A new back-propagation algorithm with momentum coefficient for medical datasets

Montes V., Chincho J.M., Alvarez M.A., Soria L.M., Ortega J.A.
Computer Languages and Systems Department, University of Seville, 41500 Seville, Spain
{vermonval, joschicar}@alum.us.es, {maalvarez, lsoria, jortega}@us.es

Abstract

The standard backward propagation of errors algorithm (abbreviated as back-propagation algorithm) is commonly used for decision making in medicine. Using the back-propagation algorithm in medical diagnosis is desirable since it avoids human subjectivity and applies a large knowledge base, which makes this algorithm very reliable. However, it is generally believed that it is very slow if it does converge, especially if the network size is not of sufficient size compared to the problem at hand. A drawback of the back-propagation algorithm is that it has a constant learning rate coefficient, while different regions of the error surface may have different characteristic gradients. Variation in the nature of the surface may require a dynamic change of learning rate coefficient. A new back-propagation algorithm with momentum has been developed in order to be used to speed up the learning process, which accelerates the convergence of back-propagation algorithm.

1 Introduction

In many real-world situations, we are faced with incomplete information or noisy. It is also important to be able to make reasonable predictions on new cases of information available and a backpropagation network adapts its weights to acquire learning from a training set.

Back propagation is to propagate error backwards, ie, from the output layer to the input layer, through intermediate hidden layers and adjusting the weights of the connections in order to reduce the error. There are various versions of the backpropagation algorithm rules and connectionist different architectures to which they can be applied.

Examples of possible applications are the diagnosis of cardiac abnormalities, prescription diets or the diagnosis of abnormalities in electrocardiograms.

The backpropagation network is based on the generalization of the delta rule. Like the perceptron, Adaline and Madaline[Widrow and Lehr, 1990], backpropagation network is characterized by a layered architecture and strictly forward connections between neurons. Use supervised gradient-based network error with respect to weights learning neural networks.

The advantages and the disavantages of the artificial neural networks are described below[Tu, 1996].

1.1 Advantages of ANNs

- Are not linear distributed systems: a neuron it is a nonlinear element so that interconnect them (neural network) will also be a nonlinear device. This property allows the simulation of nonlinear and chaotic systems, simulation with the linear classical systems, can not be performed.
- Are fault tolerant systems: a neural network, as a distributed system, allows the fault of some individual elements (neurons) without significantly altering the total system response. This makes them particularly attractive compared to existing computers, typically such systems are sequential so that a failure in one of its components implies that the whole system does not work.
- Adaptability: a neural network has the ability to modify the parameters which influence its operation according to the changes that occur in their work environment (changes in the inputs, the presence of noise, etc..). Regarding adaptability to be taken into account that it can not be excessively large since it would lead to an unstable system have to respond to small disturbances.
- Establish nonlinear relationships between data: neural networks are able to relate two sets of data by complex relationships.
- Possibility of VLSI implementation: this capability allows these systems to be applied in real-time systems, simulating biological systems using silicon elements.
- Learning capability (automatic): neural models eliminate the need for adaptation of expert systems. The data inputs are fed directly to the prediction software, without interpretation or modification (not previously involved an expert who takes a mental model).

1.2 Disadvantages of ANNs

- It is necessary to know well the problem to be modeled.
- The black box effect: the data enters into the black box and the predictions are obtained, but are not usually re-

veals the nature of the relationships between the independent and dependent variables. In some cases, the neural networks do not explain, as with other traditional approaches.

- Lengthy processes.
- Require the definition of many parameters before applying the methodology.

2 PROBEN1

The PROBEN1 benchmark[Prechelt and others, 1994] is a collection of problems prepared for learning ANNs in order to test different algorithms and get a direct comparison of results. PROBEN1 contains 15 problems divided into 12 different areas, and also provides a set of rules and conventions advised regarding documentation of the results. The problems cover arguments in classification problems and functional interpolation. In our case it is used to test the operation of the developed system.

2.1 Backpropagation algorithm

The training of the neural network using the multipropagation algorithm or propagation or back-propagation of errors consists of the following steps:

1. The structure of the network (number of layers and of neurons in each layer) is decided: L layers, n neurons in the input layer (layer 1), m in the output (layer L).
2. An activation function of neurons differentiable is chosen. Usually sigmoidal type[Harrington, 1993] (ie S-shaped), ie the logistic function $g(x) = \frac{1}{1+e^{-x}}$.
3. The w_{ij} weights and the polarizations are randomly initialized and with small values $(-0.5, 0.5)$.
4. The training data is generated: set of tuples inputs - desired outputs, ie, if there are 2 inputs, 2 outputs and M training tuples are necessary:

$$\{(x_{11}, x_{12}), (y_{d11}, y_{d12})\}, \{(x_{21}, x_{22}), (y_{d21}, y_{d22})\}, \ldots, \{(x_{M1}, x_{M2}), (y_{dM1}, y_{dM2})\}$$

5. A training data is chosen, ie r: $\{(x_{r1}, x_{r2}), (y_{dr1}, y_{dr2})\}$.
6. The outputs of the network (x_{r1}, x_{r2}) with available weights propagating the values from the input neurons to forward are calculated, ie from $l = 1$ to L:
   - $in_i =$ input receiving a i unit.
   - $a_i =$ output of the i unit.
   - If i is an input neuron $(l = 1) \rightarrow a_i = x_{ri}$.
   - In a i neuron of layer l (with $l \neq 1$):

$$in_i = \sum_j w_{ij} a_j, a_i = g(in_i)$$

7. The difference between the outputs of the network for x_r data (i outputs of neurons output layer) obtained with current weights (a_i) and desired outputs (y_{dri}), so we get the error vector with the error of each output neuron for that data is calculated.
8. The weights of the network so that the error is minimized are adjusted. In the output layer:

$$w_{ji}(t+1) = w_{ji}(t) + \eta a_j \Delta_i, \Delta_i = g'(in_i)(y_{dri} - a_i)$$

But if the neuron does not belong to the output layer do not know what is the expected value of output: the same formula can not be used. How to update the connection weights of the hidden layers? Come back Δ_j calculating the error of each unit of the hidden layer $l-1$ from error units layer l with which they are connected j.

$$w_{kj}(t+1) = w_{kj}(t) + \eta a_k \Delta_j, \Delta_j = g'(in_j) \sum_i w_{ji} \Delta_i$$

Ie, each unit j is "responsible" for the error that each of the units to which sends its output, contributing in proportion to their weight. To calculate the modification of the weights, the error is calculated in the output stage and the change propagates backward: backpropagation.

9. The above steps for each pair of training (time) are repeated and iterated until the error for all training sets is acceptable.

3 Momentum

The system is an improved version of backpropagation, which uses a term called Momentum[Phansalkar and Sastry, 1994] for the elimination of local minimum.

The term Momentum introduces error of the previous weight as a parameter for the computation of the new change. This avoids the problems common swing with backpropagation algorithm when the error surface has a minimum very narrow area. Calculating weights corresponding to:

$$\delta_j = \begin{cases} (f_j'(net_j) + c)(t_j - o_j), & \text{if } j \text{ neuron is output} \\ (f_j'(net_j) + c) \sum_k \delta_k w_{jk}, & \text{if } j \text{ neuron is hidden} \end{cases}$$

The effect of these improvements is that the flat spots on the error surface are traversed relatively quickly with some big steps while the step size is decreased when the surface is irregular. This adaptation of the step size increases learning speed significantly.

Keep in mind that the above change in weight is lost every time the parameters are changed, new patterns are loaded, or network changes.

Momentum is a heuristic optimization technique while other techniques of numerical optimization. Examples of other optimizations backpropagation algorithm are:

- The QuickProp algorithm that significantly speeds up the gradient descent backpropagation algorithm.
- The LMBP is the fastest algorithm that has been proven to train multilayer neural networks of moderate size. Its main drawback is the memory requirements, if the network has more than a few hundred parameters the algorithm becomes impractical.

- The cascade-correlation algorithm has the particularity of hidden neurons use to minimize the residual error that is output to an input pattern, in addition to grow to reach their optimum size.

4 Experimentation

In our algorithm we used basic data types such as arrays, for example, to optimize the execution time, code is more complex to use these data types as it does not give us the ease of implementation that would give us other non-core types like lists but has advantages in runtime.

For each experiment we tested our algorithm with various inputs of hidden (hidden layer), string length n (learning factor). Mostly what we can change is the learning factor to bring us closer to the results obtained in the benchmark. For each example we calculate the mean square error, standard deviation and by the time it stops.

Problem	Training Set		Validation Set		Test Set		Test Set Clas-sifications		Overfit		Total Epochs		Revelant epochs		n
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	---
Cander1	3.168	2.4E-4	1.886	7.1E-4	1.167	0.001	1.72	0.0	5.044	0.0253	132.4	0.699	47.4	0.516	0.5
Cander2	2.178	0.002	1.743	0.0041	3.218	0.0032	4.569	0.091	5.033	0.0195	87.2	1.229	28.6	0.516	0.14
Cander3	1.785	0.0021	2.925	0.003	2.803	7.5E-4	4.598	9E-16	5.027	0.019	115.7	1.059	37.2	0.422	0.122
Card1	8.998	0.012	7.654	0.009	10.687	0.017	15.698	0.0	5.039	0.02	62.2	1.229	11.1	0.316	0.068
Card2	7.494	0.017	10.03	0.025	13.76	0.039	18.60	0.0	5.063	0.031	63.6	1.578	19.8	0.422	0.015
Card3	8.675	0.006	7.248	0.014	13.419	0.024	18.60	0.0	5.039	0.019	101.6	1.174	29.6	0.516	0.029
Heart1	10.69	0.01	13.37	0.031	14.83	0.002	19.56	0.0	5.00	0.004	133.3	17.16	1.4	0.516	0.101
Heart2	11.06	6.927	11.91	0.005	14.50	0.004	17.39	3.74	5.023	0.008	182.9	0.99	36.2	0.42	0.095
Heart3	10.31	0.001	10.32	0.008	16.89	0.004	24.78	3E-15	5.019	0.013	147.3	1.15	36.2	0.42	0.065
Heartc1	10.09	0.286	7.72	0.13	15.75	0.77	18.79	0.42	5.15	0.19	139.1	44.27	85.1	10.78	0.345
Heartc2	9.67	0.01	19.7	0.02	6.00	0.02	8.0	0.0	5.00	0.003	134.1	13.14	9.2	0.42	0.185
Heartc3	10.12	0.03	12.80	0.04	11.44	0.01	13.93	0.37	5.13	0.06	24.5	0.85	5.7	0.48	0.035
Horse1	10.003	0.059	14.542	0.068	13.179	0.048	18.462	0.429	5.212	0.095	28.1	0.994	7.2	0.788	0.06
Horse2	6.434	0.231	15.314	0.072	18.70	0.036	25.01	0.247	5.06	0.038	44.1	4.86	3.9	0.567	0.075
Horse3	9.464	0.143	15.130	0.090	15.239	0.087	21.94	0.320	5.101	0.044	26.3	2.162	4.1	0.316	0.036

Figure 1: Table of results

Some conclusions we can take from the tests performed. First the mean of the times out relevant PROBEN1 documentation different from this may be because the algorithm implemented for this work that is more effective. But the results in the stdev column of Total Epochs and Revelants Epochs, are too low compared with respect to the document because each RUN out these results are very similar. Finally, the column Overfit values goes too high, this may be shaped by the algorithm is implemented.

Meanwhile, here in this document have not included all the examples to classify the test set. This is because the examples are not included in this table because their values did not correspond to the table and also proben learning factors were difficult to predict its range. This may be due to several reasons, one of them that this work has not been taken into account that the inputs or outputs can be negative as well as having problems in finding their learning facto. The foregoing on negative values, could be a point when an improved algorithm in the future.

This table contained a number of problems, all problems table PROBEN1 not be classified by the algorithm implemented in this work because when you find a learning factor that approximates the results to the table PROBEN1, if we make several executions , we do very different values at

Problem	Training Set		Validation Set		Test Set		Overfit		Total Epochs		Revelant epochs		n
	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	Mean	Stdev	
Building1	0.28	0.0	1.32	0.002	1.15	0.40	5.07	0.11	308.6	945.6	3.0.	0.0	3.77
Buildin2	1.48	0.009	1.31	0.145	1.43	0.521	19.43	18.53	303.1	947.5	214.9	675.35	3.21
Flare1	0.341	0.00	0.33	0.00	0.60	0.00	5.03	0.01	44.3	5.96	3.9	0.31	1
Flare2	0.43	0.00	0.42	0.00	0.32	0.00	5.20	0.11	37.2	1.31	17.9	1.19	0.08
Flare3	0.41	0.00	0.48	0.00	0.36	0.00	5.03	0.02	37.5	3.1	2.0	0.0	0.44

Figure 2: Problems that has no a classification in the test set

relevant times and total times, for example in learning building3.dt with factor 3.18, at different values minds if we complete several RUN.

5 Conclusions

The conclusions of this work can make training an artificial neural network, is that for almost all problems proben the target will reach. For other deployment for various reasons, has not been po-dido achieve the overall objective of the work.

As future work and extension could be discover-fix two problems in the previous sections, trying to give the reason why no data required is the PROBEN1. The algorithm is also able to expand not only have the input layer and out-put but it includes a hidden or intermediate layer endow a more general approach to ANN work.

Acknowledgments

This research is partially supported by the projects of the Spanish Ministry of Economy and Competitiveness ARTEMISA (TIN2009-14378-C02-01) and Simon (TIC-8052) of the Andalusian Regional Ministry of Economy.

References

[Harrington, 1993] Peter de B Harrington. Sigmoid transfer functions in backpropagation neural networks. *Analytical Chemistry*, 65(15):2167–2168, 1993.

[Phansalkar and Sastry, 1994] VV Phansalkar and PS Sastry. Analysis of the back-propagation algorithm with momentum. *Neural Networks, IEEE Transactions on*, 5(3):505–506, 1994.

[Prechelt and others, 1994] Lutz Prechelt et al. Proben1: A set of neural network benchmark problems and benchmarking rules. *Fakultät für Informatik, Univ. Karlsruhe, Karlsruhe, Germany, Tech. Rep*, 21:94, 1994.

[Tu, 1996] Jack V Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of clinical epidemiology*, 49(11):1225–1231, 1996.

[Widrow and Lehr, 1990] Bernard Widrow and Michael A Lehr. 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proceedings of the IEEE*, 78(9):1415–1442, 1990.