The use of Artificial Neural Networks to adjust and robustness study of experience tables of maintenance in disability

Abder OULIDI[‡] Fréderic PLANCHET[§]

Pierre CHAUVET[**]

[‡] ✉ Institut de Mathématiques Appliquées – 44 Rue Rabelais – BP 808 – 49100 ANGERS CEDEX 01

oulidi@ima.uco.fr

[§] ISFA, Université de Lyon, Université Lyon 1 –50, Avenue Tony Garnier 69007 LYON
frederic.planchet@univ-lyon1.fr

[**] Institut de Mathématiques Appliquées – 44 Rue Rabelais – BP 808 – 49100 ANGERS CEDEX 01
pierre.chauvet@uco.fr

Contents

Abstract

Pricing and, more important, reserving "life / death" and "disability" risks are strictly defined by the regulation, which imposes particular constraints on the technical rate and the laws of occurrence or maintenance. However, the assessment of portfolios reserving differs from the standard one proposed by the BCAC. Insurance companies are increasingly forced to seek the construction of experience tables to manage these risks, especially since it is unrealistic today to expect offset losses by financial products.

Traditional adjustment methods, in actuarial literature, usually used to smooth the recovery curve rate estimated usually by the robust Adjusted Kaplan-Meier estimator, induce a model error due a boundary bias. The available data are usually sparse and poor quality on the border. Thus a boundary bias is due to weight allocation by the fixed symmetric argument outside the support of the gross curve, when smoothing close to the boundary is carried out.

The objective of this work is the use of Artificial Neural Networks (ANN) for adjustment and smoothing experience tables of maintenance in disability applied to a two cycles real set data. The artificial neural networks are parametric nonlinear models able to play an "universal approximator" role achieving a local and global approximation. Two architectures networks are particularly suited to model and smooth gross output rates: Feedforward Neural Networks (FNN) and Radial Basis Functions (RBF) Networks. The robustness of the ANN globally and especially at the edge of curve can be also studied.

Graphical tests are used to compare output surfaces rates obtained by neural networks with those obtained by Whittaker-Henderson framework.

Key words:
Disability, Kaplan-Meier, Whittaker-Henderson, Multilayer Neural Networks, Radial Basis Functions Networks

1. Introduction

The disability insurance policies distinguish disability corresponding to the temporary cessation of work and that of disability corresponding to the permanent cessation of work. The temporary disability insurance policy is used to provide workers with compensation for loss of wages caused by temporary non-occupational sickness or injury.

The estimate of temporary disablement experience table (principal objective of our study) is an incontrovertible requirement for the evaluation of mathematical reserves and pricing whose behavior differs from the standard tables proposed by the BCAC ([1]) in the French regulation. The big interest carried out both by the future European prudential framework Solvency II project than by the IFRS norms related especially with the calculate of the best estimate obligate the insurers more and more to use their proper experience tables based on their own portfolio in order to better manage the risk they covered. Thus, it not necessary to use without more precaution regulatory tables proposed by the BCAB ([1]) and introduced into the insurance regulation since 1996 concerning for example temporary disability.

In this paper, we are interested to deal with pricing and reserving temporary disability insurance policy. Those actuarial specifications are usually determined by the conjunction of: (a) act of incidence, which describes the disablement, happens and, (b) act of behaviour, which measures the time, spent in disablement, and thus determines the duration of sickness benefit. These two elements are both influenced by various factors risks as duration of deductible, age of the insured at the occurrence of disablement, gender, profession, cause of loss (accident or disease) …

A very sophisticated segmentation to estimate temporary disablement experience table run afoul rapidly of the problem of sparse and quality of available data especially on the border. To build its regulatory temporary disablement tables, the French regulation considers two-dimensional tables: one dimension for the age of the insured at occurrence of disablement, and another for date of cessation of work. Thus, we are led to build doubly subscript tables. The estimation of the recovery curve rates is usually dealing with an Adjusted Kaplan-Meier estimator ([2], [13], [9], [11], [5]). This robust nonparametric estimator makes it possible to take account of censured and truncated data. However, this estimator is a step function even when the exact curve is continuous, thus commonly we smooth and adjust the obtained curves by methods like *Whittaker-Henderson* and Spline ([6], [11], [2], [5]). In actuarial science, literature is rich with techniques borrowed from survival analysis to estimate gross recovery rates ([2], [11], [5]) and techniques borrowed from numerical analysis for smooth and adjust curves obtained ([11], [5]). However, since the insurer is constrain increasingly to use its own experience tables for reserving (best estimate), a major problem arises concerning the quality and sparse of data. Those traditional adjustment methods induce a model error due a boundary bias. The available data are usually sparse and poor quality on the border. Thus, a boundary bias is due to weight allocation by the fixed symmetric argument outside the support of the gross curve, when smoothing close to the boundary is carried out.

The certify actuary gathering data of the company conducts a statistical and actuarial analysis on the collected information but often does not have the possibility to verify this information. He is therefore confronted to different choices and decisions to take, in order to eliminate or correct outliers to finally get the most faithful database to reality.

We will introduce in this paper a new smoothing method, the Artificial Neural Networks (ANN). To our knowledge, this method has not been explored yet in the adjustment and smoothing experience tables in actuarial science.

In section 2, we will introduce the theory of artificial neural networks. Two architectures networks are particularly suited to model and smooth gross output rates: Feedforward Neural Networks (FNN) and Radial Basis Functions (RBF) Networks. We also introduce fidelity and regularity criterions. FNN realize a global approximation by minimizing the total sum-squared error between the network and the data. By varying the number of neurons and using a regularization function, it is possible to slide between high/low precision and smoothness of the modelling of experience tables. However, FNN do not allow local settings on some parts of the data, and the choice of the number of neurons is empirical. RBF have the advantage to allow constructive approach (neurons are added iteratively) at the cost of a longer computation time: to global approximation, they add a local approach taking into account the particularities of some "areas" of the table.

In section 3 we present the studied portfolio and some exploratory statistics.

In section 4 we present a raw recovery rate obtained by Adjusted Kaplan-Meier. Statistical tests are used to compare output surfaces rates obtained by neural networks with those obtained by Whittaker-Henderson framework.

In section 5 we will conclude and give some recommendations.

2. Artificial Neural Networks and Data Modeling

2.1 Main concepts

Artificial neural networks (ANN) are mathematical models inspired from Biological neural networks (BNN). They retain only some concepts, BNN being much more complex and sophisticated (with several organization levels and information pathways). One can find numerous definitions for artificial neural networks. We will retain this one, from the DARPA Neural Network Study group ([4]): "a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes". The main point is that fixing the network structure and the node (so-called neuron) processing function, the ANN can be considered as a parametric model whose connection strengths are the parameters to adjust with data. Generally, a neuron is modelled by:

$$o = F\left(\vec{w}.\vec{e}\right) \tag{0.1}$$

where

$\vec{e} \in IR^n$ is an input vector (e_i is a data or the output of the i-th neuron connected to the current neuron),

$\vec{w} \in IR^n$ is the weights vector (w_i is the weight for input e_i),

$F \quad IR \to IR$ is the transfer function applied to the scalar product $\vec{w}.\vec{e}$,

$o \in IR$ is the neuron output.

The behaviour of ANN can be dynamic or not, depending on the structure and the neuron processing function: it is possible to add a delay on the connection between two neurons, and to replace (1) by formulas like $do(t)/dt = -\alpha o(t) + F\left(\vec{w}.\vec{e}(t)\right)$. It allows example to generate a time-varying output from a constant input (e.g. giving a target, generate a trajectory). In the same way, an ANN can be linear or not, but in case of linearity they are only a rewriting of classical regression models or filters. In this work, we use ANN as a non-linear regression model.

If computing the output of an ANN from its input is relatively easy, it is by far more complex to compute its parameters to achieve a precise task, for example to fit to data. Because ANN is a branch of artificial intelligence and cognitive science, this is called the learning or training phase. The goal of ANN training is to estimate a set of weights by minimizing a criterion J, the global error between the outputs of the network and a training dataset. For regression, the most commonly used error function is the classical sum-squared of errors. In this case, training relies on optimization algorithms like descent methods (gradient, Newton, Levenberg-Marquardt), heuristics and meta-heuristics (genetic algorithms).

The aim of our work is to use ANN for data adjustment in a 2-dimensional case, and to compare neural network models with the Whittaker-Henderson framework. The problem to solve can be described as follows. We have a table with x_i (i=1...N_x) and y_j (j=1...N_y) real entries. At each (x_i, y_j) is associated the real z_{ij}. We suppose that it exists a function $f \quad IR^2 \to IR$ such that $z_{ij}=f(x_i, y_j)$ for all i=1... N_x and j=1... N_y. Because f is unknown, we approximate f by an arbitrarily chosen model whose parameters have to be adjusted. Two types of ANN are well suited for this task: the Feedforward neural network (FNN) and the Radial basis function network (RBN). Their main advantages for us are: (i) FNN and RBN are functions from IR^n to IR^m (no dynamic behaviour) with $n, m \in IN^*$ to specify; (ii) their implementation is widespread, particularly for FNN, in statistical (*R*, *SPAD*, *SAS*...) and computing (*Matlab*, *Scilab* ...) software. In next paragraphs, we describe FNN and RBN relatively to our 2-dimensional regression problem.

2.2 Feedforward neural networks

A FNN is made of successive layers numbered from 1 to N_l, layers (1) to (N_l-1) being called hidden layers and layer (N_l) the output layer. Each hidden layer contains an arbitrarily number $N^{(k)}$ of neurons with the same transfer function $F^{(k)}$; these neurons are never connected between them. The number of neurons and the transfer function of the output layer depend on the problem. Neurons of a layer (k) take their inputs only from the neuron's outputs of the layer (k-1). The input $\vec{e} \in IR^n$ of the first layer is imposed by the user. The output $o \in IR^m$ of the layer (N_l) is the response of the FNN to input $\vec{e}$. The mathematical expression of the FNN is:

$$\vec{x}^{(1)} = \vec{F}^{(1)}\left(\mathbf{W}^{(1)}\vec{e}\right) \tag{0.2}$$

$$\vec{x}^{(k)} = \vec{F}^{(k)}\left(\mathbf{W}^{(k)}\vec{x}^{(k-1)}\right), \; k=2 \ldots N_l\text{-}1 \tag{0.3}$$

$$\vec{o} = \vec{F}^{(N_l)}\left(\mathbf{W}^{(N_l)}\vec{x}^{(N_l-1)}\right) \tag{0.4}$$

with

$\mathbf{W}^{(k)}$ the weights matrix of layer (k), where $w_{ij}^{(k)}$ is the weight between neuron j from layer (k-1) and neuron i from layer (k),

$x^{(k)} \in \vec{IR}^{N_k}$ the output of layer (k), where $x_i^{(k)}$ is the output of neuron i with $x_i^{(k)} = F^{(k)}\left(\sum_{j=1}^{N^{(k-1)}} w_{ij}^{(k)} x_j^{(k-1)}\right)$.

Common choices for neurons transfer functions are $F^{(k)}(x) = \frac{1}{1+e^{-x}}$ (so called *logsig* function) or $F^{(k)}(x) = \tanh(x)$: they are infinitely differentiable and their derivatives are fast to compute, allowing an efficient use of gradient-based optimization algorithms to compute the weights matrices $\mathbf{W}^{(k)}$. For the interpolation problem in 2 dimensions that we address, we must take:

$o \in IR^2$, $o \in IR$ (therefore 1 neuron on the last layer N_l),

the training dataset $D = \left\{\left(\vec{e}^{(p)}, o^{(p)}\right)\right\}_{p=1,\ldots,N_x.N_y}$ with $\vec{e}^{((i-1).N_x+j)} = \left(x_i \;\; y_j\right)^t$ and $o^{((i-1).N_x+j)} = z_{ij}$;

we choose N_l=2 and $J\left(\left(\mathbf{W}^{(k)}\right)_{k=1\ldots N_l}\right) = \frac{1}{N_x N_y}\sum_{p=1}^{N_x N_y}\left(o^{(p)} - o\left(\vec{e}^{(p)}\right)\right)^2$.

Through the criterion J (so called *mse* criterion), this FNN realizes a global approximation by minimizing the total mean sum-squared error between the network and the data. By varying the number of neurons on layers 1 to N_l-1, it is possible to slide between high/low precision

and smoothness of the modelling of the 2 dimensions table. However, FNN do not allow local settings on some parts of the data, and the choice of the number of neurons is empirical. It has been shown by Hornik, Funahashi and Cybenko in 1989 ([3], [7]) that a FNN with at least one hidden layer can arbitrarily approximate any continuous nonlinear function on a compact interval. This result is not constructive: it does not provide any information on the number of neurons to use to achieve a given approximation.

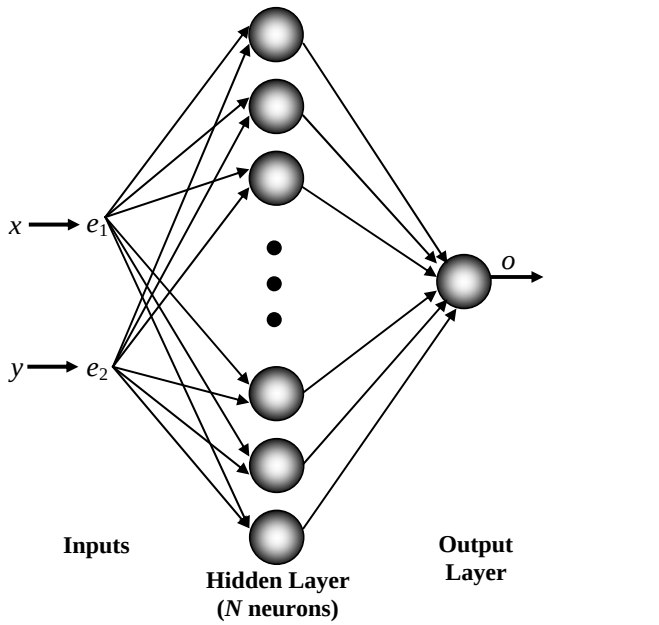


Figure 1: FNN with 2 inputs, 1 hidden layer and 1

2.3 Radial basis neural networks

A RBN can be viewed as a FNN with a specific structure that allows a constructive approach. It has one hidden layer with radial basis transfer function (the *radbas* function) receiving the inputs, and an output linear layer. In our case the output *o* is a real, therefore there is a unique output neuron:

$$o = \sum_{i=1}^{N} a_i \rho\left(\left\|\vec{e} - \vec{b_i}\right\|\right) \tag{0.5}$$

where *N* is the number of neurons on the hidden layer, $\vec{e} \in IR^2$ the input, $\vec{b_i} \in IR^2$ the weights of the i^{th} neuron, a_i the weight between the i^{th} neuron and the output neuron. The norm is Euclidian and the most common used transfer function ρ is a Gaussian:

$$\rho\left(\left\|\vec{e} - \vec{b_i}\right\|\right) = \exp\left(-\beta\left\|\vec{e} - \vec{b_i}\right\|^2\right) \tag{0.6}$$

One can see that RBN, compared to FNN, put emphasis on local data. It is clear that if input is far from the i^{th} neuron (in the sense $\left\|\vec{e} - \vec{b_i}\right\| >> 1$), the i^{th} neuron will have little effect on

output *o*. But like in FNN, the output neuron aggregates results from all hidden neurons and then realizes a global approximation too.

We use an iterative design of the network: neurons are added one at a time until the sum-squared error falls beneath an error goal or a maximum number of neurons has been reached. Thus, unlike FNN, RBN authorizes a constructive approach. Then, why not always use RBN instead of FNN? A good response is provided in ([10]): "*Radial basis networks […] tend to have many times more neurons than a comparable feedforward network with* `tanh` *or* `logsig` *neurons in the hidden layer. This is because sigmoid neurons can have outputs over a large region of the input space, while* `radbas` *neurons only respond to relatively small regions of the input space. The result is that the larger the input space (in terms of number of inputs, and the ranges those inputs vary over) the more* `radbas` *neurons required.*"

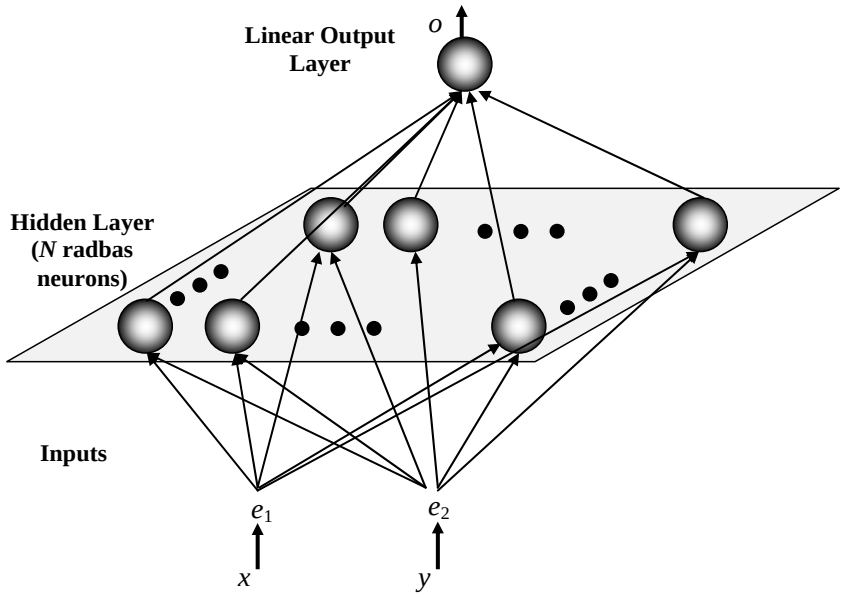


Figure 2: RBN with 2 inputs and 1 output.

2.4 Fidelity versus regularity

The choice of the number *N* of neurons on the hidden layer for the FNN and the RBN is crucial. A low number means a very poor performance, or fidelity, of the network: the *mse* stays high, therefore the network seen as a mapping function realizes a bad approximation of the theoretical function *f* (paragraph 2.1). Instead, a large number of neurons will allow the network to fit exactly all the data (resulting in a very low *mse*), including noise and biased observations, and sometimes to generate false information. The regularity of the mapping function will be low. Finally, the fidelity and the regularity of the network depend strongly on *N*. Because *N* is an integer defining the structure of the network, it cannot be adjusted like the synaptic weights. It exists some meta-heuristics (like genetic algorithms) to adjust *N* and other elements of the model (like the number of layers, the type of neuron,

etc.), but this is a low process. In our case, we will vary *N* between 2 reasonable values to find the best network.

Because we have to find a balance between fidelity and regularity, we need 2 criteria: the *mse* value at the end of the learning process for the fidelity, and a criterion for regularity. Regularity can be viewed as a capacity of generalization of the neural network model. With a big amount of data obtained from direct observation, we can build randomly two sets: S_T to train the network, S_G to test the capacity of generalization. During the training, the *mse* for S_T (=*mse*(S_T)), and the *mse* for S_G (=*mse*(S_G)), are computed at each iteration. The training is stopped when *mse*(S_G) begins to increase or *mse*(S_T)<ε and *mse*(S_G) decreases. The best *N* is one for which the network has the lowest values of *mse*(S_T) and *mse*(S_G). When the data are not so numerous, the risk to lose critical information is too high to remove data from S_G to build S_T. In our case, a good measure of the regularity is the smoothness of the surface built from the mapping neural network. A classical measure of the smoothness is the "energy", which is the space integral of the second order derivatives square of the mapping network instance:

$$E = \iint \left[\left(\frac{\partial^2 f}{\partial e_1^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial e_1 \partial e_2} \right)^2 + \left(\frac{\partial^2 f}{\partial e_2^2} \right)^2 \right] de_1 de_2 \ .$$

We will choose the network with *N* hidden neurons that minimizes *mse*(S_T) and *E*.

3. Presentation of the studied portfolio

We present her the studied portfolio and give some characteristics and some explanatory statistics. It consists on two cycles real set data borrowed from a big French insurance company.

The studied portfolio relates to employees in sick leave having a guarantee on the temporary incapacity with variable Franchise Covers. The stops are spread out over 8 years between 2000 and 2007, with sickness date going from 2000 to 2006. The adjustment and smoothing experience tables of maintenance in disability will thus applied to a two cycles real set data. In the portfolio 112.650 contracts are concerned.

We removed "extreme" data (one does not preserve that the data for an age between 23 and 64 years, and a duration between 1 and 34 months).

4. Application to a two cycles real set data

Our aim is to build a function $f\ IR^2 \to IR$ taking in input the entry age and duration of disablement, and giving the rate of maintenance in disability. This is done in this paragraph by using two kinds of neural networks: FNN and RBN. These neural networks have 2 inputs (e_1, e_2), one hidden layer and an output layer with one neuron providing output *o*. Their parameters (the "synaptic weights") are adjusted so that they give a "reasonable" approximation of an experience table of maintenance in disability. By "reasonable", we

mean a modelling function minimizing the error with the data is while keeping good smoothness properties. This is achieved for neural networks by minimizing the *mse* criterion, and by adjusting the number of neurons on the hidden layer (a kind of equivalent to the degree of an approximation polynomial). To find a neural network instance that balance performance and smoothness, we compute the final *mse* value and the energy *E* for several network instances by varying the number *N* of hidden neurons.

Next experiments were performed with the *Matlab Neural Network toolbox*.

4.1 Feedforward Neural Networks

The method used to compute the FNN parameters is the iterative *Levenberg-Marquardt* algorithm (LMA): from an initial set of parameters, LMA computes step by step a final parameter set approximating criterion's minima. Because the FNN has multiple local minima and there is no proof that it exists a unique global minima, the final parameters depend on the choice of the initial parameters. The initial set is chosen randomly; therefore, the final set and the final criteria value are random variables. For a given number of neurons on the hidden layer, we drive several runs and we compute the mean final criterion *J* value, the sample standard deviation, the best (minimal) final criterion value and the energy *E* of the network with this best final criterion value. Our FNN is defined by:

$x_i = F^{(1)}\left(w_{i1}^{(1)} e_1 + w_{i2}^{(1)} e_2\right), i = 1...N$, outputs of the hidden layer with *N* neurons;

$o = F^{(2)}\left(\sum_{j=1}^{N} w_{1j}^{(2)} x_j\right)$, the output of the network.

Therefore:

$$\frac{\partial^2 o}{\partial e_1^2} = o''.\left(\sum_{j=1}^{N} w_{1j}^{(2)} w_{j1}^{(1)} x_j'\right)^2 + o'.\sum_{j=1}^{N} w_{1j}^{(2)} \left(w_{j1}^{(1)}\right)^2 x_j'' ,$$

$$\frac{\partial^2 o}{\partial e_2^2} = o''.\left(\sum_{j=1}^{N} w_{1j}^{(2)} w_{j2}^{(1)} x_j'\right)^2 + o'.\sum_{j=1}^{N} w_{1j}^{(2)} \left(w_{j2}^{(1)}\right)^2 x_j'' ,$$

$$\frac{\partial^2 o}{\partial e_1 \partial e_2} = o''.\left(\sum_{j=1}^{N} w_{1j}^{(2)} w_{j1}^{(1)} x_j'\right).\left(\sum_{j=1}^{N} w_{1j}^{(2)} w_{j2}^{(1)} x_j'\right) + o'.\sum_{j=1}^{N} w_{1j}^{(2)} w_{j1}^{(1)} w_{j2}^{(1)} x_j'' ,$$

where:

$x_i' = F^{(1)'}\left(w_{i1}^{(1)} e_1 + w_{i2}^{(1)} e_2\right)$ and $x_i'' = F^{(1)''}\left(w_{i1}^{(1)} e_1 + w_{i2}^{(1)} e_2\right)$ for *i*=1,...*N*,

$o' = F^{(2)\prime}\left(\sum_{j=1}^{N} w_{1j}^{(2)} x_j\right)$ and $o'' = F^{(2)\prime\prime}\left(\sum_{j=1}^{N} w_{1j}^{(2)} x_j\right)$.

The smoothness measure is given by the numerical computation of $E = \iint\left[\left(\frac{\partial^2 o}{\partial e_1^2}\right)^2 + 2\left(\frac{\partial^2 o}{\partial e_1 \partial e_2}\right)^2 + \left(\frac{\partial^2 o}{\partial e_2^2}\right)^2\right] de_1 de_2$. Results are given in table 1, for the number of neurons on the hidden layer varying between 60 and 100. Training of the network is done with the *Levenberg-Marquardt* method implemented in the *neural network Matlab toolbox*.

Number of neurons	Mean MSE	Std MSE	Best MSE	Energy
60	0.42096808	0.06743410	0.26076883	8.43523682
65	0.40669444	0.05891835	0.25885191	31.40824642
70	0.40441117	0.05663192	0.28586850	7.81435477
75	0.41000166	0.19217010	0.24349132	144.39913091
80	0.38882161	0.06090817	0.27519068	6.04214295
85	0.37923952	0.06740259	0.25424002	7.19054718
90	1.48898155	7.90793967	0.22797307	6.86852183
95	0.36839737	0.05577831	0.25078559	5.83491647
100	0.35093925	0.09566895	0.23714845	6.62597431

Table 1: results in term of fidelity (Mean *mse*, Standard deviation *mse*, Best *mse*) and regularity (energy value E) for number of neurons on the hidden layer varying between 60 and 100, 100 runs for each network instance, a maximum number of iterations per run equals to 1000 and an mse goal equals to 1e-3.

4.2 Radial basis Neural Networks

The method used to compute the RBN parameters is the constructive approach described in paragraph 2.3, and proposed in *Matlab* with the function *newrb*. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons has been reached. More this maximum number of neurons is high, more the fidelity is high and the smoothness low. Like for FNN, we vary this number and for each instance of network we compute the final criterion *J* value and the energy *E*. Note that the learning process used in *newrb* is deterministic, we need only one run for each instance. The output of our RBN defined with a Gaussian function is:

$$o = \sum_{i=1}^{N} a_i \tilde{\rho}\left(\left(e_1 - b_{i1}\right)^2 + \left(e_2 - b_{i2}\right)^2\right),$$

where *N* is the number of hidden neurons and $\tilde{\rho}(x) = e^{-\beta x}$. Therefore, the derivatives used to compute *E* are:

$$\frac{\partial^2 o}{\partial e_1^2} = 2\sum_{i=1}^{N} a_i.\left(2\tilde{\rho}''.\left(e_1 - b_{i1}\right)^2 + \tilde{\rho}'\right),$$

$$\frac{\partial^2 o}{\partial e_2^2} = 2\sum_{i=1}^{N} a_i.\left(2\tilde{\rho}''.\left(e_2 - b_{i2}\right)^2 + \tilde{\rho}'\right),$$

$$\frac{\partial^2 o}{\partial e_1 \partial e_2} = 4\sum_{i=1}^{N} a_i.\tilde{\rho}''.\left(e_1 - b_{i1}\right).\left(e_2 - b_{i2}\right),$$

where $\tilde{\rho}' = \tilde{\rho}'\left(\left(e_1 - b_{i1}\right)^2 + \left(e_2 - b_{i2}\right)^2\right)$ and $\tilde{\rho}'' = \tilde{\rho}''\left(\left(e_1 - b_{i1}\right)^2 + \left(e_2 - b_{i2}\right)^2\right)$.

Results are given in table 2, for the number of neurons on the hidden layer varying between 40 and 100, and the spread parameter β=4.0.

Number of Neurons	MSE	Energy
40	0.71491747	0.06835912
45	0.64167721	0.06254472
50	0.59439553	0.06426738
55	0.56916545	0.08467440
60	0.55393950	0.08775740
65	0.54158095	0.09013596
70	0.53066322	0.08813694
75	0.52003273	0.10904256
80	0.50202921	1.24634109
85	0.49597999	2.44795301
90	0.49206411	2.34414755
95	0.48904656	2.33943448
100	0.48477024	2.28637020

Table 2: results in term of fidelity (*mse*) and regularity (energy E) for a number of RBF neurons between 40 and 100.

For a number of neurons varying from 40 to 100, one can see that the FNN has a stable best mse (between 0.227 and 0.286) and a corresponding energy that can be very irregular. But if we do not take into account the experiment for 65 and 75 neurons, the energy vary between 5.83 and 8.44. A good choice seems the FNN with 95 neurons: it has the lower energy (5.835) and a good final mse (0.251).

The RBF has a decreasing mse with the increase of the number of neurons. The corresponding energy increases in increments, with a big increment between 75 and 85 neurons. A good choice seems to be the RBN with 100 neurons: it has the lower mse of all RBN (0.485) and an energy twice smaller than the 95-neurons FNN (2.286). The RBN with fewer neurons have a bad fidelity and cannot be retained.

We have computed an approximation of the energy of the table using a second order finite difference schema: it is equal to 5.3224. Generally this schema provides an overestimation of the energy, then the 95-neurons FNN and the 100-neurons RBN with respective energy of 5.835 and 2.286 are reasonably smooth compared to the table (see figure 4).

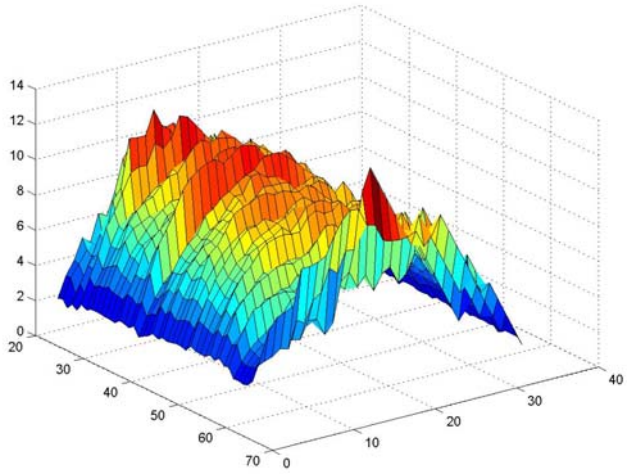


Figure 3: the data table to fit.

Our experiments show the ability of FNN and RBN to adjust the experience table, but with their own advantages and disadvantages. Without looking at the numerical results, independently from the experience tables, the training of a RBF is simpler than the training of a FNN because it is deterministic and finally faster. On the other side, a FNN is able to find a better local minimum for the mse with less neurons. Finally, the 95-neurons FNN has a better fidelity and a lower smoothness than the 100-neurons RBN. It must be noted here that first experiments were driven with all the original data (table with age between 21 and 65, duration between 0 and 35). In this case, the FNN adjusted itself better to the boundary data than the RBN.

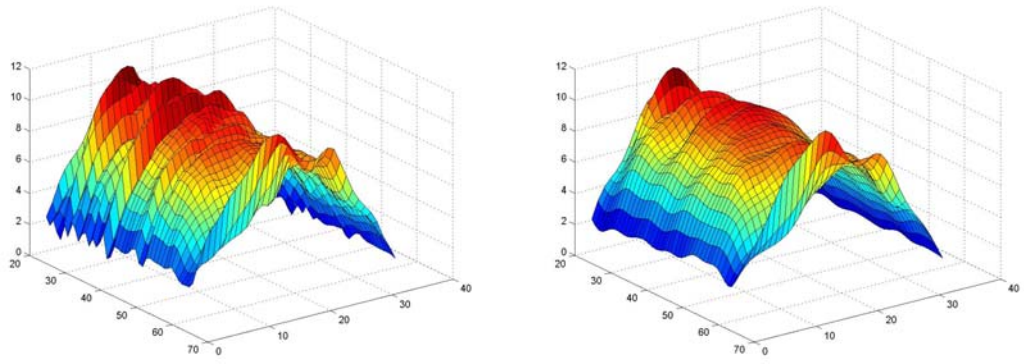


Figure 4: on left, the FNN mapping with 95 neurons (mse=0.251, Energy=5.835); on right, the RBF mapping with 100 neurons (mse=0.485, Energy=2.286).

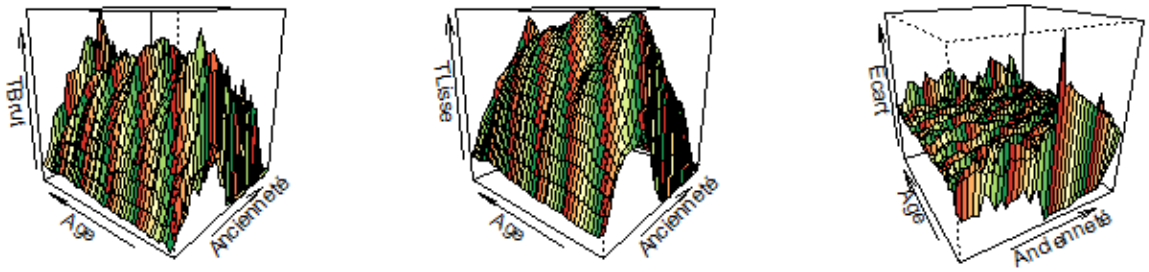
4.3 Whittaker-Henderson smoothing

Among classical methods one can use splines or Whittaker-Henderson smoothing. As shown in [12], both approaches lead to very similar results, so we choose to present here the Whittaker-Henderson model, which is simpler to implement.

Just like with RNN and RBN we have to choose weight for smoothness and regularity; more precisely we denote (cf. [11]):

$$F = \sum_{i=1}^{p}\sum_{j=1}^{q} w_{ij}\left(q_{ij} - \hat{q}_{ij}\right)^2,\ S_v = \sum_{j=1}^{q}\sum_{i=1}^{p-z}\left(\Delta_v^z q_{ij}\right)^2 \text{ and } M = F + \alpha \times S_v + \beta \times S_h$$

with α and β the smoothing control parameters. We have an explicit expression for the fitted values of q_{ij}. For the numerical application we choose $\alpha = \beta = .9$. We obtain the following[††]:



The adjustment looks very similar to the FNN mapping with 100 neurons. As an advantage for the W-H method one can observe that it is not necessary to have a special treatment for the boundary data. The choice of α and β is arbitrary and have to be validate by a back testing approach, which is outside the scope oh this paper.

5 Conclusion

Feedforward neural networks and radial-basis function neural networks seem to be promising tools to model experience table. If there is no exact method to choose the number of neurons, this difficulty is offset by the ease of implementation, particularly for RBN. The FNN proves to be a highly versatile model able to adjust difficult “shapes” of data viewed as a two dimensional map. The RBN adjustment is less easy, especially for abrupt boundaries, but the result is smoother. For the end-user, RBN have the advantage to allow a deterministic and fast training.

RBN are based on Gaussian (or symmetric) functions, which can explain the difficulty of adjustment on boundaries. We will test in a near future RBN like networks with Beta functions to improve its fidelity.

[††] The R code used here can be found at http://www.ressources-actuarielles.net/r

References

[1] BCAC (1993) Note sur le provisionnement en arrêt de travail.

[2] COX D.R. et OAKES D. (1984), *Analysis of survival data*, London, Edition Chapman and Hall.

[3] G. Cybenko (1989). Approximation by superpositions of a sigmoidal function, Mathematics of Control, Signals and Systems, Vol. 2, pp 303-314.

[4] DARPA Neural Network Study (U.S.) (1988) *DARPA Neural Network Study*. AFCEA Intl.

[5] DROESBEKE J.J, FICHET B, TASSI P., éditeurs (1989), *Analyse statistique des durées de vie: Modélisation et données censurées*, Economica.

[6] El HORR, R. (2010) Construction des tables d'expérience et étude de la sensibilité des données. Mémoire ISFA

[7] K. Funahashi (1989). On the approximate realization of continuous mappings by neural networks, Neural Networks, Vol. 2, pp183-192.

[8] KALBFLEISH J.D. et PRENTICE R.L. (1980), *The statistical analysis of failure time data*, New York: Wiley and Sons, Inc.

[9] KAPLAN E.L. et MEIER P. (1958), *Non parametric estimation from incomplete observations*, J. Amer. Statist. Assoc. 53, pp 457-481.

[10] Mathworks. Matlab Neural Network User's Guide, http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/index.html?/access/helpdesk/help/toolbox/nnet/

[11] PLANCHET, F., THEROND, P. (2006) Modèles de durée. Applications Actuarielles. Economica

[12] PLANCHET F., WINTER P. (2007) « L'utilisation des splines bidimensionnels pour l'estimation de lois de maintien en arrêt de travail », Bulletin Français d'actuariat, Vol. 7 n°13.

[13] XIE J., LIU C. (2001) Adjusted Kaplan-Meier Estimator and Log-rang test with inverse probability of treatment weighting for survival data, Statist. Med., vol. 00 :1-6