Abstract- Eigendecompositions play a very important role in a variety of signal processing applications. In this paper, we derive and study an algorithm for Generalized Eigendecomposition which is both on-line and fast converging. A rule to extract the maximum eigencomponent is first presented, and then on-line deflation is applied to estimate the minor components. Proof of convergence has been established using stochastic approximation theory.

1.0 INTRODUCTION

Principal component analysis (PCA) and Generalized eigendecomposition (GED) are widely used statistical techniques in various signal-processing applications like feature extraction, signal estimation and also detection [1], [2], [3]. Linear discriminant analysis is also a generalized eigenvalue problem [9]. There are several analytical techniques for solving the generalized eigenvalue problem [4]. These analytical techniques are computationally intensive and are not feasible for real time applications. Hence, for many applications, adaptive on-line solutions are desired. Many on-line algorithms have been developed for PCA which is a special case of generalized eigendecomposition [2], [3], [5], [17], [18]. However, there are very few adaptive techniques for generalized eigendecomposition, which are based on gradient formulation. Mao and Jain present a two-layer PCA network for solving GED [3]. The individual PCA layers are trained using the Rubner-Tavan model [6], [7]. The procedure is slow especially when the input dimensionality is very high and when multiple eigencomponents need to be extracted. A similar algorithm is presented in [13] using faster PCA algorithms. Xu et.al., demonstrate an elegant on-line local rule for GED [10]. Although the formulation is novel, strong convergence of the algorithm has not been established. Chatterjee et.al propose a learning algorithm to solve the LDA problem [9]. The main problem with the gradient-based approach is slow convergence and consequently the performance of these algorithms is directly tied to the selection of the right step-sizes. [13] tries to circumvent this problem but with added computational complexity. The goal of this paper is to present a new algorithm for GED which is faster and more robust than the gradient methods. The algorithm is similar to the RLS algorithm for Wiener filtering [11]. First we present the rule to extract the principal generalized eigenvector and then use standard deflation procedure to estimate the minor components. We will also provide the proof of convergence for
the principal generalized eigenvector. The convergence of the subsequent components will naturally follow. Various simulation tests are conducted and the results are compared with the state of the art algorithms.

2.0 GENERALIZED EIGENDECOMPOSITION (GED)

Mathematically speaking, generalized eigendecomposition or GED boils down to solving the equation $RW = R_ZWA$, where $R, R_Z$ are real square matrices [4]. From the signal processing perspective, these are the full covariance matrices of zero-mean stationary random signals; $W$ is the generalized eigenvector matrix and $A$ is the diagonal generalized eigenvalue matrix. Let $R = E\{x_i(n)x_i^T(n)\}$ and $R_Z = E\{x_i(n)x_i^T_Z(n)\}$ be the covariance matrices of two zero-mean stationary random signals $x_i(n)$ and $x_i^T(n)$ respectively. Since the matrices are symmetric and positive-definite, all the generalized eigenvalues are strictly positive. For any generalized eigenvector $w$, we can obtain the corresponding generalized eigenvalue as,

$$J(w) = \frac{w^T R w}{w^T R_Z w}$$  \hspace{1cm} (1)

Equation (1) can be used as a cost function and maximized under certain constraints to estimate the generalized eigenvalues. All the generalized eigenvalues are stationary points of (1). So, from the generalized eigenvalue equation we can write,

$$R w = \frac{w^T R w}{w^T R_Z w} R_Z w$$  \hspace{1cm} (2)

If $R = I$, then (2) reduces to the Rayleigh quotient and generalized eigenvalue problem will degenerate to PCA. Left multiplying (2) by $R^{-1}$ and rearranging, we get

$$w = \frac{w^T R w}{w^T R_Z w} R^{-1}_Z R w$$  \hspace{1cm} (3)

Equation (3) is the basis of our iterative algorithm. Let the weight vector at iteration $(n-1)$, $w(n-1)$ be the estimate of the principal generalized eigenvector. Then, the estimate of the new weight vector at iteration $n$ according to (3) is,

$$w(n) = \frac{w^T(n-1) R_Z(n) w(n-1)}{w^T(n-1) R_Z(n) w(n-1)} R^{-1}_Z(n) R_Z(n) w(n-1)$$  \hspace{1cm} (4)

It is easy to see that (4) tracks the generalized eigenvalue equation at every time step just like the RLS update rule that tracks the Wiener solution at every time step [11]. By using Sherman-Morrison-Woodbury matrix inversion lemma [4] and making further simplifications, we get,
The summations in (5) can be easily implemented using simple recursive estimators. The architecture to compute the first generalized eigenvector is a single-output neural network with weights \( w \). If \( y_1 \) and \( y_2 \) are the outputs of this network for inputs \( x_1 \) and \( x_2 \), then,

\[
 w'(n-1)R_x(n)w(n-1) = \sum_{i=1}^{n} y_1'(i), \quad \text{and} \quad w'(n-1)R_y(n)w(n-1) = \sum_{i=1}^{n} y_2'(i).
\]

Hence all the terms in (4) can be computed using the outputs of this network. The overall complexity of the algorithm is \( O(N^2) \). It is very interesting to note that the algorithm in (5) does not require a step-size. An alternative way of implementing (5) would be to compute the inverse using PCA. It is easy to show that \( R_x^{-1} = QA^{-1}Q^T \), where \( Q \) is the eigenvector matrix of \( R_x \) and \( A \) is the diagonal eigenvalue matrix. Thus (5) can be implemented using a two-layer PCA network [13].

**Rule for minor components**

Equation (5) gives us the first generalized eigenvector. For the minor components, we can use the standard deflation procedure. Consider the following pair of matrices, \( \hat{R}_1 = \left[ I - R_x w_i w_i^T \right] R_x \), \( \hat{R}_2 = R_x \), where \( w_i \) is the best estimate of the first generalized eigenvector using (5). For this pair of matrices, \( \hat{R}_1 w_i = 0 \) and \( \hat{R}_i w_i = \lambda_i R_x w_i, \ i > 1 \) [8]. The time index \( n \) is implicit and is omitted for convenience.

\[
 \hat{R} = \left[ I - R_x w_i w_i^T \right] R_x, \quad \hat{R} = R_x - R_x w_i w_i^T R_x / w_i^T R_x w_i = R_x - 2 R_x w_i w_i^T R_x / w_i^T R_x w_i + R_x w_i w_i^T R_x / w_i^T R_x w_i
\]

\[
 \hat{R} = E \left[ x_i - \frac{R_x w_i}{w_i^T R_x w_i} y_i \right], \quad y_i = w_i^T x_i(n)
\]

Using the above relation for doing on-line deflation, the update rule for the second generalized eigenvector is given by (5) with \( x_i(n) \) replaced by \( \hat{x}_i(n) \) and \( x_2(n) \)
remains the same. Note that the deflation given by (6) does not increase the complexity of the algorithm because all the terms in (6) are pre-computed in (5). Other minor components can be obtained in the same manner.

3.0 EXPERIMENTAL RESULTS

Simulation I
Consider two time series with 2000 samples, each embedded in a 3-D space. The numerical method in MATLAB gives the generalized eigenvalues as [9.8576, 0.8369, 0.3829]. Figure 1 shows the convergence plots using the rules in (5) and (6). After 900 on-line iterations, the estimates of the generalized eigenvalues using our update equations are [9.8576, 0.8350, 0.3843]. These values match very well with the numerical values given by MATLAB.

Figure 1- Convergence plots for simulation I.
Simulation II

In this simulation, we will investigate the performance of the algorithm on higher dimensional data with large generalized eigenspread. Two filtered random noise sequences of length 1000 samples each are generated and they are embedded in a 15-D space. We will estimate the first five generalized eigencomponents. Figure 2 shows the plots of convergence for our update rules. The algorithm converges in less than 5000 iterations and the increased generalized eigenspread doesn’t seem to have a negative effect on the convergence of the algorithm. This is because (5) truly tracks the generalized eigenvalue equation at every time step. The same data set was tested with the algorithm presented in [9] and the two-layer PCA model approach [3], [13]. The two-layer PCA model given in [3] performs poorly for higher data dimensions. [13] performs better than [3] and gives good results but requires more flops as it adopts the power method to train the PCA model. The algorithm presented in [9] gives good results as shown in table I but requires sufficient tweaking of the step-size (for this experiment, we found that an initial step-size of 0.01 gives good results). Moreover, the rule in [9] uses matrix multiplications. The results tabulated for the algorithm in [9] are taken after 10000 on-line iterations. The convergence of the algorithm in [9] is dependent on choosing the right step-sizes for the data, the selection of which is not trivial. This is true for most of the adaptive gradient procedures for generalized eigendecomposition. The main advantage of our algorithm is that it does not require a step-size and this guarantees convergence in all cases.

Table I – Comparison between the proposed rule and the algorithm in [9]

<table>
<thead>
<tr>
<th>Iterations required for the proposed method</th>
<th>Numerical Values</th>
<th>Proposed update rule</th>
<th>Algorithm in [9]</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_1 ) 1000</td>
<td>223.4200</td>
<td>223.4200</td>
<td>223.3540</td>
</tr>
<tr>
<td>( \lambda_2 ) 1000</td>
<td>14.3440</td>
<td>14.3440</td>
<td>14.3361</td>
</tr>
<tr>
<td>( \lambda_3 ) 2500</td>
<td>5.4511</td>
<td>5.4581</td>
<td>5.4714</td>
</tr>
<tr>
<td>( \lambda_4 ) 3500</td>
<td>2.8696</td>
<td>2.8663</td>
<td>2.8881</td>
</tr>
<tr>
<td>( \lambda_5 ) 5000</td>
<td>1.5065</td>
<td>1.5154</td>
<td>1.4826</td>
</tr>
</tbody>
</table>

4.0 CONCLUSIONS

We derived a new algorithm for on-line generalized eigendecomposition (GED) from first principles. The algorithm is robust; convergence is fast and guaranteed and does not involve a step-size. The rule extracts the first generalized eigencomponent and the minor components are estimated using an on-line deflation procedure. Proof of convergence is established by stochastic approximation theory as well as by simulations. We compared the proposed algorithm with the algorithms in [9], [3] and [13] and found superior performance over other methods. Detailed analysis of convergence with useful applications and comparison with gradient methods will be provided in a later paper.

Acknowledgements - This work was partially supported by the National Science Foundation under Grant NSF ECS-9900394.
Figure 2 – Convergence plots for simulation II
APPENDIX – CONVERGENCE ANALYSIS

In order to prove the asymptotic convergence of (4), we will resort to the stochastic approximation theory developed by Ljung [14] and by Kushner and Clark [15]. The idea is to associate the discrete-time adaptation rule to an ordinary differential equation. The convergence of the discrete-time algorithm is strongly or weakly tied to the stability of the ODE. Equation (4) is a special case of the generic stochastic approximation algorithm

\[ w(n + 1) = w(n) + \eta(n)h(w(n), x(n)) \]

The following assumptions are made to apply the theory developed by Ljung [14].

A.1 The inputs \( x_1(n), x_2(n) \) are at least wide sense stationary (WSS) with positive definite autocorrelation matrices \( R_1, R_2 \).

A.2 The sequence of weight vectors \( w(n) \) is bounded with probability 1.

A.3 The update function \( h(w(n), x_1(n), x_2(n)) \) is continuously differentiable with respect to \( w, x_1, x_2 \) and its derivatives are bounded in time.

A.4 Even if \( h(w(n), x_1(n), x_2(n)) \) has some discontinuities a mean vector field \( \bar{h}(w, x_1, x_2) = \lim_{n \to \infty} E[h(w(n), x_1(n), x_2(n))] \) exists and is regular.

A.5 There is a locally stable solution in the Lyapunov sense to the ODE. In other words, the ODE has an attractor \( w^* \), whose domain of attraction is \( D(w^*) \).

A.6 The weight vector \( w(n) \) enters a compact subset \( M \) of the basin of attraction \( D(w^*) \) infinitely often, with probability 1.

The update equation in (4) has a constant gain vector i.e., we have \( \eta(n) = 1 \) for all values of \( n \). The extension of the ODE technique to constant gain algorithms has been developed by Benveniste et.al [19]. Accordingly, the linking between the update equations to an ODE remains intact.

**Theorem I** Consider the algorithm in (4) and let the assumptions A.1, A.3 and A.4 hold. Then, \( w \to cq_1 \), where \( q_1 \) is the generalized eigenvector associated with the largest generalized eigenvalue as \( n \to \infty \) and \( c \) is an arbitrary constant.

**Proof**

The corresponding ODE for the equation in (4) is,

\[ \bar{h}(w(t)) = \left( \frac{w^T(t)R_2w(t)}{w^T(t)R_2w(t)} \right) R_2^{-1}R_1w(t) - w(t) = \frac{dw(t)}{dt} \]  

(7)

We want to find the stationary points of this differential equation. Let \( w(t) \) be expanded in terms of the complete set of generalized eigenvectors of \((R_2, R_1)\) as

\[ w(t) = \sum_{k=1}^{\infty} \theta_k(t)q_k \]  

(8)

Simplifying this further using the simultaneous diagonalization property we get,
Let us analyze the dynamics of the non-linear differential equation in (9) separately. Ideally, we want the time varying projections corresponding to the modes associated with all eigenvectors except the first one to decay to zero asymptotically.

**Case I:** $1 < k \leq m$

Define $\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)}$. Therefore, $\frac{d \alpha_k(t)}{dt} = \frac{1}{\theta_1(t)} \frac{d \theta_k(t)}{dt} - \frac{\theta_k(t)}{\theta_1(t)} \frac{d \theta_1(t)}{dt}$ and

$$\frac{d \alpha_k(t)}{dt} = -\alpha_k(t) \left[ \sum_{l=1}^{n} \theta_l(t) \right] \left( \lambda_1 - \lambda_k \right)$$

Equation (10) can be written as

$$\frac{d \alpha_k(t)}{dt} = -f(t)\alpha_k(t)(\lambda_1 - \lambda_k), \text{ where } f(t) = \sum_{l=1}^{n} \theta_l(t) \lambda_l$$

Note that $f(t) > 0$ for all $t$. Therefore, it can be easily shown using Lyapunov stability theorems that, with

$$\lambda_1 > \lambda_2 > \lambda_3 > ... > \lambda_m > 0, \text{ as } t \to \infty, \text{ } k > 1$$

$$w(t) = \pm cq_k$$

where, $c$ is an arbitrary constant.

Thus (12) clearly states that the stable stationary point of the system described by (4) is a scaled version of the maximum generalized eigenvector.

**Theorem II**

Let the assumption A.2 hold. The weight vector $w(n)$ enters a compact subset $M$ of the basin of attraction $D(w^*)$ infinitely often, with probability 1.

**Proof**

Let $M$ be a compact subset defined by the set of vectors with norm less than or equal to a finite constant. We satisfy A.2 by normalizing the weight vector. Hence, as the number of iterations increases, $w(n)$ will be within the set $M$ and will remain inside with probability 1. Also, the domain of attraction $D(w^* = cq_k)$ includes all vectors with bounded norm. Thus, $M \in D(cq_k)$. 

270
From theorems I and II, A.5 and A.6 are satisfied. Thus, from the stochastic approximation theory of algorithms with constant gains [19], the rule in (4) converges asymptotically to a stationary random variable whose mean value corresponds to the stable stationary point of the associated ODE, which is nothing but the generalized eigenvector corresponding to the maximum generalized eigenvalue.

REFERENCES


