Neural Networks: Using a Game Theoretic Derivative for Minimizing Maximal Errors and Designing Network Architecture

MARK MELTSER, MOSHE SHOHAM
Faculty of Mechanical Engineering
Technion-Israel Institute of Technology
Haifa, Israel

LARRY M. MANEVITZ
Department of Mathematics and Computer Science
University of Haifa 31905
Haifa, Israel
email: manevitz@mathcs2.haifa.ac.il

Abstract

In this short paper, we summarize (without proofs) the constructive method to approximate functions in the uniform (i.e. maximal error) norm, that was recently developed by the authors. [9] This is in contrast to other methods (e.g. back-propagation) that approximate only in the average error norm. We comment on the novelty of the approach and possible extensions. The method includes a "gradient descent" method in the maximal error norm (i.e. a non-differentiable function) and a method to constructively add neurons "on the fly" to overcome the problem of local minima in the uniform norm. This is a realization of the approximation results of Cybenko, Hecht-Nielsen, Hornik, Stinchombe, White, Gallant, Funahasi, Leshno et al and others. The approximation in the uniform norm is both more appropriate for a number of examples, such as robotic arm control, and seems to converge much faster than analogue descent measures in the average norm (e.g. back-propagation). The main novelty in the constructive proof is the use of a game-theoretic generalized derivative originally due to Danskin.

Introduction

There are gaps between the theoretical result ([1], [4], [6]) that a three level feed-forward neural network can approximate any reasonable function and the constructive learning methods (most notably "back-propagation") which give a gradient descent method to find the weights of the network based on examples [11], [7]. One such gap is that the theoretical results (*existence*) guarantee arbitrary approximation in the *maximal* error norm; while the second result (*constructive*) finds an answer by doing gradient descent in the *average* error norm. The descent methods also have the well-discussed problem of falling into local minima.

While the local minima problem is inherent in any descent method, the use of the average error norm was mostly chosen because it is a differentiable function; so the mathematical tools (e.g. chain rule) can be used to derive the back-propagation algorithm [11], [7]. The non-differentiability of the maximal error norm seems to a priori rule out using analytic gradient-style methods.

Nonetheless, the authors [9] have recently derived a constructive methodology for approximating in the maximal error norm; thereby in principle solving this problem. To do this, we adapted a more general notion of differentiation due to Danskin, developed in the context of game theory. Thus, in fact, we are able to do a form of gradient descent on the maximal error since it is differentiable in this more general sense.

We also approached the local minima problem. This is related to the correct choice of architecture (e.g. the number of "hidden level" neurons). We developed a method which allows one to "on-the-fly" add a new neuron when the descent has reached a local minimum in such a way as to always decrease the maximal error. In order to derive this method, we had to further assume that the function being approximated is twice differentiable.

These two results can be used jointly or separately. Together, they suggest the following method in principle for approximating functions in the uniform norm. First one chooses a network with a small number of hidden level neurons. One then applies the first result, modifying the weights in order to reduce the maximal error. Intuitively, one looks at the sample item(s) which produced the largest error; and modifies the weights to decrease this error. The result guarantees that this can be done without unduly increasing the errors on the other inputs. (It seems that this method converges fairly quickly.)

If, after convergence, the resulting maximal error is larger than desired (as a result of a local minimum in the uniform norm), one uses the second result to add a neuron and then continue with the descent.

The proofs of these results are a bit complicated ([9] contains full details; a preprint may be obtained from http://s11.haifa.ac.il/faculty.html and following the pointer to the author's home page); here we simply state the results. We point out, however, that it may be useful to look at the proofs as well, since the use of the Danskin derivative is a techniques which may be useful for other measures as well as for calculations involving fuzzy logic. (In this summary, we include the theorem (essentially due to Danskin) listing the crucial properties of the Danskin derivative.) We also have an additional result showing how to add a neuron even when not at a local minimum.

Note that this method is *not* a variant of back-propagation, but a different descent method. In the presentation here, one deals with all of the weights in the network (from all levels) at once and calculates a direction to decrease the maximal error directly from the known inputs that currently give the largest error. (In this paper, we developed the formulas for one hidden level, but they are directly extendible for an arbitrary number of levels.)

We also point out that we are simultaneously arranging to minimize the maximal error of each of the derivatives. However, our approach here is somewhat different than that of either [5] or [3] because we do not try to obtain the approximation of the derivative as the derivative of the neural network approximation; but rather as a direct approximation using additional outputs of the neural network. In this case, a major virtue of this approach is its simplicity; there is no need to deal with an analogue of Sobolev spaces appropriate for the maximal error norm. On the other hand, one needs to know all the derivatives at the sample data points.

Experimental results using this method are as yet spotty and we are not yet in a position to discuss the practicality of these algorithms. However, the preliminary results do indicate a fast convergence rate, which fits the intuition that if one can deal with the "outliers" one by one without allowing other sample points to deteriorate too much, then convergence should be quick.

The Theorems

These results are presented fully (with proofs) in [9]. We emphasize that all of the results are constructive; i.e. they translate directly to algorithms.

Theorem 1 is just a restatement of Cybenko's theorem [1]. Theorem 2 is a slight variant of Danskin's theorem which establishes the necessary generalized directional derivatives that are needed. Theorem 3 establishes how to constructively calculate the direction of descent. Theorem 4 is included for completeness, and shows how one can always increase the rate of descent even if one is not at a local minimum by increasing the dimension. Theorem 5 shows (using the result of Theorem 3) how to increase the dimension so that the error always decreases.

The only result presented here which requires differentiability of the function being approximated (in fact twice differentiability) is Theorem 5 which depends on the lemma (below) where the hypothesis is used. In particular the descent method holds without this hypothesis.

As stated above, our main new tool in this paper is the use of the Danskin derivative [2], which extends the notion of differentiability to include that of the maximal error function. This is quite natural because the Danskin derivative was developed in the context of game theory, and here we are trying to minimize a max function. This operator may have applications in the field of fuzzy logic where one often deals with max and min functions. [1]

We now fix some notation:

Let M_n be a compact connected subset of R^n. Suppose g is a function from M_n to R^m. If F is another function on the same domain and range, we can consider F as an approximation to g and measure the quality of approximation by using the Hilbert norm $\|g - F\|_h^2 = \int_{M_n} \sum_{i=1}^m [g_i(x) - F_i(x)]^2 dx$ to measures the *average* error; the quality of the *maximal* error is given by the uniform norm $\|g - F\|_1 = \max_{x \in M_n} \sum_{i=1}^m |g_i(x) - F_i(x)|$.

[1]Note that there are more general derivatives than the Danskin one, a fact which might be of interest in generalizations of our results; for this work the Danskin derivative suffices. On the other hand, note that one virtue of the Danskin derivative is that we are able to obtain a method for explicitly calculating the direction of descent, something that is lacking in more general methods such as distributions.

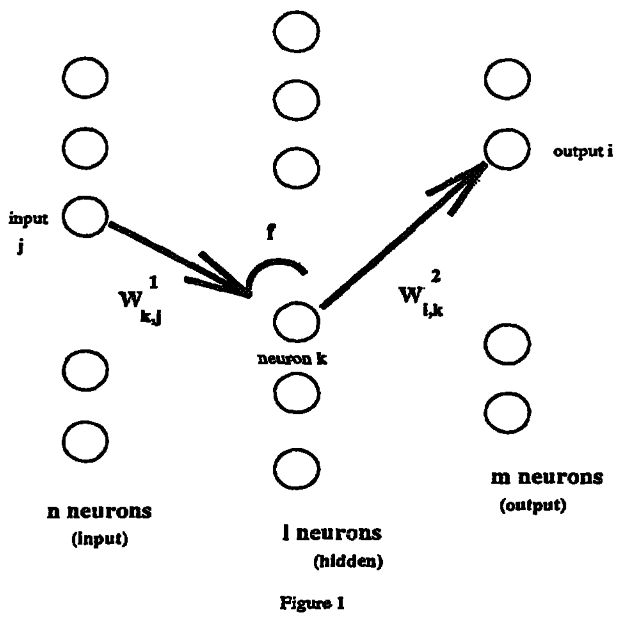

Figure 1: A 3-level Neural Network

Now assume that g is continuous and that F is generated by a neural network. To be specific, let us suppose that F is given by the smallest architecture feed-forward neural network that has the universal properties, [4], i.e. a three level neural network. (Let n the number of neurons on the first layer and m neu rons on the third layer and l neurons on the hidden layer.) Assume that each neuron in the hidden layer responds by a simple sigmoidal $f(t) = \frac{1}{1+e^{-t}}$. The third layer neurons combine their input linearly. Our constructive results are presented in this paper for such a network, but their generalization to networks with more levels or different combination functions is straightforward.

We can denote the weights by the matrices $W^{(1)}$ (a $l \times n$ matrix) for the weights between the firs t and second layer and $W^{(2)}$ (an $m \times l$ matrix) for the weights betwe en the second and the third layer and by an $l-$ length vector β the thresholds of the second layer. See figure 1.

Thus our function $F = (F_1, ...F_m)$ is given by the formula

$$F_s(x,y) = \sum_{t=1}^{l} W_{s,t}^{(2)} f(u_t(x,y)), \text{ where } u_t(x,y) = \sum_{j=1}^{n} W_{t,j}^{(1)} x_j + \beta_t$$

.

Theorem 1 *[1] For any continuous vector-function $g : M_n \to R^m$ and for any small $\epsilon > 0$ there exists a function F generated by the NN-method such that $\|g - F\|_1 < \epsilon$.*

The goal is to effectivize Cybenko's theorem, i.e. to show how to construct such approximation in the *uniform* norm. We repeat that previous constructive methods have worked in the Hilbert norm.

Denote by the vector y all of these free parameters; thus the parameters y are chosen from the space $R^{(m+n+1)l}$, i.e. $y = (y_1, \ldots, y_{(m+n+1)l})$. Note that the dimension of y is a function of l; i.e. we allow for the possibility that the number of hidden neurons changes.

Denote by $\Phi(x,y)$ and $\phi(y)$ the functions

$$\Phi(x,y) = \sum_{s=1}^{m} [g_s(x) - F_s(x,y)]^2 \text{ and } \phi(y) = \max_{x \in M_n} \Phi(x,y)$$

.

Thus, the problem is to find y, minimizing ϕ. (Since it is easy to see that there are positive constants C_1, C_2 such that $C_1\|g-F\|_1 \leq \phi(y) \leq C_2\|g-F\|_1$ it follows that ϕ as a norm is equivalent to the uniform norm.)

Theorem 1 shows that there exist choices of the parameter y (including the choice of l) so that $\phi(y)$ is arbitrarily small. One has to to show how to *constructively* find the y.

Let γ be a direction (i.e. a unit vector) in the space $R^{(m+n+1)l}$.

Definition: The *Danskin operator* in direction γ of $\phi(y)$, $(D_\gamma \phi)(y)$, is

$$(D_\gamma \phi)(y) = \lim_{\alpha \to 0^+} \frac{1}{\alpha} [\phi(y + \alpha\gamma) - \phi(y)]$$

.

The following result is an immediate variant of that originally due to Danskin [2].

Theorem 2 *Properties of the Danskin operator :*

(i) If $(D_\gamma\phi)(y)$ is negative at y^1, then $\phi(y)$ decreases in direction γ .

(ii) For each point y and each direction γ $(D_\gamma\phi)(y)$ exists.

(iii) $(D_\gamma\phi)(y)$ has the value

$$(D_\gamma\phi)(y) = \max_{z\in X(y)} < \Phi_y(z,y), \gamma > .$$

[Here $X(y) = \{x \in M_n : \Phi(x,y) = \phi(y)\}$ is the set of maximal points of the function $\Phi(x,y)$, $\Phi_y(z,y) = (\Phi_{y_1}(z,y), \ldots, \Phi_{y_{(m+n+1)l}}(z,y)) \in R^{(m+n+1)l}$ is a vector of partial derivatives $\Phi_{y_i}(z,y) = \frac{\partial\Phi}{\partial y_i}(z,y)$, and $< \Phi_y(z,y), \gamma >= \sum_{i=1}^{(m+n+1)l} \Phi_{y_i}(z,y)\gamma_i$ is a scalar product in the space $R^{(m+n+1)l}$.]

Corollary 1 *The Danskin operator $(D_\gamma\phi)(y)$ is a continuous function on the variables γ and y.*

The following theorem provides the constructive rule for our version replacing gradient descent.

Theorem 3 *Let y^1 be an arbitrary point in $R^{(m+n+1)l}$ with $\phi(y^1) > 0$ and let*

$r_i = \max_{z\in X(y^1)} \Phi_{y_i}(z,y^1)$, $q_i = \min_{z\in X(y^1)} \Phi_{y_i}(z,y^1)$,

$\sigma_1 = \sqrt{\sum_{r_i\le 0} q_i^2 + \sum_{q_i\ge 0} r_i^2}$ and $\sigma_2 = \sqrt{\sum_{r_i\le 0} r_i^2 + \sum_{q_i\ge 0} q_i^2}$.

Then exactly one of the following holds:

(i) y^1 is a local minimum point of $\phi(y)$;

(ii) the function $\phi(y)$ decreases in a direction γ^0 (we call it an antigradient of $\phi(y)$ in the point y^1) , which is defined by coordinates

$$\gamma_i^0 = \begin{cases} -\frac{q_i}{\sigma_1} & \text{if } r_i \le 0 \\ -\frac{r_i}{\sigma_1} & \text{if } q_i \ge 0 \\ 0 & \text{if } q_i < 0 < r_i \end{cases}$$

Theorem 4 *Suppose y^1 isn't a local minimum point of $\phi(y)$ in the space $R^{(m+n+1)l}$, and let γ^0 be an antigradient from the point y^1 . There exists a direction $\gamma^1 \in R^{(m+n+1)(l+1)}$ such that $(D_{\gamma^1}\phi)(y^1) < (D_{\gamma^0}\phi)(y^1)$. (This direction can be obtained constructively.)*

In order to consider the correct passage from a local minimum to a better approximation with an increase in size of the hidden layer, it is necessary to assume that g is twice differentiable. One then approximate g and its partial derivatives simultaneously. (This is necessary for the proof of the lemma that follows.)

Since the previous theorems of course hold in this case, this means that one has a constructive method for the simultaneous approximation in the uniform norm of both the function and its partial derivatives; under the assumption that the function is twice differentiable. (For a different kind of approximation to the function and its partial derivative see the papers [5] and [3].)

Lemma 1 *If y^1 is a local minimum point in the space $R^{(m+n+1)l}$ and $\phi(y^1) > 0$, then the set of maximal points $X(y^1)$ does not include an open subset.*

Theorem 5 *Let a vector y^1 be a local minimum point of the function $\phi(y)$ in the space $R^{(m+n+1)l}$ and let $\phi(y^1) > 0$. Then there exists a parameter $\mu > 0$, vectors $\vec{B} = (B_1, \dots, B_l) \in R^l$ with $B_i \geq 0$, $\vec{\lambda} = (\lambda_1, \dots, \lambda_m) \in R^m$ with $\|\vec{\lambda}\| = 1$, and $\vec{P} = (P_1, \dots, P_{n+1}) \in R^{n+1}$ such that a vector $\hat{y} \in R^{(m+n+1)(l+1)}$ with coordinates*

$$\hat{y}_i = \begin{cases} y^1_{i-s+1} - B_t\lambda_s & \text{if } i = (s-1)l + t \\ \mu\lambda_s & \text{if } i = s(l+1) \\ y^1_{i-m} & \text{if } i = m(l+1) + (n-1)t + j \\ P_j & \text{if } i = (m+n-1)(l+1) + j \\ y^1_{i-m-n} & \text{if } i = (m+n)(l+1) + t \\ P_{n+1} & \text{if } i = (m+n+1)(l+1) \end{cases}$$

satisfies the inequality $\phi(\hat{y}) < \phi(y^1)$. (These parameters are given constructively in the proof.)

In other words the new weights are given by

$$\hat{W}^{(2)}_{s,t} = \begin{cases} W^{(2)}_{s,t} - B_t\lambda_s & (\ s = 1, \dots, m; t = 1, \dots, l\ ) \\ \mu\lambda_s & (\ s = 1, \dots, m; t = l+1\ ) \end{cases}$$

About the Algorithms

The algorithms are described in the proofs given in [9]. (Space constraints preclude a complete description here.) There is a further paper planned [10] describing an implementation and experiments. The descriptions of the algorithms is complicated somewhat because of the necessity at each stage to worry about the possibility of multiple data giving the same maximal error. However, if we suppose that the maximal error always occurs at a singleton, then the descent method is very straight-forward.

To perform gradient descent (Theorem 3), one calculates the usual partial derivatives for each of the weight parameters of the error function Φ at the maximal input error point. Then one uses this gradient information to determine a direction to move. The existence of multiple maximal error points requires looking at the partial derivatives of each of these points. Thus this method is relatively simple to implement.

To add a neuron (Theorem 5) when one is in a (L_∞) minimum is somewhat more complicated. One first has to identify a "critical neuron" in the hidden level . Roughly speaking, one finds a neuron which behaves the most like another neuron in the hidden level for points in the domain. The new neuron is chosen then to mimic this critical neuron (i.e. $\vec{P}_0$ is the same as the weights to the critical neuron). All input weights and output weights from old neurons are kept the same, and only the output weight from the "critical neuron " is reduced. Unfortunately, the above is not sufficient to guarantee that an outgoing weight from the new neuron can be chosen so that the maximal error always decreases. (In the notation of Theorem 5, this is reflected in the choice of μ.) However, a secondary use of gradient descent using the Danskin derivative, modifying the choice of $\vec{P}$ and $\vec{B}$, guarantees the possibility of choosing such a μ giving such a decrease.

References

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

[2] J. Danskin. *The Theory of Max - Min* , Springer - Verlag, New - York, 1967.