Nonnegative Least Squares Learning for the Random Neural Network

Stelios Timotheou*

Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College
London SW7 2BT, UK
{stelios.timotheou}@imperial.ac.uk

Abstract. In this paper, a novel supervised batch learning algorithm for the Random Neural Network (RNN) is proposed. The RNN equations associated with training are purposively approximated to obtain a linear Nonnegative Least Squares (NNLS) problem that is strictly convex and can be solved to optimality. Following a review of selected algorithms, a simple and efficient approach is employed after being identified to be able to deal with large scale NNLS problems. The proposed algorithm is applied to a combinatorial optimization problem emerging in disaster management, and is shown to have better performance than the standard gradient descent algorithm for the RNN.

1 Introduction

One of the most prominent features of neural networks is their ability to learn from examples. Supervised learning has been extensively employed in a plethora of neural networks models, including the random neural network (RNN), which is of interest in this paper. RNN is a recurrent neural network inspired by the pulsed behaviour of natural neuronal networks [1, 2]. In contrast to other recurrent neural models, its steady state can be described by a product form probability distribution of the neuron states, whereas its signal flow equations are analytically solvable. Moreover, as is the case for conventional artificial neural networks, RNN is able to approximate continuous and bounded functions [3, 4]. A gradient descent supervised learning algorithm for RNN that sequentially updates the weights of the network for each pattern was introduced in [5]. The algorithm has been successfully applied in many problems including image processing [6], magnetic resonance imaging [7] and wafer surface reconstruction [8]; a survey of applications can be found in [6].

Apart from its usefulness as a neural network, RNN can be used as a modelling tool to capture the behaviour of interactive agents in several contexts. The

* This research was undertaken as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Networks) project and is jointly funded by a BAE Systems and EPSRC strategic partnership (EP/C548051/1).

RNN has a direct analogy to queueing networks and extends them by introducing the notion of "negative" customers for work removal. This notion has inspired a great amount of scientific work resulting in numerous extensions of queueing networks called *G-networks* [9]. Recently, a variation of RNN was employed to model both the interactions taking place in gene regulatory networks [10] and the synchronized firing effects in neuronal networks [11, 12].

Therefore, introducing efficient learning algorithms for the RNN is important not only for applications where the RNN is used as a neural network tool but also in cases where a combination of modelling and learning is required. In this paper, a new batch learning algorithm for the RNN based on an approximation of the network's steady-state equations, is proposed. The obtained system is linear with nonnegativity constraints and can be solved using linear Nonnegative Least Squares (NNLS) techniques.

Linear least squares techniques have been utilized in feedforward connectionist neural networks [13, 14]. The learning process becomes linear by either neglecting the scaling effect or approximating the nonlinear activation function. Least squares solution is only applied to single layers of neurons and not the whole network. On the contrary, the proposed approach is developed for a recurrent network with least squares applied to the whole network.

The remaining of the paper is structured as follows: In section 2, a concise description of the RNN and its gradient descent learning algorithm is provided. In section 3, supervised learning in RNN is formulated as a NNLS problem which in turn is solved using the RNN-NNLS algorithm developed in section 4. Following, in section 5, is the comparison of its performance with that of the standard gradient algorithm when applied to a combinatorial optimization problem arising in disaster management. The conclusions are summarized in section 6.

2 The Random Neural Network

The RNN model is comprised of N neurons that interact with each other sending excitatory and inhibitory signals. Neuron i is excited if its potential $k_i(t) \in \mathbb{N}$ is positive and it is quiescent or idle if $k_i(t) = 0$. The probability of neuron i being excited is equal to q_i. Neuron i receives excitatory or inhibitory spikes from the outside world according to Poisson processes of rates Λ_i and λ_i. Excitatory spikes increase the internal state of the receiving neuron by 1, whereas inhibitory spikes reduce this by 1 if neuron i is excited. If a neuron i is active, it can fire a spike at time t with an independent exponentially distributed firing rate r_i. If this happens then its potential $k_i(t)$ is reduced by 1. The resulting spike reaches neuron j as an excitatory spike with probability $p^+(i,j)$ or as an inhibitory spike with probability $p^-(i,j)$, or it departs from the network with probability $d(i)$. Because the probabilities associated with neuron i must sum up to 1 it is true that: $\sum_{j=1}^{N} [p^+(i,j) + p^-(i,j)] + d(i) = 1$.

The signal-flow equations of the network are described by the following system of nonlinear equations:

$$q_i = \min\{1, \frac{\lambda^+(i)}{r_i + \lambda^-(i)}\}, \quad 0 \leq q_i \leq 1$$
$$\lambda^+(i) = \Lambda_i + \sum_{j=1}^{N} r_j q_j p^+(j,i) \tag{1}$$
$$\lambda^-(i) = \lambda_i + \sum_{j=1}^{N} r_j q_j p^-(j,i)$$

The above system of nonlinear equations (1) always exists and it is unique [1, 2, 5], whereas the steady-state probability distribution of the neuron potentials is in product form: $\pi(\underline{k}) = \prod_{i=1}^{N}(1-q_i)q_i^{k_i}$, if $q_i < 1$.

Similar to connectionist models, RNN has weights, $w^+(i,j)$ and $w^-(i,j)$, that are used for learning. These weights represent the firing rates of excitatory and inhibitory signals from neuron i to neuron j respectively, and are given by $w^+(i,j)$=$r_i p^+(i,j) \geq 0$ and $w^-(i,j) = r_i p^-(i,j) \geq 0$. From the definition of the RNN weights, an expression for the firing rate r_i is derived:

$$r_i = \sum\nolimits_{j=1}^{N} [w^+(i,j) + w^-(i,j)]/(1-d(i)) \tag{2}$$

Supervised learning in RNN can be performed by mapping the positive and negative values of the input patterns to the parameters $\underline{\Lambda}_k = [\Lambda_{1k}, \, ... \, , \Lambda_{Nk}]$, $\underline{\lambda}_k = [\lambda_{1k}, \, ... \, , \lambda_{Nk}]$ respectively, and the output patterns to the parameters $\underline{q}_k = [q_{1k}, \, ... \, , q_{Nk}]$. In order to update the RNN weights $w^{\pm}(u,v)$ according to the gradient descent rule, is is required to find expressions for the derivatives $\partial q_i / \partial w^{\pm}(u,v)$. Although this is challenging task because the system of equations (1) is nonlinear, as proven in [5], these terms can be expressed as a linear system of equations with unknowns $\partial \underline{q} / \partial w^{\pm}(u,v)$ that can be efficiently solved.

The general steps of the RNN gradient descent supervised learning algorithm are as follows [5]:

1. Initialize the weights of the network
2. For each input-output pattern do the following:
   - Initialize appropriately the parameters $\underline{\Lambda}_k$, $\underline{\lambda}_k$ and $\underline{q}_k$
   - Solve the system of nonlinear equations (1)
   - Use the results obtained to calculate the terms $\partial \underline{q} / \partial w^{\pm}(u,v)$
   - Update the weights according to the gradient descent rule
3. Repeat the procedure in step 2 until convergence

3 NNLS Formulation of RNN Supervised Learning

The purpose of supervised learning is to find an optimal set of weights so that for a set K of input patterns, the network's output closely represents the desired one. As already mentioned, the output of the k-th pattern is associated with the parameters $\underline{q}_k$, which are derived from the system of equations (1). Assuming that $0 \leq \lambda^+(i)/(r_i + \lambda^-(i)) \leq 1, \forall i, k$ and that $d(i) = 0,\ \forall i$ then combining (1) with (2) for all patterns yields:

$$\begin{aligned}
& q_{ik} \sum_{j=1}^{N} w^-(i,j) + q_{ik} \sum_{j=1}^{N} q_{jk} w^-(j,i) + \\
& + q_{ik} \sum_{j=1}^{N} w^+(i,j) - \sum_{j=1}^{N} q_{jk} w^+(j,i) = \Lambda_{ik} - q_{ik}\lambda_{ik},\ \forall i,\ k \qquad (3) \\
& \text{and } w^+(i,j)\ \geq\ 0,\ w^-(i,j)\ \geq\ 0\ \forall\ i,\ j
\end{aligned}$$

The above system is comprised of NK equations. If all q_{ik} are known then the only unknowns are the $2N^2$ weights $w^+(i,j)$ and $w^-(i,j)$ and the resulting system is linear with nonnegativity constraints. The assumption that all q_{ik} are known is valid when there is a large number of output neurons which are set to their desired output values; for the other neurons, q_{ik} can be set to random values.

Because of the nonnegativity constraints and the fact that $K \neq 2N$ a unique solution to the system (4) may not exist. In fact, the system can be either over-determined or under-determined depending on the value of K. The best that can be done is to minimize the least squares error, which can be accomplished by formulating the system of equations (4) as a linear NNLS problem:

$$\min_{\underline{w}\geq 0} f(\underline{w})\ =\ \min_{\underline{w}\geq 0} 0.5\|\mathbb{A}\underline{w} - \underline{b}\|_2^2, \quad \mathbb{A} \in \mathbb{R}^{NK\times 2N^2},\ \underline{b} \in \mathbb{R}^{NK\times 1},\ \underline{w} \in \mathbb{R}^{2N^2\times 1} \qquad (4)$$

where $\|\ \cdot\ \|_2$ is the L2-norm. The gradient of $f(\underline{w})$ is $\nabla f(\underline{w})\ =\ \mathbb{A}^T(\mathbb{A}\underline{w}) - \mathbb{A}^T\underline{b}$. If $\mathbb{A}$ is a full rank matrix, NNLS is a strictly convex quadratic optimization problem.

Let row ik of matrix $\mathbb{A}$ be the one associated with equation ik of system (4) and ij^+ and ij^- columns be the ones corresponding to the variables $w^+(i,j)$ and $w^-(i,j)$ of $\underline{w}$. Then, $\mathbb{A}(ik, ij^+) = \mathbb{A}(ik, ij^-) = q_{ik}$, $\mathbb{A}(ik, ji^+) = -q_{jk}$ and $\mathbb{A}(ik, ji^-) = q_{ik}q_{jk}\ \forall j \neq i$, while for $j = i$ $\mathbb{A}(ik, ii^-) = q_{ik} + q_{ik}^2$. The other elements of row ik of $\mathbb{A}$ are equal to zero. In addition, the elements of $\underline{b}$ are given by:

$$b(ik) = \Lambda_{ik} - q_{ik}\lambda_{ik}, \quad \forall\ i, k \qquad (5)$$

Notice that $\mathbb{A}$ is of high dimensionality ($NK \times 2N^2$ elements). However, it is highly sparse since each of its $2N^2$ element rows, contain only $4N - 3$ nonzero elements. Moreover, the nonzero values of $\mathbb{A}$ can be easily calculated using the q_{ik} values. Consequently, vector $\underline{q} \in \mathbb{R}^{NK\times 1}$ can be stored instead of $\mathbb{A}$ and used to calculate simple operations involving $\mathbb{A}$ such as matrix-vector products. Consequently, the developed approach should not require the storage of $\mathbb{A}$ and involve simple operations with $\mathbb{A}$.

4 Nonnegative Least Squares Learning

The algorithms for the solution of NNLS problems can be divided into two broad classes: *active set algorithms* and *iterative approaches.*

Active set algorithms [15, 16] rely on the fact that any variables that take negative or zero values when the unconstrained least squares problem is solved, do not contribute to the solution of the constrained problem. These constraints form a set called *active*, while the set of constraints corresponding to positive variables is called *passive.* All constraints are initially inserted into the active set. Afterwards, an iterative procedure is followed to identify and remove variables from the particular set. This involves the solution of the unconstrained problem formed by the variables in the passive set. Consequently, active set methods require matrix inversion operations and are not appropriate.

Iterative methods are based on standard techniques used in nonlinear optimization to update $\underline{w}$ such as gradient descent methods and interior point methods. A popular approach for NNLS problems is to update the current solution with the use of *projected gradient methods* (6). These methods can identify several active set constraints at a single iteration and do not require sophisticated matrix operations. In the projected gradient methods, the update takes place towards the steepest descent direction. However, by using the projection operation (7), it is ensured that the new point is within the feasible region.

$$\underline{w}^{\tau+1} = P[\underline{w}^{\tau} - s^{\tau}\mathbb{D}^{\tau}\nabla f(\underline{w})], \quad s^{\tau} \geq 0,\ \mathbb{D}^{\tau} \in \mathbb{R}^{2N^2 \times 2N^2} \tag{6}$$

$$P[w_i] = \begin{cases} w_i, & w_i \ > \ 0 \\ 0, & w_i \ \leq \ 0 \end{cases} \tag{7}$$

Many different projected gradient methods have been developed which employ different techniques for selecting the step size s^{τ} and the gradient scaling matrix $\mathbb{D}^{\tau}$.

Lin proposed an algorithm based on first order projected gradient information that makes no use of $\mathbb{D}^{\tau}$ [17]. Lin introduced an efficient modification of the "Armijo rule along the projection arc" (APA) [18] to update the value for the step size s^{τ}. In APA, $s^{\tau+1}$ is found by starting from a large value for s^{τ} and exponentially decreasing it until condition (8) is satisfied.

$$f(\underline{w}^{\tau+1}) - f(\underline{w}^{\tau}) \leq \sigma\nabla f(\underline{w})^{T}(\underline{w}^{\tau+1} - \underline{w}^{\tau}) \tag{8}$$

In Lin's approach, $s^{\tau+1}$ is obtained by starting the search for the new step-size from the previous optimal value and appropriately increasing or decreasing it until condition (8) is satisfied (Algorithm 1).

This algorithm is not only simple and efficient but also it is appropriate for the purposes of this paper since it requires only matrix-vector multiplications involving $\mathbb{A}$ and there is no need to store the latter. Specifically, the calculations involving $\mathbb{A}$ are $\mathbb{A}\underline{w}$ to find $f(\underline{w})$ as well as $\underline{z} = \mathbb{A}\underline{w}$, $\mathbb{A}^{T}\underline{z}$ and $\mathbb{A}^{T}\underline{b}$ to get $\nabla f(\underline{w})$.

A weakness of the above approach is that it relies on first order gradient information and has slow convergence.

Algorithm 1 Projected Gradient Algorithm for the NNLS problem

```
Set $\sigma = 0.01,\ \beta = 0.1$
Set $s^0 = 1$ and $\underline{w}^1 = 0$
repeat
  $s^\tau \leftarrow s^{\tau-1}$
  if {(8) is satisfied} then
    repeat
      $s^\tau \leftarrow s^\tau/\beta$
      Set $\underline{w}^{\tau+1} \leftarrow P[\underline{w}^\tau - s^\tau \nabla f(\underline{w}^\tau)]$
    until {((8) is not satisfied) or ($\underline{w}(s^\tau/\beta) = \underline{w}(s^\tau)$)}
  else
    repeat
      $s^\tau \leftarrow s^\tau \beta$
      Set $\underline{w}^{\tau+1} \leftarrow P[\underline{w}^\tau - s^\tau \nabla f(\underline{w}^\tau)]$
    until {(8) is satisfied}
  end if
  Set $\underline{w}^{\tau+1} \leftarrow P[\underline{w}^\tau - s^\tau \nabla f(\underline{w}^\tau)]$
until {Convergence}
```

A quasi-Newton method that uses the gradient scaling matrix $\mathbb{D}^\tau$ achieving fast convergence was proposed in [19]. This algorithm not only exploits the active set constraints but it also utilizes an approximation of the Hessian matrix to update $\mathbb{D}^\tau$. Nevertheless, as the algorithm progresses, $\mathbb{D}^\tau$ becomes dense and requires storage making the algorithm prohibitive for the problem examined in this paper.

Taking everything into consideration, the projected gradient method proposed in [17] has been employed. The slow convergence of the algorithm is not a major issue because obtaining a solution close to optimum is sufficient if the fact that the problem itself is an approximation is regarded.

The proposed RNN-NNLS learning algorithm for obtaining the weights of RNN according to the input-output pairs is outlined in Algorithm 2. Two significant issues related to the algorithm are highlighted. Firstly, the NNLS algorithm is executed without matrix $\mathbb{A}$ as input; $\underline{q}$ is sufficient to perform all the matrix-vector product operations associated with $\mathbb{A}$. Hence, the order of memory required is the same as for the standard RNN learning algorithm. Secondly, when solving the nonlinear system of equations (1) in the main for-loop of the algorithm, only the non-output neurons are updated; the values of the output neurons are kept constant to their desired values. The procedure is repeated for a number of iterations and the best acquired solution is exploited.

5 Experimental Results

For the evaluation of the proposed NNLS learning algorithm, a problem emerging in disaster management is considered. The problem deals with the allocation of emergency units to locations of injured civilians which is a hard combinatorial

Algorithm 2 Supervised RNN-NNLS Batch Learning Algorithm

```
Based on the input patterns initialize $\Lambda_{ik}$ and $\lambda_{ik}$ $\forall$ $i$, $k$
Set $0 < q_{i_{out}k} < 1$ $\forall$, $k$, $i_{out}$, according to the desired output of neuron $i$ of the $k$-th pattern. Index $i_{out}$ denotes an output neuron
Initialize the remaining neurons $0 < q_{\overline{i_{out}}k} < 1$ $\forall$ $i$, $k$ randomly and based on $q_{i_{out}k}$, $q_{\overline{i_{out}}k}$ form $\underline{q} \in \mathbb{R}^{NK\times 1}$
for a number of iterations do
  Update $\underline{b}$ according to (5)
  $\underline{w} \leftarrow ProjectedGradientNNLS(\underline{q}, \underline{b})$
  for all k do
    Update $q_{\overline{i_{out}}k}$ by solving the nonlinear system of equations (1)
  end for
end for
```

optimization problem. Problems of this nature require fast and close to optimal decisions. A heuristic method that could be employed to solve such problems is to train a neural network off-line with instances of the optimization problem in the same physical context as the disaster and then exploit the trained neural network to solve any problem instances that arise during the emergency. The developed algorithm is applied to such a problem for training and decision and its performance is compared to that obtained when training is performed with the standard gradient descent RNN learning algorithm [5].

In the emergency problem considered, a number of civilians I_j is injured at incident j with a total of N_L such simultaneous and spatially distributed incidents. N_U emergency units are positioned at different locations and need to collect the injured. Unit i can collect $c_i > 0$ injured and has response time to incident j, $T_{ij} > 0$. Under the assumption that one unit can be allocated to only one incident and that the total capacity of the emergency units, $c_t = \sum_i (c_i)$, is sufficient to collect all the injured, the goal is to find an allocation matrix $\underline{x}$ with elements x_{ij} which minimizes total response time $f(\underline{x})$:

$$\min \; f(\underline{x}) = \sum_{i=1}^{N_U}\sum_{j=1}^{N_L} T_{ij}x_{ij} \tag{9}$$

subject to the constraints:

$$\sum_{j=1}^{N_L} x_{ij} = 1 \;\forall\; i, \quad \sum_{i=1}^{N_U} c_i x_{ij} \geq I_j \;\forall\; j, \quad x_{ij} \in \{0,1\} \forall\; i,\; j \tag{10}$$

The binary variables x_{ij} represent whether emergency unit i is allocated to incident j. The first constraint designates that an emergency unit must be allocated to exactly one incident, whilst the second indicates that the total capacity of the units allocated to an incident must be at least equal to the number of civilians injured there. The above problem is NP-hard.

Assuming that the locations and capacities of the emergency units as well as the locations of the incidents are fixed the problem can be mapped to a supervised learning context, where the inputs to the network are the parameters I_j and the outputs correspond to the binary variables x_{ij}. To represent the output variables two neurons are used, a "positive" and a "negative". If $x_{ij} = 1$, then the excitation level of the corresponding "positive" output neuron is high (close to 1) and the excitation level of the "negative" output neuron is low (close to 0). If $x_{ij} = 0$, then the reverse is true. Thus, the corresponding neural network architecture is comprised of a fully recurrent neural network with N_L input neurons and $2N_U N_L$ output neurons.

Training was performed for values of N_U ranging from 8 to 20 and $N_L = \{3, 5\}$ using 250 optimally solved training optimization instances. For every set of training instances, the parameters $T_{ij} \in (0, 1)$ and $c_i \in \{1, 2, 3, 4\}$ were fixed to uniformly distributed random values, while for each of the training pattern $I_j \in \mathbb{N}$ were also randomly generated from the uniform distribution in the range $[0.5 \cdot c_t/N_L, 1.1 \cdot c_t/N_L]$.

Testing was performed using a distinct generated set of 200 test cases, with the same probability distributions for all parameters, so that the training and testing sets were disjoint. To emphasize the fact that $\mathbb{A}$ cannot be stored notice that for $N_U = 20$ and $N_L = 5$ a network of $N = 205$ neurons is produced and $\mathbb{A}$ is a 51250×84050 matrix.

The results produced by the two methods were evaluated based on:

1. The ratio $f_{NN}(\underline{x})/f_{opt}(\underline{x})$ that shows how close to optimality the solutions are (Fig.1)
2. The percentage of instances solved so that all of the injured were evacuated (feasible solutions)(Fig. 2)
3. The percentage of injured civilians collected (Fig. 3)

All metrics were averaged over the testing instances.

As illustrated in Fig. 1 both algorithms approach the optimal solutions equally well. Concerning the percentage of injured civilians collected and the percentage of instances where all civilians are collected, Figures 2 and 3 imply that for $N_U = 8$ the gradient descent algorithm performs better than the RNN-NNLS one. This may be the case because for small networks there are less local minima and gradient descent can reach a good solution. However, for the rest of the cases, the RNN-NNLS learning algorithm performs substantially better that the gradient descent one. This shows that the developed algorithm can work quite well for large-scale networks and that the approximation is more accurate as the network grows.

6 Conclusions

A novel algorithm for learning in the RNN, the RNN-NNLS, has been presented. The algorithm is based on approximating the system of equations describing RNN by a linear system with nonnegative variables. For the solution of the formulated NNLS problem an efficient and simple projected gradient algorithm, shown to be able to solve the problem at hand without the storage of

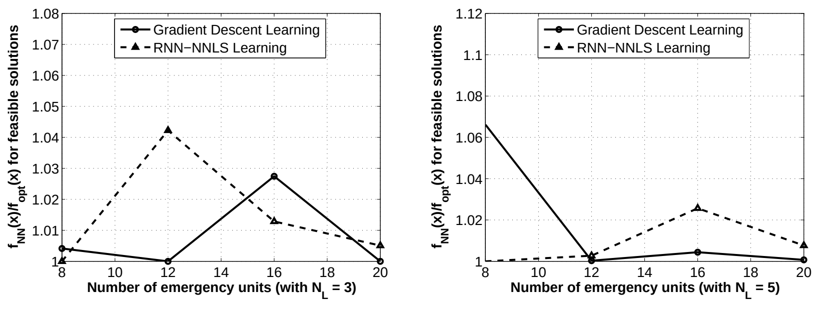

Fig. 1. Average of $f_{NN}(\underline{x})/f_{opt}(\underline{x})$ for the solutions where the units are able to remove all the injured civilians (i.e. the "feasible" ones)

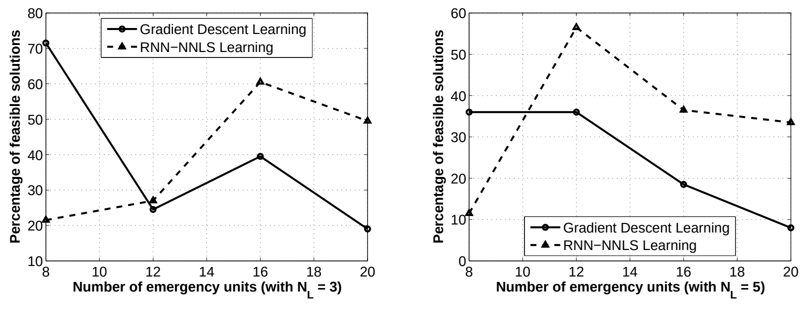

Fig. 2. Percentage of solutions in which all injured civilians are evacuated; these solutions are called "feasible" in the graphs, for want of a better term

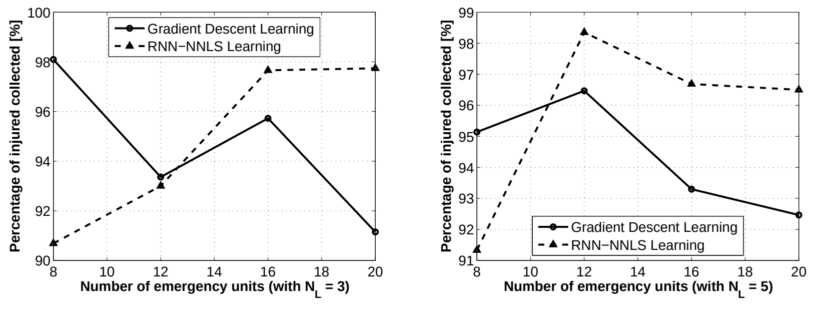

Fig. 3. Percentage of injured civilians that are collected

any large matrices, is employed. RNN-NNLS algorithm appears to have better performance than the gradient descent RNN algorithm when applied to a hard combinatorial optimization problem associated with the allocation of emergency units to locations of injured civilians.

References

1. Gelenbe, E.: Random neural networks with positive and negative signals and product form solution. Neural Computation **1**(4) (1989) 502–510
2. Gelenbe, E.: Stability of the random neural network. Neural Computation **2**(2) (1990) 239–247
3. Gelenbe, E., Mao, Z.H., Li, Y.D.: Function approximation with spiked random networks. IEEE Trans. on Neural Networks **10**(1) (1999) 3–9
4. Gelenbe, E., Mao, Z.H., Li, Y.D.: Function approximation by random neural networks with a bounded number of layers. J. Differential Equations and Dynamical Systems **12**(1 & 2) (2004) 143–170
5. Gelenbe, E.: Learning in the recurrent random network. Neural Computation **5** (1993) 154–164
6. Bakircioglu, H., Kocak, T.: Survey of random neural network applications. European Journal of Operational Research **126**(2) (2000) 319–330
7. Gelenbe, E., Feng, T., Krishnan, K.R.R.: Neural network methods for volumetric magnetic resonance imaging of the human brain. Proceedings of the IEEE **84**(1) (1996) 1488–1496
8. Gelenbe, E., Kocak, T.: Wafer surface reconstruction from top-down scanning electron microscope images. Microlectronic Engineering **75** (2004) 216–233
9. Artalejo, J.R.: G-networks: A versatile approach for work removal in queueing networks. European Journal of Operational Research **126**(2) (2000) 233–249
10. Gelenbe, E.: Steady-state solution of probabilistic gene regulatory networks. Physical Review E **76**(1) (2007) 031903
11. Gelenbe, E., Timotheou, S.: Random Neural Networks with Synchronised Interactions. accepted for publication in *Neural Computation* , MIT Press (2008)
12. Gelenbe, E., Timotheou, S.: Synchronised Interactions in Spiked Neuronal Networks. accepted for publication in *The Computer Journal*, Oxford Journals (2008)
13. Biegler-Konig, F., Barmann, F.: A learning algorithm for multilayered neural networks based on linear least squares problems. Neural Networks **6**(1) (1993) 127–131
14. Fontenla-Romero, O., Erdogmus, D., Principe, J.C., Alonso-Betanzos, A., Castillo, E.: Linear least-squares based methods for neural networks learning. In: Artificial Neural Networks and Neural Information Processing ICANN/ICONIP 2003. (2003) 84–91
15. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. PrenticeHall (1987)
16. Bro, R., Long, S.D.: A fast non-negativity-constrained least squares algorithm. Journal of Chemometrics **11**(5) (1997) 393–401
17. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural Computation **19**(10) (2007) 2756–2779
18. Bertsekas, D.: Nonlinear Programming. Athena Scientific (1995)
19. Kim, D., Sra, S., Dhillon, I.: A new projected quasi-newton approach for the nonnegative least squares problem. Technical report, Dept. of Computer Sciences, The University of Texas at Austin (Dec. 2006)