A NOVEL NEURAL NETWORK FOR SURVIVAL ANALYSIS

A. Eleuteri[1,2],R. Tagliaferri[3,4],L. Milano[2,6],S. De Placido[5],M. De Laurentiis[5]

Received March 26, 2003

Abstract. In this paper a feedforward neural network architecture to model survival probabilities is illustrated. The two main features of the network are that non-linearity are captured in the survival function, and the time variable is embedded in the model so it is able to extract its interactions with other system features. The model is described in a hierarchical Bayesian framework. Some experiments with synthetic and real world data show the capabilities of the model.

1 Introduction In literature we find many different modeling approaches to survival analysis. Conventional parametric models may involve too strict assumptions on the distributions of failure times and on the form of the influence of the system features on the survival time, assumptions which usually extremely simplify the experimental evidence, particularly in the case of medical data [Kalbfleisch et al., 1980]. In contrast, semi-parametric models do not make assumptions on the distributions of failures, but make instead assumptions on how the system features influence the survival time; furthermore, usually these models do not allow for direct estimation of survival times. Finally, non-parametric models only allow for a qualitative description of the data.

Neural networks have been recently used for survival analysis, for three surveys on the current use of neural networks we refer to [Ripley et al., 1998], [Schwarzer et al., 2000], [Eleuteri et al., 2002].

In this paper we describe a novel neural network architecture which overcomes all the limitations of currently available neural network models without making assumptions on the underlying survival distribution and which is proved to be flexible enough to model the most complex survival data. We describe our model in a Bayesian framework, which, as advocated by [Raftery et al., 1994] in the context of survival analysis, helps taking into account model uncertainty, which can help to improve the predictive performance of a model.

The remainder of the paper is organized as follows. In Section 2 we describe the fundamentals of survival analysis and pose the censoring problem. In Section 3 we describe some of the most used standard modeling techniques, which will be used in comparison to our model. In Section 4 we describe the architecture of our neural network and we define a hierarchical Bayesian description of the neural network and a Monte Carlo estimator of the survival function. In Section 5 some experimental results are shown, in comparison with some commonly used modeling techniques.

2 Survival and hazard functions Let $T(\mathbf{x})$ denote an absolutely continuous random variable (rv) describing the failure time of a system defined by a vector of features $\mathbf{x}$. If $F_T(t|\mathbf{x})$ is the cumulative distribution function for $T(\mathbf{x})$, then we can define the *survival*

2000 *Mathematics Subject Classification.* 62F15;62N05;68T05.
Key words and phrases. Survival analysis; Neural networks; Conditional probability estimation.

function:

$$S(t|\mathbf{x}) = P\{T(\mathbf{x}) > t\} = 1 - F_T(t|\mathbf{x}) = 1 - \int_0^t f_T(u|\mathbf{x})\mathrm{d}u \tag{1}$$

which is the probability that the failure occurs after time t.

The requirements for a function $S(t|\mathbf{x})$ to be a survival function are, uniformly w.r.t. $\mathbf{x}$:

1. $S(0|\mathbf{x}) = 1$ (there cannot be a failure before time 0),
2. $S(+\infty|\mathbf{x}) = 0$ (asymptotically all events realize),
3. $S(t|\mathbf{x})$ must be non-increasing as t increases.

From the definitions of survival and density of failure, we can derive the *hazard function* which gives the probability density of an event occuring around time t, given that it has not occured before t. It can be shown that the hazard has this form:

$$\lambda(t|\mathbf{x}) = \frac{f_T(t|\mathbf{x})}{S(t|\mathbf{x})} \ . \tag{2}$$

In many studies (e.g. in medical statistics or in quality control), we do not observe realizations of the rv $T(\mathbf{x})$. Rather, this variable is associated with some other rv $Y \sim r$ such that the observation is a realization of the rv $Z = q(T, Y)$. In this case we say that the observation of the time of the event is *censored*. If Y is independent from T, we say that censoring is *uninformative*.

In this paper we will assume *uniform right censoring*, i.e. $q \equiv \min(\cdot, \cdot)$ and Y uniformly distributed over an interval $[0, a]$.

Survival data can then be represented by triples of the form $(t, \mathbf{x}, y)$ where $\mathbf{x}$ is a vector of features defining the process, t is an observed time, and y is an indicator variable:

$$y = \begin{cases} 1 & \text{if } t \text{ is an event time} \\ 0 & \text{if } t \text{ is a censored time} \end{cases} \ . \tag{3}$$

It is possible to show [Eleuteri et al., 2002] that in the case of right censoring, the joint sample density of a survival process for a set of independent observations $D = \{(t_k, \mathbf{x}_k, y_k)\}$ can be written as:

$$L(D) \equiv \prod_k l(y_k|t_k, \mathbf{x}_k) = S(t_k|\mathbf{x}_k)^{1-y_k} f_T(t_k|\mathbf{x}_k)^{y_k} \ . \tag{4}$$

Note that censored data models can be seen as a particular class of missing data models in which the densities of interest are not sampled directly [Robert et al., 1999].

3 Models for Survival Analysis In the next sections we will describe some of the most commonly used models, both for homogeneous (time-only) and heterogeneous modeling.

3.1 The Kaplan-Meier non-parametric estimator The Kaplan-Meier (KM) estimator is a non-parametric maximum likelihood estimator. It is piecewise constant, and can be thought of as an empirical survival function for censored samples. It is only homogeneous.

Let k be the number of events in the sample, $t_1, t_2, \ldots, t_k$ the event times (supposed ordered), e_i the number of events at time t_i and r_i the number of times (events or censored) greater than or equal to t_i. The estimator is given by the formula:

$$S_{\mathrm{KM}}(t) = \prod_{i:t_i<t} \frac{r_i - e_i}{r_i} \ . \tag{5}$$

It should be noted that the estimator is noisy when the data are few, since it is piecewise constant. Therefore KM estimates from datasets sampled from the same distribution but with different number of samples can differ quite a lot.

Despite its restrictions, this is the most widely used tool for analyzing survival data, because its estimation is very fast and it allows qualitative inference on the data.

3.2 The Weibull homogeneous model The Weibull homogeneous model assumes a specific functional form for the density of event times, that is:

$$f_T(t) = \mathcal{W}e(a,b) \equiv abt^{b-1}\exp(-at^b)\mathbb{I}_{(0,+\infty)}(t) \tag{6}$$

where a and b are positive parameters and $\mathbb{I}(\cdot)$ is the set indicator function. This density implies that the hazard can be modelled as:

$$\lambda(t) = abt^{b-1} \ . \tag{7}$$

3.3 Proportional hazards model The most used survival specification which takes into account system features is to allow the hazard function to have the form:

$$\lambda(t|\mathbf{x}) = \lambda(t)\exp(\mathbf{w}\cdot\mathbf{x}) \tag{8}$$

where $\mathbf{w}$ are the parameters of the model, $\lambda(t)$ is some parameterised homogeneous hazard function. This is called a *proportional hazards* (PH) model. The main drawbacks of such model are assumptions on the form of the homogeneous hazard and on the interaction between the covariates and the hazard. Furthermore, a linearity interaction between the covariates is usually assumed, which, while simplifying the analysis and interpretation of the parameters, it also hampers the capabilities of the model. Despite this, it is one of the most used models in literature, and in the case of Weibull homogeneous hazard, it can also be seen as an *accelerated failure time* (AFT) model since the predictors act additively on the logarithm of failure time.

4 A neural network model In this section we define a neural network (NN) which, given a data set of system features and times, provides a model for the survival function (and implicitly, by suitable transforms, of the other functions of interest). Furthermore, a Bayesian description of the model is given.

4.1 Network architecture It is well known [Bishop, 1996] that a sigmoid is the appropriate activation function for the output of a NN to be interpretable as a probability. If $V(a) = (1+\exp(-a))^{-1}$ is such a sigmoid function (a is the net input to the activation), by taking into account eq.1 and using $S = 1 - V$ we can define the activation function of the NN as:

$$S(a) = \frac{1}{1+\exp(a)} \ . \tag{9}$$

This definition is not sufficient to define a survival function, since we must also enforce the requirements defined in the previous section as follows (for a more detailed discussion on these conditions, see [Eleuteri et al., 2002]):

1. $S(0|\mathbf{x}) = 1,\ S(+\infty|\mathbf{x}) = 0.$

   These requirements are satisfied if we choose one of the activation functions in the hidden layer to be of the form $h(t|w_{t_1}) \equiv \log(tw_{t_1})$. The only input feeding into this *time unit* through the weight w_{t_1} is the time input (neither biases nor other inputs feed this unit). Furthermore, the weights into and out of the hidden time unit must be non-negative.

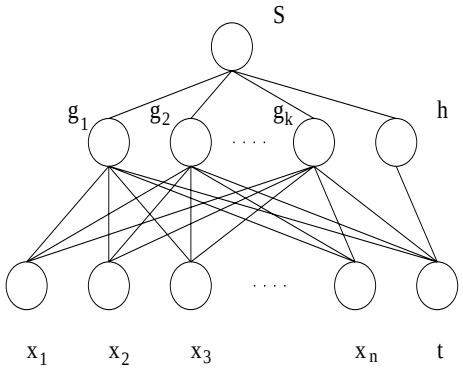

Figure 1: Architecture of a neural network for survival analysis.

2. $S(t|\mathbf{x})$ must be non-increasing as t increases.

   This requirement is satisfied if all hidden unit activations are monotonic non-decreasing and if all the weights on paths connecting the time input to the network output are non-negative.

In figure 1 the architecture of the SNN is shown (the bias connections are not shown).

We have thus defined a neural network model which describes a class of semi-parametric survival functions $S(t|\mathbf{x}, \mathbf{w})$, where $\mathbf{w}$ is the weights vector.

4.2 The Bayesian approach to modeling In the conventional maximum-likelihood approach to training, a single weight vector is found which minimizes an error function; in contrast, the Bayesian scheme considers a probability distribution over weights. We will build a Bayesian statistical model, which is composed by a parametric statistical model (the likelihood) and a prior distribution over the weights. For an in-depth discussion on Bayesian learning in the context of neural networks see [Neal, 1996].

The learning process in this case is described by a prior distribution $p(\mathbf{w})$ which is modified when we observe the data D through the likelihood $p(D|\mathbf{w})$. This process can be expressed by Bayes' theorem:

$$p(\mathbf{w}|D) = \frac{p(D|\mathbf{w})p(\mathbf{w})}{\int p(D|\mathbf{w})p(\mathbf{w})\mathrm{d}\mathbf{w}} \quad . \tag{10}$$

The likelihood is given in eq.4, in which S is the function realized by the network and f_T is given by the Jacobian of the network, which can be efficiently evaluated with a backpropagation procedure [Eleuteri et al., 2002].

The prior over weights should reflect any knowledge we have about the mapping we want to build. For the SNN model, we must then take into account the fact that we want a smooth mapping and that we have both constrained and unconstrained weights; furthermore, we should take into account the specialized rôle the weights have in the network depending on their position. We can thus define the following weight groups:

- Unconstrained weights:
  - a group for each of the covariate input, comprising the weights out of that input into the hidden squashing units;

  - one group for the biases feeding the hidden squashing units;
  - one group for the output bias.

- Constrained weights:
  - one group for the weights out of the input time unit feeding the hidden squashing units;
  - one group for the weight out of the input time unit feeding the hidden time unit;
  - one group for the weight out of the hidden time unit;
  - one group for the weights out of the hidden squashing units;

If n is the number of inputs then we have $n + 5$ groups.

We assign a distribution to each group, governed by its set of parameters (hyperparameters). Since the weights are a priori statistically independent, then the prior over the whole set of weights will be the product of the priors over each group.

For the unconstrained weights we can choose a Gaussian prior of the form:

$$p(\mathbf{w}_u|\boldsymbol{\alpha}_u) = \prod_{k \in \mathcal{G}_u} \left(\frac{\alpha_k}{2\pi}\right)^{W_k/2} \exp\left(-\frac{\alpha_k}{2} \sum_{i_k=1}^{W_k} w_{i_k}^2\right) \tag{11}$$

where $\mathbf{w}_u$ is the vector of unconstrained weights, $\mathcal{G}_u$ is the set of the unconstrained weight groups, W_k is the cardinality of the k-th group and $\boldsymbol{\alpha}_u$ is the vector of hyperparameters (inverse variances).

For all the constrained weights except those out of the hidden squashing units we choose an exponential prior:

$$p(\mathbf{w}_c^{(\mathrm{e})}|\boldsymbol{\alpha}_c^{(\mathrm{e})}) = \prod_{k \in \mathcal{G}_c^{(\mathrm{e})}} {\alpha_k}^{W_k} \exp\left(-\alpha_k \sum_{i_k=1}^{W_k} w_{i_k}\right) \mathbb{I}(\mathbf{w}_c^{(\mathrm{e})}) \tag{12}$$

where $\mathbf{w}_c^{(\mathrm{e})}$ is the vector of constrained weights, $\mathcal{G}_c^{(\mathrm{e})}$ is the set of the unconstrained weight groups, W_k is the cardinality of the k-th group, $\boldsymbol{\alpha}_c^{(\mathrm{e})}$ is the vector of hyperparameters (inverse means) and $\mathbb{I}(\cdot)$ is the indicator function over the positive half-space in which the weights are constrained.

For the constrained weights out of the hidden squashing units we choose a *stable* [Samorodnitsky et al., 1994] Lévy density:

$$p(\mathbf{w}_c^{(L)}|\gamma) = \left(\frac{\gamma}{2\pi}\right)^{N/2} \exp\left(-\sum_{i=1}^{N} \left(\frac{\gamma}{2w_i} + \frac{3}{2}\log w_i\right)\right) \mathbb{I}(\mathbf{w}_c^{(L)}) \ . \tag{13}$$

where γ is a scale parameter and N is the number of hidden squashing units.

For a complete description and motivation of the above choices, we refer to [Eleuteri et al., 2002].

4.3 Markov chain Monte Carlo methods for sampling the posterior The best prediction we can obtain given a new input and the observed data D can be written in the following way:

$$p(y|(\mathbf{x},t),D) = \int p(y,\mathbf{w},\boldsymbol{\alpha}|(\mathbf{x},t),D)\mathrm{d}\mathbf{w}\mathrm{d}\boldsymbol{\alpha} = \int p(y|(\mathbf{x},t),\mathbf{w})p(\mathbf{w},\boldsymbol{\alpha}|D)\mathrm{d}\mathbf{w}\mathrm{d}\boldsymbol{\alpha} \tag{14}$$

where $p(y|(\mathbf{x},t),\mathbf{w})$ is the predictive model of the network. This can be seen as the evaluation of the expectation of the network function with respect to the posterior distribution of the network weights. In our case the integral cannot be solved analytically, so we must use a numerical approximation scheme.

The posterior density in eq.10 can be efficiently sampled by using a Markov chain Monte Carlo (MCMC) method, which consists in generating a sequence of weight vectors, and then using it to approximate the integral in eq.14 with the following Monte Carlo estimator (M is the number of samples from the posterior):

$$p(y|(\mathbf{x},t),D) \approx \frac{1}{M}\sum_{i=1}^{M} p(y|(\mathbf{x},t),\mathbf{w}_i) \tag{15}$$

which is guaranteed to converge by the Ergodic Theorem to the true value of the integral as M goes to infinity [Meyn et al., 1993].

The MCMC method we used for generating the sequence of weight vectors is the Slice Sampling method. For a detailed discussion see [Neal, 2000].

4.4 Gibbs sampling As we have seen, the priors over the weights used in the inference of the posterior are themselves parametrized. Instead of setting the values of these hyperparameters to arbitrary values, we can make them part of the inference process, by sampling from eq.10. This can efficiently be done by using the Gibbs sampling process. In a first step the hyperparameters are kept constant, and we use an MCMC method to sample from eq.10. The second step consists in keeping the weights constant, while the hyperparameters are sampled from the following distribution:

$$p(\boldsymbol{\alpha}|\mathbf{w},D) = \frac{p(\boldsymbol{\alpha},\mathbf{w},D)}{p(\mathbf{w},D)} = \frac{p(D|\mathbf{w})p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha})}{p(D|\mathbf{w})p(\mathbf{w})} \propto p(\mathbf{w}|\boldsymbol{\alpha})p(\boldsymbol{\alpha}) \ . \tag{16}$$

This process can be done efficiently if we choose a hyperprior $p(\boldsymbol{\alpha})$ which is conjugate to the hyperposterior $p(\boldsymbol{\alpha}|\mathbf{w},D)$ (i.e. they both belong to the same family). For a detailed discussion on the topic of hierarchical Bayesian modeling we refer to [Robert, 2001].

It is possible to show that in the case of Gaussian, exponential and Lévy priors, by choosing gamma hyperprior distributions, the resulting hyperposteriors are gamma distributions [Eleuteri et al., 2002].

5 Experiments To assess the performance of our model, we tested it on synthetic homogeneous and real world heterogeneous data. All the software for the simulations has been written by the authors in the MATLAB language.

5.1 Synthetic homogeneous data In the first experiment the failure density is:

$$f_T(t) = \frac{(t-2)^4}{4}\exp\frac{1}{20}\left(-32-(t-2)^5\right) \ . \tag{17}$$

The density was sampled with the MCMC slice sampling algorithm; 5000 samples were generated, with about 13% uniformly right censored data in the interval $[0,10]$. The data were then uniformly split into two disjoint data sets: a training set of 3000 samples and a testing set of 2000 samples (with the same percentage of censored data in each set). Three models were fitted to the data: a standard Weibull model, a KM non-parametric estimator and a SNN model with ten hidden units.

The SNN model was trained using the MCMC slice sampling algorithm with Gibbs sampling of the hyperparameters. The first 100 samples were omitted from the chain, to

allow its burn-in. Then, 100 samples were generated, and the last 50 were used to obtain the Monte Carlo estimator of the SNN output, based on an inspection of the energies of the chain. Note that in a real use of the algorithm, a simple inspection of the energies might not be sufficient to assess convergence; instead, a formal test of convergence, like the *Estimated Potential Scale Reduction* diagnostic ([Robert et al., 1999], [Gelman et al., 1992]) should be used. The Weibull model was trained by standard Maximum Likelihood techniques.

As can be seen in figure 2, the Weibull model cannot fit the data at all. The SNN model of the survival function is similar to the KM estimator of the data. Note that two KM estimator curves have been shown, one for the complete data set, the other for the test set only. This is important because the KM estimator is noisy when the data are few; however it asymptotically converges to the true survival function, in absence of censoring.

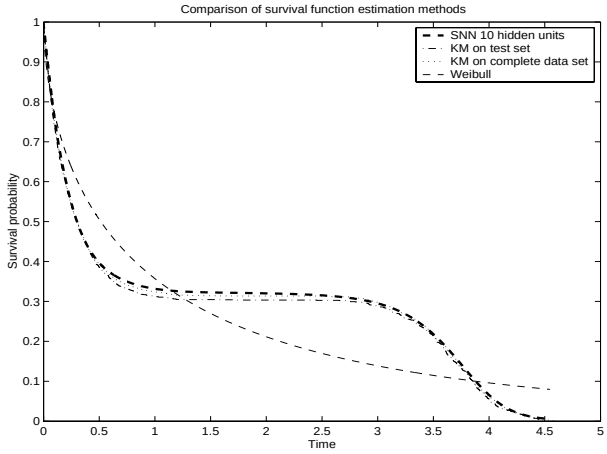

Figure 2: Weibull, KM and SNN models of a synthetic survival function (evaluated on the test set).

5.2 Real heterogeneous data These data are taken from trials of chemotherapy for stage B/C colon cancer[1]. It is composed of 1776 samples, with 12 covariates. The dataset was split into a training set of 1076 samples, and a test set of 700 samples. About 49% of the data are censored, which makes the analysis of the data a complex task. Two models were fitted: a Weibull PH model and a SNN model with 30 hidden units. KM estimates were then evaluated for each model.

The SNN model was trained with the MCMC slice sampling algorithm with Gibbs sampling of the hyperparameters. As in the previous experiment, the first 100 samples were omitted from the chain. Then, 240 samples were generated and the last 60 samples were used to obtain the Monte Carlo estimator of the SNN output, based on an inspection of the energies of the chain. The Weibull PH model was trained by standard Maximum Likelihood techniques.

The test data was stratified in three quantiles based on the survival estimates of the models at $t = 220$ weeks. Then, the mean (w.r.t. the covariates) survival probabilities were estimated, togheter with KM estimates of the quantiles induced by the models. If the output of the model and KM estimates are similar, then the model captures the distribution of the data. As can be seen in fig.3 and fig.4, the SNN model has good performances, while the Weibull PH model cannot fit the data at all.

[1] The data are distributed with the freeware R statistical software, available at http://www.r-project.org

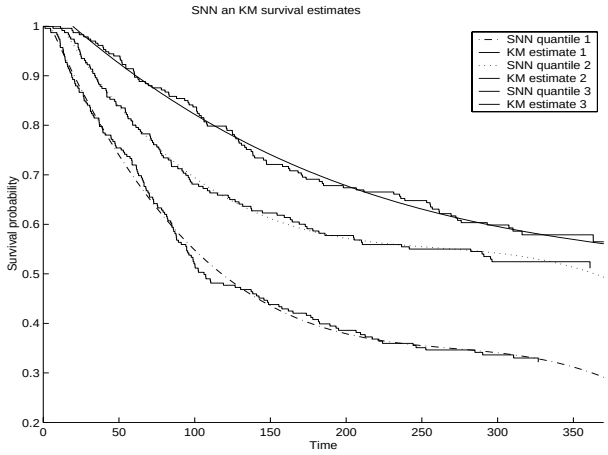

Figure 3: SNN model of a real survival function with covariates (evaluated on the test set).

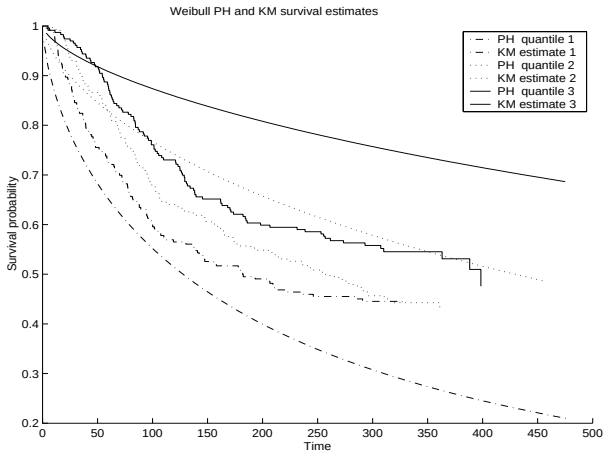

Figure 4: Weibull PH model of a real survival function with covariates (evaluated on the test set).

6 Discussion and perspectives In this paper we have described a novel NN architecture that addresses survival analysis in a neural computation paradigm.

Starting from well defined mathematical requirements on the survival function, we have defined a feedforward neural network which, by a careful choice of the architecture and the activation functions for the neurons, satisfies those requirements.

Experiments on complex synthetic and real survival data demonstrate that the SNN model can approximate both homogeneous and heterogeneous survival functions, and its performances are better than those of the most used model used in literature, namely the Weibull Proportional Hazards and Accelerated Failure Time model.

Acknowledgments This work was partially supported by MIUR Progetto di Rilevanza Nazionale, MIUR-CNR, AIRC.

References

[Bishop] C. M. Bishop. Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1996)

[Eleuteri et al.] A. Eleuteri, R. Tagliaferri, L. Milano, M. De Laurentiis. A novel neural network–based survival analysis model. Accepted for publication on Neural Networks.

[Gelman et al.] A. Gelman, D. B. Rubin. Inference from iterative simulation using multiple sequences (with discussion). Statist. sci. **7** (1992) 457–511

[Kalbfleisch et al.] J.D. Kalbfleisch, R. L. Prentice. The statistical analysis of failure time data. Wiley, New York (1980)

[Meyn et al.] S. P. Meyn, R. L. Tweedie. Markov Chains and Stochastic Stability. Springer-Verlag, London (1993)

[Neal] R. M. Neal. Bayesian Learning for Neural Networks. Springer, New York (1996)

[Neal] R. M. Neal. Slice Sampling. Technical Report No. 2005, Department of Statistics, University of Toronto (2000).

[Raftery et al.] A. E. Raftery, D. Madigan, C. T. Volinsky. Accounting for Model Uncertainty in Survival Analysis Improves Predictive Performance. Bayesian Statistics **5** (1994) 323–349

[Ripley et al.] B. D. Ripley, R. M. Ripley. Neural Networks as Statistical Methods in Survival Analysis. Artificial Neural Networks: Prospects for Medicine (R. Dybowsky and V. Gant eds.), Landes Biosciences Publishers (1998)

[Robert et al.] C. P. Robert, G. Casella. Markov Chain Monte Carlo Methods. Springer, New York (1999)

[Robert] C. P. Robert. The Bayesian Choice, Second Edition. Springer, New York (2001)

[Samorodnitsky et al.] G. Samorodnitsky, M. S. Taqqu. Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance. Chapman & Hall, New York (1994).

[Schwarzer et al.] G. Schwarzer, W. Vach, M. Schumacher. On the misuses of artificial neural networks for prognostic and diagnostic classification in oncology. Statistics in medicine **19** (2000) 541–561

[1] Dip. di Matematica ed Applicazioni "R.Caccioppoli", Università di Napoli
[2] INFN sezione di Napoli, e-mail: eleuteri@na.infn.it
[3] Dip. di Matematica ed Informatica, Università di Salerno
[4] INFM unità di Salerno
[5] Dip. di Endocrinologia ed Oncologia Mol. e Clinica, Università di Napoli
[6] Dip. di Fisica, Università di Napoli