Multilayer Perceptron (MLP) Application Guidelines

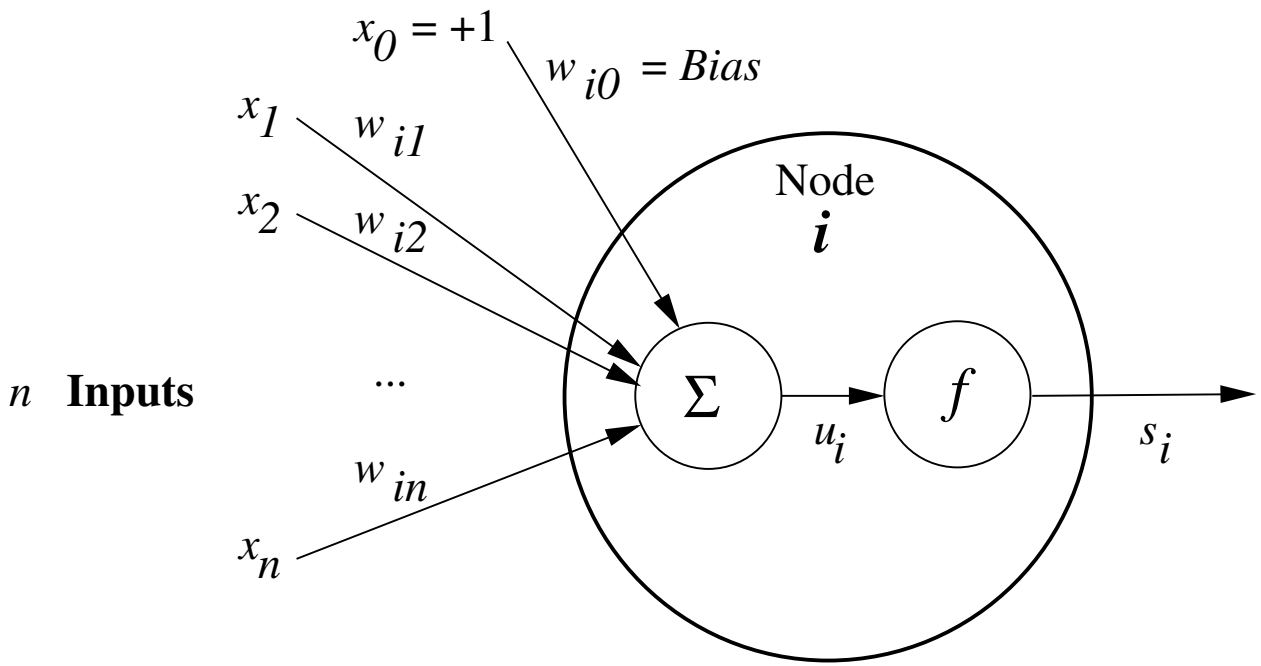
Paulo Cortez

pcortez@dsi.uminho.pt
http://www.dsi.uminho.pt/~pcortez
Departamento de Sistemas de Informação
University of Minho - Campus de Azurém
4800-058 Guimarães - Portugal

Node Activation

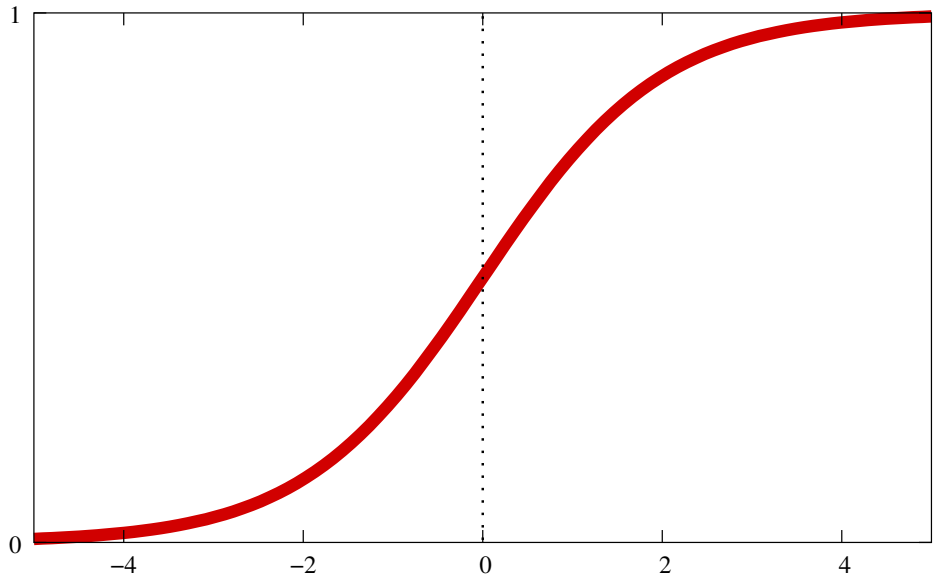
Further information in [Bishop, 1995] and [Sarle, 2003].

- Each node **outputs** an activation function applied over the weighted sum of its **inputs**: $s_i = f(w_{i,0} + \sum_{j \in I} w_{i,j} \times s_j)$

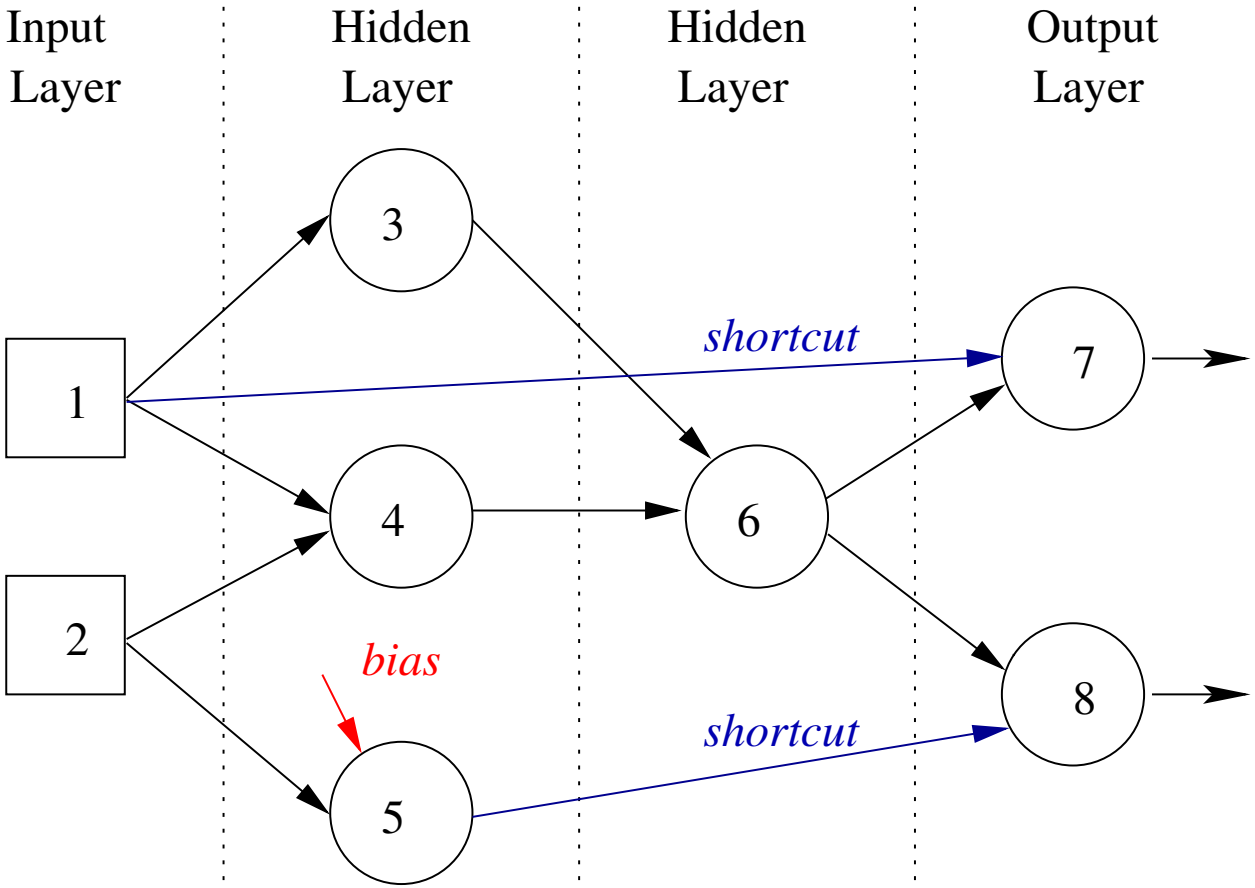


Activation functions

- Linear: $y = x$;
- Tanh: $y = tanh(x)$;
- Logistic (or sigmoid): $y = \frac{1}{1+e^{-x}}$;



- Only **feedforward** connections exist;
- Nodes are organized in **layers**;



Why MLPs?

- **Popularity** - the most used type of **NN**;
- **Universal Approximators** - general-purpose models, with a huge number of applications;
- **Nonlinearity** - capable of modeling complex functions;
- **Robustness** - good at ignoring irrelevant inputs and noise;
- **Adaptability** - can adapt its weights and/or topology in response to environment changes;
- **Easy of Use** - black-box point of view, can be used with few knowledge about the relationship of the function to be modeled.

Who is interested in MLPs?

- **Computer Scientists** : study properties of non-symbolic information processing;
- **Statisticians**: perform flexible data analysis;
- **Engineers**: exploit **MLP** capabilities in several areas;
- **Cognitive Scientists**: describe models of thinking;
- **Biologists**: interpret DNA sequences;
- ...

Applications [Cortez et al., 2002]

- Supervised Learning (Input/Output Mapping):
  - Classification (discrete outputs):
    - Diabetes in Pima Indians;
    - Sonar: Rocks vs Mines;
  - Regression (numeric outputs):
    - Prognostic Breast Cancer;
    - Pumadyn Robot Arm;
  - Reinforcement Learning (Output is not perfectly known):
    - Control (e.g., autonomous driving);
    - Game Playing (e.g., checkers);

Software [Sarle, 2003]

- Commercial:
  - SAS Enterprise Miner Software;
  - MATLAB Neural Network Toolbox;
  - STATISTICA: Neural Networks;
  - Clementine
    `http://www.spss.com/spssbi/clementine/index.htm`;
- Free:
  - Stuttgart Neural Network Simulator (SNNS) (C code source);
  - Joone: Java Object Oriented Neural Engine (code source);
  - R (statistical tool) `http://www.r-project.org/`;
  - WEKA (data mining)
    `http://www.cs.waikato.ac.nz/~ml/weka/`;
- Built Your Own: Better control, but take caution!

Classification Metrics

- The *confusion matrix* [Kohavi & Provost, 1998] matches the **predicted** and **actual** values;

Table: The 2×2 *confusion matrix*.

$\downarrow$ **actual** $\setminus$ **predicted** $\rightarrow$	**negative**	**positive**
negative	TN	FP
positive	FN	TP

- Three accuracy measures can be defined:
  - the **Sensitivity** (*Type II Error*) $= \frac{TP}{FN+TP} \times 100\,(\%)$;
  - the **Specificity** (*Type I Error*) ; $= \frac{TN}{TN+FP} \times 100\,(\%)$
  - the **Accuracy** $= \frac{TN+TP}{TN+FP+FN+TP} \times 100\,(\%)$;

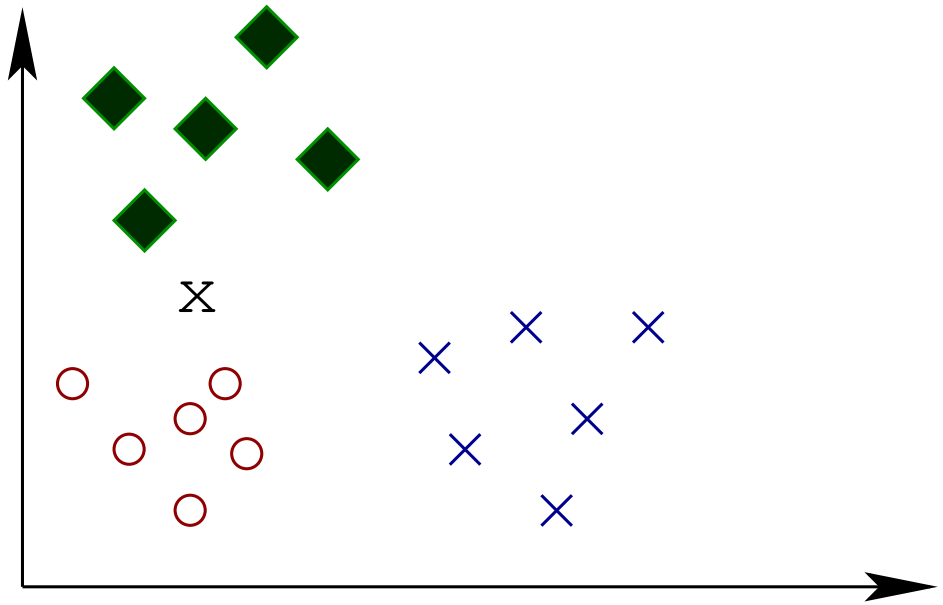
Regression

- The **error** e is given by: $e = d - \widehat{d}$ where d denotes the desired value and the $\widehat{d}$ estimated value (given by the model);
- Given a dataset with the function pairs $x_1 \longrightarrow d_1, \cdots, x_N \longrightarrow d_N$, we can compute:
  - **Mean Absolute Deviation (MAD)**: $MAD = \frac{\sum_{i=1}^{N} |e_i|}{N}$
  - **Sum Squared Error (SSE)**: $SSE = \sum_{i=1}^{N} e_i^2$
  - **Mean Squared Error (MSE)**: $MSE = \frac{SSE}{N}$
  - **Root Mean Squared Error (RMSE)**: $RMSE = \sqrt{MSE}$
  - Correlation metrics (scale independent):
    - Person [-1, 1];
    - Press r squared: $R^2 = 1 - \frac{\sum (y - \widehat{y})^2}{\sum (y - \overline{y})^2}$

Data Collection and Preprocessing

- Learning samples must be representative, hundreds or thousands of cases are required;
- Only numeric values can be fed (**discretization**);
- Treat missing values (**delete or substitute**);
- **Rescaling**:
  - **inputs**:
    - $y = \frac{x-(max+min)/2}{(max-min)/2}$ (**maximum–mininum**: range [-1,1])
    - $y = \frac{x-\bar{x}}{s}$ (**Mahalanobis scaling**: mean 0 and std 1)
  - **numeric outputs**: within [0,1] if logistic and [-1,1] if tanh.
    - $y = \frac{(x-min)}{max-min}$ (**maximum–mininum**: range $[0, 1]$)

- **Nominal variables** (with 2 or more non-ordered classes):
  - **1-of-C** coding: use one binary variable per class;
  - Color example: Blue - 1 0 0, Red - 0 1 0, Green - 0 0 1.



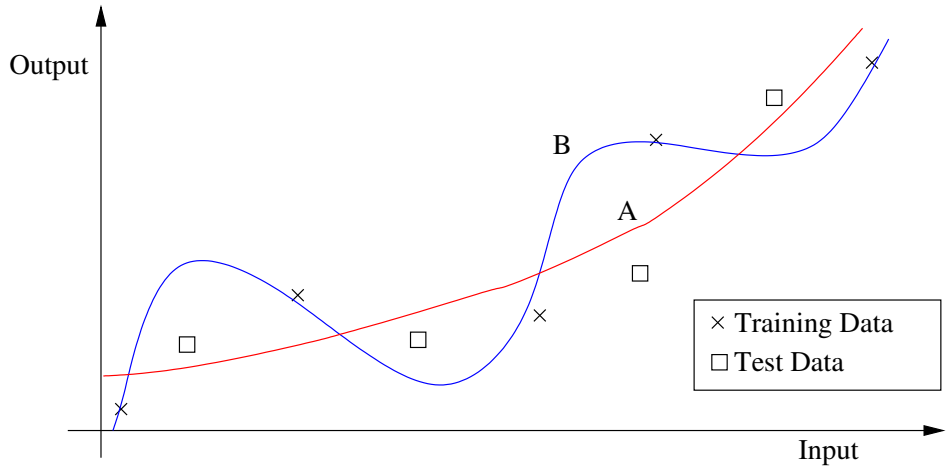
Consult [Flexer, 1996] for further details.

- **Holdout**, split the data into two exclusive sets, using random sampling:
  - **training**: used to set the **MLP** weights (2/3);
  - **test**: used to infer the **MLP** performance (1/3).
- **K-fold**, works as above but uses rotation:
  - data is split into K exclusive folds of equal size;
  - each part is used as a test set, being the rest used for training;
  - the overall performance is measured by averaging the K runs results;
  - **10-fold** most used if hundreds of examples;
  - **leave-one-out** (N-fold) used if less than 100 or 200 examples;
- **Third extra set** is needed if parameter tunning;

Training Algorithms

- Gradient-descent ($O(n)$) [Riedmiller, 1994]:
  - **Backpropagation (BP)** - most used, yet slow;
  - **Backpropagation with Momentum** - faster than **BP**, requires additional parameter tunning;
  - **QuickProp** - faster than **BP with Momentum**;
  - **RPROP** - faster than **QuickProp** and stable in terms of its internal parameters;
- **Levenberg-Marquardt**
  - One of the most accurate algorithms;
  - $O(n^2)$ in terms of computer (memory, speed) requirements (use with small networks);

Overfitting



- If possible use **large datasets**: $N \gg p$ (weights);
- **Model Selection** - apply several models and then choose the one with the best generalization estimate;
- **Regularization** - use learning penalties or restrictions:
  - **Early stopping** - stop when the estimate arises;
  - **Weight decay** - in each epoch slightly decrease the weights;
  - **Jitter** - training with added noise.

MLP Capabilities

- **Linear** learning when:
  - there are no hidden layers; or
  - only linear activation functions are used.
- **Nonlinear** learning:
  - any continuous function mapping can be learned with one hidden layer;
  - discontinuous complex functions can be learned with more hidden layers.
- It is better to perform only **one** classification/regression task per network; i.e., use **C/1** output node(s).
- Activation Functions:
  - Hidden Nodes: use the **logistic** (or **tanh**);
  - Output Nodes: if outputs bounded, apply the same function; else use the **linear** function;

Design Approaches

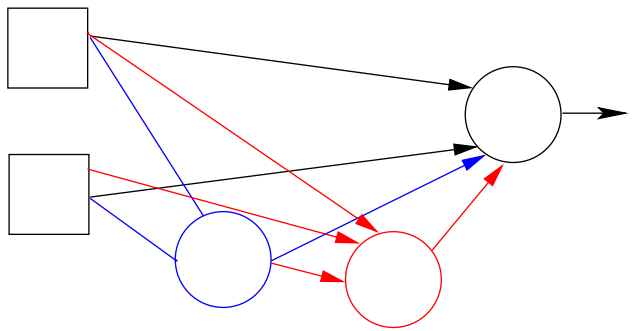
- Most common: use fully connected **NN** with one hidden layer, number of hidden nodes set by **trial-and-error**;
- **Hill-Climbing**: uses one searching point, being a new solution created from a previous one. Examples:
  - **Pruning Algorithms**;
  - **Constructive Algorithms**.
- **Beam search**: keeps a population of possible solutions. Examples:
  - **Genetic Algorithms**;
  - **Genetic Programming**;

Pruning Algorithms

- **Strategy**: Start with a **large** network and **prune** nodes and/or connections;
- Some pruning methods [Thimm & Fiesler, 1996]:
  - **Smallest Weight**;
  - **Autoprune** - based on a probability that a weight becomes zero;
  - **Optimal Brain Surgeon** - more complex, uses a full Hessian matrix.

Constructive Algorithms

- **Strategy**: Start with the **simplest** network and **add** nodes and/or connections;
- Several algorithms available [Kwok & Yeung, 1999]:
  - **Dynamic Node Creation** - one hidden unit added at a time, training of the whole network;
  - **Projection Pursuit Regression** - uses different activation functions, trains only the new hidden unit;
  - **Cascade Correlation**



Evolutionary Neural Networks [Yao, 1999]

- **Strategy**:
  - Start with a **population** of random generated **MLPs** (encoded in **chromosomes**);
  - Assign to each **MLP** a **fitness** (**Evaluation**);
  - Create new **MLPs** by **mutation** and **crossover** operations (**Recombination**);
  - Select the fittest **MLPs** and repeat the process until some criteria is met (**Selection**);

Evolutionary Neural Networks [Yao, 1999]

- **Representation**:
  - **Strong** - direct, low-level encoding of connections (most used) or nodes.
    - Used for small networks, prevents network from growing to large;
    - More efficient;
  - **Weak** – indirect, high-level encoding.
    - Not every structure is probable (favors regular networks);
    - Better scalability and biological plausibility;

Organ Failure Diagnosis [Silva et al., 2004]

- In **Intensive Care Units (ICUs)**, scoring the severity of illness has become a routine in daily practice;
- Yet, prognostic models (**logistic regression**) are static (computed with data collected within the first 24 hours);
- Organ dysfunction is a critical **ICU** task:
  - its rapid detection allow physicians to quickly respond with therapy;
  - multiple organ failure will increase the death probability;
- The **Sequential Organ Failure Assessment (SOFA)** is a costly and time consuming diary index (from 0 to 4);
- An organ is considered to fail when $SOFA \geq 3$.

Aim

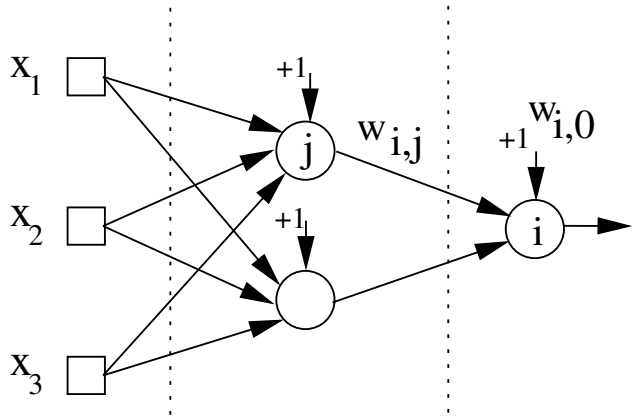
- Use **MLPs** for failure prediction (identified by high *SOFA* values) of six organs: **respiratory**, **coagulation**, **liver**, **cardiovascular**, **central nervous system** and **renal**.
- Several approaches will be tested, using different **feature selection**, **preprocessing** and **modeling** configurations;
- A particular focus will be given to the use of **daily intermediate adverse events**, obtained from four hourly bedside measurements.
- Clinical Data:
  - Part of the **EURICUS II** database;
  - Encompasses 5355 patients from 42 **ICUs** and 9 EU countries, during 10 months;
  - One entry per day (with a total of 30570).

Attribute	Description	Domain
respirat	Respiratory	$\{0, 1, 2, 3, 4\}$
coagulat	Coagulation	$\{0, 1, 2, 3, 4\}$
liver	Liver	$\{0, 1, 2, 3, 4\}$
cardiova	Cardiovascular	$\{0, 1, 2, 3, 4\}$
cns	Central nervous system	$\{0, 1, 2, 3, 4\}$
renal	Renal	$\{0, 1, 2, 3, 4\}$
admfrom	Admission origin	$\{1, \ldots, 7\}$
admtype	Admission type	$\{1, 2, 3\}$
sapsII	SAPSII score	$\{0, \ldots, 160\}$
age	Patients' age	$\{18, \ldots, 100\}$
nbporms	Number of *BP* events	$\{0, \ldots, 28\}$
nhrorms	Number of *HR* events	$\{0, \ldots, 26\}$
no2orms	Number of *O2* events	$\{0, \ldots, 30\}$
nurorms	Number of *UR* events	$\{0, \ldots, 29\}$

Preprocessing

- Six new attributes created, by sliding the $SOFA_{d-1}$ values into into each previous example;
- The last day of the patient's admission entries were discarded (remaining a total of 25309);
- The new attributes were transformed into binary variables: 0 , $if SOFA_d < 3$; 1 , *else*.
- Inputs rescaled into $[-1, 1]$ (**1-of-C** coding for the nominal attributes: *SOFA*, *admfrom* and *admtype*);

Modeling



- Fully connected MLP with **bias**, one hidden layer ($n_h = n/2$) and logistic activation functions;
- One **MLP** per organ $\rightarrow$ one output (predicted class given by the nearest class value);

Training, Metrics and Validation

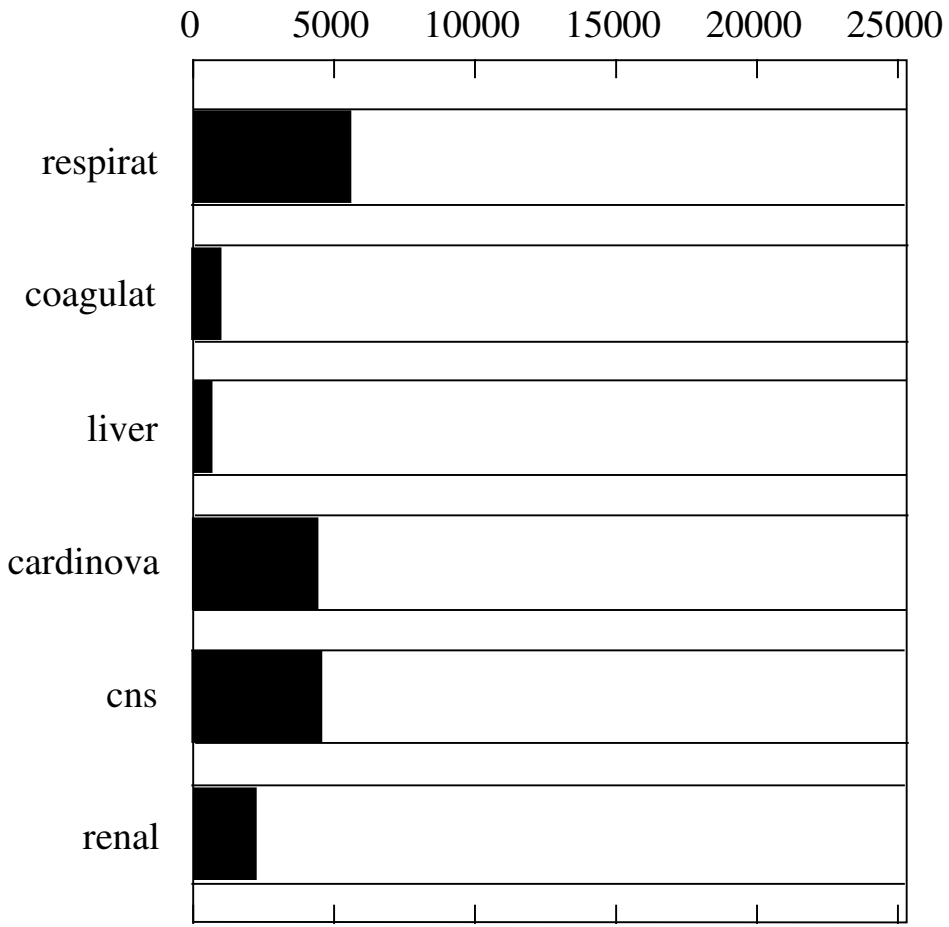
- Random weight initialization;
- **RPROP** algorithm, being stopped when the training error slope approaches zero or after 100 epochs;
- Metrics
  - **Sen.** (sensitivity);
  - **Spe.** (specificity);
  - **Ac.** (accuracy);
- Validation
  - Holdout with resampling;
  - Training set uses the first 2/3 of the examples;
  - Test set uses the rest of the examples (1/3);
  - Thirty runs applied to each model;

Feature Selection

Organ	A			B			C			D		
	Ac.	*Sen.*	*Spe.*	*Ac.*	*Sen.*	*Spe.*	*Ac.*	*Sen.*	*Spe.*	*Ac.*	*Sen.*	*Spe.*
respirat	86.3	72.4	90.2	86.2	70.0	90.8	77.9	4.4	98.8	77.6	1.8	99.4
coagulat	97.4	68.8	98.7	97.3	59.6	99.0	95.8	4.6	99.9	95.7	0.0	100
liver	98.3	68.6	99.1	98.3	60.2	99.4	97.3	7.6	99.9	97.3	0.0	100
cardinova	94.2	84.1	96.3	94.2	84.0	96.3	82.8	7.5	99.0	82.2	0.5	99.8
cns	95.7	92.7	96.4	95.7	92.3	96.4	83.5	23.4	97.1	81.6	0.4	99.9
renal	95.5	71.3	97.8	95.3	66.6	98.1	91.4	5.7	99.7	91.1	0.3	100
Mean	94.6	76.3	96.4	94.5	72.1	96.7	88.1	8.9	99.1	87.6	0.5	99.96

A - SOFA, **B** - All, **C** - Case mix and events, **D** - events.
Ac. - Accuracy, Sen. - Sensitivity, Spe - Specificity.

Balanced Training



Balanced Training

- There is a higher number of false (0) than true (1) examples!
- Solution:
  - **Balance** the training data: 2/3 of true examples, plus an equal number of false examples;
  - Test set with other 1/3 true entries and a number of negative ones in proportion to the original dataset;

Balanced Results

Organ	C			D		
	Acc.	*Sen.*	*Spe.*	*Acc.*	*Sen.*	*Spe.*
respirat	61.3	66.4	59.8	67.1	41.1	74.5
coagulat	67.6	66.8	67.7	73.7	41.5	75.1
liver	70.0	71.6	70.0	66.9	36.5	67.8
cardiova	65.9	62.5	66.7	68.2	37.9	74.8
cns	73.6	63.9	75.7	66.8	36.3	73.7
renal	67.8	65.6	68.0	73.2	37.6	76.6
Mean	67.7	66.2	68.0	69.3	38.5	73.8

Acc. - Accuracy, Sen. - Sensitivity, Spe - Specificity.

Improving Learning

- Balanced training decreased N, thus reducing the computational effort;
- The number of hidden nodes was increased to $n_h = 16$, being the maximum number of epochs set to 1000;

Organ	**C**			**C** (improved)		
	Acc.	*Sen.*	*Spe.*	*Acc.*	*Sen.*	*Spe.*
respirat	61.3	66.4	59.8	63.3	70.4	61.3
coagulat	67.6	66.8	67.7	70.0	72.0	69.9
liver	70.0	71.6	70.0	72.5	77.3	72.4
cardiova	65.9	62.5	66.7	69.1	66.3	69.8
cns	73.6	63.9	75.7	75.2	72.2	75.8
renal	67.8	65.6	68.0	71.9	70.5	72.0
Mean	67.7	66.2	68.0	70.3	71.5	70.2

Acc. - Accuracy, Sen. - Sensitivity, Spe - Specificity.

Comparison with Other Techniques

- Two methods were selected from the WEKA software:
  - **Naive Bayes** - a statistical algorithm based on probability estimation;
  - **JRIP** - a learner based on "IF-THEN" rules;

Orgão	Naive Bayes			JRIP			Neural Networks		
	Acc.	*Sen.*	*Spe.*	*Acc.*	*Sen.*	*Spe.*	*Acc.*	*Sen.*	*Spe.*
respirat	73.5	25.2	87.3	62.8	61.9	63.0	63.3	70.4	61.3
coagulat	83.3	24.8	85.8	67.8	62.4	68.0	70.0	72.0	69.9
liver	70.8	54.3	71.2	75.7	73.7	75.7	72.5	77.3	72.4
cardiova	73.4	33.4	82.0	66.6	70.3	65.8	69.1	66.3	69.8
cns	76.3	41.3	84.2	77.6	74.4	78.3	75.2	72.2	75.8
renal	76.8	45.6	79.9	69.1	68.5	69.2	71.9	70.5	72.0
Mean	75.7	37.4	81.7	69.9	68.5	70.15	70.3	71.5	70.2

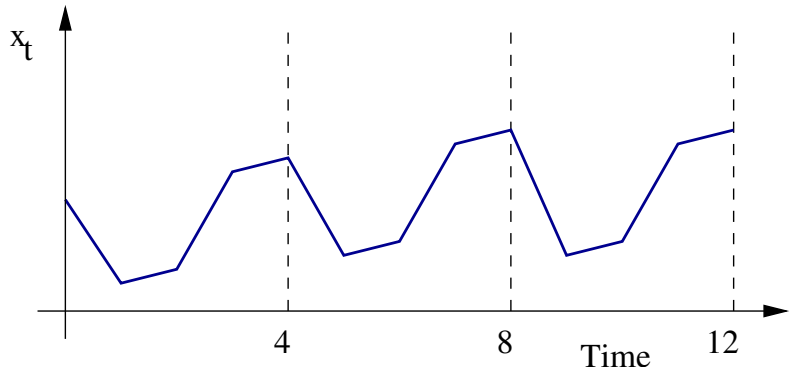
Acc. - Accuracy, Sen. - Sensitivity, Spe - Specificity.

Conclusions

- It is possible to diagnose organ failure by using cheap and fast intermediate outcomes (overall accuracy of 70%);
- The proposed approach opens room for the development of automatic clinical decision tools.

Time Series Forecasting [Cortez et al., 2001]

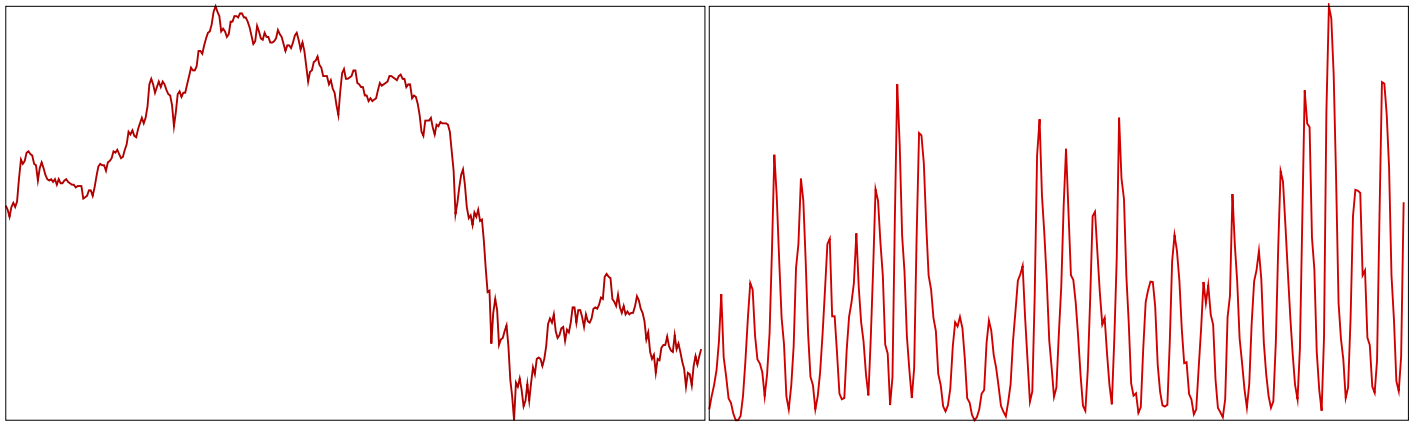
- A **Time Series** contains **time ordered** observations of an event:



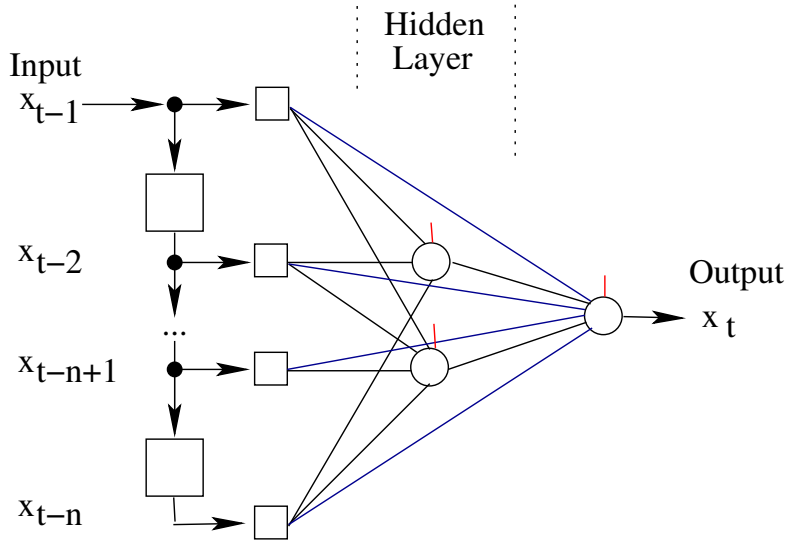
- **Time Series Forecasting (TSF)** uses past patterns to predict the future;
- Several **TSF** methods available:
  - **Exponential Smoothing (ES)**;
  - **Box-Jenkins (BJ)** methodology.

Time Series Data

Ten series were chosen from different domains, being classified into: **Seasonal and Trended (ST)**; **Seasonal (S)**; **Trended (T)**; **Nonlinear (N)**; and **Chaotic (C)**.



MLP model (TDFN)



- Training cases defined by a **Sliding Time Window**. Example: for the series $1, 2, 3, 4, 5, 6$, the sliding window $< 1, 2, 4 >$ defines the training examples $1, 3, 4 \rightarrow 5$ and $2, 4, 5 \rightarrow 6$;
- The number of parameters is given by: $p_{MLP} = n(n_h + 1) + 2n_h + 1$ (n_h – number of hidden nodes).

Training, Metrics and Validation

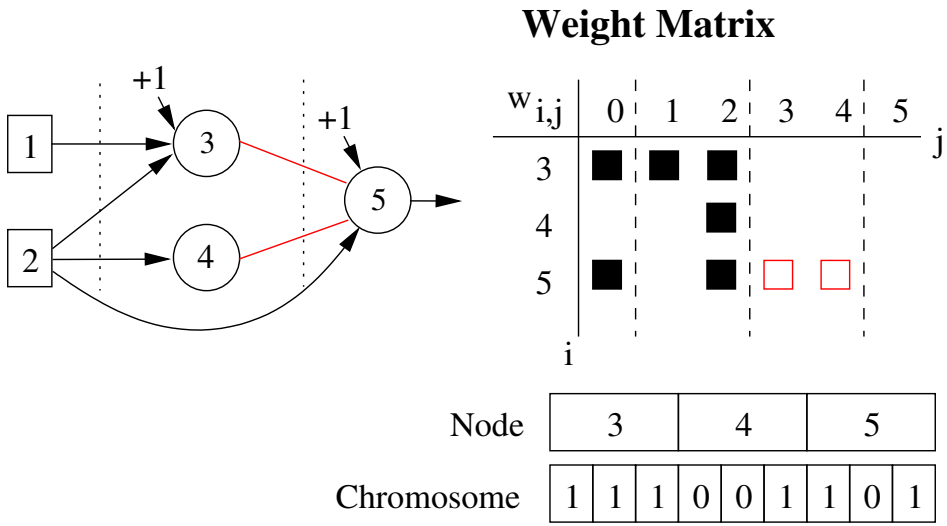
- Random weight initialization;
- **RPROP** algorithm, being stopped when the training error slope approaches zero or after 1000 epochs;
- Metrics from Information Theory:
  - **Akaike Information Criterion (AIC)**$= N \ln(SSE/N) + 2p$;
  - **Bayesian Information Criterion (BIC)**$= N \ln(SSE/N) + p \ln(N)$;
- Training Error ($RMSE$);
- *Theil's U*$= \frac{RMSE}{\sqrt{\frac{\sum_{i=t+1}^{t+L} (x_t - x_{t-1})^2}{L}}}$ (L – the number of forecasts);
- Validation:
  - No sampling, examples ordered by time;
  - Training set with first 90% of the examples (test set last 10%);
  - Thirty runs applied to each model;

Model Selection

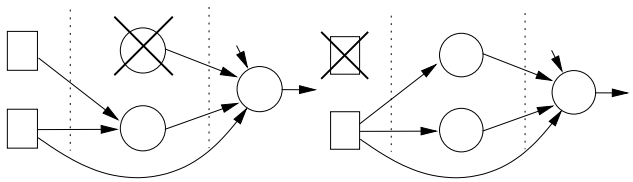
- Best sliding window and **MLP** topology? The **sunspots** simulations.

Window	n_h	p	$RMSE_t$	AIC	BIC	$RMSE_f$
$< 1, 2, \ldots, 13 >$	0	14	14.7	1355	1404	18.2±0.1
	6	104	11.6	1419	1784	20.5±0.7
	12	194	**9.0**	1474	2155	20.2±0.8
$< 1, 2, 9, 10, 11, 12 >$	0	5	14.8	1345	1369	17.8±0.3
	5	47	11.5	1302	1467	**17.0**±0.6
	13	111	9.4	1328	1717	19.0±0.8
$< 1, 2, 10, 11 >$	0	5	15.1	1352	1369	18.1±0.0
	1	11	14.1	1329	**1368**	17.8±0.3
	8	53	10.7	**1278**	1464	19.4±0.5

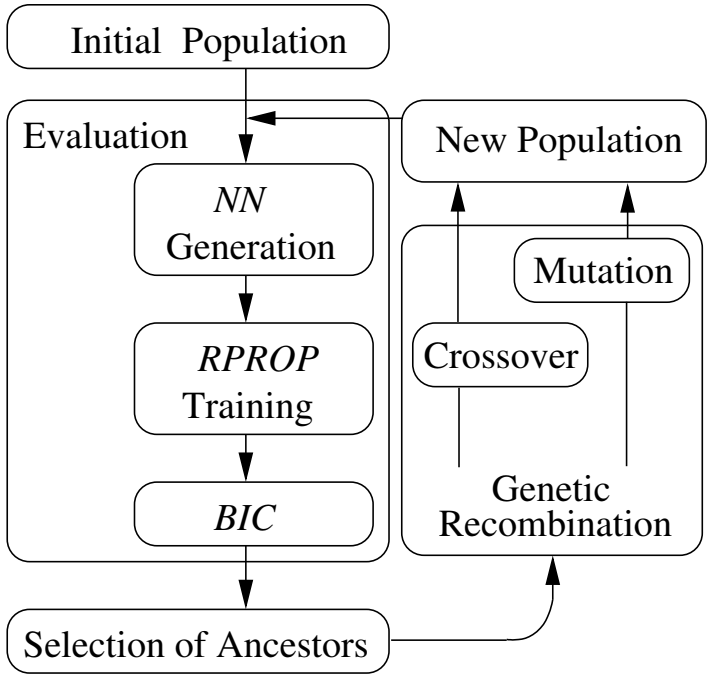
Evolutionary Neural Network Encoding



- Pruning:



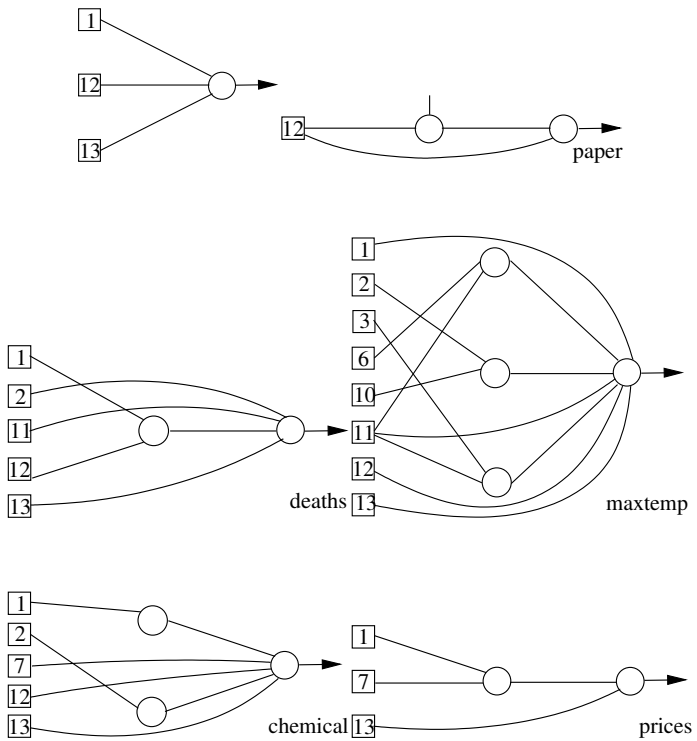
ENN Structure



- Uses a **Genetic Algorithm**, each bit codes a connection:
  - Fitness measured by the **BIC** value;
  - **Genetic Operators**: **2-point crossover** and **bit mutation**;
  - Maximum number of nodes set to 13 input and 6 hidden ones;

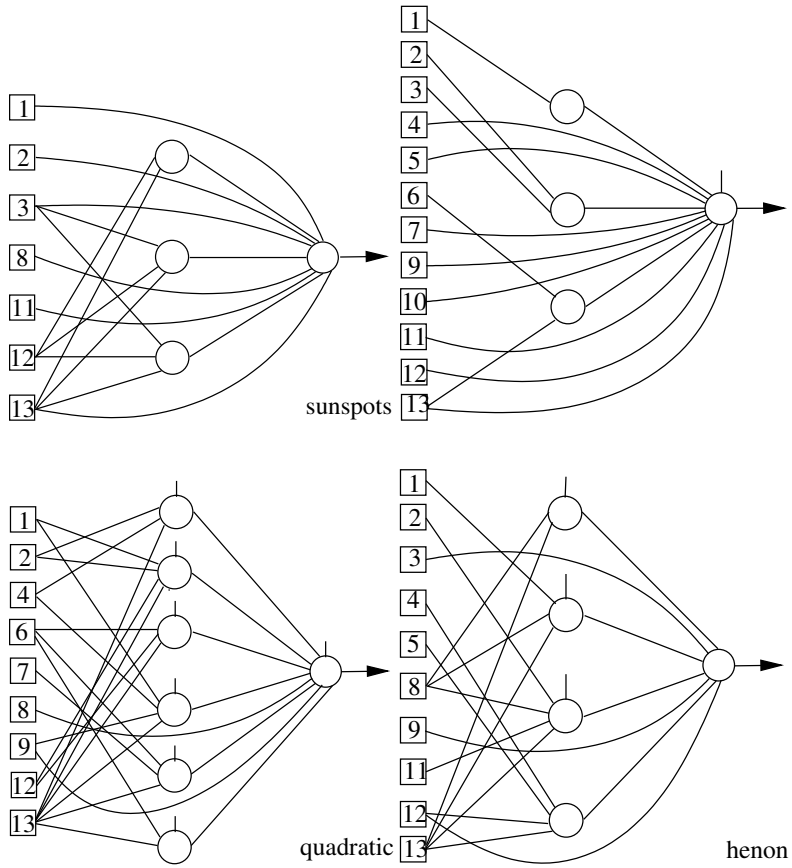
Optimal Topologies

Series: **passengers**, **paper**, **deaths**, **melbourne**, **chemical** and **prices**:

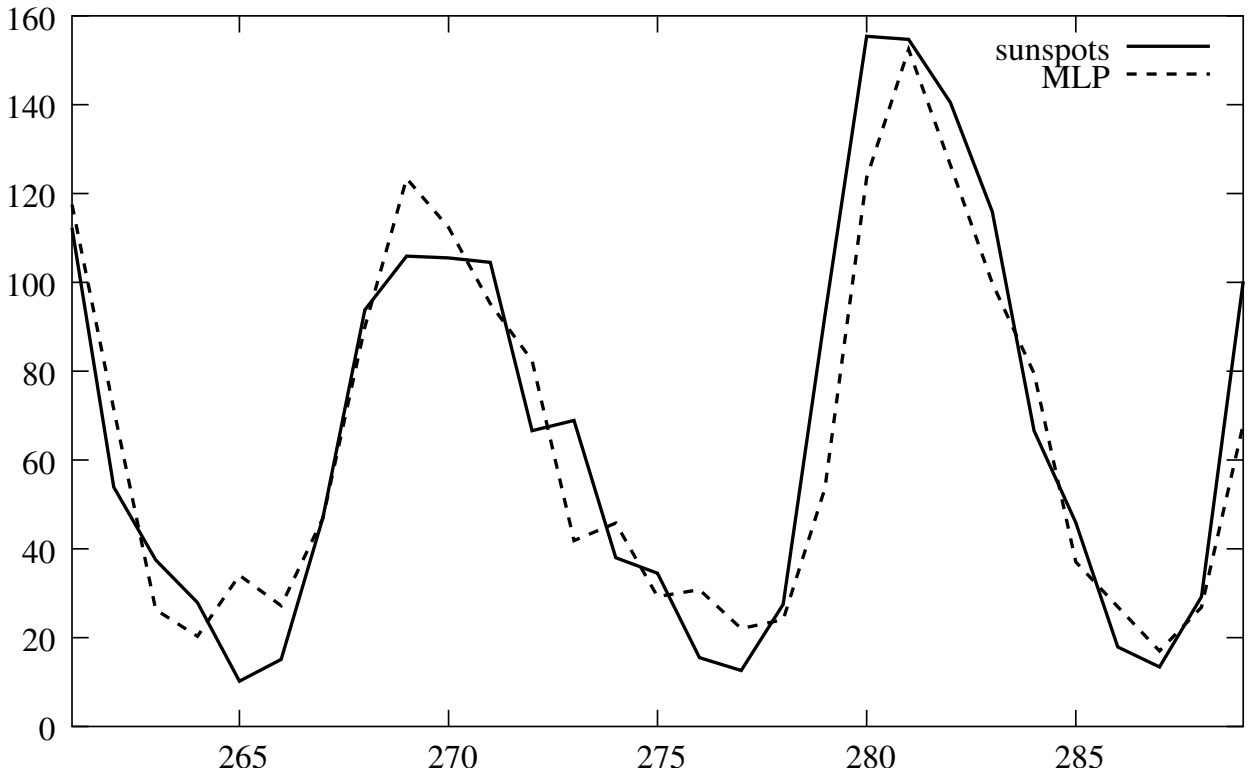


Optimal Topologies

Series: **sunspots**, **kobe**, **quadratic** and **henon**:



Example of the Sunspots series forecasts



Comparison (Theil's U values)

Series	ES	BJ	ENN
passengers (ST)	**0.104**	0.118	0.123
paper (ST)	**0.035**	0.076	0.057
deaths (S)	0.501	0.496	**0.420**
melborne (S)	0.137	0.186	**0.125**
chemical (T)	**0.830**	0.861	0.873
prices (T)	1.00	1.008	**0.997**
sunspots (N)	0.762	0.434	**0.287**
kobe (N)	0.823	0.027	**0.020**
quadratic (C)	0.424	0.424	**0.000**
henon (C)	0.417	0.326	**0.053**

Conclusions

- **ES** gives a better overall performance in the **seasonal** series;
- This scenario differs when considering other series, where the **evolutionary approach outperforms** both conventional **TSF** methods, specially for the last four **nonlinear** series;
- The **ENN** works autonomously and does not require any kind of statistical preprocessing or analysis;
- The drawback is the increase of computational effort;

Artificial Life Environment

- There is energy (e.g. grass) all over the field;
- Two populations of beings: preys and predators;
- Each artificial being is modeled by a MLP:
  - Inputs: **vision** within a certain range and angle (nothing, prey or predator);
  - Outputs: **actions** (nothing, move forward, rotate left or right);
  - Weights are directly coded into chromosomes, using real-valued representations;
- Evolutionary learning;


Bishop, C.
Neural Networks for Pattern Recognition.
Oxford University Press.


Cortez, P., Rocha, M., and Neves, J.
Evolving Time Series Forecasting Neural Network Models.
In *Proceedings of the Thirtd International Symposium on Adaptive Systems: Evolutionary Computation and Probabilistic Graphical Models (ISAS 2001)*, pages 84–91, Havana, Cuba.


Cortez, P., Rocha, M., and Neves, J.
A Lamarckian Approach for Neural Network Training.
Neural Processing Letters, 15(2):105–116.


Flexer, A.

Statistical evaluation of neural networks experiments: Minimum requirements and current practice.
In *Proceedings of the 13th European Meeting on Cybernetics and Systems Research*, volume 2, pages 1005–1008, Vienna, Austria.

Kohavi, R. and Provost, F.
Glossary of Terms.
Machine Learning, 30(2/3):271–274.

Kwok, T. and Yeung, D.
Constructive algorithms for structure learning in feedforward neural networks for regression problems problems: A survey.
IEEE Transactions on Neural Networks, 8(3):630–645.

Riedmiller, M.

Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16.

Sarle, W.
Neural Network Frequently Asked Questions.
Available from ftp://ftp.sas.com/pub/neural/FAQ.html.

Silva, A., Cortez, P., Santos, M., Gomes, L., and Neves, J.
Multiple Organ Failure Diagnosis Using Adverse Events and Neural Networks.
In et al., I. S., editor, *Proceedings of the 6th International Conference on Enterprise Information Systems (ICEIS 2004)*, pages 401–408, Porto, Portugal. INSTICC.

Thimm, G. and Fiesler, E.
Neural Network Pruning and Pruning Parameters.

In *The 1st Workshop on Soft Computing*.

Yao, X.

Evolving Artificial Neural Networks.

In *Proc. of the IEEE*, 87(9): 1423-1447.