Towards the dominant eigenvalue and the corresponding eigenvector of a symmetric square matrix using neural networks

H. Khelil[1(*)], N. Khelil[2] and L. Djerou[3]
[1] *Mohamed Boudiaf University, M'sila, Algeria.*
[2] *Mohamed Khider University, Biskra, Algeria.*
[3] *Mohamed Khider University, Biskra, Algeria.*
E-mail [1] : *hathem.khelil@univ-msila.dz*, E-mail [2] : *naceur.khelil@univ-biskra.dz*,
E-mail [3] : *l.djerou@univ-biskra.dz*

Abstract

In this paper, a neural network is introduced and utilized to calculate the dominant eigenvalue and related eigenvector of a symmetric square matrix. These neural networks are built because of their capacity to deal with custom loss functions and compute X estimations for large-scale problems, these neural networks are constructed.

Keywords: *Recurrent neural networks, Dominant eigenvalue, Eigenvector, Symmetric matrix.*

1. Introduction

Neural network-based Eigen problem solving is an exciting and relatively new topic of computational mathematics and machine learning *[3]- [7]*. The aim of an Eigen problem is to find the eigenvalues and eigenvectors of a given symmetric square matrix, which has several applications in the fields of physics, engineering, data analysis, and many more. Numerical algorithms like the power iteration, QR iteration, or Jacobi method have historically been used to solve eigen problems. In particular, for big matrices or high-dimensional issues, these methods can be computationally expensive and time-consuming *[2]- [4]*. An alternate strategy that might offer quicker and more effective solutions for specific Eigen problem types is neural networks.

In this work, neural networks are described and utilized to find the dominating eigenvalue and accompanying eigenvector of a symmetric square matrix. These neural networks are developed due to their capacity to deal with custom loss functions and compute x estimations for large-scale challenges. Thus, the goal of this work is to solve a linear system *[1]*.

$$(A - \lambda)x = 0 \qquad (1)$$

The rest of this paper is structured as follows. Section (2) will look at the attributes of the network model's equilibrium points. In section (3), Neural Implementation Description, an intriguing depiction of network solutions will be constructed. The network's convergence will be covered in section example and discussion (4). Finally, we end with a conclusion in section (5).

2. Problem formulation and neural solution

Let A of $R^{n\times n}$ be a symmetric matrix. We consider the constrained maximization problem *[6]:*

$$\begin{cases} max(x^TAx)^2 \\ subject\ to\ x^Txx = 1 \end{cases} \tag{2}$$

We can use the Lagrange multiplier method to solve such optimization problems. Consider λ the Lagrange multiplier. The task then becomes identical to maximizing.

$$E(x,\lambda) = (x^TAx)^2 - \lambda(x^Tx - 1)$$
(3)
$$\frac{\partial E}{\partial x} = 2Ax - \lambda(2x) = 0 \Rightarrow Ax = \lambda x$$
$$\frac{\partial E}{\partial \lambda} = 2Ax - \lambda(2x) = 0 \Rightarrow x^Tx = 1$$

This implies that x, λ must be an eigen pair of A. For any solution $\lambda = \lambda_i$, $x = v_i$, the objective function takes the value, $v_i^TAv_i = v_i^T(\lambda_i v_i) = \lambda_i v_i^T v_i = \lambda_i$. Therefore, the eigenvector v_1 (corresponding to dominant eigenvalue λ_1 of A is the global maximizer, and it yields the absolute maximum value λ_1 . Similarly, the eigenvector v_n corresponding to the least eigenvalue λ_n is the global minimizer with absolute minimum λ_n.

$$E(x,\lambda) = (x^TAx)^2 - \lambda(x^Tx - 1)$$

Gradient descent can be used to minimize$-E(x,\lambda)$. The gradient of $E(x,\lambda)$ with respect to x is

$$\begin{aligned}\nabla_x E &= -2(x^TAx)Qx + 2\lambda x \\ &= -2(x^TAx)(Ax - \lambda x)\end{aligned}$$

In dynamical system form, the gradient above can be expressed as:

$$\Delta_t x = -2(x^TAx)(Ax - (x^TAx)x)$$

or, according to the learning rule:

$$\Delta_t x = \eta(x^TAx)(Ax - (x^TAx)x) \tag{4}$$

This type is obviously a recurrent neural network. Network (4) is a nonlinear differential equation *[8]- [9]*. Its dynamic behavior is critical and vital in its applications. From an engineering standpoint, neural networks with well-understood dynamic characteristics are most

appealing. The dynamic behaviors of the network model are clearly explored in this research.
Equation (4) can be utilized to solve the typical eigenvalue problem as indicated in equation (1). In this work, we limit ourselves to find the dominant eigenvalue and the corresponding eigenvector of a symmetric square matrix *[5]*.

3. Description of Neural Implementation

The given research focuses primarily on the classical neural network differential equation, as indicated in equation (4), where $A = (a_{ij}), i, j = 1,2, \dots, n$ are symmetric matrix that need to calculate their dominant eigenvalue and related eigenvector, $x(t) = [x_1(t), x_2(t), \dots, x_n(t)]$ is a column vector representing the states of neurons in the neural network dynamic system, as well as the elements of a symmetric square matrix A representing the connection weights between those neurons.

In practice, we always need a non-zero column vector,
$x(0) = [x_1(0), x_2(0), \dots, x_n(0)]^T$ to begin the neural network system using the following update rule:
$x(t+1) = x(t) + \eta x^T(t)Ax(t)\left(Ax(t) - x^T(t)Ax(t)x(t)\right)$, where t denote the iteration at t and η is a small time step.

Iteration ends when $\|x(t+1) - x(t)\| < \varepsilon$, where ε is a small constraint error that can be predetermined. If $\|x(t+1) - x(t)\| < \varepsilon$, we could consider that $\|x(t+1) - x(t)\| = 0$, that is to say, $Ax(t) = x^T(t)Ax(t)x(t)$, according to the theory in section (2), $x(t)$ is the eigenvector corresponding to the dominant eigenvalue, which can be written as $x^T(t)Ax(t)$.

4. Example and Discussion

This section will provide an example of computer simulation results to demonstrate the preceding theory. The simulation will demonstrate that the suggested network can calculate the the eigenvector corresponding to the dominant eigenvalue of any symmetric matrix.

In a simple way, a symmetric matrix A could be randomly generated. Allow B to be any randomly generated real matrix, define $A = BB^T$. Obviously A is a symmetric matrix.

$$A = \begin{bmatrix} 0.0263 & 0.1316 & 0.2145 & 0.1210 \\ 0.1316 & 0.3624 & 0.1967 & 0.5974 \\ 0.2145 & 0.1967 & 0.2030 & 0.0128 \\ 0.1210 & 0.5974 & 0.0128 & 0.6820 \end{bmatrix}$$

$$x = \begin{bmatrix} 0.5383 \\ 0.9961 \\ 0.0782 \\ 0.4427 \end{bmatrix}$$

For the reader's reference, the true largest modulus of the eigenvalues of A and the corresponding eigenvector computed by MATLAB are:
Dominant eigenvalue, $\lambda_{max} = 1.1955$,
$v_{max} = [0.1776 \quad 0.6089 \quad 0.1688 \quad 0.7545]^T$

Table 1. Code

```
function [c, v] = evann(A,x)
B=rand(4); A=B.*B'; Initial=rand(4,1);
x=Initial;
eta = 0.01;
for i=1:50
x = x+eta*(x'*A*x)*(A*x-(x'*A*x)*x);
c = (x'*A*x); v=x;
end
end
```

When using the network model the estimated dominant eigenvalue is $\lambda_{max} = 1.1955$ and the corresponding eigenvector is:
$v_{max} = [0.1781 \quad 0.6091 \quad 0.1696 \quad 0.7541]^T$.

As shown in Figure 1 the convergence of computational of $\lambda(t)$ to λ_{max}, which is synthesized by weight-updating formula (4). Evidently, after 20 iterations, $\lambda(t)$ in the above MATLAB code Table. 1 could converge to 1.1955 the dominant eigenvalue of A.

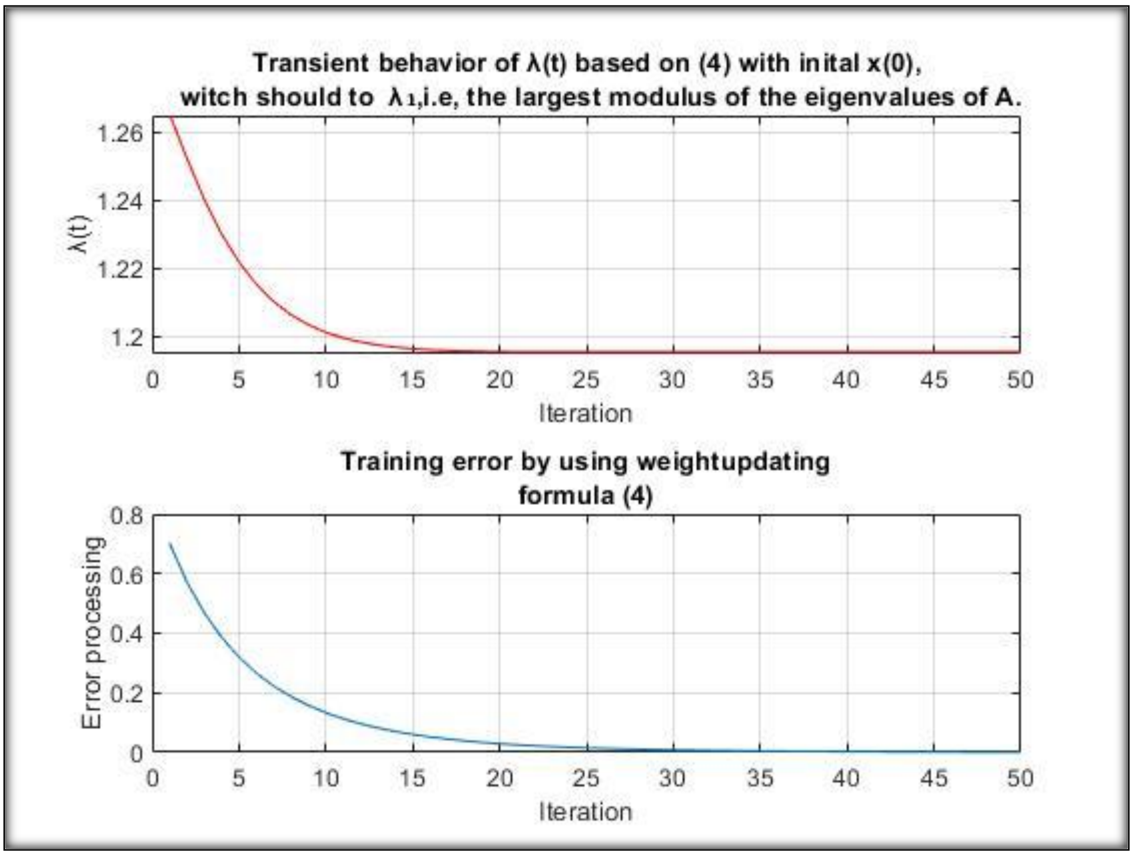

Figure 1. Estimation of the dominant eigenvalue of the matrix A.

5. Conclusion

This paper examines the issue of finding the eigenvectors of a symmetric square matrix using a neural network approach. It has been suggested that a class of artificial recurrent neural networks can be used to calculate the eigenvector that correspond to the dominant eigenvalue of any real symmetric square matrix. This network concept can produce excellent computing performance since it supports asynchronous parallel processing. The mathematical knowledge of the network model's dynamic behaviors is rigorous and unambiguous. According to simulation findings, the network model is effective.

References

[1] *M. B. Allen III and E. L. Isaacson, "Numerical-Analysis for Applied Science", Wiley-Interscience. Publications,* ***(1998).***

[2] *Burden, R. L. and Faires, J. D., "Numerical Analysis", Fourth Edition. PWS-Kent, Boston, MA* ***(1989).***

[3] *K. I. Diamantaras, K. Hornik and M. G. Strintzis, "Optimal linear compression under unreliable representation and robust PCA neural models", in IEEE Transactions on Neural Networks, vol. 10, no. 5, , Sept.* ***(1999)****, pp. 1186-1195.*

[4] *B.N. Parlett, "The Symmetric Eigenvalue Problem". Englewood Cliffs, NJ: Prentice-Hall,* ***(1980)****.*

[5] *Shaham, U., Stanton, K. P., Li, H., Basri, R., Nadler, B. & Kluger, Y. " SpectralNet: Spectral Clustering using Deep Neural Networks". ICLR (Poster),: OpenReview.net.* ***(2018)*** *April 30-May 3.*

[6] *William W. Hager, "Minimizing a quadratic over a sphere", SIAM Journal on Optimization, vol. 12, no.1,* ***(2001)*** *, pp. 188--208.*

[7] *Yi, Zhang et al., "Neural networks based approach for computing eigenvectors and eigenvalues of symmetric matrix", Computers & Mathematics With Applications 47* ***(2004)****, pp. 1155-1164.*

[8] *Zou, Xiongfei & Tang, Ying & Bu, Shirong & Luo, Zhengxiang & Zhong, Shouming. Neural-Network-Based Approach for Extracting Eigenvectors and Eigenvalues of Real Normal Matrices and Some Extension to Real Matrices, Journal of Applied Mathematics****.(2013)****. 10.1155/2013/597628.*

[9] *Zhang Q, Leung YW. "A class of learning algorithms for principal component analysis and minor component analysis". IEEE Trans Neural Netw.* ***(2000)****;11(1): , pp.200-4. doi: 10.1109/72.822522. PMID: 18249751.*