Stochastic Computation of Dominant Eigenvalue and the Law of Total Variance

George M. Georgiou, Kerstin Voigt, Haiyan Qiao

***Abstract*—Oja's neuron is extended to find the dominant eigenvalue alongside the computation of the dominant eigenvector. This is achieved through a stochastic gradient descent learning rule that computes the second moment of the neuron output. The effectiveness of this family of learning rules is further demonstrated in a network that verifies the law of total variance. The inputs are generated by a doubly stochastic process, and conditional means and variances are accurately computed and propagated in the network. The law of total variance has been recently used in the analysis of biological experiments to explain neural processes.**

I. Introduction

VARIANCE of inputs is an important statistical quantity both in artificial neural networks, and in related areas such as pattern recognition, signal processing and neuroscience. The weight vector in Oja's neuron [1] converges to the direction of the dominant eigenvector of the correlation matrix of the inputs. However, at the conclusion of training, no information on the variance of the inputs is retained nor can it be reconstructed since the correlation matrix is neither computed nor saved.

We address this by extending Oja's neuron to computing the dominant eigenvalue alongside the dominant eigenvector. The dominant eigenvalue itself is the variance of the inputs in the direction of the dominant eigenvector. Its computation is achieved by applying similar techniques of a recent paper [2] in which it was shown that statistical quantities, besides the mean, such as variance, moments, central moments, skewness, and kurtosis can be effectively computed via stochastic gradient descent. The learning rule is similar in spirit with Oja's rule.

In [3] measurements were taken of the firing of single neurons from the lateral intraparietal area (LIP) during a decision making task, and the spike counts were analyzed using the two parts of the RHS of the law of total variance. The authors argued that with this perspective they were able to expose several neural mechanisms and to help differentiate alternative decision making models. While [3] addresses a complex decoding problem using as a tool the law of total variance, this paper pursues the related encoding problem in the context of artificial neural networks. The doubly stochastic process that generates the data is known, and the main goal is to verify whether the law of total variance is satisfied when the expectations and variances are stochastically computed, instead of using statistical formulas. In this part we also use the family of learning rules of [2].

George M. Georgiou, Kerstin Voigt and Haiyan Qiao are with the School of Computer Science and Engineering, California State University, San Bernardino, CA 92407, USA (email: {georgiou, kvoigt, hqiao}@csusb.edu).

II. Extending Oja's neuron to compute the dominant eigenvalue

Oja's neuron [1] is a simple neuron that computes the principal eigenvector of the correlation matrix $C = \mathrm{E}[XX^T]$ of the inputs $X_i \in \mathbb{R}^n$. The superscript T indicates vector or matrix transpose. The convention that vectors are column vectors is followed. The weight vector $W \in \mathbb{R}^n$ is updated via Oja's rule:

$$W(n+1) = W(n) + \alpha\, y(n)(X(n) - y(n)W(n)), \qquad (1)$$

where α is the learning rate, a small positive constant, and $y = W^T X$ is the linear output of the neuron. By suitably decreasing α during the course of the algorithm, weight vector $|W| \to 1$, i.e. W is normalized, and W converges to the direction of the dominant eigenvector of C. This W as a consequence maximizes the variance $\mathrm{E}[y^2]$. [4], [5] Although the term *variance* is used, more appropriately it should be *second moment* since there is no subtraction of the expected value (the mean) from y. The maximization of $\mathrm{E}[y^2]$ can be seen from the following. At equilibrium, i.e. when W is not changing,

$$\mathrm{E}[y^2] = \mathrm{E}[W^T XX^T W] = W^T CW. \qquad (2)$$

The dominant eigenvector W is normalized and it maximizes $W^T CW$ for all $\|W\| = 1$, a well-known fact, and hence $\mathrm{E}[y^2]$ is maximized as well. Furthermore, at that state the dominant eigenvalue of C is,

$$\lambda_{\max} = \mathrm{E}[y^2]. \qquad (3)$$

The problem addressed here is how to extend Oja's neuron to compute the dominant eigenvalue alongside the dominant eigenvector. A weight λ, an adjustable variable, is added at the output of the Oja's neuron, which will be trained to converge to $\lambda_{\max}$ (Figure 1). The method used is stochastic gradient descent from [2]. The error function R of a sample of N inputs is

$$R = \frac{1}{2}\sum_{j=1}^{N}(y_j^2 - \lambda)^2, \qquad (4)$$

The error R is minimized when its partial derivative with respect to λ is zero, and at that time λ equals the sample second moment:

$$\lambda = \frac{1}{N}\sum_{j=1}^{N} y_j^2. \qquad (5)$$

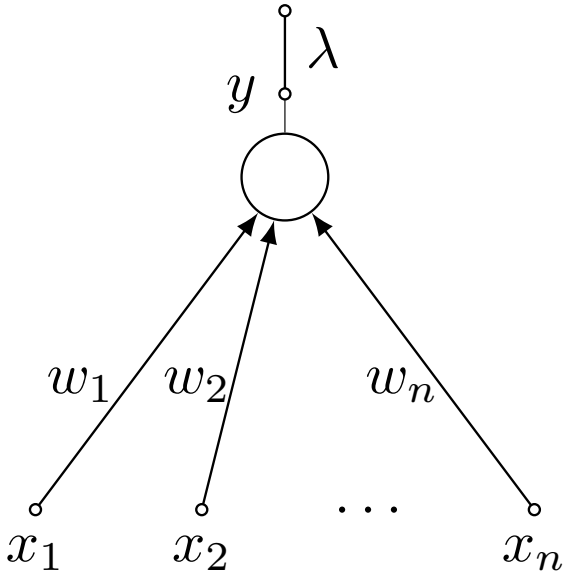

Fig. 1. The augmented Oja's neuron: the weight λ is added at the output which converges to the dominant eigenvalue $\lambda_{\max}$. The linear output of the neuron is y.

Weight λ can be computed by performing instantaneous, i.e. stochastic, gradient descent on R, using n as the time step:

$$\lambda(n+1) = \lambda(n) + \alpha(y^2(n) - \lambda(n)). \tag{6}$$

Using a small or decreasing α, $\lambda \to \lambda_{\max}$, as desired. The update of (6) is performed in parallel with (1). Hence, the dominant eigenvector and eigenvalue are computed simultaneously.

Oja's neuron has been extended to networks of multiple neurons, e.g. [6], [7]. Their weights converge to the other less significant principal eigenvectors. The present method of eigenvalue computation can be extended naturally to these models to compute the corresponding eigenvalues.

III. Results for Extended Oja's Neuron

To find the dominant eigenpairs of the correlation matrix C of the inputs, Oja's rule (Equation (1)) was used to compute the dominant eigenvector and the rule in Equation (6) to compute the dominant eigenvalue. The two rules were updated in parallel in the same loop.

The data set consisted of 100 vectors $X \in \mathbb{R}^4$, where each of their components was drawn from a uniform distribution from the interval $[-0.6, 0.4]$. The error for the dominant eigenvalue is defined as $e_\lambda = |\lambda - \lambda_{\max}|$, the absolute value of the difference of λ and $\lambda_{\max}$, the dominant eigenvalue of correlation matrix C. The error for the dominant eigenvector is defined as $e_W = \|W - W_1\|$, the Euclidean norm of the difference of W and the dominant eigenvector W_1. A constant learning rate $\alpha = 0.01$ was used. The eigenpairs consistently converged close to their correct values. Figures 2 and 3 show two sample runs. The two errors vs epochs are shown. It was observed that consistently λ converged faster to the correct

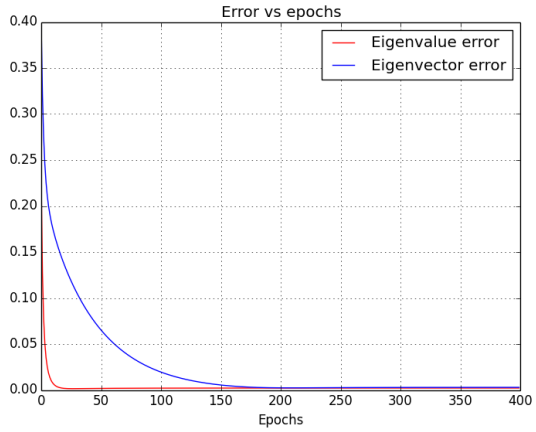

Fig. 2. Run 1: The error from the actual dominant eigenpair.

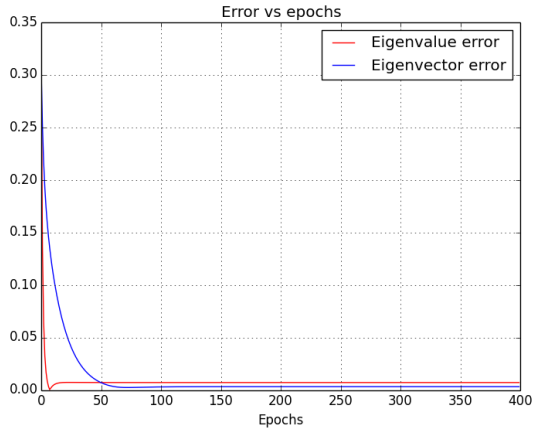

Fig. 3. Run 2: The error from the actual dominant eigenpair.

value than the average of each of the components of W did. Using the same parameters and random distribution as before,

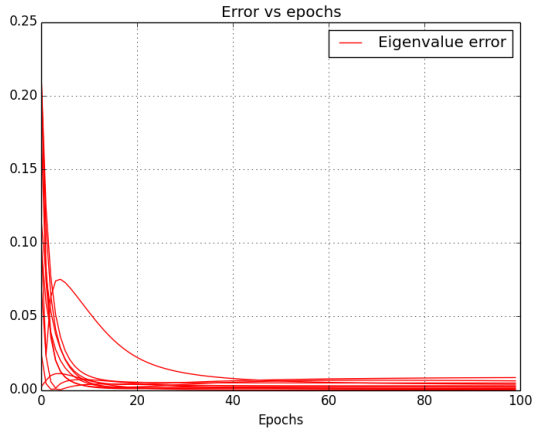

Fig. 4. The error e_λ from the actual dominant eigenvalue.

10 sets of 100 random input vectors were generated. The

dominant eigenvalue of the correlation matrix was calculated in each case, and the error e_λ as it converged was plotted in Figure 4. Within 50 epochs, in all cases the error became less than 0.01.

IV. Simulating the Law of Total Variance

In statistics the law of total variance, also known as Eve's law, for r.v. Y conditioned on r.v. X is expressed with this equation:

$$\mathrm{Var}[Y] = \mathrm{E}[\mathrm{Var}[Y|X]] + \mathrm{Var}[\mathrm{E}[Y|X]]. \tag{7}$$

Both $\mathrm{Var}[Y|X]$ and $\mathrm{E}[Y|X]$ are random variables themselves since they depend on X. When the random variables are discrete,

$$\mathrm{E}[\mathrm{Var}[Y|X]] = \sum_i \mathrm{Var}[Y|X=i]\, p(X=i) \tag{8}$$

$$\mathrm{Var}[\mathrm{E}[Y|X]] = \sum_i (\mathrm{E}[Y|X=i] - \mu)^2\, p(X=i), \tag{9}$$

where $p(X=i)$ indicates probability and $\mu = \mathrm{E}[\mathrm{E}[Y|X]] = \mathrm{E}[Y]$.

The total variance $\mathrm{Var}[Y]$ consists of two parts: the expectation of the variances and the variance of the expectations. The former is often referred to as the *unexplained* variance and the latter as the *explained* variance. To visualize the distinction, when X is a discrete variable, the variance in the *unexplained* part is understood to be the within the group variation, where groups are defined by each of the values of X, and the variance in the *explained* part is the variation between the groups. [8]

We simulate an example of a doubly stochastic process with dependent r.v. Y and independent r.v. X. The two random variables have discrete joint probability distributions shown in Table I. Variable X takes three values from $\{0, 1, 2\}$

TABLE I
Joint probabilities of X and Y.

	Y=1	Y=2	Row sum
X=0	0.1	0.2	0.3
X=1	0.1	0.3	0.4
X=2	0.2	0.1	0.3
Sum	0.4	0.6	1.0

with corresponding probabilities found under the "Row sum" column. The values of X are merely labels and do not enter any of the computations. Variable Y is taking values from $\{1, 2\}$. Using the probabilities in the "Sum" row of the table, the expected value and variance of Y are readily calculated: $\mathrm{E}[Y] = 1.6$ and $\mathrm{Var}[Y] = 0.24$.

The law of total variance for this example is represented as a network in Figure 5. A sequence of values $y_i = (Y|X=i)$ is generated, one at a time, according to the probabilities in Table I, and each value is propagated through the network. The E[.] and Var[.] boxes in the path of an $Y|X=i$ value are updated: Each E[.] box has an adjustable variable m_i that converges to the mean of its inputs y_i, and is updated according the learning rule for the mean:

$$m_i(n+1) = m_i(n) + \alpha_i(y_i(n) - m_i(n)), \tag{10}$$

where time step n is incremented only when there is new input y_i to box E[.]. The small positive value α_i is the learning rate. The output of each E[.] box is $m_i(n+1)$, the approximation of the mean, and it feeds into the next stage and sometimes into a Var[.] box. Each Var[.] box in the first layer has an adjustable weight v_i that converges to the variance of its inputs y_i. It is updated according the learning rule for the variance:

$$v_i(n+1) = v_i(n) + \alpha_i((y_i(n) - m_i(n))^2 - v_i(n)), \tag{11}$$

where time step n is incremented only when there is new input y_i to box Var[.]. The update rule uses the mean value m_i. The variance approximation of the output of each box Var[.] is $v_i(n+1)$ and it feeds into the next stage.

The stage before the summation box corresponds to the two terms of the RHS of Equation (7). The Var[.] box maintains and updates the variance adjustable variable v. In addition, it maintains and updates variable m for the mean of its inputs, which is needed in the update learning rule of v. The learning rules for m and v are those in Equations (10) and (11), respectively. Note that m computes μ in Equation (9) and v computes the LHS of the same equation. The E[.] box maintains and updates a variable μ_E for the mean of its inputs. The learning rule used for μ_E is that of Equation (10), and it computes the LHS of Equation (8).

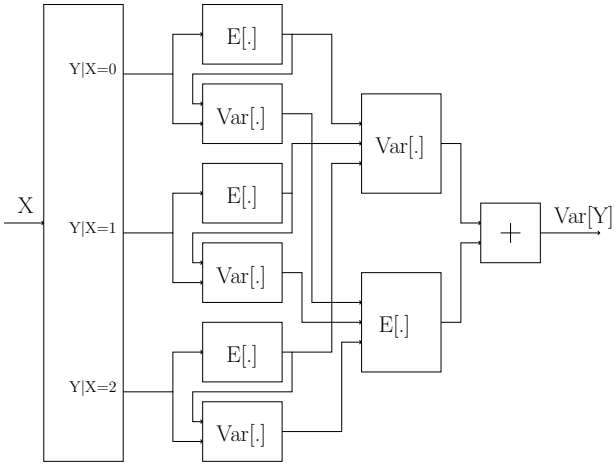

Fig. 5. Simulating the law of total variance.

V. Results of the simulation of the law of total variance

The results of a typical run (Run 1) are shown in Figures 6, 7, and 8. They show each of the variables, mean or variance, that is updated in the boxes of Figure 5. In total there are five means, the m_i's, and four variances, the v_i's, that are shown in the figures. All of them were initialized to 0, as it can be seen in the figures. Initialization from other values in $[0, 1]$ did not affect convergence to the steady state of their

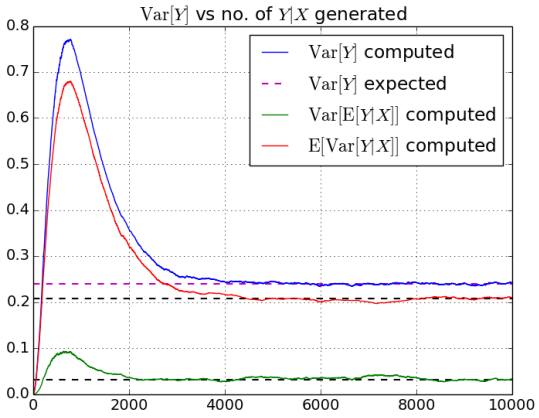

Fig. 6. Run 1: Stochastic computation of $\mathrm{Var}[Y]$ using the law of total variance.

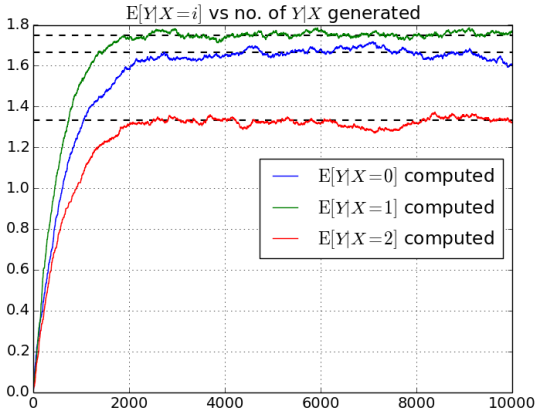

Fig. 7. Run 1: Stochastic computation of $\mathrm{E}[Y|X=i], i=0,1,2$.

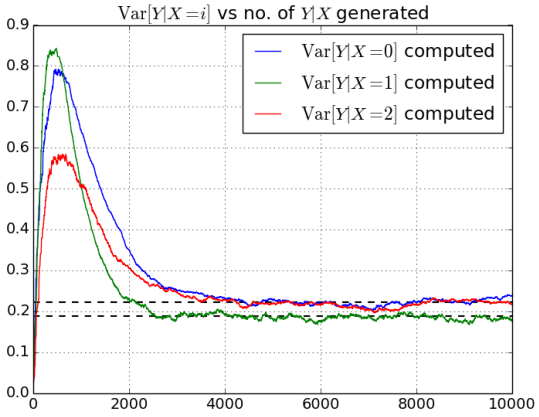

Fig. 8. Run 1: Stochastic computation of $\mathrm{Var}[Y|X=i], i=0,1,2$.

values. The calculated value of each quantity is represented by the corresponding horizontal line. Although the learning rate α_i could be individualized to each quantity, the same value of 0.005 was used for all, and it was kept constant. This insensitivity to the learning rate is a testament to the robustness of the stochastic learning rules for the means and variances. Experiments with greater and smaller values of α_i showed expected behavior: larger values resulted in faster convergence but in more jagged curves, smaller values resulted in slower convergence but smoother curves. All values eventually converge to their corresponding calculated values, and they remain there slightly oscillating. Oscillation can be eliminated or minimized by reducing the learning rate.

Figure 6 shows $\mathrm{Var}[Y]$, the final output in Figure 5, i.e. the LHS of Equation (7). It is smoother than all other curves which shows that the cumulative effect of combining the other quantities results in canceling out of errors. The accuracy of $\mathrm{Var}[Y]$ at the output, as can be seen from the graph, is remarkable, given that nine inherently inexact learning rules contribute to its value. In addition, each generated value of $Y|X=i$ is presented only once to the network, as opposed to forming a fixed dataset and repeatedly presenting it in epochs.

VI. Conclusion

Oja's neuron, which computes the dominant eigenvector, was augmented to compute the corresponding dominant eigenvalue of the correlation matrix of inputs using a simple stochastic gradient descent rule. The eigenvalue is the second moment of the projected inputs in the direction of the dominant eigenvector, which conveys important variance information. Experiments have shown the dominant eigenvalue is computed just as efficiently as, and in parallel with, the dominant eigenvector. The effectiveness of the stochastic gradient descent techniques in [2] was further demonstrated by applying them multiple times in an experiment that verified the law of total variance. Means and variances were accurately propagated in the network, and the correct final value of the variance was obtained. Figures were presented to show the convergence behavior of intermediate computations of means and variances.

References

[1] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biology*, vol. 15, pp. 267–273, 1982.

[2] G. M. Georgiou and K. Voigt, "Stochastic computation of moments, mean, variance, skewness, and kurtosis," *Electronics Letters*, vol. 51, no. 9, pp. 673 – 674, 30 April 2015.

[3] A. K. Churchland, R. Kiani, R. Chaudhuri, X.-J. Wang, A. Pouget, and M. N. Shadlen, "Variance as a signature of neural computations during decision making," *Neuron*, vol. 69, no. 4, pp. 818–831, 2011.

[4] J. Hertz, A. Krogh, and R. Palmer, *Introduction to the Theory of Neural Computation*, ser. Advanced book program. Addison-Wesley Publishing Company, 1991.

[5] S. O. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Prentice Hall, 11 2008.

[6] T. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, pp. 459–473, 1989.

[7] E. Oja, "Neural networks, principal components, and subspaces," *International journal of neural systems*, vol. 1, no. 01, pp. 61–68, 1989.

[8] J. Blitzstein and J. Hwang, *Introduction to Probability*, ser. Chapman & Hall/CRC Texts in Statistical Science. CRC Press, 2015.