A Variational Formulation for the Multilayer Perceptron

Roberto Lopez and Eugenio Oñate

International Center for Numerical Methods in Engineering (CIMNE),
Technical University of Catalonia (UPC),
Barcelona, Spain
rlopez@cimne.upc.edu, onate@cimne.upc.edu
www.cimne.com

Abstract. In this work we present a theory of the multilayer perceptron from the perspective of functional analysis and variational calculus. Within this formulation, the learning problem for the multilayer perceptron lies in terms of finding a function which is an extremal for some functional. As we will see, a variational formulation for the multilayer perceptron provides a direct method for the solution of general variational problems, in any dimension and up to any degree of accuracy. In order to validate this technique we use a multilayer perceptron to solve some classical problems in the calculus of variations.

1 Introduction

Queen Dido of Carthage was apparently the first person to attack a problem that can readily be solved by using the calculus of variations. Dido, having been promised all of the land she could enclose with a bull's hide, cleverly cut the hide into many lengths and tied the ends together. Having done this, her problem was to find the closed curve with a fixed perimeter that encloses the maximum area [1]. The problem is based on a passage from Virgil's Aeneid:

> *The Kingdom you see is Carthage, the Tyrians, the town of Agenor;*
> *But the country around is Libya, no folk to meet in war.*
> *Dido, who left the city of Tyre to escape her brother,*
> *Rules here-a long a labyrinthine tale of wrong*
> *Is hers, but I will touch on its salient points in order...*
> *Dido, in great disquiet, organised her friends for escape.*
> *They met together, all those who harshly hated the tyrant*
> *Or keenly feared him: they seized some ships which chanced to be ready...*
> *They came to this spot, where to-day you can behold the mighty*
> *Battlements and the rising citadel of New Carthage,*
> *And purchased a site, which was named 'Bull's Hide' after the bargain*
> *By which they should get as much land as they could enclose with a bull's hide.*

Despite the circle appears to be an obvious solution to Dido's problem, proving this fact is rather difficult. Zenodorus proved that the area of the circle is larger

S. Kollias et al. (Eds.): ICANN 2006, Part I, LNCS 4131, pp. 159–168, 2006.

than that of any polygon having the same perimeter, but the problem was not rigorously solved until 1838 by Jakob Steiner [1].

Although the history of variational calculus dates back to the ancient Greeks, it was not until the seventeenth century in western Europe that substantial progress was made. A problem of historical interest is the brachistochrone problem, posed by Johann Bernoulli in 1696 [1]. The term brachistochrone derives from the Greek 'brachistos' (the shortest) and 'chronos' (time):

Given two points A and B in a vertical plane, what is the curve traced out by a particle acted on only by gravity, which starts at A and reaches B in the shortest time?

Sir Isaac Newton was challenged to solve the problem, and did so the very next day. In fact, the solution to the brachistochrone problem, which is a segment of a cycloid, is credited to Johann and Jacob Bernoulli, Sir Isaac Newton and Guillaume de L'Hospital [1]. In that context, Dido's problem was called the isoperimetric problem, and it was stated as:

Of all simple closed curves in the plane of a given length l, which encloses the maximum area?

The aim of both the brachistochrone and the isoperimetric problems is to find a function which is the minimal or the maximal value of a specified functional. By a functional, we mean a correspondence which assigns a number to each function belonging to some class [2]. The calculus of variations gives methods for finding extremals of functionals, and problems that consist in finding minimal and maximal values of functionals are called variational problems [2].

While some simple variational problems can be solved analytically, the only practical technique for general problems is to approximate the solution using direct methods [2]. The fundamental idea underlying the so called direct methods is to consider the variational problem as a limit problem for some function optimization problem in many dimensions [2]. Unfortunately, variational problems are difficult to solve, and new numerical methods need to be developed in order to overcome that difficulties.

Neural networks is one of the main fields of artificial intelligence. The multilayer perceptron is an important model of neural network, and much of the literature in the area is referred to that model. Traditionally, the learning problem for the multilayer perceptron has been formulated in terms of the minimization of an error function of the free parameters in the network, in order to fit the neural network to an input-target data set [3]. In that way, the only learning tasks allowed for the multilayer perceptron are data modelling type problems, such as function regression or pattern recognition.

In this work we present a variational formulation for the multilayer perceptron. Within this formulation, the learning problem for the multilayer perceptron lies in terms of solving a variational problem by minimizing a performance functional of the function space spanned by the network. The choice of a suitable performance functional depends on the particular application. On the other hand, the performance functional might need the integration of functions, ordinary differential equations or partial differential equations in order to be evaluated.

As we will see, neural networks are not only able to solve data modelling problems, but also a wide range of mathematical and physical problems. More specifically, a variational formulation for neural networks provides a direct method for solving general variational problems.

In order to validate this numerical method for the solution of variational problems, we use a multilayer perceptron to solve the brachistochrone problem and the isoperimetric problem, and compare the results provided by the neural network to the analytical results.

2 The Multilayer Perceptron Function Space

A neuron model is the basic information processing unit in a neural network, and the perceptron is the characteristic neuron model in the multilayer perceptron [4]. On the other hand, artificial neurons can be combined in a network architecture to form a neural network. The characteristic network architecture in the multilayer perceptron is the so called feed-forward architecture [4]. In this way, a multilayer perceptron can be defined as a feed-forward network architecture composed of perceptron neuron models.

Mathematically, a multilayer perceptron spans a parameterized function space V from an input $X \subseteq \mathbf{R}^n$ to an output $Y \subseteq \mathbf{R}^m$ [5]. Elements of V are parameterized by the free parameters in the network, which can be grouped together in a s-dimensional free parameter vector $\underline{\alpha} = (\alpha_1, ..., \alpha_s)$. The dimension of the function space V is therefore s. The elements of the function space spanned by a multilayer perceptron are of the form

$$\begin{aligned} \mathbf{y} : \mathbf{R}^n &\to \mathbf{R}^m \\ \mathbf{x} &\mapsto \mathbf{y}(\mathbf{x}; \underline{\alpha}). \end{aligned}$$

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 from one finite dimensional space to another up to any desired degree of accuracy, provided sufficiently many hidden neurons are available. In this sense, multilayer perceptron networks are a class of universal approximators [6].

3 The Variational Problem

Traditionally, the learning problem for the multilayer perceptron has been formulated in terms of the minimization of an error function of the free parameters in the network, in order to fit the neural network to an input-target data set [3]. In that way, the only learning tasks allowed for the multilayer perceptron are data modelling type problems.

In a variational formulation for the multilayer perceptron, the concept of error function, $e(\underline{\alpha})$, is changed by the concept or performance functional, $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$ [5]. A performance functional for the multilayer perceptron is of the form

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

The performance functional defines the task that the network is required to accomplish and provides a measure of the quality of the representation that the network is required to learn. In this way, the choice of a suitable performance functional depends on the particular application. As we will see, changing the concept of error function by the concept of performance functional allows us to extend the number of learning tasks for the multilayer perceptron to any variational problem. Some examples are optimal control problems [5], inverse problems [7] or optimal shape design problems.

The learning problem for the multilayer perceptron can then be formulated in terms of the minimization of a performance functional of the function space spanned by the neural network [5]:

Problem 1 (Variational problem for the multilayer perceptron). Let V be the space of all functions $\mathbf{y}(\mathbf{x};\underline{\alpha})$ spanned by a multilayer perceptron, and let s be the dimension of V. Find a function $\mathbf{y}^*(\mathbf{x};\underline{\alpha}^*) \in V$ for which the functional $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$, defined on V, takes on a minimum or a maximum value.

A variational problem for the multilayer perceptron can be specified by a set of constraints, which are equalities or inequalities that the solution must satisfy. Such constraints are expressed as functionals. An easy approach is to reduce the constrained problem into an unconstrained problem by adding a penalty term to the original performance functional for each constraint in the problem [5].

4 The Reduced Function Optimization Problem

The performance functional, $F[\mathbf{y}(\mathbf{x};\underline{\alpha})]$, has a performance function associated, $f(\underline{\alpha})$, which is defined as a function of the free parameters in the network [5],

$$\begin{aligned} f: \mathbf{R}^s & \to \mathbf{R} \\ \underline{\alpha} & \mapsto f(\underline{\alpha}). \end{aligned}$$

The minimum or maximum value of the performance functional is achieved for a vector of free parameters at which the performance function takes on a minimum or maximum value, respectively. Therefore, the learning problem for the multilayer perceptron, formulated as a variational problem, can be reduced to a function optimization problem [5]:

Problem 2 (Reduced function optimization problem for the multilayer perceptron). Let $\mathbf{R}^s$ be the space of all vectors $\underline{\alpha}$ spanned by the free parameters of a multilayer perceptron. Find a vector $\underline{\alpha}^* \in \mathbf{R}^s$ for which the function $f(\underline{\alpha})$, defined on $\mathbf{R}^s$, takes on a minimum or a maximum value.

In this sense, a variational formulation for the multilayer perceptron provides a direct method to approximate the solution of general variational problems, in any dimension and up to any desired degree of accuracy [5].

The training algorithm is entrusted to solve the reduced function optimization problem. There are many different training algorithms for the multilayer perceptron, which have different requirements and characteristics. One of the most used is the conjugate gradient [3].

5 The Brachistochrone Problem

In this section we solve the brachistochrone problem by means of a multilayer perceptron, and compare the neural network results to the analytical results. For this example we take the points $A = (a, f_a)$ and $B = (b, f_b)$ to be $A = (0, 1)$ and $B = (1, 0)$. The problem is solved with the Flood library [8].

The first step is to choose a network architecture to represent the curve. Here a multilayer perceptron with a sigmoid hidden layer and a linear output layer is used. This neural network is a class of universal approximator [6]. The curve is to be represented in cartesian coordinates $y = y(x)$, so the network must have one input and one output neuron. As an initial guess, we use six neurons in the hidden layer. Such a multilayer perceptron defines a family V of parameterized functions $y(x; \underline{\alpha})$ of dimension $s = 19$, which is the number of free parameters in the network. Figure 1 is a graphical representation of this network architecture.

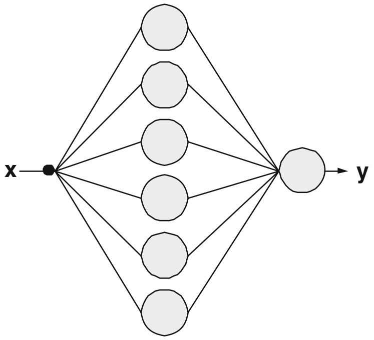

Fig. 1. Network architecture for the brachistochrone problem

The second step is to derive a performance functional for the brachistochrone problem. The time for a particle to travel from point A to point B along a curve $y(x; \underline{\alpha})$ is given by the integral [1]

$$T[y(x; \underline{\alpha})] = \frac{1}{\sqrt{2g}} \int_a^b \sqrt{\frac{1 + [y'(x; \underline{\alpha})]^2}{y_a - y(x; \underline{\alpha})}} dx, \tag{1}$$

where $g = 9.81$ is the gravitational acceleration. The constraints of this problem can be expressed as error functionals,

$$E_A[y(x;\underline{\alpha})] = y(a) - y_a = 0, \tag{2}$$

$$E_B[y(x;\underline{\alpha})] = y(b) - y_b = 0. \tag{3}$$

Making use of Equation (1) and considering the constraints (2) and (3), we get

$$F[y(x;\underline{\alpha})] = \rho_T \frac{1}{\sqrt{2g}} \int_a^b \sqrt{\frac{1+[y'(x;\underline{\alpha})]^2}{y_a - y(x;\underline{\alpha})}} dx + \rho_A \left(y(a;\underline{\alpha}) - y_a\right)^2 + \rho_B \left(y(b;\underline{\alpha}) - y_b\right)^2, \tag{4}$$

where ρ_T, ρ_A and ρ_B are penalty term ratios, which are set to 10^{-3}, 1 and 1, respectively. Please note that evaluation of the performance functional in Equation (4) requires a numerical method for the integration of functions. Here we choose the Simpson's composite method [9] with 100 integration intervals. The brachistochrone problem for the multilayer perceptron can then be stated as:

Problem 3 (Brachistochrone problem for the multilayer perceptron). Let V be the space of all functions $y(x;\underline{\alpha})$ spanned by a multilayer perceptron with 1 input, 6 sigmoid neurons in the hidden layer and 1 linear output neuron. The dimension of V is 19. Find a vector of free parameters $\underline{\alpha}^* \in \mathbf{R}^{19}$ that addresses a function $y^*(x;\underline{\alpha}^*) \in V$ for which the functional (4), defined on V, takes on a minimum value.

The third step is to choose a suitable training algorithm for solving the reduced function optimization problem. Here we use a conjugate gradient with Polak-Ribiere search direction and Brent optimal step size methods for training [3]. The tolerance in the Brent's method is set to 10^{-6}. On the other hand, training of the neural network with the conjugate gradient requires the evaluation of the performance function gradient vector $\nabla f(\underline{\alpha})$ [3]. This is carried out by means of numerical differentiation. In particular, we use the symmetrical central differences method [3] with an ϵ value of 10^{-6}. In this example, we set the training algorithm to stop after 1000 epochs of the training process. Table 1 shows the training results for this problem.

Table 1. Training results for the brachistochrone problem

$F[y^*(x;\underline{\alpha}^*)]$	$T[y^*(x;\underline{\alpha}^*)]$	$E_A[y^*(x;\underline{\alpha}^*)]$	$E_B[y^*(x;\underline{\alpha}^*)]$
$3.404 \cdot 10^{-4}$	0.576	$2.889 \cdot 10^{-3}$	$-5.716 \cdot 10^{-5}$

The errors made in the constraints by the multilayer perceptron are of order 10^{-3} and 10^{-5} for points A and B, respectively. The descent time provided by that neural network is 0.576, while that provided by the analytical result is 0.577.

This yields a percentage error of around -0.173%. Results here are good, since the constraint errors made by the network are very small and the descent time provided by the network is very similar to the descent time for the brachistochrone. Figure 2 illustrates the neural network solution to the brachistochrone problem.

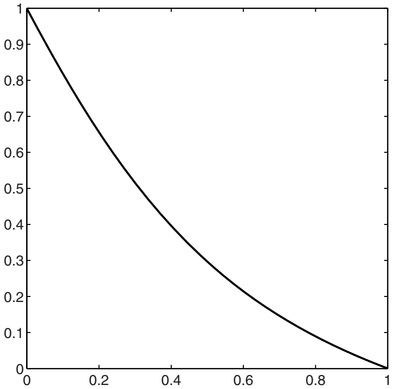

Fig. 2. Neural network solution to the brachistochrone problem

6 The Isoperimetric Problem

In this section we solve the isoperimetric problem by means of a multilayer perceptron, and compare the neural network results to the analytical results. The perimeter goal is set to be $l = 1$. This problem is solved with the Flood library [8].

Here a multilayer perceptron with a sigmoid hidden layer and a linear output layer is used. This neural network is a class of universal approximator [6]. The closed curve is to be represented in polar coordinates $r = r(\theta)$, for $\theta \in [0, 2\pi]$, so the network must have one input and one output neuron. We use 6 neurons in the hidden layer. This multilayer perceptron spans a family V of functions $r(\theta; \underline{\alpha})$ with dimension $s = 19$. Figure 3 is a graphical representation of this network architecture.

In order to derive a performance functional for the isoperimetric problem we first consider the expression for the area enclosed by a polar curve [1],

$$A[r(\theta; \underline{\alpha})] = \frac{1}{2} \int_0^{2\pi} [r(\theta; \underline{\alpha})]^2 d\theta. \qquad (5)$$

The perimeter constraint is expressed as an error functional, by considering the arc length of a curve in polar coordinates [1]

$$\begin{aligned} E_P[r(\theta; \underline{\alpha})] &= \int_0^{2\pi} \sqrt{[r(\theta; \underline{\alpha})]^2 + [r'(\theta; \underline{\alpha})]^2} d\theta - l \\ &= 0. \end{aligned} \qquad (6)$$

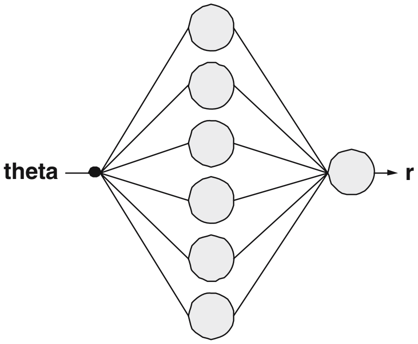

Fig. 3. Network architecture for the isoperimetric problem

Similarly, the join constraint can be expressed as an error functional,

$$\begin{aligned} E_J[r(\theta;\underline{\alpha})] &= r(2\pi;\underline{\alpha}) - r(0;\underline{\alpha}) \\ &= 0. \end{aligned} \tag{7}$$

Making use of Equations (5), (6) and (7), we obtain the performance functional to be minimized,

$$\begin{aligned} F[r(\theta;\underline{\alpha})] =& -\rho_A \frac{1}{2}\int_0^{2\pi} [r(\theta;\underline{\alpha})]^2 d\theta \\ &+ \rho_P \left(\int_0^{2\pi} \sqrt{[r(\theta;\underline{\alpha})]^2 + [r'(\theta;\underline{\alpha})]^2} d\theta - l \right)^2 \\ &+ \rho_J \left(r(2\pi;\underline{\alpha}) - r(0;\underline{\alpha}) \right)^2 , \end{aligned} \tag{8}$$

where $\rho_A = 10^{-3}$, $\rho_P = 1$ and $\rho_J = 1$ are penalty term ratios. Evaluation of (8) requires a numerical method for the integration of functions. Here we choose the Simpson's composite method [9] with 100 integration intervals. The isoperimetric problem for the multilayer perceptron is then stated as:

Problem 4 (Isoperimetric problem for the multilayer perceptron). Let V be the space of all functions $r(\theta;\underline{\alpha})$ spanned by a multilayer perceptron with 1 input, 6 sigmoid neurons in the hidden layer and 1 linear output neuron. The dimension of V is 19. Find a vector of free parameters $\underline{\alpha}^* \in \mathbf{R}^{19}$ that addresses a function $r^*(\theta;\underline{\alpha}^*) \in V$ for which the functional (8), defined on V, takes on a minimum value.

Here we use a conjugate gradient with Polak-Ribiere search direction and Brent optimal step size methods for training [3]. The tolerance in the Brent's method is set to 10^{-6}. Evaluation of the performance function gradient vector $\nabla f(\underline{\alpha})$ [3] is carried out by means of the symmetrical central differences method [3], with $\epsilon = 10^{-6}$. The training algorithm is set to stop after 100 epochs.

Table 2. Training results for the isoperimetric problem

$F[r^*(\theta;\underline{\alpha}^*)]$	$A[r^*(\theta;\underline{\alpha}^*)]$	$E_P[r^*(\theta;\underline{\alpha}^*)]$	$E_J[r^*(\theta;\underline{\alpha}^*)]$
$-6.332 \cdot 10^{-6}$	0.079579	$-1.266 \cdot 10^{-5}$	$-4.228 \cdot 10^{-9}$

The perimeter error is of order 10^{-5}, while the join error is of order 10^{-9}. The area of the closed curve provided by the neural network is 0.079579, while that of the circle is 0.079577. This yields a percentage error of around $2.513 \cdot 10^{-3}\%$. Results here are also good, since the error made in all the constraints by the network is very small and the area provided by the network is very similar to that of the circle. Figure 4 illustrates the network solution to the isoperimetric problem.

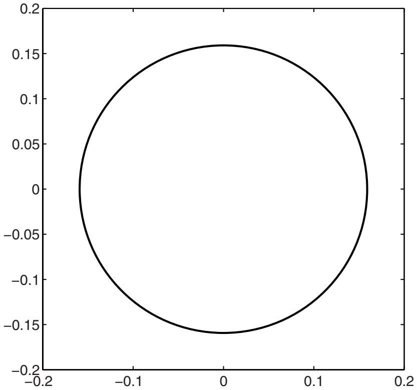

Fig. 4. Neural network solution to the isoperimetric problem

7 Conclusions

A variational formulation for neural networks provides a direct method for the solution of general variational problems, in any dimension and up to any degree of accuracy. This numerical method has been validated for two classical problems in the calculus of variations with analytical solution. Ongoing work focuses on the solution of variational problems in engineering. Some examples are optimal control, inverse or optimal shape design problems.

References

1. Weisstein, E. W.: MathWorld - A Wolfram Web Resource. http://mathworld. wolfram.com (2006).
2. Elsgolc, L. E.: Calculus of Variations. Pergamon Press (1961).

3. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1995).
4. Šíma, J. and Orponen, P.: General-Purpose Computation with Neural Networks: A Survey of Complexity Theoretic Results. Neural Computation 15 (2003) 2727-2778.
5. Lopez, R., Balsa-Canto, E. and Oñate, E.: Artificial Neural Networks for the Solution of Optimal Control Problems. Proceedings of the Sixth Conference on Evolutionary and Deterministic Methods for Design, Optimisation and Control with Applications to Industrial and Societal Problems (2005).
6. Hornik, K., Stinchcombe, M. and White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2-5 (1989) 359-366.
7. Dadvand, P., Lopez, R. and Oñate, E.: Artificial Neural Networks for the Solution of Optimal Control Problems. Proceedings of the International Conference ERCOFTAC 2006 (2006).
8. Lopez, R.: Flood: An Open Source Neural Networks C++ Library. www.cimne.com/flood (2005).
9. Stoer, J. and Bulirsch, R.: Introduction to Numerical Analysis. Springer-Verlag (1980).