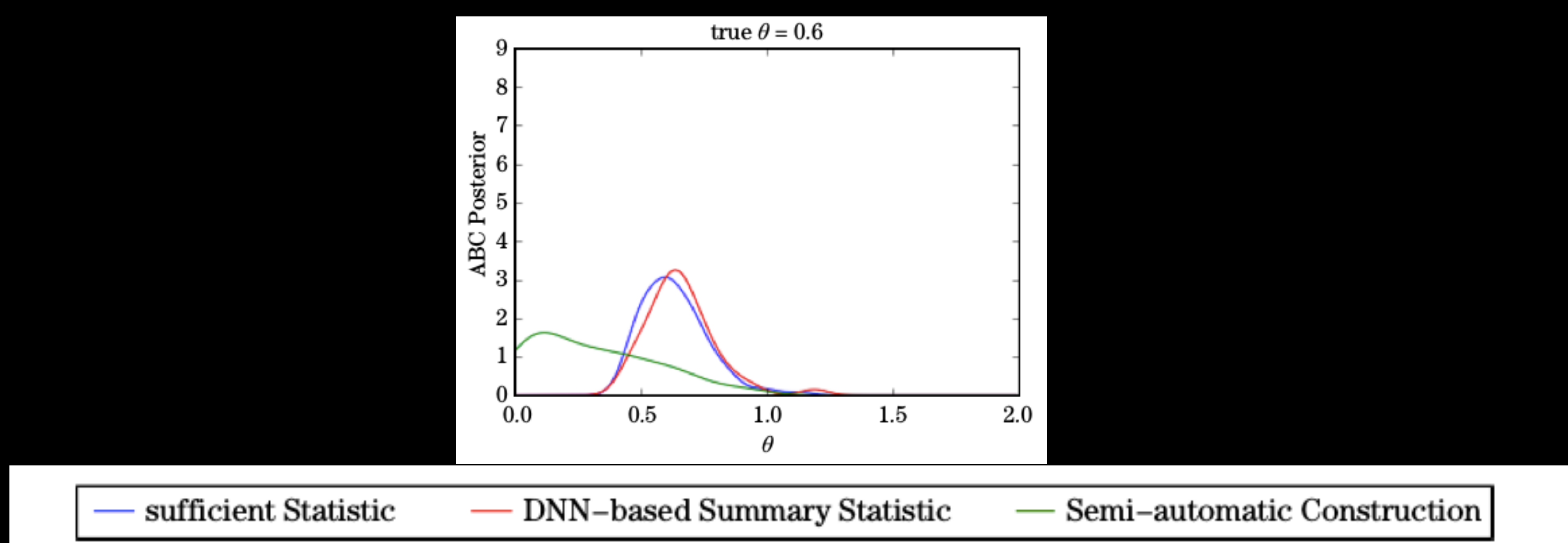


Figure 3. Ising model on 4×4 lattice.

$$X_j \in \{-1,1\}$$

$$p(X \mid \theta) = \frac{exp\left(\theta \sum_{j\sim k} X_j X_k\right)}{Z(\theta)}$$

- **Even if the pdf is unavailable, data X can be simulated given θ using Monte Carlo methods such as the Metropolis algorithm**
- **ABC inference can be performed by using the sufficient statistic $S^*(X) = \sum_{j\sim k} X_j X_k$**
- **Alternatively one can use a DNN to estimate θ**



Jiang et al. 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta | Z_n]$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta | Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta \mid Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

$\longrightarrow$

Auxiliary model AR(10)

$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta | Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

$\longrightarrow$

Auxiliary model AR(10)

$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

From a simulated MA(2) estimate the parameters $Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta | Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Auxiliary model AR(10)

$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

From a simulated MA(2) estimate the parameters $Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$

Input: estimated parameters of the auxiliary model

$Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta \mid Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Auxiliary model AR(10)

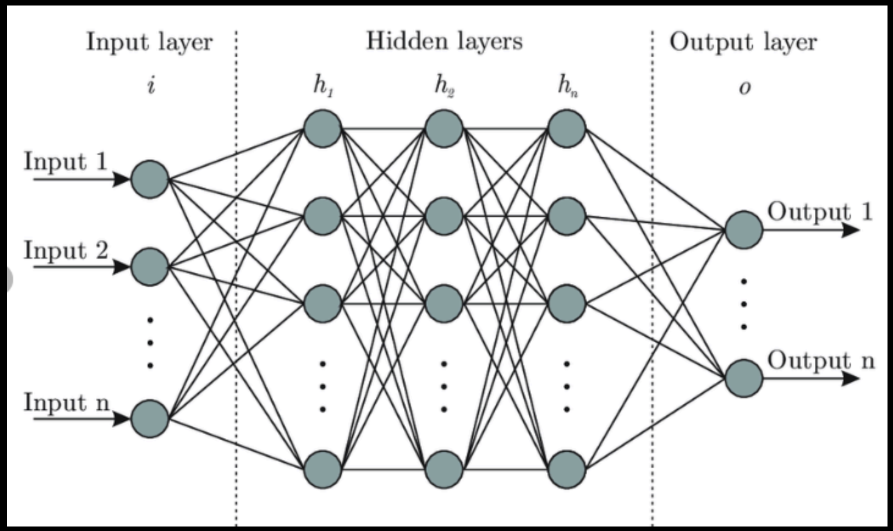
$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

From a simulated MA(2) estimate the parameters $Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$

2-layer DNN with 100 and 20 neurons

Input: estimated parameters of the auxiliary model

$Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$



Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta | Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Auxiliary model AR(10)

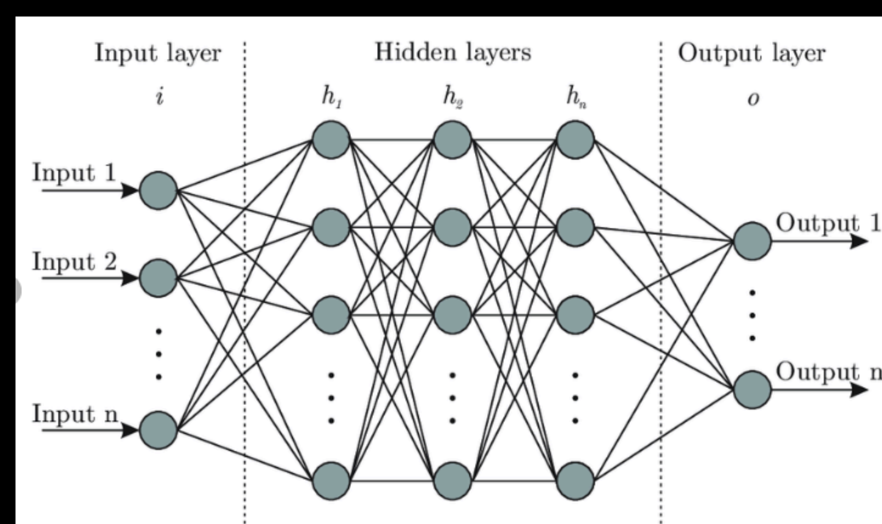
$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

From a simulated MA(2) estimate the parameters $Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$

2-layer DNN with 100 and 20 neurons

Input: estimated parameters of the auxiliary model

$Z_n = \{\hat{\rho}_0, \ldots, \hat{\rho}_{10}\}$



Output: parameters $\theta_1,\ \theta_2$

Creel 2017

An alternative: Indirect inference

Idea: use a statistic Z_n to capture the information in the sample, and then use the statistic as the input to a DNN, which will be trained to approximate $\mathbb{E}[\theta \,|\, Z_n]$

Data Generating Process MA(2)

$$X_t = \epsilon_t + \theta_1 \epsilon_{t-1} + \theta_2 \epsilon_{t-2}$$

Auxiliary model AR(10)

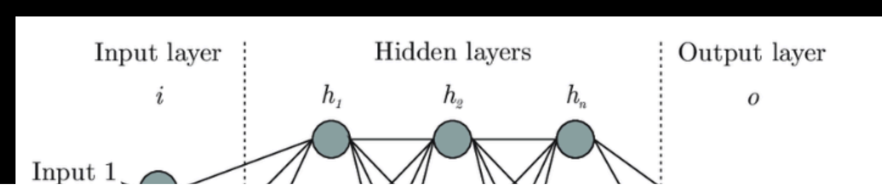
$$X_t = \rho_0 + \sum_{s=1}^{10} \rho_s X_{t-s} + v_t$$

From a simulated MA(2) estimate the parameters $Z_n = \{\hat\rho_0, \ldots, \hat\rho_{10}\}$

2-layer DNN with 100 and 20 neurons

Input: estimated parameters of the auxiliar

$Z_n = \{\hat\rho_0$



arameters $\theta_1,\ \theta_2$

Table 2
MA(2) model. Results for RMSE in test sets. Results for the $E(\theta|Y_n = y_n)$ target are from Jiang et al. (2015), p. 18.

	Target of net	
Parameter	$E(\theta\|Y_n = y_n)$	$E(\theta\|Z_n = z_n)$
θ_1	0.021	0.010
θ_2	0.024	0.011

Creel 2017

Estimating noisy dynamical systems from short time series with Deep Neural Networks

arXiv.org > q-fin > arXiv:2104.04960

Search...

Help | Advanc

Quantitative Finance > Mathematical Finance

[Submitted on 11 Apr 2021]

Analysis of bank leverage via dynamical systems and deep neural networks

Fabrizio Lillo, Giulia Livieri, Stefano Marmi, Anton Solomko, Sandro Vaienti

We consider a model of a simple financial system consisting of a leveraged investor that invests in a risky asset and manages risk by using Value-at-Risk (VaR). The VaR is estimated by using past data via an adaptive expectation scheme. We show that the leverage dynamics can be described by a dynamical system of slow-fast type associated with a unimodal map on [0,1] with an additive heteroscedastic noise whose variance is related to the portfolio rebalancing frequency to target leverage. In absence of noise the model is purely deterministic and the parameter space splits in two regions: (i) a region with a globally attracting fixed point or a 2-cycle; (ii) a dynamical core region, where the map could exhibit chaotic behavior. Whenever the model is randomly perturbed, we prove the existence of a unique stationary density with bounded variation, the stochastic stability of the process and the almost certain existence and continuity of the Lyapunov exponent for the stationary measure. We then use deep neural networks to estimate map parameters from a short time series. Using this method, we estimate the model in a large dataset of US commercial banks over the period 2001-2014. We find that the parameters of a substantial fraction of banks lie in the dynamical core, and their leverage time series are consistent with a chaotic behavior. We also present evidence that the time series of the leverage of large banks tend to exhibit chaoticity more frequently than those of small banks.

shot

Comments: 51 pages, 12 figures

Subjects: **Mathematical Finance (q-fin.MF)**; Dynamical Systems (math.DS); Risk Management (q-fin.RM)

Cite as: arXiv:2104.04960 **[q-fin.MF]**

(or arXiv:2104.04960v1 **[q-fin.MF]** for this version)

Submission history

From: Sandro Vaienti [view email]

[v1] Sun, 11 Apr 2021 08:48:43 UTC (4,020 KB)

The economic problem (in a nutshell)

Lillo et al 2021

The economic problem (in a nutshell)

- A representative bank wants to maximize leverage (i.e the ratio between assets and equity) by taking more debt, but is constrained by the regulator so that

$$\lambda_t = \frac{1}{\alpha \sigma_{e,t}}$$

where $\sigma_{e,t}$ is the expected risk (variance) of its assets.

Lillo et al 2021

The economic problem (in a nutshell)

- A representative bank wants to maximize leverage (i.e the ratio between assets and equity) by taking more debt, but is constrained by the regulator so that

$$\lambda_t = \frac{1}{\alpha \sigma_{e,t}}$$

  where $\sigma_{e,t}$ is the expected risk (variance) of its assets.

- At each quarter t the bank decides the leverage $\lambda_t,\ t \in \mathbb{Z}$ based on expectation $\sigma_{e,t}$

Lillo et al 2021

The economic problem (in a nutshell)

- A representative bank wants to maximize leverage (i.e the ratio between assets and equity) by taking more debt, but is constrained by the regulator so that

$$\lambda_t = \frac{1}{\alpha \sigma_{e,t}}$$

where $\sigma_{e,t}$ is the expected risk (variance) of its assets.

- At each quarter t the bank decides the leverage $\lambda_t,\ t \in \mathbb{Z}$ based on expectation $\sigma_{e,t}$
- In order to form expectations, the banks uses an adaptive scheme

$$\sigma^2_{e,t} = \omega \sigma^2_{e,t-1} + (1-\omega)\hat{\sigma}^2_{e,t}$$

where $\hat{\sigma}^2_{e,t}$ is an estimation of the risk of the investment (i.e. the variance of price increments) in the n days of the previous quarter

Lillo et al 2021

The economic problem (in a nutshell)

- A representative bank wants to maximize leverage (i.e the ratio between assets and equity) by taking more debt, but is constrained by the regulator so that

$$\lambda_t = \frac{1}{\alpha \sigma_{e,t}}$$

  where $\sigma_{e,t}$ is the expected risk (variance) of its assets.

- At each quarter t the bank decides the leverage $\lambda_t,\ t \in \mathbb{Z}$ based on expectation $\sigma_{e,t}$

- In order to form expectations, the banks uses an adaptive scheme

$$\sigma_{e,t}^2 = \omega \sigma_{e,t-1}^2 + (1-\omega)\hat{\sigma}_{e,t}^2$$

  where $\hat{\sigma}_{e,t}^2$ is an estimation of the risk of the investment (i.e. the variance of price increments) in the n days of the previous quarter

- In the attempt of keeping the planned leverage, the bank trades and this moves the price of the assets, increasing also its variance. When trading V shares, the price moves (on average) by V/γ

Lillo et al 2021

The map

The map

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

The map

$$\phi_{t+1} = T(\phi_t;\theta) + \sigma(\phi_t;\theta)\epsilon_t$$

$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

The map

$$\theta = (b, \omega, n)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1 - \omega)\left(\frac{1 - \phi^*}{\phi^*}\right)^2$$

The map

$$\theta = (b, \omega, n)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$T(\phi_t; \theta) = \frac{|\phi_t(1 - \phi_t)|}{\sqrt{b\phi_t^2 + \omega(1 - \phi_t)^2}}$$

$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1 - \omega)\left(\frac{1 - \phi^*}{\phi^*}\right)^2$$

The map

$$\theta = (b, \omega, n)$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$T(\phi_t; \theta) = \frac{|\phi_t(1 - \phi_t)|}{\sqrt{b\phi_t^2 + \omega(1 - \phi_t)^2}}$$

$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1 - \omega)\left(\frac{1 - \phi^*}{\phi^*}\right)^2$$

The map

$$\theta = (b, \omega, n)$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$T(\phi_t; \theta) = \frac{|\phi_t(1-\phi_t)|}{\sqrt{b\phi_t^2 + \omega(1-\phi_t)^2}}$$

$$\sigma(\phi_t; \theta) = \frac{b\phi_t^3\sqrt{1-\phi_t^2}}{\sqrt{n}\left(b\phi_t^2 + \omega(1-\phi_t)^2\right)^{3/2}}$$

$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1-\omega)\left(\frac{1-\phi^*}{\phi^*}\right)^2$$

The map

$$\theta = (b, \omega, n)$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

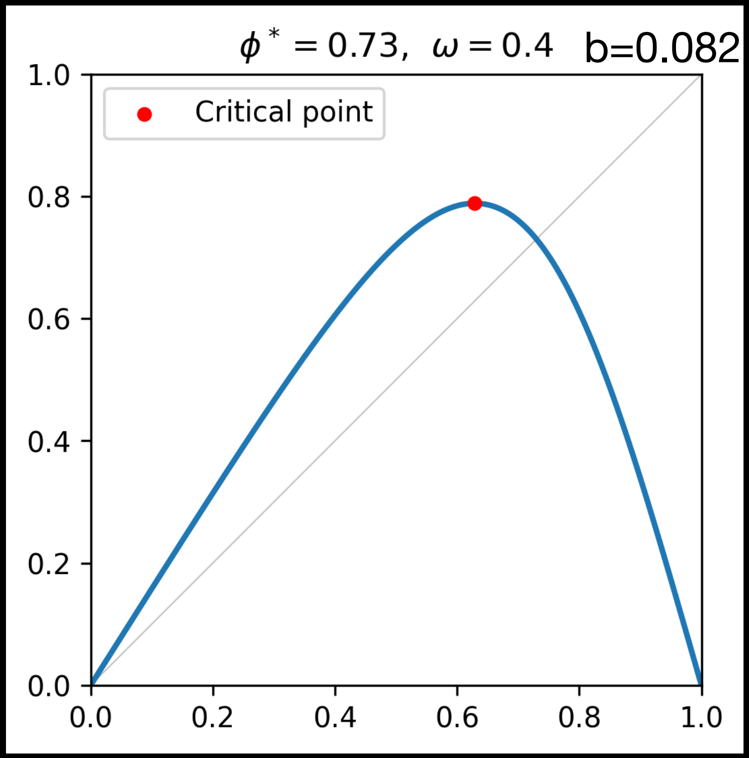
$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$T(\phi_t; \theta) = \frac{|\phi_t(1-\phi_t)|}{\sqrt{b\phi_t^2 + \omega(1-\phi_t)^2}}$$

$$\sigma(\phi_t; \theta) = \frac{b\phi_t^3\sqrt{1-\phi_t^2}}{\sqrt{n}\left(b\phi_t^2 + \omega(1-\phi_t)^2\right)^{3/2}}$$

$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1-\omega)\left(\frac{1-\phi^*}{\phi^*}\right)^2$$



The map

$$\theta = (b, \omega, n)$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

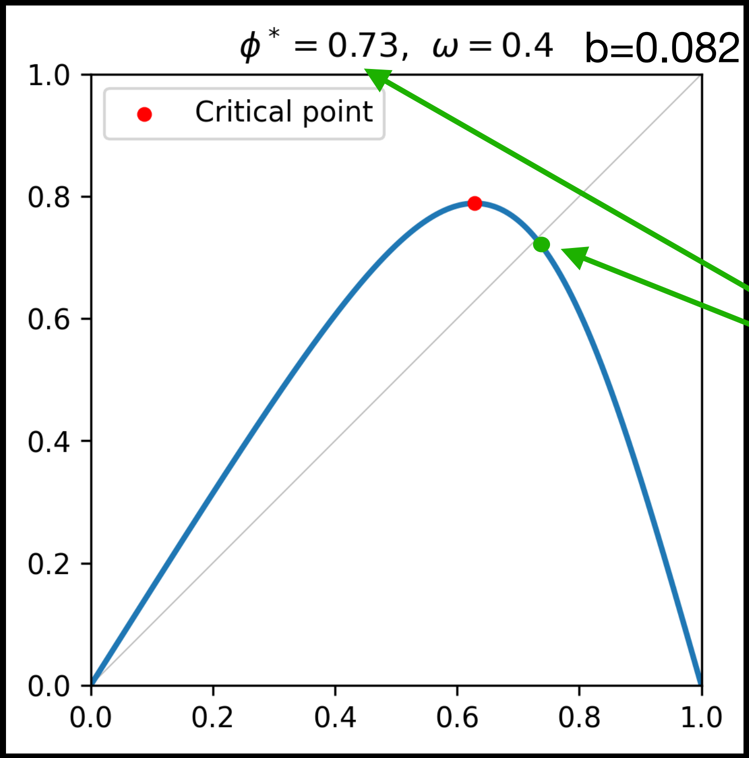
$$\phi_t := \frac{\lambda_t - 1}{\gamma}$$

$$T(\phi_t; \theta) = \frac{|\phi_t(1 - \phi_t)|}{\sqrt{b\phi_t^2 + \omega(1 - \phi_t)^2}}$$

$$\sigma(\phi_t; \theta) = \frac{b\phi_t^3\sqrt{1 - \phi_t^2}}{\sqrt{n}\left(b\phi_t^2 + \omega(1 - \phi_t)^2\right)^{3/2}}$$

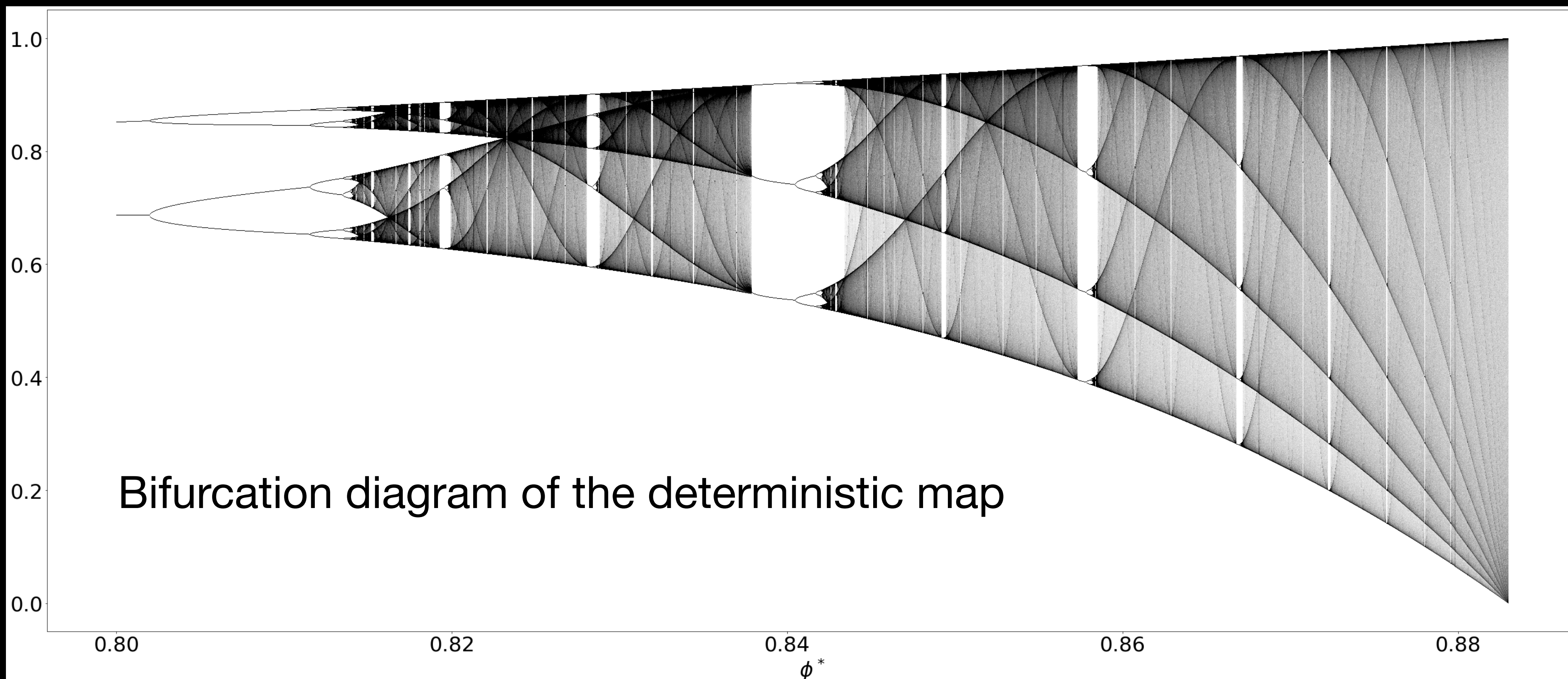
$$\phi^* = \frac{1 - \alpha\sqrt{\Sigma_\epsilon}}{1 + \alpha\gamma\sqrt{\Sigma_\epsilon}}$$

$$b = (1 - \omega)\left(\frac{1 - \phi^*}{\phi^*}\right)^2$$

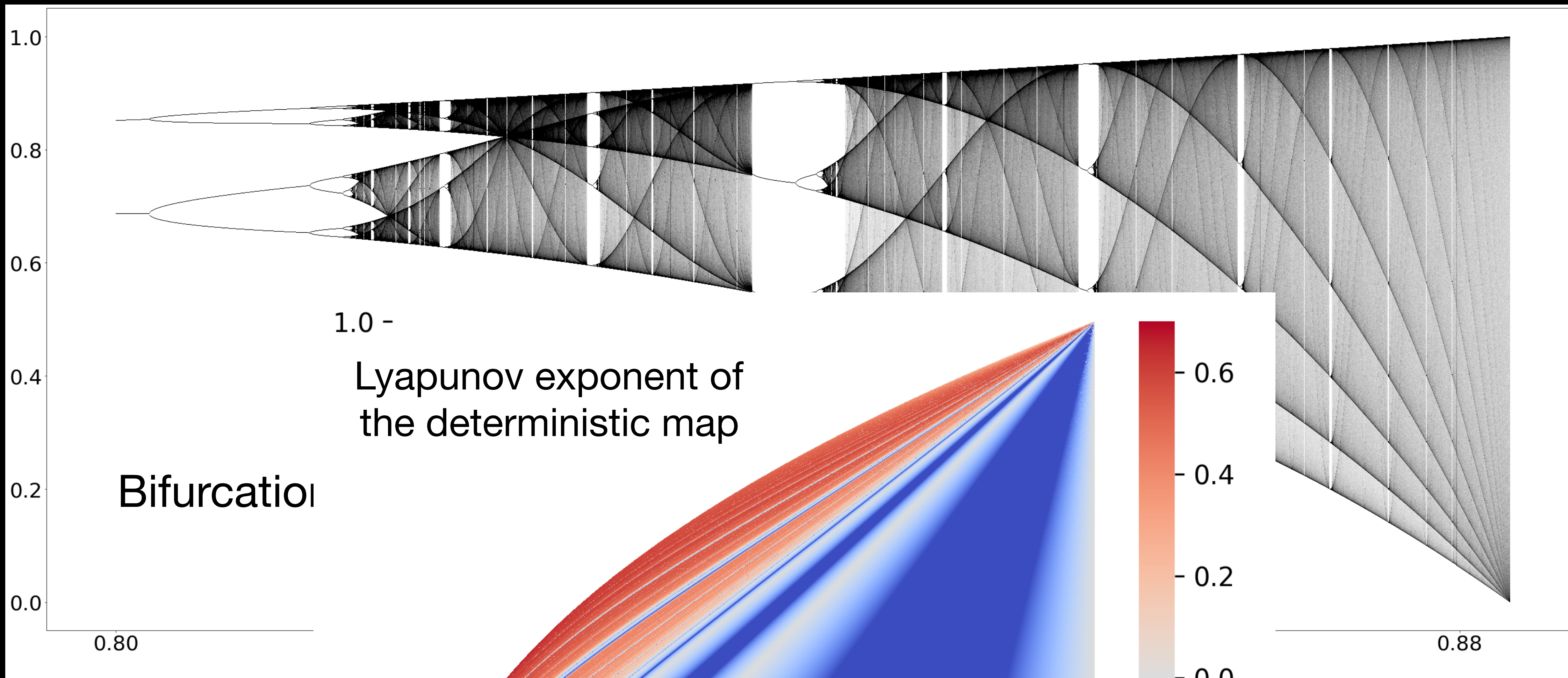


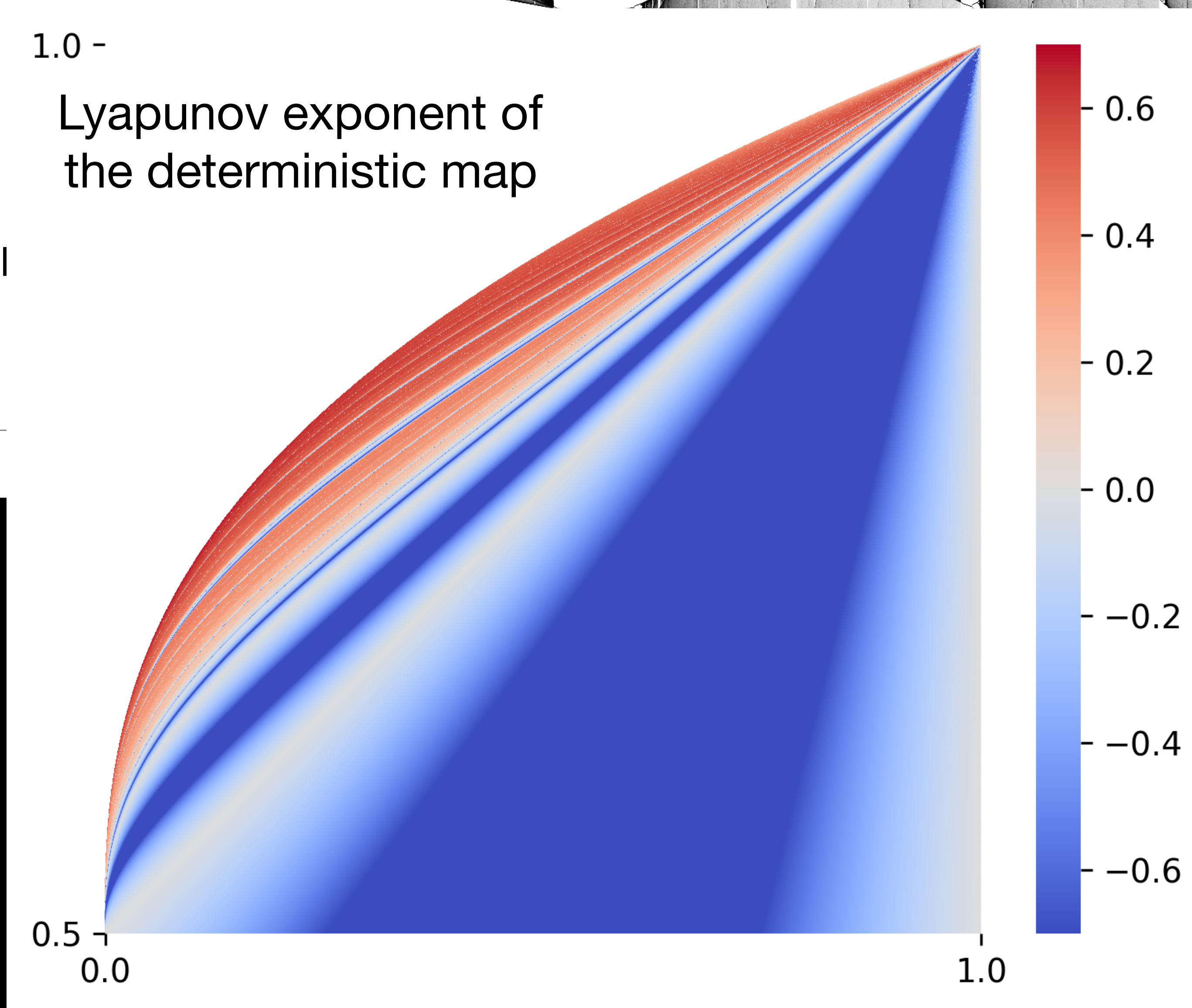
Fixed point of the map

Properties of the map

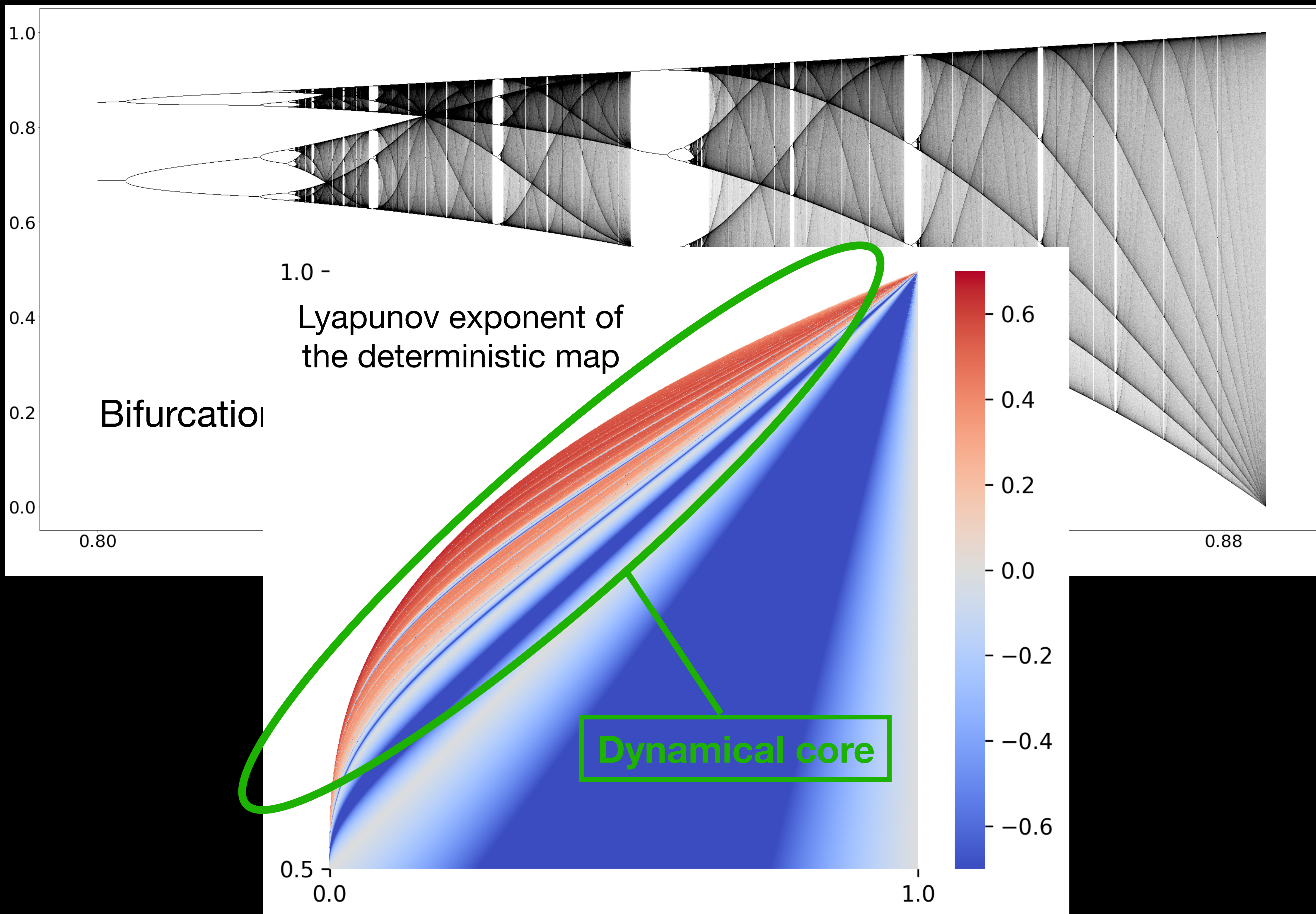


Properties of the map

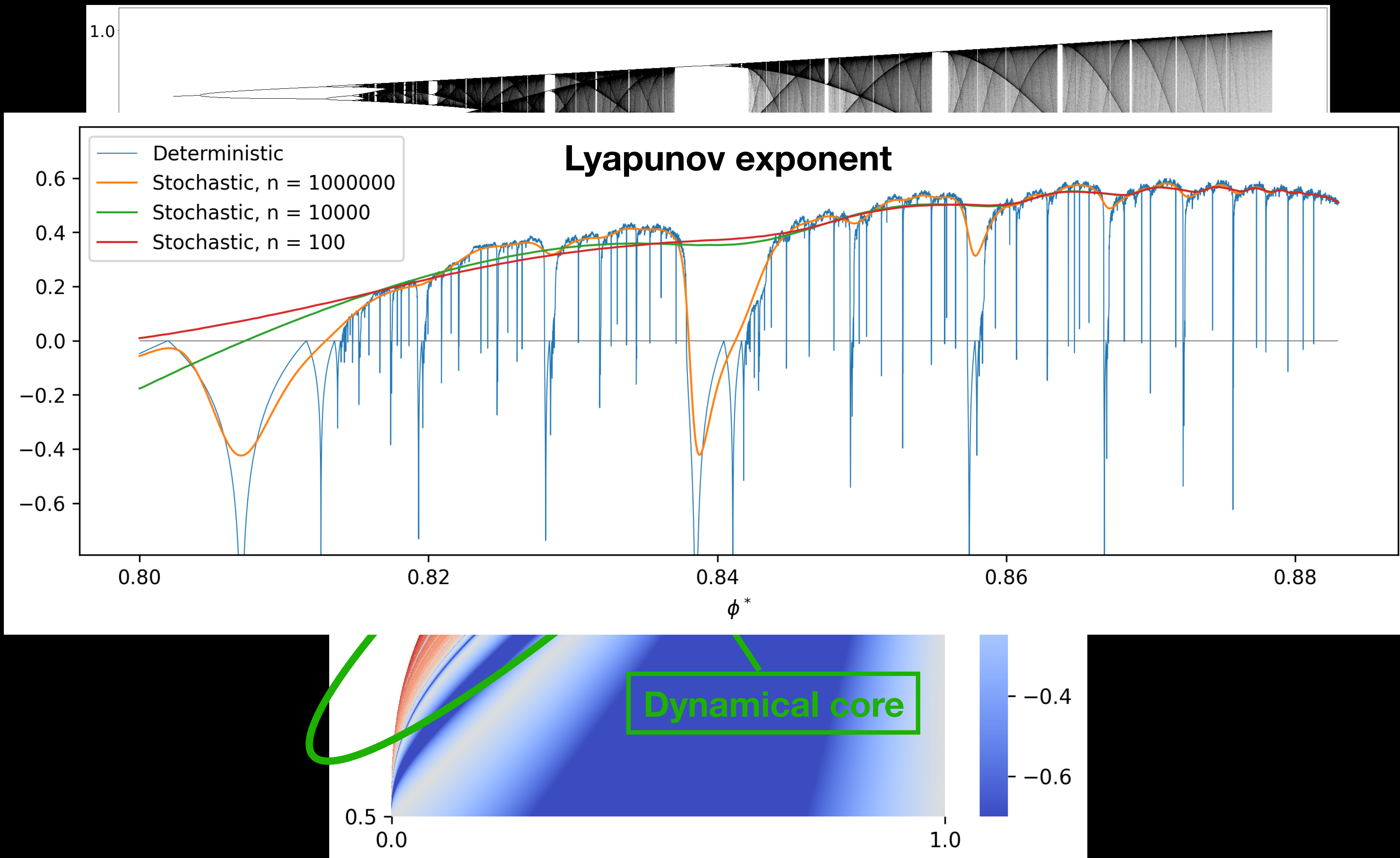




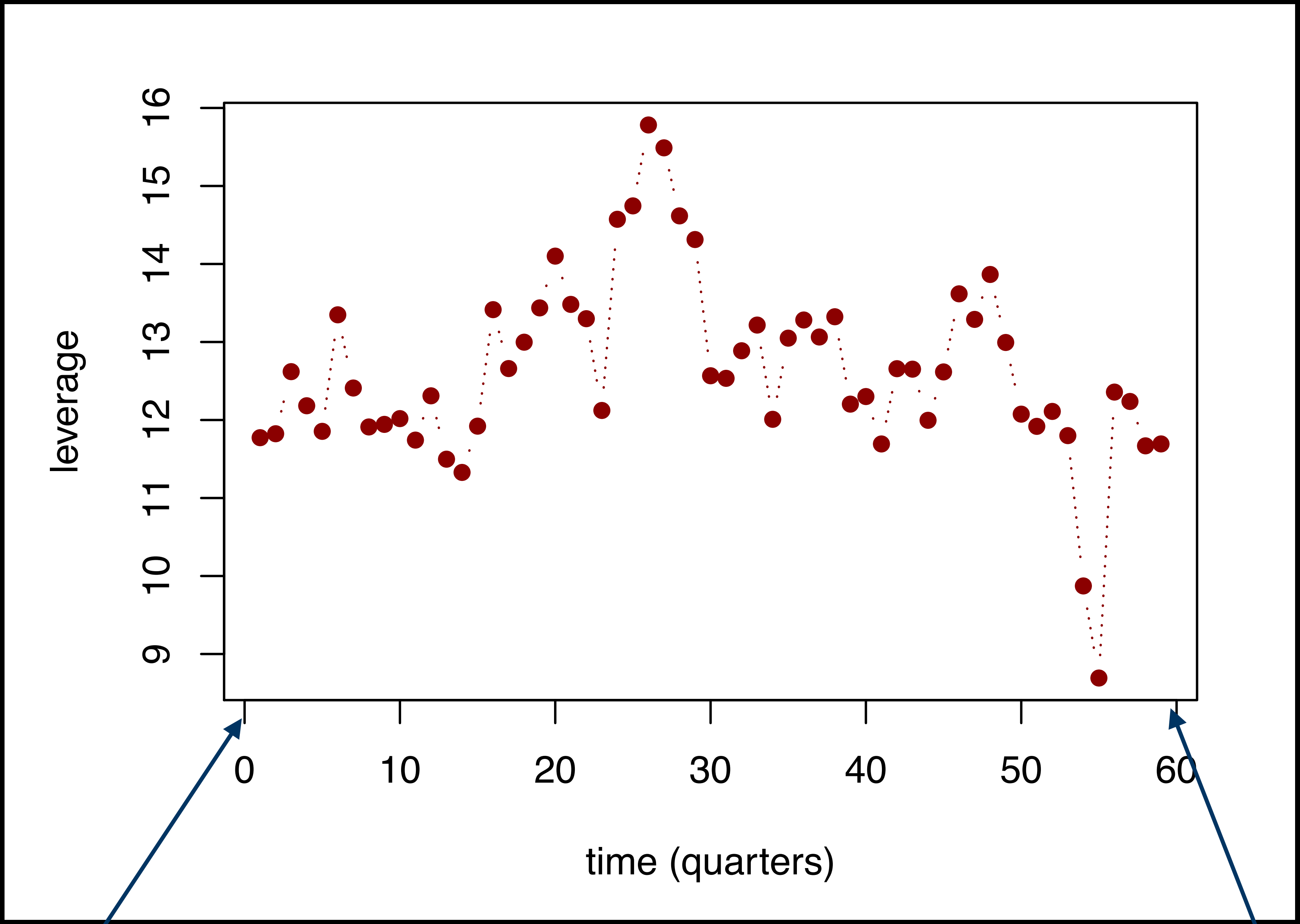
Properties of the map



Properties of the map



An example from real data



March 2001

Short time series!

December 2014

How to estimate map parameters?

How to estimate map parameters?

$$\phi_{t+1} = T(\phi_t; \theta) + \sigma(\phi_t; \theta)\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

How to estimate map parameters?

$$\phi_{t+1} = T(\phi_t;\theta) + \sigma(\phi_t;\theta)\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$p(\phi_{t+1}\,|\,\phi_t,\theta) = \frac{1}{\sqrt{2\pi\sigma^2(x_t;\theta)}}e^{-\frac{(\phi_{t+1}-T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}}$$

How to estimate map parameters?

$$\phi_{t+1} = T(\phi_t;\theta) + \sigma(\phi_t;\theta)\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$p(\phi_{t+1} \,|\, \phi_t, \theta) = \frac{1}{\sqrt{2\pi\sigma^2(x_t;\theta)}} e^{-\frac{(\phi_{t+1} - T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}}$$

Joint probability $$p(\phi_2, \ldots, \phi_T \,|\, \phi_1, \theta) = \prod_{t=1}^{T-1} p(\phi_{t+1} \,|\, \phi_t, \theta)$$

How to estimate map parameters?

$$\phi_{t+1} = T(\phi_t;\theta) + \sigma(\phi_t;\theta)\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$p(\phi_{t+1}\,|\,\phi_t,\theta) = \frac{1}{\sqrt{2\pi\sigma^2(x_t;\theta)}} e^{-\frac{(\phi_{t+1}-T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}}$$

Joint probability $$p(\phi_2,\ldots,\phi_T\,|\,\phi_1,\theta) = \prod_{t=1}^{T-1} p(\phi_{t+1}\,|\,\phi_t,\theta)$$

Log-likelihood function

$$\mathcal{L}(\theta) = -\frac{T-1}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{T-1}\log\sigma^2(\phi_t;\theta) - \sum_{t=1}^{T-1}\frac{(\phi_{t+1}-T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}$$

How to estimate map parameters?

$$\phi_{t+1} = T(\phi_t;\theta) + \sigma(\phi_t;\theta)\epsilon_t$$

$$\epsilon_t \sim \mathcal{N}(0,1)$$

$$p(\phi_{t+1}\,|\,\phi_t,\theta) = \frac{1}{\sqrt{2\pi\sigma^2(x_t;\theta)}} e^{-\frac{(\phi_{t+1}-T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}}$$

Joint probability $$p(\phi_2,\ldots,\phi_T\,|\,\phi_1,\theta) = \prod_{t=1}^{T-1} p(\phi_{t+1}\,|\,\phi_t,\theta)$$

Log-likelihood function

$$\mathscr{L}(\theta) = -\frac{T-1}{2}\log 2\pi - \frac{1}{2}\sum_{t=1}^{T-1}\log\sigma^2(\phi_t;\theta) - \sum_{t=1}^{T-1}\frac{(\phi_{t+1}-T(\phi_t;\theta))^2}{2\sigma^2(\phi_t;\theta)}$$

Maximum Likelihood Estimator

$$\theta^* = \arg\max_{\theta\in\Omega}\mathscr{L}(\theta)$$

Problems

Problems

1. The likelihood function is typically non-convex in the parameters and its numerical optimization can end up in one of the many local maxima

Problems

1. The likelihood function is typically non-convex in the parameters and its numerical optimization can end up in one of the many local maxima

2. We may observe only one event out of two, or even out of three, four, etc. If we observe, for instance, only the second iterate of the process, the observed map is

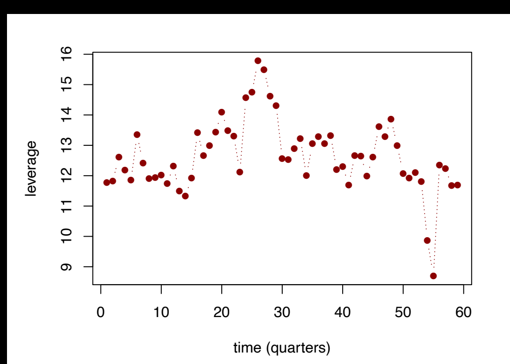
$$\phi_{t+2} = T(T(\phi_t;\theta);\theta) + \sigma(\phi_t;\theta)\epsilon_t) + \sigma(\phi_{t+1};\theta)\epsilon_{t+1}$$

and the transition probabilities $p(\phi_{t+2} \,|\, \phi_t;\theta)$ are no longer Gaussian (as it would be the case if we observe the first iterate).

The solution: Deep Neural Networks

The solution: Deep Neural Networks

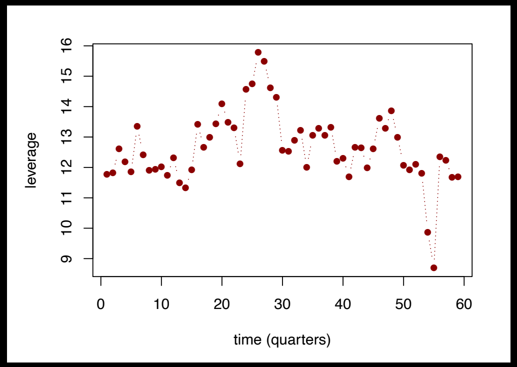
Input T=59 time series

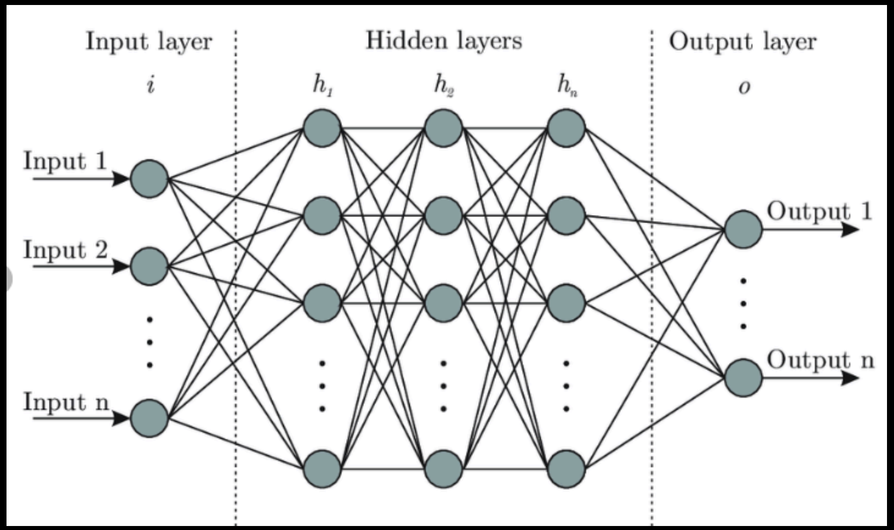


The solution: Deep Neural Networks

CNN1

Input T=59 time series

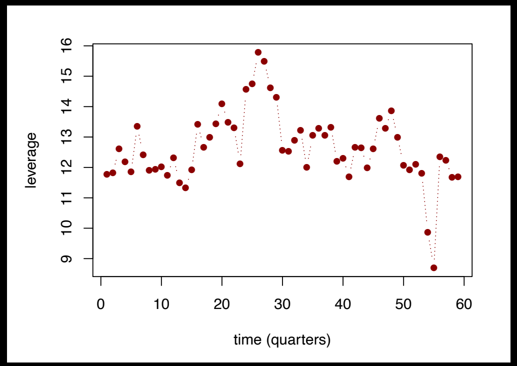


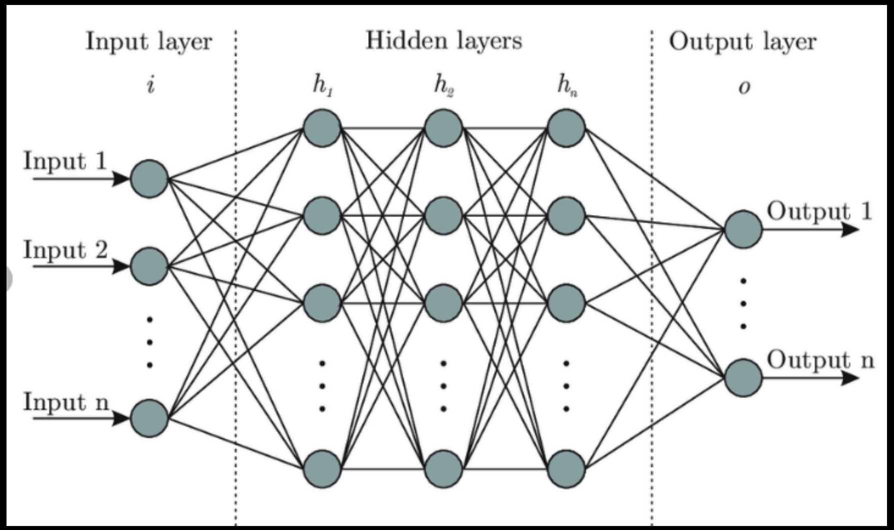


The solution: Deep Neural Networks

CNN1

Input T=59 time series



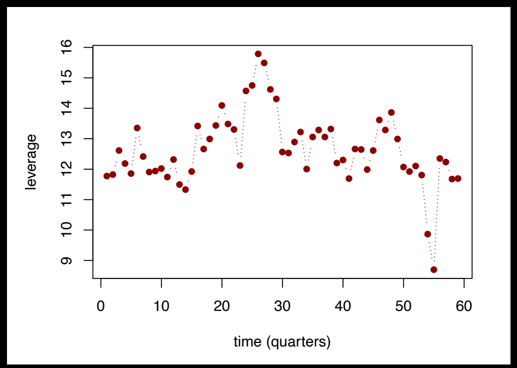


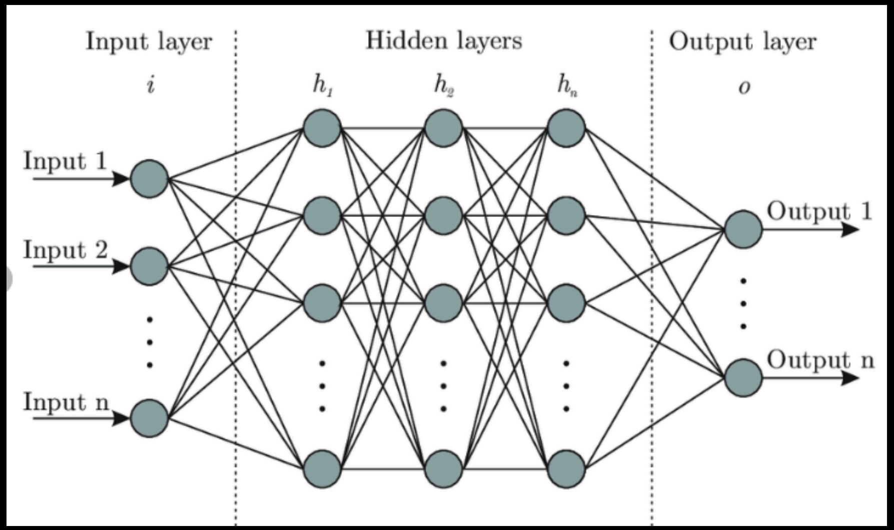
Output: Iterate k (=1,2,3)

The solution: Deep Neural Networks

CNN1

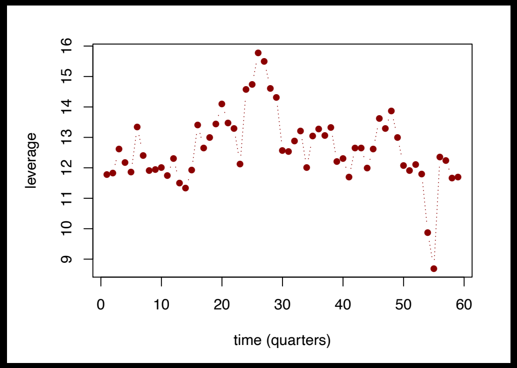
Input T=59 time series





Output: Iterate k (=1,2,3)

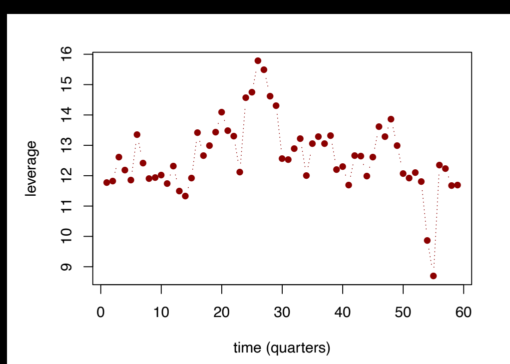
Input T=59 time series

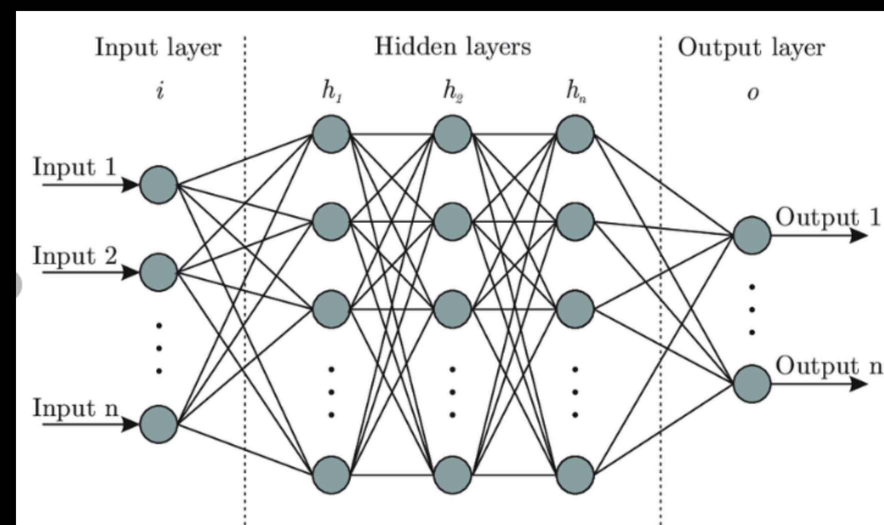


The solution: Deep Neural Networks

CNN1

Input T=59 time series

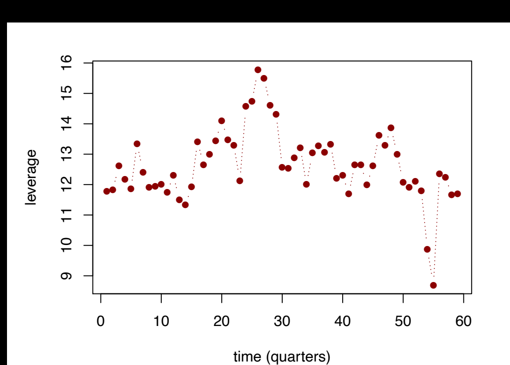


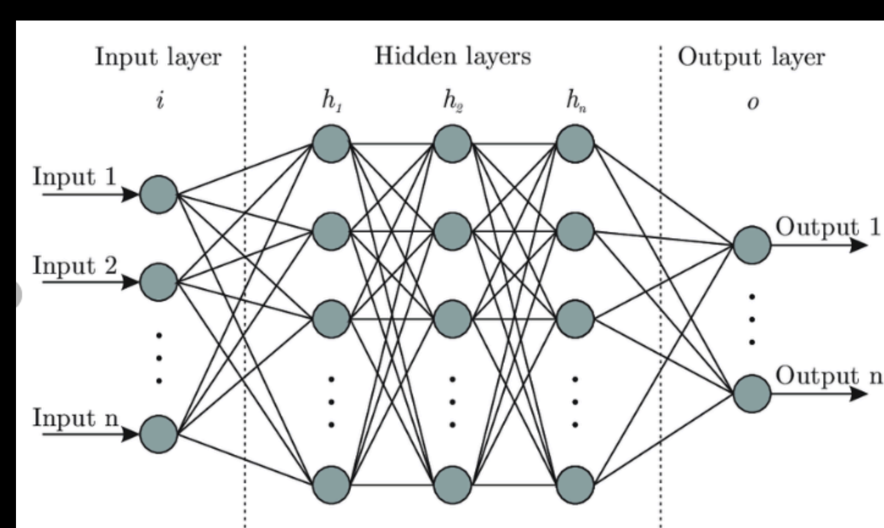


Output: Iterate k (=1,2,3)

CNN2(k)

Input T=59 time series

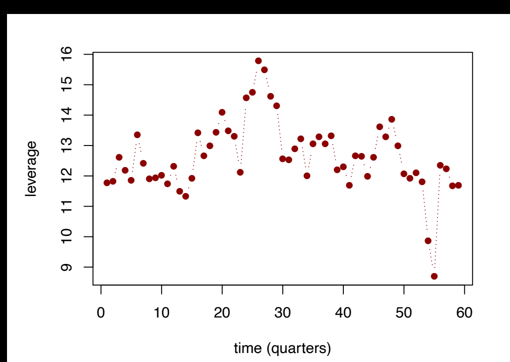


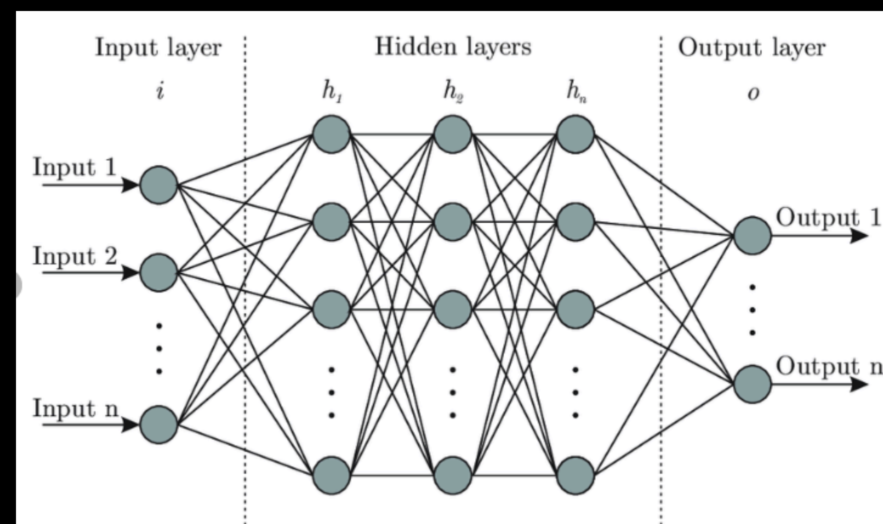


The solution: Deep Neural Networks

CNN1

Input T=59 time series

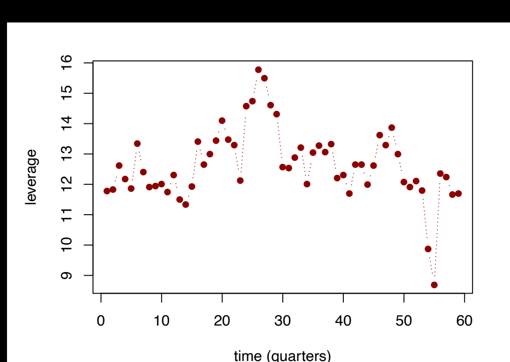


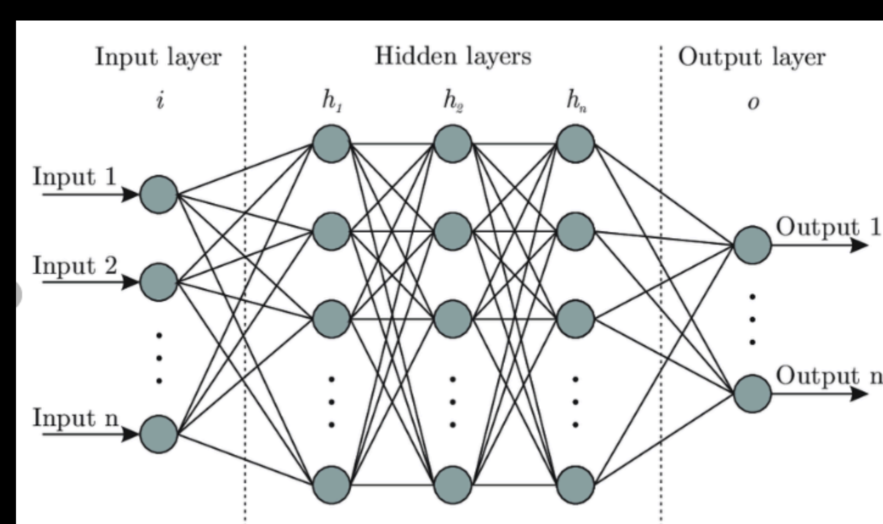


Output: Iterate k (=1,2,3)

CNN2(k)

Input T=59 time series





Output: Parameters (ϕ^*, ω) of the map

The NN architecture

The input is the whole time series (T=59)

The output are the two parameters of interest

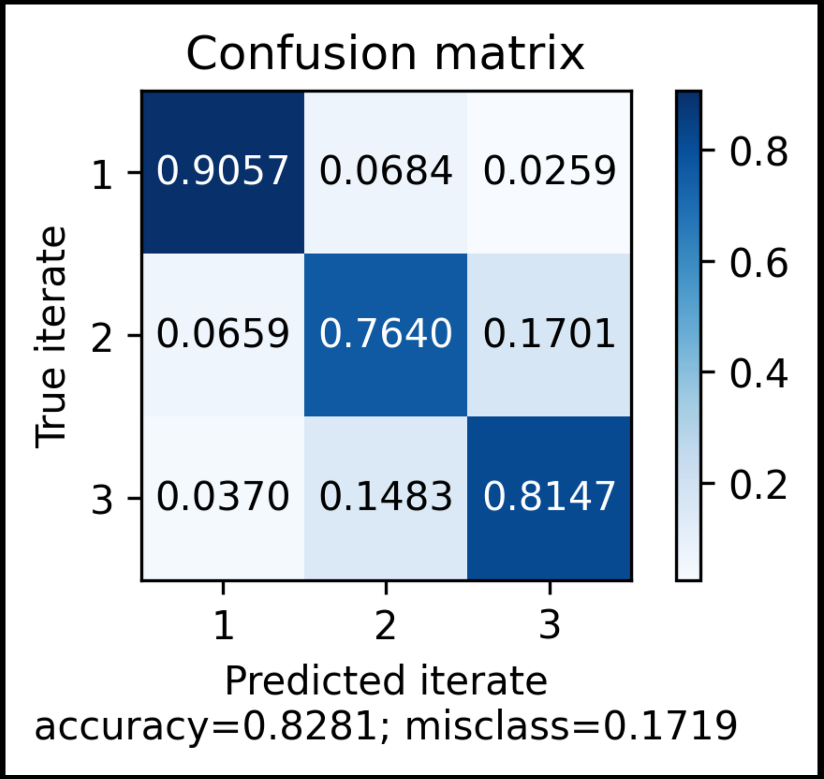
The number of iterations is determined by another NN

```
Model CNN1: "convolutional_categorical_model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 59, 1)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 58, 128)           384
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 29, 64)            16448
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 15, 64)            8256
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 8, 64)             8256
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 4, 64)             8256
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 2, 64)             8256
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 1, 64)             8256
_________________________________________________________________
flatten (Flatten)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               8320
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 195
=================================================================
Trainable params: 74,883

Model CNN2: "convolutional_model"
...
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 130
=================================================================
Trainable params: 74,818
```

Figure 8: Architectures of the CNN1 model used to estimate the iterate k and the CNN2(k) model used to estimate the parameters (ϕ^*, ω) for each k. The two models differ only in the output layer.

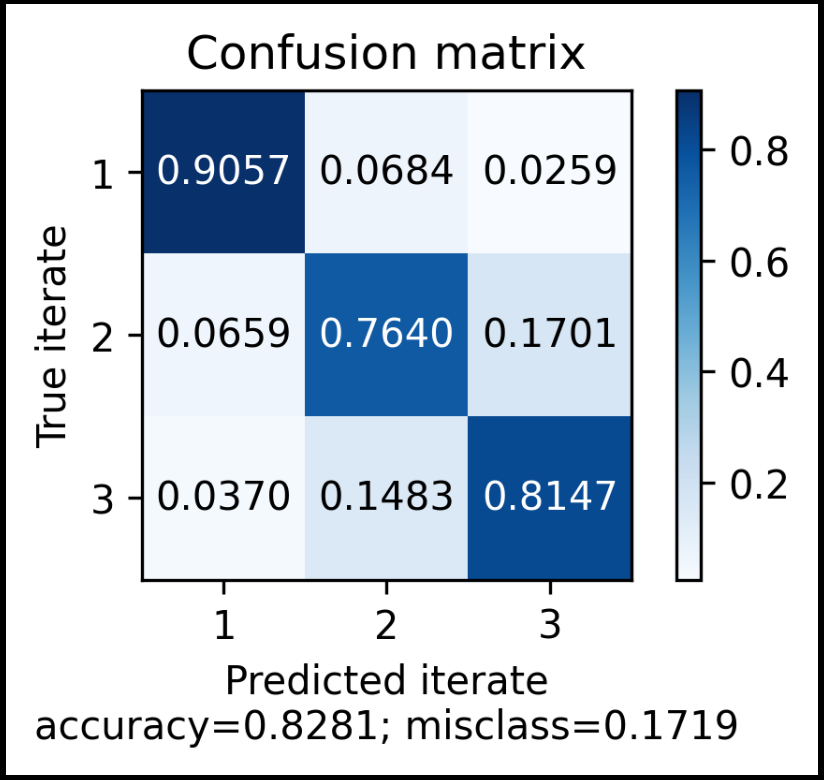
Training & testing



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

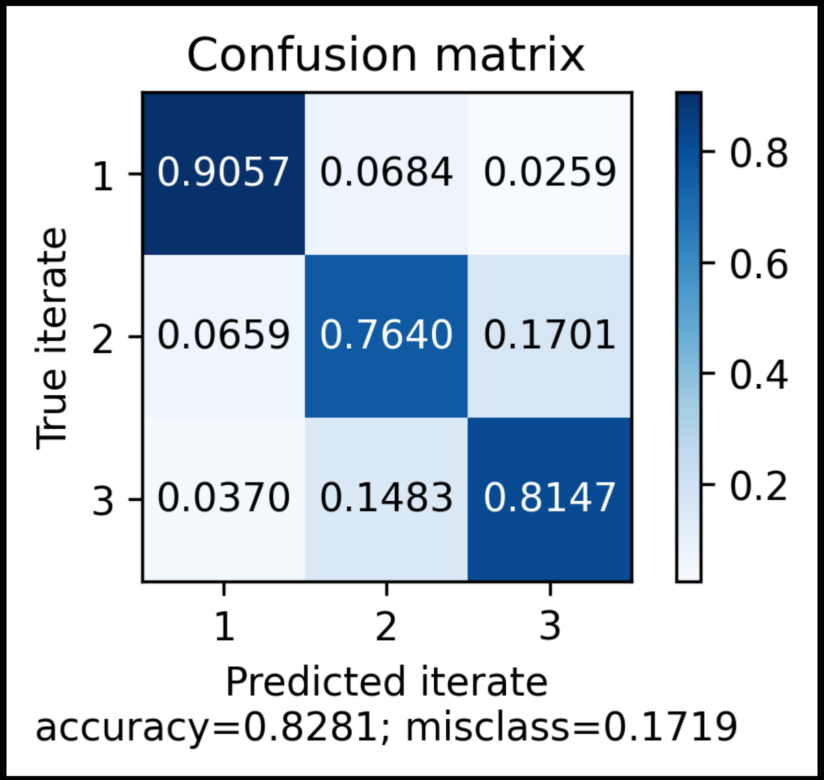
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

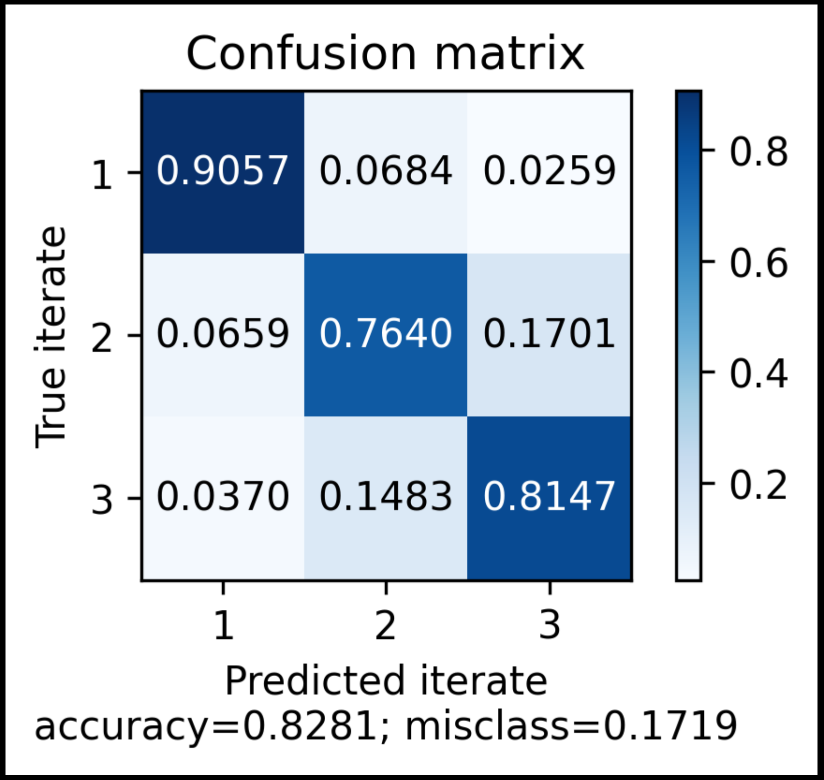
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

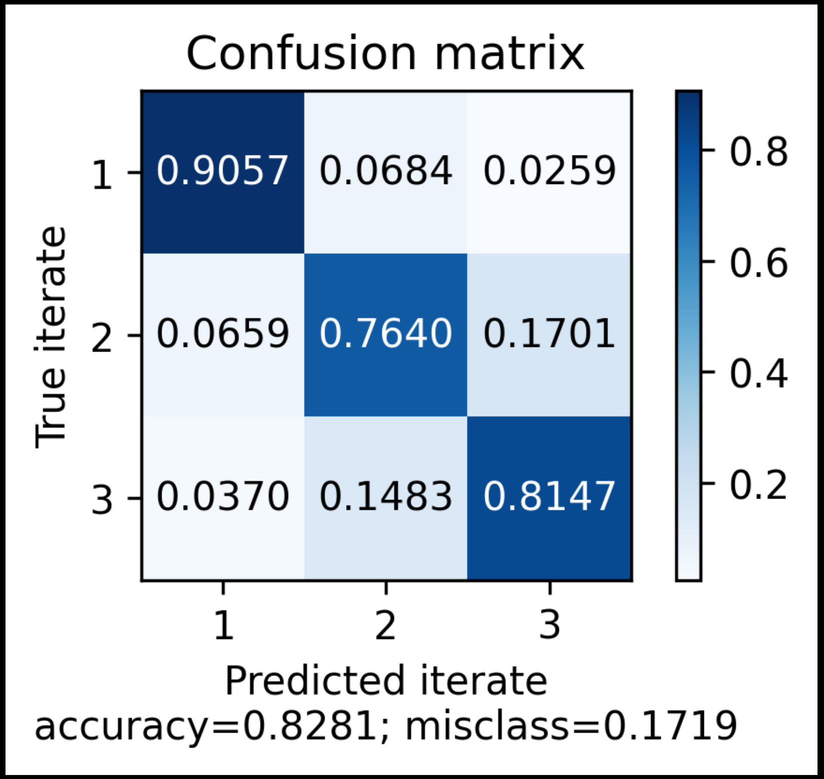
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

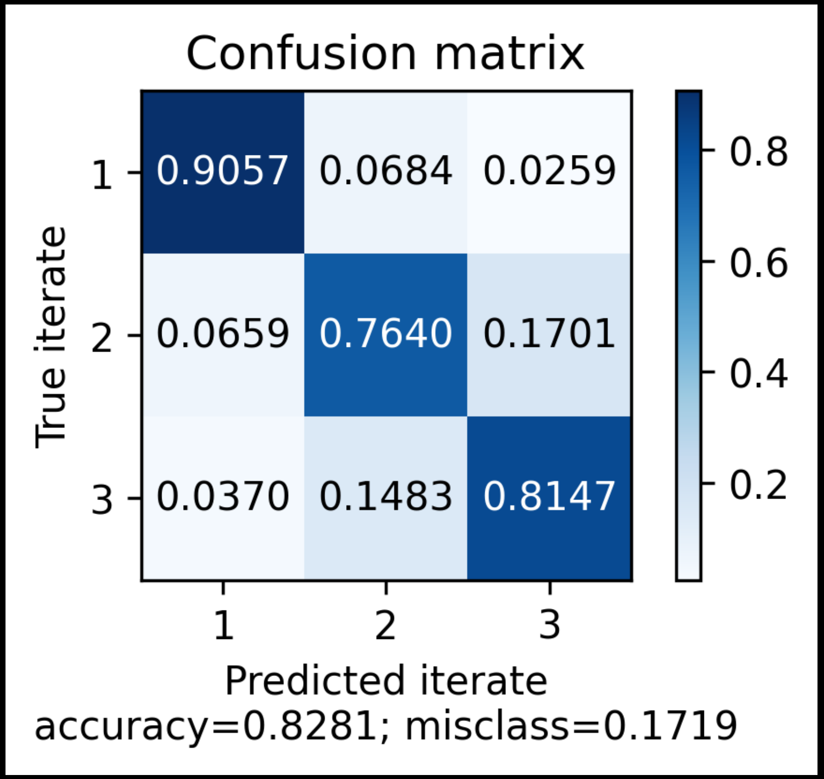
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.
- We choose k = 1,2,3 because of our empirical application.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

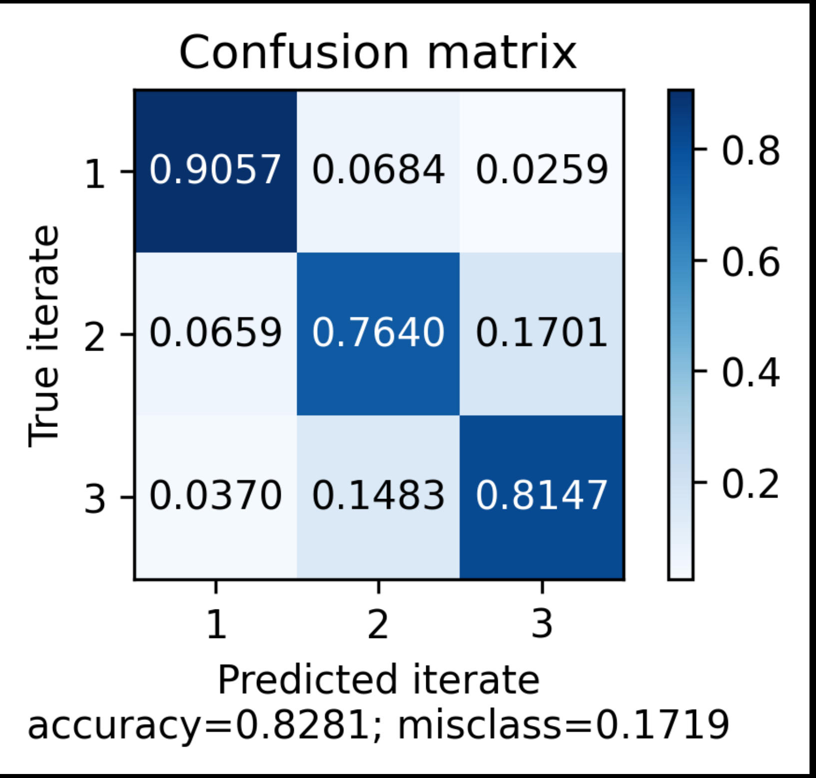
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.
- We choose k = 1,2,3 because of our empirical application.
- The figure shows the accuracy of CNN1 to estimate the iterates on test data.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

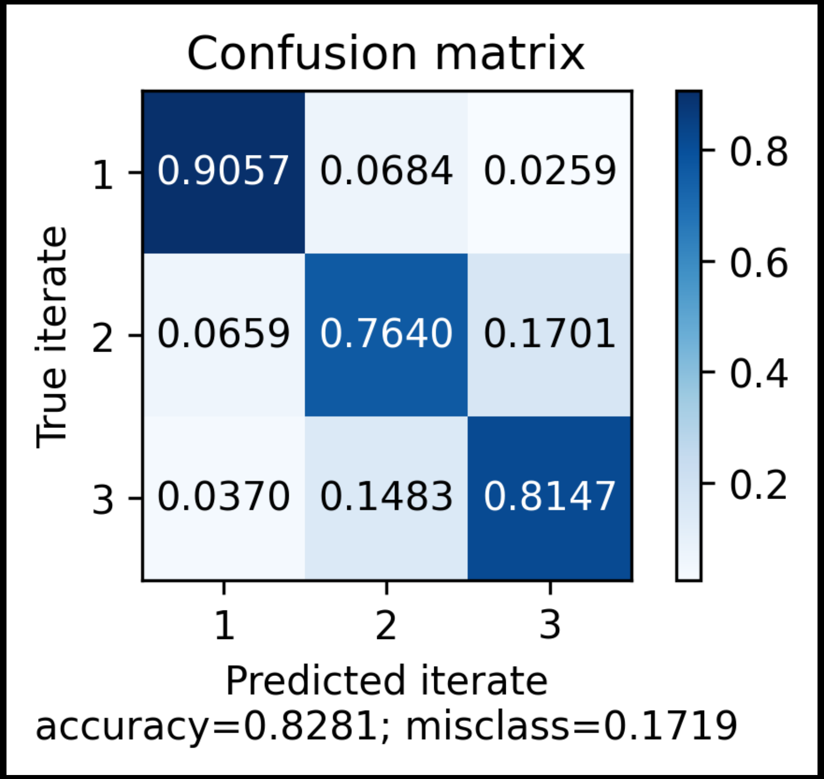
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.
- We choose k = 1,2,3 because of our empirical application.
- The figure shows the accuracy of CNN1 to estimate the iterates on test data.
- The MSE of CNN2(k) on the test set is about 0.001 for each k.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

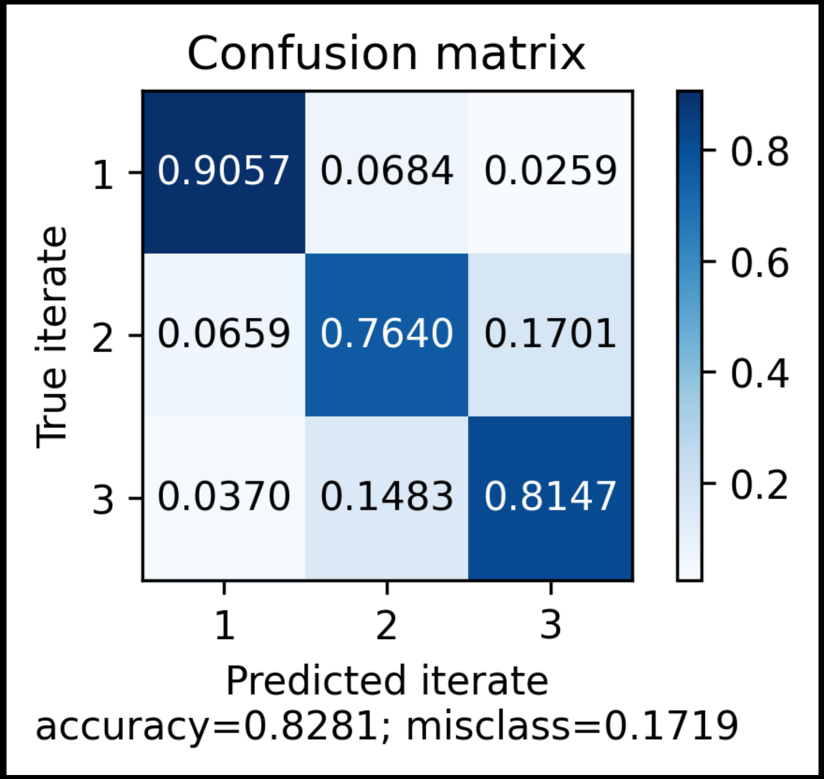
- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.
- We choose k = 1,2,3 because of our empirical application.
- The figure shows the accuracy of CNN1 to estimate the iterates on test data.
- The MSE of CNN2(k) on the test set is about 0.001 for each k.
- Since both φ* and ω are uniformly distributed in [0, 1], the MSE is quite small and the NN effective.



Accuracy of the CNN1 model used to estimate the iterates.

Training & testing

- We used a training set of 10^6 samples simulated from the model with values of the parameters $\theta = (\phi^*, \omega, n)$ which uniformly span the parameter space.
- For both steps, when simulating the series, the initial state of the system was taken randomly from a uniform distribution on $[0,1]$.
- We tested our methods on a testing set of 100,000 out of sample time series.
- We choose k = 1,2,3 because of our empirical application.
- The figure shows the accuracy of CNN1 to estimate the iterates on test data.
- The MSE of CNN2(k) on the test set is about 0.001 for each k.
- <u>Since both φ^* and ω are uniformly distributed in [0, 1], the MSE is quite small and the NN effective.</u>



Accuracy of the CNN1 model used to estimate the iterates.

Data

- Data set of US Commercial Banks and Savings and Loans Associations provided by the Federal Financial Institutions Examination Council (FFIEC)

- Quarterly balance sheet data

- We compute the leverage from the balance sheet

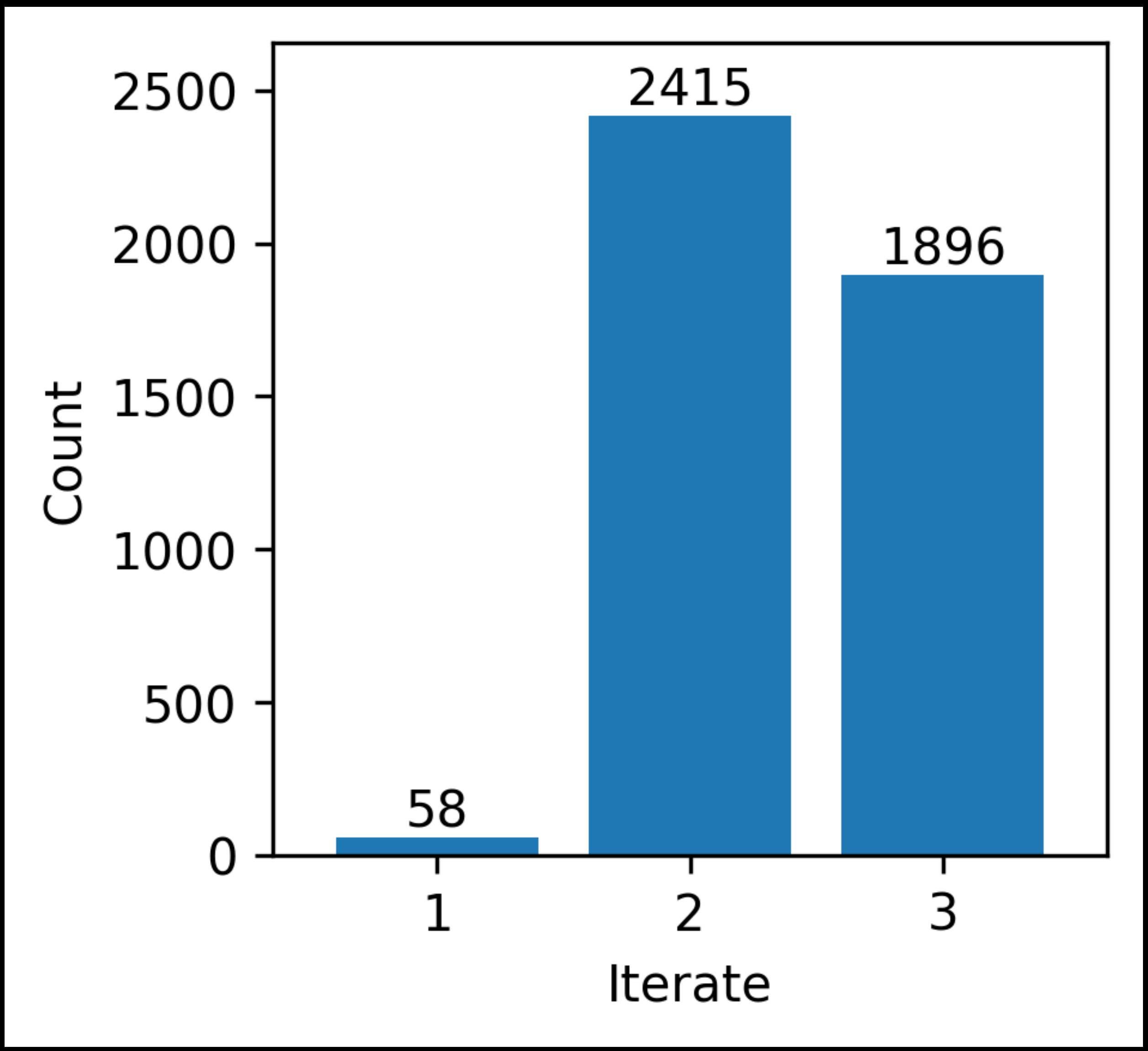
- Time period going from March 2001 to December 2014, for a total of 59 quarters.

- We have data for 5031 banks

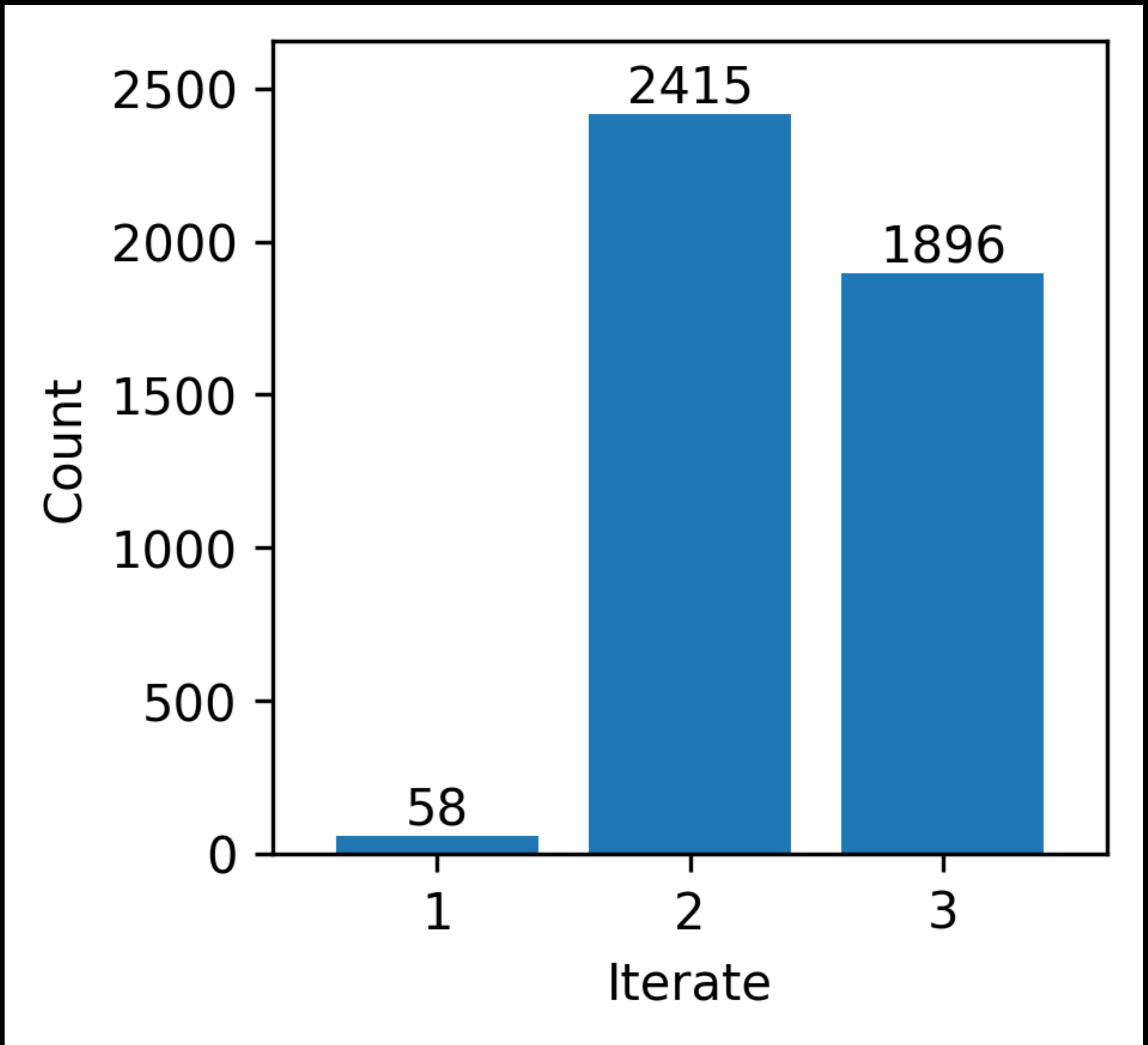
- 5031 time series of length T=59

Results

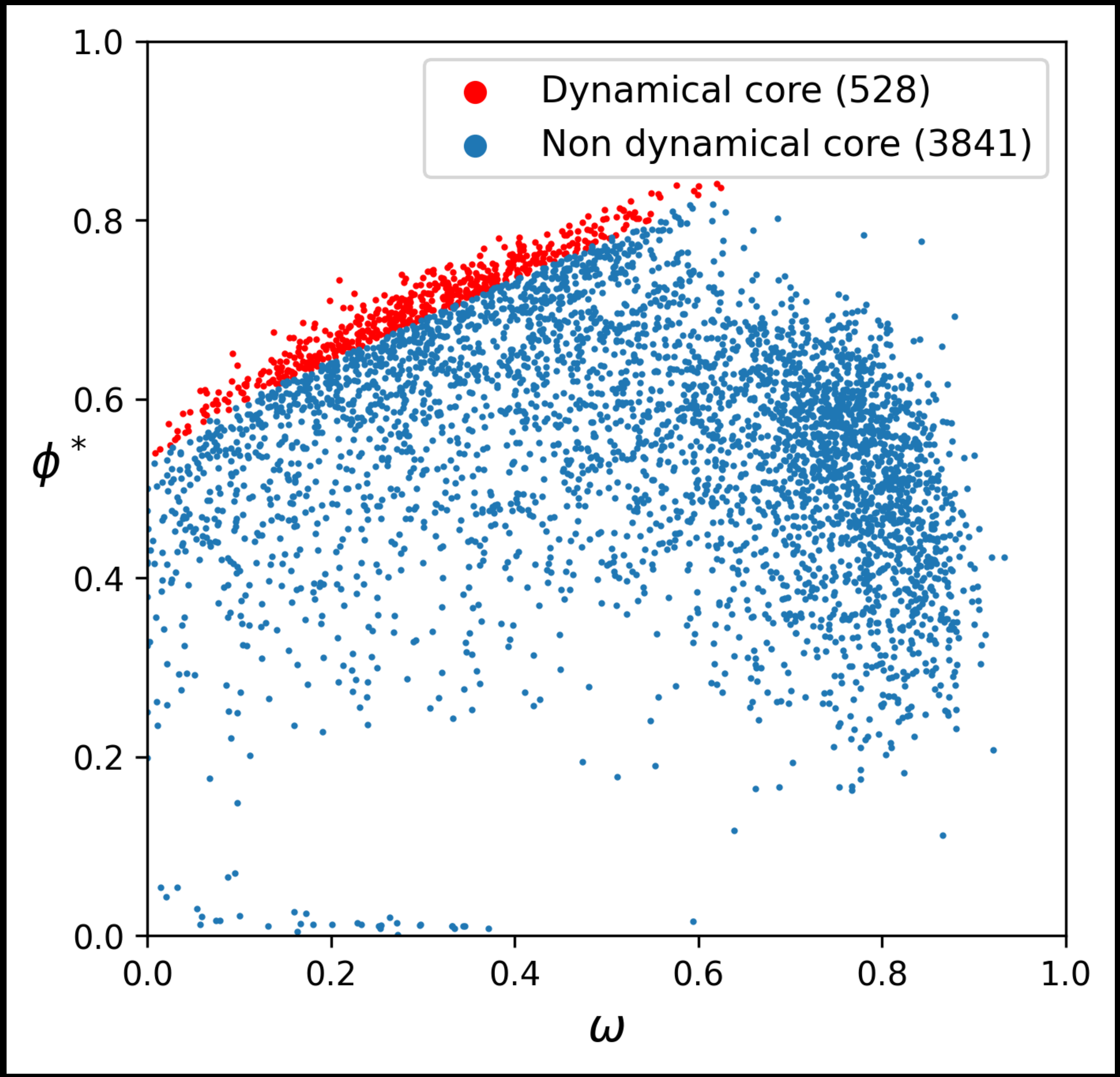
Results



Results

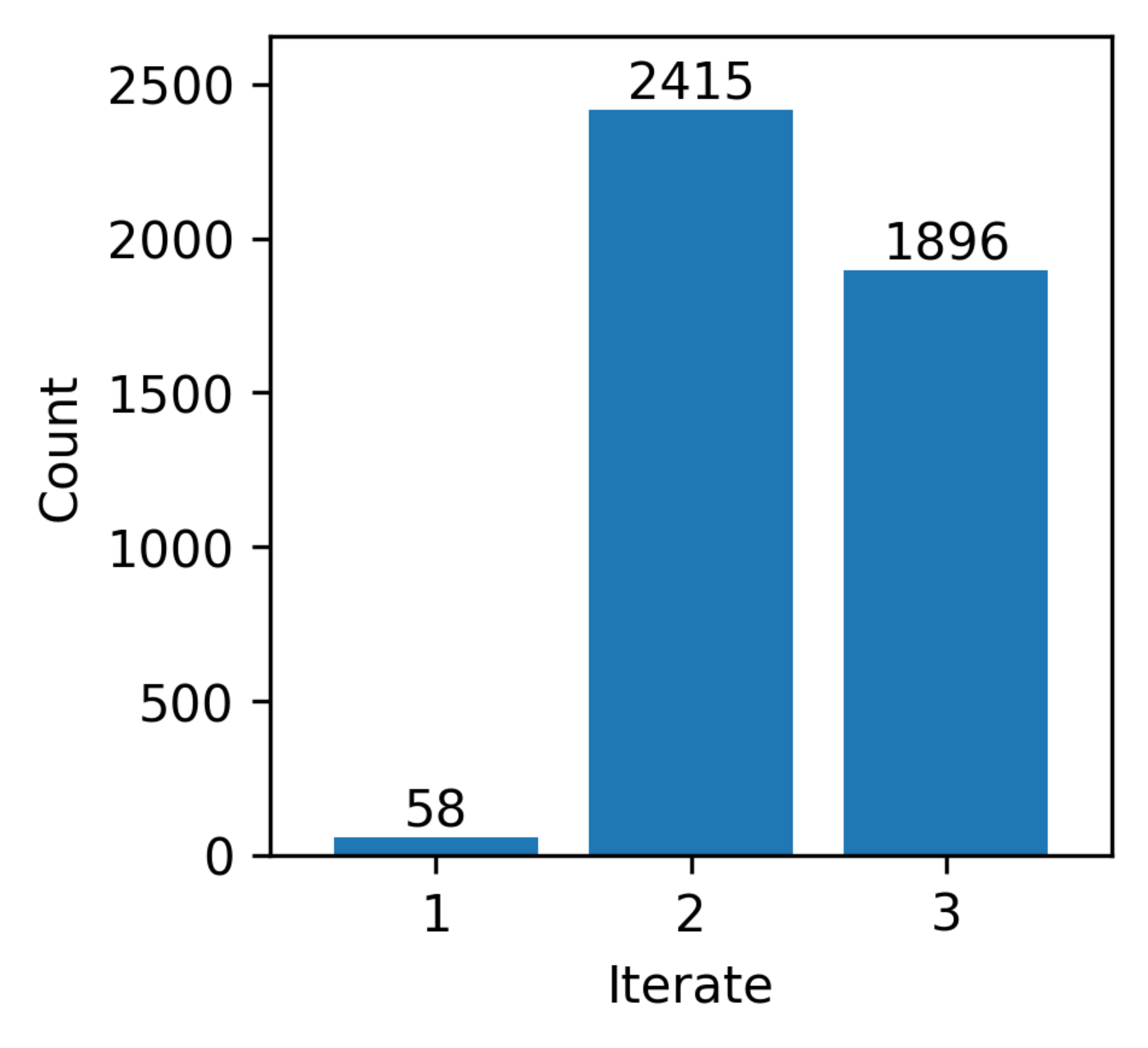


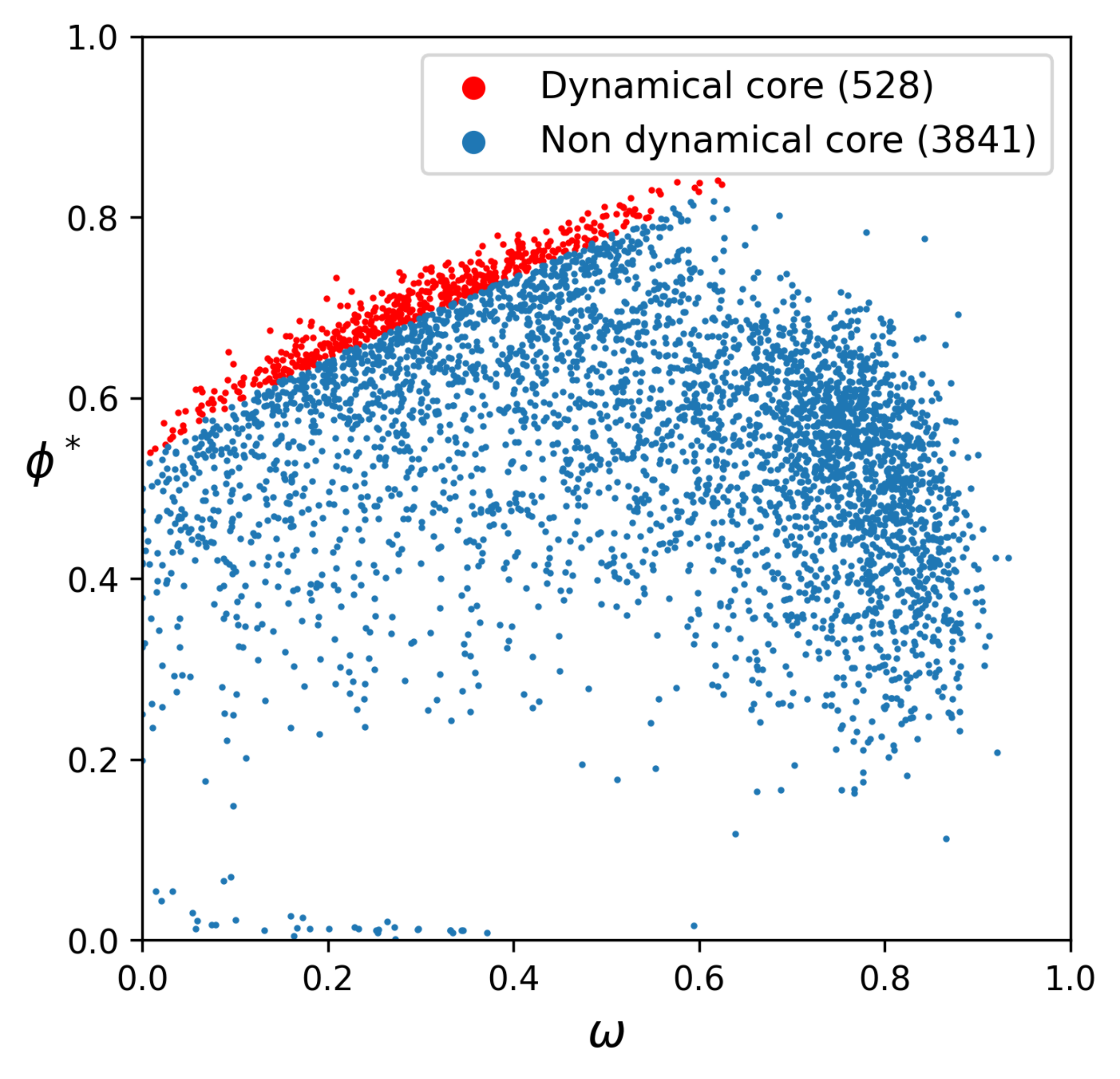
2500
2000
1500
1000
500
0
Count
2415
1896
58
1
2
3
Iterate



Dynamical core (528)
Non dynamical core (3841)
ϕ^*
ω
0.0
0.2
0.4
0.6
0.8
1.0

Results





Larger banks are found more likely in the dynamical core (and therefore have more likely a chaotic leverage dynamics)

Comparing with other methods

Chaos Detection Tree Algorithm (Toker et al. 2020) based on entropy and surrogate time series

Comparing with other methods

Simulation from the map

Chaos Detection Tree Algorithm (Toker et al. 2020) based on entropy and surrogate time series

Iterate	Series length	n	Dynamical core			Not dynamical core		
			S (%)	P (%)	C (%)	S (%)	P (%)	C (%)
1	59	5	33.5	5.14	**61.4**	**57.4**	4.81	37.7
		20	29.3	4.28	**66.5**	**69.9**	3.28	26.8
		100	24.3	6.99	**69.6**	**88.1**	2.94	8.97
	295	5	2.2	1.7	**96.1**	22.5	6.24	**71.2**
		20	0.1	1.9	**98**	43.8	10.4	**45.8**
		100	0	2.3	**97.7**	**73.8**	8.35	17.9
	590	5	0	0.7	**99.3**	13.1	6.08	**80.9**
		20	0	0.4	**99.6**	33.6	8.5	**57.9**
		100	0	0.4	**99.6**	**66.4**	8.04	25.6
	1180	5	0	0.1	**99.9**	10.9	3.44	**85.6**
		20	0	0	**100**	27.7	5.57	**66.8**
		100	0	0	**100**	**60.2**	5.29	34.5
2	59	5	**75.7**	2.17	22.2	**83.8**	1.33	14.9
		20	**80.1**	1.65	18.2	**92.8**	0.26	6.91
		100	**86.6**	1.43	11.9	**96.6**	0.56	2.81
	295	5	39.4	0	**60.6**	40.5	3.26	**56.2**
		20	38.6	0.6	**60.8**	**70**	3.7	26.3
		100	21	1.2	**77.8**	**83.9**	2.96	13.1
	590	5	27.6	0	**72.4**	25.6	3.23	**71.2**
		20	10.6	0	**89.4**	**52.3**	4.24	43.4
		100	4.8	0.6	**94.6**	**74.7**	2.88	22.4
	1180	5	11	0	**89**	13.7	2.22	**84.1**
		20	0.2	0	**99.8**	39.4	2.82	**57.7**
		100	0.2	0	**99.8**	**64.1**	2.24	33.7

Comparing with other methods

Simulation from the map

Chaos Detection Tree Algorithm (Toker et al. 2020) based on entropy and surrogate time series

For short time series (and strongly for iterated maps) Chaos Detection Tree Algorithm wrongly assign stochastic nature to chaotic time series

Iterate	Series length	n	Dynamical core			Not dynamical core		
			S (%)	P (%)	C (%)	S (%)	P (%)	C (%)
1	59	5	33.5	5.14	**61.4**	**57.4**	4.81	37.7
		20	29.3	4.28	**66.5**	**69.9**	3.28	26.8
		100	24.3	6.99	**69.6**	**88.1**	2.94	8.97
	295	5	2.2	1.7	**96.1**	22.5	6.24	**71.2**
		20	0.1	1.9	**98**	43.8	10.4	**45.8**
		100	0	2.3	**97.7**	**73.8**	8.35	17.9
	590	5	0	0.7	**99.3**	13.1	6.08	**80.9**
		20	0	0.4	**99.6**	33.6	8.5	**57.9**
		100	0	0.4	**99.6**	**66.4**	8.04	25.6
	1180	5	0	0.1	**99.9**	10.9	3.44	**85.6**
		20	0	0	**100**	27.7	5.57	**66.8**
		100	0	0	**100**	**60.2**	5.29	34.5
2	59	5	**75.7**	2.17	22.2	**83.8**	1.33	14.9
		20	**80.1**	1.65	18.2	**92.8**	0.26	6.91
		100	**86.6**	1.43	11.9	**96.6**	0.56	2.81
	295	5	39.4	0	**60.6**	40.5	3.26	**56.2**
		20	38.6	0.6	**60.8**	**70**	3.7	26.3
		100	21	1.2	**77.8**	**83.9**	2.96	13.1
	590	5	27.6	0	**72.4**	25.6	3.23	**71.2**
		20	10.6	0	**89.4**	**52.3**	4.24	43.4
		100	4.8	0.6	**94.6**	**74.7**	2.88	22.4
	1180	5	11	0	**89**	13.7	2.22	**84.1**
		20	0.2	0	**99.8**	39.4	2.82	**57.7**
		100	0.2	0	**99.8**	**64.1**	2.24	33.7

Comparing with other methods

Chaos Detection Tree Algorithm (Toker et al. 2020) based on entropy and surrogate time series

For short time series (and strongly for iterated maps) Chaos Detection Tree Algorithm wrongly assign stochastic nature to chaotic time series

Simulation from the map

Iterate	Series length	n	Dynamical core			Not dynamical core		
			S (%)	P (%)	C (%)	S (%)	P (%)	C (%)
1	59	5	33.5	5.14	**61.4**	**57.4**	4.81	37.7
		20	29.3	4.28	**66.5**	**69.9**	3.28	26.8
		100	24.3	6.99	**69.6**	**88.1**	2.94	8.97
	295	5	2.2	1.7	**96.1**	22.5	6.24	**71.2**
		20	0.1	1.9	**98**	43.8	10.4	**45.8**
		100	0	2.3	**97.7**	**73.8**	8.35	17.9
	590	5	0	0.7	**99.3**	13.1	6.08	**80.9**
		20	0	0.4	**99.6**	33.6	8.5	**57.9**
		100	0	0.4	**99.6**	**66.4**	8.04	25.6
	1180	5	0	0.1	**99.9**	10.9	3.44	**85.6**
		20	0	0	**100**	27.7	5.57	**66.8**
		100	0	0	**100**	**60.2**	5.29	34.5
2	59	5	**75.7**	2.17	22.2	**83.8**	1.33	14.9
		20	**80.1**	1.65	18.2	**92.8**	0.26	6.91
		100	**86.6**	1.43	11.9	**96.6**	0.56	2.81
	295	5	39.4	0	**60.6**	40.5	3.26	**56.2**
		20	38.6	0.6	**60.8**	**70**	3.7	26.3
		100	21	1.2	**77.8**	**83.9**	2.96	13.1
	590	5	27.6	0	**72.4**	25.6	3.23	**71.2**
		20	10.6	0	**89.4**	**52.3**	4.24	43.4
		100	4.8	0.6	**94.6**	**74.7**	2.88	22.4
	1180	5	11	0	**89**	13.7	2.22	**84.1**
		20	0.2	0	**99.8**	39.4	2.82	**57.7**
		100	0.2	0	**99.8**	**64.1**	2.24	33.7

Real bank data

	Periodic	Chaotic	Stochastic
Non dynamical core	382 (9.98%)	648 (16.93%)	2798 (73.09%)
Dynamical core	107 (20.34%)	176 (33.46%)	243 (46.20%)

Table 2: Number of banks by classes.

Some conclusions

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form

- Two approaches: direct (the input of the DNN is the time series) vs indirect (the input of the DNN are statistics of an auxiliary model)

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form
- Two approaches: direct (the input of the DNN is the time series) vs indirect (the input of the DNN are statistics of an auxiliary model)
- The indirect DNN does not require a machine for each time series length

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form
- Two approaches: direct (the input of the DNN is the time series) vs indirect (the input of the DNN are statistics of an auxiliary model)
- The indirect DNN does not require a machine for each time series length
- We show that DNN is effective in estimating parameters of short time series from dynamical systems with heteroschedastic noise

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form

- Two approaches: direct (the input of the DNN is the time series) vs indirect (the input of the DNN are statistics of an auxiliary model)

- The indirect DNN does not require a machine for each time series length

- We show that DNN is effective in estimating parameters of short time series from dynamical systems with heteroschedastic noise

- The proposed method is especially useful when we are not sure we are observing the dynamical system every elementary time step

Some conclusions

- Deep Neural Networks can be very effective in estimating parameters of a (time series) model, especially when the likelihood function is not known in closed form
- Two approaches: direct (the input of the DNN is the time series) vs indirect (the input of the DNN are statistics of an auxiliary model)
- The indirect DNN does not require a machine for each time series length
- We show that DNN is effective in estimating parameters of short time series from dynamical systems with heteroschedastic noise
- The proposed method is especially useful when we are not sure we are observing the dynamical system every elementary time step
- Financial application: by using the DNN estimation method, we show that for a sizeable fraction of (large) banks the leverage dynamics is chaotic