Neural Network Optimization

Mina Niknafs

Abstract

In this report we want to investigate different methods of Artificial Neural Network optimization. Different local and global methods can be used. Backpropagation is the most common method for optimization. Other methods like genetic algorithm, Tabu search, and simulated annealing can be also used. In this paper we implement GA and BP for ANN.

1. Introduction

In machine learning, artificial neural networks (ANNs) are a family of models inspired by biological neural networks and are used to approximate functions that can depend on a large number of inputs and are generally unknown. ANNs are presented as systems of interconnected "neurons" which exchange messages between each other. The connections have weights that can be tuned, making neural nets adaptive to inputs and capable of learning [1]. The ability of the ANNs to accurately approximate unknown functions [2,3], underlies this growing popularity.

Neural networks have been used to solve a wide variety of tasks that are hard to solve using ordinary rule-based programming, including computer vision and speech recognition, Character Recognition, Image Compression, Stock Market Prediction, Traveling Saleman's Problem, and Medicine, Electronic Nose, Security, and Loan Applications [1].

The word network in the term 'artificial neural network' refers to the inter–connections between the neurons in the different layers of each system. An example system has three layers (see figure 1). The first layer has input neurons which send data via synapses to the second layer of neurons (hidden layer), and then via more synapses to the third layer of output neurons. More complex systems will have more layers of neurons, some having increased layers of input neurons and output neurons. The synapses store parameters called "weights" that manipulate the data in the calculations.

An ANN is typically defined by three types of parameters:

1. Architecture (The interconnection pattern between the different layers of neurons)
2. The learning process for updating the weights of the interconnections
3. The activation function that converts a neuron's weighted input to its output activation.

Mathematically, a neuron's network function $f(x)$ is defined as a composition of other functions $g_i(x)$, which can further be defined as a composition of other functions. This can be conveniently represented as a network structure, with arrows depicting the dependencies between variables. A widely used type of composition is the nonlinear weighted sum,

where $f(x) = K\left(\sum_i w_i g_i(x)\right)$, where K (commonly referred to as the activation function) is some predefined function, such as the hyperbolic tangent. It will be convenient for the following to refer to a collection of functions g_i as simply a vector $g = (g_1, g_2, \ldots, g_n)$.

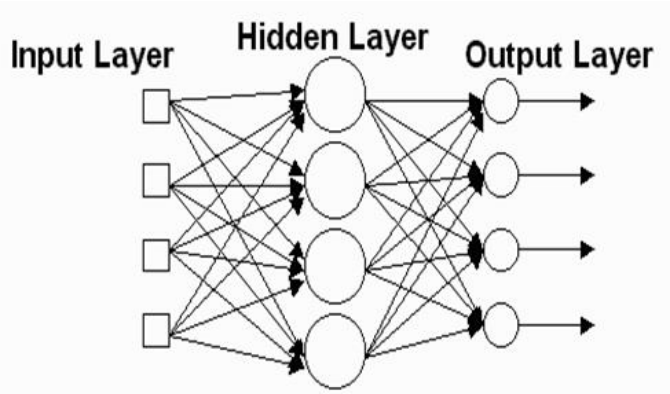


Figure 1: ANN with one hidden layer

What has attracted the most interest in neural networks is the possibility of learning. Given a specific task to solve, and a class of functions F, learning means using a set of observations to find $f^* \in F$ which solves the task in some optimal sense. This entails defining a cost function (objective function) $C : F \to \mathbb{R}$ such that, for the optimal solution f^* , $C(f^*) \leq C(f) \forall f \in F$ – i.e., no solution has a cost less than the cost of the optimal solution [1].

Most ANNs contain some form of 'learning rule' which modifies the weights of the connections according to the input patterns that it is presented with. The cost function C is an important concept in learning, as it is a measure of how far away a particular solution is from an optimal solution to the problem to be solved. Learning algorithms search through the solution space to find a function that has the smallest possible cost. As a simple example, consider the problem of finding the model f, which minimizes $C = E\left[(f(x) - y)^2\right]$, for data pairs (x, y) drawn from some distribution $\mathcal{D}$. In practical situations we would only have N samples from $\mathcal{D}$ and thus, for the above example, we would only minimize $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - y_i)^2$. Thus, the cost is minimized over a sample of the data rather than the entire distribution generating the data.

2. Artificial Neural Network Optimization

Generally when we talk about ANN optimization, the objective function is mean square error function (loss/cost function). We have to find optimize values weights of neural network to minimize the objective function. Although, gradient based search techniques such as back-propagation are currently the most widely used optimization techniques for training neural networks, it has been shown that these gradient techniques are severely limited in their ability to

find global solutions. Global search techniques have been identified as a potential solution to this problem. Other techniques can be used. Two well-known global search techniques, Simulated Annealing and the Genetic Algorithm can be used as well [4].

Because of its ease of use, an overwhelming majority of these applications have used some variation of the gradient technique, backpropagation (BP) [5,6] for optimizing the networks. Although, backpropagation has unquestionably been a major factor for the success of past neural network applications, it is plagued with inconsistent and unpredictable performances.
Recently, [7] demonstrated that for a variety of complex functions the genetic algorithm was able to achieve superior solutions for neural network optimization than backpropagation and [8] found that another global search heuristic, Tabu Search (TS), is able to systematically achieve superior solutions for optimizing the neural network than those achieved by backpropagation. In addition to GA and TS another well-known global search heuristic is simulated annealing. Simulated annealing has been shown to perform well for optimizing a wide variety of complex problems. Authors in [4] compared BP, SA, and GA for ANN. They concluded that although the most frequently used algorithm for optimizing neural networks is backpropagation, it is likely to obtain local solutions. They showed that simulated annealing, a global search algorithm, performs better than backpropagation, but is also uses a point to point search. Both BP and SA have multiple user determined parameters which may significantly impact the solution. Since there are no established rules for selecting these parameters, solution outcome is based on chance. The genetic algorithm appears to be able to systematically obtain superior solutions to simulated annealing for optimizing neural networks. These solutions provide the researcher with superior estimates of interpolation data. The genetic algorithm's process of moving from one population of points to another enables it to discard potential local solutions and also to achieve the superior solutions in a computationally more efficient manner.

2.1. Backpropagation

Backpropagation, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks used in conjunction with an optimization method such as gradient descent. The method calculates the gradient of a loss function with respect to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.

Backpropagation requires a known, desired output for each input value in order to calculate the loss function gradient. It is therefore usually considered to be a supervised learning method, although it is also used in some unsupervised networks such asautoencoders. It is a generalization of the delta rule to multi-layered feedforward networks, made possible by using the chain rule to iteratively compute gradients for each layer. Backpropagation requires that the activation function used by the artificial neurons (or "nodes") be differentiable.

Gradient search techniques such as backpropagation are designed for local search. They typically achieve the best solution in the region of their starting point. Obtaining a global solution is often

dependent on a fortuitous choice of starting values. Global search techniques are known to more consistently obtain the optimal solution.

2.2. Genetic Algorithm

For difficult non-linear functions GA has been shown to perform exceedingly well in obtaining global solutions [9,10]. Basically, an objective function, such as minimization of the sum of squared errors or sum of absolute errors, is chosen for optimizing the neural network. Using the chosen objective function, each candidate point out of the initial population of randomly chosen starting points is used to evaluate the objective function. These values are then used in assigning probabilities for each of the points in the population. For minimization, as in the case of sum of squared errors, the highest probability is assigned to the point with the lowest objective function value. Once all points have been assigned a probability, a new population of points is drawn from the original population with replacement. The points are chosen randomly with the probability of selection equal to its assigned probability value. Thus, those points generating the lowest sum of squared errors are the most likely to be represented in the new population. The points comprising this new population are then randomly paired for the crossover operation. Each point is a vector (string) of n parameters (weights). A position along the vectors is randomly selected for each pair of points and the preceding parameters are switched between the two points. This crossover operation results in each new point having parameters from both parent points. Finally, each weight has a small probability of being replaced with a value randomly chosen from the parameter space. This operation is referred to as mutation. Mutation enhances the GA by intermittently injecting a random point in order to better search the entire parameter space. This allows the GA to possibly escape from local optima if the new point generated is a better solution than has previously been found, thus providing a more robust solution. This resulting set of points now becomes the new population, and the process repeats until convergence.
Since this method simultaneously searches in many directions, the probability of finding a global optimum greatly increases. The algorithm's similarity to natural selection inspires its name. As the GA progresses through generations, the parameters most favorable for optimizing the objective function will reproduce and thrive in future generations, while poorly performing parameters die out, as in "survival of the fittest".

2.3. Simulated Annealing

Annealing, refers to the process which occurs when physical substances, such as metals, are raised to a high energy level (melted) and then gradually cooled until some solid state is reached. The goal of this process is to reach the lowest energy state. In this process physical substances usually move from higher energy states to lower ones if the cooling process is sufficiently slow, so minimization naturally occurs. Due to natural variability, however, there is some probability at each stage of the cooling process that a transition to a higher energy state will occur. As the energy state naturally declines, the probability of moving to a higher energy state decreases. A detailed description of the simulated annealing algorithm and its use for optimization can be found in [11]. In essence, simulated annealing draws an initial random point to start its search. From this point, the algorithm takes a step within a range predetermined by the user.

This new point's objective function value is then compared to the initial point's value in order to determine if the new value is smaller. For the case of minimization, if the objective function value decreases it is automatically accepted and it becomes the point from which the search will continue. The algorithm will then proceed with another step. Higher values for the objective function may also be accepted with a probability determined by the Metropolis criteria. By occasionally accepting points with higher values of the objective function, the SA algorithm is able to escape local optima. As the algorithm progresses, the length of the steps declines, closing in on the final solution. The Metropolis criteria uses the initial user defined parameters, T (temperature) and RT (temperature reduction factor) to determine the probability of accepting a value of the objective function that is higher. Paralleling the real annealing process, as T decreases the probability of accepting higher values decreases. T is reduced by the function $T_{i+1} = RT*T_i$, where I is the i_{th} iteration of the function evaluation after every NT iterations. NT is the preset parameter which establishes the number of iterations between temperature reductions. For this study these three parameters were set to values suggested in the literature [11]. Unlike the version of the GA used in this study in which all parameters are dynamically assigned by the algorithm, the performance of the SA algorithm depends on user defined parameters. Since the performance of the SA is significantly impacted by the choice of these parameters a range of values were used for three variables (T, RT, NT). We examine twelve different combinations in order to allow SA a full opportunity to obtain a global solution.

2.4. Tabu Search

Tabu search, proposed by Glover [12], is a meta-heuristic method that has received widespread attention recently because of its flexible framework and several significant successes in solving NP-hard problems [13]. The method of neighborhood exploration and the use of short-term and long-term memories distinguish Tabu Search from local search and other heuristic search methods, and result in lower computational cost and better space exploration. TS involves a lot of techniques and strategies, such as short-term memories (tabu list), long-term memories and other prior information about the solution can used to improve the intensification and diversification of the search. It can be confirmed that the strategy of intensification and diversification is very important at most time, therefore, a novel adaptive search strategy of intensification and diversification, proposed in literature [14] by us, was employed to improve the efficiency of TS for neural network optimization. [15] proposed an adaptive TS approach for ANNs.

3. Implementation

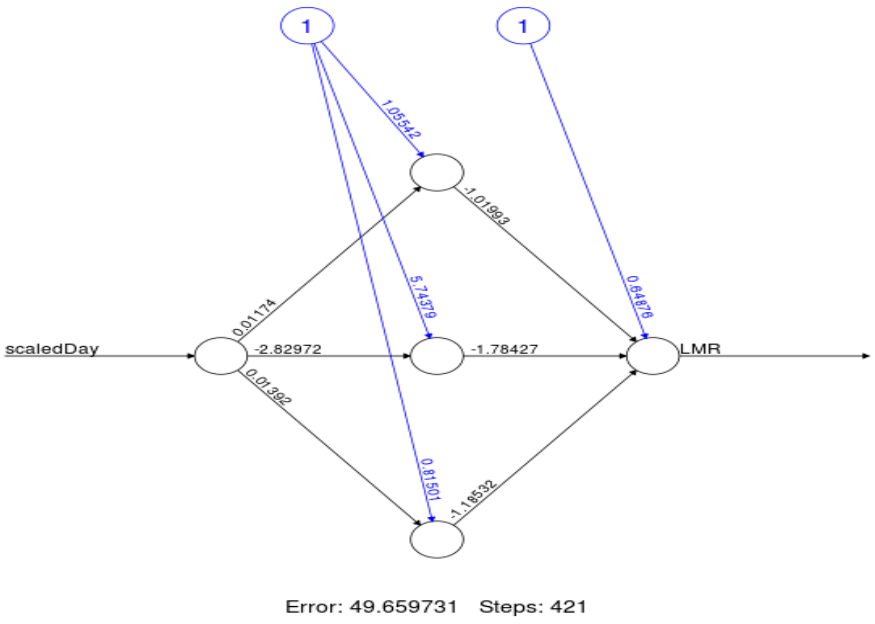
In this section I use two R libraries to compare BP and GA for neural network optimization. I used "nueralnet" package for BP and "ANN" package for GA. The implemented code is as follows:

```
library("neuralnet")
data = read.csv2("mortality.csv")
data$scaledDay = scale(data$Day)
set.seed("7235")
model1 <- neuralnet(LMR ~ scaledDay, data = data, hidden = 3, act.fct ="tanh", stepmax = 1e6,
threshold = 0.1)
```

```
plot(model1)


library("ANN")
input = as.matrix(data$scaledDay)
output = as.matrix(data$LMR)
model2 =  ANNGA(x    = input,
          y          =output,
          design  =c(1, 3, 1),
          population  =100,
          mutation = 0.3,
          crossover = 0.7,
          maxGen  =1000,
          error   =0.001)
```

I considered a very simple architecture for ANN, One input and output node and a hidden layer with 3 nodes. You can see the result of BP in the following figure.



This is the output of GA:

```
Call:
ANNGA.default(x = input, y = output, design = c(1, 3, 1), population = 100,
          mutation = 0.3, crossover = 0.7, maxGen = 1000, error = 0.001)

**************************************************************************
Mean Squared Error----------------------------> 7.824982686
R2--------------------------------------------> -8.32637258
Number of generation--------------------------> 1001
```

Weight range at initialization----------> [25 , -25]
Weight range resulted from the optimisation-----> [24.58616307 , -22.85414239]
****

Since tow models has many parameters to set and they depends to the random start points, here I can't make any conclusion by comparing results.

References

[1] https://en.wikipedia.org/wiki/Artificial_neural_network
[2] Funahashi, K., "On the Approximate Realization of Continuous Mappings by Neural Networks," Neural Networks, 2(3), 183-192, 1989.
[3] Hornik, K., Stinchcombe, M., & White, H., "Multilayer Feed-Forward Networks are Universal Approximators," Neural Networks, 2(5), 359-66, 1989.
[4] R. Ghosh, M. Ghosh, J. Yearwood, and A. Bagirov, "Comparative analysis of genetic algorithm, simulated annealing and cutting angle method for artificial neural networks," 2005, pp. 62–70
[5] LeCun, Y., "Learning Processes in an Asymmetric Threshold Network," Disordered Systems and Biological Organization, 233-40. Berlin: Springer Verlag, 1986.
[6] Rumelhart, D., Hinton, G., & Williams, R., "Learning Internal Representations by Error Propagation," In D. Rumelhart & J. McCleeland (Eds.), Parallel distributed processing: Explorations in the microstructure of cognition (Vol. 1). Cambridge, MA: MIT Press, 1986.
[7] Sexton, R., Dorsey, R., & Johnson, J, "Toward a Global Optimum for Neural Networks: A Comparison of the Genetic Algorithm and Backpropagation," forthcoming in Decision Support Systems
[8] Sexton, R., Allidae, B., Dorsey, R., and Johnson, J., "Global Optimization for Artificial Neural Networks: A Tabu Search Application," forthcoming in European Journal of Operational Research
[9] Dorsey, R. E., & Mayer W. J., "Genetic Algorithms for Estimation Problems with Multiple Optima, Non-Differentiability, and other Irregular Features," Journal of Business and Economic Statistics, 1995, 13(1), 53-66.
[10] Dorsey, R. E., Johnson, J. D., & Mayer, W. J., "A Genetic Algorithm for the Training of Feedforward Neural Networks," Advances in Artificial Intelligence in Economics, Finance and Management (J. D. Johnson and A. B. Whinston, eds., 93-111). Vol. 1. Greenwich, CT: JAI Press Inc., 1994.
[11] Corana, A., Marchesi, M., Martini, C., & Ridella, S., "Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing' Algorithm," ACM Transactions on Mathematical Software, 1987, 13, 262-80.
[12]Glover, F. and Laguna, M.: "Tabu Search", Boston, Kluwer Academic Publishers (1997)
[13] Sexton, R.S., Alidaee, B., Dorsey, R.E. et al.: "Global Optimization for Artificial Neural Networks", A Tabu Search Application. European Journal Operational Research, 106 (1998) 570-584
[14]He, Y., Liu, G., and Qiu, Y.: A Novel Adaptive Search Strategy of Intensification and Diversi-
fication in Tabu Search. Journal of Computer Research & Development, 41 (2004) 162 ~166
[15] He, Y., Qiu, Y., Liu, G., et al. (2005). "Optimizing weights of neural network using an adaptive tabu search approach", In The second international symposium on neural networks 2005 (ISNN-2005) (pp. 672–676), Chongqing, China.