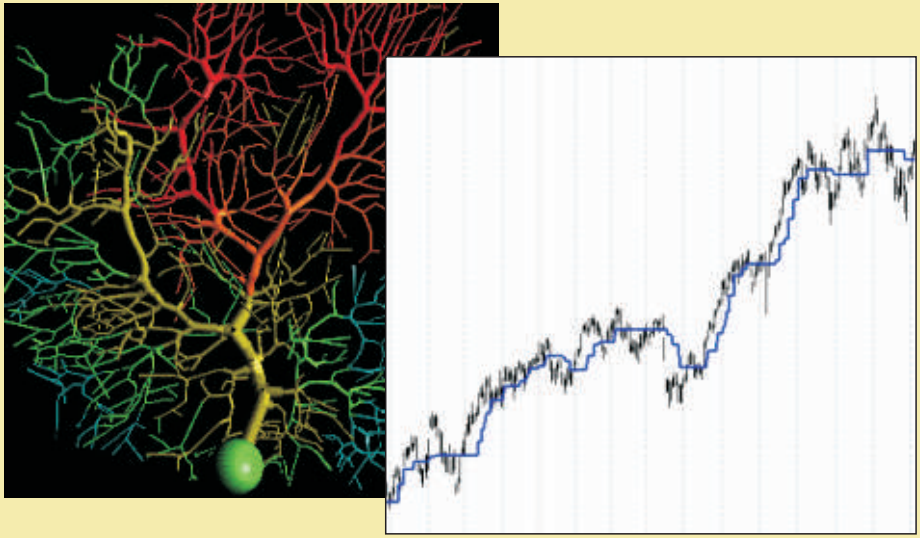
Forecast of stock prices by using neural networks in comparison to log-normal estimation

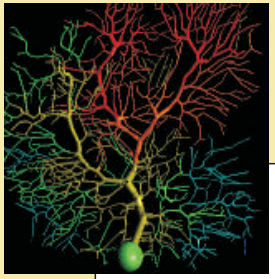


Stefan Markus Giebel
Faculté de Droit, d'Economie et de Finance, Uni Luxembourg

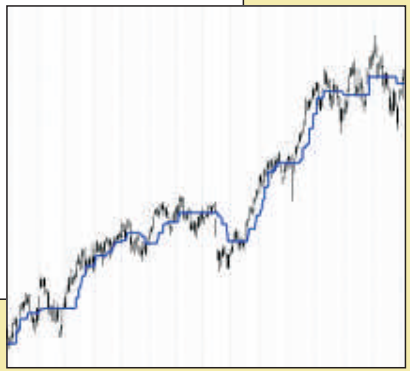
Martin Rainer
Institute of Applied Mathematics, METU Ankara

UNIVERSITÉ DU LUXEMBOURG

Overview



1. Forecast
2. Neural networks
3. Log-Normal estimation
4. Results
5. Outlook



Forecast

Predictions are always difficult, especially if they involve the future.

Vorhersagen sind immer schwierig. Insbesondere wenn sie die Zukunft betreffen. Niels Bohr (1885-1962)

identify beforehand

s: Foresight

http://www.hellenica.de/Griechenland/Mythos/Kassandra.html

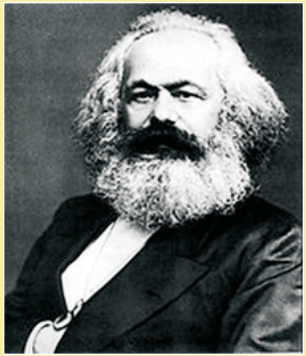
Forecast

“Recognizing” requires that there are causal links between the past and the future.

➡ Continuation of the recent development

A developed discipline in science has to use mathematics.
Eine Wissenschaft ist erst dann entwickelt, wenn sie sich der Mathematik bedient.
Karl Marx (1818-1883)

➡ Description of the development in mathematics



Forecast for shares

History: 2008 Prognosis: January - May 2009

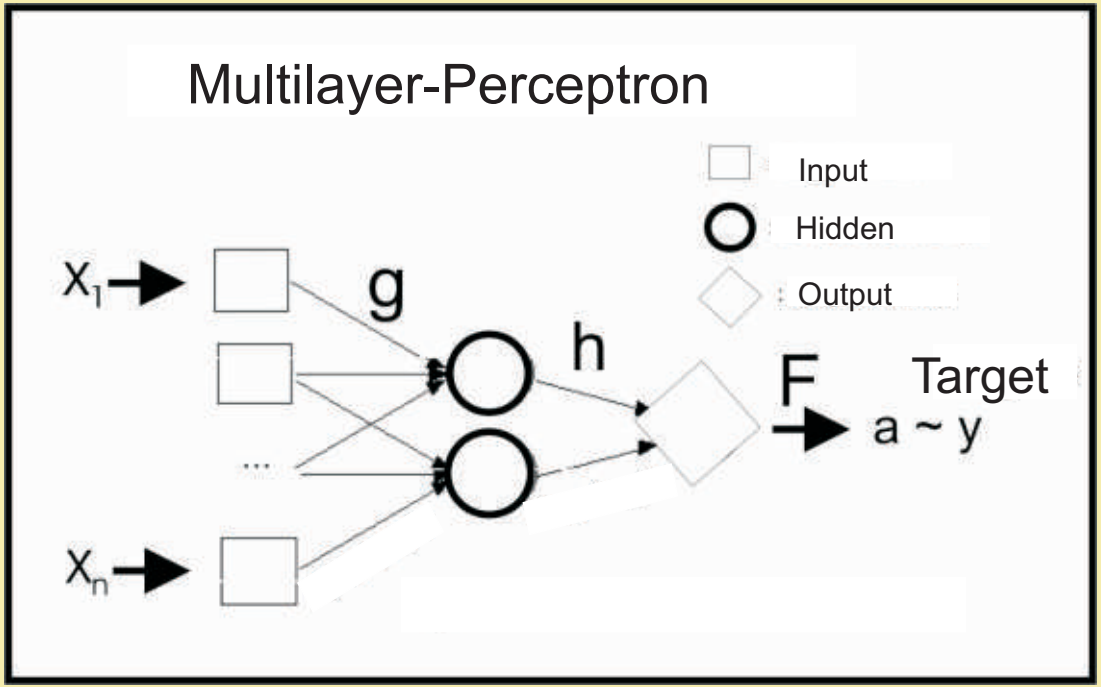
http://www.dradio.de/images/39512/landscape/

Difficulties

- Crisis
- War
- Political conflicts
- Terrorist attacks
- Religious conflicts
- Consumption
- Unemployement

Neural networks

Supervised neuronal networks



Often used only for fuzzy-problems
Example: „1": „up“ and „0": „down/equal“

Neural networks Algorithm

Step 1.

$$E = \sum_{k=1}^{N} (\tilde{y}_k - y_k)^2$$

Differences between estimation and reality

Aim: Minimum of error E

Step 2.

$$z_j = g(\sum_{i=1}^{n} w_i \cdot x_i)$$

Hidden layer (Sigmoid-function)

Step 3.

$$a = h(\sum_{j=1}^{m} u_j \cdot z_j)$$

Output layer (Sigmoid-function)

Step 4.

$$\tilde{y} = F(a)$$

Function for output (Interpretation)

Step 5.

$$w_i^{new} = w_i^{old} + \alpha \cdot \Delta w_i$$

Initialisation of raise in hidden layer

back propagation

At start: random weights

Step 6.

$$u_j^{new} = u_j^{old} + \alpha \cdot \Delta u_j$$

Initialisation of raise in output layer

Log-normal estimation

log-normal diffusion process

asset price $S > 0$

$$\frac{dS}{S} = \mu dt + \sigma dW$$

$$y := f(S) = \ln S$$

$$dy = \mu_y dt + \sigma_y dW$$

$$\mu_y := \mu - \frac{\sigma^2}{2}$$

$$\sigma_y := \sigma$$

time-integrated white noise

$$\int_0^t dW \sim N(0, \sqrt{t})$$ normally distributed

Log-normal estimation

solution of the SDE

$$
\begin{aligned}
y(t) &= y(0) + \mu_y t + \sigma_y \int_0^t dW \\
&= y(0) + \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma \int_0^t dW \\
S(t) &= e^{y(t)} \\
&= S(0) e^{\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma \int_0^t dW} \\
E[S(t)] &= E[e^{y(t) + \frac{1}{2}\sigma^2 t}] \\
&= S(0) e^{\mu t} \quad .
\end{aligned}
$$

Log-normal estimation

$k\sigma$ bound

$$y_{k\sigma}(t) = y(0) + \left(\mu - \frac{\sigma^2}{2}\right)t + k\sigma\sqrt{t}$$

$$\begin{aligned} S_{k\sigma}(t) &= S(0)e^{\left(\mu-\frac{\sigma^2}{2}\right)t+k\sigma\sqrt{t}} \\ &= S_{med}(t)e^{k\sigma\sqrt{t}} \quad . \end{aligned}$$

$k = 0$ yields the median curve

$$S_{med}(t) = S(0)e^{\left(\mu-\frac{\sigma^2}{2}\right)t}$$

forward (expected) price $F(t) := E[S(t)]$

$$F(t) = S_{med}(t)e^{\frac{1}{2}\sigma^2 t}$$

Parameter estimation

discretization

$$dt_i := t_i - t_{i-1}$$

$$dS_i := S_i - S_{i-1} \quad \text{return} \quad \frac{dS_i}{S_{i-1}} := \frac{S_i}{S_{i-1}} - 1$$

$$dy_i := \ln \frac{S_i}{S_{i-1}}$$

(dt_i-weighted) estimators

$$\hat{\mu} := \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dt_i \left[\frac{1}{dt_i} \frac{dS_i}{S_{i-1}} \right] \qquad \hat{\mu}_y := \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dy_i$$

$$= \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n \frac{dS_i}{S_{i-1}}$$

$$\hat{\sigma}^2 := \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n dt_i \left[\frac{1}{dt_i} \left(\frac{dS_i}{S_{i-1}} \right)^2 \right]$$

$$= \frac{1}{\sum_{i=1}^n dt_i} \sum_{i=1}^n \left(\frac{dS_i}{S_{i-1}} \right)^2$$

Results

Neural networks (number of hidden neurons: Log(n)+2, error<0,1€)

Shares	R (whole sample)	R(prediction)	R(history)
Telekom	0.810478	0.737837	0.759025
Merck	0.82425	0.207414	0.895684
Solarworld	0.111865	0.000000	0.11978
Adidas	0.136267	0.000000	0.145877
Lufthansa	0.142905	0.000000	0.151364
Lukoil	0.817559	-0.60211	0.810965

Raw material	R (whole sample)	R(prediction)	R(history)
Gold	-0.0522381	0.550732	-0.100322

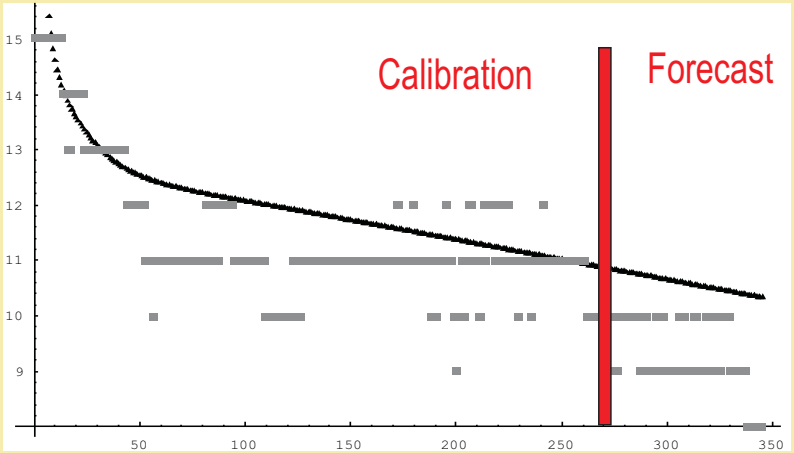
Log-normal estimation

Shares	R (whole sample)	R(prediction)	R(history)
Telekom	0.80811722	0.84651056	0.65511935
Merck	0,85899285	0.19996539	0.79160877
Solarworld	0.68058422	-0.57964574	0.11978
Adidas	0.86983954	0.02700922	0.77273266
Lufthansa	0.89928454	0.42845253	0.79888028
Lukoil	0.73229756	-0.84188944	0.60332597

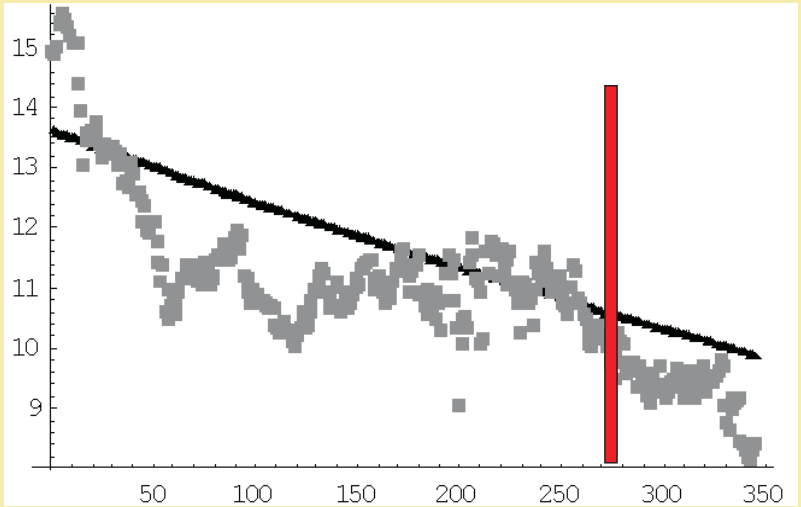
Raw material	R (whole sample)	R(prediction)	R(history)
Gold	-0.18802864	0.25678479	-0.67329928

Results

Neural network

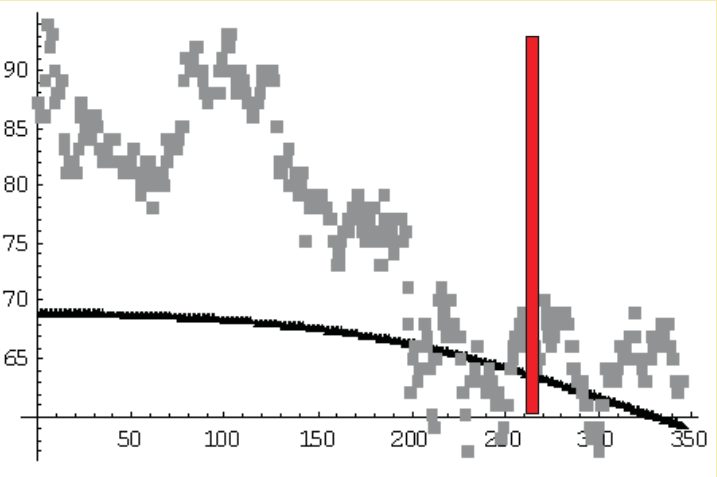


Log-normal estimation

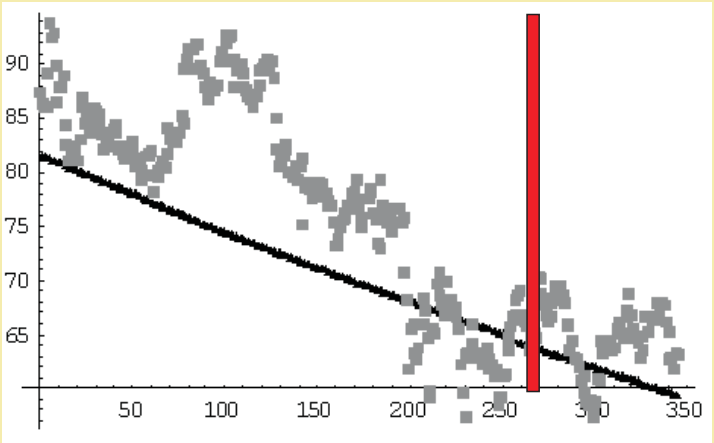


Results

Neural network

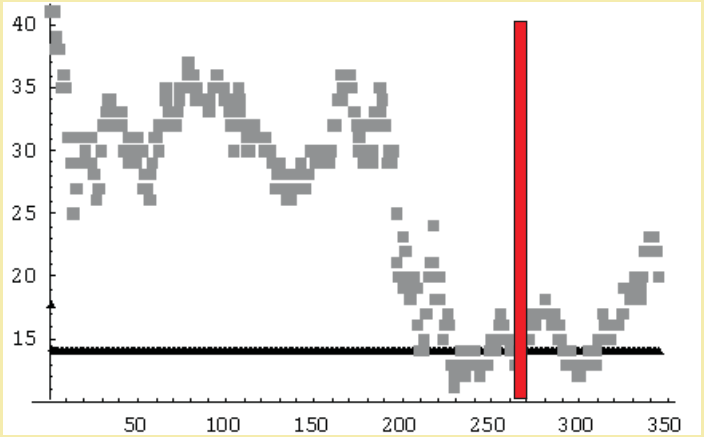


Log-normal estimation

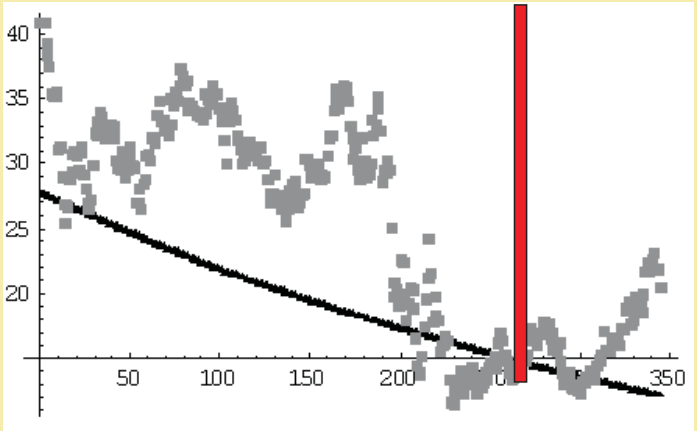


Results

Neural network

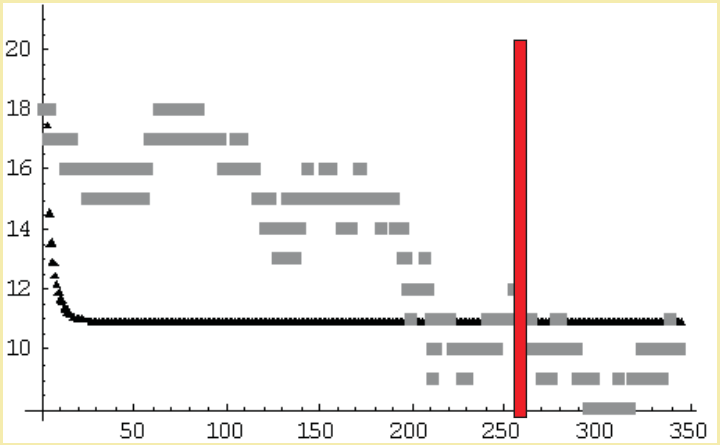


Log-normal estimation

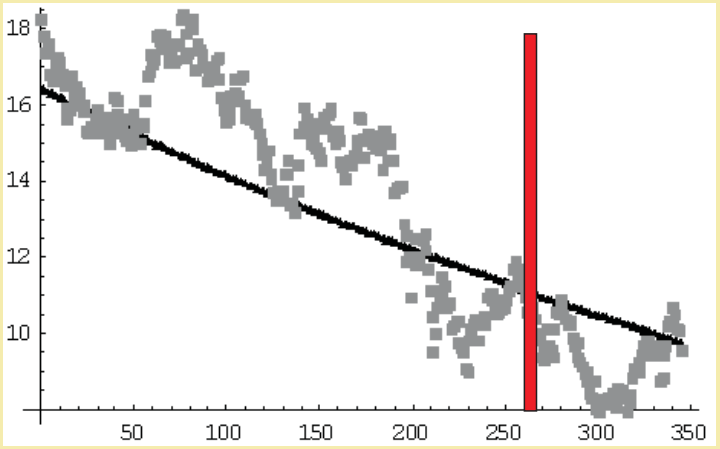


Results

Neural network

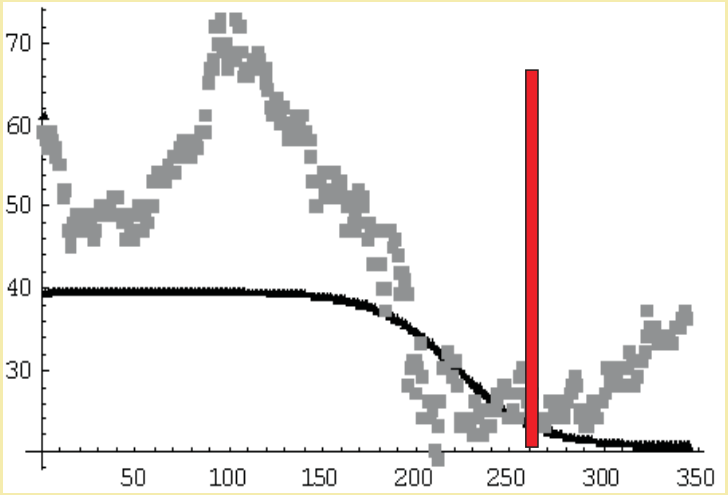


Log-normal estimation

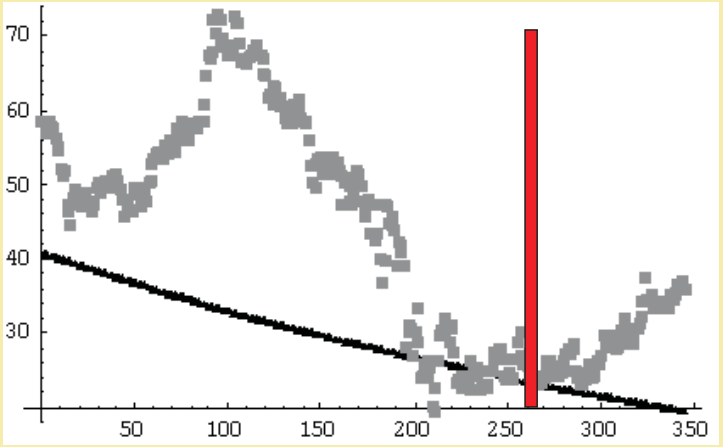


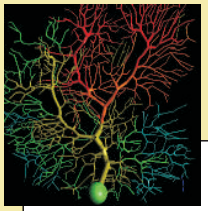
Results

Neural network



Log-normal estimation



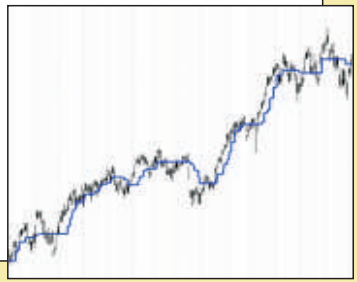


Results

- Neural networks are suitable to minimize the error

- Neural networks are also suitable for metric targets, not only for fuzzy problems

- Log-normal estimation is more suitable to get the tendency



Outlook:
Incomplete Markets

bid-ask spread

$$P_{min}(t) < P(t) < P_{max}(t)$$
$$P_{i,min} < P_i < P_{i,max}$$

martingale pricing measure Q

→ forward value (as seen from $t = 0$) $F_0(t) = E_Q S(t)$

interval of volatilities

$$\hat{\sigma}_{min} < \hat{\sigma} < \hat{\sigma}_{max}$$

→ market-price-of-risk

$$\lambda_{min} < \lambda = \frac{\mu - r}{\sigma} < \lambda_{max}$$

→ equivalent martingale measures $Q(\lambda)$
(via Girsanov transformation)

Outlook: Incomplete Markets

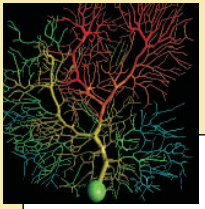
if the market is incomplete due to uncertain volatility σ

1. estimate volatility with error bounds consistent with the bid-ask spread (e.g. using Bayesian techniques such as Markov chain Monte Carlo for a prior distribution of σ)

2. apply an uncertain volatility model, which treats the volatility as the corresponding random variable.

3. evaluate different scenarios (best case, worst case, minimal entropy) for price depending on the uncertain parameter (the volatility).

...similarly for other parameters



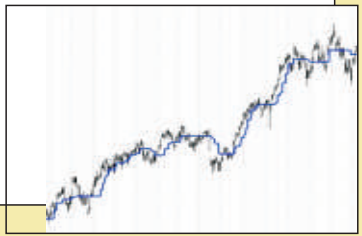
Outlook

- Combination of stochastic processes and neural networks (using parameter estimation for start)

- Maximize a dependency measure, e.g. correlation R rather than the error

- Using more market variables: volatility, interest rate etc.

-Using stochastic process with more parameters



Stefan Markus Giebel
Faculté de Droit, d'Economie et de Finance,
Uni Luxembourg

Martin Rainer
Institute of Applied Mathematics, METU Ankara

uni.lu
UNIVERSITÉ DU
LUXEMBOURG