Neurocomputing 71 (2008) 2538–2543

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An extended class of multilayer perceptron

R. Lopez *, E. Oñate

International Center for Numerical Methods in Engineering (CIMNE), Edificio C1, Gran Capitán s/n, 08034 Barcelona, Spain

ARTICLE INFO

Available online 29 April 2008

PACS:
84.35.+i
02.30.Xx

Keywords:
Multilayer perceptron
Independent parameters
Boundary conditions
Lower and upper bounds

ABSTRACT

In this work an extended class of multilayer perceptron is presented. This includes independent parameters, boundary conditions and lower and upper bounds. In some cases, such extensions contain a priori information of the problem. On some other situations they are necessary in order to define a correct representation for the solution.

The use of this augmented class of neural network is illustrated through a case study in the optimal control theory. The numerical results are compared against the analytical solution.

1. Introduction

Variational problems arise in numerous applications. The theories of function regression and pattern recognition [2], for instance, concern specific kinds of problems in the calculus of variations. Other types include optimal control [8], inverse analysis [9] or optimal shape design [3].

Mathematically, the aim of a variational problem is to find a function which is the optimal (minimal or maximal) value of a specified functional [5]. A functional means a correspondence which assigns a number to each function belonging to some class.

However, most variational problems cannot be solved analytically, and the only practical technique to approach their solution is to approximate them by using a direct method [1]. The fundamental idea underlying the so-called direct methods is to reduce the variational problem at hand into a function optimization problem in many dimensions.

The solving approach of a direct method consists of three steps [1]. The first step is to choose a suitable function space in which the solution to the problem is to be approximated. The elements of this family of functions are parameterized by a set of real numbers. In the second step the variational problem is formulated by selecting an appropriate objective functional, defined on the function space chosen before. The third step is to solve the reduced function optimization problem. This is performed with some algorithm capable of finding an optimal set of parameters.

A variational formulation for the multilayer perceptron provides a direct method for the solution of variational problems [11]. Indeed, any learning task for that neural network can be stated in terms of minimizing some objective functional. The reduced function optimization problem is then solved by the training algorithm. Multilayer perceptron neural networks are able to span a function space with universal approximation properties [6]. They hereby cause neural computation to be a very appropriate paradigm for the solution of variational problems.

This work presents a class of multilayer perceptron which is extended with independent parameters, boundary conditions and lower and upper bounds. From these, an adoption of independent parameters is a novel subject in the field of neural networks. Incorporating boundary conditions is not new [14], although it has not been discussed too often in the community. Finally, inclusion of lower and upper bounds is a trivial but also an essential issue.

In some situations these extensions can improve the performance by the numerical method. In other cases they allow to deal with applications which would be untractable otherwise. In summary, this augmented class of neural network might be able to span a more suited function space for some variational problems.

The use of a variational formulation for the multilayer perceptron with an extended class of that neural network is investigated through the solution of a classical example in the optimal control theory. In particular the car problem is solved here [8], and the approximated results by this direct method are compared against the exact ones by the analytical solution.

2. Motivation

Consider a car which is to be driven along the x-axis from some position x_i at velocity v_i and acceleration a_i to some desired

* Corresponding author. Tel.: +34 934017399; fax: +34 934016517.
E-mail addresses: rlopez@cimne.upc.edu (R. Lopez), onate@cimne.upc.edu (E. Oñate).

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2007.12.037

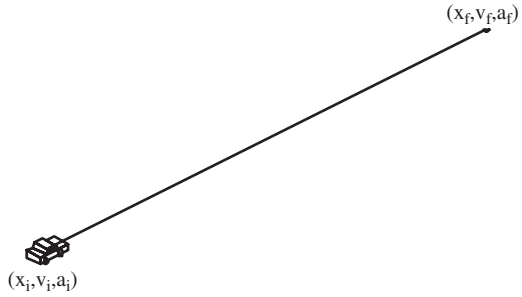

Fig. 1. The car problem statement.

position x_f at desired velocity v_f and desired acceleration a_f in a minimum time t_f, see Fig. 1 [8].

Let us approximate the car by a unit point mass that can be accelerated by using the throttle or decelerated by using the brake. Selecting position and velocity as state variables, the mathematical model of this system is given by two ordinary differential equations with their corresponding initial conditions,

$$\dot{x}(t) = v(t), \tag{1}$$
$$\dot{v}(t) = a(t), \tag{2}$$
$$x(0) = x_i, \tag{3}$$
$$v(0) = v_i, \tag{4}$$

for $t \in [0, t_f]$, and where the final time t_f is undefined. The control variable here is the car acceleration, which must hold

$$a(t_i) = a_i, \tag{5}$$
$$a(t_f) = a_f. \tag{6}$$

The acceleration is bounded by the capability of the engine, and the deceleration is limited by the braking system parameters. If the maximum acceleration is $\max(a) > 0$, and the minimum deceleration is $\min(a) < 0$, such bounds on the control variable can be written

$$\min(a) \leqslant a(t) \leqslant \max(a). \tag{7}$$

As the goal is to make the car reach the final point as quickly as possible, the objective functional is given by

$$F[a(t)] = t_f. \tag{8}$$

On the other hand, the car is to be driven to a desired position x_f and a desired velocity v_f, hence $x(t_f) = x_f$ and $v(t_f) = v_f$. Such constraints on the state variables can be expressed as error functionals,

$$E_x[a(t)] \equiv x(t_f) - x_f = 0, \tag{9}$$

$$E_v[a(t)] \equiv v(t_f) - v_f = 0, \tag{10}$$

where E_x and E_v are called the final position and velocity errors, respectively.

For this case study the initial position, initial velocity, final position, final velocity, initial acceleration, final acceleration, minimum acceleration and maximum acceleration are set to $x_i = 0$, $v_i = 0$, $x_f = 1$, $v_f = 0$, $a_i = 0$, $a_f = 0$, $\min(a) = -1$ and $\max(a) = 1$, respectively. This particular example has an analytical solution for the optimal control given by [8]

$$a^*(t) = \begin{cases} 0, & t = 0, \\ 1, & 0 < t < 1, \\ -1, & 1 < t < 2, \\ 0, & t = 2, \end{cases} \tag{11}$$

which provides a minimum final time $t_f^* = 2$.

The statement and the solution itself of this car problem points out a number of significant issues. First, some variational problems might require a function space with independent parameters associated to it. Indeed, the final time is not part of the control, but it represents the interval when it is defined. Second, the elements of this family of functions may need to satisfy boundary conditions. The control here must hold given initial and final values. Third, these functions and the independent parameters might be bounded. Certainly, the control has lower and upper bounds, and the final time must be positive. Finally, this kind of applications demand spaces of functions with very good approximation properties, since they are likely to have very nonlinear solutions. Here the optimal control even exhibits discontinuities.

From all that, although the car problem is conceptually quite simple, its numerical solution can be a very difficult task. Also, this problem can be solved analytically, which makes it a very suited example for testing the performance of direct methods.

3. Approach

As it was said in Section 1, a variational formulation for the multilayer perceptron provides a direct method for the solution of variational problems [11].

Fig. 2 depicts an activity diagram for the learning problem in that neural network. The solving approach here consists of three steps. The first step is to choose a suitable parameterized function space in which the solution to the problem is to be approximated. The elements of this family of functions are those spanned by a multilayer perceptron. In the second step the variational problem is formulated by selecting an appropriate objective functional, defined on the function space chosen before. The third step is to solve the reduced function optimization problem. This is performed with a training algorithm capable of finding an optimal set of free parameters.

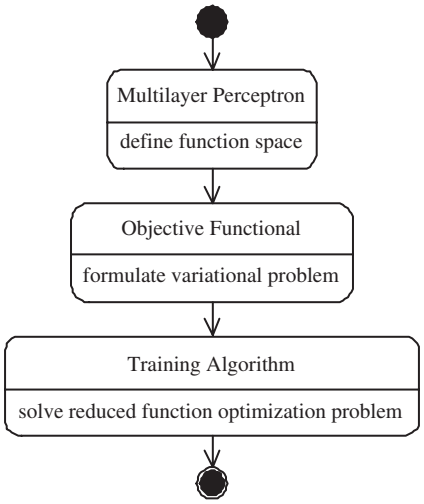

Fig. 2. State diagram for the learning problem in the multilayer perceptron.

In the next subsections a more detailed explanation of these three steps is provided.

3.1. Function space

Mathematically, a multilayer perceptron spans a family of functions from a finite dimensional input to a finite dimensional output [11]. These functions are parameterized by all the biases and synaptic weights in the neural network [12]. The dimension of the function space spanned by a multilayer perceptron is therefore equal to the total number of biases and synaptic weights.

The universal approximation properties state that a multilayer perceptron with as few as one hidden layer of sigmoid neurons and an output layer of linear neurons provides a general framework for approximating any function up to any desired degree of accuracy, provided sufficiently many hidden neurons are available [6]. Also, some extensions to the multilayer perceptron will, in many cases, improve the function space it spans.

If some information not related to input–output relationships is needed, then the problem is said to have independent parameters. The independent parameters are not a part of the neural network, but they are associated to it.

Consequently, a multilayer perceptron with associated independent parameters spans a space of functions V from an input space $X \subseteq \mathbb{R}^n$ to an output space $Y \subseteq \mathbb{R}^m$, where n and m are the number of inputs and outputs, respectively. The elements of this family of functions are parameterized by both, the biases and synaptic weights vector, $\underline{\alpha} = (\alpha_1, \ldots, \alpha_p)$, and the independent parameters vector, $\underline{\beta} = (\beta_1, \ldots, \beta_q)$. The total set of free parameters is thus $(\underline{\alpha}, \underline{\beta}) = (\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$ and the dimension of V is $p + q$. The functions here are of the form

$$\begin{aligned} &\mathbf{y} : X \to Y \\ &\quad \mathbf{x} \mapsto \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}). \end{aligned}$$

Thus, the free parameter vector constitutes the parameterization of the function space, and it is composed of the biases and synaptic weights and the independent parameters. The first group defines the output from the neural network for a given input. The second group provides some separate sort of information. In this way, distinct values for the free parameters cause distinct elements in the function space which a specific multilayer perceptron defines.

If some outputs are specified for given inputs, then the problem is said to include boundary conditions. A boundary condition between some input $x = a$ and some output $y = y_a$ is written $y(a) = y_a$. In order to deal with boundary conditions the output signals from the neural network can be postprocessed as follows:

$$\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) = \varphi_0(\mathbf{x}) + \varphi_1(\mathbf{x})\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}), \tag{12}$$

where the function $\varphi_0(\mathbf{x})$ is called a particular solution term and the function $\varphi_1(\mathbf{x})$ is called an homogeneous solution term. The first must hold $\varphi_0(a) = y_a$ if there is a condition $y(a) = y_a$. The second must hold $\varphi_1(a) = 0$ if there is a condition $y(a) = y_a$. It is easy to see that this approach makes all the elements of the function space to satisfy the boundary conditions.

The expressions of the particular and homogeneous solution terms depend on the problem at hand. For the common situation of one input and one output variables and two boundary conditions $y(a) = y_a$ and $y(b) = y_b$, we could have

$$\varphi_0(x) = y_a + \frac{y_b - y_a}{b - a}x, \tag{13}$$

$$\varphi_1(x) = (x - a)(x - b). \tag{14}$$

The particular and homogeneous solution terms might be difficult to derive if the number of input and output variables is high and the number of boundary conditions is also high.

If some output variables are restricted to fall in some interval, then the problem is said to have lower and upper bounds. An easy way to treat lower and upper bounds is to postprocess the outputs from Eq. (12) in the next way:

$$\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) = \begin{cases} \min(\mathbf{y}), & \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) < \min(\mathbf{y}), \\ \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}), & \min(\mathbf{y}) \leqslant \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) \leqslant \max(\mathbf{y}), \\ \max(\mathbf{y}), & \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) > \max(\mathbf{y}), \end{cases} \tag{15}$$

where $\min(\mathbf{y})$ and $\max(\mathbf{y})$ represent the minimum and maximum allowed values for the output variables, respectively.

Similarly, if some independent parameters are bounded they can be postprocessed in the following manner:

$$\underline{\beta} = \begin{cases} \min(\underline{\beta}), & \underline{\beta} < \min(\underline{\beta}), \\ \underline{\beta}, & \min(\underline{\beta}) \leqslant \underline{\beta} \leqslant \max(\underline{\beta}), \\ \max(\underline{\beta}), & \underline{\beta} > \max(\underline{\beta}), \end{cases} \tag{16}$$

where $\min(\underline{\beta})$ and $\max(\underline{\beta})$ represent the minimum and maximum allowed values for the independent parameters, respectively.

3.2. Variational problem

In order to formulate the variational problem, an appropriate objective functional must be selected. An objective functional for the multilayer perceptron is of the form

$$\begin{aligned} &F : V \to \mathbb{R} \\ &\quad \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) \mapsto F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]. \end{aligned}$$

The objective functional defines the task that the neural network is required to accomplish and provides a measure of the quality of the representation that it is required to learn. In this way, the choice of a suitable objective functional depends on the particular application.

The simplest variational problems for the multilayer perceptron are those in which no constraints are posed on the solution $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*)$. In this way, the general unconstrained problem can be formulated as follows:

Problem 1 (*Unconstrained variational problem*). *Let V be the space of all functions $\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})$ spanned by a multilayer perceptron, and let $p + q$ be the dimension of V. Find a function $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*) \in V$ for which the functional*

$$F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})],$$

defined on V, takes on a minimum value.

In other words, the unconstrained variational problem for the multilayer perceptron is stated in terms of the minimization of the objective functional [11].

A variational problem for the multilayer perceptron can be specified by a set of constraints, which are equalities or inequalities that the solution $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*)$ must satisfy. Such constraints can be expressed as error functionals:

$$\begin{aligned} &E_i : V \to \mathbb{R} \\ &\quad \mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta}) \mapsto E_i[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})], \end{aligned}$$

for $i = 1, \ldots, l$, and where l is the number of constraints. Thus, the general constrained problem can be formulated as follows:

Problem 2 (*Constrained variational problem*). *Let V be the space of all functions $\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})$ spanned by a multilayer perceptron, and let $p + q$ be the dimension of V. Find a function $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*) \in V$ such that*

$$E_i[\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*)] = 0,$$

for $i = 1, \ldots, l$, and for which the functional

$F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]$,

defined on V, takes on a minimum value.

In other words, the constrained variational problem for the multilayer perceptron consists of finding a function which makes all the constraints to be satisfied and the objective functional to be an extremum.

A possible approach for solving a constrained variational problem is to reduce it into an unconstrained problem. This can be done by adding a penalty term to the objective functional for each of the constrains in the original problem. Adding a penalty term gives a large positive or negative value to the objective functional when infeasibility due to a constrain is encountered. The general constrained variational problem for the multilayer perceptron can then be reformulated as follows:

Problem 3 (*Reduced unconstrained variational problem*). *Let V be the space consisting of all functions $\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})$ that a given multilayer perceptron can define, and let $p + q$ be the dimension of V. Find a function $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*) \in V$ for which the functional*

$$F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})] + \sum_{i=1}^{l} \rho_i \|E_i[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]\|^2,$$

defined on V and with $\rho_i > 0$, for $i = 1, \ldots, l$, takes on a minimum value.

The parameters ρ_i are called the penalty term weights. Note that, while the squared norm of the error in the constraint is the metric most used, any other suitable metric can be used.

For large values of ρ_i, it is clear that the solution $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*)$ of Problem 3 will be in a region where $E_i[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]$ are small. Thus, for increasing values of ρ_i, it is expected that the solution $\mathbf{y}^*(\mathbf{x}; \underline{\alpha}^*, \underline{\beta}^*)$ of Problem 3 will approach the constraints and, subject to being close, will minimize the objective functional $F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]$. Ideally then, as $\rho_i \to \infty$, the solution of Problem 3 will converge to the solution of Problem 2.

3.3. Reduced function optimization problem

The objective functional, $F[\mathbf{y}(\mathbf{x}; \underline{\alpha}, \underline{\beta})]$, has an objective function associated to it, $f(\underline{\alpha}, \underline{\beta})$, which is defined as a function of the free parameters in the neural network:

$$f : \mathbb{R}^{p+q} \to \mathbb{R}$$
$$(\underline{\alpha}, \underline{\beta}) \mapsto f(\underline{\alpha}, \underline{\beta}).$$

The minimum value of the objective functional is achieved for a vector of free parameters at which the objective function takes on a minimum value. Therefore, any learning task for the multilayer perceptron, formulated as a variational problem, can be reduced to a function optimization problem [11]:

Problem 4 (*Reduced function optimization problem*). *Let $\mathbb{R}^{p+q}$ be the vector space of all free parameters $(\underline{\alpha}, \underline{\beta})$ of a multilayer perceptron. Find a vector $(\underline{\alpha}^*, \underline{\beta}^*) \in \mathbb{R}^{p+q}$ for which the function*

$f(\underline{\alpha}, \underline{\beta})$,

defined on $\mathbb{R}^{p+q}$, takes on a minimum value.

In this sense, a variational formulation for the multilayer perceptron provides a direct method for solving variational problems. The universal approximation properties cause neural computation to be a very appropriate paradigm for the solution of these problems.

The use of gradient information is of central importance in finding training algorithms which are sufficiently fast to be of practical use for large scale applications [2]. For a multilayer perceptron, the gradient vector ∇ of the objective function $f(\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q)$ is written as

$$\nabla f(\alpha_1, \ldots, \alpha_p, \beta_1, \ldots, \beta_q) = \left(\frac{\partial f}{\partial \alpha_1}, \ldots, \frac{\partial f}{\partial \alpha_p}, \frac{\partial f}{\partial \beta_1}, \ldots, \frac{\partial f}{\partial \beta_q} \right). \quad (17)$$

When the desired output of the neural network for a given input is known, the objective function gradient can usually be found analytically using a backpropagation algorithm [15]. In some other circumstances exact evaluation of the gradient is not possible and numerical differentiation must be applied [2].

Finally, the training algorithm is entrusted to solve the reduced function optimization problem. The training process is determined by the way in which the adjustment of the parameters in the neural network takes place.

There are many different training algorithms, which have a variety of different computation and storage requirements. Moreover, there is not a training algorithm best suited to all locations [16]. Training algorithms might require information from the objective function only, the gradient vector of the objective function or the Hessian matrix of the objective function [2]. These methods, in turn, can perform either global or local optimization. Some of the most used are gradient descent [2], conjugate gradient [2], the quasi-Newton method [2], evolutionary algorithms [4] or particle swarm optimization [7].

Overall, a variational formulation for the multilayer perceptron provides a direct method to approximate the solution of variational problems, in any dimension and up to any desired degree of accuracy [11]. In some cases a standard multilayer perceptron will define a correct representation for the solution. In some other occasions some extensions to this class of neural network shall be required. In this regard, any lack of success in a learning task must arise from a wrong choice of the function space, the lack of the objective functional or inadequate training.

4. Results

Here an extended class of multilayer perceptron is trained to find the optimal control and the corresponding optimal trajectory for the car problem formulated in Section 2. The problem is solved with the Flood library [10].

4.1. Function space

The first step is to choose a function space to represent the control $a(t)$. Here a multilayer perceptron with a sigmoid hidden layer and a linear output layer is used. The network architecture must have one input, the time t, and one output neuron, the acceleration a. The number of neurons in the hidden layer is set to be six, see Fig. 3.

On the other hand information about the final time is required. Thus, an independent parameter t_f must be associated to the multilayer perceptron.

This neural network spans a family V of functions $a(t; \underline{\alpha}, t_f)$ of dimension $p + q = 19 + 1$, where $p = 19$ is the number of biases and synaptic weights and $q = 1$ is the number of independent parameters. Elements of V are of the form

$$a : [0, t_f] \to \mathbb{R}$$
$$t \mapsto a(t; \underline{\alpha}, t_f).$$

Note that the final time t_f is unspecified. Moreover, the goal in this problem is to find a control signal which makes this independent parameter to be minimum.

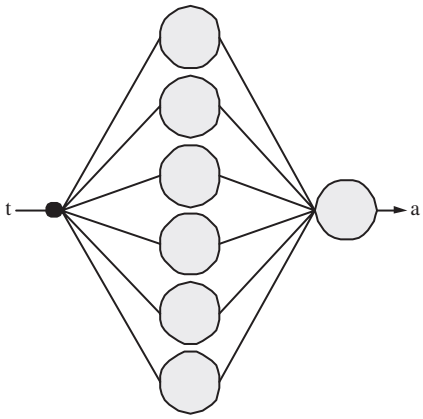

Fig. 3. Network architecture for the car problem.

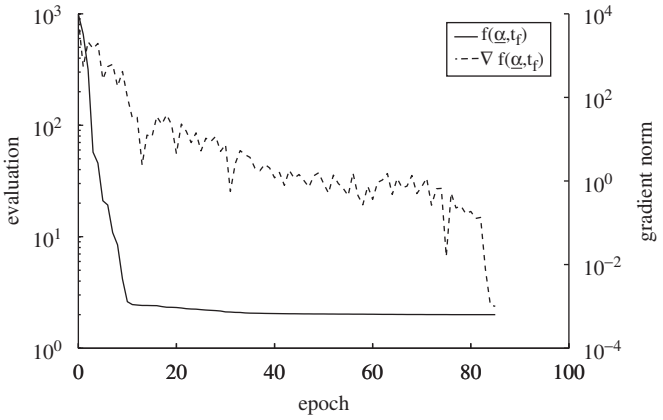

Fig. 4. Evaluation and gradient norm training histories for the car problem.

Here the control values for the initial and final times are given. More specifically, they are set to be $a(0)=0$ and $a(t_f)=0$. A particular solution term for this problem can be $\varphi_0=0$, and an homogeneous solution term can be $\varphi_1=t(t-t_f)$. Then the output signals from the neural network are postprocessed so as to satisfy the boundary conditions in the form

$$a(t;\underline{\alpha})=t(t-t_f)a(t;\underline{\alpha}). \tag{18}$$

Indeed, it is easy to see that all the functions defined by Eq. (18) satisfy $a(0)=0$ and $a(t_f)=0$.

On the other hand, the control is lower and upper bounded by the brake and the engine performances, respectively. In particular, the acceleration is restricted to lie in the interval $a(t)\in[-1,1]$. The outputs from Eq. (18) are thus postprocessed in the form

$$a(t;\underline{\alpha},t_f)=\begin{cases}-1, & a(t;\underline{\alpha},t_f)<-1,\\ a(t;\underline{\alpha},t_f), & 0\leqslant a(t;\underline{\alpha},t_f)\leqslant 1,\\ 1, & a(t;\underline{\alpha},t_f)>1.\end{cases} \tag{19}$$

Finally, the final time must be equal or greater than zero, $t_f\in[0,\infty)$, so this independent parameter is bounded as follows:

$$t_f=\begin{cases}0, & t_f<0,\\ t_f, & t_f\geqslant 0.\end{cases} \tag{20}$$

Here all the free parameters in the neural network are initialized at random. In this way, a random control with a random value for the final time are used as an initial guess to solve the problem.

4.2. Variational problem

The second step is to select an objective functional, in order to formulate the variational problem. From Eqs. (8)–(10), the objective functional must be composed of three terms, the objective itself (final time) and a penalty term for each of the two constraints (final position and final velocity). Thus, a possible objective functional for this case study is

$$F[a(t;\underline{\alpha},t_f)]=t_f+\rho_X(x(t_f)-1)^2+\rho_V(v(t_f))^2, \tag{21}$$

where ρ_X and ρ_V are called the error position and error velocity penalty term weights. Both of them are set to 1000.

Note that evaluating the objective functional in Eq. (21) requires a numerical method for integration of ordinary differential equations, in order to obtain the final position and velocity for a given acceleration signal and a given final time. Here the Runge–Kutta–Fehlberg method [13] is chosen, and the tolerance set to 10^{-15}.

On the other hand, a backpropagation algorithm for the objective function gradient $\nabla f(\underline{\alpha},t_f)$ is not possible to be derived here, since the target outputs from the neural network are not known [15]. Instead, the central differences method for numerical differentiation is to be used with $\varepsilon=10^{-6}$ [2].

Table 1
Training results for the car problem

$N=85$
$M=6563$
$CPU=197$
$f(\underline{\alpha}^*,t_f^*)=1.999$
$\|\nabla f(\underline{\alpha}^*,t_f^*)\|=0.001$
$e_x(\underline{\alpha}^*,t_f^*)=5.714\times10^{-4}$
$e_v(\underline{\alpha}^*,t_f^*)=4.444\times10^{-4}$
$t_f^*=1.999$

4.3. Reduced function optimization problem

The third step is to pick out a suitable training algorithm and solve the reduced function optimization problem. Here a quasi-Newton method with BGFS train direction and Brent optimal train rate methods is used [2]. The tolerance in the Brent method is set to 10^{-6}. This training algorithm has demonstrated fully convergence to the global optimum in this application.

In this example, the training algorithm is set to stop when it cannot perform any better. At this time, the Brent's method gives zero train rate for any train direction.

The evaluation and the gradient norm of the initial guess are 992.051 and 7573.454, respectively. The quasi-Newton method here needs 85 epochs to converge. After training, the evaluation and the gradient norm fall to 1.999 and 0.001, respectively. Fig. 4 depicts the training history for this two variables. Note that a base 10 logarithmic scale is used for the y-axis in both plots.

Table 1 shows the training results for this case study. Here N denotes the number of epochs, M the number of evaluations, CPU the computing time in seconds for a laptop AMD 3000, f the final objective function value, $\|\nabla f\|$ the final gradient norm, e_x the final position error, e_v the final velocity error and t_f^* the optimal final time. It can be seen that the final errors in the position and the velocity of the car are very small. Also, the final time found by the neural network matches that provided by the optimal function in Eq. (11). More specifically, the errors made in the constraints are less than 10^{-3} and the error made in the final time is around 0.1%.

The optimal control obtained by the neural network is plotted in Fig. 5. This signal is very similar to the analytical solution in Eq. (11).

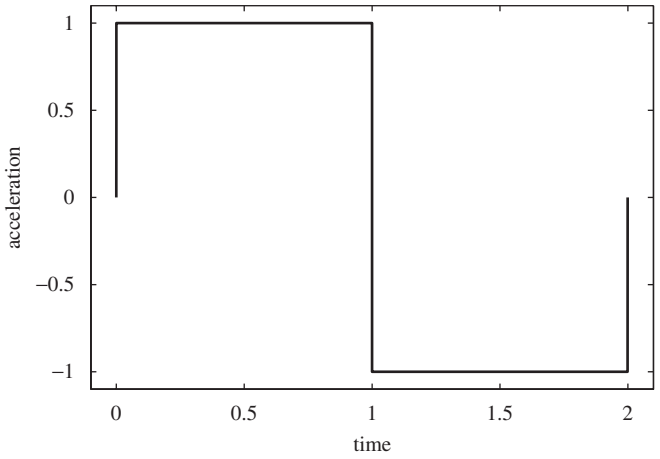

Fig. 5. Optimal control for the car problem.

5. Conclusions

An extended class of multilayer perceptron with independent parameters, boundary conditions and lower and upper bounds might be able to span a more suited function space for some variational problems.

The use of that class of neural network within a variational formulation has been investigated through the solution of the car problem, demonstrating very good agreement between the approximate and the exact values.

Future work is focussed on using the proposed approach for different applications in engineering. Special interest is put on the fields of optimal control, inverse analysis and optimal shape design.

References

[1] J.T. Betts, A survey of numerical methods for trajectory optimization, AIAA J. Guidance Control Dyn. 21 (2) (1998) 193–207.
[2] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[3] D. Bucur, G. Buttazzo, Variational Methods in Shape Optimization Problems, Birkhauser, Basel, 2005.
[4] D.B. Fogel, An introduction to simulated evolutionary optimization, IEEE Trans. Neural Networks 5 (1) (1994) 3–14.
[5] I.M. Gelfand, S.V. Fomin, Calculus of Variations, Prentice-Hall, Englewood Cliffs, NJ, 1963.
[6] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Networks 2 (5) (1989) 359–366.
[7] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[8] D.E. Kirk, Optimal Control Theory. An Introduction, Prentice-Hall, Englewood Cliffs, NJ, 1970.
[9] A. Kirsch, An Introduction to the Mathematical Theory of Inverse Problems, Springer, Berlin, 1996.
[10] R. Lopez, Flood: an open source neural networks C+ library, 2007 ⟨www.cimne.com/flood⟩.
[11] R. Lopez, E. Oñate, A variational formulation for the multilayer perceptron, in: Proceedings of the ICANN 2006, Lecture Notes in Computer Science, vol. 4132(I), Springer, Berlin, 2006, pp. 159–168.
[12] J. Šíma, P. Orponen, General-purpose computation with neural networks: a survey of complexity theoretic results, Neural Comput. 15 (2003) 2727–2778.
[13] J. Stoer, R. Bulirsch, Introduction to Numerical Analysis, Springer, Berlin, 1980.
[14] Y. Weiwu, Z. Mingguang, Z. Chunkai, S. Huihe, Least squares support vector machine regression with boundary condition, in: Proceedings of the IEEE International Conference on Neural Networks and Signal Processing, 2003, pp. 79–81.
[15] P.J. Werbos, Beyond regression: new tools for prediction and analysis in the behavioral sciences, Ph.D. Thesis, Harvard University, 1974.
[16] D.H. Wolpert, W.G. MacReady, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.

Roberto Lopez was born in Salamanca (Spain) in 1976. He received the Physics Degree from the University of Salamanca in 2000 and the M.Sc. in Computational Physics from the University of Salford in 2001. He is currently working at the International Center for Numerical Methods in Engineering (CIMNE) and doing the Ph.D. in Artificial Intelligence at the Technical University of Catalonia.

Eugenio Oñate was born in Valencia (Spain) in 1953. He received the Civil Engineering Degree from the Technical University of Valencia in 1976 and the Ph.D. in Civil Engineering from the University College of Swansea in 1979. He is currently a professor of structural mechanics at the School of Civil Engineering of the Technical University of Catalonia and the director of the International Center for Numerical Methods in Engineering (CIMNE).