International Core Journal of Engineering Volume 7 Issue 3, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

Application of Neural Network Method based on HJB Equation

Xunpeng Sun
Shanghai Maritime University, Shanghai 200135, China.
gastbying@163.com

Abstract

In this paper, a cybernetic method is proposed for batch and instantaneous update of weights in neural networks. Using the popular Hamilton-Jacobi-Bellman (HJB) equation, a new law of optimal weights is generated. The main contribution of this paper is that for any neural network using HJB, a closed form solution with optimal cost and weight updates can be obtained. This approach has been compared with some of the best performance learning algorithms available. The results show that this method has better performance in computation time and effect. Benchmark data, such as 8-bit parity, breast cancer, and credit approvals, as well as 2D Gabor functions have been used to verify our claims, and this article introduces this approach using the decoder as an example.

Keywords

Hamilton-Jacobi-Bellman (HJB); Back Propagation Neural Network.

1. Introduction

Deep learning algorithms have recently become more and more popular and useful algorithms, but the success of deep learning or deep neural networks[1] is due to the endless emergence of neural network model architectures. In this paper, the author reviews the architecture development of deep neural networks in the past 18 years since 1998.

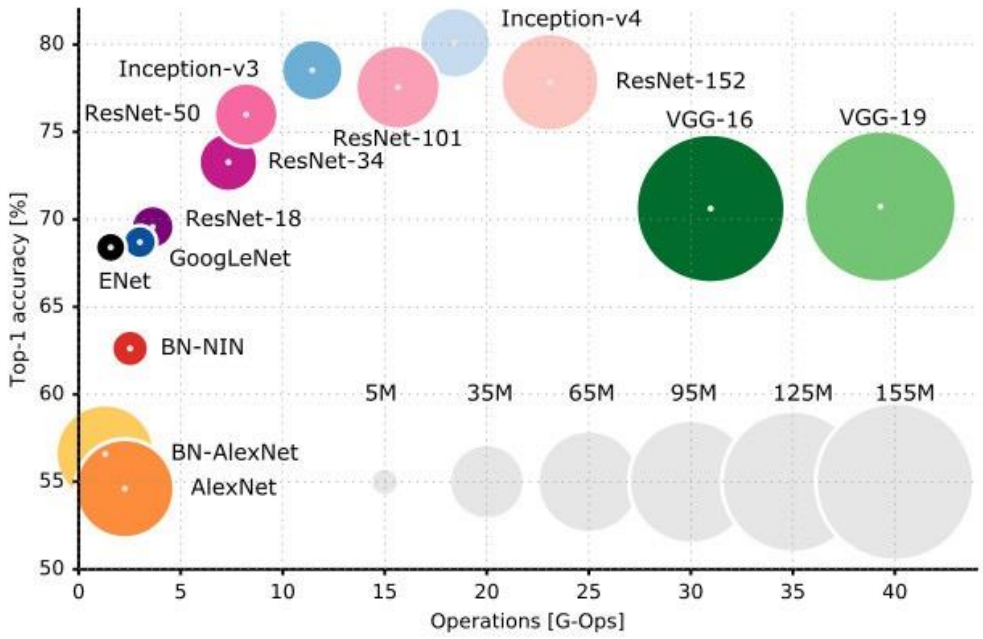

Fig 1. The accuracy and complexity of the algorithms

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

On the axis of the graph we can see that the abscissa is the complexity of the operation, and the ordinate is the accuracy. At the beginning of model design, the greater the weight of the model, the higher the accuracy of the model. Later, after the appearance of network architectures such as ResNet, Googlenet, Inception, etc., the weight parameters of the model continue to decline when the same or higher accuracy is achieved. Now, it's important to note that it doesn't mean that the more to the right of the x-coordinate, the more time it takes. The time is not counted here, but the model parameters and the accuracy of the network are compared vertically and horizontally.

In 1943, psychologist Warren McCulloch and mathematical logician Walter Pitts put forward and gave the concept of artificial neural network and the mathematical model of artificial neural network in their joint paper A Logical Calculus of the Ideas Immanent in Nervous Activity, which initiated the era of artificial neural network research. In 1949, psychologist Donald Hebb described The rules of neuronal learning in a paper he published in The Organization of Behavior.

Further, American neuroscientist Frank Rosenblatt proposed a machine that can simulate human perception, and called it "perceptron". In 1957, at the Cornell Aeronautical Laboratory, he successfully completed the simulation of a perceptron on an IBM704, and in 1960, he implemented a perceptron-based neural computer -- Mark1, which could recognize some English letters.

In 1985, Geoffrey Hinton used multiple hidden layers to replace the original single feature layers in the perceptron, and used a back-propagation algorithm (proposed in 1969, practicable in 1974) to calculate network parameters

In 1989, Yann LeCun [2][3] et al. used deep neural networks to recognize the handwritten characters of postcodes in letters. Later, LeCun further used CNN (Convolutional Neural Network) [4][5][6] to complete handwritten character recognition of bank checks, and the recognition accuracy reached commercial level. Although the algorithm was a great success, it took about three days to train on the dataset.

The network structure is divided into input layer, multiple hidden layers and output layer. The weight is initialized randomly before the network is trained, and the network parameters are adjusted by the BP algorithm [7][8][9].

The BP algorithm doesn't always work well. Even with stochastic gradient descent, BP algorithm is still easy to fall into local optimal solution. And with the increase of network layers, the difficulty of training becomes more and more difficult.

In 2006, Hinton proposed the Deep Belief Network (DBN)[10][11][12], a deep network model. Use a greedy unsupervised training approach to problem solving and get good results. The training method of DBN (Deep Belief Networks) reduces the difficulty of learning hidden layer parameters. The relationship between the training time and the size and depth of the network is almost linear. In 2010, the U.S. Defense Department's DARPA plans to fund its first deep learning program.

In 2011, Microsoft Research and Google speech recognition researchers successively used DNN technology to reduce the error rate of speech recognition by 20%-30%, which is the biggest breakthrough in this field in 10 years

In 2012, Hinton reduced ImageNet's top five error rate for image classification problems from 26 percent to 15 percent. In the same year, Andrew Ng and Jeff Dean built the Google Brain project, using the parallel settlement platform containing 16,000 CPU cores to train the deep network of more than 1 billion neurons, making a breakthrough in the field of jade and jade recognition and image recognition.

In 2013, DNN Research, the company Hinton founded, was acquired by Google, and Yann LeCun joined Facebook's AI lab.

In 2014, Google increased the accuracy of language recognition from 84 percent in 2012 to 98 percent today, and the accuracy of language recognition on mobile Android systems increased by 25 percent. In terms of face recognition, Google's Facenet system achieved 99.63% accuracy on the LFW[13][14].

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

In 2015, Microsoft used the residual learning method of deep neural network to reduce the classification error rate of ImageNet to 3.57%, which was lower than the human eye recognition error rate of 5.1% in similar experiments, and the neural network it used has reached 152 layers.

Since the emergence of BP algorithm, fast convergence has been the focus of researchers. One of the earliest works of Hagan and Menhaj [2] was to use Levenberg-Marquardt (LM) algorithm [15][16][17] to train the weights of multi-layer networks in batch mode. In order to improve the performance and convergence speed of BPS algorithm [18], several improved algorithms are proposed.

In this paper, we suggest using Hamilton-Jacobi-Bellman (HJB) equation for solving this problem. The HJB equationcomes from dynamic programming, which is a popularapproach for optimal control of dynamical systems. In the proposed scheme, the weight update law has beenconverted into a control problem and dynamic optimizationhas been used to derive the update law. The derivation ofthe HJB based weight update law is surprisingly simple andstraightforward. The closed form solution for the optimalcost and optimal weight update law have simple structures.The proposed approach has been compared with some of theexisting best performing learning algorithms and is found tobe faster in convergence in terms of computational time.

2. HJB based offline learning of BP

2.1 2.1.1 BP Neural Network

Consider two layers of BPNN with N0 −N1− N2 structure, where N0, N1 and N2 represent the number of neurons in the input layer, hidden layer and output layer, respectively.

where, i_0 i_1 i_2 index the neurons in the input, hidden andoutput layers, respectively.

The output $y_p \in \mathbb{R}^{Nl}$ (l is the index of output layer) for agiven input pattern $x_p \in \mathbb{R}^{N0}$ can be written as,

$$y_p = f(\hat{w}, x_p) \tag{1}$$

Here, $\hat{w} \in \mathbb{R}^{Nw}$ is the vector of weight parameters involvedin the BPNN to be trained, with total $N_w$weight parameters.The derivative of y w.r.t. time t is

$$\dot{y}_p = \frac{\partial f(\hat{w}, x_p)}{\partial \hat{w}} \dot{\hat{w}} = J_p \dot{\hat{w}} \tag{2}$$

where, $J_p = \frac{\partial f(\hat{w}, x_p)}{\partial \hat{w}}$ is the Jacobian matrix, whose elements are $J_{p,ij} = \partial y_{p,i} / \partial \omega_j$. The desired output is given by $y_p^d = f(w, x_p)$ and its derivative with respect to time is

$$\dot{y}_p^d = J_p \dot{w} = 0 \tag{3}$$

The estimation error is $e_p = y_p^d - y_p$ and its derivative w.r.t. time t is

$$\dot{e}_p = \dot{y}_p^d - \dot{y}_p = -J_p \dot{\hat{w}},\ \dot{y}_p^d = 0 \tag{4}$$

The optimization of the neural network weights is formu-lated as a control problem.

$$\dot{e}_p = -J_p \dot{\hat{w}} = -J_p u \tag{5}$$

The control input updates the weights as $u = \dot{\hat{w}}$.

In the batch mode, all the patterns are learnt simultaneously. Hence, the dynamics can be written jointly as

$$\dot{e} = -Ju \tag{6}$$

where, $e = [e_1^T, e_2^T, \ldots, e_{N_p}^T]^T$, $J = [J_1^T, J_2^T, \ldots, J_{N_p}^T]^T$. Hence, e is an $N_l N_p \times 1$ vector, J is an $N_l N_p \times N_\omega$ matrix and u is an $N_\omega \times 1$ vector.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

2.2 Optimal Weight Update

the cost function is defined over time interval $(t, T]$ as

$$V(e(t)) = \int_t^T L(e(\tau), u(\tau))\, d\tau \tag{7}$$

Where

$$L(e, u) = \frac{1}{2}(e^T e + u^T R u) \tag{8}$$

With R as a constan $N_\omega \times N_\omega$ matrix. Here,t signifiesthe iterations to update w. Our goal is to find an optimal weight update law $u(t)$ which minimizes the aforementioned cost function. We can assume $T \to \infty$. We use the following formulation which is popularly known as the Hamilton-Jacobi-Bellman (HJB) equation .

$$\min_u \{\frac{dV^*}{de}\dot{e}(t) + L(e(t), u(t))\} = 0 \tag{9}$$

Putting the expressions for $\dot{e}(t)$ and $L(e(t), u(t))$ from eqs.(6) and (8), respectively,

$$\min_u \{-\frac{dV^*}{de}Ju(t) + \frac{1}{2}e(t)^T e(t) + \frac{1}{2}u(t)Ru(t)\} = 0 \tag{10}$$

Here, $\frac{dV^*}{de}$ is $1 \times N_l N_p$ vector. Differentiating with respect to u, we get the optimal update law as

$$u^*(t) = R^{-1}J^T(\frac{dV^*}{de})^T \tag{11}$$

In order to find the expression for $(\frac{dV^*}{de})$, we put the optimal u from eq. (11) in eq. (10),

$$e(t)^T e(t) - (\frac{dV^*}{de})JR^{-1}J^T(\frac{dV^*}{de})^T = 0 \tag{12}$$

A proper solution of eq. (12) should lead to $JR^{-1}J^T$ to be positive definite.

This is an under-determined system of equations. However, the optimal input must stabilize the system. The stability of thesystem can be analyzed with the help of a Lyapunov functiondefined as

$$\upsilon(e) = \frac{1}{2}e^T e \tag{13}$$

The equilibrium point $e = 0$ is stable if $\dot{\upsilon}(e)$ is negative definite.

$$\dot{\upsilon}(e(t)) = e^T \dot{e} \tag{14}$$

$$= -e^T J u^*(t) \tag{15}$$

$$= -e^T JR^{-1}J^T(\frac{dV^*}{de})^T \tag{16}$$

If one selects the following form of dV^*/de,

$$\frac{dV^*}{de} = e(t)^T C(t)^T \tag{17}$$

where, $C(t)$ is chosen to be a positive definite matrix, then $\dot{\upsilon}(e(t))$ becomes negative definite. In order to find the expres-sion for $C(t)$, we substitute eq. (17) in eq. (12),

$$e(t)^T(I - C(t)^T JR^{-1}J^T C(t))e(t) = 0 \tag{18}$$

where, I is $N_p \times N_p$ identity matrix. For this to be true forall $e(t)$,

$$C(t)^T JR^{-1}J^T C(t) = I \tag{19}$$

To find a solution for $C(t)$, we decompose $JR^{-1}J^T = U\Sigma U^T$ into eigenvectors U and a diagonal matrix of eigenval-uesΣ. Since, $JR^{-1}J^T$ is a symmetric positive definite matrix, $U^T U = UU^T = I$ and all eigenvalues are positive. Now, $C(t)$ can assume the following form to satisfy eq. (19)

$$C(t) = U\Sigma^{-\frac{1}{2}}U^T \tag{20}$$

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

This form assures $C(t)$ to be positive definite, so that thesystem is stable around $e = 0$. However, for numerical stabil-ity while implementation, a small positive term $(= 10^{-4}I)$ isadded to Σso as to avoid numerical instability.

Finally, we get the optimal weight update law by combining equations (11) and (17) as

$$\dot{\hat{w}} = u^*(t) = R^{-1}J^T C(t)e(t) \tag{21}$$

where, $C(t)$ is given by eq. (20).

For the BPNN considered in eqs. the Jacobian canbe obtained as follows. For the output layer,

$$\frac{\partial f(w,x_p)}{\partial \omega_{i_2 i_1}} = y_{p,i_2}(1 - y_{p,i_2})v_{i1} \tag{22}$$

and for the hidden layer,

$$\frac{\partial f(w,x_p)}{\partial w_{i_1 i_0}} = \sum_{i_2} y_{p,i_2}(1 - y_{p,i_2})\, \omega_{i_2 i_1} v_{i_1}(1 - v_{i_1}) x_{p,i_0} \tag{23}$$

In the online mode, the weight vector is updated with each input pattern. In this case, the network dynamics can be presented as

$$y = f(\hat{w}) \tag{24}$$

where the network output is simply observed as a function ofnetwork weights only. Here the desired output y^d is observedand the network response is compared to find

$$e = y^d - y \tag{25}$$

The problem is to find u so as to minimise

$$V(e(t)) = \int_t^\infty \frac{1}{2}(e^T e + u^T R u)d\tau \tag{26}$$

One should note that this cost function is not pertaining to any specific pattern rather the network is subjected to various inputs while the instantaneous errore is computed according to Eq. (25). The optimal instantaneous weight update law, as derived using the HJB equation and Lyapunov stability criterion, is given by

$$\dot{\hat{w}} = u^*(t) = R^{-1}J^T C(t)e(t) \tag{27}$$

$$C(t) = U\Sigma^{-\frac{1}{2}}U^T \tag{28}$$

With U and Σas the eigenvectors and eigenvalues of $JR^{-1}J^T$.

3. Conclusion

The efficiency of the proposed HJB based optimal learning scheme is substantiated with the help of several benchmark learning problems. Notably, the purpose of the presented work is to optimize a given cost function and not explicitly to design robust classification algorithms.

An example of the method proposed in this paper for image processing is given below

Fig 2 are the results of image processing by different methods. From left to right, they are Adadelta, Adagrad, Adam, LF and our proposed HJB approximation method, The HJB approximation method proposed in this paper has the best decoding efficiency of 76.83%.

This section analyzes the convergence behavior of various algorithms used in this paper. It has been observed that the proposed HJB algorithm is much faster than the other algorithms for both online as well as offline learning. It will be useful at this stage to compare the computational complexity of one iteration of each of these algorithms. All the algorithms discussed in this paper are based on deterministic iterative update of weights. Moreover, all of these involve the calculation of the Jacobian J. They mostly differ in their weight update schemes, u. Certainly, a single iteration of HJB and LM schemes involves more computations than BP and LF schemes. However, the number of

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

iterations required for HJB and LM are significantly small as compared to those for BP and LM. Hence, the former ones take much less convergence time than the later ones

HJB equation is well known for dynamic optimization. In this paper, the weight update process in BPNN has been formulated as a dynamic optimization problem. Applying HJB equations, the optimal weight update laws have been derived in both batch and online (instantaneous) modes.

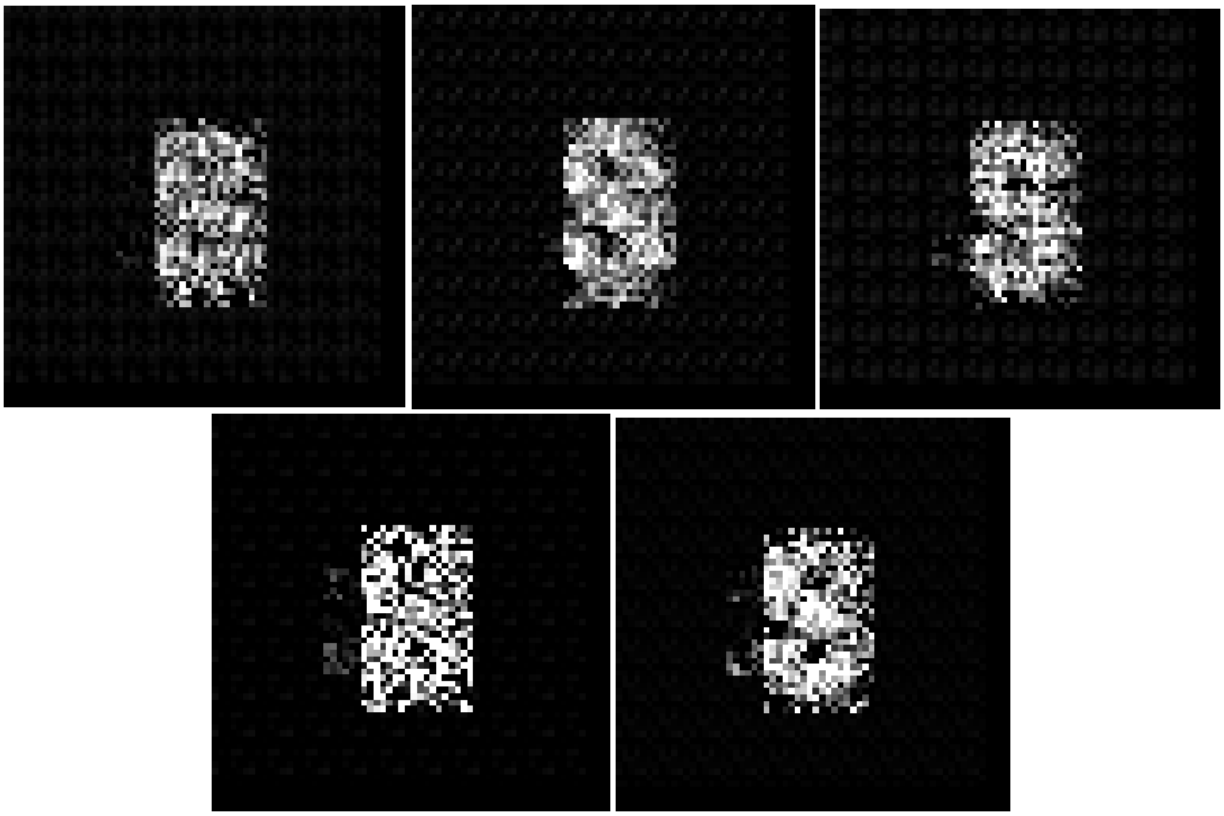

Fig 2. The results of five approximate methods

This paper analytically analyzes the convergence behavior of various popular learning algorithms as why they are liable to be stuck in local minima. It is shown that the Lyapunov function drags the solution to the local minimum. The proposed scheme ensures faster convergence rate as compared to the existing schemes, including LM. Moreover, the weight update law using HJB equation has been derived for both offline as well as online modes, whereas LM can be applied only in the offline mode. Our study also shows thatthe proposed HJB scheme works well even with large network size, while the LM method deteriorates in performance as the size of the network increases, as also observed by Xie.

Acknowledgements

Natural Science Foundation.

References

[1] Nishani E , Cico B . Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation[C]// 2017 6th Mediterranean Conference on Embedded Computing (MECO). IEEE, 2017.

[2] Cox R, Haskell B, Lecun Y, et al. On the Applications of Multimedia Processing to Telecommunications [M]. 1997.

[3] Tygert M, Bruna J, Chintala S, et al. A Mathematical Motivation for Complex-Valued Convolutional Networks[J]. Neural Computation, 2016:1-11.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202103_7(3).0036

[4] Samudre P, Shende P, Jaiswal V. Optimizing Performance of Convolutional Neural Network Using Computing Technique[C]// 2019 IEEE 5th International Conference for Convergence in Technology (I2CT). IEEE, 2019.

[5] Moore B J, Berger T, Song D. Validation of a Convolutional Neural Network Model for Spike Transformation Using a Generalized Linear Model[C]// 2020 42nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) in conjunction with the 43rd Annual Conference of the Canadian Medical and Biological Engineering Society. IEEE, 2020.

[6] Xin R, Zhang J, Shao Y. Complex Network Classification with Convolutional Neural Network[J]. Tsinghua Science & Technology, 2018, 25(4).

[7] Mei N, Qian F, Yan L, et al. Energy Efficiency Prediction of Screw Chillers on BP Neural Network Optimized by Improved Genetic Algorithm[C]// 2018 International Computers, Signals and Systems Conference (ICOMSSC). 2018.

[8] Shifei D, Chunyang S. Application of optimizing BP neural networks algorithm based on Genetic Algorithm [C]// Control Conference. IEEE, 2010.

[9] Mei N, Qian F, Yan L, et al. Energy Efficiency Prediction of Screw Chillers on BP Neural Network Optimized by Improved Genetic Algorithm[C]// 2018 International Computers, Signals and Systems Conference (ICOMSSC). 2018.

[10] Luo X, Xu S. Forest Mapping from Hyperspectral Image Using Deep Belief Network[C]// 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN). 2019.

[11] Tao C, Wang X, Gao F, et al. Fault diagnosis of photovoltaic array based on deep belief network optimized by genetic algorithm[J]. Chinese Journal of Electrical Engineering, 2020, 6(3):106-114.

[12] Arsa D M S, Jati G, Mantau A J, et al. Dimensionality reduction using deep belief network in big data case study: Hyperspectral image classification[C]// International Workshop on Big Data & Information Security. IEEE, 2017.

[13] Kayal S. Experiments on the LFW database using curvelet transforms and a random forest-kNN cascade[C]// International Conference on Digital Information Processing & Communications. IEEE, 2012.

[14] Kayal, University A , Espoo, et al. Face verification experiments on the LFW database with simple features, metrics and classifiers[J].

[15] Yilmaz S, Dlmen E, Beyhan S. Cascaded ABC-LM Algorithm Based Optimization and Nonlinear System Identification[C]// International Conference on Electronics, Computer and Computation, IEEE. IEEE, 2013.

[16] Cheng H, Cui L, Li J. Application of improved BP neural network based on LM algorithm in desulfurization system of thermal power plant[C]// 2017 Chinese Automation Congress (CAC). 2017.

[17] Li X , Wang Y. Prediction model of biogas production for anaerobic digestion process of food waste based on LM-BP neural network and particle swarm algorithm optimization[J]. 2017:7629-7633.

[18] Xian, Zhou, Chao, et al. Low-Complexity Carrier Phase Recovery for Square M-QAM Based on S-BPS Algorithm[J]. IEEE Photonics Technology Letters, 2014, 26(18):1863-1866.