Proceedings of IEEE WCCI ICNN'94 Vol. I. pp. 61--65, IEEE Inc., Orlando, Florida, 1994.
URL ftp://iserv.iki.kfki.hu/pub/papers/icnn94/olah.everyday.ps.Z
WWW http://iserv.iki.kfki.hu/adaptlab.html

Complexity of Learning: The Case of Everyday Neural Networks

B. Oláh and Cs. Szepesvári

***Abstract*— We have examined two slightly different domains of the learning problem. We have found a polynomial time result, and also an NP-completness result in these two domains. The domains are choosen so, that commonly used neural network architectures are included.**

I. Introduction

Nowadays artificial neural networks (ANNs) receive an increasing attention. However recent computer architectures do not allow yet the implementation of large ANNs. Thus it is an important question to examine how the learning time of ANNs scales respect to their size (and/or with the size of the tasks). Judd has introduced a computational framework for the learning problem [1] and proved, that learning in neural networks *in general* is too hard, i.e. in the worst case learning in neural networks is NP-complete. However, in his proof he restricts the domain of neural network architectures and tasks in such a way, that "everyday" neural network architectures, such as the one of the back-propagation algorithm, are not included in this domain. Consequently Judd's proof does not tell anything for these types of networks. In this article we present two straightforward proof about the complexity of learning for "everyday" ANNs. The first theorem says, that for extended binary tasks and in the worst-case, the problem is NP-complete, while the second says, that for binary tasks, there exist a polynomial time algorithm for loading. From these results, we conclude that for every neural network application domain one has to examine the time complexity of the specific learning problem.

II. Terminology

Here we consider a special case of learning in ANNs: namely supervised learning. In this scheme a network is presented with a set of input-output pairs, and it is the task of the network to reach to a configuration that allows it to establish a function which is compatible with the task. In this section we take some restrictions on the problems considered and will try to clarify the meaning of the above sentence.

A. Tasks

At first – as usual – we restrict ourselves to the class of *extended binary* $T(s,r)$ *tasks* (here s and r denote the stimulus and response size of the task respectively): such a task is a collection of SR-items, where each SR-item consists of a stimuli and a response strings of size s and r, respectively:

$$T \subset P(\{0,1\}^s \times \{0,1,*\}^r.^1$$

Here the symbol $*$ is the so called *"don't care" symbol.* For any x and y from $\{0,1,*\}$ we say that they are *compatible*, iff $x = y$ or $* \in \{x,y\}$. Further we say that strings p and q from $\{0,1,*\}^t$ are compatible, iff for every index i $(1 \le i \le r)$ the i^{th} character of p and the i^{th} character of q are compatible. We will denote this relationship by $p \approx q$. The *size of a task* is by definition $n+s+r$, where n is the number of elements of T. We will denote by $S(T)$ the set of all stimuli in T. Now let f be a function from $\{0,1\}^s$ to $\{0,1\}^r$. We may say that a task $T = T(s,r)$ is *compatible* with f, if for any pair (s,r) from T the relation $f(s) \approx r$ holds. A task T is said to be *well-defined*, if there exists a function f, such that f is compatible with T. The set of all extended binary tasks will be denoted by $\mathcal{T}^*$. A task is said to be strictly binary (or simply binary), iff it do not uses the don't care symbol. The set of all binary tasks will be denoted by $\mathcal{T}_0^*$.

B. Architectures

Our next restriction is on the class of neural network architectures. Here we consider just the *feed-forward* type architectures. Such an architecture may be represented as a tuple $A = (P,V,S,R,E)$, where P is the finite set of neurons, V is the set of configurable neurons, S is the set of input sites $(S = P - V)$, $R \subset P$ is the set of output sites, and

The authors are with the Attila József University of Szeged, Szeged, Hungary-6720

[1] $P(X)$ denotes the set of the subsets of X, i.e. the power-set of X.

E is the set of the (directed) connections:

$$E \subset \{(v_i, v_j) : v_i \in P, v_j \in V, i < j\},$$

where $\{v_1, v_2, \ldots, v_n\}$ is a suitable ordering of the elements of V. A family of architectures and the set of all architectures is denoted by $\mathcal{A}$ and $\mathcal{A}^*$, respectively.

B.1. FCL architectures

In this article we focus on a special case of feedforward networks, which are used in most of the applications. We shall call this family of architectures the FCL (fully connected layered) architecture family. The neurons of a network from this class are arranged in layers: there are no connections inside a layer, but the network is fully connected between consecutive layers. The first layer is the set of all input neurons S (this is the zero$^{\text{th}}$ layer), and the last is the set of all output neurons R. The size of an architecture is by definition $s+n$, where s and n denote the number of input sites and the number of configurable neurons, respectively. A FCL architecture is fully determined by its layer sizes: we will say that A is a FCL architecture of type $(n_0, n_1, \ldots, n_p)$ if there are p layers in the network, and the number of neurons in the layer i is n_i. Often n_i is said to be the size of the i^{th} layer. For $0 < i < p$, we say that the i^{th} layer is a hidden-layer of the network.

C. Working mechanism

The architecture of a network determines almost entirely its working mechanism, which is the following: at first a stimulus is presented to the network through its input sites. It will be the output of the input sites as well. In the following steps the neurons, whose input neurons have computed their outputs already in an earlier step, can compute their outputs. The process ends with the determination of the outputs of the output neurons, which will also be the output of the whole network. The next question is how a neuron computes its output.

D. Node functions

We assume, that the computation of every neuron may be represented as the computation of a function. The class of functions computable by the neurons in the network is the subject of a further restriction. In practice it is widely accepted, that the computing capacities of neurons is restricted to the computation of linearly separable functions. The class of these functions will be denoted by LSF. Such a function may be represented in the following manner: For every input site of the neuron there is a weight associated with it. When the neuron receives its inputs it computes the weighted sum of its inputs and compares the result to a threshold. If the result exceeds the threshold, the output of the neuron will be one, otherwise it will be zero. Another class, which is easy to manage in proofs, is the class of all Boole functions. This class will be denoted by LUF (as Look Up Functions).

E. Configured networks

The class of functions from which a network may choose its node-functions is called the *basis* of learning. We will denote bases by $\mathcal{F}$. A *configured network* is an architecture A together with a set $F = \{f_1, \ldots, f_n\}$, where f_i is the function computed by the i^{th} configurable node v_i from V. We say that an architecture A is configured using the basis $\mathcal{F}$, if every function, f, from the configuration F of A is the element of $\mathcal{F}$. We denote by $\mathcal{C}_{\mathcal{F}}(A)$ the set of all possible configurations of an architecture A using the basis $\mathcal{F}$.

F. Task performed by a network

To complete our framework we define the meaning of "a task is learned (or performed) by an architecture." A configured network determines a function from its stimulus space to its response space. Let (A, F) be a configured network, and let us denote this function by $\mathcal{M}_F(A)$. We say, that a configured network (A, F) performs the task T, iff $\mathcal{M}_F(A)$ is compatible with T. This relationship will be denoted by $A <_F T$. A task is said to be *performable* by A using the basis $\mathcal{F}$, if there is a configuration F from $\mathcal{C}_{\mathcal{F}}(A)$, such that (A, F) establishes T. This relationship will be denoted by $A <_{\mathcal{F}} T$.

G. The computational problem

The *loading problem* (or learning problem) is the following: Fix a domain $\mathcal{D} \subset \mathcal{A}^* \times \mathcal{T}^*$, and a basis $\mathcal{F}$. For an arbitrary pair (A, T) from $\mathcal{D}$ find a configuration F from $\mathcal{C}_{\mathcal{F}}(A)$, such that $A <_F T$. This is a search problem. The *performability* problem is the appropriate decision problem: in this case on a fixed domain $\mathcal{D}$ and for a fixed basis $\mathcal{F}$ one has to decide for any given pair (A, T) from $\mathcal{D}$ whether $A <_{\mathcal{F}} T$ or not. The performability problem is the problem of recognizing the following (parametrized) language:

$$\mathrm{Perf}_{(\mathcal{D},\mathcal{F})} = \{(A, T) \in \mathcal{D} : A <_{\mathcal{F}} T\}.$$

Clearly if an algorithm can solve the search problem, it is also possible to solve the decision problem using the same algorithm. Thus the search problem is at least as hard as the decision problem.

In the next section we will focus on the loading problem of FCL networks using the basis LUF. These results show, that the loading problem for this domain is NP-complete.

III. Loading with LUF functions

Fix the basis LUF. First, it is easy to check that the loading of FCL networks having only one layer is easy: it can be solved in time $O(n \times |T|)$, where n is the number of output neurons of the network and $|T|$ denotes the number of elements of task T.

A. Compatible groupings

We begin with an observation: Let us suppose, that the task T has been loaded into a multi-layered FCL network. Let us take a look on an intermediate layer of the network. It is clear, that for every pair of stimuli s_1 and s_2, for which responses r_1 and r_2 are different, the outputs of the neurons in the examined layer are different for the two cases. On the other hand let us take an arbitrary output-scene of that layer, and the set of stimuli for which this scene appears on this layer. It is clear again, that for every pair of stimuli from this set, every appropriate response pair has to be the same. These comments motivate the definition of *compatible groupings*:

Let us fix a task T, and let $S = S(T)$. Then $\{S_0, S_1, \ldots, S_{k-1}\} \subset P(S)$ is said to be a compatible grouping of T, iff $\cup_i S_i = S$, S_is are pairwise disjuncts, and for an arbitrary index i and stimuli $s_1, s_2 \in S_i$, and for any r_1 and r_2 for which $(s_1, r_1) \in T$ and $(s_2, r_2) \in T$ the relation $r_1 \approx r_2$ holds. Number k is said to be the size of the compatible grouping.

B. Results

Let us denote the minimum size of compatible groupings of T by $k(T)$. For the sake of compactness let us denote by len_2 the function $[\log_2(n)] + 1$, where $[x]$ denotes the integer-part of the real number x. Furthermore let us denote by b_s the function, which maps the set $\{0, 1, \ldots, 2^s - 1\}$ to its binary representation. Both len_2 and b_s are computable in polynomial time and size of their arguments. By definition the *dimension* of a task T means the number $\mathrm{len}_2(k(T) - 1)$. (If $k(T) < 2$ then we redeclare the dimension of T to be 0.) Our above observations show, that for any given basis $\mathcal{F}$ from $A <_{\mathcal{F}} T$ it follows, that $\dim(T)$ is less then or equal to the minimum of the neuron-number of the hidden layers of the network A. For $\mathcal{F} = \mathrm{LUF}$ the reverse statement is also true. We will prove it in two main steps.

Theorem III.1 *Let A be a 2-layered FCL network. If $\dim T \leq n$, where n is the number of neurons in the middle layer, than the task T is performable by A using the basis* LUF .

Proof. Let A be a network of type (s, n, r) and T be a task, for which there exists a task T whose dimension is less then or equal to n. Let this grouping be, say, $\{S_0, S_1, \ldots, S_{k-1}\}$, where $\mathrm{len}_2(k-1) \leq n$. Let A_1 and A_2 be the FCL networks of type (s, n) and (n, r), respectively. Furthermore let us decompose the task T into two subtasks T_1 and T_2, where

$$T_1 = \{(s, b_n(i)) : s \in S_i\}$$

and

$$T_2 = \{(b_n(i), r) : \exists s \in S_i, (s, r) \in T\}.$$

It is an easy task to check, that T_1 and T_2 are well defined tasks, and $A_1 <_{\mathrm{LUF}} T_1$ and $A_2 <_{\mathrm{LUF}} T_2$. If $F_i \in \mathcal{C}_{\mathrm{LUF}}(A_i)$ denotes the configurations, for which $A_i <_{F_i} T_i$ $(i = 1, 2)$, then it is clear that for the joint configuration $F = F_1 \cup F_2$ the relation $A <_F T$ holds. □

Let us remark, that for the above defined T_2 it is true, that $\dim T_2 \leq \dim T$. To prove this, it is enough to check, that the sets $S_i' = f_1(S_i)$ forms a compatible grouping of T_2, where f_1 denotes an arbitrary fixed function which is compatible with T_1. Now the following theorem follows immediatly:

Theorem III.2 *Let A be an FCL network, having at least two layers. Then for any given task T, $A <_{\mathrm{LUF}} T$ if and only if $\dim T \leq n$, where n is the minimum of the element number of the inter-layers of A.*

Proof. The necessity of the performability condition has been proven already. Now we will prove the other direction by induction of the number of layers in the network. For 2-layered networks we have seen the result already. Now, let us pick a p-layered network (with $p > 2$) of type $(n_0, n_1, \ldots, n_p)$, and let $n = \min_{0<i<p} n_i$. Suppose, that we have proven the statement for $(p-1)$-layered networks. Let $\{S_0, \ldots, S_{k-1}\}$ be the compatible grouping of T, for which $\mathrm{len}_2(k-1) \leq n_1$. Let A_1 and A_2 be FCL networks of type (n_0, n_1) and $(n_1, n_2, \ldots, n_p)$, respectively. Define the decomposition of T into $T_1 = T_1(n_0, n_1)$ and $T_2 = T_2(n_1, n_p)$ as above. It is clear that T_1 and T_2 are well defined, and if $A_i <_{\mathrm{LUF}} T_i$ for $i = 1, 2$, then $A <_{\mathrm{LUF}} T$. Thus, again, it is enough to prove, that T_1 and T_2 are performable by A_1 and A_2, respectively. Naturally $A_1 <_{\mathrm{LUF}} T_1$ as A_1 is a one-layered network. To prove, that $A_2 <_{\mathrm{LUF}} T_2$ consider the following inequality series:

$$\dim T_2 \leq \dim T \leq \min_{0<i<p} n_i \leq \min_{1<i<p} n_i.$$

This, together with the inductive assumption yields the desired result. □

This result shows, that it is enough to concentrate on the hardness of the questions of type $\dim T \leq n$.

It is easy to show, that for $n > 1$ the problem of deciding for any task T, whether $\dim T \leq 1$ is true or not, is NP-complete. (For $n = 1$, the problem is solvable in polynomial time.) More precisely:

Theorem III.3 *For $p > 2$ the problem of deciding for any task T, whether $k(T) \leq p$ is true, is NP-complete. For $p = 2$, the problem is solvable in polynomial time.*

Proof. By reduction from the problem of p-colorability of graphs. The proof is omitted. The interested reader is referred to [2]. □

Theorem III.4 *Loading of FCL networks, that has at least two layers, using the basis* LUF *is NP-complete. More precisely if $\mathcal{A}_i$ denotes the set of all FCL networks, that has exactly i layer and in any of their hidden layers has at least two neurons, then for any domain $\mathcal{D}$, which includes any of the sets $\mathcal{A}_i \times \mathcal{T}^*$ as a subset, the problem $Perf_{(\mathcal{D},\mathrm{LUF})}$ is NP-complete.*

Proof. By reduction from the problem of compatible groupings. The proof can be found in [2]. □

IV. Loading with LSF functions

The previous theorem cannot be applied directly to the loading problem of FCL networks using LSF functions. The reason is, that we have used intensively the fact, that a one-layer network, using the basis LUF , can load any extended binary task T within polynomial time. This is not the case when only the basis LSF is enabled. But the following observation will help:

Proposition IV.1 *Let $T = T(s, r)$ be an arbitrary, fixed task. Then there exists a polynomial P, such that for the FCL network A of type $(s, P(t+s+r), r)$, where $t = |T|$, it is true, that $A <_{\mathrm{LSF}} T$.*

Proof. The outline of the proof is the following: at first we will prove the result for tasks of type $T(s, 1)$. For such a task it is enough to use a FCL network of type $(s, t, 1)$. In order to show this, let us consider the conjunctive normal form corresponding to task T. This has at most t clauses. It is easy to see, that disjunctions and conjunctions of any variables may be represented by a single neuron equipped with an function form LSF . Thus any clause of the conjunctive normal form may be represented by a single neuron of the hidden layer, and the conjunction of clauses can be represented by the output neuron as well. We get that for this special task and network $A <_{\mathrm{LSF}} T$. For a task $T = T(s, r)$ simply put together r disjunct networks of type $(s, t, 1)$. It is clear that the resulted network of type (s, rt, r) can perform the task T. □

Now, using the same argument as of Theorem III. 4 it is straightforward, that

Theorem IV.1 *The performability problem of the FCL networks, that has at least 3 hidden layer, is NP-complete.*

In most of the applications networks having only one or two hidden layers are used. Unfortunatly, for networks, that has two hidden layers the corresponding performability problem is NP-complete too. To prove this, we need the following proposition:

Proposition IV.2 *For any task $T = T(2, t)$, for which $|S(T)| \leq 3$, it is true, that for the FCL network A of type $(2, t)$ $A <_{\mathrm{LSF}} T$.*

Proof. The reason of it is, that the only function in 2-dimensions, that is not linearly separable, is the function XOR. Since any task, which has at most 3 elements can be extended to a function which is not the function XOR, thus such tasks are compatible with an LSF function. □

Theorem IV.2 *The performability problem of FCL networks having exactly two hidden layers, is NP-complete, even for the basis* LSF*.*

Proof. By reduction from the problem of compatible groupings. We know, that for $p = 3$, and for any task T, deciding whether $k(T) \leq 3$ or not, is NP-complete. Let $T = T(s, r)$ be any task, and let A_T be the FCL network of type $(s, P(s, 2, t), 2, r)$, where $t = |T|$ and P is the polynomial defined in Proposition IV.1. Now, we want to show, that $k(T) \leq 3$ if and only if $A_T <_{\mathrm{LSF}} T$. As usual let us decompose the task T into subtasks T_1 and T_2, and the network A into FCL networks, A_1 and A_2, of type $(s, P(s, 2, t), 2)$ and $(2, r)$, respectively (see Theorem III.1 and III.2). Since $|T_1| \leq |T|$, $A_1 <_{\mathrm{LSF}} T_1$ holds, clearly, according to Proposition IV.1. Thus it is sufficient to prove, that $k(T) \leq 3$ is equivalent to $A_2 <_{\mathrm{LSF}} T_2$. But observe, that by the definition of T_2 $|S(T_2)| = k(T)$, and using Proposition IV.2 this yields to the desired result. Note, that A_T can be constructed in polynomial time in the size of the task T, which completes the proof. □

V. Binary tasks: loading in polynomial time

The hardness of problem of finding a compatible grouping for a task T vanishes, if we restrict the set of tasks to (strictly) binary tasks. It is clear, that for any binary task, there is polynomial algorithm, which finds the least compatible grouping: since for two binary response strings it is not questionable, that the corresponing stimuli should be in the same group or not. Clearly if and only if the two response strings are identical, the corresponding stimuli has to be in the same group. Thus $k(T)$, in this case, reduces to the number of different response strings contained in T. The following theorem is an immediate consequence of this argument:

Theorem V.1 *For the basis* LUF *loading binary tasks into FCL networks of any type can be solved in polynomial time. The same statement holds for the basis* LSF *and one layer FCL networks.*

Proof. At first let us prove the first part of the theorem. For this, let us fix the basis LUF. For FCL networks, that has only one layer, it has been proven already, that loading of any (even extended) binary task can be done in polynomial time, respect to the size of the task and the size of the network. For multi-layered networks, the statement follows from the fact, that for binary tasks, finding an optimal compatible grouping of T can be done in polynomial time, respect to the size of the task. Then loading (if the task is performable by the architecture A) can be done by recursively splitting the task into subtasks and loading these into one-layered parts of A.

Now, we will prove the second part of the statement. Fix the basis LSF. We note, that for one-layered FCL networks, the statement can be reformulated as a set of linear inequalilities. (For a task $T = T(s, 1)$ of size t, the number of inequalities will be t, and the number of variables will be $s + 1$ corresponding to the weights and the threshold of the LSF function. Thus, for a task of type $T = T(s, r)$ the number of inequalities and variables is polynomial, respect to the size of the task.) It is known, that such an inequality system can be solved within polynomial time (or it can be shown, also in polynomial time, that no solution exists for it). Thus for one-layered FCL networks loading is polynomial. □

Note, that for multi-layered FCL networks this argument fails since finding optimal compatible groupings for the basis LSF is known to be NP complete.

In order to rephrase why learning is hard for extended binary tasks let us examine the learning problem, for extended binary task. Pick a task T, and a SR-item of it, say (s, r), such that r contains at least one don't care symbol. "Showing" this item to the network means, that output neurons, for which on the appropriate position in string r the don't care symbol $*$ stands, are not told, what to learn. They have the freedom to find out what to do. This liberty is, what makes learning difficult.

VI. Conclusion

We emphasize, that Judd's proof of NP-completness is a worst-case complexity result on a specific domain. To make it clear, first we refined the notation of performability. Then we have examined the domain of "everyday", intra-layer fully connected neural networks. Judd's proof does not extends to this case, since in his domain there are only "rarely" connected neural networks. One may think, that enabling more connections makes the problem easier. But this is just one part of the story. We have found two mainly different results for these type of networks, according to the set of tasks enabled: For the set of extended binary tasks and for the set of strictly binary tasks the loading problem is NP-hard and is solveable in polynomial time, respectively. The only point, which remained questionable is the loading of extended binary tasks into two layered FCL networks, using basis LSF. Our conjencture is, that this is a NP-complete problem too.

Acknowledgements

We thank for P.Hajnal for helpful discussions, that resulted in the formulating of the basic NP-completness Theorem III.3.

References

[1] J. Judd, *Neural Network Design and the Complexity of Learning.* A Bradford Book, MIT Press, Cambridge, Massachuttes, 1990.

[2] B. Oláh, *The complexity of learning in neural networks.* Master's thesis, Attila József University of Szeged, 1993. in Hungarian.