Available online at www.sciencedirect.com

SCIENCE DIRECT®

Nonlinear Analysis: Real World Applications 7 (2006) 937–942

Nonlinear Analysis
Real World Applications

www.elsevier.com/locate/na

Studying the performance of artificial neural networks on problems related to cryptography

E.C. Laskari[a, b, *], G.C. Meletiou[c], D.K. Tasoulis[a, b], M.N. Vrahatis[a, b]

[a]*Computational Intelligence Laboratory, Department of Mathematics, University of Patras, GR-26110 Patras, Greece*
[b]*University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece*
[c]*A.T.E.I. of Epirus, P.O. Box 110, GR-47100 Arta, Greece*

Received 1 April 2005; accepted 5 December 2005

Abstract

Cryptosystems rely on the assumption that a number of mathematical problems are computationally intractable, in the sense that they cannot be solved in polynomial time. Numerous approaches have been applied to address these problems. In this paper, we consider artificial neural networks and study their performance on approximation problems related to cryptography.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Artificial neural networks; Discrete logarithm; Diffie–Hellman problem; Factorization; Approximation

1. Introduction

Public key cryptography has motivated a number of hard and complex mathematical problems. These problems are related to computational algebra (finite fields, finite groups, ring theory), computational number theory, theory of probability, logic, Diophantine's complexity, and algebraic geometry among others. The efficiency of the contemporary cryptosystems relies on the assumption that these problems are computationally intractable, meaning that their computation cannot be completed in polynomial time. Such problems are factorization [29], the discrete logarithm [1,25,27] and the knapsack problem [20]. Numerous techniques have been proposed to tackle these problems including algebraic and number theoretic methods, software oriented methods and interpolation techniques (see [4,14,16,22,23,33,34]).

In this paper we consider the artificial neural networks (ANNs) approach and study their performance on problems from the field of cryptography. Specifically, we study the approximation of three problems. The first problem under consideration is the *discrete logarithm problem* (DLP) [1,24,25,27], and is defined as follows: compute the smallest integer $x > 0$ satisfying the relation $g^x = h$, where g, h are elements of a finite cyclic group generated by g. The security of various public-key and private-key cryptosystems such as the *Diffie–Hellman exchange protocol* [5,24], the *El Gamal public key cryptosystem*, as well as, *the El Gamal digital signature scheme* [6], is based on the assumption that DLP is computationally intractable [1,4,15,16,22,23,25,27,33,34]. In the case of a finite field of prime order p a primitive element α modulo p is fixed. We assume that u is the smallest nonnegative integer with $\alpha^u \equiv y \pmod{p}$. Then u is called the *index*, or the *discrete logarithm* of y.

* Corresponding author. University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR-26110 Patras, Greece.
E-mail address: elena@math.upatras.gr (E.C. Laskari).

1468-1218/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.nonrwa.2005.12.002

The second problem at hand is the *Diffie–Hellman key-exchange protocol problem* (DHP) [3,4,7,14,33]. Let G be a finite cyclic group of order $\|G\| = m$. Consider g a generator of G; x, y integers satisfying $0 \leqslant x, y \leqslant m-1$ be the private keys of two users, and $u, v \in G$, with $u = g^x$, $v = g^y$ stand for the corresponding public keys. Then the problem amounts to computing g^{xy} from u and v, where g^{xy} is the symmetric key for secret communication between the two users. Consider the special case of the DHP, where $u = v$. The term Diffie–Hellman mapping refers to the function: $g^x \longmapsto g^{x^2}$. The definition of the *Diffie–Hellman mapping problem* (DHMP) follows naturally from the previous definition of DHP.

Finally, we study the *factorization problem related to the RSA cryptosystem* [29]. For the RSA cryptosystem to be secure, the factorization of $N = p \cdot q$ must be computationally intractable. The cryptanalyst has to compute p or q from N. To achieve this it is sufficient to obtain $\phi(N)$ from N, or equivalently to factorize N, since $\phi(N) = (p-1)\cdot(q-1)$ [19,29].

2. Approximation through artificial neural networks

ANNs have been motivated by biological systems and more specifically by the human brain. Formally, "an ANN is a massively parallel distributed processor made of simple processing units, the neurons, which has a natural propensity for storing experiential knowledge and making it available to use. It resembles the brain in two ways. First, knowledge is acquired by the network from its environment through a learning process, and second, interneuron connection strengths, called weights, are used to store the acquired knowledge" [8].

Each artificial neuron is characterized by an input/output (I/O) relation and implements a local computation. The output of any unit is determined by its I/O characteristics, its interconnection to other units, and possibly external inputs. The overall functionality of a network is determined by its topology, the training algorithm applied, and its neuron characteristics. In this paper from the wide family of neural network types, we focus on feedforward neural networks (FNNs). In FNNs all neuron's connections lead to one direction and the neurons can be partitioned in layers. In detail, the inputs form an input layer, while the output neurons form the output layer. All other neurons are assigned to a number of hidden layers. Each neuron in a layer is fully connected to all the neurons of the successive layer. The function of the hidden layers of neurons is to intervene between the network's input and output in a useful manner, as adding hidden layers on a network enables it to extract high-order statistics [8]. This kind of networks can be described with a series of integers that denote the number of neurons at each layer. For example, a network with a topology $2-3-3-1$ is a network with two inputs at the input layer, two hidden layers with three neurons each, and an output layer with a single neuron.

The operation of such networks consists of iterative steps. The input layer neurons are generally assigned to real inputs, while the remaining hidden and output layer neurons are passive. In the next step the neurons of the first hidden layer collect and sum their inputs and compute their output, which is applied to the second layer. This procedure is propagated to all layers until the final outputs of the network are computed.

The computational power of neural networks derives from their parallel distributed structure and their inherent ability to adapt to specific problems, learn and generalize. These characteristics provide ANNs the ability to solve complex problems. In [9,35] the following statement has been proved: "Standard feedforward networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy". It has also been proved in [26] that a single hidden layer feedforward network with r units in the hidden layer, has a lower bound on the degree of the approximation of any function. The lower bound obstacle can be alleviated if more than one hidden layers are used. Mairov and Pincus in [26] have proved that, on the unit cube in $\mathbb{R}^n$ any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having $2n+1$ units in the first layer and $4n+3$ units in the second. This implies that any lack of success in applications can be attributed to inadequate training, an insufficient number of hidden units, or the lack of a deterministic relationship between input and target. Our ambition is to use this powerful computational tool in the case of the discrete algebraic structures appearing in cryptology and derive similar results.

The training process involves modification of the network weights by presenting to it training samples, called patterns, for which the desired outputs are a priori known. The ultimate goal of training is to assign to the weights (free parameters) of the network W, values such that the difference between the desired output (target) and the actual output of the network is minimized. The adaptation process starts by presenting all the patterns to the network and computing a total error function $E = \sum_{k=1}^{P} E_k$, where P is the total amount of patterns used in the training process and E_k is the partial network error with respect to the kth training pattern, computed by summing the squared discrepancies between the actual network outputs and the desired values of the kth training pattern.

The training patterns can be applied several times to the network but in a different order. Each full pass of all the patterns that belong to the training set, T, is called a *training epoch*. If the adaptation method succeeds in minimizing the total error function then it is obvious that its aim has been fulfilled. Thus training is a nontrivial minimization problem. The most popular training method is *back propagation* method [8], which is based on the steepest descent optimization method. The back propagation learning process applies small iterative steps which correspond to the training epochs. At each epoch t the method updates the weight values proportionally to the gradient of the error function $E(w)$. The whole process is repeated until the network reaches a steady state, where no significant changes in the weights are observed, or until the overall error value drops below a predetermined threshold. At this point we conclude that the network has learned the problem "satisfactorily". The total number of epochs required can be considered as the speed of the method. More sophisticated training techniques can be found in [8,10–13,28,32].

3. Experimental setup and results

Training methods: The ANNs training methods considered in this study were the standard back propagation (BP) [30], the back propagation with variable stepsize (BPVS) [12], the resilient back propagation (RPROP) [28], the on-line adaptive back propagation (OABP) [11] and the scaled conjugate gradient (SCG) method [21]. All methods were extensively tested with a wide range of parameters. In most of the testing cases, the training methods did not exhibit significantly different performance, except from the standard BP, which encountered difficulties in training most of the times.

Network architecture: The definition of an "optimal" network architecture for any particular problem is quite difficult and remains an open problem. To this end we tested a variety of topologies with different number of hidden layers and with various numbers of neurons at each layer. The results reported are the best results obtained for each problem.

Normalization: To make the adaptation of the network easier, the data are transformed through the normalization procedure, that takes place right before training. Assuming that the data presented to the network are in $\mathbb{Z}_p$, where p is prime, the space $S = [-1, 1]$, is split in p subspaces. Thus, numbers in the data are transformed to analogous ones in the space S. At the same time, the network output is transformed to a number within $\mathbb{Z}_p$ using the inverse operation.

Evaluation: To evaluate the network performance we first measured the percentage of the training data, for which the network was able to compute the exact target value. This measure is denoted by μ_0. However, as network output was restricted within the range $[-1, 1]$, very small differences in output, rendered the network unable to compute the exact target but rather to be very close to it. This fact resulted in the insufficiency of the μ_0 measure as a performance indicator. Thus we employed the $\mu_{\pm v}$ measure. This measure represents the percentage of the data for which the difference between desired and actual output does not exceed $\pm v$ of the real target.

We note that the "near" measure, $\mu_{\pm}$, has different meaning for the DLP and the DHMP. The "near" $\mu_{\pm}$ measure is very important in the case of the DLP. If the size of the $(\pm v)$ interval is $O(\log(p))$ then the "near" measure can replace the "complete" μ_0 one. In general, for some "small" values of v the "near" measure is acceptable since the discrete logarithm computation can be verified i.e. computation of exponents over finite fields [27]. However, the verification of the DH-mapping is an open problem (the DHDMP). Sets of possible values for the DH-mapping can be used to compute sets of possible values for the DH-key. The values of the DH-key can be tested in practice; they are symmetric keys of communication between the two users. The percentage of success for the "near" measure for DHMP can be compared with the corresponding percentage for the DLP. The results of the comparison can be related to the conjecture that the two problems are computationally equivalent.

3.1. *Studying the discrete logarithm problem and the Diffie–Hellman mapping problem*

In previous work, we have tested both DLP and DHMP for several small primes p [17,18]. The ANNs in this case succeeded in training and generalizing, reaching up to 100%. Next, larger primes were tested rendering the task of training networks harder. Having so many numbers normalized in the range $[-1, 1]$ posed problems for the adaptation process. Thus, small changes in the network output caused complete failure, raising the need for larger architectures. In

Table 1
Results for networks trained on the DLP and DHMP

p	Topology	Epochs	$\mu_0(\%)$	$\mu_{\pm 2}(\%)$	$\mu_{\pm 5}(\%)$	$\mu_{\pm 10}(\%)$	Problem
83	$1-5-5-1$	20 000	20	30	48	70	DLP
	$1-5-5-1$	20 000	20	35	51	70	DHMP
97	$1-5-5-1$	25 000	20	30	48	70	DLP
	$1-5-5-1$	20 000	20	35	51	70	DHMP

Table 2
Results for the second setting of the DLP

Range of p	Topology	Epochs	$\mu_{\pm 0}(\%)$	$\mu_{\pm 15}(\%)$	$\mu_{\pm 20}(\%)$	$\mu_{\pm 30}(\%)$	$\mu_{\pm 40}(\%)$
101–199	$2-15-1$	600 000	100	100	100	100	100
503–1009	$2-25-1$	600 000	82	93	96	96	98
1009–2003	$2-30-1$	600 000	17	40	46.7	51.8	54.1
1009–2003	$2-3-3-3-1$	20 000	7.5	34.3	44.8	64.2	71.6

Table 3
Results for networks trained for the $\phi(N)$ mapping with $N = p \cdot q \leqslant 10^4$

Topology	Epochs	$\mu_0(\%)$	$\mu_{\pm 2}(\%)$	$\mu_{\pm 5}(\%)$	$\mu_{\pm 10}(\%)$	$\mu_{\pm 20}(\%)$
$1-5-5-1$	80 000	3	15	35	65	90
$1-7-8-1$	50 000	6	20	50	70	100

cases with very large primes, the network performance on training was very poor. Some indicative results on training are reported in Table 1.

In this contribution, the DLP is studied in a different setting. More specifically, we study here the case where, for several values of the prime p and the primitive element α, the value of $y = \alpha^x \pmod p$, remains fixed. We have tested the DLP in this setting for p assuming values between 101 and 2003, with several network topologies and training methods. In this case, there was a differentiation among the results obtained by different methods. For small values of p, i.e. from 101 to 199, the best results on the approximation of x, were obtained by the AOBP method. For larger values of p, the best results were given by the SCG method. Results on this new setting are reported in Table 2. All these results refer to training the ANNs on the approximation of the value of discrete logarithm x. Comparing the results exhibited in Tables 1 and 2, it seems that for the DLP problem, the approximation capability of the FNNs is better in the new setting.

3.2. *Addressing the factorization problem*

The ability of neural networks to address the RSA cryptosystem has also been investigated. In our previous work, we tried to approximate the $\phi(N)$ mapping $(p \cdot q \to (p-1) \cdot (q-1))$, with input patterns being numbers $N = p \cdot q$, where p and q are primes, and as target patterns the $\varphi(N) = (p-1) \cdot (q-1)$ numbers. In this case the results were different. The normalization problem ceased to be an obstacle. What is really intriguing in this case is the generalization performance of the networks. Clearly, the networks were able not only to adapt to the training data, but also to achieve very good results with respect to the test sets [18]. Indicating results on training are exhibited in Table 3.

In this study, the factorization problem is also viewed with a different setting. More specifically, approximating the value of the function $p^2 + q^2$, given the value of N, leads directly to the factorization of N to its factors p and q. Thus, we tested the ANNs on the approximation of the aforementioned function for several instances of N. The results for this problem are reported in Table 4.

Table 4
Results for the second setting of the factorization problem for N between 143 and 1003

Topology	Epochs	$\mu_{\pm 0}(\%)$	$\mu_{\pm 15}(\%)$	$\mu_{\pm 20}(\%)$	$\mu_{\pm 30}(\%)$	$\mu_{\pm 40}(\%)$
$1-15-1$	200 000	35.1	36.8	42.1	43.8	45.6
$1-20-1$	600 000	35.1	43.8	45.6	52.6	56.2

Although the two settings of the factorization problem are computationally equivalent the approximation capabilities of the FNNs for this problem seems to be better for the first setting.

4. Discussion and concluding remarks

In previous work [17,18], we have studied the neural network approach to address the discrete logarithm problem (DLP), the Diffie–Hellman mapping problem (DHMP), and the factorization problem related to the RSA cryptosystem. It is known that if a method for computing indices over finite fields is available, then the RSA cryptosystem will break. In other words, the DLP is no easier than the factorization problem related to the RSA. Our experimental results conform to this conclusion.

In this paper, we extend our previous results by studying the performance of ANNs on the same problems but in a different setting. Concerning the DLP, ANNs are used in the case where the prime number p and the primitive element modulo p vary. Therefore the basic algebraic structure (the finite field) changes. Concerning the factorization problem for the RSA, in addition to $\phi(N)$, an other function is tested.

In general, the problem can be considered solved if the measure μ_0 is 100% and the architectural topology is small enough (the number of weights is $O(\log(p))$. On the other hand, measures like $\mu_\pm$ seem promising in the sense that although the exact target might not be accomplished, the output of the network is indeed close to that. Thus, with a predetermined number of trial and error procedures the problem can be resolved. The "near" $\mu_\pm$ measure is very important in the case of the DLP. If the size of the $(\pm v)$ interval is $O(\log(p))$ then the "near" measure can replace the "complete" μ_0 one. In general, for "small" values of v the "near" measure is acceptable since the discrete logarithm computation can be verified (computation of exponents over finite fields).

Here we consider only artificial feedforward neural networks. In a future correspondence we intend to apply various other networks and learning techniques including nonmonotone neural networks [2], probabilistic neural networks [31], self-organized maps [10], recurrent networks and radial basis function networks [8].

Acknowledgements

The authors would like to thank the reviewers for their concern. Moreover, they acknowledge the partial support by the "Archimedes" research programme awarded by the Greek Ministry of Education and Religious Affairs and the European Union.

References

[1] L. Adleman, A subexponential algorithm for discrete logarithm problem with applications to cryptography, 20th FOCS, 1979, pp. 55–60.
[2] B. Boutsinas, M.N. Vrahatis, Artificial nonmonotonic neural networks, Artif. Intell. 132 (2001) 1–38.
[3] R. Canetti, J. Friedlander, I. Shparlinski, On certain exponential sums and the distribution of Diffie–Hellman triples, J. London Math. Soc. (2) 59 (3) (1999) 799–812.
[4] D. Coppersmith, I. Shparlinski, On polynomial approximation of the discrete logarithm and the Diffie–Hellman mapping, J. Cryptology 13 (2000) 339–360.
[5] W. Diffie, M. Hellman, New directions in cryptography, IEEE Trans. Inf. Theory 22 (1976) 644–654.
[6] T. El Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inf. Theory 31 (1985) 469–472.
[7] E. El Mahassni, I. Shparlinski, Polynomial representations of the Diffie–Hellman mapping, Bull. Aust. Math. Soc. 63 (3) (2001) 467–473.
[8] S. Haykin, Neural Networks, Macmillan College Publishing Company, 1999.
[9] K. Hornik, Multilayer feedforward networks are universal approximators, Neural Networks 2 (1989) 359–366.
[10] T. Kohonen, Self-Organized Maps, Springer, Berlin, 1997.

[11] G.D. Magoulas, V.P. Plagianakos, M.N. Vrahatis, Adaptive stepsize algorithms for on-line training of neural networks, Nonlinear Anal. 47 (5) (2001) 3425–3430.

[12] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Effective backpropagation training with variable stepsize, Neural Networks 10 (1) (1997) 69–82.

[13] G.D. Magoulas, M.N. Vrahatis, G.S. Androulakis, Increasing the convergence rate of the error backpropagation algorithm by learning rate adaptation methods, Neural Comput. 11 (7) (1999) 1769–1796.

[14] U. Maurer, S. Wolf, The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms, SIAM J. Comput. 28 (1999) 1689–1721.

[15] G. Meletiou, Explicit form for the discrete logarithm over the field GF(p, k), Arch. Math. Brno 29 (1–2) (1993) 25–28.

[16] G. Meletiou, G.L. Mullen, A note on discrete logarithms in finite fields, Appl. Algebra Eng. Commun. Comput. 3 (1) (1992) 75–79.

[17] G. Meletiou, D.K. Tasoulis, M.N. Vrahatis, A first study of the neural network approach to the RSA cryptosystem, IASTED 2002 Conference on Artificial Intelligence, 2002, pp. 483–488.

[18] G.C. Meletiou, D.K. Tasoulis, M.N. Vrahatis, Cryptography through interpolation approximation and computational intelligence methods, Bull. Greek Math. Soc. 48 (2003) 61–75.

[19] A.J. Menezes, C.P. Van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, 1996.

[20] R.C. Merkle, M.E. Hellman, Hiding information and signatures in trapdoor knapsacks, IEEE Trans. Inf. Theory 24 (1978) 525–530.

[21] M.F. Møller, A scaled conjugate gradient algorithm for fast supervised learning, Neural Networks 6 (1993) 525–533.

[22] G.L. Mullen, D. White, A polynomial representation for logarithms in $GF(q)$, Acta Arithmetica 47 (1986) 255–261.

[23] H. Niederreiter, A short proof for explicit formulas for discrete logarithms in finite fields, Appl. Algebra Eng. Commun. Comput. 1 (1990) 55–57.

[24] A.M. Odlyzko, Discrete logarithms in finite fields and their cryptographic significance, EUROCRYPT 84 (1984).

[25] A.M. Odlyzko, Discrete logarithms: the past and the future, Des. Codes Cryptogr. 19 (2000) 129–145.

[26] A. Pincus, Approximation theory of the MLP model in neural networks, Acta Numerica (1999) 143–195.

[27] S.C. Pohlig, M. Hellman, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, IEEE Trans. Inf. Theory 24 (1978) 106–110.

[28] M. Riedmiller, H. Braun, A direct adaptive method for faster backpropagation learning: the RPROP algorithm, in: Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, 1993, pp. 586–591.

[29] R. Rivest, A. Shamir, L. Adlemann, A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM 21 (1978) 120–126.

[30] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland (Eds.), Parallel Distributed Processing, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[31] D.F. Specht, Probabilistic neural networks, Neural Networks 3 (1) (1990) 109–118.

[32] M.N. Vrahatis, G.S. Androulakis, J.N. Lambrinos, G.D. Magoulas, A class of gradient unconstrained minimization algorithms with adaptive stepsize, J. Comput. Appl. Math. 114 (2) (2000) 367–386.

[33] A. Winterhof, A note on the interpolation of the Diffie–Hellman mapping, Bull. Aust. Math. Soc. 64 (3) (2001) 475–477.

[34] A. Winterhof, Polynomial interpolation of the discrete logarithm, Des. Codes Cryptogr. 25 (1) (2002) 63–72.

[35] H. White, Connectionist nonparametric regression: multilayer feedforward networks can learn arbitrary mappings, Neural Networks 2 (1989) 359–366.