PATTERNS

Detection, recognition and reconstruction

R. Vilela Mendes

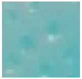
April 2017

Patterns

Pattern: *a discernible regularity in the world*

Pattern recognition is at the basis of our interaction with the world and evolution has provided for very sophisticated ways of pattern recognition. However beware of the dominance of pattern finding and associative recognition mechanisms.

Patterns



- Likewise, pattern detection, recognition and dynamical reconstruction are at the basis of all technological devices and applications.
- *Topics covered in this survey*

  - Statistical versus structural pattern recognition

  - Features and classification

  - Clustering and decision trees

  - Neural networks and ambiguity detection

  - Equivalence of distributed computing systems

  - Neural networks and logic programming systems

  - Deep learning

  - Immune networks and anomaly detection

  - Languages, grammars and control

  - Modelling and stochastic processes

Pattern recognition and classification

- **Statistical pattern recognition:** (data to numbers, *feature extraction*, classification by statistical techniques)
  - *Clustering* to define *classes*
  - *Classification* by *distance to class mean* or *nearest neighbor*
  - *Bayesian decision making*
  - *Decision trees*
  - *Support Vector Machines* (SVM)
  - *Artificial Neural Networks* (ANN), *Deep learning*
- **Structural pattern recognition:** (data to grammars or graphs, recognition by parsing or matching)
  - *Characters, strings, factors, texts*
  - *String Matching, Edit Distance*
  - *Bad-character and good suffix heuristics*
  - *Grammars, terminal symbols, production rules*
  - *Languages: free, context-sensitive, context-free, regular*
  - *Parsing: Bottom-up, Top-down*

Feature extraction and classification

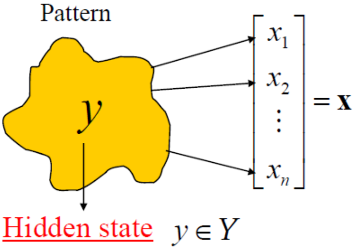
Two essential steps:

1. Feature Extraction - identify characteristics that are very similar for objects in the same category, and very different for objects in different categories.

Invariant features - those that are invariant to irrelevant transformations of the underlying data.

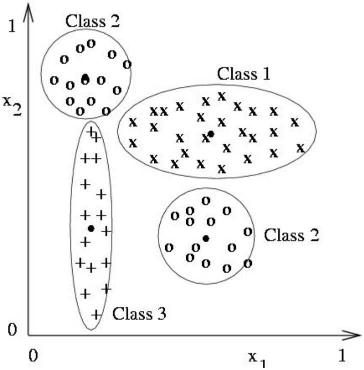
2. Classification - to assign a category to the object based on the feature vector. The perfect *feature extractor* would yield a representation that is trivial to classify.

The perfect *classifier* would yield a perfect model from an arbitrary set of features. Seldom possible.



Statistical pattern recognition (clustering)

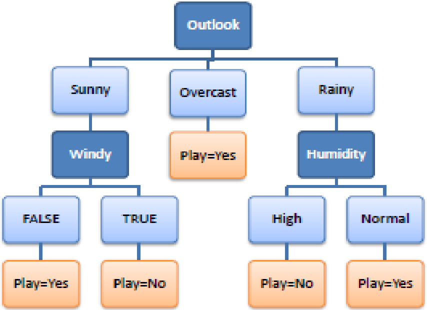
- *Features and feature vectors* (density, hardness, crystal shape, ...) $\rightarrow (x_1, x_2, x_3, \ldots)$
- *Classes in the space of features defined by* **clustering**



http://label2.ist.utl.pt/vilela/Cursos/Spect_Clust_SL.pdf

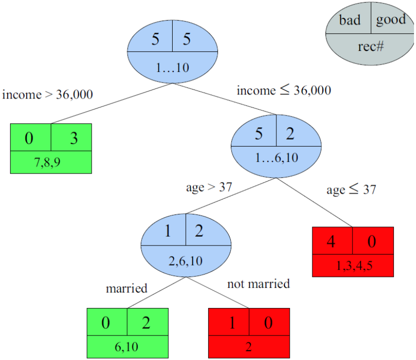
Statistical pattern recognition (distance and decision trees)

- *Classification by distance* (to class mean or k nearest neighbors) Compute the distance between the feature vector X and the mean of each class or to the k nearest neighbors. If distance is below a threshold choose the closest class, otherwise reject.
- *Decision trees:* Graphical representation of a set of decision rules *A decision tree for golf players*



A decision tree for credit scoring

Record	age	married?	own house	income	gender	class
1	22	no	no	28,000	male	bad
2	46	no	yes	32,000	female	bad
3	24	yes	yes	24,000	male	bad
4	25	no	no	27,000	male	bad
5	29	yes	yes	32,000	female	bad
6	45	yes	yes	30,000	female	good
7	63	yes	yes	58,000	male	good
8	36	yes	no	52,000	male	good
9	23	no	yes	40,000	female	good
10	50	yes	yes	28,000	female	good



Neural networks (ambiguity detection)

- R. Ligeiro and RVM; Soft Computing 2017, DOI 10.1007/s00500-017-2525-7

- In physical or social systems, dealing with uncertainty is a critical issue. Uncertainty may arise from **inaccuracies (1)** in the measurement of the variables or if the variables that are measurable do not provide a complete specification of the system (**ambiguity (2)**). Here we were mostly concerned with ambiguous data, with "incomplete" rather than "inaccurate" measurements.
- The ambiguity situation is a complex issue because, in general, the uncertainty is not uniform throughout the parameter space. For example in credit scoring, the NINJA situation (*no income, no job, no asset)* situation is a clear sign of no credit reliability, but most other situations are not so clear-cut.
- The system consists of 2 coupled networks, one to learn the most probable output value and the other to provide the expected error (or variance) of the result for that particular input.

Ambiguity detection by ANN

Learning the average and variance of random functions

- A random function $\theta\left(\overrightarrow{X}\right)$ with distribution $F_{\overrightarrow{X}}(\theta)$. For simplicity take θ to be a scalar and the index set $\left\{\overrightarrow{X}\right\}$ vector-valued, $\overrightarrow{X} \in \mathbb{R}^{i}$. We allow for different distribution functions at different points of the index set.
- The $\overrightarrow{X}$ values are inputs to a multilayer (feedforward) network, $\{W\}$ denoting the full set of connection strengths, the output being $Y\left(\overrightarrow{X}\right) = f_W\left(\overrightarrow{X}\right)$. The aim is to chose a set of connection strengths $\{W\}$ that annihilates the expectation value

$$\mathbb{E}\left\{\sum_{\left\{\overrightarrow{X}\right\}}\left(f_W\left(\overrightarrow{X}\right)-\theta\left(\overrightarrow{X}\right)\right)^2\right\}=0$$

Ambiguity detection by ANN

- However, what the backpropagation algorithm does is to minimize $\mathbb{E}\left(f_W\left(\overrightarrow{X}\right)-\theta\left(\overrightarrow{X}\right)\right)^2$ for each realization of the random variable $\overrightarrow{X}$. Hence, let us fix $\overrightarrow{X}$ and consider $f_W\left(\overrightarrow{X}\right)$ evolving in learning time. The time-evolution of the network output is

$$\begin{aligned} & f_W\left(\overrightarrow{X},\tau+1\right) \\ = \;& f_W\left(\overrightarrow{X},\tau\right)-2\eta\frac{\partial f_W}{\partial W}\bullet\left(f_W\left(\overrightarrow{X}\right)-\theta\left(\overrightarrow{X}\right)\right)\frac{\partial f_W}{\partial W} \quad (1)\end{aligned}$$

where $\Delta W=-\eta\frac{\partial e}{\partial W}$, η being the learning rate and $e=\left(f_W\left(\overrightarrow{X}\right)-\theta\left(\overrightarrow{X}\right)\right)^2$ the error function.

Ambiguity detection by ANN

Let the input random variable $\theta\left(\overrightarrow{X}\right)$ at learning step τ be modeled as

$$\theta\left(\overrightarrow{X}\right)_{\tau} = \overline{\theta}\left(\overrightarrow{X}\right) + B\left(\overrightarrow{X}, \tau\right)$$

$\overline{\theta}\left(\overrightarrow{X}\right)$ being the expectation (average) value of the variable and $B\left(X, \tau\right)$ a zero-mean Wiener process. Then taking expectation values in Eq.(1), and because the $\frac{\partial f_W}{\partial W}$ quantities are deterministic functions, one obtains

$$\begin{aligned} & \mathbb{E}\left[f_W\left(\overrightarrow{X}, \tau+1\right)\right] \\ = \; & \mathbb{E}\left[f_W\left(\overrightarrow{X}, \tau\right)\right] - 2\eta\frac{\partial f_W}{\partial W} \bullet \left(\mathbb{E}\left[f_W\left(\overrightarrow{X}, \tau\right)\right] - \overline{\theta}\left(\overrightarrow{x}\right)\right)\frac{\partial f_W}{\partial W} \end{aligned}$$

Ambiguity detection by ANN

- Notice that we are now dealing with two time scales: the time scale of the $\theta\left(\overrightarrow{X}\right)$ random variable and the time scale of the leaning process. If the learning rate η is sufficiently smaller than the sampling rate of the $\theta\left(\overrightarrow{X}\right)$ random variable, the last equality may be approximated by

$$\begin{aligned} & f_W\left(\overrightarrow{X},\tau+1\right) \\ = \; & f_W\left(\overrightarrow{X},\tau\right)-2\eta\frac{\partial f}{\partial W}\bullet\left(f_W\left(\overrightarrow{X},\tau\right)-\bar{\theta}\left(\overrightarrow{X}\right)\right)\frac{\partial f_W}{\partial W}\end{aligned}$$

A fixed point is obtained at

$$f_W\left(\overrightarrow{X}\right)=\bar{\theta}\left(\overrightarrow{X}\right),$$

the average value of the random variable θ at the argument $\overrightarrow{X}$.

Ambiguity detection by ANN

Similarly if a second network (with output $g_{W'}\left(\overrightarrow{X}\right)$) and the same input $\overrightarrow{X}$ is constructed according to the law and error function

$$g_{W'}\left(\overrightarrow{X},\tau'+1\right)=g_{W'}\left(\overrightarrow{X},\tau'\right)+\frac{\partial g_{W'}}{\partial W'}\bullet\Delta W'$$

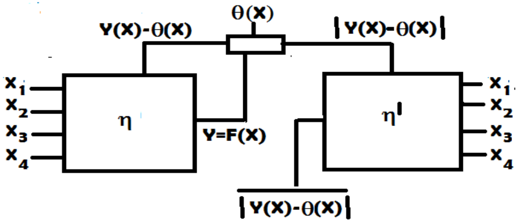
$$e'=\left(g_{W'}\left(\overrightarrow{X}\right)-\left(f_W\left(\overrightarrow{X}\right)-\theta\left(\overrightarrow{X}\right)\right)^2\right)^2$$

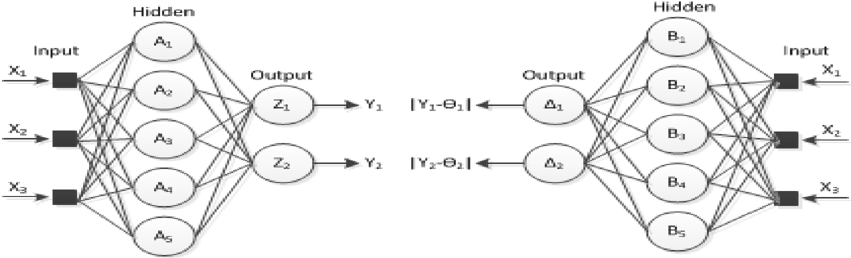
and $\Delta W'=-\eta'\frac{\partial e'}{\partial W'}$, then, under the same assumptions as before, $g_{W'}\left(\overrightarrow{X}\right)$ has the fixed point

$$g_{W'}\left(\overrightarrow{X}\right)=\overline{\left(\theta\left(\overrightarrow{X}\right)-\bar{\theta}\left(\overrightarrow{X}\right)\right)^2}$$

In conclusion: the first network reproduces the average value of the random function θ for each input $\overrightarrow{X}$ and the second one the variance.

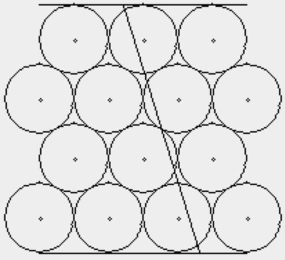
Ambiguity detection by ANN

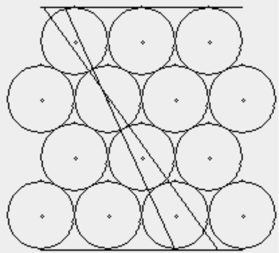




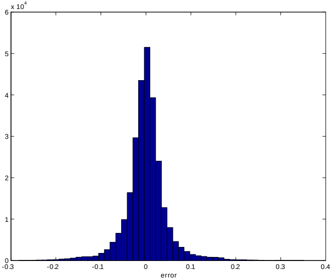
Ambiguity detection by ANN

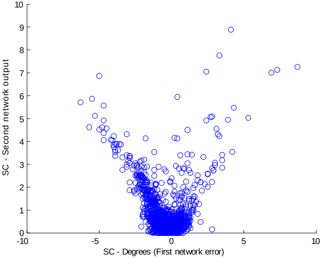
Example 1: Measuring track angles by straw chambers

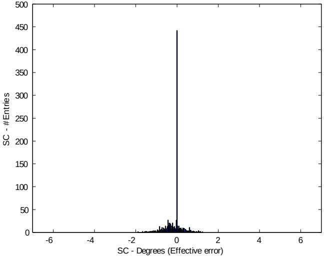




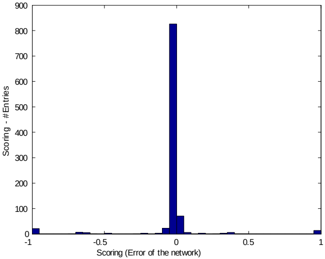
Measuring track angles by straw chambers

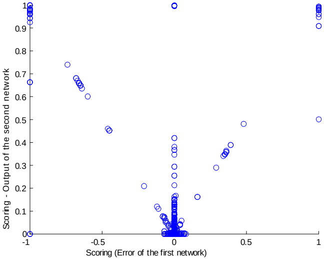


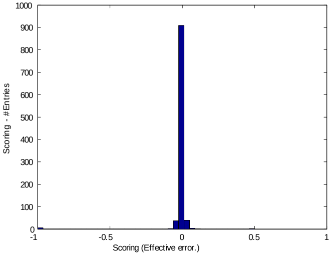




A credit scoring model



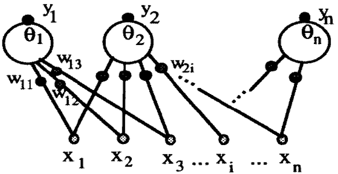




Equivalence of connectionist (distributed computing) systems

- *Neural networks, Classifier Systems, Immune networks, Autocatalytic chemical reaction networks*, J. Doyne Farmer; *A Rosetta stone for connectionism*, Physica D 42 (1990) 153-187
  (Requires node parameters in the equivalent NN because in some of the other systems the W_{ij}'s are fixed chemical affinities)
- It turns out that the addition of a node parameter with learning capabilities may improve the performance of NN's.
  J. A. Dente and RVM; *Unsupervised learning in general connectionist systems*, Network: Computation in Neural Systems 7 (1996) 123–139

$$y_i = \theta_i \sum w_{ij} x_j$$



NN's and logic programming languages

- Furthermore, the equivalence of the computational power of all these distributed computation systems was proved to be as wide as a large class of logic programming languages.
- This is carried out in the framework of a Horn-clause logic programming language, Prolog.
- Why?

  - Rules are easier to implement in logic programming than on neural networks.

  - Conversely neural network implementations are usually faster.

  - Also useful for the design of multicomputer networks
- J. Martins and RVM; *Neural networks and logical reasoning systems: A translational table*, International Journal of Neural Systems 11 (2001) 179–186.

A three-minute course on Prolog

- Prolog is not a **procedural language**, it is a **deductive** language. Computation of a Prolog program is a logical consequence of the axioms or facts in a data base.
- Prolog is a frequently used language in Artificial Intelligence where **manipulation of symbols** and **inference** about them is a common task.
- Prolog consists of a series of **rules** and **facts**. A program is run by presenting some **query** and seeing if this can be proved against the known rules and facts.
- **Atoms:**
  fred, oranges, a, b, human, 'Professor', +, ...
- **Variables:**
  X, Yatch, Z, Human, _range, ...
- **Complex terms:**
  functor (argument1, ..., argument*n*).
- **Lists :** [Head|Tail] ; [a,b,c] Head=a and Tail=[b,c]
  on(Item,[Item|Rest]).

A three-minute course on Prolog

- **Facts:** (defined by functors)

  ```
  eats(fred,oranges).
  eats(fred,grapefruit).
  eats(mary,apples).
  eats(john,apples).
  parent(fred,mary).
  parent(john,fred).
  human(socrates).
  ```

- **Rules** (in Prolog :- means $\Longleftarrow$)

  ```
  mortal(X) :- human(X).
  ancestor(X,Y) :- parent(X,Y).
  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).          (recursion)
  ```

- **Querries :**

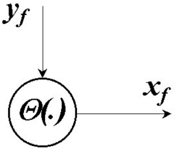
  ```
  ?- eats(fred,oranges).
  yes
  ```

A three-minute course on Prolog

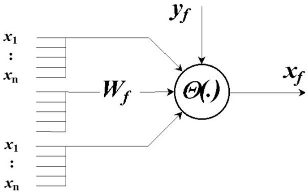
-
  ```
  ?- eats(mary,apples).
  no
  ?- eats(fred,X).
  X=oranges
  X=grapefruit
  ?- eats(X,apples).
  X=mary
  X=john
  ?- mortal(socrates).
  yes
  ?- ancestor(john,mary).
  yes
  ```
- Nowadays Prolog has been superseded (completed) by *Constraint Logic Programming*.

NN's and logic programming languages

- Constants = nodes without inputs, except polarization



- Functors = nodes with nth-order synapses



$$x_f = \theta\left(\sum_{i_1\cdots i_n} W_{f,i_1,\cdots,i_n} x_1 \cdots x_n + y_f\right)$$

NN's and logic programming languages

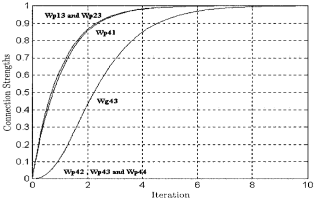
$$
\begin{aligned}
&f(b,c)\,.\\
&m(a,c)\,.\\
&m(d,a)\,.\\
&p(X,Y) \Leftarrow m(X,Y)\,.\\
&p(X,Y) \Leftarrow f(X,Y)\,.\\
&g(X,Z) \Leftarrow p(X,Y) \wedge p(Y,Z)\,.
\end{aligned}
$$

$$W_{fbc} = 1; \quad W_{mac} = 1; \quad W_{mda} = 1$$

$$
\begin{aligned}
W_{pXY} &\geq W_{mXY}\\
W_{pXY} &\geq W_{fXY}\\
W_{gXZ} &\geq W_{pXY} W_{pYZ}
\end{aligned}
$$

$$
\begin{aligned}
V_1 = &\frac{1}{2}\sum_{XY} \chi(W_{mXY} - W_{pXY})\\
&+ \frac{1}{2}\sum_{XY} \chi(W_{fXY} - W_{pXY})\\
&+ \frac{1}{2}\sum_{XYZ} \chi(W_{pXY}W_{pYZ} - W_{gXZ})
\end{aligned}
$$

$\chi = x^2\theta\,(x)\,;\ \frac{dW_{pXY}}{d\tau} = -\frac{\partial V_1}{\partial W_{pXY}}$ (same for W_{gXY})



NN's and logic programming languages

In general: for each rule of the form

$$g\left(X_1,\cdots,X_n\right)\Longleftarrow\prod_i\wedge f_i\left(Y_1,\cdots,Y_{p_i}\right)$$

a term

$$\frac{1}{2}\sum_{\{X\}\{Y\}}\chi\left(\prod_i W_{f,Y_1,\cdots,Y_{p_i}}-W_{g,X_1,\cdots,X_n}\right)$$

is added to the potential and the connection strengths are obtained by gradient evolution. The synapse dynamics is a *forward chaining step* and after the evolution the network contains all the available information about the problem.

Information may be retrieved by simply looking at the connection strengths or, alternatively by a query scheme:

$$?\Longleftarrow g\left(a,b\right)$$

Excite the polarizations y_a and y_b and see whether the node g lights up.

NN's and logic programming languages

$$? \Longleftarrow g\left(h\left(a,b\right),c\right) \wedge f\left(c,d\right)$$

Excite the polarizations y_a, y_b, y_c and y_d and the answer is positive if the nodes g and f light up simultaneously.
Queries with variables:

$$\begin{aligned} ? &\Longleftarrow m\left(d,X\right) \wedge p\left(X,c\right) \\ ? &\Longleftarrow g\left(h\left(X,c\right),a\right) \wedge f\left(c,Y\right) \end{aligned}$$

In the first case the non-zero entries of the vector

$$\alpha_X = W_{mdX} W_{pXc}$$

are the instantiations that satisfy the query and, in the second, they are the non-zero entries of the tensor

$$\beta_{XY} = W_{gha} W_{hXc} W_{fcY}$$

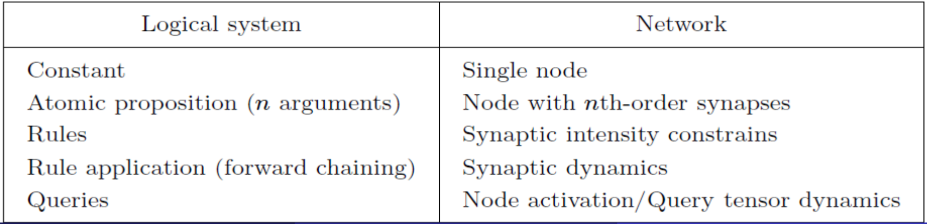
Alternatively looking at the connection strengths as potential knowledge, the non-zero entries of the tensors may be generated dynamically

NN's and logic programming languages

$$\frac{d\alpha_X}{d\tau} = -\frac{\partial V_2}{\partial \alpha_X}; \qquad \frac{d\beta_{XY}}{d\tau} = -\frac{\partial V_3}{\partial \beta_{XY}}$$

$$\begin{aligned} V_2 &= \sum_X \left(W_{mdX} W_{pXc} - \alpha_X \right)^2 \\ V_3 &= \sum_{XY} \left(W_{gha} W_{hXc} W_{fcY} - \beta_{XY} \right)^2 \end{aligned}$$

The dynamics of the query tensors plays the role of *backward chaining* in logic programming.



Logical system	Network
Constant	Single node
Atomic proposition (n arguments)	Node with nth-order synapses
Rules	Synaptic intensity constrains
Rule application (forward chaining)	Synaptic dynamics
Queries	Node activation/Query tensor dynamics

A note on deep learning

- *Representation learning* allows a machine to be fed with raw data and to automatically discover the representations needed for detection or classification.
- *Deep-learning* is representation-learning with multiple levels of representation. The representation at one level (starting with raw data) is transformed into a representation at a higher, more abstract level. The idea is that higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations.
- The key aspect of deep learning is that these layers of features are not designed by human engineers, they are learned from data using an automatic procedure.
- Several configurations: convolution networks, deep belief networks, etc
- *Project proposal: do deep learning using logic programming languages*

Immune system

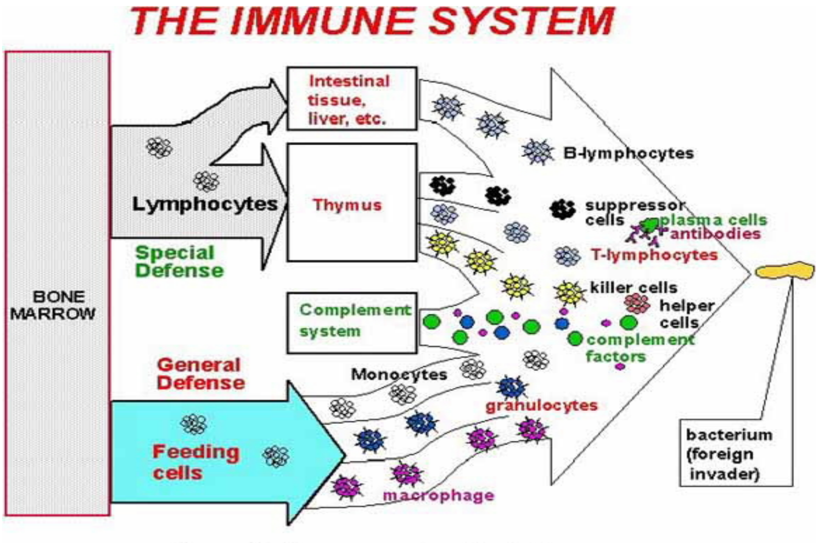
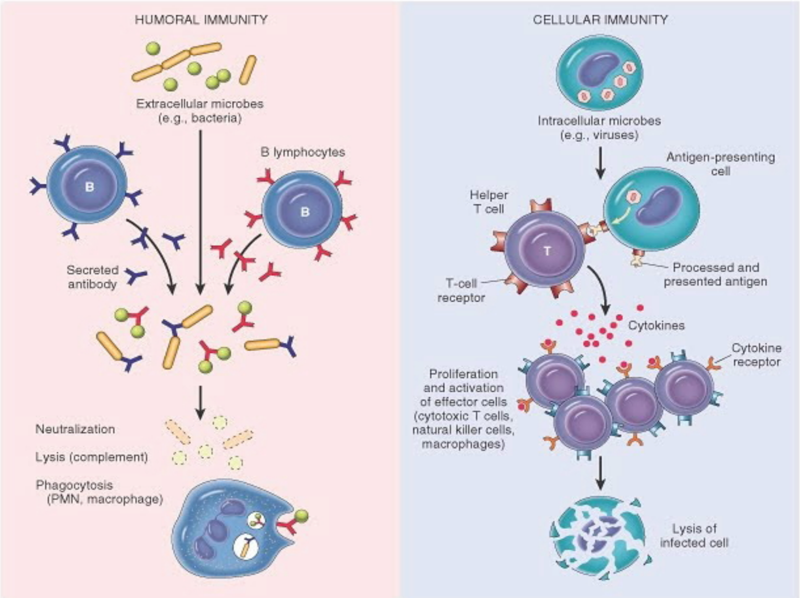
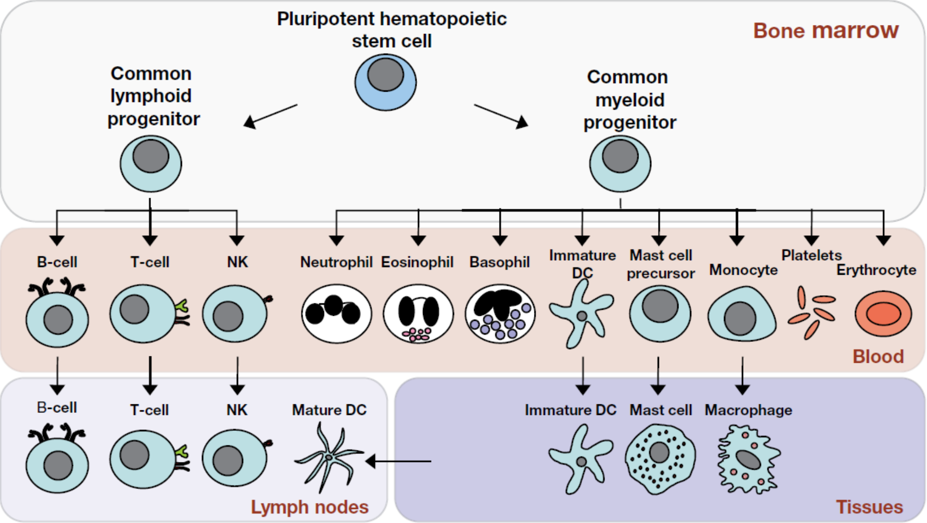


Figure 20. Immune system block diagram.

Immune system



Immune system



Immune-like networks as anomaly detection devices

- The mammal immune system is an information processing system of complexity comparable (or even larger) to the brain. In the same way as the brain is the main inspiration for artificial neural networks, the immune system provides inspiration for technological information processing systems. Remember also the equivalence result.
- - Distinguishing self and non-self
  - Uniqueness
  - Imperfect detection
  - Mutation
  - Learning and memory
  - Novelty detection
- These features make immune-like systems a tool of choice for anomaly detection. It has been used to detect anomalies in electromechanical systems, production systems, computer networks, etc.
- An example: P. C. Branco, J. A. Dente and RVM; *Using Immunology Principles for Fault Detection*, IEEE Transactions on Industrial Electronics 50 (2003) 362-373

Structural pattern recognition (Linguistic approach)

Formal language concepts

- To apply *grammatical inference* procedures, a dynamical system must be considered as an entity (linguistic source) capable of generating a specific language. The grammar G of the language is the set of rules that specifies all the words in the language and their relationships. Once the grammar is found, the grammar itself is a model for the source.
- To define a grammar G, one specifies a *terminal alphabet* Σ_T, a *nonterminal alphabet* Σ_N, a *start symbol* S, and a set of *productions* P

$$G = \{\Sigma_T, \Sigma_N, S, P\}$$

- *Terminal Alphabet* T: Set of symbols that make up the *words*, a word being a string of symbols.
- *Nonterminal Alphabet* N: Set of auxiliary symbols that are used to generate the words by the production rules.

Linguistic approach

- *Start Symbol S*: A special nonterminal symbol used to start the generation of words.
- *Productions P*: Set of substitution rules (denoted $a \rightarrow b$) used to generate the allowed words in the language.
- Example: The grammar

  $$G = \{\Sigma_T = (a; b); \Sigma_N = (A); P = (S \rightarrow bA; A \rightarrow aA; A \rightarrow a)\}$$

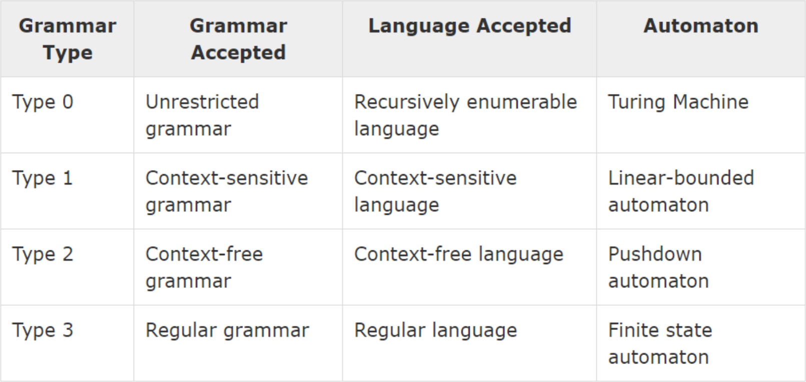
  generates the language with words consisting of one b symbol followed by any number of a symbols.
- The set of productions encodes the dynamics of the system that generates the language.
- *Grammatical inference* is an algorithm by which a grammar is inferred from a set of sample words produced by the system considered as the linguistic source. Notice that there is not a unique relation between a language and the grammar because distinct grammars may generate the same language.

Linguistic approach

- Grammatical inference is the identification of a grammar from a set of positive and negative examples. One may also consider a “teacher” that answers questions concerning the language to be inferred.
- In dynamical system identification by grammatical inference, we are in the restricted setting of language identification from positive examples alone. Therefore, a finite sample cannot characterize an infinite language. It may at most specify the class of languages, which possesses that finite sample as an allowed one.
- A practical problem, when learning from a finite set of examples, is that some relevant productions may never be inferred. If a metric is defined in the space of words, this problem may be solved to some extent by *grammatical interpolation techniques*.
- Another practical problem arises when the system is perturbed by noise. Then, productions may be inferred which are not characteristic of the system but are a result of the noise perturbation. A possible practical solution is to keep only those productions that appear many times in the learning process.

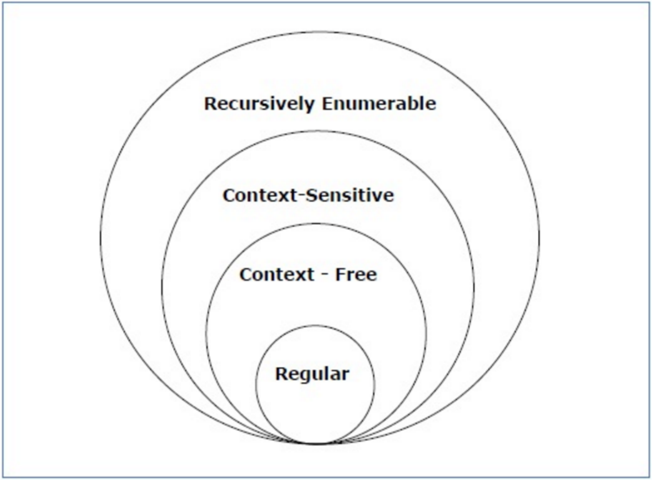
Linguistic approach

Language classification (Chomsky)



Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

Linguistic approach



Linguistic approach

- **Regular languages**: must have a single non-terminal on the left-hand side and a right-hand side consisting of a single terminal or single terminal followed by a single non-terminal. Productions must be of the form $X \rightarrow a$ or $X \rightarrow aY$
- **Context-free languages**: Productions must be in the form $A \rightarrow \gamma$, with A nonterminal and $\gamma \in (T \cup N)*$ (String of terminals and non-terminals).
- **Context-sensitive languages**: Productions must be in the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ with A non-terminal and $\alpha, \beta, \gamma \in (T \cup N)*$ (Strings of terminals and non-terminals).
- **Recursively enumerable languages**: Productions have no restrictions.

Linguistic approach

- An example: *Grammatical Inference in Controlled Dynamical Systems* J. F. Martins, J. A. Dente, A. J. Pires, and RVM; *IEEE Transactions on systems, man and cybernetics 31 (2001) 234-242.*
- A model for a controlled dynamical system has the general form

$$\begin{aligned} x_{t+1} &= f\left(x_t, U_t\right) \\ y_t &= h\left(x_t\right) \end{aligned}$$

$x =$ state variable
$y =$ output (or observed) variable;
$U =$ input (or control) variable.

- By the equation above a functional relationship is established between

the output variables at different times

$$y_{t+1} = g\left(y_k, U_t\right)$$

In most systems not all state variables are observable. Therefore this equation does not provide a complete specification of the system.

Grammatical inference in controlled dynamical systems

- In general, specification of the dynamics in terms of the output variables requires a set of functional relationships involving many time steps in the past, for example

  $$y_{t+1} = g_3\left(y_t, y_{t-1}, y_{t-2}, U_t\right)$$

  or

  $$y_{t+1} = g_2\left(y_t, y_{t-1}, U_t\right)$$

  depending on the value of y_t.
- *Quantification* is the creation of alphabets for the output variable y and the control variable U. Associate the terminal alphabet $\sum_T$ to the output variable y and the nonterminal alphabet $\sum_N$ to the control variable U.
- *P-type productions* (λ is the empty symbol)

  $$\begin{aligned} y_1 \cdots y_p U_k &\rightarrow y_1 \cdots y_p y_{p+1} \delta \\ \delta &\rightarrow U_j \\ \delta &\rightarrow \lambda \end{aligned}$$

Grammatical inference in controlled dynamical systems

- *Learning a language from data:* a sequence of control signals is applied to the system. Then:

  1 - A *0-type production* is assumed for every newly occurring control symbol.

  2 - A new *(n + 1)-type production* is generated each time the data conflicts with the previously established *n-type productions*. The conflicting *n-type productions* are promoted to *(n+1)-type productions* or are deleted if there is not sufficient information in the past to do so.
- **An example**: Consider the following symbol sequences

U variable	A	B	A	A	B	A
y variable	e	d	c	b	d	e

$t = 1 \Longrightarrow A \rightarrow e\delta$

$t = 2 \Longrightarrow B \rightarrow d\delta$

$t = 3 \Longrightarrow dA \rightarrow dc\delta$ and the production $A \rightarrow e\delta$ is deleted because no previous information is available on the symbol e

Grammatical inference in controlled dynamical systems

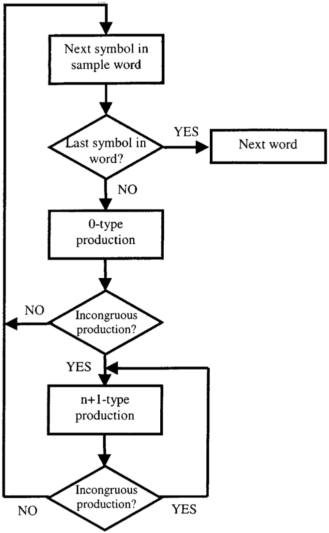
$t = 4 \Longrightarrow cA \rightarrow cb\delta$
$t = 5 \Longrightarrow B \rightarrow d\delta$ which had already been found
$t = 6 \Longrightarrow bdA \rightarrow bde\delta$ and $dA \rightarrow dc\delta$ is revised to $edA \rightarrow edc\delta$

- Finally we are left with

$$\begin{array}{rcl} B & \rightarrow & d\delta \\ cA & \rightarrow & cb\delta \\ bdA & \rightarrow & bde\delta \\ edA & \rightarrow & edc\delta \end{array}$$

- Because of the equivalence of language to their recognizing automata, this process may be automatically implemented by the construction of the automata. See:
"*From controlled dynamical systems to context-dependent grammars: A connectionist approach*", Eng. Appl. of Artificial Intelligence 22 (2009) 192–200.

Flow chart for grammar inference



Modelling and stochastic processes

- The theory of stochastic processes is a powerful tool to model the evolution (dynamical patterns) of physical and social systems.
  http://label2.ist.utl.pt/vilela/Cursos/Proba.pdf
  http://label2.ist.utl.pt/vilela/Cursos/Brownian.pdf
  http://label2.ist.utl.pt/vilela/Cursos/Beyond.pdf
- Of special importance are processes which are not Brownian, because either have long-range dependence (fractional) or fat tails (Lévy, etc.).
- **Some keys facts for modelling:**

  - *Brownian motion*:
  $\mathbb{E}[B(t)] = 0$; $\mathbb{E}[B(t)B(s)] = \min(t,s) = \frac{1}{2}\{t+s-|t-s|\}$
  - *White noise*: $\mathbb{E}[W(t)\,W(s)] = \delta(t-s)$; $B(t) = \int_0^t W(\tau)\,d\tau$
  - *Martingale*: $\mathbb{E}[F(t)\,|F(s)] = F(s)$ if $t > s$
  $B(t)$, $B(t)^2 - t$ and $B(t)^3 - 3\int_0^t B(\tau)\,d\tau$ are martingales
  - *Ito formula*:
  $df(B(t),t) = \partial_1 f(B(t),t)dB(t) + \frac{1}{2}\partial_1^2 f(B(t),t)dt + \partial_2 f(B(t),t)dt$

Modelling and stochastic processes

- - *Feynman-Kac formula*: $f(x,t) = \mathbb{E}_{x,t}\left[e^{\int_t^T V(B(s))ds}\right]$ satisfies the equation $\partial_t f + \frac{1}{2}\partial_x^2 f + V(x)f = 0$ with the final condition $f(x,T) = 0$

  - *Geometrical Brownian motion*: $dX(t) = \mu X(t)dt + \sigma X(t)dB(t)$; $X(t) = e^{\mu t - \sigma t^2/2 + \sigma B(t)}$

  - *Self-similar processes*: $X(at) \overset{d}{=} a^H X(t)$; $H =$ Hurst exponent, Brownian motion is self-similar and has stationary increments

  - *Lamperti transformation*: If $Y(t)$ is stationary $X(t) = t^H Y(\log t)$ is $H-$self-similar, conversely if $X(t)$ is $H-$self-similar $Y(t) = e^{-tH} X(e^t)$ is stationary.

  - *Fractional Brownian motion*: processes with stationary increments and unit variance have covariance

$$\mathbb{E}[B_H(t)B_H(s)] = \frac{1}{2}\left\{t^{2H} + s^{2H} - |t-s|^{2H}\right\}$$

B_H is self-similar with Hurst coefficient H and $B_{1/2} = B$.

Modelling and stochastic processes

- Brownian motion has independent increments, but for $H \neq \frac{1}{2}$ there is long-range dependence
  $H > \frac{1}{2}$, positive correlation, a persistent process
  $H < \frac{1}{2}$, negative correlation, an anti-persistent process
  - *Fractional Gaussian noise*: $Y_t = B_H(t+1) - B_H(t)$
  - *Central limit theorem*: Let $X_1, X_2, \cdots, X_n$ be independent identically distributed random variables with mean μ and finite variance $\sigma < \infty$. Then for $S_n = X_1 + X_2 + \cdots + X_n$

$$P\left(\frac{S_n - n\mu}{\sqrt{n}} \leq x\right) \underset{n\to\infty}{\xrightarrow{\mathcal{L}}} \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{\sigma^2}}$$

However without the condition $\sigma < \infty$,

Modelling and stochastic processes

- - *Lévy theorem:* $X_1, X_2, \cdots, X_n$ independent identically distributed random variables. If there are constants a_n and b_n such that

$$P\left(\frac{X_1 + X_2 + \cdots + X_n - a_n}{b_n} \leq x\right) \underset{n\to\infty}{\overset{\mathcal{L}}{\rightarrow}} F(x)$$

Then $F(x)$ is stable, meaning that if X_1, X_2 have the same law, then also

$$X_3 = \frac{1}{\alpha}\left(a_1 X_1 + a_2 X_2 - b\right)$$

For the characteristic function it means

$$\phi(a_1\lambda)\,\phi(a_2\lambda) = e^{i\lambda b}\phi(\alpha\lambda)$$

- - The characteristic function $\phi(\lambda)$ of symmetric stable distribution is

$$\phi(\lambda) = e^{-c|\lambda|^{\alpha}}$$

with $\alpha \in (0, 2]$. The $\alpha = 2$ case is Gaussian. Therefore all others correspond to infinite variance.

Modelling and stochastic processes

- - *Infinitely divisible random variables:* for all n

  $$X = X^{(1/n)} + X^{(1/n)} + \cdots + X^{(1/n)}$$

  n terms. Or

  $$\phi_X(\lambda) = \left(\phi_{X^{(1/n)}}(\lambda)\right)^n$$

  Infinitely divisible random variables (Lévy-Khintchine)

  $$\phi_X(\lambda) = E\left[e^{i\lambda X}\right] = \exp\left\{ib\lambda - \frac{\lambda^2 c}{2} + \int_R \left(e^{i\lambda x} - 1 - i\lambda x \mathbf{1}_{|x|<1}\right)\nu(dx)\right\}$$

  b - drift, c - diffusion, $\nu(dx)$ - jump measure

- An example: **Modelling long-range dependence in population dynamics**

Long range dependence in population dynamics

H. C. Mendes, A. Murta and RVM; *Long range dependence and the dynamics of exploited fish populations*, Advances in Complex Systems 18 (2015) 1550017.

- If $X(t)$ is a random variable square-integrable in the measure generated by Brownian motion, then

  $$dX(t) = \mu(t)\,dt + \sigma(t)\,dB(t)$$

  where $\mu(t)$ and $\sigma(t)$ are well defined stochastic processes. Therefore a process that looks like Brownian motion for short times may, for example, have long dependence on the dynamics of the fluctuations $\sigma(t)$.

The dynamics of exploited fish populations

- A few years ago Niwa, studying 27 commercial fish stocks in the North Atlantic, concluded that the variability in the population growth (annual changes in the logarithm of population abundance $S(t)$)

  $$r(t) = \ln\left(\frac{S(t+1)}{S(t)}\right)$$

  is described by a Gaussian distribution.
- The population variability would be a geometric random walk

  $$r(t) = \frac{dS(t)}{S(t)} = \sigma_r dB(t)$$

  The independence of the increments of Brownian motion implying that $r(t)$ is a purely random process.
- A sobering conclusion. Natural processes that look purely random, are processes that depend on some many uncontrollable variables that any attempt to handle them is outside our reach. This would be a serious blow to, for example, the implementation of sustainability measures.

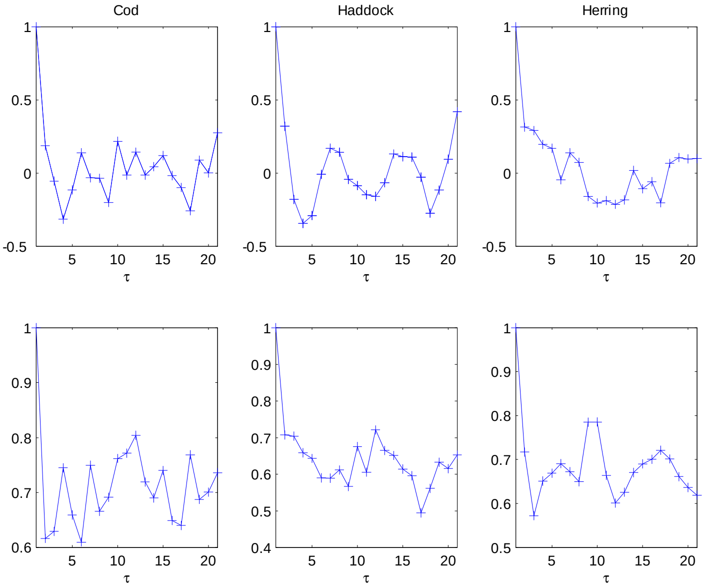
The dynamics of exploited fish populations

- Reanalyze some of the same type of data: Spawning-stock biomass (SSB) for commercial fish stocks in the North Atlantic. The SSB time-series data is derived from age-based analytical assessments estimated by the 2013 working groups of the International Council for the Exploration of the Sea (ICES), based on the compilation of data from sampling of fisheries (e.g. commercial catch-at-age) and from scientific research surveys.
- Select three North Atlantic stocks for which the annual time-series of SSB covers at least 60 years: Northeast Arctic cod (Gadus morhua), Arctic haddock (Melanogrammus aeglefinus) and the North Sea autumn-spawning herring (Clupea harengus).
- Autocorrelation functions for $r(t)$ and $|r(t)|$

$$C(r,\tau) = \frac{\mathbb{E}\{r(t)\, r(t+\tau)\}}{\sigma^2}$$

$$C(|r|,\tau) = \frac{\mathbb{E}\{|r(t)|\, |r(t+\tau)|\}}{\sigma^2}$$

The dynamics of exploited fish populations



The dynamics of exploited fish populations

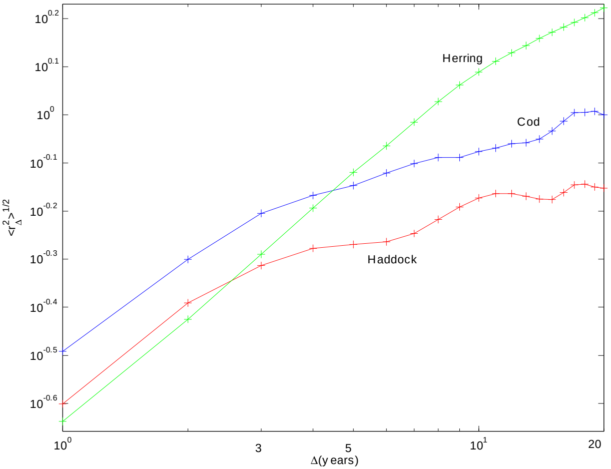
- Already for time lags of one year, autocorrelations are at noise level, suggestive of uncorrelated processes.
- However, if $S(t)$ is indeed a geometrical Brownian motion, scaling properties of $r(t)$ should be checked

$$r_{\Delta}(t) = \ln\left\{\frac{S(t+\Delta)}{S(t)}\right\} = \sum_{i=1}^{\Delta} r(t+i)$$

- The geometrical Brownian motion hypothesis would imply

$$\left(\mathbb{E}\left\{r_{\Delta}^{2}\right\}\right)^{1/2} \backsim \Delta^{1/2}$$

The dynamics of exploited fish populations

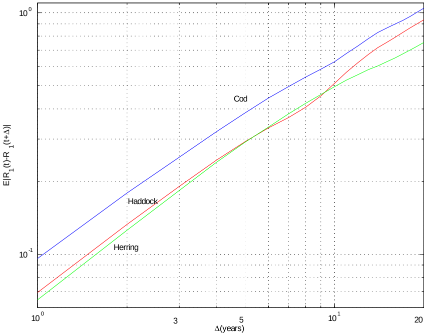


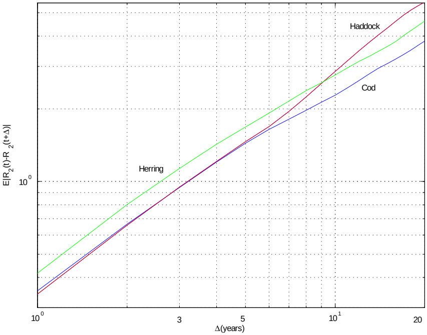
The dynamics of exploited fish populations

- At the species level the geometrical Brownian motion is not a good hypothesis. Even for Herring, where the data seems to follow a scaling law, the slope at large Δ is closer to 0.7 than to 0.5.
- Whatever is actually determining the stochastic process for each species is somehow washed out when averaging over all the 27 species as Niwa did. No surprise, recall the stochastic analysis theorem mentioned before.
- Reconstruct the dynamics of $\sigma(t)$ from the data: Compute the local value of $\sigma(t)$ by the standard deviation of $r(t)$. (6−years window). The cumulative processes and scaling properties of R_1 and R_2

$$\begin{aligned} \sum_{i=1}^{t} \sigma(i) &= \beta_1 t + R_1(t) \\ \sum_{i=1}^{t} \ln \sigma(i) &= \beta_2 t + R_2(t) \end{aligned}$$

The dynamics of exploited fish populations





The dynamics of exploited fish populations

- R_1 and R_2 obey an approximate scaling law with exponents H in the range $0.8 - 0.9$. Hence R_1 and R_2 may be modelled by fractional Brownian motion implying that the fluctuations of σ and $\ln \sigma$, away from an average value, are modeled by Gaussian fractional noise.
- Alternative models for the population fluctuations

$$\begin{aligned} dS(t) &= \sigma(t) S(t) dB_t \\ \sigma(t) &= \beta_1 + \alpha_1 \left(B_{H_1}(t) - B_{H_1}(t-1) \right) \end{aligned}$$

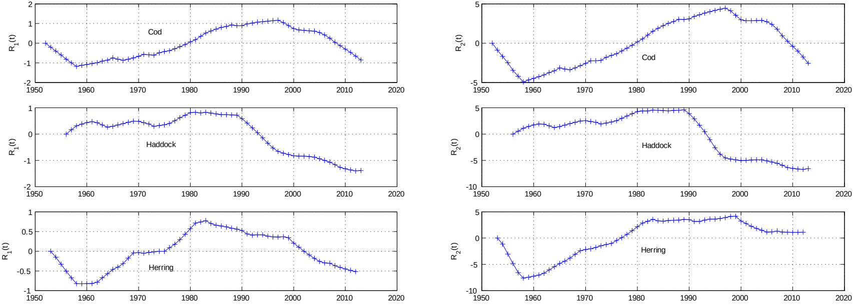
$$\begin{aligned} dS(t) &= \sigma(t) S(t) dB_t \\ \ln \sigma(t) &= \beta_2 + \alpha_2 \left(B_{H_2}(t) - B_{H_2}(t-1) \right) \end{aligned}$$

with the following values for the Hurst coefficients H_1 and H_2

	H_1	H_2
Cod	0.86	0.87
Haddock	0.89	0.9
Herring	0.93	0.87

The dynamics of exploited fish populations

- The dynamics of the fluctuations is a species-dependent long range memory process.
- The cumulative amplitude fluctuations R_1 R_2



Some additional references

- I. N. Silva et. al; "*Artificial Neural Networks: A practical course*", Springer 2017.
- P.-Y. Yin (Ed.); "*Pattern Recognition Techniques, Technology and Applications*", 2016.
- S. Theodoridis and K. Koutroumbas; *An introduction to pattern recognition: A Matlab approach*, Academic Press 2010.
- S. Rogers and M. Girolami; "*A first course on machine learning*", CRC Press 2017.
- Y. Bengio, A. Courville and P. Vincent; *"Representation Learning: A Review and New Perspectives"*, arXiv:1206.5538
- S. Forrest and C. Beauchemin; *"Computer immunology"*, Immunological Reviews 216 (2007) 176–197.
- Y. LeCun, Y. Bengio and G. Hinton; *"Deep learning"*, Nature 521 (2015) 436-444.
- A. Niederlinski; *"A gentle guide to constraint logic programming via Eclipse"*, Gliwice, 2014.

Some additional references

- P. Doukhan, G. Oppenheim and M. S. Taqqu; "*Theory and Applications of Long-Range Dependence"*, Springer 2003.
- W. Wieczorek; *"Grammatical inference: Algorithms, Routines and Applications"*, Springer 2017.