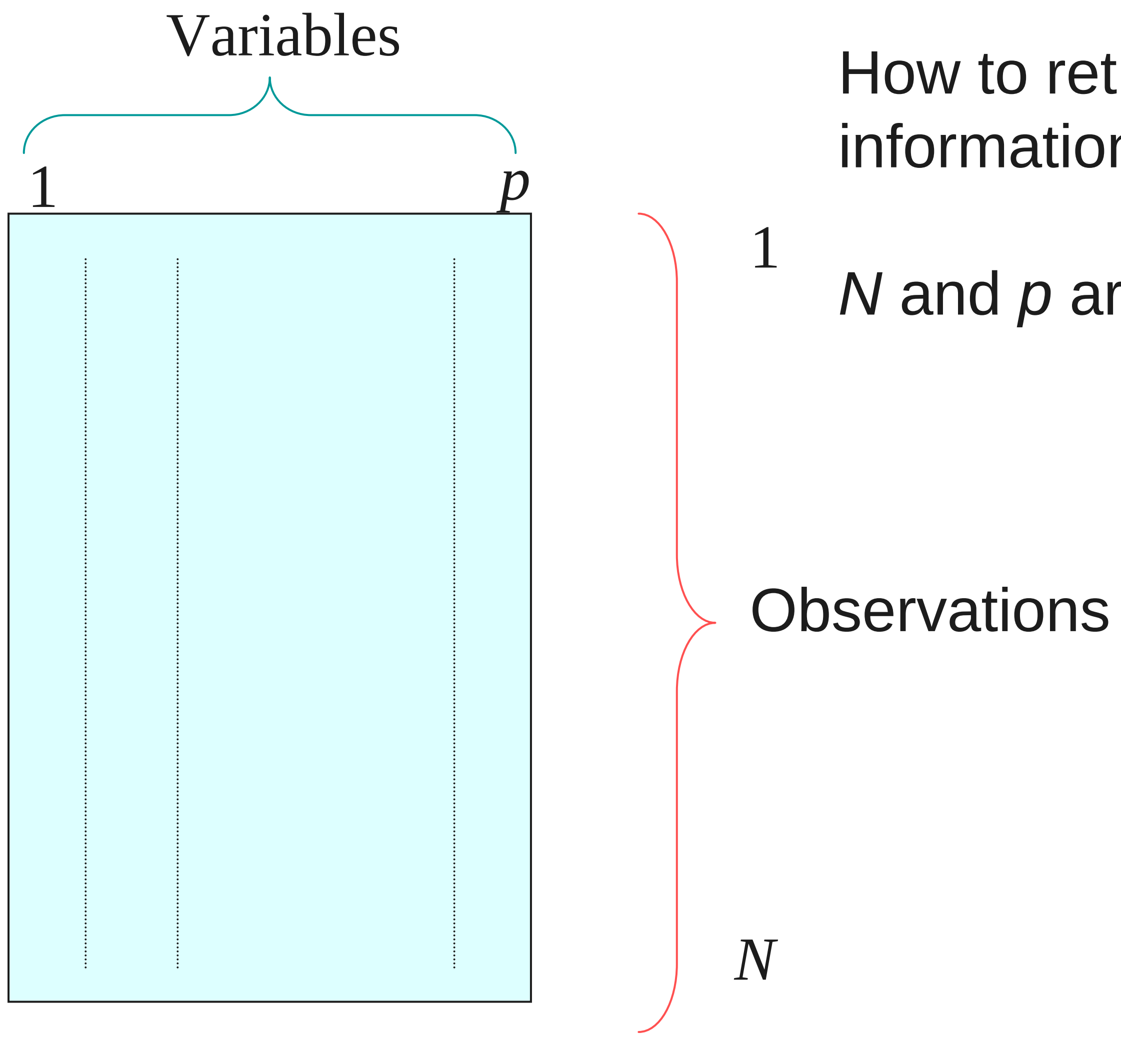
Data Analysis using Self-Organizing Maps

Marie Cottrell and Patrick Letrémy

CES UMR 8174

Data Analysis, Data Mining



How to retrieve information?

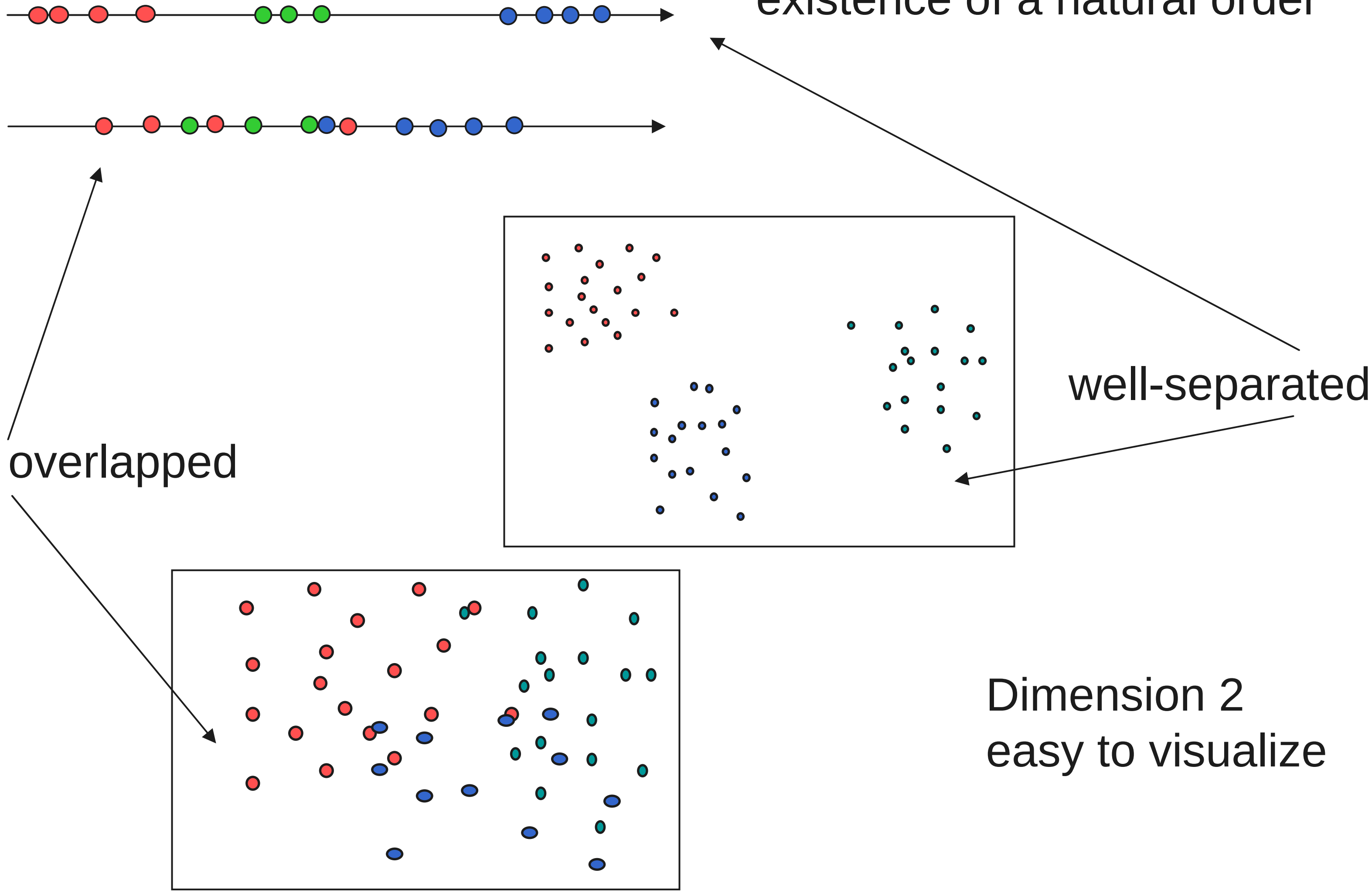
N and *p* are large

Several goals to achieve

- Extraction of code-vectors (prototypes, representative vectors): **quantization**
- Definition and description of classes: **clustering and typology**
- **Representation and visualization** of multidimensional data
- **Forecasting** vectors or trajectories
- Very frequent case: how to manage with **incomplete data**
- A posteriori allocation of **supplementary observations**

Examples: dimension 1 and 2

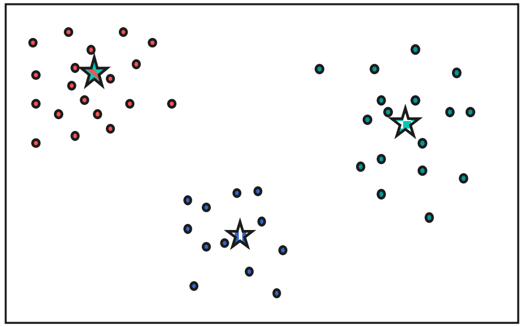
Dimension 1
existence of a natural order



Dimension 2
easy to visualize

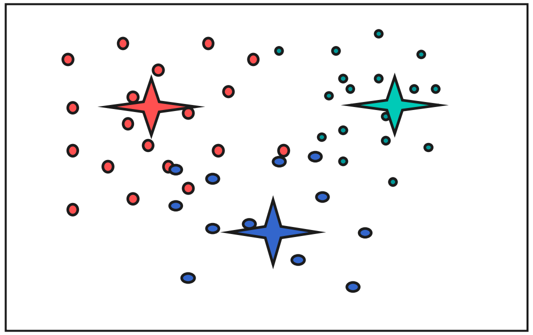
Quantization ⟺ Clustering

- They are reciprocal problems



- If the code-vectors are known
  - **the classes** are built by the
  - nearest-neighbors method

- If the **classes** are known
  - the code-vectors are computed as the mean values



- For dimension > 2, the visualization need specific methods (projections, PCA, …)

The code-vectors

- Each code-vector (which is the prototype of one class) is a vector in R^p
- We denote by *n* the numbers of code-vectors (= the number of clases)

- These vectors are also called
  - ***weight vectors*** (biological vocabulary)
  - ***prototypes*** (vectorial quantization vocabulary)
  - ***centroïdes*** (statistical vocabulary)

At time t, they are denoted by

$$C^{(n)}(t) = \left(C_1^{(n)}(t), C_2^{(n)}(t), \cdots, C_n^{(n)}(t)\right)$$

Several similar methods for clustering and computing the code-vectors

- **I: Moving centers algorithm (Forgy algorithm), deterministic (MC)**
- **II: Kohonen algorithm, batch version, deterministic (KBATCH)**
  It takes into account the neighborhood between classes
- **III: Simple Competitive Learning (SCL), stochastic version of the moving centers algorithm, k-means algorithm**
- **IV: Stochastic Kohonen algorithm (SOM)**
  It takes into account the neighborhood between classes

Algorithms III et IV are adaptive algorithms:
Learning on-line algorithms

- **Goal: - to build well separated and homogeneous classes**
- **- to compute accurate and representative code-vectors**

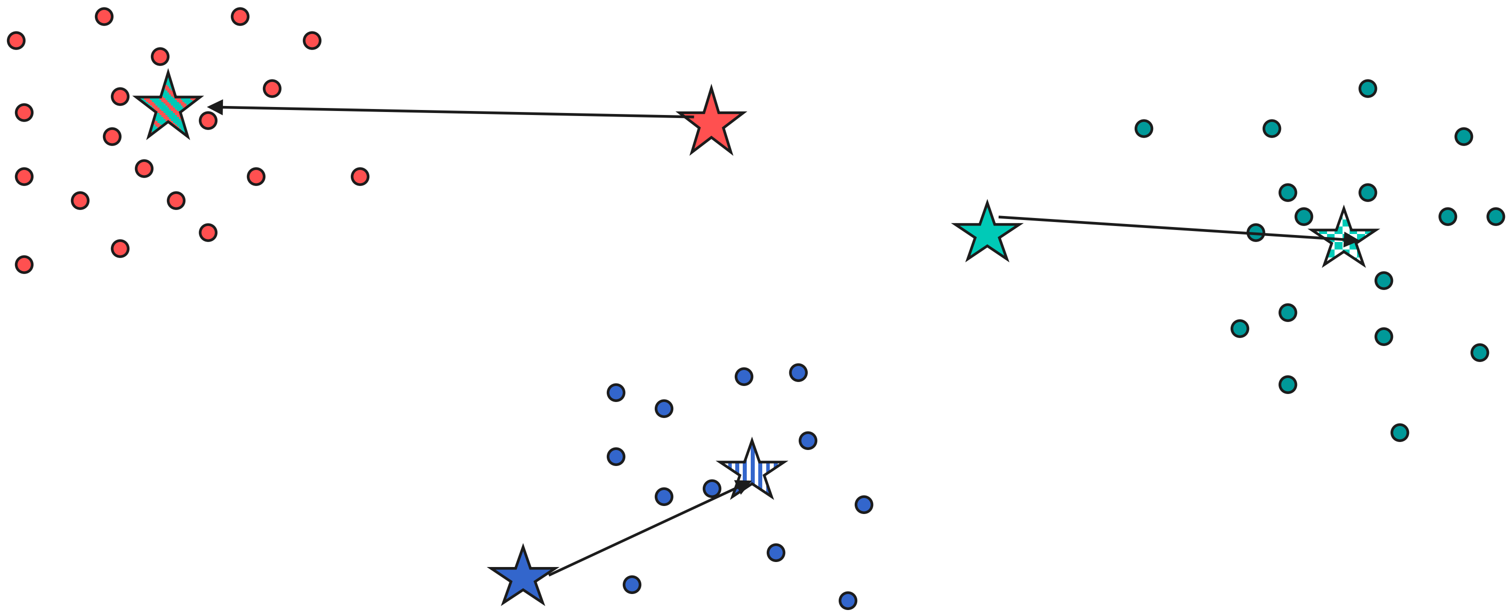
Classification Algorithms

	No neighborhood	*With neighborhood*
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

I – Deterministic algorithm Moving centers (FORGY)

	No neighborhood	With neighborhood
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

- At each step, classes are defined by the nearest-neighbor method, the code-vectors are computed as the mean points of the classes, and repeat…

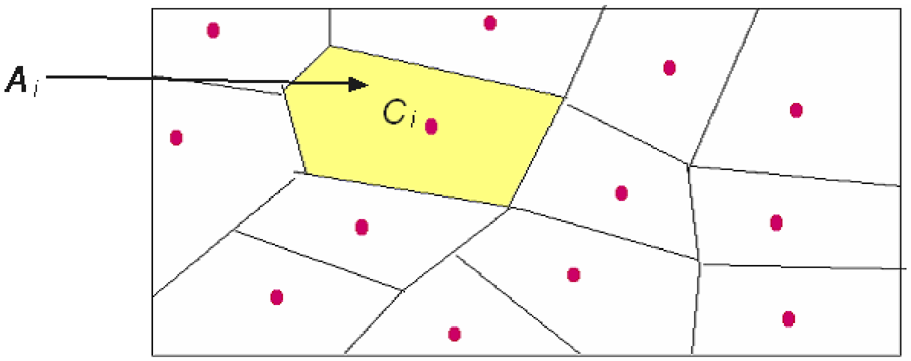


- Starting from randomly chosen code-vectors, we define the classes, then the code-vectors, and so on.

Property : The distortion is decreasing

- **Limit code-vectors depend on the initial choice**
- **The distortion (the error we get by replacing the data by the code-vectors) decreases, the algorithm converges to a local minimum**

$$V_n(C_1^{(n)}, C_2^{(n)}, \dots, C_n^{(n)}) = \frac{1}{N} \sum_{i=1}^{n} \sum_{x \in A_i(C^{(n)})} \left\| C_i^{(n)} - x \right\|$$

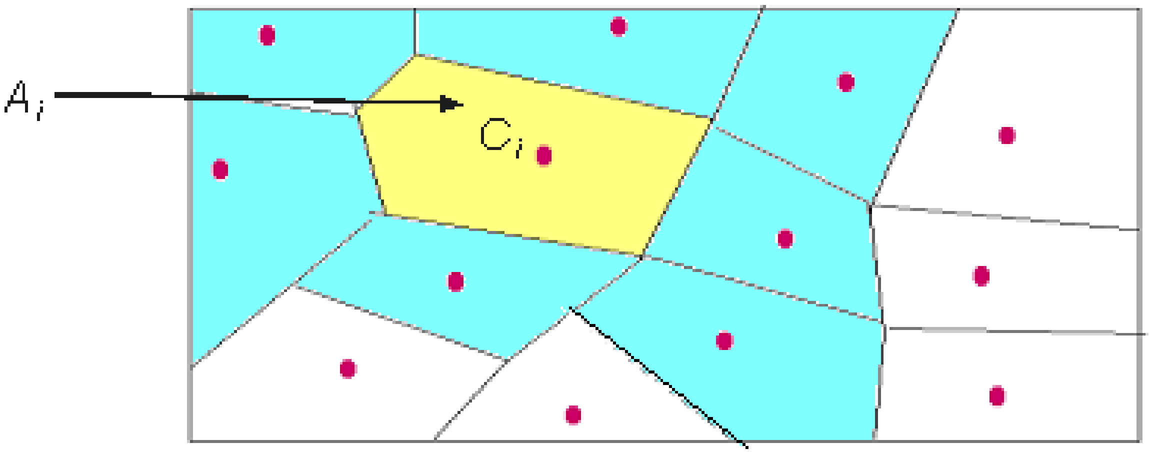


C_i is the code-vector of class A_i

II Batch Kohonen algorithm (KBATCH)

	No neighborhood	*With neighborhood*
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

- It generalizes the moving centers algorithm
- At each step, the classes are defined as before, and the code-vectors are computed as the gravity centers of a subset which includes the class **and the neighbor classes**,

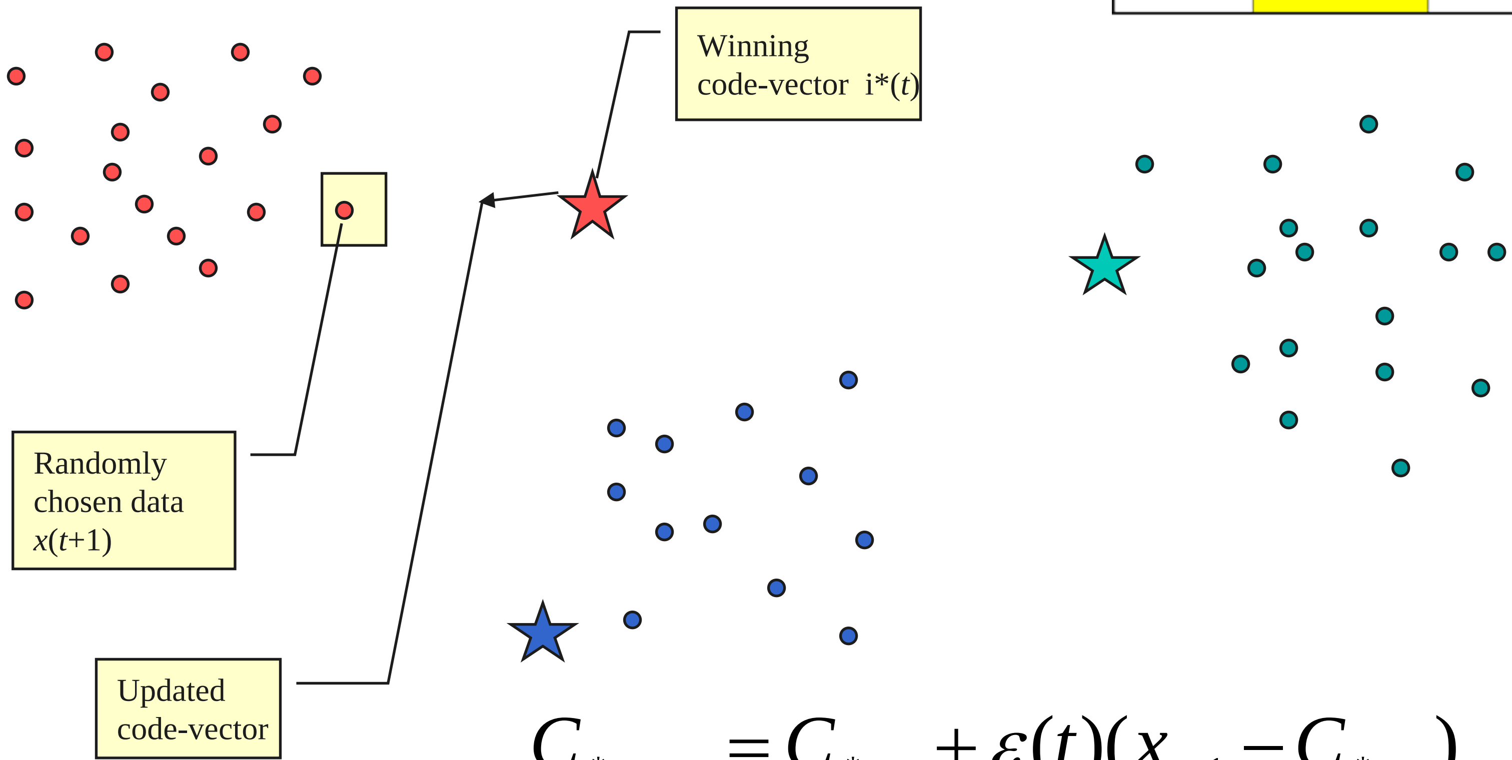


The neighborhood structure is a priori defined, the initialization is randomly chosen. The algorithm produces “organization”: classes which are neighbor in the structure, become neighbor in the data space.

III Stochastic Algorithm (SCL)

	No neighborhood	*With neighborhood*
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

Randomly chosen initial code-vectors,
on-line algorithm



$$C_{i^*(t+1)} = C_{i^*(t)} + \varepsilon(t)(x_{t+1} - C_{i^*(t)})$$

At each step, an observation is randomly picked.

Only one code-vector is updated: it is the nearest, i.e. the winning code-vector. **Competitive Learning. The mean distortion decreases.**

IV Kohonen Algorithm
Self-Organizing Map (SOM)

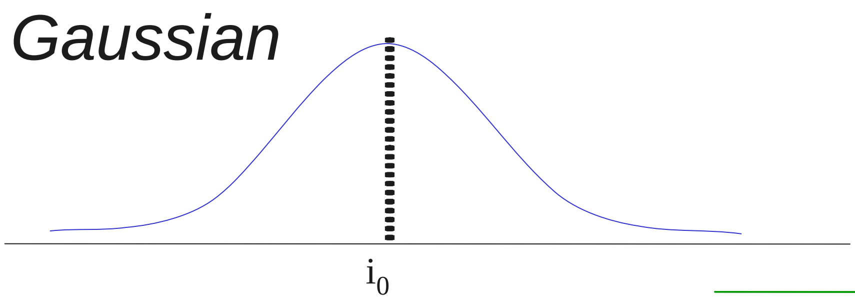
	No neighborhood	*With neighborhood*
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

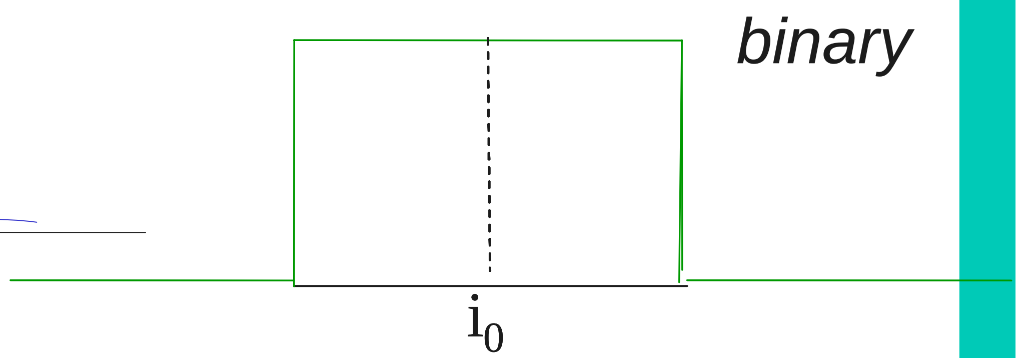
- **Generalization of the SCL algorithm**
  - **Stochastic on-line algorithm**
  - **Takes into account the neighborhood of each class**
- **Stochastic version of the BATCH algorithm**
  - **Avoids some local minima**
  - **Provides “organized” classes (see examples) and accurate code-vectors**

The neighborhood structure

- A ***neighborhood structure*** is defined over the classes.

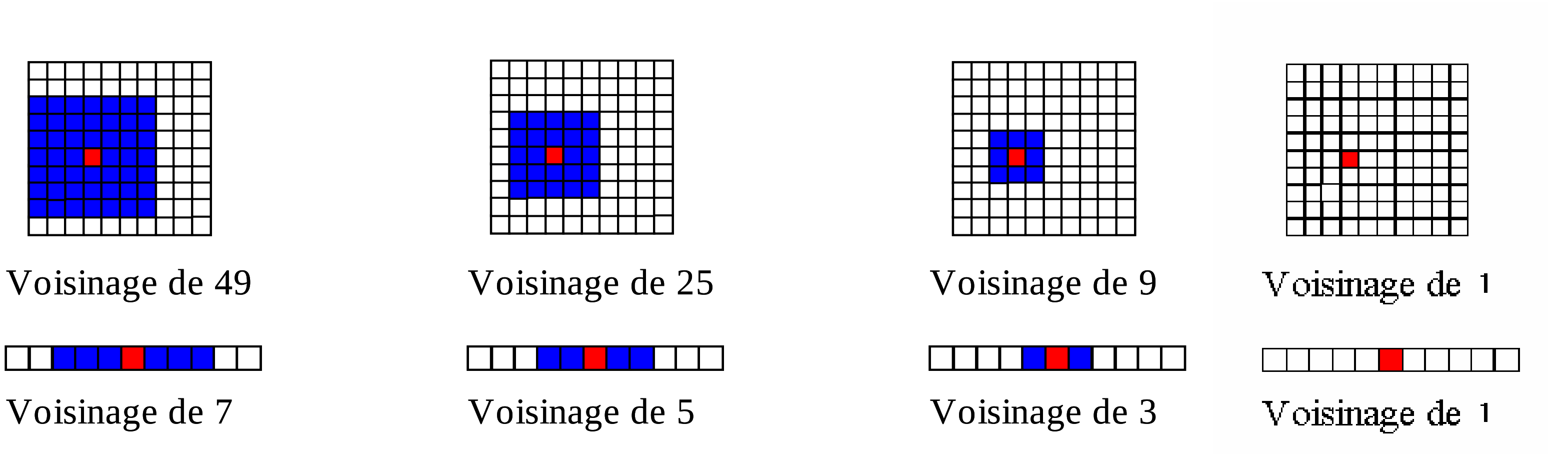
- If $I=\{1, 2, \ldots, n\}$, is the set of the indices, the neighborhood structure is defined by a ***function*** $\sigma(i,j)$ which
  - is = 1 when $i = j$,
  - is symmetrical,
  - only depends on the distance between the indices
  - is a decreasing function of this distance

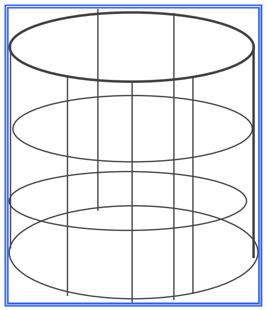


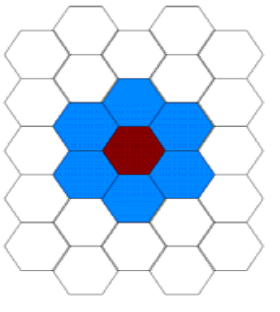


The shape and the size of the neighborhoods

- They are entirely defined by the neighborhood function, but for visualization purpose, they correspond to a simple geometric disposition: string, grid, cylinder, hexagonal network, etc.







The Kohonen algorithm

- For a given state of the code-vectors $C^{(n)}$ and for an input x, we denote by $i_0(C^{(n)}, x)$ the index of the ***winning code-vector***, that is the index of which the codevector is the nearest of x.

- Then the algorithm is defined as follows:

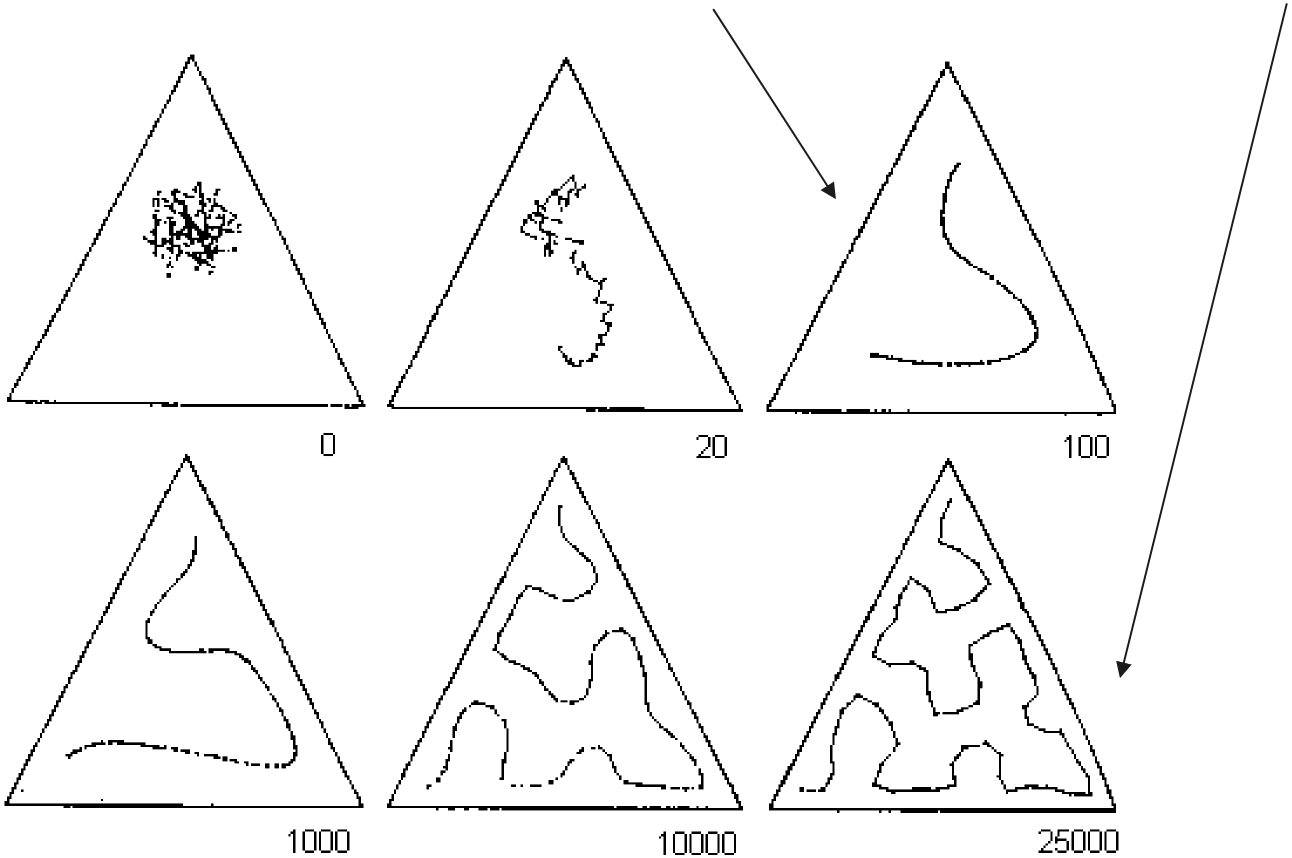
$$\begin{cases} i_0(C^{(n)}(t), x(t+1)) = Arg\min_i \left\| C_i^{(n)}(t) - x(t+1) \right\| \\ C_i^{(n)}(t+1) = C_i^{(n)}(t) + \varepsilon_t \sigma(i_0, i)\ \ (x(t+1) - C_i^{(n)}(t)), \forall i \in I \end{cases}$$

The winning code-vector and its neighbors are moved towards the input

If ε_t and σ do not depend on t, it is a Markov Chain

Example : 2 dimensional data

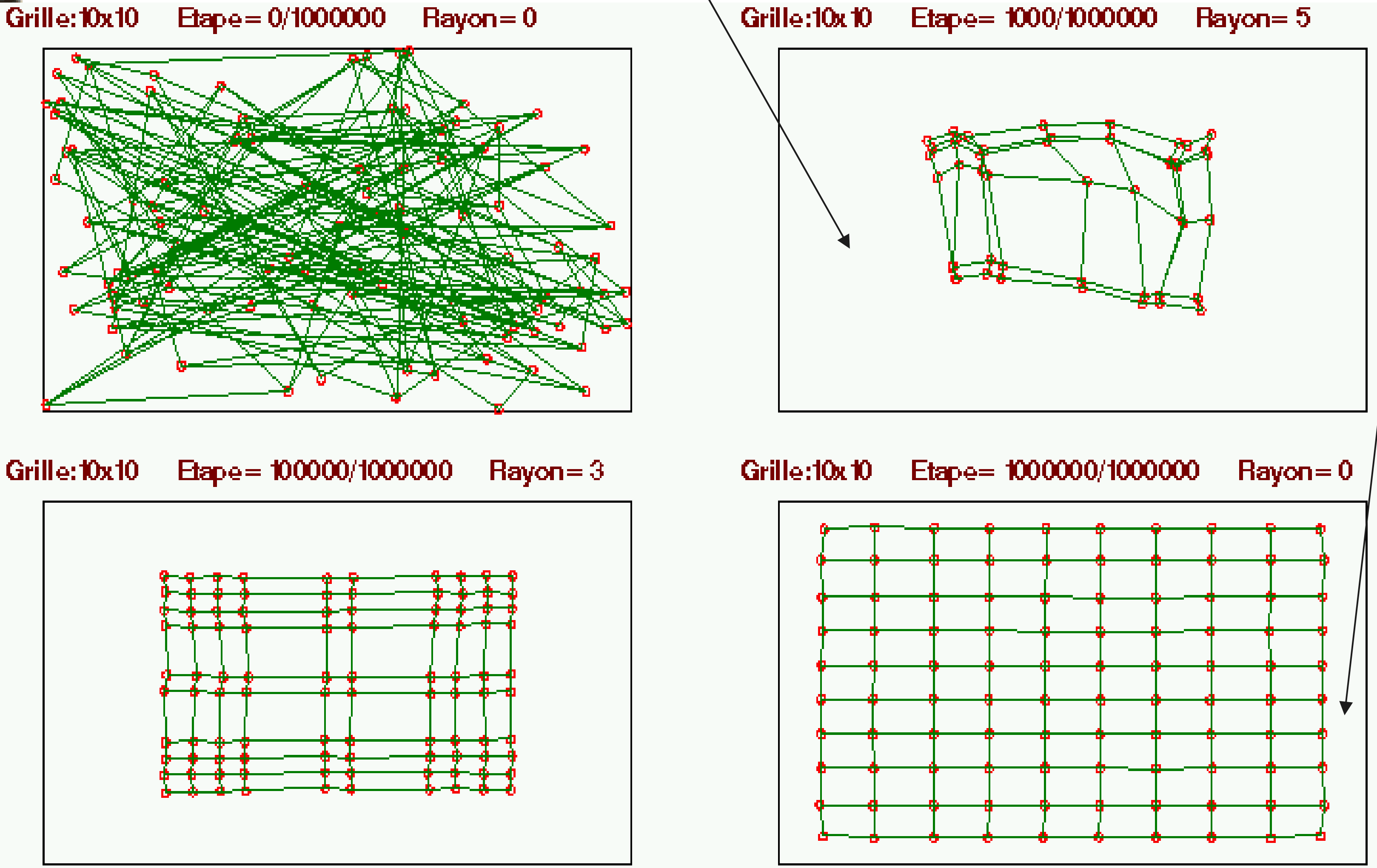
Learning algorithm: organization, and then quantization



Neighborhoods on a string (Kohonen, 1995)

2 dimensional data

Learning algorithm: organization, and then quantization



Neighborhoods on a grid

The algorithm (some mathematical hints)

The algorithm can be written

$$C^{(n)}(t+1) = C^{(n)}(t) - \varepsilon_t H(x(t+1), C^{(n)}(t))$$

It is a stochastic algorithm, and we would like to use the Robbins-Monro method, with the hypothesis

$$\sum \varepsilon_t = \infty \;\; and \;\; \sum \varepsilon_t^2 < \infty$$

But, this hypothesis does not allow us to solve the problem:
This algorithm does not derive from a potential function, as long as data are distributed according to a continuous density function

H* *is not a gradient in the general case

Points of study

- ***Convergence of the algorithm***, when t tends towards infinity, when ε is constant or decreasing
- Nature of ***stable limit states, uniqueness***
- ***Organization*** of the final state,
  - in dimension 1, organization means “order”,
  - but in larger dimension?

- ***Some references:***
  - *Kohonen* (82, 84) sketch of proof of convergence
  - *Cottrell-Fort* (87) dimension 1, uniform distribution, 2 neighbors on a string
  - *Ritter et al.* (86, 88) study the stationary state for any dimension, (assuming that it exists)
  - For dimension 1, extension to more general distributions by *Bouton-Pagès* (93,94), but without uniqueness of the stationary state, even after ordering
  - For dimension 1, extension of the proof of organization with a more general neighborhood function *Erwin et al.* (92)
  - Proof of the non existence of a potential function by *Erwin et al.* (92)
  - Almost sure CV. towards an unique stationary state for dimension 1, after reordering, for general distributions and neighborhoods (*Fort-Pagès*, 93,95, 97)
  - CV in the multi-dimensional frame (*Sadeghi*, 2001)

Convergence ?

- One complete proof is available only in the one dimensional case (for the data and for the network)

  (Cottrell, Fort, 1987, followed by other authors)

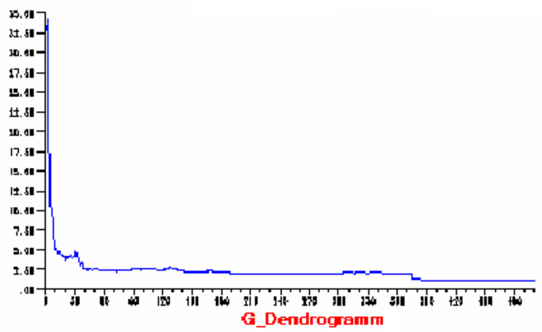
- Some frames provide ***hints*** to try to rigorously prove the results which can be observed in numerical experiments
- The tools : ***Markov Chains, Ordinary Differential Equation (ODE), Batch algorithm***

- ***(and some modified algorithms, like Neural Gas, Heske algorithm, etc. have better properties)***
- There is a particular case: when the data are available in a finite database

Extended Distortion: « Cost function » (finite database)

When the size of the database is finite, equal to N, one can prove that the algorithm derives from a potential (Ritter, Martinetz and Shulten result)

$$V_n(C^{(n)}) = \frac{1}{2N} \sum_{i=1}^{n} \sum_{x \in A_i(C^{(n)})} \left(\sum_{j=1}^{n} \sigma(i-j) \left\| C_i^{(n)} - x \right\|^2 \right)$$



This « potential » is not differentiable, and this property does not provide a true proof for convergence

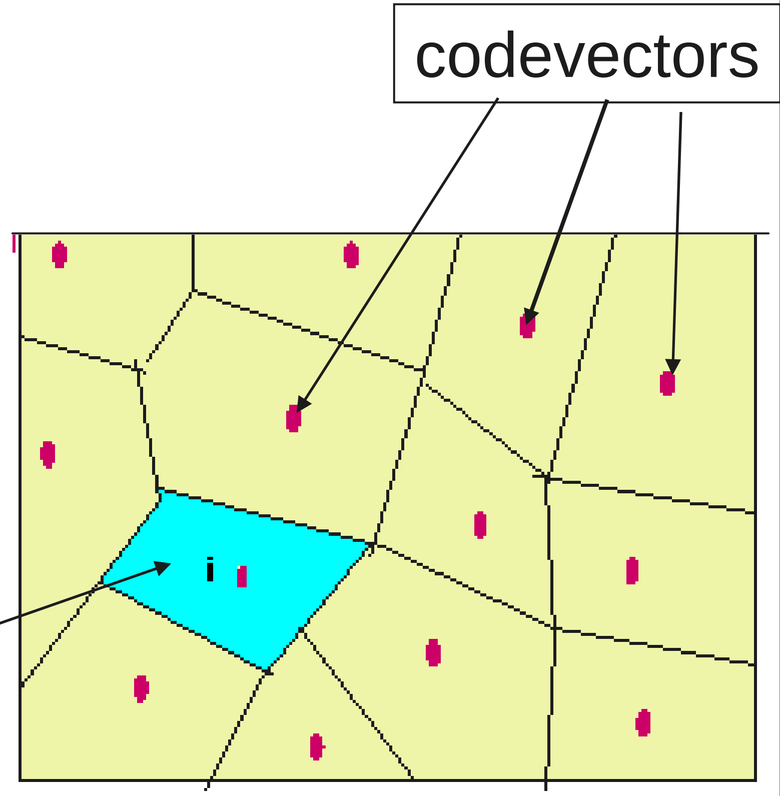
However, it provides an intuitive interpretation of the behavior of the algorithm

Classes, Voronoï tessels

- In any case, even if the convergence cannot be fully proved, the Kohonen algorithm computes the code-vectors and then the classes are defined as the associated Voronoï tessels

A Voronoï tessel (class)

is defined as the set of the data which are nearest to $C_i^{(n)}$ than to all other code-vectors



$A_i(C^{(n)})$

Comparison between the four algorithms

	No neighborhood	*With neighborhood*
Deterministic	Moving Centers (Forgy, MC)	Batch Kohonen algorithm (KBATCH)
Stochastic	Simple Competitive Learning (SCL), k-means	Kohonen algorithm, Self-Organizing Maps (SOM)

- The four methods provide well-done classes, well-separated, homogeneous
- Both stochastic algorithms are on-line algorithms, easy to develop, avoiding most part of local minima, and weakly depending on the initial choices of the code-vectors
- Both Kohonen algorithms have nice properties of visualization, due to the organization property
- **SOM algorithm has all the desired properties**

How to use the SOM for data analysis

- The SOM algorithm groups the observations into classes
- Each class is represented by its code-vector
- Inside a class, the elements are similar and resemble the elements of neighbor classes
- The Kohonen classes can be grouped into larger super-classes which are easier to describe. These super-classes group only contiguous classes, due to the organization
- This property provides a nice visualization along the Kohonen maps

- In each unit of the map, one can represent the code-vector, the contents, by list or by graph.

Examples from Ibbou, Rousset, Gaubert, Mangeas, Debodt, Grégoire…

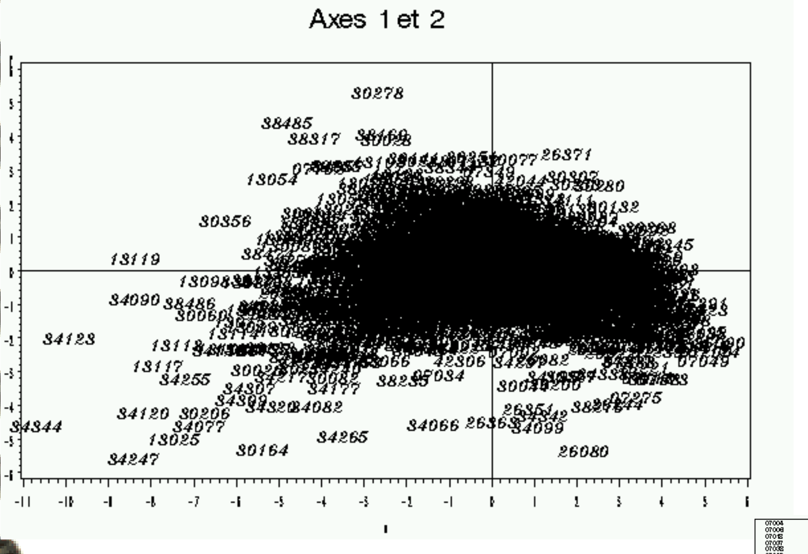
Example 1: 7-dimensional data

- Census of 1783 districts in 1936, 1954, 1962, 1968, 1975, 1982, 1990.
- French Rhône Valley districts, from mountain to seaside
- Ardèche, Bouches-du-Rhône, Drôme, Gard, Hérault, Isère, Haute-Loire, Vaucluse
- The values are divided by the sum of the 7 census for normalization reasons
- Chi-square distance
- Build a Kohonen map:
  - 10 000 iterations, 8 by 8 squared grid, and the 64 classes are grouped into 5 super-classes (82% of the total inertia)

Ex 1: The data (extract))

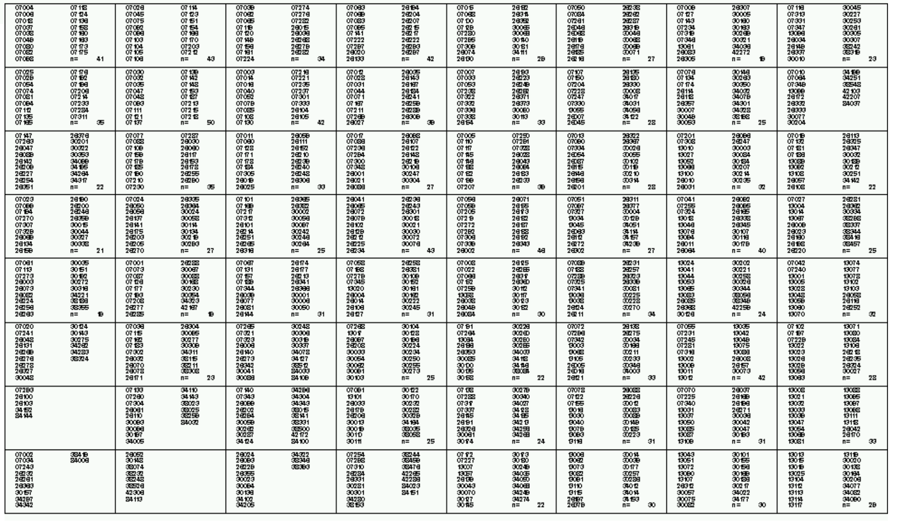
CODE	NOM	POUR36	POUR54	POUR62	POUR68	POUR75	POUR82	POUR90
07001	ACCONS	0.178	0.147	0.137	0.113	0.118	0.154	0.153
07002	AILHON	0.196	0.146	0.117	0.103	0.098	0.127	0.213
07003	AIZAC	0.210	0.170	0.150	0.135	0.105	0.110	0.120
07004	AJOUX	0.304	0.173	0.143	0.123	0.097	0.085	0.076
07005	ALBA-LA-ROMAINE	0.158	0.122	0.131	0.145	0.144	0.137	0.164
07006	ALBON	0.257	0.190	0.157	0.127	0.107	0.083	0.079
07007	ALBOUSSIERE	0.164	0.150	0.144	0.139	0.138	0.120	0.144
07008	ALISSAS	0.125	0.121	0.147	0.133	0.125	0.161	0.188
07009	ANDANCE	0.137	0.145	0.135	0.139	0.155	0.138	0.151
07010	ANNONAY	0.121	0.125	0.142	0.160	0.160	0.150	0.143
07011	ANTRAIGUES	0.202	0.149	0.134	0.133	0.125	0.131	0.127
07012	ARCENS	0.187	0.146	0.152	0.145	0.128	0.122	0.120
07013	ARDOIX	0.151	0.139	0.122	0.130	0.141	0.152	0.166
07014	ARLEBOSC	0.199	0.159	0.149	0.143	0.129	0.113	0.108
07015	ARRAS-SUR-RHONE	0.147	0.158	0.155	0.155	0.128	0.122	0.135
07016	ASPERJOC	0.197	0.162	0.145	0.130	0.130	0.117	0.119
07017	ASSIONS	0.175	0.136	0.152	0.140	0.123	0.135	0.138
07018	ASTET	0.277	0.215	0.152	0.127	0.100	0.067	0.062
07019	AUBENAS	0.112	0.121	0.129	0.151	0.169	0.162	0.156
07020	AUBIGNAS	0.209	0.114	0.140	0.113	0.125	0.133	0.167
07022	BAIX	0.135	0.128	0.135	0.128	0.115	0.202	0.157
07023	BALAZUC	0.215	0.154	0.127	0.112	0.109	0.141	0.142
07024	BANNE	0.194	0.157	0.134	0.118	0.122	0.134	0.140
07025	BARNAS	0.262	0.175	0.153	0.133	0.095	0.089	0.093
07026	BEAGE	0.234	0.183	0.149	0.142	0.118	0.097	0.078
07027	BEAUCHASTEL	0.100	0.103	0.120	0.127	0.184	0.192	0.174
07028	BEAULIEU	0.175	0.153	0.147	0.136	0.136	0.132	0.122
07029	BEAUMONT	0.308	0.166	0.127	0.112	0.094	0.100	0.093
07030	BEAUVENE	0.227	0.167	0.166	0.142	0.105	0.099	0.094
07031	BERRIAS-ET-CASTELJAU	0.183	0.148	0.147	0.140	0.139	0.121	0.123
07032	BERZEME	0.221	0.184	0.141	0.147	0.105	0.111	0.091
07033	BESSAS	0.180	0.134	0.133	0.133	0.134	0.154	0.134
07034	BIDON	0.203	0.117	0.079	0.092	0.102	0.187	0.219
07035	BOFFRES	0.224	0.175	0.146	0.135	0.105	0.110	0.105
07036	BOGY	0.164	0.137	0.133	0.119	0.128	0.144	0.175
07037	BOREE	0.269	0.200	0.157	0.135	0.100	0.074	0.064
07038	BORNE	0.311	0.188	0.167	0.124	0.073	0.058	0.079
07039	BOZAS	0.208	0.177	0.164	0.144	0.114	0.106	0.088
07040	BOUCIEU-LE-ROI	0.212	0.163	0.145	0.147	0.116	0.104	0.113
07041	BOULIEU-LES-ANNONAY	0.115	0.116	0.126	0.137	0.157	0.166	0.183
07042	BOURG-SAINT-ANDEOL	0.091	0.090	0.107	0.173	0.168	0.181	0.190
07044	BROSSAINC	0.173	0.160	0.157	0.144	0.124	0.117	0.124
07045	BURZET	0.243	0.202	0.154	0.129	0.095	0.095	0.082
07047	CELLIER-DU-LUC	0.234	0.166	0.149	0.128	0.113	0.098	0.111
07048	CHALENCON	0.223	0.174	0.164	0.147	0.097	0.099	0.095
07049	CHAMBON	0.331	0.205	0.154	0.110	0.071	0.069	0.061
07050	CHAMBONAS	0.159	0.145	0.148	0.148	0.138	0.127	0.135
07051	CHAMPAGNE	0.131	0.129	0.129	0.132	0.143	0.155	0.182
07052	CHAMPIS	0.197	0.169	0.159	0.141	0.121	0.096	0.118
07053	CHANDOLAS	0.181	0.141	0.141	0.142	0.136	0.132	0.127

Ex 1: The districts, PCA versus SOM



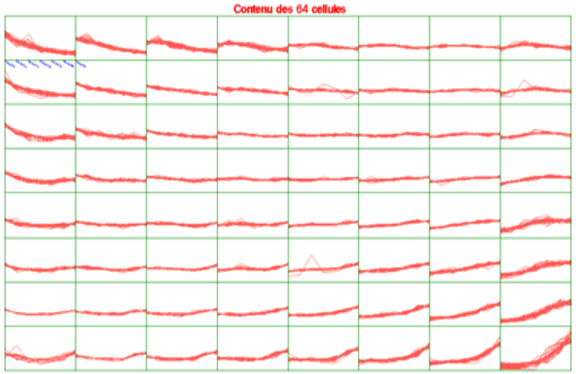
No evidence for classes (PCA)

64 organized classes (SOM)

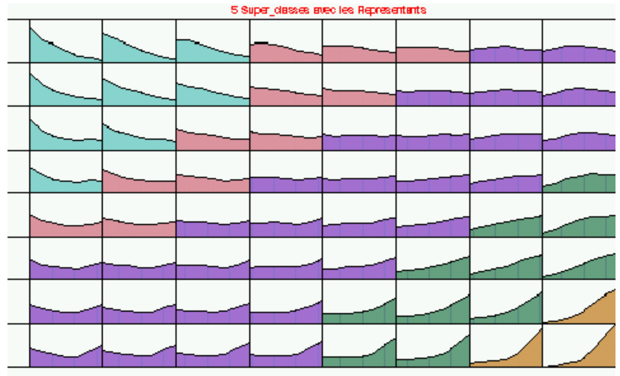


Example 1: ***7*-**dimensional data (7 census of the French Rhône Valley districts from 1936 to 1990), 1783 districts

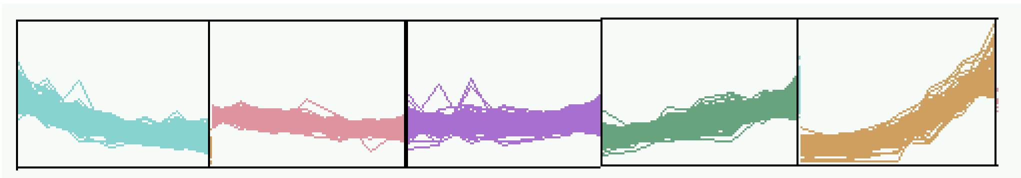
The code-vectors have 7 components



Contents of all the 64 classes in a Kohonen 2-dim map

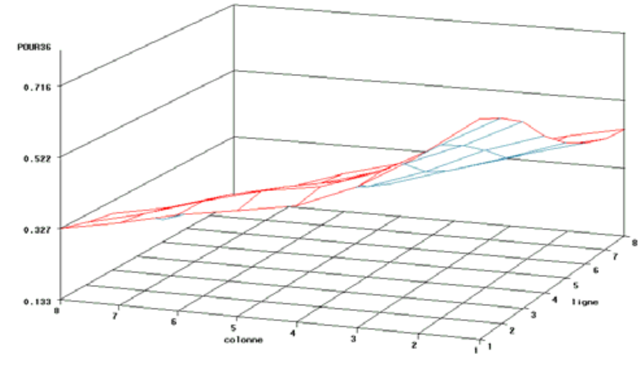


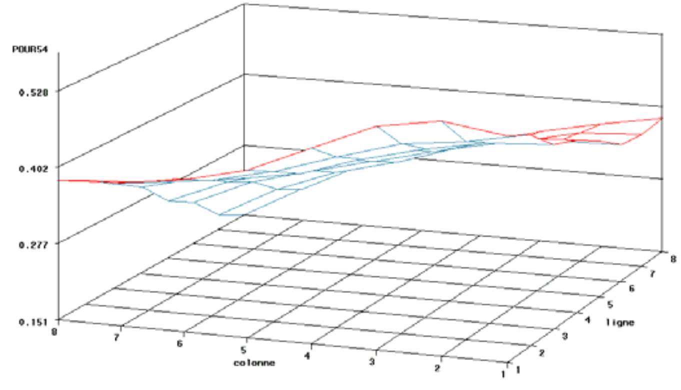
The code-vectors are displayed over the 2-dim map, and are « well » organized. The 64 classes are grouped into 5 super-classes

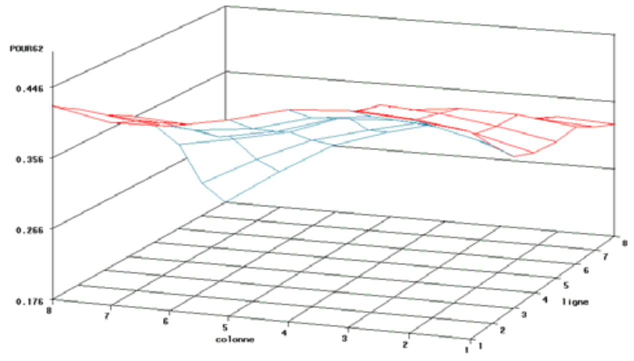


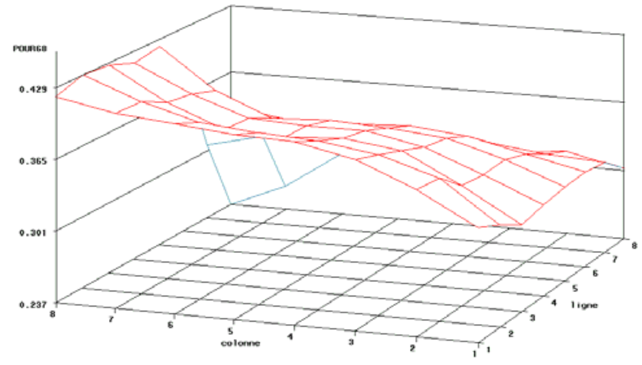
(perfect organization) Contents of the 5 super-classes

The first four census

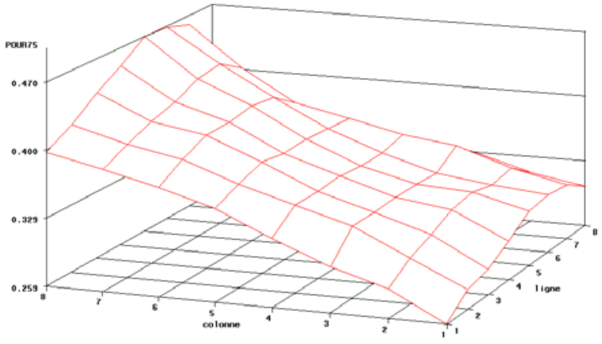


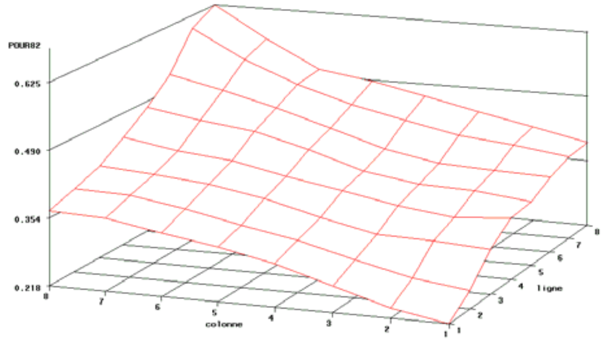


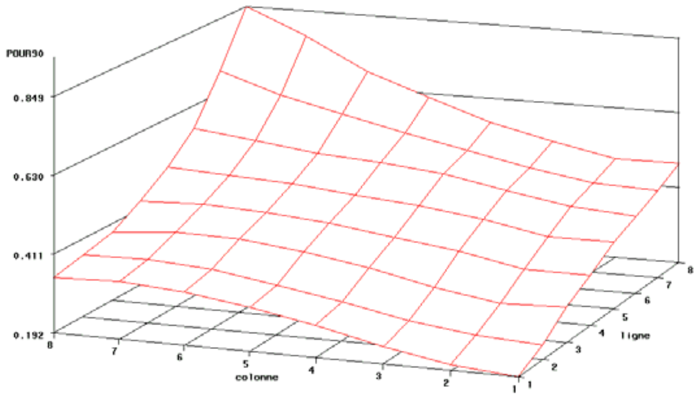




The last three census

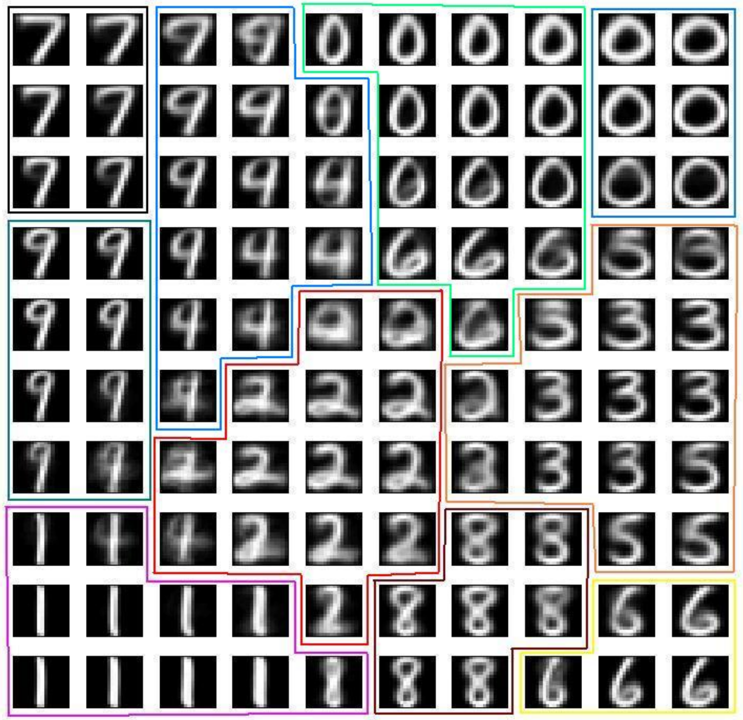


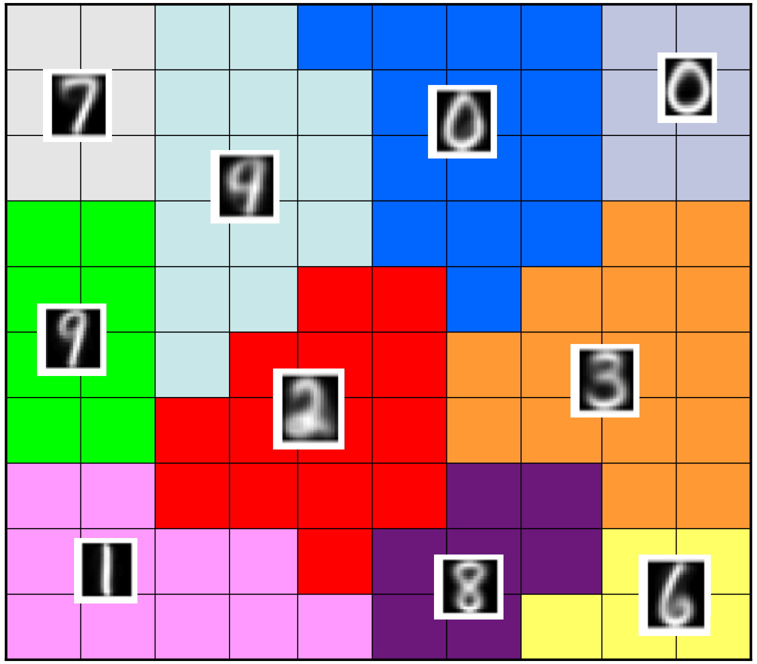




Ex 2: Handwritten Characters

The characters are transformed into 256-dim vectors. These vectors are classified using a Kohonen algorithm and displayed on a 10 by 10 map. They are well-organized. All the vectors are displayed here.



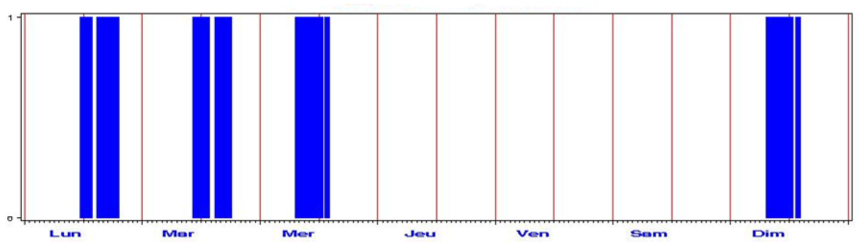


We can define 10 super-classes. Their mean values are computed, and we observe that digit 5 is missing, because the digits 5 belong to the same class than digit 3.

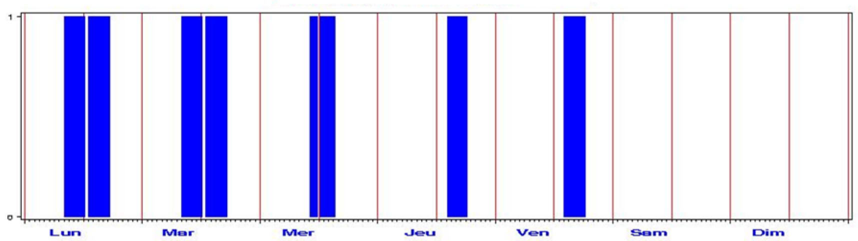
Ex 3: Working patterns for part-time employment

- 566 part-time workers
- 473 open-ended employment contracts, 93 fixed-term contracts
- 505 women, 61 men
- During a week, at each quarter-hour, a binary code is filled by each individual: 1 if he works, 0 if not
- So the dimension of each observation is 4×24×7=672
- On another hand, each individual is known by a lot of individual variables (sex, age, nature of contract, activity level, etc.)

Example: he work on Monday, Tuesday, Wednesday and Sunday a.m., with a small break



He works on Monday, Tuesday, Wednesday , Thursday and Friday



10 classes and 5 super-classes (from Monday to Sunday)

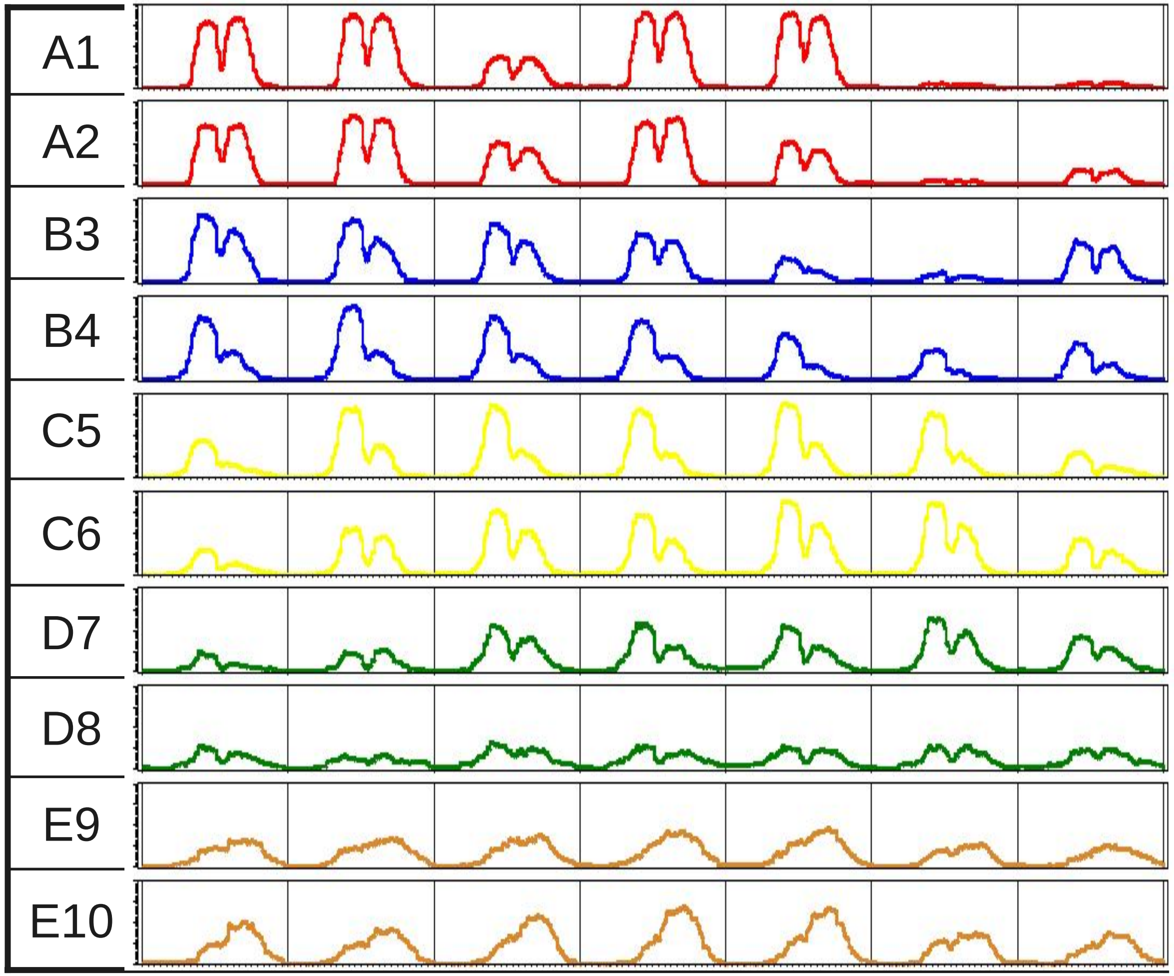
Each super-class groups 2 consecutive classes

10-unit Kohonen string (1-dim map)

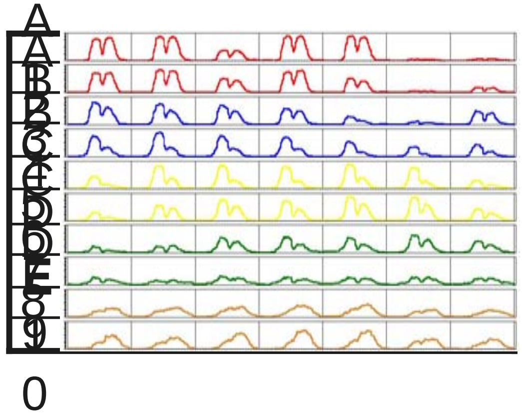
We represent the code-vectors

Y-axis is a number between 0 and 1, proportion of individuals in the class who are considered to be actively working at that moment

The organization is visible

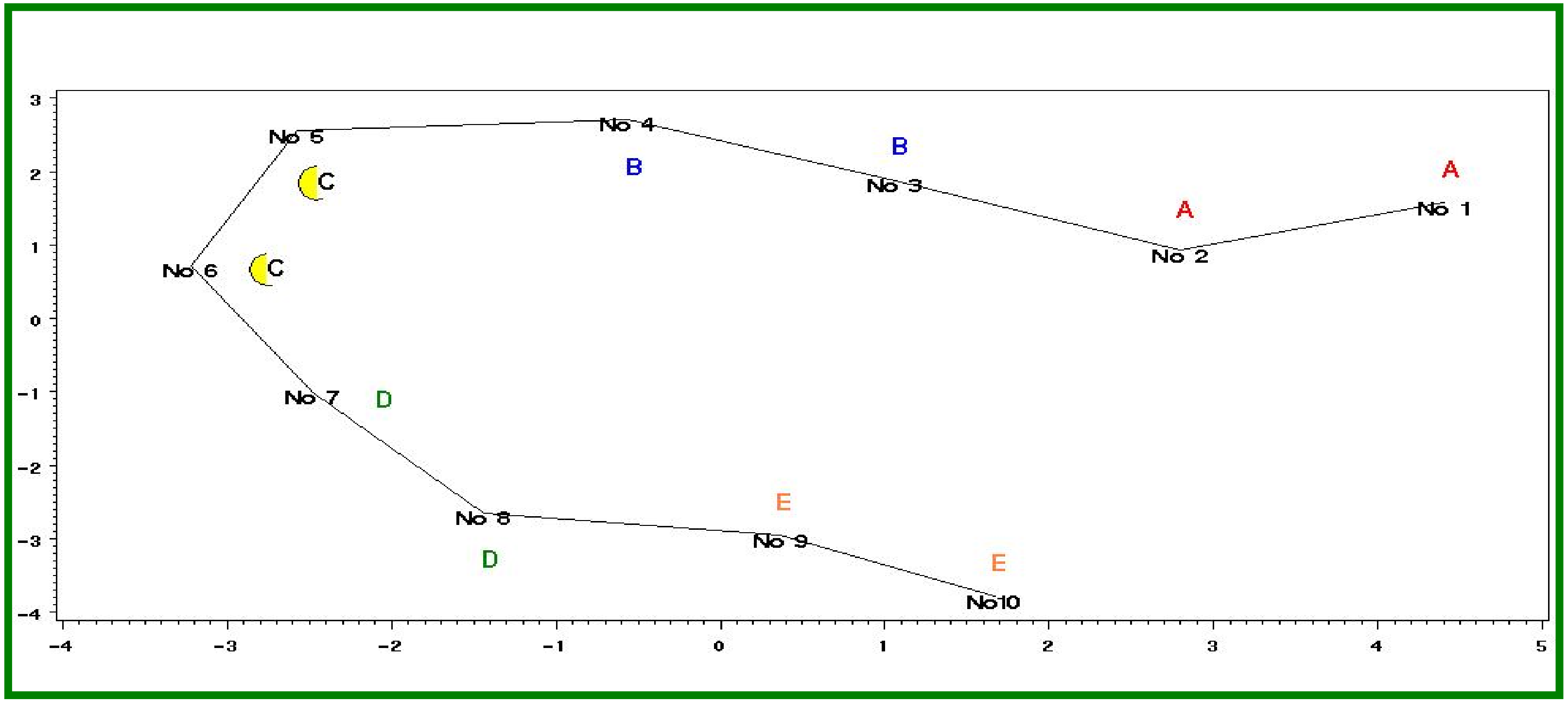


Typology of the super-classes



- **A**: Work neither nights, Saturdays or Sundays, and have a lower activity level on Wednesdays. The weekly report and the individual questionnaire converge.
- B: Do not work nights; very limited activity levels Saturdays (slightly less than in the questionnaire); mostly active in the morning the other days of the week. With respect to the Sunday question, a clear divergence between the weekly report (60% are at work at 10AM) and the questionnaire (77% state they never work Sundays).
- C: Do not work nights; mostly work Wednesday and Saturday morning with lower level of activity Sundays (and Mondays). With respect to the Sunday question, a slight divergence between the weekly report (30% are at work at 10AM) and the questionnaire (9% state they usually work Sundays).
- D: A little night work but less than in the questionnaire; reduced activity level Saturdays and Sundays; mainly work Wednesday mornings (much lower activity levels Mondays and Tuesdays). Little divergence between the weekly report and the questionnaire.
- E: A little night work (but more than in the questionnaire); mostly work Monday or Friday afternoons with lower activity levels Saturdays and Sundays; slight divergence for Wednesday between weekly report (48% are at work at 4PM) and the questionnaire (64% state they usually works Wednesdays).

Multidimensional Scaling projection for the 10 code-vectors



To verify the good organization of the 10 code-vectors

Ex 4: Forecasting for vectorial data with fixed size

- Problem : predict a curve (or a vector)

- Example : an electricity consumption curve for the next 24 hours, the time unit is the half-hour and one has to simultaneously forecast the 48 values of the complete following day (data from EDF, or from Polish consumption)

- First idea : to use a recurrence
  - Predict at time *t*, the value X_{t+1} of the next half-hour
  - Consider this predicted value as an input value and repeat that 48 times

- PROBLEM :
  - with ARIMA, crashing of the prediction, which converges to a constant depending on the coefficients
  - with neural non linear model, chaotic behavior due to theoretical reasons
- New method based on Kohonen classification

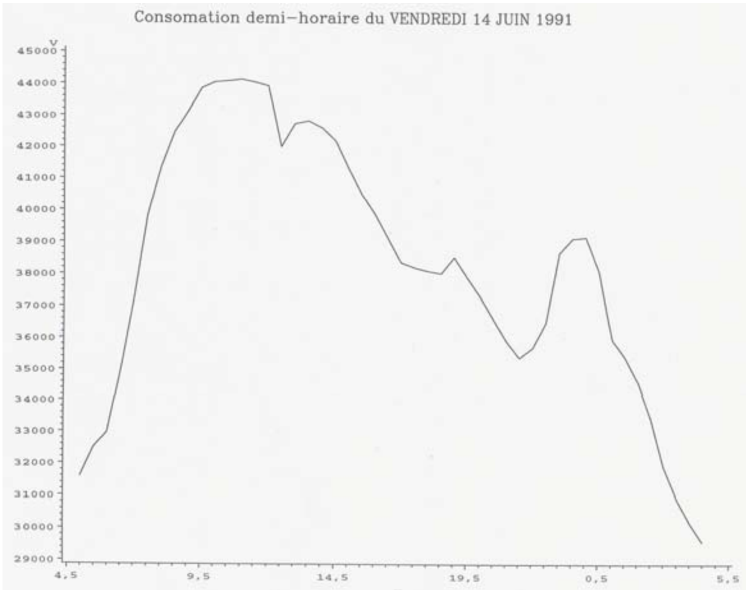
The data

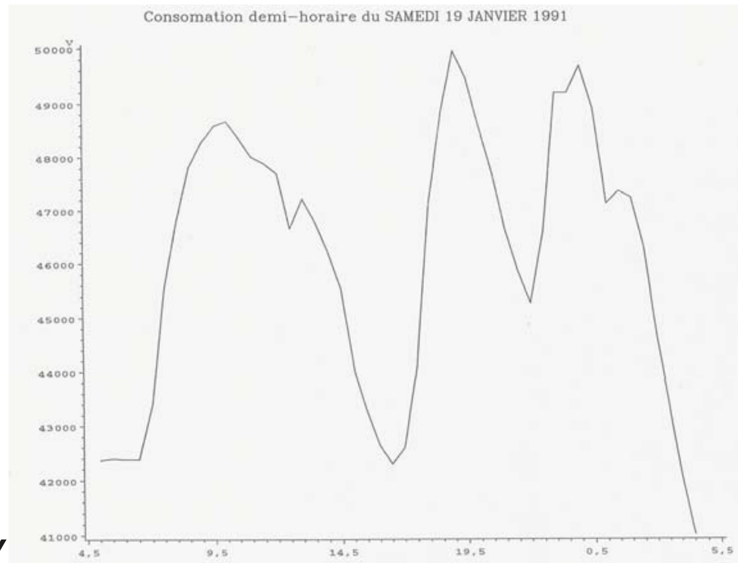
The power curves are quite different from one day to another

It strongly depends on

- the season
- the day in the week
- the nature of the day (holiday, work day, saturday, sunday, ...)

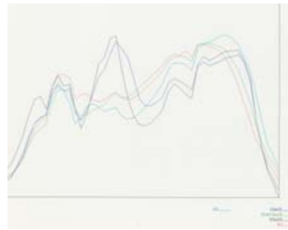
Friday in June





Saturday in January

Method



- **Decompose the curve into three characteristics**

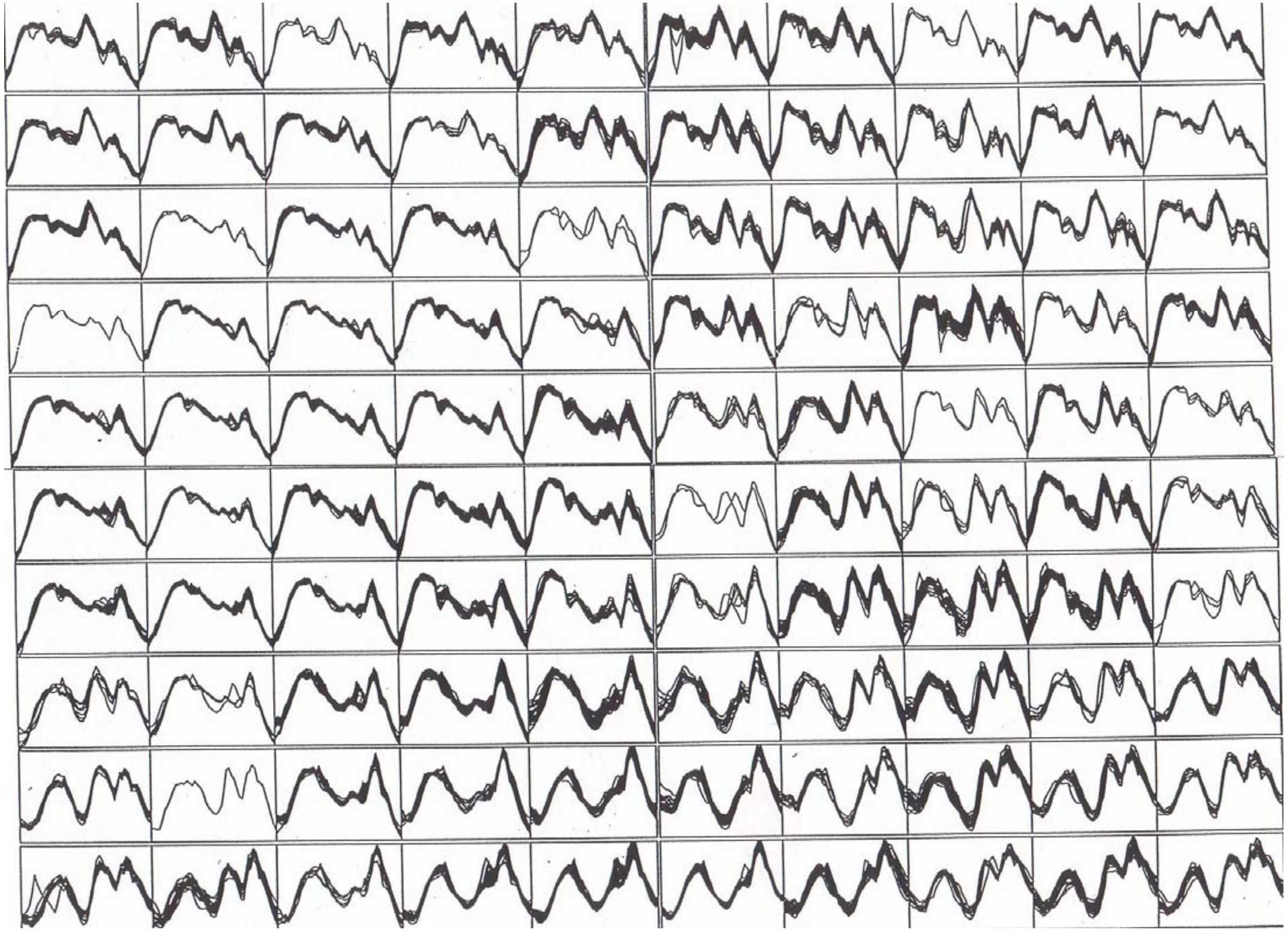
the mean m, the variance σ^2, the profile P defined by

$$P(j) = \left(P(j,h), h = 1, \cdots, 48\right) = \left(\frac{V(j,h) - m(j)}{\sigma(j)}\right)$$

j is the day, *h* is the half-hour

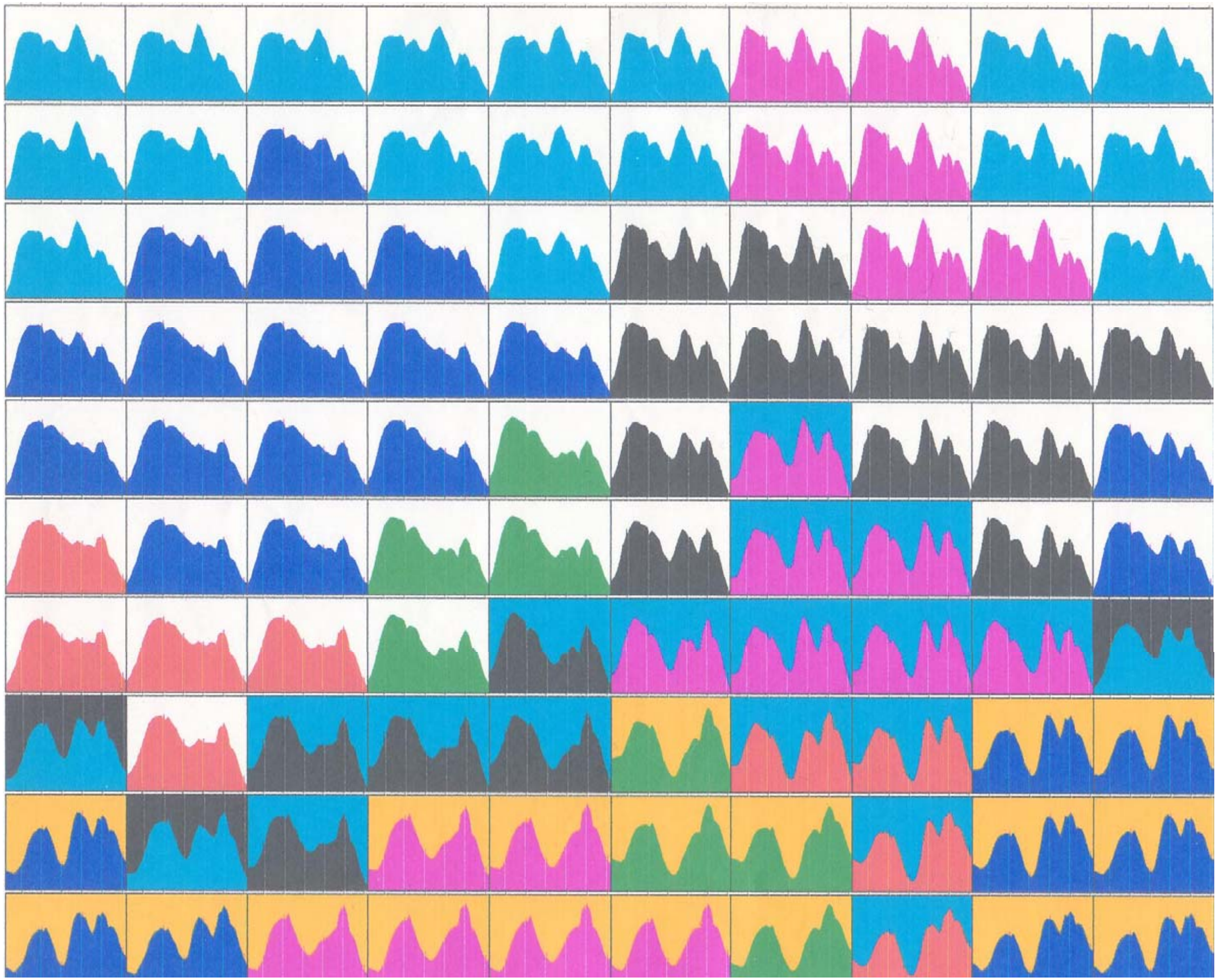
- The **mean** and the **variance** are forecast with an ARIMA model or with a Multilayer Perceptron, where the input variables are some lags, meteo variables, nature of the day
- The 48 - vectors are normalized to compute the profile: their norms are equal to 1
- **Achieve a classification of the profiles with cylindrical neighborhood**
- For a given unknown day, build its typical profile and redress it (multiply by the standard deviation and add the mean)
- Note that the origin is taken at 4 h 30: the value at this point is relatively stable from one day to another

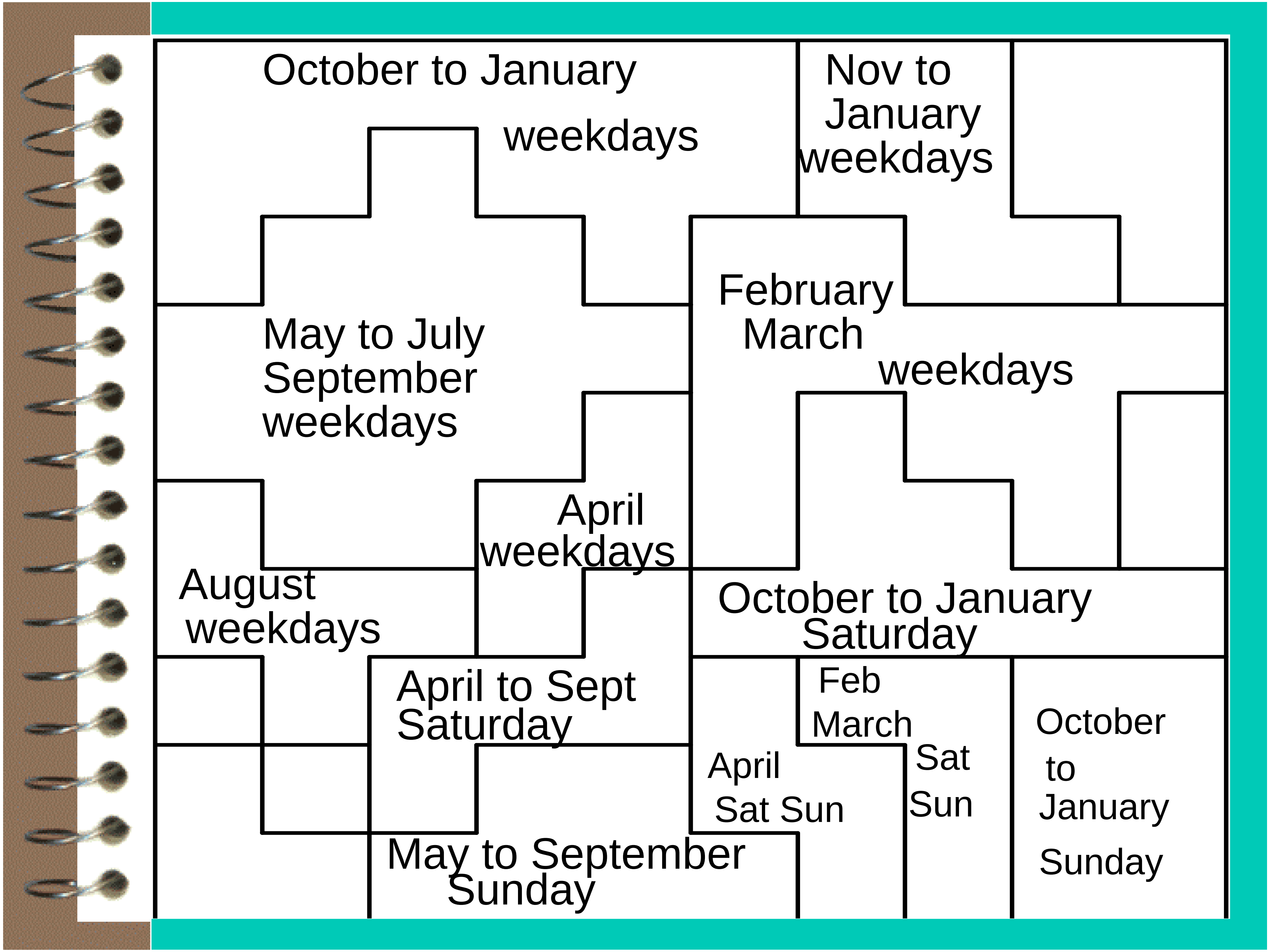
Classification of the profiles on a 10 by 10 cylinder



The distance between two profiles is computed with the same weight for each half-hour
The weather does not influence the profile : it acts only on the mean and the variance
Classification of the profiles, (vectors in R48, with norm 1, and sum 0)

13 super-classes on the cylinder (that we can describe)




October to January
weekdays
Nov to
January
weekdays
May to July
September
weekdays
February
March
weekdays
April
weekdays
August
weekdays
October to January
Saturday
April to Sept
Saturday
Feb
March
Sat
Sun
October
to
January
Sunday
April
Sat Sun
May to September
Sunday

Forecasting: corrected curves

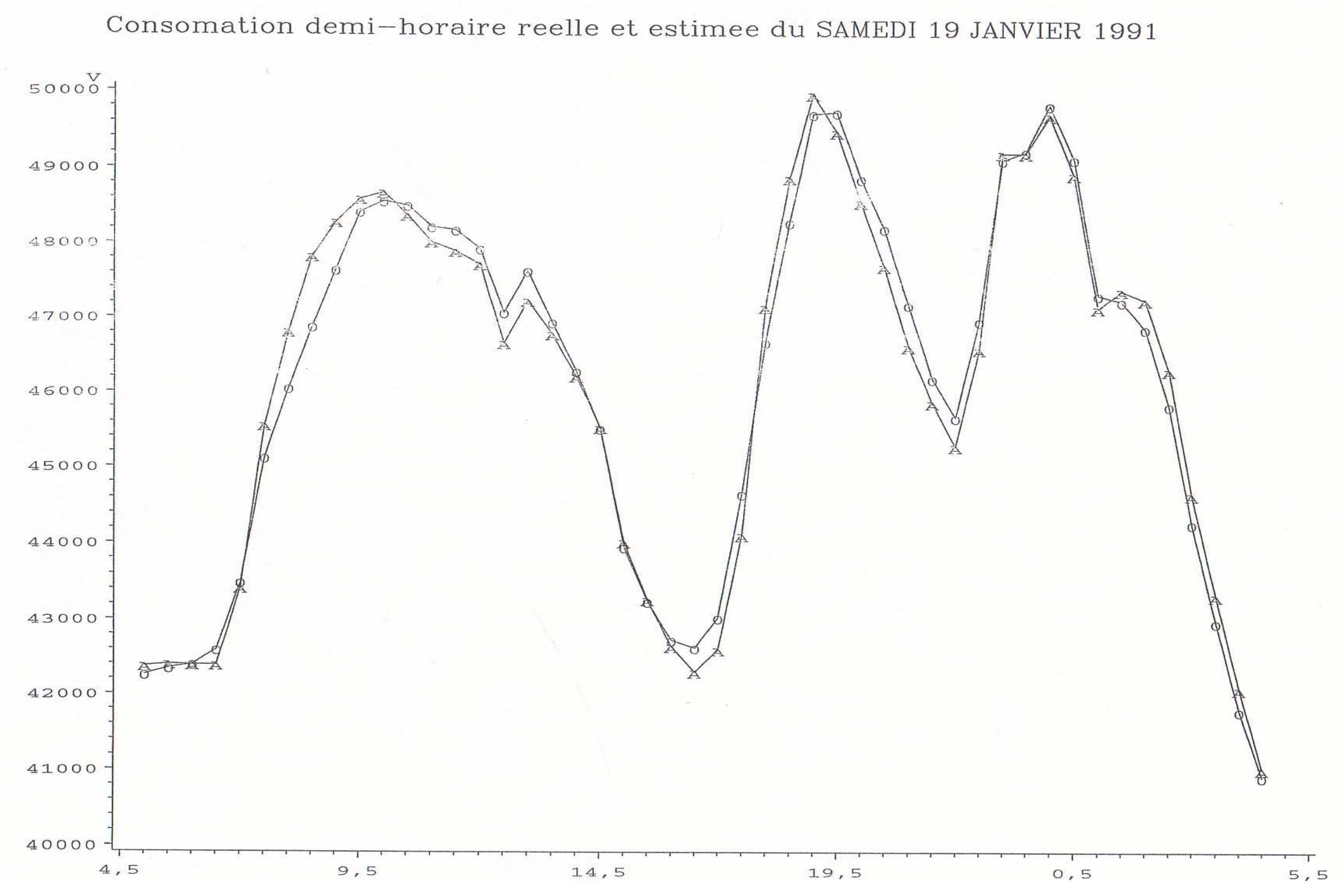
- For a day j, let a_{ji} be the number of instances of the day j in the class i
- Let C_i be the code-vector of the class i
- The estimated profile of the day j is

$$\hat{P}(j) = \frac{\sum_{i=1}^{100} a_{ji} C_i}{\sum_{i=1}^{100} a_{ji}}$$

- This profile is corrected and the forecasted curve is

$$\hat{V}(j) = \sigma(j)\hat{P}(j) + m(j)$$

Examples of real and forecast curves



Ex 5: Shocks on the Interest Rate Structure

Use of double SOM for prediction

- The shocks of the ***interest rate structure*** (or deformations, or changes) are very important, since they modify the values of financial assets
- Using Gaussian hypothesis, we can use a Monte Carlo simulation which is well adapted for short-term horizon, but fails when applied to long-term prediction
- One can observe scenarios which tend to be explosive. They can give also negative values.

- So to generate long-term scenarios still remain a problem hard to solve. ***Clearly, there are many complex relations which cannot be taken into account in the usual modeling by Gaussian distribution***

Generation of long-term paths

- We propose a method to generate ***long-term paths*** of interest rate structures, ***positive for every step*** (to avoid any arbitrage), in order to evaluate path dependent asset prices over time.

- It is a **non parametric approach**, without any a priori hypothesis (neither on the process, neither on the dynamics)
- It is only based on past observed values

short rate (1 year): blue
long rate (15 years): pink

The Data set: 2088 interest rate structures of the US Treasury Notes and Bonds market between 1/5/1987 to 5/10/1995 (from one year to 15 years maturities).

Simulation Algorithm

- ***Interest rate shocks*** : deformations (or differences) of the interest rate structure on a 10 days lag, as recommended by the Basle Committee on Banking Supervision.

$$\text{Shock} = \text{Curve}(t) - \text{Curve}(t-10)$$

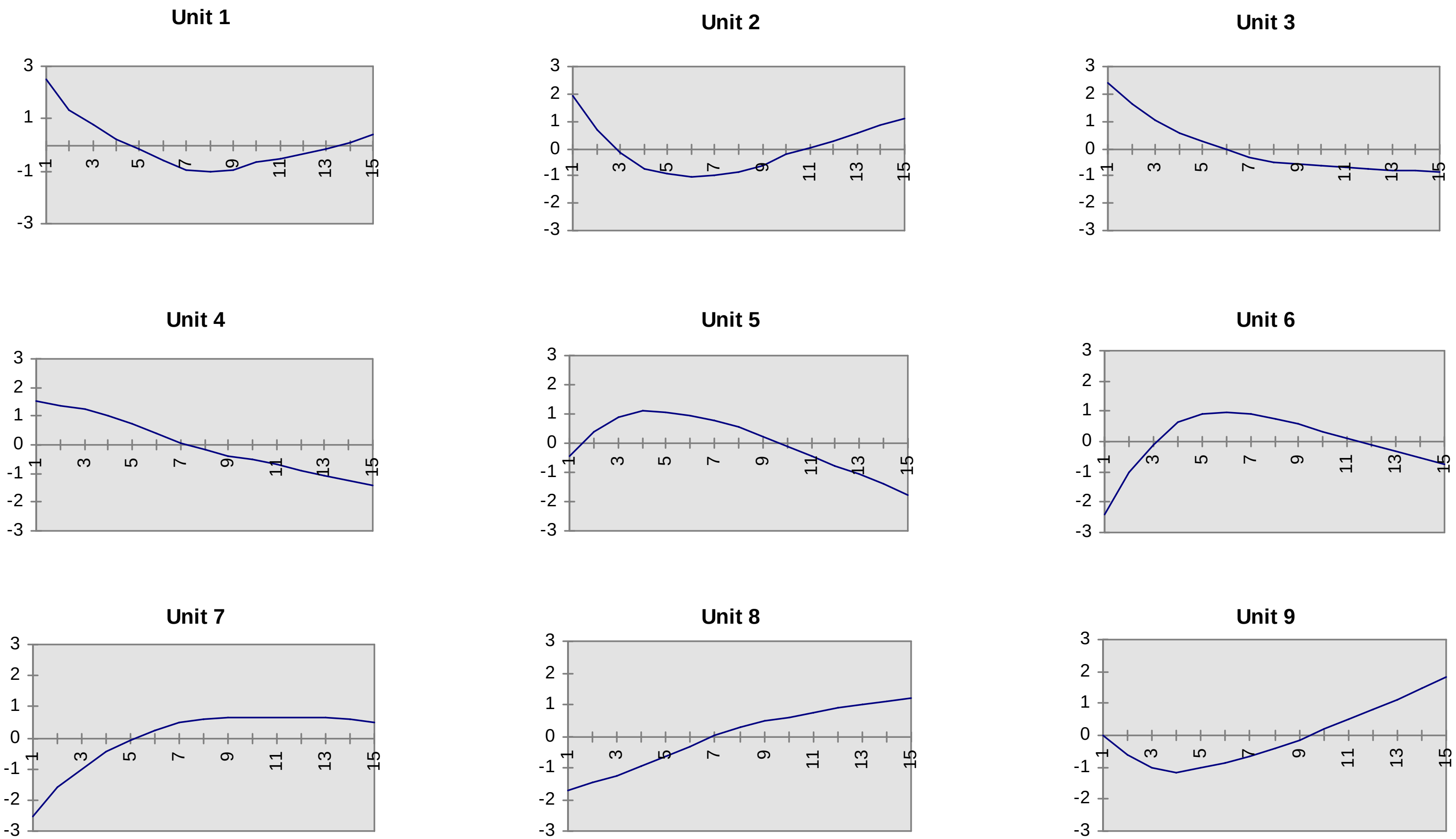
- To generate long-term path of interest rates structures, we use two SOM algorithms to form two classifications

- One for the initial interest rate curve,
- One for the shocks of these curves

- We compute the distributions of the deformations conditionally to the initial structures
- We generate paths of the process, using Monte Carlo procedure

Classification of the initial structures into 9 classes after neutralization of the short term level

It is clear that the explicative variables for the clustering are the slope (linked to the ***spread***, i.e. the difference between the long term rate and the short term one) and the ***curvature***.



1-dimensional Kohonen map, with 9 classes

The Conditional Probabilities

- Then, we classify the shocks using a 30-units one-dimensional SOM map. The number 30 (as well as 9 before) is chosen in order to maximize the indicators of good homogeneity and strong discrimination of the classes (Fisher, Wilks Statistics)
- The discrimination comes from the ***spread***, the ***curvature*** and the ***level of the short rate***
- Each shock can be associated to its initial curve (10 days before), and this fact allows to ***count*** the number of shocks of each class (among 30) associated to an initial structure belonging to each class (among 9)
- ***So it is possible to compute the 9 empirical conditional distributions of shocks***
- We verify that they are all strongly different (without assuming it as an hypothesis). ***Relations between shocks and initial rate structures are confirmed*** .

Conditional Probabilities

	Classes of IIRS									
Classes of IRS	1	2	3	4	5	6	7	8	9	Population
1	0.00%	0.00%	0.00%	0.00%	1.81%	0.00%	0.00%	0.00%	0.00%	0.34%
2	2.37%	0.00%	0.00%	0.41%	2.33%	0.00%	0.00%	0.00%	0.00%	0.67%
3	7.10%	0.00%	0.00%	0.83%	0.26%	0.71%	0.00%	1.07%	2.01%	1.11%
4	1.18%	0.00%	0.88%	4.15%	2.07%	0.00%	1.88%	2.14%	2.01%	1.59%
5	1.78%	2.46%	1.75%	2.49%	3.10%	0.35%	0.00%	3.74%	0.00%	1.83%
6	5.33%	2.11%	1.75%	3.32%	3.36%	3.53%	0.47%	3.21%	4.02%	3.03%
7	1.18%	0.00%	0.00%	0.83%	2.33%	0.00%	1.88%	2.67%	0.00%	1.06%
8	7.10%	0.00%	0.00%	0.41%	0.00%	1.41%	0.00%	0.00%	6.03%	1.40%
9	7.69%	9.47%	7.89%	5.81%	3.10%	7.07%	8.45%	2.14%	6.03%	6.21%
10	6.51%	8.77%	13.16%	8.71%	8.53%	6.36%	7.04%	5.35%	2.51%	7.36%
11	1.18%	7.02%	7.89%	5.39%	6.72%	8.13%	4.69%	5.35%	2.01%	5.63%
12	6.51%	3.16%	3.51%	8.71%	4.65%	1.77%	5.16%	6.95%	6.03%	5.00%
13	3.55%	5.61%	1.75%	7.47%	2.58%	4.59%	0.47%	5.35%	5.53%	4.19%
14	1.78%	3.16%	0.00%	3.32%	2.84%	0.35%	1.41%	7.49%	1.51%	2.50%
15	5.33%	2.46%	3.51%	2.07%	0.78%	3.53%	0.47%	5.35%	4.52%	2.79%
16	0.00%	0.35%	0.00%	0.00%	2.33%	0.35%	0.00%	1.07%	0.00%	0.63%
17	2.96%	5.26%	4.39%	3.32%	6.72%	9.19%	7.98%	5.88%	3.02%	5.73%
18	3.55%	5.61%	10.53%	6.64%	3.36%	8.13%	15.49%	4.28%	4.52%	6.54%
19	7.69%	4.91%	6.14%	4.98%	2.07%	4.95%	4.23%	4.28%	6.03%	4.67%
20	4.73%	3.86%	5.26%	7.05%	2.84%	8.13%	4.69%	5.88%	9.55%	5.58%
21	1.18%	1.75%	0.88%	3.73%	4.91%	2.12%	1.88%	1.60%	1.51%	2.50%
22	5.33%	4.21%	0.88%	0.41%	1.55%	3.89%	1.41%	5.35%	6.53%	3.18%
23	4.73%	6.67%	3.51%	4.15%	6.46%	7.77%	8.45%	4.81%	7.54%	6.26%
24	1.18%	3.86%	1.75%	2.49%	8.01%	2.47%	4.23%	5.35%	5.03%	4.23%
25	4.14%	5.61%	7.89%	6.64%	3.10%	8.48%	4.69%	3.21%	7.54%	5.53%
26	1.78%	3.86%	0.00%	0.41%	2.58%	2.47%	3.76%	3.74%	4.02%	2.65%
27	2.96%	3.51%	4.39%	2.90%	4.91%	2.47%	4.23%	1.60%	0.00%	3.13%
28	1.18%	4.56%	3.51%	3.32%	2.58%	0.35%	5.16%	0.00%	2.01%	2.55%
29	0.00%	1.75%	2.63%	0.00%	3.10%	1.06%	1.41%	1.60%	0.50%	1.44%
30	0.00%	0.00%	6.14%	0.00%	1.03%	0.35%	0.47%	0.53%	0.00%	0.67%
Total	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%

Test du Chi2

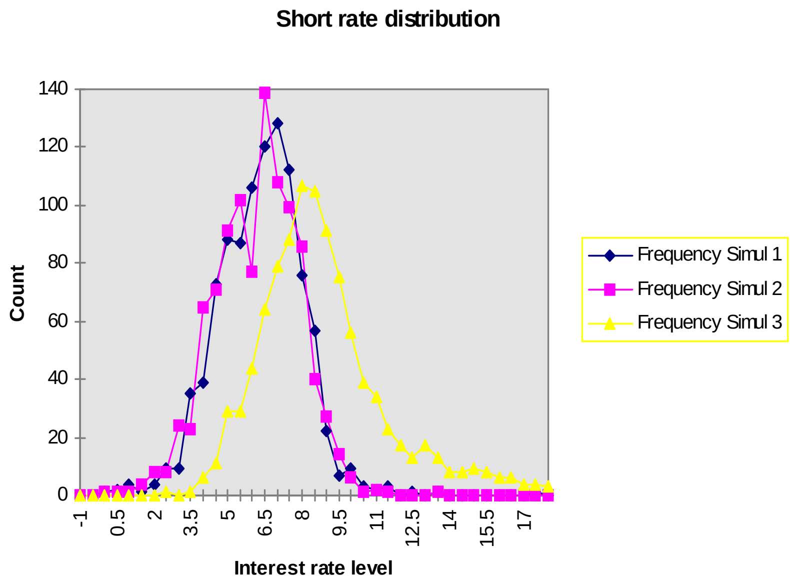
	Chi2
Unit1	140.00
Unit2	42.04
Unit3	85.48
Unit4	54.58
Unit5	151.07
Unit6	244.39
Unit7	55.38
Unit8	73.37
Unit9	59.87

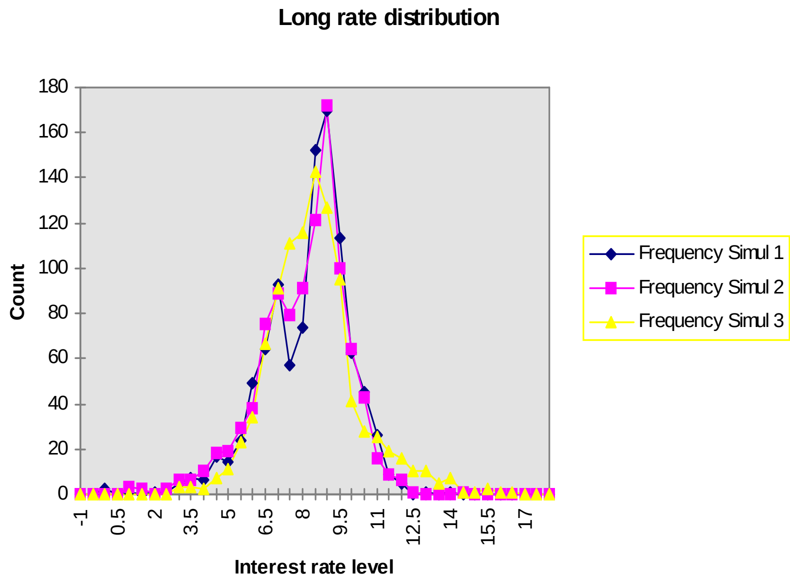
We can reject the independence hypothesis at a very high level of confidence (+/-100%

Simulating Paths on a Long Term Horizon

- ***The procedure is the following :***
- Randomly draw an initial interest rate structure (IIRS)
- Determine the winning unit of the SOM map associated with the initial interest rate structure (9 classes)
- According to the conditional distribution of frequencies of the interest rate shocks, randomly draw a shock.
- Apply the shock to the interest rate structure, take the result as IIRS
- Repeat the procedure 125 times to construct an interest rate structure evolution on a five year horizon (125 times the 10 days covered by the interest rate shock).
- For each simulation, repeat the procedure 1000 times to build the distribution of probability of interest rate structures, starting from the same initial interest structure.

Examples: distributions of 1-year rate and 15-years rate observed on the simulation





- The simulation procedure is stable.
- Short rates are clearly more volatile than long rates.

Conclusion

- ***We shown that the method does not generate any explosive path, even on a long-term horizon***
- ***No hypothesis is necessary (as mean reverting, or equilibrium, or Gaussian) to get only positive values***

- We have compared with the well-known Cox Ingersol Ross (CIR) interest rate model and use the moment method to verify that the theoretical and the simulated paths have the same properties. The results are quite good. ***The validation is conclusive.***
- We could get similar results using any method to compute the conditional probabilities, but the SOM classification is quite satisfactory and allows nice representation and easy interpretation of the classes

Ex 6: Adaptation of the Kohonen algorithm when there are missing values

- x : p-dimensional data vector with missing values,
- M_x : set of the numbers of missing components, subset of {1, 2, ..., p}.
- $(C_1, C_2, ..., C_n)$: code-vectors of each class
- A each step, we randomly draw an observation x
- Compute the winning code-vector associated to x defined by

$$i_0(C, x) = Arg\min_i \|x - C_i\|$$

- where the distance

$$\|x - C_i\|^2 = \sum_{k \notin M_x} (x_k - C_{i,k})^2$$

is computed on the non-missing components of vector x.

Two ways of using

First way of using

- The observations which contain missing components are used for the learning stage together with the full data.
- The distance is restricted to non-missing values, the winning code-vector is defined as above, the updating is carried out only for non-missing components.
- For $j = i_0$, or j neighbor of i_0, and for $k \notin M_{x}$,

$$C_j(t+1) = C_j(t) + \varepsilon(t+1)(x(t+1) - C_j(t))$$

Second way of using

- The observations which contain too many missing values are not used for the learning stage.
- They are classified after computing the code-vectors, as additional observations, with the nearest neighbor rule, restricted to the non-missing components.
- We put x in class i, where i is the winning unit (minimum of distance)

Estimation of the missing values

- If *x* is classified into class *i*, for each index *k* in M_x, one estimates x_k by :

$$\hat{x}_k = C_{i,k}$$

- At the end of learning, there is « 0 neighbor », so the code-vectors are asymptotically near the class means.

- This method for estimation consists in estimating the missing values of a variable by the **mean value of its class.**

- It is clear that this estimation is all the more accurate as the classes are homogeneous and well separated one.

- As it is very easy to consider a large number of classes, the classes are small in general, and the variance of each class is small.

Computation of membership probabilities

- It is also possible to use a probabilistic classification rule, by computing the membership probabilities for the supplementary observations (be they complete or incomplete), by putting:

$$P(x \in \text{Class } i) = \frac{\exp(-\|x - C_i\|^2)}{\sum_{k=1}^{n} \exp(-\|x - C_k\|^2)}$$

- Confirmation of the quality of the organization, since significant probabilities correspond to neighboring classes.
- To estimate the missing values, one can compute the weighted mean value of the corresponding components.
- If x is an incomplete observation, and for each k in M_x, one estimates x_k by :

$$\hat{x}_k = \sum \text{P}(x \in \text{Class } i)\ C_{i,k}$$

- Distribution, confidence intervals, etc.

Example : the countries

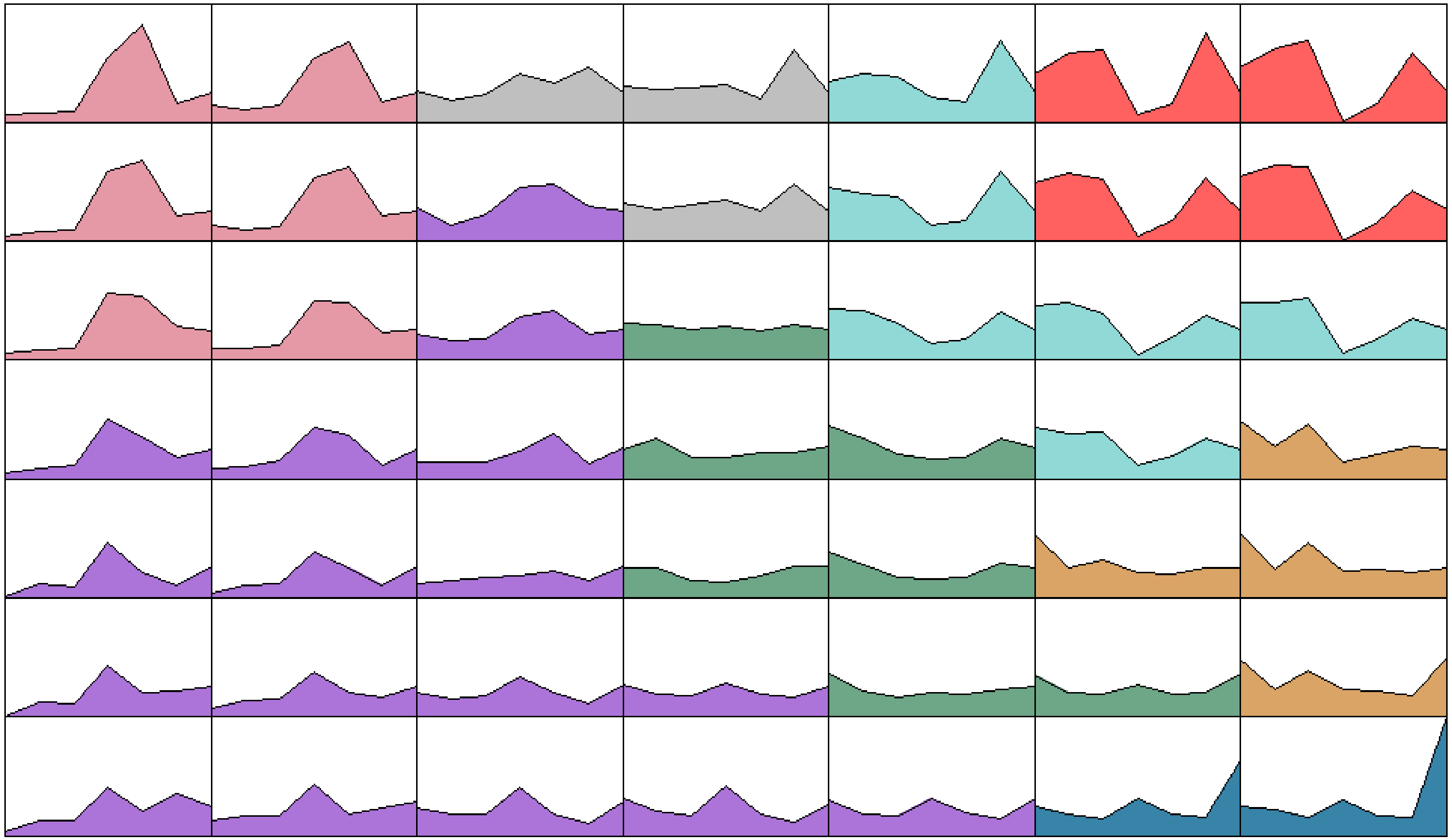
- 182 countries in 1996
  – each country is described by seven ratios
  – 114 countries with no-missing values
  – 52 countries with only one missing value
  – 16 countries with 2 or more than 2 missing values

- The measured variables are:
  – annual population growth (ANCRX),
  – mortality rate (TXMORT),
  – illiteracy rate (TXANAL),
  – population proportion in high school (SCOL2),
  – GDP per head (PNBH),
  – unemployment rate (CHOMAG),
  – inflation rate (INFLAT)

Learning: computation of the code-vectors

- One uses the 114 + 52 = 166 countries which are complete or almost complete (no more than one missing value)
- Centered and standardized data
- Kohonen map with 7 by 7 units
- Rich countries in the top left hand corner, very poor ones in the top right hand corner. Ex-socialist countries are not very far from the richest, etc.
- As for the 16 countries which are classified after the learning as supplementary observations, the logic is respected.
- Monaco and Vatican are displayed with rich countries, and Guinea with very poor countries, etc.

The code-vectors after convergence (1500 iterations)



The 182 countries on the map

Danemark Japon Liechtenstein Luxembourg Suisse	Emirats arabes uni Etats Unis Islande Saint-Marin *Monaco* *Vatican*	Sainte-Lucie *Palau*	Afrique du sud Iran Liban Macedoine	Inde Indonesie Namibie Zimbabwe	Banglade Djibouti Haiti Malawi Mozambiq Nepal Senegal *Bouthan* *Cambodge* *Turkmeni*	Burkina Faso Erythree Mali Niger Sierra Leone Somalie Soudan Tchad *Guinee*
Allemagne Autriche Belgique France Norvege Pays Bas Suede	Australie Canada	Bahamas Bahrein Israel Qatar	Algerie Egypte	Cameroun Cap-Vert Honduras Kenya Nigeria *Micronesie*	Burundi Rwanda Tanzanie Togo Yemen	Afghanistan Angola Benin Ethiopie Gambie Liberia
Espagne Finlande Irlande	Italie Nouvelle Zelande Royaume Uni	Argentine Brunei Koweit	Mongolie Perou Tunisie Turquie	Birmanie Congo Laos Lesotho Nicaragua	Gabon Madagascar Ouganda Zambie	Comores Cote d' Ivoire Ghana Mauritanie Pakistan
Grece Slovenie	Chypre Coree du Sud Malte Portugal Singapour Taiwan	Andorre *Nauru*	Bolivie Bresil Tuvalu	Maldives Marshall	Maroc Papouasie Swaziland	Guatemala Irak Salomon Vanuatu
Bielorussie Estonie Lettonie Lituanie Russie Tcheque (Rep) Ukraine	Moldavie *Cuba*	Bosnie Thailande *Seychelles* *Tonga*	Albanie Guyana *Samoa* *Surinam*	Paraguay Vietnam	Jordanie Syrie	Arabie Saoudite Botswana Oman *Libye*
Croatie Hongrie Roumanie	Uruguay	Chili Chine Fidji Kirghizstan Maurice	Colombie Equateur Mexique Panama *Coree du Nord*	Belize Salvador Venezuela		
Bulgarie Dominique Grenade Jamaique Pologne Slovaquie Yougoslavie	Kazakhstan Sri Lanka	Azerbaidjan	Kiribati Ouzbekistan Philippines Tajikistan	Costa Rica Malaisie	Armenie	Georgie Zaire

Control of organization (heuristic)

Significant membership probabilities

Danemark Japon Liechtenstein Luxembourg Suisse	Emirats arabes uni Etats Unis Islande Saint-Marin *Monaco* *Vatican*	Sainte-Lucie *Palau*	Afrique du sud Iran Liban Macedoine	Inde Indonesie Namibie Zimbabwe	Banglade *Turkmenst* Djibouti Haiti Malawi Mozambiq Nepal Senegal *Bouthan* *Cambodge*	Burkina Faso Erythree Mali Niger Sierra Leone Somalie Soudan Tchad *Guinee*
Allemagne Autriche Belgique France Norvege Pays Bas Suede	Australie Canada	Bahamas Bahrein Israel Qatar	Algerie Egypte	Cameroun Cap-Vert Honduras Kenya Nigeria *Micronesie*	Burundi Rwanda Tanzanie Togo Yemen	Afghanistan Angola Benin Ethiopie Gambie Liberia
Espagne Finlande Irlande	Italie Nouvelle Zelande Royaume Uni	Argentine Brunei Koweit	Mongolie Perou Tunisie Turquie	Birmanie Congo Laos Lesotho Nicaragua	Gabon Madagascar Ouganda Zambie	Comores Cote d' Ivoire Ghana Mauritanie Pakistan
Grece Slovenie	Chypre Coree du Sud Malte Portugal Singapour Taiwan	Andorre *Nauru*	Bolivie Bresil Tuvalu	Maldives Marshall	Maroc Papouasie Swaziland	Guatemala Irak Salomon Vanuatu
Bielorussie Estonie Lettonie Lituanie Russie Tcheque (Rep) Ukraine	Moldavie *Cuba*	Bosnie Thailande *Seychelles* *Tonga*	Albanie Guyana *Samoa* *Surinam*	Paraguay Vietnam	Jordanie Syrie	Arabie Saoudite Botswana Oman *Libye*
Croatie Hongrie Roumanie	Uruguay	Chili Chine Fidji Kirghizstan Maurice	Colombie Equateur Mexique Panama *Coree du Nord*	Belize Salvador Venezuela		
Bulgarie Dominique Grenade Jamaique Pologne Slovaquie Yougoslavie	Kazakhstan Sri Lanka	Azerbaidjan	Kiribati Ouzbekistan Philippines Tajikistan	Costa Rica Malaisie	Armenie	Georgie Zaire

- Cuba : proba > 0.03, (blue), 0.06 for class (5, 2) in yellow

Ex 7:Classifications of trajectories in the market labor

- 3% sampling of a very large survey (one million of observations), i.e. 30 000 individuals
- 14 variables from more than one hundred
- Between 1990 and 2002, all the active population except farmers
- One observation is a couple (individual, year), each one is present 3 consecutive times
- The variables are: full or part-time, employment or not, reason if not, job contract, and so on.
- All the variables are categorical, so we first transform the data into real-valued ones by using a Multiple Correspondence Analysis and keeping all the components for the computation (not for the visualization)

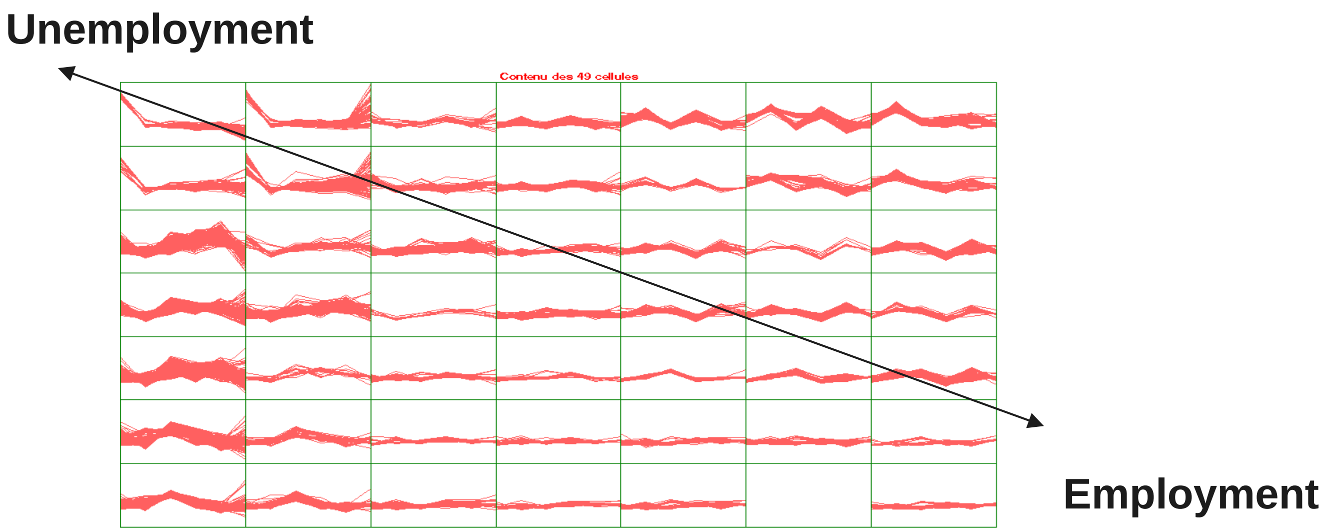
SOM over the MCA components

- Axis 1: **Unemployed**, employed
- Axis 2: Craftsmen, trade profession, **self-employed**
- Axis 3: **Clerks**, part-time contracts
- Axis 4: : Managers, **professionals**, full-time, not fixed term contracts
- Axis 5: **Operatives**, fixed-term contracts, occasional works
- Axis 6: Seniority in unemployment, with **alimonies**

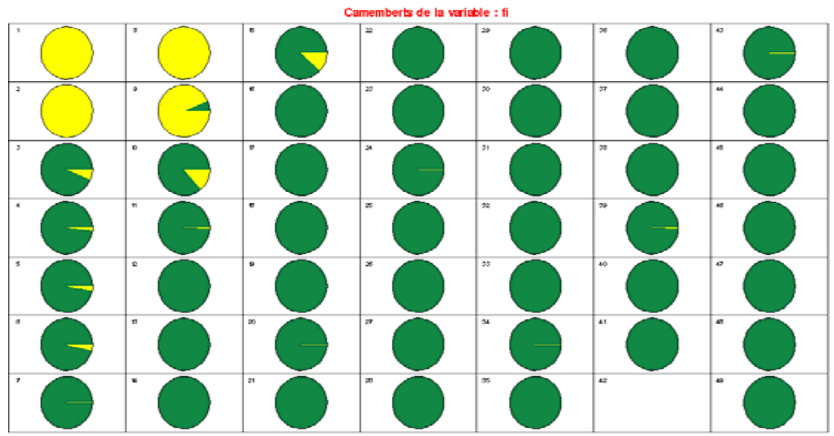
- We classify the 30 000   45-dimensional observations on a **(7 × 7) Kohonen map**.
- 100 000 iterations (about 3 by observation).
- We use the 45 components to keep the whole information.
- **No other preprocessing**, they are already centered, **no weighting** (the first components which are the most significant and the most variant will have naturally more weight).
- In the following graphs, we only display the 6 first components of each observation (those with the largest variance).

SOM (first 6 components), 49-classes representation

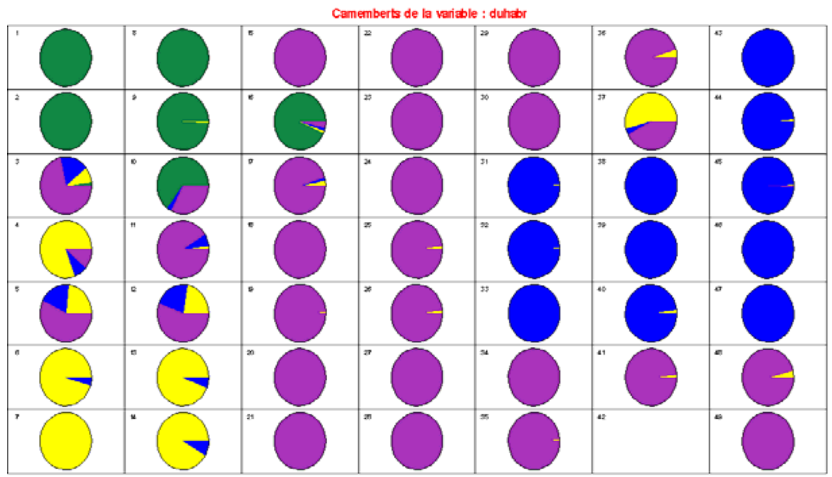
- Contiguity on the Kohonen map
- At the left top, **unemployed people**
- At the right top, **craftsmen**, **self-employed**
- In the middle, the most frequent situation: **operatives, clerks** at the bottom
- In the right middle, the **managers**, the professionals and technical workers,
- At the left bottom, the **part-time contracts**, etc.



Distribution of the variable job-no job: yellow = unemployed, green = employed

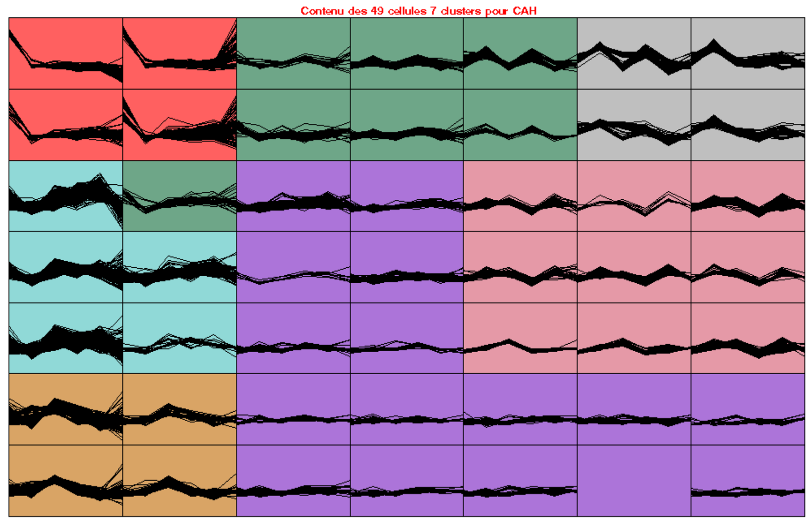


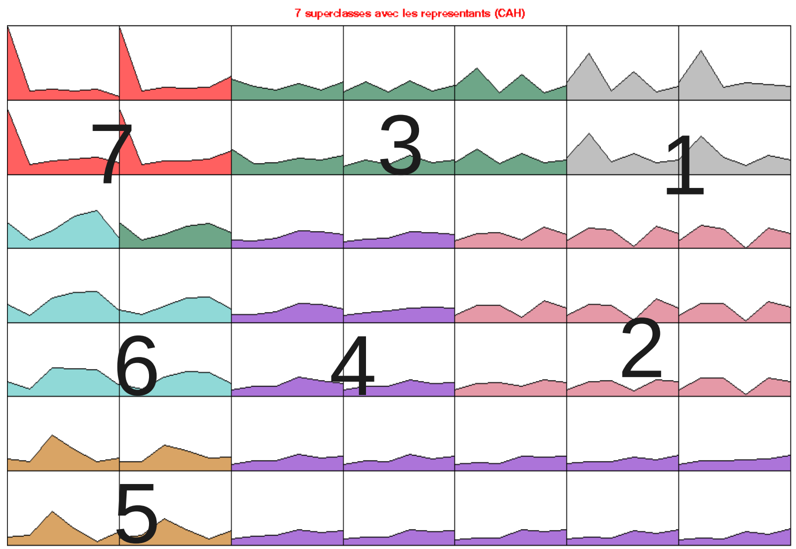
Part-time (yellow), no employment (green), full-time (violet, blue)



Defining 7 clusters

- Hierarchical classification of the 49 code-vectors, to get 7 clusters (70% of explained variance)
- These clusters can be considered as the segments of the labor market
- They only contain contiguous classes (self-organizing property)
- Easy to describe



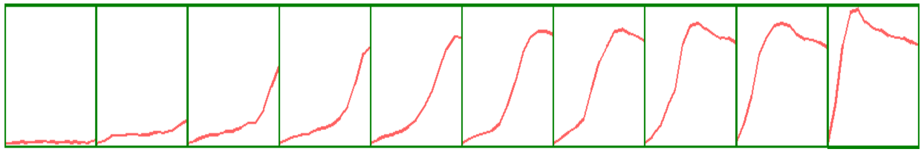


1: self-employed, 2: professionals,
3: managers
4 to 7: clerks and operatives from the best to the worst situation

Transition probabilities, simulated trajectories and classification

- In fact, each individual is questioned three consecutive years.
- His changes of clusters are **observed only three times,** so we have to reconstruct the trajectories (we cannot observe them).
- Idea: to model these changes by a Markov chain,
  – To simulate trajectories by estimating the probability transitions
  – To compute a limit distribution (if conditions do not change)

For each initial state in 1990,
we consider the set of trajectories
which start from it, and we build
7 one-dimensional Kohonen maps
with 10 units,
one map by starting situation

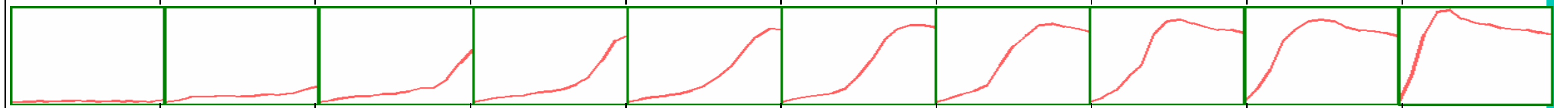
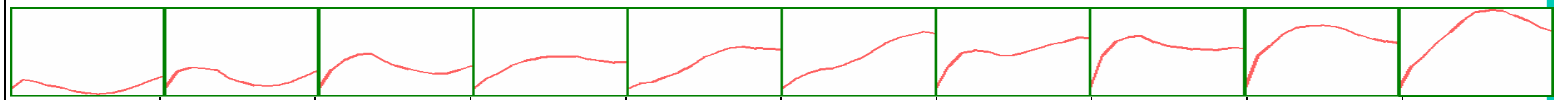
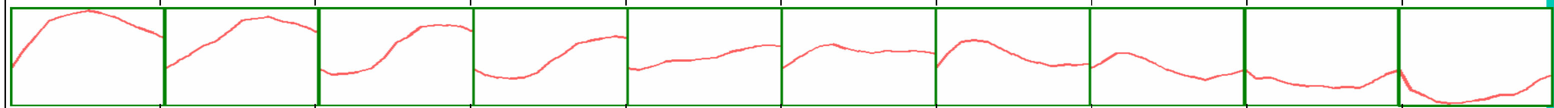
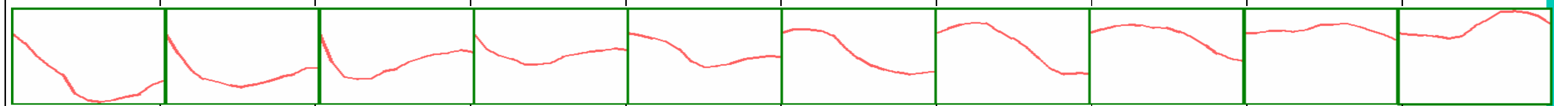
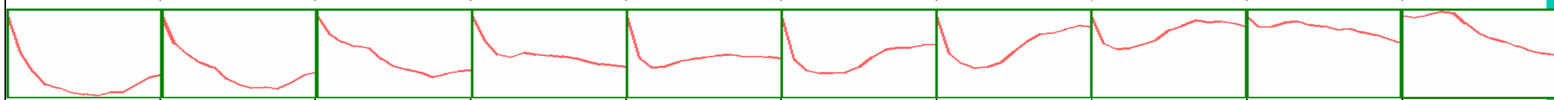
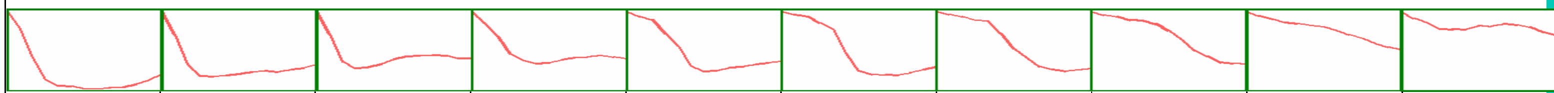


22.8 %	6.3 %	11.4 %	7.8 %	9.7 %	9.1 %	7.5 %	8.5 %	8.5 %	8.4 %

The code-vectors reveal typical
dynamical behaviors

Trajectories along 13 years, starting from C1 in 1990
9.43% of the population belong to C1 in 1990
Class 1 is stable for 23% of the initial population

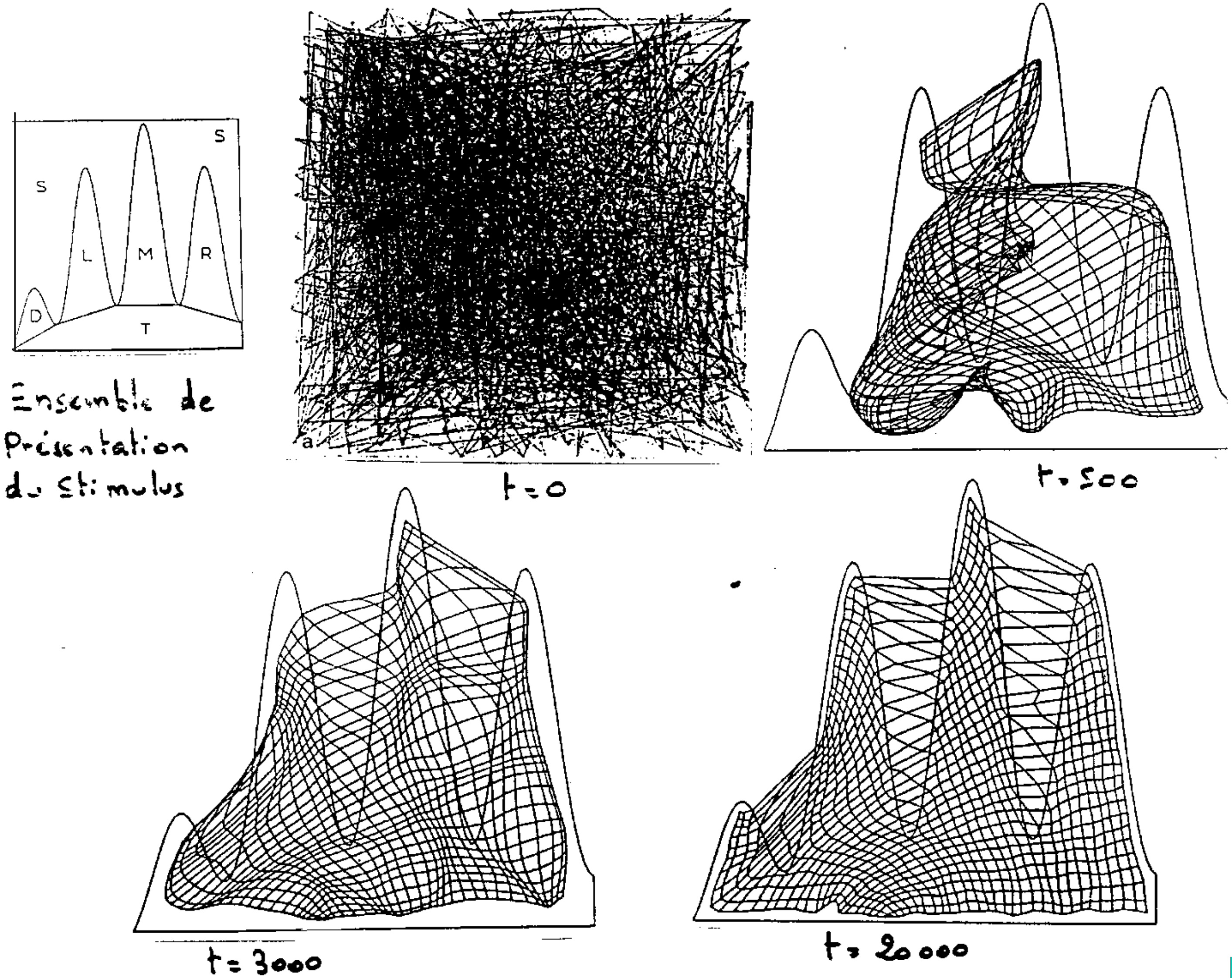
Classification of trajectories

	No 1	No 2	No 3	No 4	No 5	No 6	No 7	No 8	No 9	No10
C1  9.43%	22.81%	6.29%	11.42%	7.85%	9.69%	9.09%	7.53%	8.45%	8.49%	8.38%
C2  17.41%	9.02%	7.91%	9.54%	11.41%	13.52%	9.82%	11.76%	11.24%	7.47%	8.31%
C3  9.36%	9.62%	7.26%	8.62%	10.79%	13.14%	15.53%	8.19%	9.37%	6.87%	10.61%
C4  40.72%	8.82%	7.98%	8.56%	6.85%	9.44%	20.61%	13.97%	7.33%	7.83%	8.60%
C5  7.56%	9.22%	8.25%	9.26%	11.51%	12.44%	8.87%	7.72%	11.65%	12.00%	9.09%
C6  8.68%	11.29%	7.57%	9.91%	15.05%	13.79%	11.25%	8.29%	6.37%	6.99%	9.49%
C7  6.85%	9.44%	8.08%	13.14%	11.39%	11.48%	9.05%	9.54%	9.88%	8.22%	9.78%

Conclusion

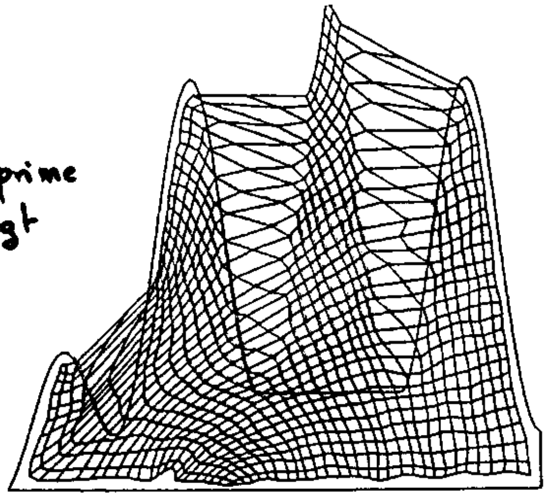
- SOM is a good candidate for data mining, classification, visualization of large dimensional data
- Can be used for prediction
  - For fixed-size vectors
  - For complex behavior usinf the initial curves and their possible deformations
- Can be adapted to qualitative variables, by using Multiple components analysis, or by directly using the Burt tables or the Complete Disjonctive tables

Adaptatif (Ritter et Schulten)

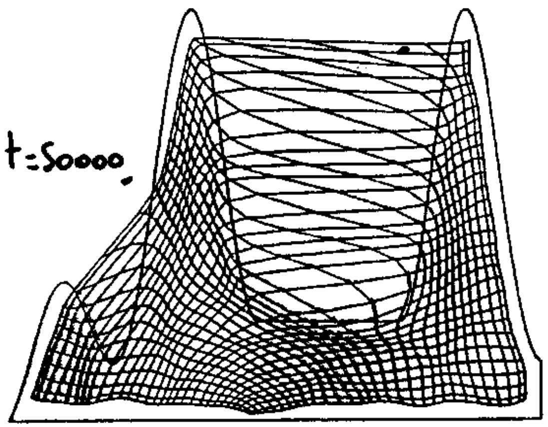


Adaptatif (Ritter et Schulten)

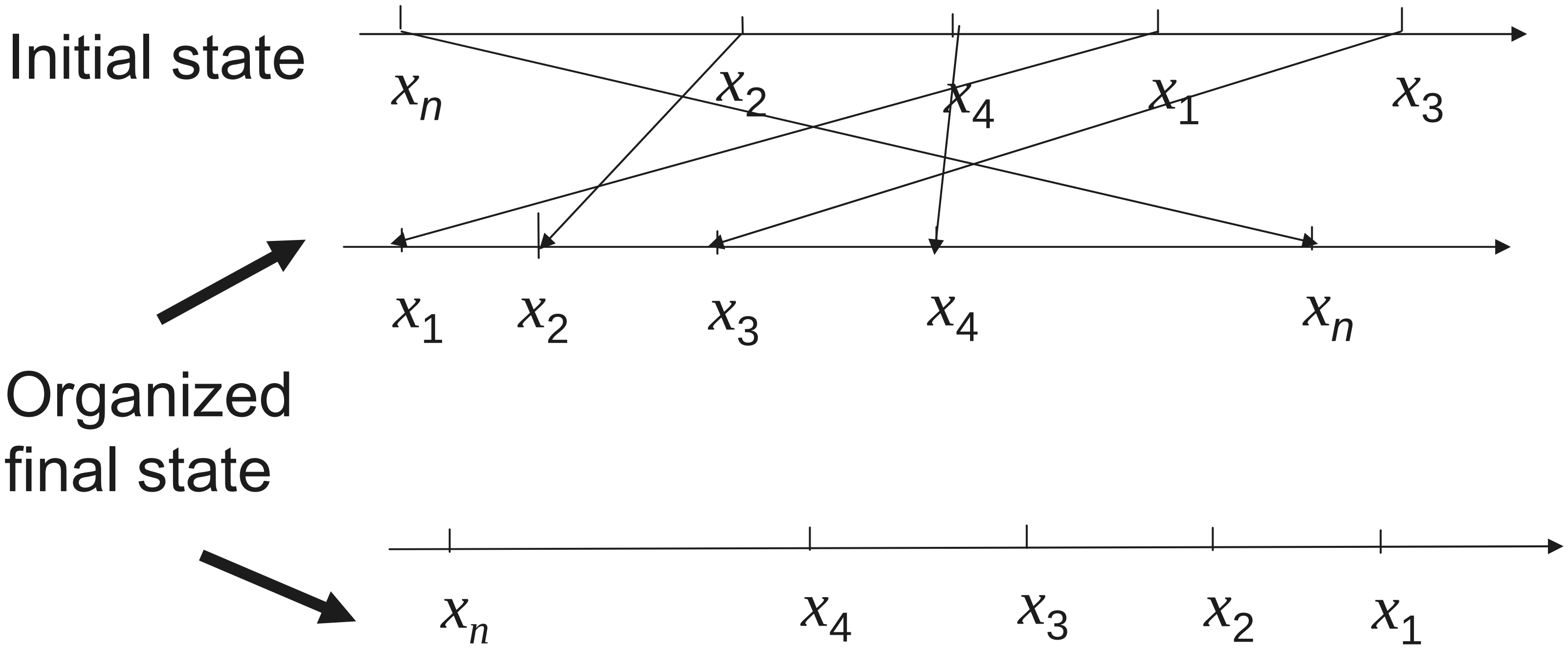
On supprime
un doigt



t=50000



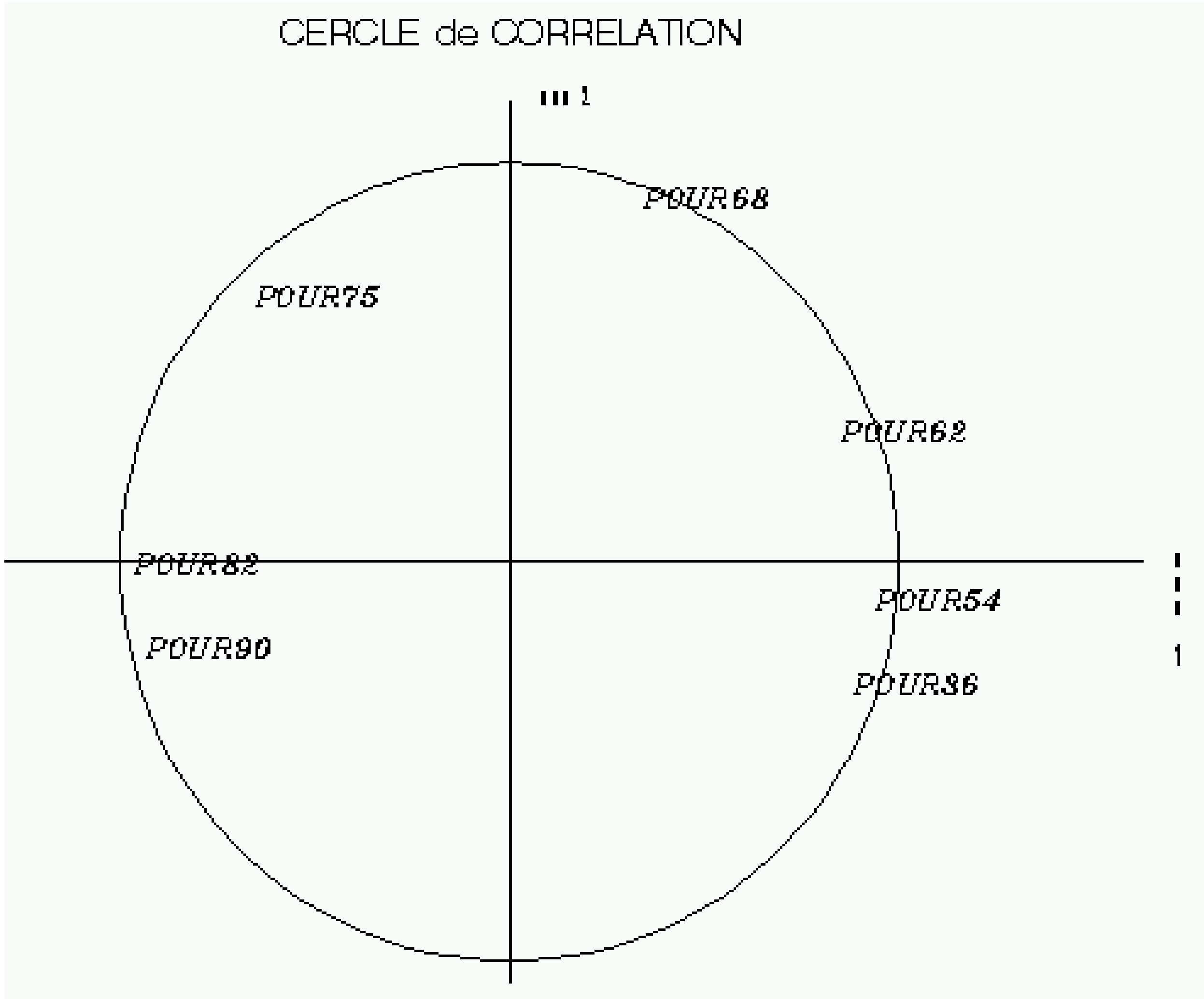
Example : one dimensional data



Decreasing or increasing disposition

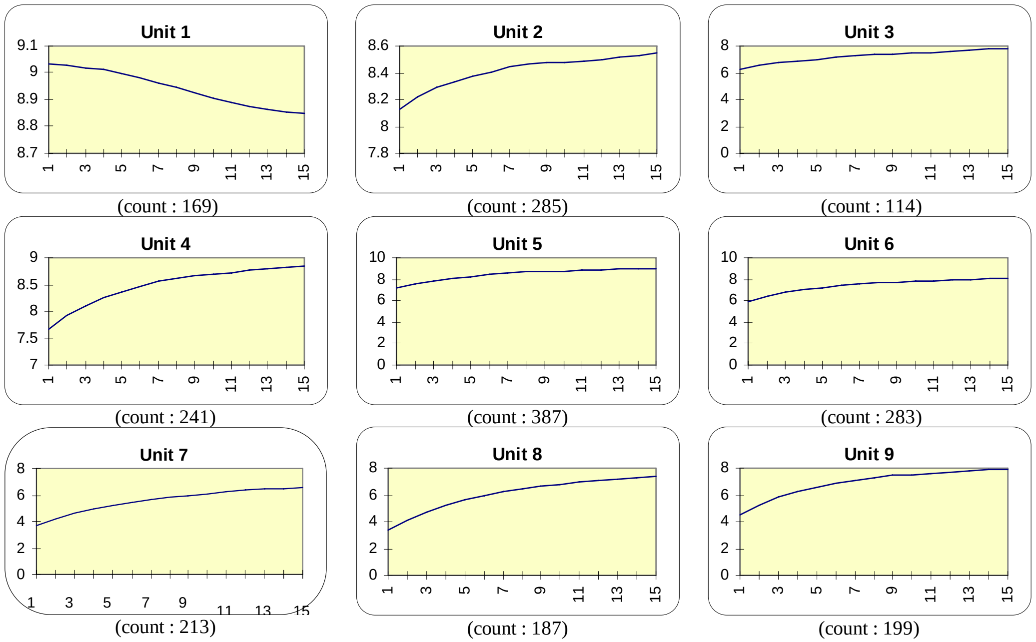
Neighborhoods on a string

Ex 2: PCA on the districts (88% on axes 1 and 2)



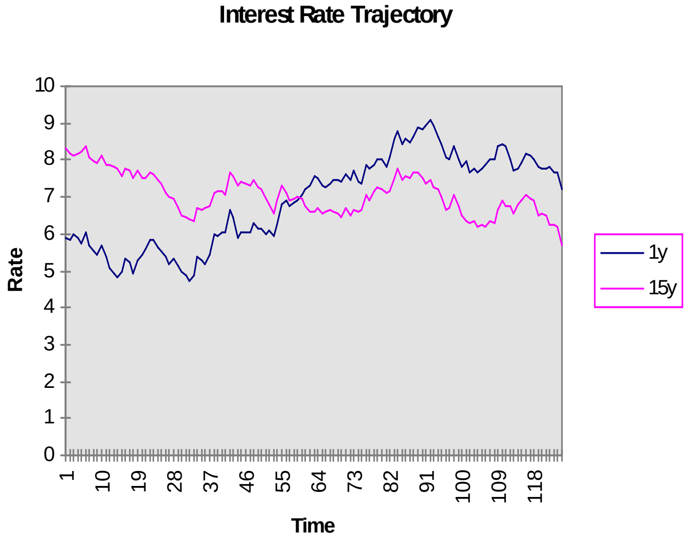
- Données manquantes (Brest)
- Caractères
- Trajectoires Cuba 2008
- Emploi du temps Acseg 2002
- Pollution (IWANN)
- Courbes électriques (IWANN)
- Double quantif pour prédiction IWANN 03

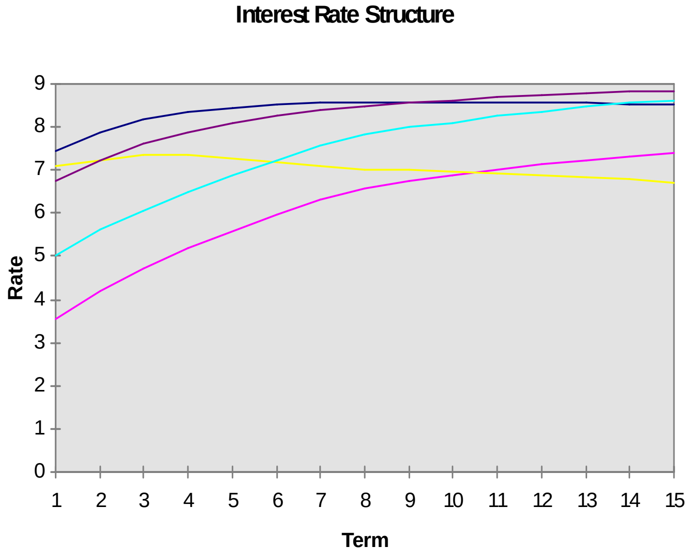
Classification of the initial interest rate structures into 9 classes, in a Kohonen string



- Class one corresponds to a specific shape (8% of the curves)
- The others are discriminated by the level (highly correlated with the ***short term rate level*** at 1 year)
- If we neutralizes the effect of the short term rate level, the code vectors of the classes take another form (in next slide)
- It is clear that the explicative variables for the clustering are the slope (linked to the ***spread***, i.e. the difference between the long term rate and the short term one) and the ***curvature***.

Outputs of the Simulation Procedure





- Forward interest rates are all positives, at each step, for each run.

Clustering the classes

- Advantages of the Kohonen algorithm
  - The similar vectors belong to neighbor classes
  - The typical profile is chosen as representative of the class
  - It is very simple to go to on the computation on new data, starting from the last values of the weights
- To facilitate the interpretation of the classes, the 100 classes are grouped into 13 classes, according to a hierarchical classification
- The limits of the new classes corresponds to the greatest inter-classes distances for the 100-classes classification
- One can observe that there is a significant arrangement on the map : from the top to the bottom, one can encounter successively the weekdays of Autumn and Winter, the weekdays of Spring and Summer, and the Saturdays and Sundays
- These super classes are only used for representation

Distribution of the 182 countries according to their IDH level (from 1 to 6), *small* (1=green, 2=yellow), *medium* (3=blue, 4=violet), *high* (5=gray,6=red)