Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Economics and Finance xx (2013) 1–11

Procedia
Economics and Finance

www.elsevier.com/locate/procedia

Emerging Markets Queries in Finance and Business 2013 – EMQFB2013

An Artificial Neural Network-based technique for on-line hotel booking

Marco Corazza[a, b*], Giovanni Fasano[c, d], Francesco Mason[c]

[a]*Ca' Foscari University of Venice – Department of Economics, Cannaregio n. 873, Venezia 30121, Italy*
[b]*Advanced School of Economics, Cannaregio n. 873, Venezia 30121, Italy*
[c]*Ca' Foscari University of Venice – Department of Management, Cannaregio n. 873, Venezia 30121, Italy*
[d]*National Research Council – Maritime Research Centre (CNR-INSEAN), Via di Vallerano n. 139, Roma 00128, Italy*

Abstract

In this paper the use of Artificial Neural Networks (ANNs) in on-line booking for hotel industry is investigated. The paper details the description, the modeling and the resolution technique of on-line booking. The latter problem is modeled using the paradigms of machine learning, in place of standard '*If-Then-Else*' chains of conditional rules. In particular, a supervised three layers MultiLayer Perceptron (MLP) ANN is adopted, which is trained using information from previous customers' reservations. Performances of our ANNs are analyzed: they behave in a quite satisfactory way in managing the (simulated) booking service in a hotel. The customer requires single or double rooms, while the system gives as a reply the confirmation of the required services, if available. Moreover, in the case rooms or services are not at disposal, we highlight that using our approach the system proposes alternative accommodations (from two days in advance to two days later with respect to the requested day). Numerical results are given, where the effectiveness of the proposed approach is critically analyzed. Finally, we outline guidelines for future research.

Keywords: On-line booking; hotel reservation; machine learning; supervised multilayer perceptron networks.

1. Introduction

Aim of this paper is presenting some results about the use of *Artificial Neural Networks* (ANNs), as automatic tools for the management of on-line booking of hotels. In particular, our objective is to analyze at what extent some classes of ANNs - namely supervised *MultiLayer Perceptron* (MLP) networks - can help, and in the limit case can replace, employees of booking offices in hotels, in order to manage the operations of on-line booking of rooms and services. This is a very challenging aspect of the use of sophisticated computer science tools in tourism industry. We will show that our proposal may be at least a support system to decision makers, who can directly gain advantage of the ANNs framework we analyze, using a reasonable amount of computational resources.

* Corresponding author. Tel.: +11-39-41-234-6921; fax: +11-39-41-234-7444.
E-mail address: E-mail address: corazza@unive.it

The problem we focus on regards the analysis, the implementation and the training of an ANN, which is able to "learn" from past data (collected by previous customers' requests), and which can provide information for new customers' booking. In particular, we want to assess a tool with the capability of offering a set of services and facilities. Furthermore, we claim that our proposal can often represent a reliable support, which requires neither too long training time nor expensive devices to work.

In this paper we also present extensive numerical experiences, as well as the motivations behind them, showing that the tool we propose appears often of interest. However, as expected, some limits of our tool may arise, regarding its performance: we will detail the latter and warn the readers about the harmful or uncritical application of our proposal.

The remainder of the paper is organized as follows. In the next Section we describe some relevant aspects and remarkable issues of on-line booking. We describe how the latter aspects can be properly taken into account, in order to support the solution of the on-line booking problem for hotels. Then, in Section 3 some basic elements of machine learning and supervised MLP networks are given. Finally, in Section 4 the numerical experiments we have carried out are illustrated, as well as the final results we obtained.

2. Basics on on-line booking for hotel industry

In the last two decades, the development of computer science tools, including software, hardware and information exchange paradigms, yielded two synergic phenomena for our purposes: the increasing computational power, as well as the increasing capacity of data storing, and the rise of the WEB as an environment in which market operators may freely interact.

On the other hand, Information Security allows firms to focus on fast interactive management of market relationships. The firms can indeed use information technology with the purpose to behave on the market in an active fashion, managing directly communication, proposals, contracts which were left beforehand in the range of each agent's job.

The WEB is now a keyword in the business horizon of the firms. Of course, this implies that several problems may arise, in order to handle the interactions of firms accessing the WEB. As a consequence, many services and WEB facilities have increased their importance, and have been charged with the burden of managing the on-line market. Among them, the *Customer Relationship Management* (CRM) has the significant role of providing WEB instruments for the firms, having in mind their specific claims and objectives.

Broadly speaking, in the research area of tools for the WEB, which are oriented to ease and to customize the interaction between the firms and the on-line market, we basically identify two renowned approaches. On one hand we have non-interactive instruments, mainly devoted to information and advertising of the proposals of each firm. On the other hand, more complex tools which can yield a stronger interaction among partners are widely spreading. The latter tools are pretty often useful to finalize the interaction towards a service delivery or a sale.

In any case, the WEB spurs competitors to use and to share simple and appealing interfaces: this implies that the WEB must be suited both to customers necessities and to the needs of the firms. Each interface must guarantee bidirectional and clear information exchange and spreading, allowing several different operations (e.g. qualification, communication, confirmation of an order, as well as modification or cancelation of it). The latter considerations inevitably apply to on-line traders and tour operators.

In the hotel industry, the new possibilities introduced above are much relevant. Similarly to CRM for hotel industry, we have two prevalent approaches in the literature of on-line hotel booking. We have, on one hand, instruments which simply focus on advertising the hotel characteristics on the WEB. On the other hand, there are more specific instruments devoted to the on-line booking, where "booking" is intended as including the whole service and policies offered by the hotel to the customers. The tools offered by the WEB in the second approach are undoubtedly more complete and sophisticated: this is because nowadays every hotel tends to provide a variety of services to customers, which are usually not limited to the simple reservation. As a consequence, on-line booking has to deal with strategic instruments, in order to pursue two relevant aspects of the on-line market: customer loyalty and customer take over. Thus, the software used to support customers booking must be also a "guide", in order to route customer's preferences since the early phase in which he/she states his/her own preliminary requirements or chooses services.

Many factors contribute to the complexity of the resulting software. From an operational point of view, it must ensure robustness and continuity of the service, a multi-language environment and safety of transactions. On the other

hand, articulated *Definition Paths* (DPs)[1] are required, based on the characterization of the customer. This can be done through suitable on-line questions (posed to the customer) or through the analysis of the customers in the hotel data base (or in the data base of the hotel cluster). Indeed, the analysis of historical data may suggest some recursive preferences of customers, possibly related to the venue of the hotel. Then, the latter process of collecting data can yield the proposal of specific services, guiding the customer in the choice. Therefore, the ideal on-line tool which supports the user must progressively update its data base, paying a specific attention to reference both the preferences of customers and the offered services.

In order to detail more accurately the DPs, observe that they are expected to include:

- The time periods required for booking;
- The final price list;
- The characteristics of the customers, i.e. their nationality, social status, ...;
- The motivation of customers trip, i.e. business trip, holiday trip, ...

Moreover, to complete the on-line booking process started with a DP, the latter is usually followed by:

- The communication of acceptance, or of alternative proposals;
- The specification of the selected services.

The specification of the selected services is often a non-trivial result, since services are most of the times interrelated. Thus, one service can imply or exclude another one, on the basis of rules system, which can be strongly structured. In a dynamic framework, due to the time window chosen by the customer or to some other factors, more complex situations can arise. For example, prices can significantly vary, following the seasonal trend, the number of free chambers, or the competition among hotels. In addition, also cultural events can affect prices in a specific time period. To sum up, automatic procedures and instruments are definitely needed, which dynamically allow the interaction of customers and hotels, in a fast changing scenario.

The building and the maintenance of DPs, within automatic procedures devoted to on-line booking, should consider the following two problems:

- The used software can hardly manage all the *If-Then-Else* boolean rules which are necessary to build the DPs;
- The difficulties to update the rules, as well as the introduction of new ones, claims for skilled and specialized operators, and represents a time-consuming operation.

In turn, the costs of maintenance of the software for the on-line booking can often prohibitively increase. Moreover, difficulties may arise in order to respect deadlines. The investigation of how to overcome the latter drawbacks is among the objectives of this paper.

3. Basics on machine learning

In order to partially clarify the issue in the last paragraph of the previous section, we introduce here some preliminary concepts regarding *Machine Learning*, along with some of its possible applications. Machine learning involves procedures and techniques, which are naturally related to the capabilities of analysis and data processing of human beings. As a consequence, machine learning is also strictly related to adaptive behaviors. Indeed, daily experience suggests that all the living organisms interact with the environment, both exchanging information and adaptively modifying their reactions. The latter facts, not to mention the huge literature about the meaning of "intelligence", indicates that a formal assessment of machine learning paradigms is mandatory, if technical achievements are sought.

On this guideline, ANNs, regardless of their specific approach, attempt to emulate the cognitive capabilities and processing models of developed individuals. In particular, neural networks are often based on a system of elementary entities and rules, whose main purpose is that of emulating the human brain paradigms. To our knowledge, the first theoretical results from emulating the elements of the human brain, are due to W.S. McCulloch and W.Pitts (1943), and F.Rosenblatt (1958-1960) (see [1]), who tried to model the basic behavior of the *neurons*. They also showed that modeling the neuron as the input/output function

$$y_i(x_i) = sgn(w_i^T x_i - \theta_i), \tag{1}$$

[1] We call "*Definition Path*" the set of questions we ask the customer, in order to carefully assess his/her booking preferences (location and services).

where $x_i \in \mathbb{R}^n$ is the *input* vector, $y_i \in \mathbb{R}$ is the output, $w_i \in \mathbb{R}^n$ is a vector of *weights*, $\theta_i \in \mathbb{R}$ is a real *threshold*, then all the standard boolean functions (i.e. AND, OR, NOT, NAND, NOR) may be modeled by combining a set of functions like (1), where the parameters w_i and θ_i are suitably chosen (see for instance [2, 3, 4] for the algorithm of *perceptron*).

3.1. Paradigms of machine learning

In this section we briefly describe guidelines for a taxonomy of models and methods of machine learning. To this purpose we represent the overall ANN as

$$y = \mathscr{F}(x), \tag{2}$$

where $x \in \mathbb{R}^n$ is the input vector, $y \in \mathbb{R}^m$ is the output vector and $\mathscr{F}(\cdot)$ contains a set of neurons like (1).

The ANN will be trained when the parameters w_i and θ_i are assessed, according with the chosen learning mechanism. Observe that in general the function $sgn(\cdot)$ in (1) may be replaced by several other functions (e.g. the Heavyside function, the sigmoidal function,...), which are usually addressed as the *activation functions* of the neuron. The latter fact yields different properties of the overall resulting ANN (see [3]). Basically, the learning processes used to set the parameters w_i and θ_i of each neuron may be classified as follows:

- *Supervised learning*: where the ANN (2) is trained using *ex ante* a sequence of ordered input/output pairs, indicated as $\{(x_\ell, y_\ell)\}$;
- *Unsupervised learning*: where the ANN (2) is trained using *ex ante* only a sequence of inputs, indicated as $\{x_\ell\}$.

The first learning paradigm is often addressed as *learning by teacher* (like students who learn each x_ℓ and are *corrected* by the teacher who suggests y_ℓ). Also observe that supervised learning is strongly advisable when $\{x_\ell\}$ and $\{y_\ell\}$ are both available. Another important learning classification for ANNs is given by:

- *Batch learning*: where the sequences $\{(x_\ell, y_\ell)\}$ or $\{x_\ell\}$ are entirely provided before the learning process starts;
- *Online learning*: where the data $(x_1, y_1), (x_2, y_2), \ldots$ or $x_1, x_2, \ldots$ are progressively, i.e. *online*, provided during the learning process.

When the data for learning are all available and are not too many (with respect to the storage or the computation involved by the learning process), then batch learning should be preferable (it will be indeed our choice as described in Section 4.2).

On the other hand, the structure of the ANNs may strongly affect the learning process. In particular, since $\mathscr{F}(\cdot)$ is a collection of connected neurons we can assess the ANN after choosing:

1. The nature of neurons (for instance using the form (1)), which can be:
   - *Static*: if the input/output relation of each neuron *does not* depend on time;
   - *Dynamic*: if the input/output relation of each neuron *does* depend on time;
2. The inner parameters of the neurons and the parameters associated with the connections among the neurons;
3. The architecture of the ANN (e.g. double-layer, multilayer, ...).

Finally, it is worth mentioning that in an ANN the output of the network may be suitably recombined with the input, in order to give more generality to the learning process. This implies that if the ANN is represented by a graph, where the neurons are the nodes and the connections among them are the arcs, we can distinguish between:

- *Feedforward ANN*: represented by an acyclic graph;
- *Feedback ANN*: represented by a graph containing cycles.

The taxonomy in this section is necessary in order to motivate our choice to consider a specific ANN for solving our online-booking problem for hotels, described in Section 4.1. Thus, observe that all the data for training are available from previous online-booking of customers, and can be stored. Moreover, considering that we are concerned with a single period analysis, we are going to use in Section 4.2 an MLP feedback ANN (see also [5]), which is trained by a supervised learning process. In the next section we specify the two main purposes that our ANN is asked to pursue: the latter will further motivate our choice for an MLP feedback ANN, trained by a supervised process.

3.2. Practical use of ANNs

ANNs are commonly designed to automatically solve some categories of practical problems, using the data related to those problems. We briefly summarize here three relevant classes of problems whose solution often involves ANNs.

Pattern Recognition, where the training data are suitably classified into classes, so that possible new data will be assigned to the correct classes [1, 6] (several problems of *data clustering* belong to the latter category).

Function Approximation. Suppose the set $I = \{(x_i, y_i) : x_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}^m,\ i = 1, \ldots, N\}$ is given, and there exists the function $y = f(x)$ such that the following interpolation conditions hold

$$y_i = f(x_i),\ i = 1, \ldots, N. \tag{3}$$

We want to build the ANN $y = \mathscr{F}(x)$ such that for a given $\varepsilon > 0$

$$\| f(x) - \mathscr{F}(x) \|_* < \varepsilon, \qquad \forall x \in \mathbb{R}^n, \tag{4}$$

i.e. the ANN $y = \mathscr{F}(x)$ should be an approximation of $f(x)$ over $\mathbb{R}^n$, in the sense specified by the norm $\| \cdot \|_*$. Of course, the function approximation problem (4) admits in general more than one solution, and may be naturally tackled using a supervised approach (see also [7]).

Regression problem. Given the set $I = \{(x_i, y_i) : x_i \in \mathbb{R}^n,\ y_i \in \mathbb{R}^m,\ i = 1, \ldots, N\}$, we want to build the ANN $F : W \times \mathbb{R}^n \to \mathbb{R}^m$ such that $y = F(w, x)$, where W is a set of possible parameters. Unlike the case of function approximation, here we want to find the set of parameters $\bar{w}$ such that $F(\bar{w}, x)$ is the regression function

$$F(\bar{w}, x) = \int_{-\infty}^{+\infty} y \cdot p_{Y/X}(y/x) dy,$$

where $p_{Y/X}(y/x)$ is the conditional probability density function of variable Y given X, associated with the set I. As well known (see [8, 9]), $F(\bar{w}, x)$ solves

$$\min_w \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} [y - F(w, x)]^2 p_{X,Y}(x, y) dx dy,$$

and in general $p_{X,Y}(x, y)$ is not *a priori* known.

As a consequence of the classification above, the training of an ANN, i.e. the assessment of its parameters, must agree with one of the three categories in this section.

Moreover, note that the ANN $\mathscr{F}(x)$ which satisfies (4), in general does not satisfy the interpolation condition (3). The latter issue is much relevant for the ANNs we will adopt, and as showed in Section 4.2, it strongly affects both the training process and the effectiveness of the final ANN.

We complete this Section with a brief consideration on the complexity of ANNs. On this purpose, for the sake of completeness we should analyze the essential concept of VC-dimension (Vapnik-Cervonenkis-dimension) for ANNs. However, the latter concept is not among the main issues of this paper (see [8, 3] for an extensive description), thus we simply report that the VC-dimension of an ANN is a measure of the capability of that ANN to discriminate among the pairs (x_i, y_i) of input/output in the set I. Intuitively speaking, this means that a larger number of data will in general require a more complex structure of the ANN, i.e. a network where a larger number of parameters (degrees of freedom) w must be included. Thus, it is possible to assess in general the VC-dimension of each ANN in terms of the number of parameters of the ANN. On this guideline we have the following result [3], which gives an indication in order to set the architecture of an ANN.

Proposition 3.1 *Let $F(w, x)$ be an ANN, where $w, x \in \mathbb{R}^n$ and w is a vector of parameters. Suppose $F(w, x)$ is built using neurons like* (1)*. Then, the VC-dimension of $F(w, x)$ is $O(n \log n)$.*

3.3. The Multi-Layer Perceptron (MLP) ANNs we used

MLP ANNs use the functional expression (1), or similar, of the neurons, and are organized as a sequence of layers. The figure 1.(a) sketches the basic structure of a feedforward MLP as a graph, where the neurons are the

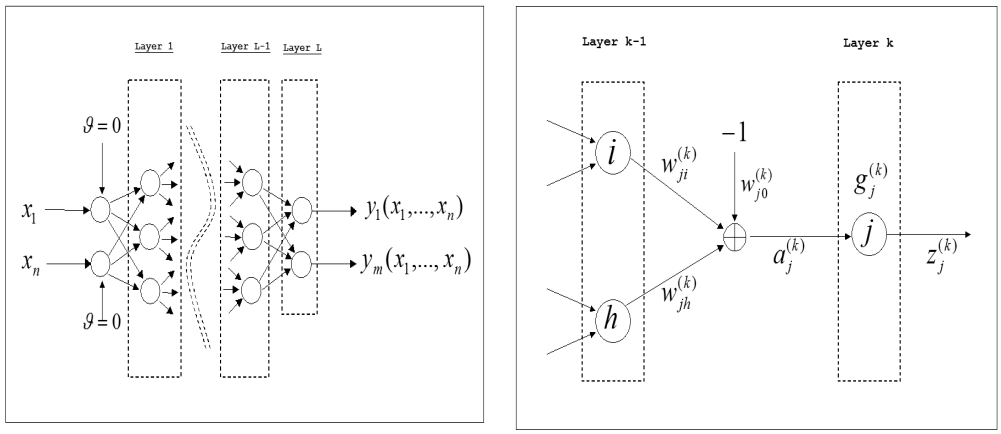

Fig. 1. (a) Structure of a supervised MLP ANN. (b) Meaning of the expression $a_j^{(k)} = -w_{j0}^{(k)} + \sum_{i=1}^{n^{(k-1)}} w_{ji}^{(k)} z_i^{(k-1)}$.

nodes (circles) and the connections among neurons are the arcs. Now we briefly describe the latter structure and the problems involved in training an MLP.

As evident from figure 1.(a) an MLP is organized into layers, each including a number of neurons, so that a more complex MLP will possibly contain a larger number of layers and/or a larger number of neurons each layer. Also observe that the standard architecture of an MLP requires that if the node v_i is connected with the node v_j, then v_i and v_j must belong to different layers. Moreover, in an MLP with L layers the input and output layers have a special structure. Indeed, the threshold θ in the neurons of the input layer is zero, while the output of the neurons in the output layer (i.e. the L-th layer) coincides with the output of the MLP. As regards the figures 1.(a) and 1.(b), the following comments hold:

- Between the i-th neuron in the $(k-1)$-th layer and the j-th neuron in the k-th layer there is an arc, with the associated *weight* $w_{ji}^{(k)} \in \mathbb{R}$;
- At the j-th neuron in the k-th layer we associate the activation function $g_j^{(k)}(\cdot)$, such that

$$z_j^{(k)} = g_j^{(k)}\left(a_j^{(k)}\right),$$

where $a_j^{(k)} = -w_{j0}^{(k)} + \sum_{i=1}^{n^{(k-1)}} w_{ji}^{(k)} z_i^{(k-1)}$, $w_{j0}^{(k)}$ is the threshold of the j-th neuron in the k-th layer, $z_j^{(k)}$ is the output of the j-th neuron in the k-th layer, and $n^{(k-1)}$ is the number of neurons in the $(k-1)$-th layer.

For the sake of clarity we report that using the MLP $F(w,x)$, with just two layers, if the activation function $g(\cdot)$ is not a polynomial, then any continuous function $h(x)$, $h : \mathbb{R}^n \to \mathbb{R}^m$, can be approximated with $F(w,x)$, over any compact set $\Omega \subset \mathbb{R}^n$. In other words, we have (see also [10, 7, 11, 3])

$$\sup_{x \in \Omega} \|h(x) - F(w,x)\|_* < \varepsilon, \ \varepsilon > 0,$$

for a choice of the vector $w \in \mathbb{R}^{n^{(k-1)}+1}$.

4. Our case-study: presentation and numerical results

In this section we first give an overview of our on-line hotel booking problem, then we describe the ANN-based approach we implemented in order to tackle this problem. Finally, we report the results obtained by the application of our approach.

4.1. Our on-line hotel booking problem

The on-line hotel booking problem we studied can be briefly described using the following items.

- We start by considering a given hotel characterized by a prefixed structure in terms of typology of rooms (for instance: single, double, ...), number of rooms for each typology, other services (for instance: availability of meeting rooms, of car parking, ...);
- At current time t a possible customer asks the hotel the current or future availability of some kind of services (for instance: a double room and a car parking at time $t+5$)[2];
- At the same time t an operator (typically a *human* one) checks for the availability of the requested service;
- If such a service is satisfiable, then the operator gives a positive answer to the customer; otherwise the operator provides the customer with an alternative solution (for instance: two single rooms and a car parking at time $t+5$, a double room without a car parking at time $t+4$, ...).

The last item contains all the elements of difficulty of the problem we are dealing with. In fact, the following considerations hold.

- First, in order to provide an automatic answer to the customer's request a suitable database is necessary (but a wide description of the latter issue goes beyond the purpose of this paper).
- Second, if the request of the customer is satisfiable, then *a solution* has to be searched for by interrogating the database. Generally, this is an hard task considering the combinatorial nature of the searching problem. In order to exemplify this point, let we consider the case one-single-room-and-a-car-parking for a hotel having s free single rooms and p free car parking: the number of possible solutions might be $s \cdot p$.
- Third, if the request of the customer is not satisfiable, then *one or more alternative solutions* have to be searched for by interrogating the database. Generally, this is again an hard task given the combinatorial nature of the searching problem. In order to exemplify this point, it is enough to consider the case two-single-rooms-and-a-double-one-instead-of-a-triple-room, for an hotel having s free single rooms and d free double rooms: the number of possible solutions should be $s(s-1)d$.

Therefore, it is evident that, yet in case of small or medium hotel, a human operator or a standard automatized system should not be able to suitably interrogate the database in a *commercially reasonable time*, being this due to the intrinsic difficulty[3] of the search problem itself.

Given these premises, the on-line hotel booking problem we have dealt with may be formulated as follows. Instead of performing a more or less refined searching algorithm (pursuing the fulfillment of customer's requests by an *If-Then-Else* procedure), we can address a suitable function (if any) that receives as inputs the state of the hotel (for instance: the current number of free single/double rooms, the number of free single/double rooms tomorrow, ...) along with the requests of the customer, and provides as output a solution (if any), or alternative solutions if the requests are not satisfiable.

Obviously, the functional form (say $f(x)$) of such a function and the value of its coefficients strongly depend on the hotel structure. So, a priori the latter function may be not easily specifiable. Because of that, we decided to use MLP ANNs as "inferential tool", considering their capability to be *universal* approximators of unknown functions, mapping known input datasets into known output datasets. Of course other choices could be considered, e.g. the Radial Basis Functions (RBFs) may have represented fruitful alternative supervised ANNs.

4.2. The MLP ANN-based approach

As detailed in the previous section, the performances of an MLP ANN may be strongly affected by its architectural structure (typically: the number of hidden layers and the number of artificial neurons for each of these layers) and by the considered learning algorithm. In its turn, the choice of the architectural structure and the learning algorithm depend on the pursued targets. Recalling what was reported in Section 3.2, with regard to the practical uses of ANNs, we may consider to build our ANN pursuing two mutually exclusive targets:

[2] The unit measure of time we consider is the day.

[3] Observe that if the customer has m specific requests, each request R_i, $i = 1, \ldots, m$, can be TRUE/FALSE depending on the fact that it can be satisfied or not. Thus, the overall request of the customer reduces to a satisfiability problem (known in the literature as the propositional logic problem SAT), which is NP-complete (see [12]), i.e. it cannot be solved in a polynomial computational time with the input string.

- *Aapproximating* the searched for function $y = f(x)$ by means of a suitable ANN (say $y = \mathscr{F}(x)$);
- *Applying regression* to the searched for function $y = f(x)$, yielding the appropriate function $y = F(w,x)$.

Roughly speaking, the main difference between the two approaches consists in the following fact. With "function approximation" we mean that $\mathscr{F}(x)$ has to exploit the generalization capabilities of ANNs (therefore, $\mathscr{F}(x)$ is expected to be substantially based on small- or medium-sized MLPs)[4]. On the other hand, with "function regression" here we mean that $F(w,x)$ mainly attempts to consider the interpolation properties of ANNs (therefore, $F(w,x)$ is expected to be essentially based on medium- and large-sized MLPs).

4.3. Numerical results

The first necessary step to implement any supervised MLP ANN consists in building a suitable dataset, collecting both the inputs and the outputs into pairs. Furthermore, in order to assure our experimentation as reliable as possible, we need possibly a large dataset. So, in accordance with the indication of north-eastern Italian tourism experts, we randomly generated a significant database of simulated pairs (x_i, y_i), with $i = 1, \ldots, 50000$, where $x_i \in \mathbb{R}^n$ represents the i-th vector of the inputs and $y_i \in \mathbb{R}^m$ represents the i-th vector of the outputs.

As far as the input vector structure is concerned, each input vector x_i contains information regarding:

- The request of the customer, namely: the availability of some kind of services in $t + \Delta t$, with $\Delta t = 0, 1, \ldots$, and the number of days in advance, with respect to $t + \Delta t$, this request has been received by the hotel;
- The state of the hotel as described in Section 4.1. In particular, we have taken into account an hypothetical small-medium sized hotel having only single and double rooms; further, in case the customer's request is not satisfiable, alternative solutions (if any) are provided, for at most two days before $t + \Delta t$ and two days after $t + \Delta t$.

Therefore, on the overall $n = 13$ entries of the input vector have been used.

As far as the input vector structure is concerned, in order to avoid as much as possible the well known "curse of dimensionality" phenomenon (see for instance [1]), we decided to use an univariate discrete output. In particular, the output takes values in the set

$$\{0, 1000, 2000, \ldots, 7000, 8000, 9000\},$$

depending on the fact that the customer's request is satisfiable ($y = 0$), the customer's request is not satisfiable but an alternative solution exists in $t + \Delta t - 1$ ($y = 1000$) or exists in $t + \Delta t + 1$ ($y = 2000$) and so on, up to "no alternative solutions exist" ($y = 9000$). Notice that the output values are so spread in order to facilitate the learning[5].

At this point, as stated above, we have randomly generated 50000 significant simulated pairs (x_i, y_i). However, after a careful glance of this dataset, it appears that the pairs are not uniformly distributed with respect to the output value (see table 1). This is an indirect confirmation of the complex nature of the on-line hotel booking problem we are considering. Of course, such a non-uniform distribution could make the supervised learning process of our ANN (particularly) difficult, since some output values are not as frequent as others.

Table 1. Percentage distribution of the pairs with respect to the output value.

Output values	0	1000	2000	3000	4000	5000	6000	7000	8000	9000
Pair percentages	19.60%	3.52%	3.37%	5.75%	1.51%	3.37%	5.24%	3.91%	7.89%	45.81%

As premised in Section 4.2, in the second step we dealt with the implementation of small/medium sized ANN MLPs, in order to approximate the searched for unknown function. The guidelines we essentially followed for the design of these ANN MLPs were:

- The use of one or two – and no more – hidden layers. As known, an MLP with an hidden layer is able to linearly separate the output space, and an MLP with two hidden layers is in principle able to non-linearly separate the same space;
- The use of a low number of artificial neurons for each of the considered hidden layer(s);
- The use of standard sigmoid squashing function for the artificial neurons belonging to the hidden layer(s), and the use of the linear squashing function for the output artificial neuron;

[4] The adjective "sized" is referred both to the number of hidden layers and to the number of artificial neurons contained in each of these layers.

[5] ≪[I]*n many practical applications the choice of pre-processing will be one of the most significant factors in determining the performance of the final system*≫ (see [1] at page 295).

- The use of standard learning algorithms of Error Back Propagation (EBP) type like, for instance, the Levenberg-Marquardt EBP one or the Scaled Conjugate Gradient EBP one.

As far as the learning phase is concerned:

- For each of the considered MLP ANNs we used a subset of 40000 pairs (x_i, y_i) (always the same) of the 50000 pairs randomly generated, in order to train the MLP ANNs. Then, once the learning was complete, we used the subset of the remaining 10000 pairs to check the performances of each trained MLP ANN. Notice that these subsets, respectively known as "training set" and "testing set", are built in such a way that they both respect the percentage distribution in table 1. This allows the comparison of the performances of a given MLP ANN, outreached during the training phase, with the performances of the same MLP ANN reached during the validation phase;
- Likely, the most widespread rules for learning stopping are the ones which arrest the learning process after a prefixed number of epochs[6], and the ones which stop the learning at the epoch in which the MLP ANN commits an error[7] lower than a prefixed threshold. In our approach we considered a stopping learning rule given by a mixing of the ones just described. Indeed, we do not stop the learning process

until the number of epochs is lower or equal than a prefixed value

AND

until the error committed by the ANN MLP is greater than a prefixed threshold.

In particular, we used different values for the maximum number of epochs, and we used the value 10^{-6} for the error threshold.

In terms of the results, most of the adopted different architectures for our MLP ANNs have performed in quite similar ways, none of them being fully satisfactory (with respect to the others). In table 2 we report some results obtained by using one of the most outstripping of such MLP ANNs, the one characterized by: 2 hidden layers; 25 artificial neurons in the first hidden layer and 10 artificial neurons in the second hidden layer; the scaled conjugate gradient EBP learning algorithm. Further, we considered the following values for the maximum number of epochs: 5000, 15000, 75000 and 250000 respectively.

As far as the notation used in table 2 is concerned: "%training" indicates the number of pairs (x_i, y_i) belonging to the training set adopted; "%validation" indicates the number of pairs (x_i, y_i) belonging to the validation set used; "%i", with $i \in \{0, 1000, 2000, \ldots, 7000, 8000, 9000\}$, indicates the number of pairs (x_i, y_i) belonging to the validation set having output value equal to i, which have been correctly recognized once the learning process is completed. Notice that:

- Only for a large number of epochs, 250000 in the presented results, the considered MLP ANNs seem effective, in fact %training $= 73.367\%$ and %validation $= 71.474\%$. In any case, with particular reference to the performances obtained during the validation phase, although the results are appreciable from a research standpoint, they might not be usable for commercial purposes (a failure rate of about 30% may be still unacceptable);
- As usual in MLP ANNs applications, the performances obtained during the training phase are better than the ones obtained during the validation phase in all the tests, although the discrepancy is not so evident;
- As conjectured above, the non uniform distribution od the pairs with respect to the output values makes the learning process more difficult, in correspondence of the less frequent output values. As a case in point, for each test compare the value of %4000 (4000 is the least frequent output value) with the value of %9000 (9000 is the most frequent output value).

The third and last step we faced was, as premised in Section 4.2, the implementation of medium/large-sized MLP ANNs, in order to apply a form of regression (instead of approximation) to the searched for unknown function. The guidelines we followed during the design of the latter MLP ANNs were substantially the same used for the design of the small/medium-sized MLP ANNs. Only the following differences were considered:

- The use of two or three – and no less – hidden layers;
- The use of an higher number of artificial neurons for each of the considered hidden layers.

Also as far as the learning phase is concerned, we essentially adopted the settings used in the case of the small/medium-sized MLP ANNs. In term of the results, even now all the variously configured MLP ANNs have performed in quite

[6] In the ANN jargon, the term "epoch" means an iteration of the iterative training phase, during which all the input-output pairs belonging to the training set are used to train the MLP ANN.

[7] Usually, the error is expressed in terms of Mean Square Error calculated between the true output values and the output values provided by the ANN.

Table 2. Results related to a small/medium-sized ANN MLP.

Item	*Test 1*	*Test 2*	*Test 3*	*Test 4*
Number of epochs	5000	15000	75000	250000
%training	36.245%	28.578%	47.395%	73.367%
%validation	36.107%	28.036%	46.899%	71.474%
%0	20.274%	8.537%	25.600%	66.413%
%1000	16.000%	11.733%	20.253%	41.600%
%2000	26.582%	29.114%	24.829%	37.658%
%3000	30.993%	32.534%	18.440%	50.514%
%4000	15.603%	19.149%	14.201%	33.333%
%5000	14.201%	25.444%	20.038%	34.024%
%6000	25.000%	22.710%	33.333%	55.916%
%7000	31.129%	24.518%	50.713%	52.066%
%8000	30.760%	24.822%	68.331%	55.463%
%9000	50.869%	39.125%	81.133%	91.445%

similar ways, but unlike in the case of the small/medium-sized ANN MLPs, all the resulting configurations are *quite satisfactory*. In table 3 we report some results obtained by one of our on-average-performing MLP ANN. The features of the considered MLP ANN are summarized as follows: 2 hidden layers; 95 artificial neurons in the first hidden layer and 76 artificial neurons in the second hidden layer; the scaled conjugate gradient EBP learning algorithm. Further, we considered the following values for the maximum number of epochs: 1250, 2500, 5000 and 7500 respectively. Observe that:

- Even with a small number of epochs, 7500 in the presented results, the training process for the considered MLP ANNs seems satisfactory. In fact, %training $= 98.818\%$ and %validation $= 97.900\%$. Furthermore, observe the close-to-100% values of $\%i$, with $i \in \{0, 1000, 2000, \ldots, 7000, 8000, 9000\}$. With particular reference to the performances obtained during the validation phase, the results are fully appreciable from a research standpoint and surely usable for commercial purposes, as several north-eastern Italian tourism experts confirmed;
- As usual in MLP ANNs applications, and as in the case of the small/medium-sized MLP ANNs, the performances obtained during the training phase are better than the ones obtained during the validation phase in all the test, although not in an evident way;
- As conjectured above and as in the case of the small/medium-sized ANN MLPs, the non-uniform distribution of the pairs in the training set, with respect to the output value, makes the learning process more difficult in correspondence of less frequent output values.

Table 3. Results related with a medium/large-sized MLP ANN.

Item	*Test 1*	*Test 2*	*Test 3*	*Test 4*
Number of epochs	1250	2500	5000	7500
%training	28.115%	45.233%	96.993%	98.818%
%validation	27.646%	44.579%	95.939%	97.900%
%0	1.067%	12.907%	94.461%	97.358%
%1000	16.267%	36.267%	83.733%	91.733%
%2000	46.835%	34.494%	86.076%	90.823%
%3000	32.363%	26.712%	91.952%	92.980%
%4000	17.021%	23.404%	67.376%	88.653%
%5000	20.710%	26.036%	88.757%	95.858%
%6000	40.458%	42.748%	97.137%	97.901%
%7000	38.017%	46.281%	92.837%	97.521%
%8000	43.112%	53.801%	98.931%	99.525%
%9000	33.825%	62.371%	99.758%	99.934%

5. Some final remarks

We have proposed a solution technique alternative to existing ones for an on-line-booking problem. In particular, to this purpose we have used MLP ANNs. The considered ANNs show the strong influence, on the performance, of the number of layers and the number of nodes in the MLP. Moreover, considering the computational time needed for training and validate our different MLP ANNs, we can observe that in many cases they were noticeably high, using a commercial PC. However, we should consider that the training process of the ANNs is substantially an off-line process for hotel booking.

Obviously, other experiences must be considered, in order to establish at what extent our proposal is effective in on-line booking. Hotel industry people would be glad to have instruments which manage different kinds of services: they wish also to "foresee" and "anticipate" customer's preferences, in the sense that they wish to be able to offer services, such that the customer is not yet fully aware.

References

[1] Bishop CM. Neural networks for pattern recognition. New York: Oxford University Press; 1995.

[2] Grippo L, Sciandrone M. Metodi di ottimizzazione per le reti neurali. Rapporto Tecnico, Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica 2003;09-03:1-67. [In Italian]

[3] Haykin S. Neural Networks - A comprehensive foundation. 2nd ed. New Jersey: Prentice Hall International; 1999.

[4] Novikoff ABJ. On convergence proofs on perceptrons. Proc. of the Symposium on the Mathematical Theory of Automata 1962;XII:615-22.

[5] Leshno M, Lin VYa, Pinkus A, Schocken S. Multilayer feedforward networks with a non-polynomial activation function can approximate any function. Neural Networks 1993;6:861-7.

[6] Burges CJC. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 1998;2:121-67.

[7] Poggio T, Girosi F. Networks for approximation and learning. Proc. of the IEEE 1990;78:1481-97.

[8] Vapnik VN. Statistical learning theory. Hoboken: John Wiley & Sons; 1998.

[9] Vapnik VN. The nature of statistical learning theory. Berlin: Springer Verlag Inc; 2000. [Series: Statistics for engineering and information science]

[10] Pinkus A. TDI-subspaces of $C(\mathbb{R}^d)$ and some density problems from neural networks. J of Approximation Theory 1996;85:269-87.

[11] Borisovich Y, Bliznyakow N, Izrailevich Y, Fomenko T. Introduction to topology. Moscow: Mir Publishers, 1985.

[12] Garey MR, Johnson DS. Computers and intractability: a guide to the theory of NP-completeness. San Francisco: W.H. Freeman and Company; 1979.) [Series of Books in the Mathematical Sciences]

[13] Dennis AR, Williams ML. Electronic brainstorming: theory, research, and future directions. In: Paulus PB, Nijstad BA., editors. Group Creativity, New York: Oxford University Press; 1999, p. 160-78.