Artificial Neural Networks in financial applications

February 2003

Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

Artificial Neural Networks (ANN)

- ANN are:

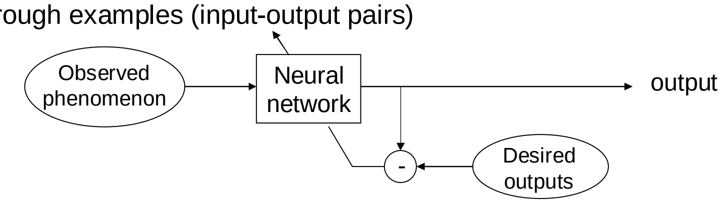
regression and/or *classification*

models with some nice properties
(compared to other models)

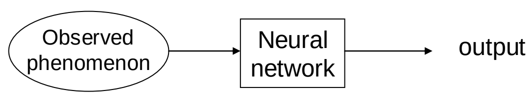
  - Universal approximation
  - Learning from examples without assumption about the distributions
  - ~~Easy~~ possible scalability to large dimensions

Learning in ANN

- Supervised learning: building an input-output relation known through examples (input-output pairs)



- Unsupervised learning: modeling a property of the input data (for example density)



Content

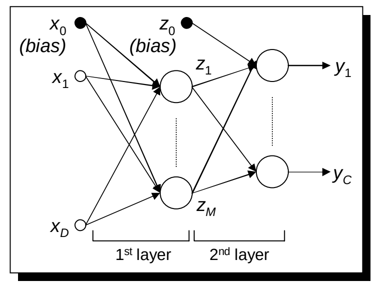
- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

Multi-layer perceptron (MLP)

$$y_k(\boldsymbol{x}) = h\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{h}\left(\boldsymbol{w}^{(2)} \boldsymbol{g}\left(\boldsymbol{w}^{(1)} \boldsymbol{x}\right)\right)$$

parameters



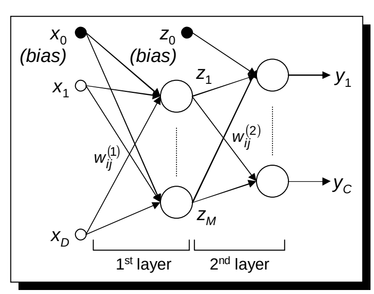
- Convention: 2 layers of weights (in literature: sometimes 3 layers of *units* or *neurons*)
- *g* and *h* continuous, bounded, non-linear functions (tanh, sigmoid)
- *h* can be linear but not *g* (otherwise only one layer)

Learning in MLP

- Learning =
  - definition of an error criterion *E*
  - evaluation of derivatives of *E* w.r.t. parameters *w*
  - adjustments of parameters *w* according to derivatives
- Error criterion (for one output):

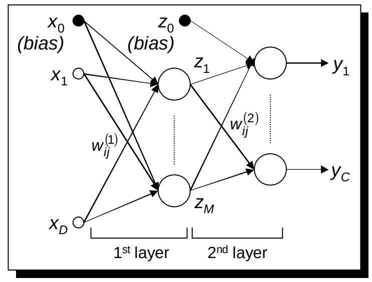
$$E = \sum_{p=1}^{P} \left\| y^p - d^p \right\|^2$$

Training examples



Learning in MLP

- “Back-propagation”: $\dfrac{\partial E}{\partial w_{ij}}$
  - For last layer $w_{ij}^{(2)}$: easy
  - For other layers: computed according to derivatives of *next* layer
    → *back-propagation* of derivatives



- More elaborated learning methods:
  - Conjugate gradients
  - Levenberg-Marquardt
  - …

Universal approximation property

- *“A 2-layer MLP can approximate arbitrarily well any (functional) continuous mapping, provided the number M of hidden units is sufficiently large”*

- But:
  - what about generalization?
  - what about the number *M* of hidden units?
  - What about initialization and local minima?
  - Learning is slow, difficult, needs expertise, etc.

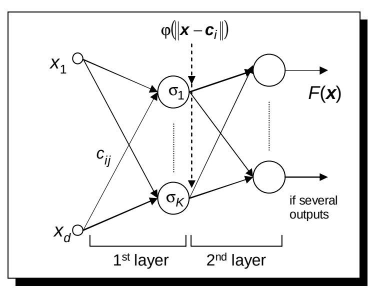
Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

Radial-Basis Function Networks (RBFN)

$$F(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|)$$

$$\varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_i^2}\right)$$



- Recommended:

$$F(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_i^2}\right) + \sum_{i=1}^{D} w'_i \, x_i + w'_0$$

because $K << P$

- RBFN are a generalization of the
  - interpolation problem
  - regularization problem

RBFN: learning strategies

$$F(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|) \qquad \varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_i^2}\right)$$

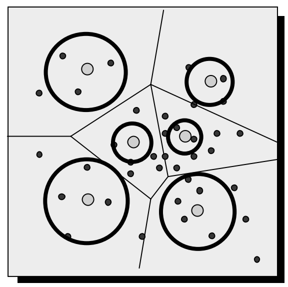
- Parameters to be determined: $\boldsymbol{c}_i$, σ_i, w_i
- Traditional learning strategy: splitted computation
  1. centers $\boldsymbol{c}_i$
  2. widths σ_i
  3. weights w_i

RBFN: computation of centers

- Idea: centers $\boldsymbol{c}_i$ must have the (density) properties of learning points $\boldsymbol{x}^k$
  $\rightarrow$ vector quantization
  - selected at random (in learning set)
  - competitive learning
  - frequency-sensitive learning
  - Kohonen maps
- This phase only uses the $\boldsymbol{x}^k$ information, not the t^k

RBFN: computation of widths

- Universal approximation property: valid with identical widths
- In practice (limited learning set): variable widths σ_i
- Idea: RBFN use *local* clusters



- choose σ_i according to standard deviation of clusters

RFBN: computation of weights

$$F(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|) \qquad \varphi(\|\boldsymbol{x} - \boldsymbol{c}_i\|) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_i^2}\right)$$

constants !

- Problem becomes linear !
- Solution of least square criterion leads to

$$E(F) = \frac{1}{2P} \sum_{p=1}^{P} \left(t^p - F(\boldsymbol{x}^p)\right)$$

$$\boldsymbol{w} = \boldsymbol{\Phi}^+ \boldsymbol{t} = \left(\boldsymbol{\Phi}^T \boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^T$$

where

$$\boldsymbol{\Phi} \equiv \varphi_{ki} = \varphi\left(\left\|\boldsymbol{x}^k - \boldsymbol{c}_i\right\|\right)$$

- In practise: use SVD !

RBFN: gradient descent

$$F(\boldsymbol{x}) = \sum_{i=1}^{K} w_i \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{c}_i\|^2}{2\sigma_i^2} \right)$$

1 2 3

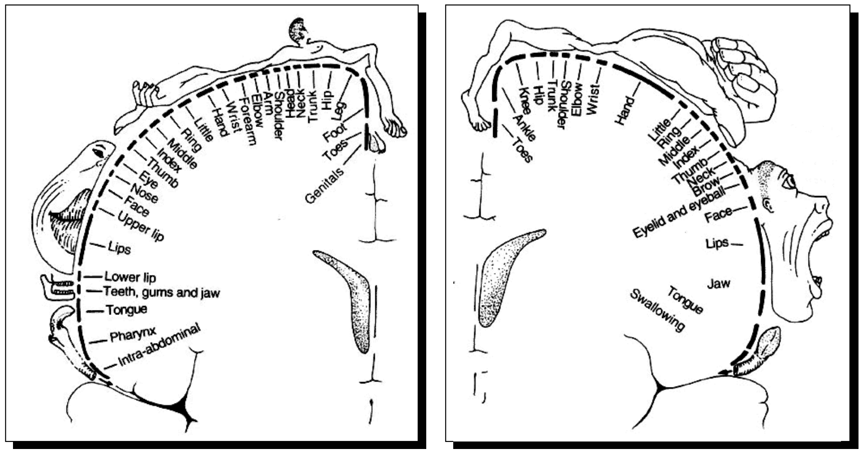
- 3-steps method: supervised unsupervised

- Once $\boldsymbol{c}_i$, σ_i, w_i have been set by the previous method, possibility of gradient descent on *all* parameters
- Some improvement, but
  - learning speed
  - local minima
  - risk of non-local basis functions
  - etc.

Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

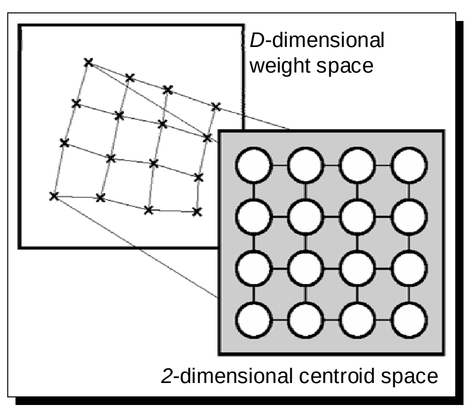
Self-organizing maps (SOM): biological inspiration



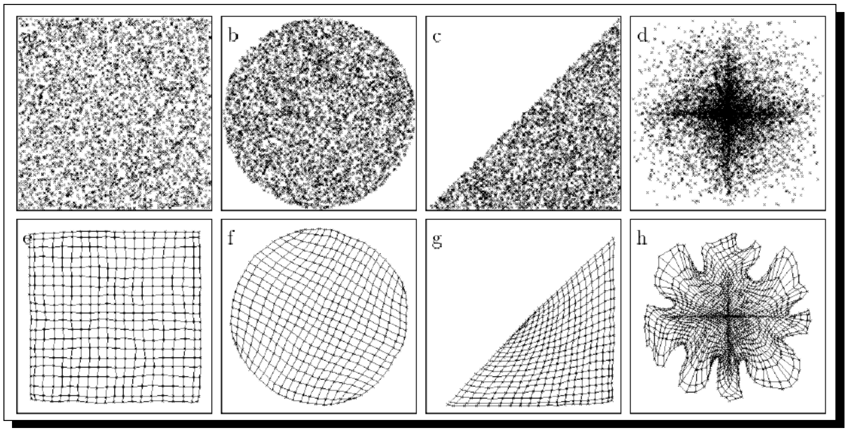
Human sensory and motor maps

Kohonen map

- Vector quantization
  and
  topological ordering
- 2-dimensional grid
  for visualisation !
- Also possible:
  - 1-dimensional string
    (sometimes used)
  - x-dimensional cube
    (rarely used)



Examples after convergence



2-dimensional input spaces!

SOM equations

- Choice of the winner

$$y_k = \max_i \left(f\left(\mathbf{w}_i^{\mathsf{T}} \mathbf{x} \right) \right)$$

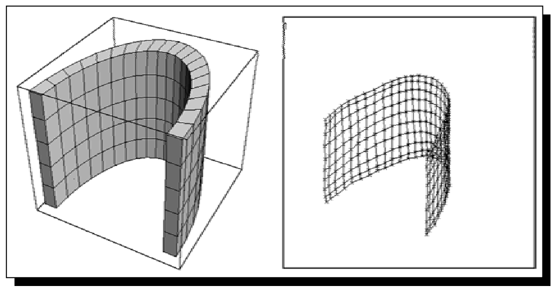
- Adaptation of weights

$$\mathbf{w}_j(t+1) = \begin{cases} \mathbf{w}_j(t) + \alpha(t)\left(\mathbf{x} - \mathbf{w}_j(t)\right) & \text{if } d(k,j) < r(t) \\ \mathbf{w}_j(t) & \text{otherwise} \end{cases}$$

- $r(t)$ is made decreasing with time

SOM used as data analysis tool

- Mapping (projection) of a continuous distribution to a discrete set (the centroids)



- After learning: nearest neighbour rule in the input space

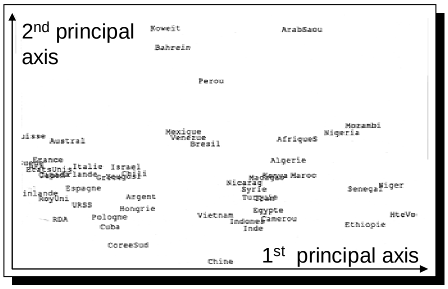
Macroeconomical data (1/2)

- Factors: annual increase (%), infant mortality (‰), illiteracy ratio (%), scool attendance (%), GIP, annual GIP increase (%)

Afrique du sud	2.9	89.0	50.0	19.0	2680.0	-2.9
Algerie	2.9	114.0	58.5	47.9	2266.0	0.1
Arabie Saoudite	4.2	111.0	75.4	39.7	10827.0	-10.8
Argentine	1.2	44.0	5.3	69.5	2264.0	2.0
Australie	1.3	10.4	0.0	86.0	9938.0	-1.2
Bahrein	3.8	57.0	20.9	76.3	8960.0	-10.1
Bresil	2.2	75.0	23.9	62.3	1853.0	-3.9
Cameroun	2.4	106.0	55.1	44.5	939.0	6.5
Canada	1.0	10.0	0.9	93.0	9857.0	3.0
Chili	1.7	42.0	7.7	85.2	1853.0	-0.5
Chine	1.4	71.0	31.0	44.0	231.0	10.0
Coree du Sud	1.6	33.0	8.3	82.1	1716.0	9.3
Cuba	0.7	16.8	8.9	78.7	2046.0	5.2
Egypte	2.7	74.0	58.1	45.8	626.0	6.0
Espagne	0.9	9.6	6.8	88.0	5316.0	2.3
Etats Unis	1.0	11.2	0.8	91.0	11732.0	3.3
Ethiopie	2.7	145.0	85.0	23.1	140.0	7.4
Finlande	0.6	6.5	0.6	98.0	10286.0	5.1
France	0.4	9.1	1.2	86.0	11326.0	0.5
Grece	1.1	15.1	11.7	81.0	4060.0	0.3
Haute Volta	1.7	208.0	88.6	7.6	240.0	3.6
Hongrie	0.0	20.0	0.9	42.0	1963.0	0.9
Inde	1.8	121.0	57.6	71.7	260.0	6.5
Indonesie	1.7	99.0	32.3	41.3	488.0	5.0
Iran	2.7	105.0	57.2	57.9	2346.0	5.2
Irlande	1.2	11.0	1.0	93.0	4813.0	0.5
Israel	2.2	15.0	6.7	74.0	4531.0	1.1
Italie	0.4	13.0	4.6	73.0	6869.0	-1.2
Japon	0.9	6.6	0.8	92.0	9704.0	3.0
Kenya	4.0	85.0	52.9	59.3	376.0	3.6
Kowait	6.5	33.0	35.9	73.0	20900.0	-0.5
Madagascar	2.7	69.0	38.8	30.4	259.0	0.9
Maroc	2.5	104.0	65.0	34.9	864.0	0.6
Mali	2.8	152.0	86.5	16.7	190.0	1.5
Mexique	2.6	54.0	17.3	70.1	1900.0	-4.6
Mozambique	2.7	150.0	66.8	16.1	155.0	-6.9
Nicaragua	4.4	88.0	10.0	52.5	760.0	5.1
Niger	3.0	143.0	90.2	9.2	330.0	2.5
Nigeria	3.3	133.0	66.0	29.3	807.0	-4.0
Perou	2.8	85.0	19.3	72.0	997.0	-12.0
Pologne	0.9	24.6	0.6	77.0	2545.0	4.5
RDA	-0.2	11.4	0.5	89.0	5103.0	4.2
RFA	-0.1	12.0	0.7	87.0	12176.0	1.0
Royaume Uni	-0.1	10.1	0.8	83.0	8655.0	3.5
Sénégal	2.6	152.0	77.5	19.2	430.0	2.3
Suède	0.1	7.0	0.6	85.0	13920.0	1.8
Suisse	0.6	8.0	0.9	88.0	15522.0	-0.1
Syrie	3.8	60.0	46.3	50.7	1717.0	5.8
Turquie	2.1	119.0	31.2	42.0	1491.0	3.0
URSS	0.9	28.8	0.8	96.0	4562.0	4.0
Venezuela	3.0	40.0	19.0	57.7	3823.0	-2.0
Vietnam	2.3	97.0	13.0	59.5	220.0	5.2
Yougoslavie	0.9	31.0	13.2	83.0	2067.0	-1.3

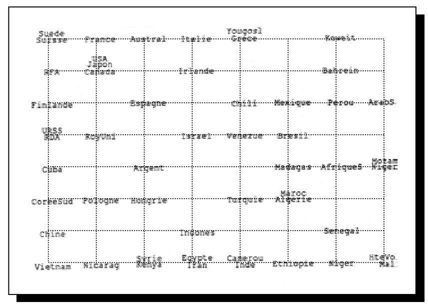
From “Data analysis: How to compare Kohonen neural networks to other techniques?”, F. Blayo, P. Demartines, in IWANN'91 (Granada, Spain) proceedings, Springer-Verlag Lecture Notes in Computer Sciences 540, pp. 469-476.

Macroeconomical data (2/2)



PCA

Kohonen



Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

of units or parameters in ANN

- # units: illustration of bias-variance dilemma
  - # units increases: better learning, overfitting ↗
  - # units decreases: poorest learning, overfitting ↘

- A posteriori test with ≠ numbers of units/parameters!

- Aim: optimal generalization error

$$E_{gen}(\theta) = \lim_{T \to \infty} \sum_{t=1}^{T} \frac{(g(x_t, \theta) - y_t)^2}{T} \longrightarrow \boxed{\hat{E}_{gen}(\theta)}$$

Estimation of generalization error

$$E_{gen}(\theta) = \lim_{T \to \infty} \sum_{t=1}^{T} \frac{(g(x_t, \theta) - y_t)^2}{T} \longrightarrow \boxed{\hat{E}_{gen}(\theta)}$$

- Estimates of generalization error:
  - (k-fold) cross-validation
  - leave-one-out
  - bootstrap

- General principle: use different samples to
  - learn
  - validate (compare and select models)
  - test (assess the performances)

Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

of inputs in ANN

- Learning in high-dimensional spaces:

  number of samples $\approx e^{\text{space dimension}}$
- → necessity to *reduce* the input space dimension!
- How?
  - Selection of input variables
    - Test and errors (comparison of models)
  - Projection of input variables
    - Principal Component Analysis
    - Curvilinear Component Analysis
- How much?
  - A posteriori measures (idem model selection)
  - Specific methods (ex: time series)

Input variables selection and projection

- Starting with many input variables, then reduce their number
- Two options:
  1. selection of input variables
     - ↗ interpretability
     - ↘ limited to existing variables
  2. projection of input variables
     - linear: PCA
     - non-linear: CCA, Kohonen, etc.
       forecasting: based on **Taken's theorem**

Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

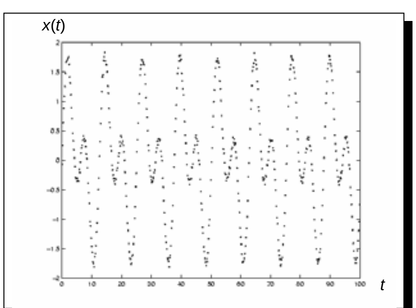
Application of dimension reduction to forecasting tasks

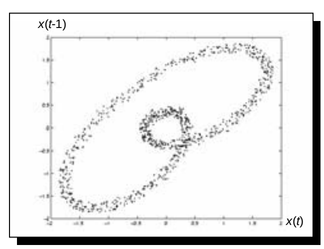
- Regressor: - past values $x(t\text{-}i)$
  - exogenous data $in(j)$
- Forecasting

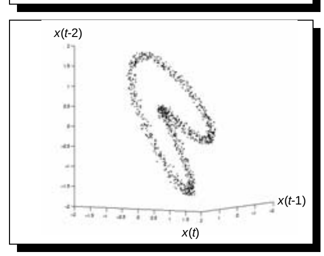
$$x(t+1) = f\left(x(t), x(t-1), \ldots, x(t-k), in(1), in(2), \ldots, in(l)\right)$$

- Non-linear forecasting:
  - 1. optimise regressor on linear predictor
    2. use the same regressor with non-linear predictor f
  - trials and errors (computational load !)
  - (non-linear) projection of regressor variables

Forecasting: Taken's theorem 1/2







time series

intrinsic dimension (q) = 1

- Takens' theorem:
  $q \leq$ size of regressor $\leq 2q+1$
  (AR model)

Forecasting: Taken's theorem 2/2

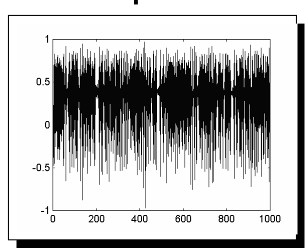
- Takens' theorem:
  $q \leq$ size of regressor $\leq 2q+1$
- In the $2q+1$ space, there exists a q-surface without intersection points
- Projection from $2q+1$ to q possible !

Forecasting: 1st example 1/2

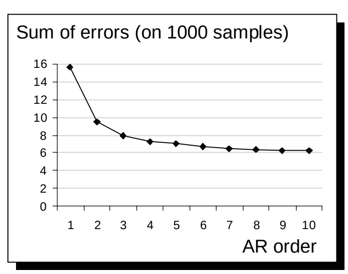
- Artificial series

  $$x(t+1) = ax(t)^2 + bx(t-2) + \varepsilon(t)$$

  Two past values!

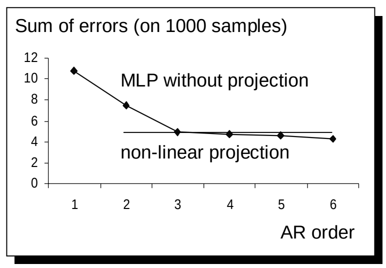


- Linear AR model



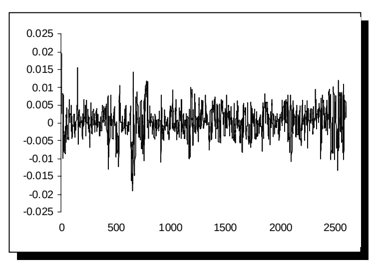
Forecasting: 1st example 2/2

- Non-linear AR model
  - initial regressor: size=6
  - intrinsic dimension: 2
  - CCA from dim=6 to dim=2
  - MLP on 2-dim data



Forecasting: 2nd example 1/2

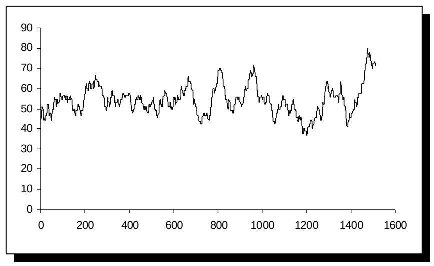
- Daily returns of BEL20 index



- 42 indicators from inputs and exogenous variables:
  - returns: x_t, x_{t-10}, x_{t-20}, x_{t-40}, …, y_t, y_{t-10}, …
  - differences of returns: x_t-x_{t-5}, x_{t-5}-x_{t-10}, …, y_t-y_{t-5}
  - oscillators: K(20), K(40), …
  - moving averages: MM(10), MM(50), …
  - exponential moving averages: MME(10), MME(50), …
  - etc

Forecasting: 2nd example 2/2

- Method:
  - 42 indicators
  - PCA → 25 variables
  - CCA → 9 variables ← Grassberger-Proccacia: intrinsic dimension = 9
  - RBF → forecasting
- Result: % of correct approximations of sign (90-days average)
- In average:
  57.2% on test set

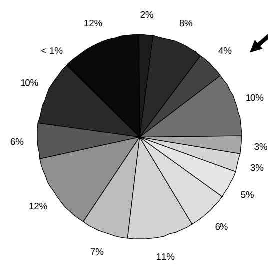


Content

- Artificial neural networks
  - Introduction
  - Multi-Layer Perceptron
  - Radial-Basis Function Networks
  - Self-Organizing Maps
- About the choice of parameters
  - # of units or parameters
  - # of inputs
- Two application examples in finance
  - Time series forecasting
  - Classification of investment funds

Why This Study ?

Announced Strategy ←



Why This Study ?

Announced Strategy ←

≠

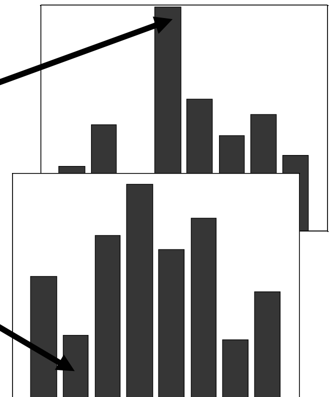
Implemented Strategy

Why This Study ?

Announced Strategy

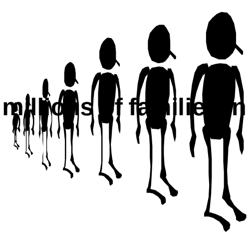
≠

Implemented Strategy



Goal of managers:
To be compared to other ones with weaker performances

Why This Study ?



44 millions of families in USA

Not able to estimate the real risk

Classifications exist
They are not very good

Kim T.-H., Stone D. and Tomas M., *Mutual fund objective misclassification.* Journal of Economics and Business, 2000. **52**: p. 309-323.

Extraction of Features
Linear Regression

$$R_i(t) = b_{1i}F_1(t) + b_{2i}F_2(t) + \cdots + b_{ni}F_n(t) + e_i(t)$$

$R_i(t)$: Return of a fund

$F_2(t)$: Index Return

b_{ij} determined using a Least-Square method

Extraction of Features
Linear Regression under constraints

$$R_i(t) = b_{1i}F_1(t) + b_{2i}F_2(t) + \cdots + b_{ni}F_n(t) + e_i(t)$$

$$\sum_j b_{ij} = 1$$

$$b_{ij} \geq 0, \forall i, j$$

b_{ij} percentages of investment

Extraction of Features
Multi-colinearity problem and PCA

$$R_i(t) = b_{1i} F_1(t) + b_{2i} F_2(t) + \cdots + b_{ni} F_n(t) + e_i(t)$$

Not equal, but highly correlated

b_{ji} are difficult to determine

Extraction of Features
Multi-colinearity problem and PCA

$$R_i(t) = b_{1i} F_1(t) + b_{2i} F_2(t) + \cdots + b_{ni} F_n(t) + e_i(t)$$

$$\boldsymbol{F_1, F_2, \ldots, , F_n} \xrightarrow{\textbf{PCA}} \boldsymbol{G_1, G_2, \ldots, , G_m}$$

$$R_i(t) = c_{1i} G_1(t) + c_{2i} G_2(t) + \cdots + c_{mi} G_m(t) + e_i(t)$$

$$\boldsymbol{G_1, G_2, \ldots, , G_m} \xrightarrow{\textbf{Inverse of PCA}} \boldsymbol{F_1, F_2, \ldots, , F_n}$$

$$R_i(t) = b'_{1i} F_1(t) + b'_{2i} F_2(t) + \cdots + b'_{ni} F_n(t) + e_i(t)$$

Classification of investment funds

- Extraction of features
  - Linear Regression (under constraints)
  - *Multi-colinearity* problem and PCA
- Classification
  - Kohonen Maps
  - Ward algorithm
- Application
  - CRSP database (the Chicago University)
  - Comparison with classification from the ICDI and S&P's Fund Services.

CRSP database (the Chicago University)

	indexes
1	**Dow Jones 30**
2	**Lehman Brothers' US Credit Bond Index**
8	**Salomon Brothers' Non-US Government Bond Index**
9	**S&P400 Medium Capitalization**
14	**FTSE100**
17	**FTSE Small Capitalization**
18	**UK Bank Bills 3 month**
21	**TOPIX100**
24	**Japan Benchmark 2 year Government Index**
26	**CAC40**
29	**France Benchmark 2 year Government**
32	**Germany Money Market 3 month**

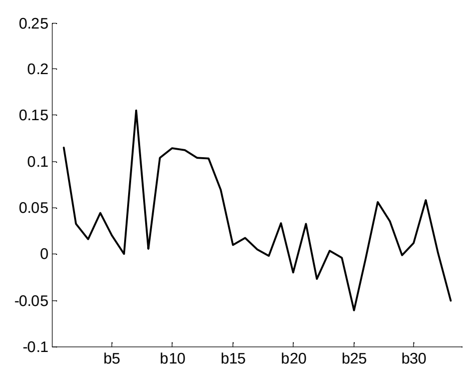
5822 funds

33 indexes

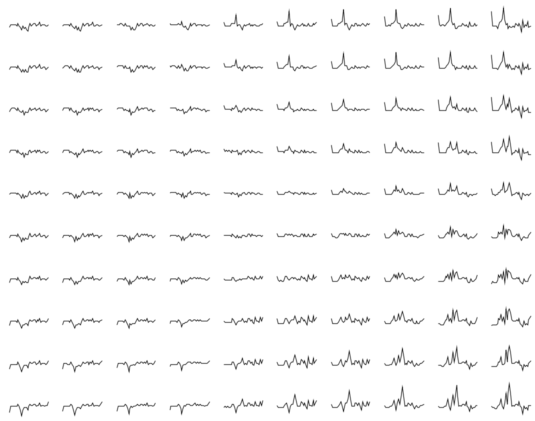
CRSP database (the Chicago University)

	indexes	b'_j
1	**Dow Jones 30**	0.1156
2	**Lehman Brothers' US Credit Bond Index**	0.0322
8	**Salomon Brothers' Non-US Government Bond Index**	0.0055
9	**S&P400 Medium Capitalization**	0.1042
14	**FTSE100**	0.6090
17	**FTSE Small Capitalization**	0.0044
18	**UK Bank Bills 3 month**	0.0336
21	**TOPIX100**	0.0326
24	**Japan Benchmark 2 year Government Index**	-0.0048
26	**CAC40**	-0.0057
29	**France Benchmark 2 year Government**	-0.0012
32	**Germany Money Market 3 month**	0.0016

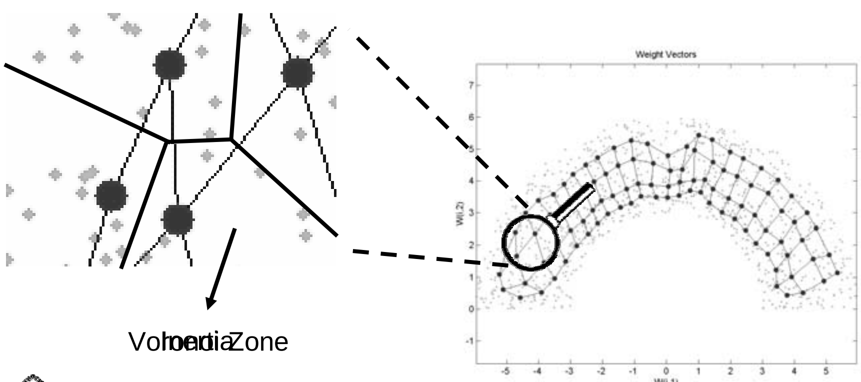
CRSP database (the Chicago University)



CRSP database
Kohonen



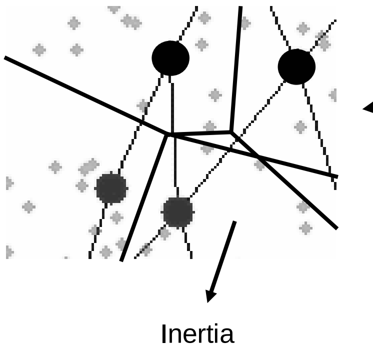
Classification
Ward Algorithm



Classification
Ward Algorithm

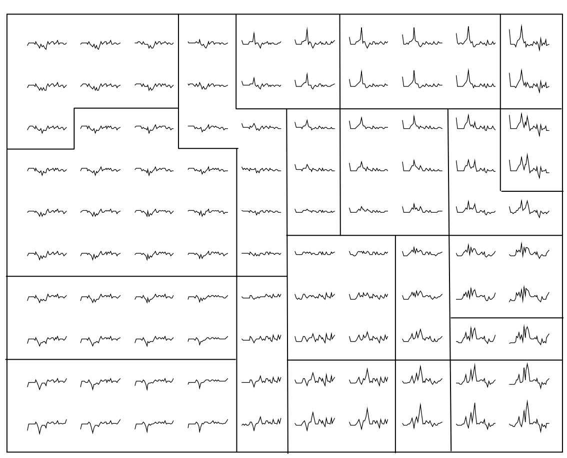
Intra-class inertia

$$I_w = \frac{1}{N} \sum_{l=1}^{P} N_l I_l$$



Merge the most similar centroids

CRSP database
Ward: 20 classes



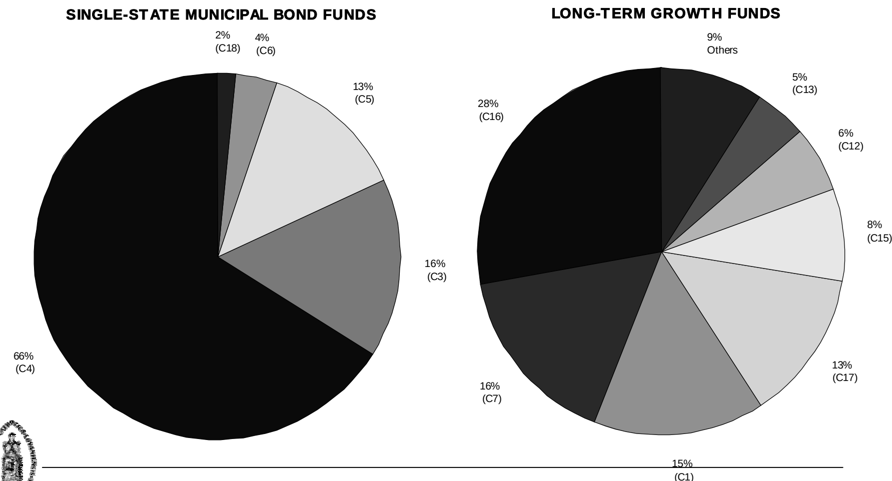
Comparison to classification from the ICDI and S&P’s Fund Services.

Strategies		
Aggressive Growth	Government Securities	Taxable Money Market
Balanced	International Equities	High Quality Municipal Bonds
High Quality Bonds	Income	Option Income
High Yield Bonds	Long-Term Growth	Precious Metals
Global Bonds	Tax-Free Money Market	Sector Funds
Global Equity	Gov Securities Money Market	Special Funds
Growth & Income	High Quality Municipal Bonds	Total Return
Ginnie Mae Funds	Single-State Municipal Bonds	Utility Funds

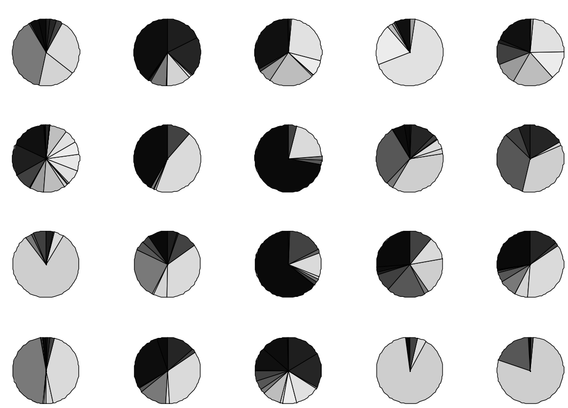
Comparison to classification from the ICDI and S&P’s Fund Services.

Strategies		
Aggressive Growth	Government Securities	Taxable Money Market
Balanced	International Equities	High Quality Municipal Bonds
High Quality Bonds	Income	Option Income
High Yield Bonds	Long-Term Growth	Precious Metals
Global Bonds	Tax-Free Money Market	Sector Funds
Global Equity	Gov Securities Money Market	Special Funds
Growth & Income	High Quality Municipal Bonds	Total Return
Ginnie Mae Funds	Single-State Municipal Bonds	Utility Funds

Comparison to classification from the ICDI and S&P's Fund Services.



Comparison to classification from the ICDI and S&P's Fund Services.



Analysis

Intra-class inertia for classifications:

- 0.07 for the Kohonen/Ward classification
- 0.13 for the reference classification

Why some differences between the two classifications ?

- classification from ICDI based on information given by the managers ?
- reference classification not so sophisticated ?
- strategy not constant ?

Conclusion

- ANN are regression/classification tools:
  - powerful in theory
  - powerful in practice
  - but application needs caution and expertise:
    - choosing # of parameters
    - choosing # and which inputs
    - learning procedure (local minima, etc.)
    - validation !
    - comparison with more classical tools
    - etc.