A FAST, ON-LINE ALGORITHM FOR PCA AND ITS CONVERGENCE CHARACTERISTICS

Yadunandana N. Rao and Jose C. Principe
Computational NeuroEngineering Laboratory
EB, #33, PO Box 116130
University of Florida
Gainesville, FL 32611-6130
E-mail: {yadu, principe}@cnel.ufl.edu

Abstract- Eigendecompositions play a very important role in a variety of signal processing applications. In this paper, we derive and study an algorithm for principal component analysis (PCA) which is both on-line and fast converging and which has been presented earlier as a heuristic alternative to the power method. A rule to extract the maximum eigencomponent is first presented, and then on-line deflation is applied to estimate the minor components. The algorithm is compared with the traditional Sanger’s rule through simulations. The convergence properties of the algorithm are explored thoroughly and we present a complete proof explaining the behavior of the algorithm.

1.0 INTRODUCTION

Principal component analysis (PCA) is a widely used statistical technique in various signal-processing applications like feature extraction, signal estimation and also detection [1], [2], [3]. There are several analytical techniques for solving the eigenvalue problem or PCA [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 algorithms have been developed for on-line PCA such as Sanger’s rule or the generalized Hebbian algorithm (GHA) [5], the Rubner-Tavan model [6], [7], and the APEX model which is a variation of the Rubner-Tavan model [2]. Most of these algorithms are based on gradient search methods and have convergence problems when multiple components need to be estimated. The power method, a subspace analysis technique has been used to estimate the principal eigencomponents [4], [13], [14]. Although the convergence characteristics of these methods are excellent, the update rules are non-local and computationally intensive. [15] proposes an on-line algorithm for PCA as a special case of subspace analysis but the formulation is again based on gradient search. [14] gives a rule for estimating the principal subspace, which can be used to find the principal eigencomponent only. We also present an algorithm for estimating the principal eigencomponent based on the Rayleigh-Ritz theorem and end up with the same update equation as in [14].
Moreover, [14] does not present any further details regarding the behavior of the algorithm and its convergence issues, which are the main goals of the paper.

2.0 PRINCIPAL COMPONENT ANALYSIS (PCA)

Mathematically speaking, eigendecomposition or PCA boils down to solving the equation $RW = \Lambda W$, where $R$ is any real square matrix [4]. From the signal processing perspective $R$ is the full covariance matrix of a zero-mean stationary random signal, $W$ is the eigenvector matrix and $\Lambda$ is the diagonal eigenvalue matrix. We now present the algorithm to solve the PCA problem on-line. Assume a zero-mean stationary signal $x(n)$, with a covariance matrix $R$. From the Rayleigh-Ritz theorem [11], [8] the maximum eigenvalue is a stationary point of the Rayleigh quotient

$$r(w) = \frac{w^T R w}{w^T w}$$

where, $w$ is the first principal eigenvector. Indeed, $w$ is a stationary point if and only if

$$\frac{\partial r(w)}{\partial w} \propto R w \quad w^T w = 0 \Rightarrow R w = \frac{w^T R w}{w^T w} w = r(w) w$$

Hence, the eigenvalue equation can be written as $R w = \frac{w^T R w}{w^T w} w$. Assuming $w^T w = 1$, which is true for any eigenvector, we can write $R w = (w^T R w) w$. Note that the quantity $(w^T R w)$ is a scalar. Hence we can express $w$ as

$$w = \frac{R w}{w^T R w}$$

Equation (2) is the basis of the iterative algorithm. It basically states that there is a relationship between $w$ and its rotated version by $R$. Let the weight vector at iteration $(n-1)$, $w(n-1)$ be the estimate of the maximum eigenvector. Then, the estimate of the new weight vector at iteration $n$ according to (2) is,

$$w(n) = \frac{R(n) w(n-1)}{w^T (n-1) R(n) w(n-1)}$$

$R(n)$ is the estimate of the covariance matrix at the $n^{th}$ time step. Equation (3) is the same as equation (27) in [14] except that a matrix $W$ replaces vector $w$. Note that the update rule for $w(n)$ tracks the eigenvalue equation assuming $w^T (n) w(n) = 1$ at every time step. The denominator in (2) is nothing but an estimate of the maximum eigenvalue. It is interesting to compare this technique with the RLS algorithm [11] for optimal filtering. The RLS weight update computes the Wiener solution at every time step using the previous weights to compute the apriori error. Here, we use the previous weight vector and apply a rotation to it. The expression in (3) can be further simplified to
where, $y_\alpha(k) = w^\alpha(k-1) x(k) = x^\alpha(k) w(k-1)$. Note that the output $y_\alpha(k)$ is computed using the previous weights. The update rule in (4) for extracting the first principal component is local, on-line and linear in complexity, $O(N)$ with respect to the weights. The summations in (4) increase with iterations, but practically we can replace the summations with recursive estimators. It is interesting to compare Oja’s rule [12] with (4). Oja’s update rule is

$$
\dot{w}(n) = \dot{w}(n-1) + \eta [y(n) x(n) - y^T(n) w(n-1)]
$$

where $\eta$ is the step-size. It is well known that Oja’s rule gives the principal component [12]. The major difference lies in the fact that (4) does not have a step-size. The normalization is also intuitively satisfying because it is nothing but an estimate of the maximum eigenvalue. This results in faster convergence of the algorithm when compared against the gradient based methods as rightly mentioned in [14]. The power method has been a very effective tool to compute the maximum eigencomponent [4], [14], [13]. Power method weight update rule is

$$
\dot{w}(n) = \frac{R w(n-1)}{\sqrt{w^T(n-1) R w(n-1)}}
$$

(5)

Note that (5) has direct matrix multiplications and requires normalization at every time step. Therefore the power method can be inefficient when the input data dimension is high, as this increases the dimension of $R$. Also note that the constraint $w^T(n) w(n) = 1$ in (4) need not be imposed after every update. The norm starts off initially with random values, but when the weight vector approaches the eigenvector i.e., when $w(n) \to \alpha V$, where $\alpha$ is a scalar, the norm oscillates between $\alpha$ and $1/\alpha$. This observation is in conformance with the argument in [14]. Thus, a normalization is required only at the end of training. Moreover, the toggling in the norm can be used as a stopping criterion for training. A rigorous proof of convergence using stochastic approximation theory is presented in the end.

**Rule for minor components**

The minor eigencomponents can be estimated by the deflation technique, which does a transformation on $R$ giving $\hat{R} = (I - w_i^T R) R$ [4]. Neglecting time index $n$,

$$
\hat{R} = R - w_i^T R - R w_i^T + w_i^T (R w_i) R w_i^T = (R - w_i^T R) (I - w_i w_i^T)
$$

Expanding using expectations, and putting $y_i = w_i^T x = x^T w_i$,

$$
\hat{R} = E\{xx^T\} - E\{w_i x y_i\} - E\{y_i w_i x^T\} + E\{w_i y_i x^T\} = E\{x - w_i y_i\} (x - w_i y_i)^T
$$

or $\hat{R} = E\{\delta x \delta x^T\}$, where $\delta = x - w_i y_i$.  

\begin{equation}
\sum_{k=1}^{n} x(k) y_\alpha(k)
\end{equation}

\begin{equation}
\sum_{k=1}^{n} y_\alpha^T(k)
\end{equation}
Using the above relation for doing on-line deflation, the update rule for the second eigencomponent is given by

\[
    w_i(n) = \frac{\sum_{k=1}^{n} \hat{x}(k)y_{i}(k)}{\sum_{k=1}^{n} y_{i}^2(k)}
\]

(6)

Note that the update rule has the same form as (4), except that \(\hat{x}(k)\) is the signal with the principal component subtracted and \(y_{i}(k)\) is the output computed with the previous weights \(w_i(n-1)\). Other minor components can be obtained in the same manner.

**Simulation**

The algorithm is tested with a violin sound time series, which has an eigenspread of 802 when the input dimensionality is 5. For comparison purposes, we tried Sanger’s rule with a step size of 0.004. The parameters were varied to obtain the best possible results with Sanger’s rule. Figure 1 shows a plot of the time series and the convergence characteristics of the algorithm given by (4) and (6). Figure 2 shows the convergence plots for Sanger’s rule. Table I makes a comparison with the results obtained from Sanger’s rule along with the numerical values.

**TABLE I - COMPARISON WITH NUMERICAL VALUES AND SANGER’S RULE ESTIMATES**

<table>
<thead>
<tr>
<th></th>
<th>Proposed rule and rule in [14]</th>
<th>Sanger’s rule</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterations</td>
<td>10000</td>
<td>25000</td>
<td>-</td>
</tr>
<tr>
<td>(\lambda_1)</td>
<td>0.6423</td>
<td>0.6424</td>
<td>0.6423</td>
</tr>
<tr>
<td>(\lambda_2)</td>
<td>0.1595</td>
<td>0.1595</td>
<td>0.1594</td>
</tr>
<tr>
<td>(\lambda_3)</td>
<td>0.0157</td>
<td>0.0149</td>
<td>0.0155</td>
</tr>
<tr>
<td>(\lambda_4)</td>
<td>0.0051</td>
<td>0.0085</td>
<td>0.0052</td>
</tr>
<tr>
<td>(\lambda_5)</td>
<td>0.0009</td>
<td>0.0078</td>
<td>0.0008</td>
</tr>
</tbody>
</table>

From figure 2 we observe the much faster convergence of the algorithm. From table I we conclude that the accuracy of the results is also better for the minor components than the Sanger’s rule and other similar methods.

**3.0 CONCLUSIONS**

We independently derived an algorithm from first principles for on-line PCA similar to the algorithm given in [14]. Although the update rules look alike the formulations are totally different. The rule extracts the first principal component and the minor components are estimated using an on-line deflation procedure. The algorithm is simple to implement, linear in complexity, fast converging and stable. Proof of convergence is established by stochastic approximation theory as well as
by simulations. Test cases involved data with both low and high eigenspreads. The algorithm was also compared successfully with the traditional Sanger’s rule. **Acknowledgements** - This work was partially supported by the National Science Foundation under Grant NSF ECS-9900394.

---

**APPENDIX**

We now prove that the algorithm in (3) is globally asymptotically stable. The proof uses stochastic approximation tools proposed by Ljung [16] and also by Kushner and Clark [17]. For details, the reader is referred to [16], [17] and [18]. Following the details in [18], we can write the expected value of the weight update as,

\[ \overline{f}(w(t)) = \frac{Rw(t)}{w^T(t)Rw(t)} - w(t) = \frac{dw(t)}{dt} \]  

(7)

We want to find the stationary points of this differential equation. Let \( w(t) \) be expanded in terms of the complete set of orthonormal eigenvectors of the correlation matrix \( R \) as (see [18])

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

(8)

---

**Figure 1** - Convergence plots for violin sound time series

![Plot of violin time series](image1.png)

![Weight loads for the maximum eigenvector](image2.png)

![Estimation of eigenvalues and eigenvectors](image3.png)
$q_k$ is the $k^{th}$ eigenvector of $R$ and the coefficient $\theta_k(t)$ is a time-varying projection of $w(t)$ onto $q_k$. Now, $w^\top(t)Rw(t) = \sum_{i=1}^{m} \lambda_i \theta_i^2(t)$ and $Rw(t) = \sum_{k=1}^{m} \lambda_k q_k \theta_k(t)$.

Therefore, (8) can be written as

$$\sum_{k=1}^{m} \frac{d\theta_k(t)}{dt} q_k = \sum_{k=1}^{m} \lambda_k q_k \theta_k(t) - \sum_{k=1}^{m} \theta_k(t)q_k.$$  

(9)

Simplifying further, we get,

$$\frac{d\theta_k(t)}{dt} = \frac{\lambda_k \theta_k(t)}{\sum_{i=1}^{m} \lambda_i \theta_i^2(t)} - \theta_k(t).$$

(10)

Let us analyze the dynamics of the non-linear differential equation in (10) 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 1:** $1 < k \leq m$

Define $\alpha_k(t) = \frac{\theta_k(t)}{\theta_1(t)}$. Differentiating both sides with respect to $t$, we get
\[
\frac{d\alpha_i(t)}{dt} = \frac{1}{\theta_i(t)} \frac{d\theta_i(t)}{dt} - \alpha_i(t) \frac{d\theta_i(t)}{dt}
\]

Applying (10) and using the definition of \(\alpha_i(t)\), we get,

\[
\frac{d\alpha_i(t)}{dt} = \sum_{j=1}^{m} \lambda_j \alpha_j(t) - \sum_{j=1}^{m} \lambda_j \alpha_j(t) = -\alpha_i(t) \left[ \sum_{j=1}^{m} \lambda_j \theta_j^2(t) \right]
\]

Equation (12) can be written as

\[
\frac{d\alpha_i(t)}{dt} = -f(t)\alpha_i(t)(\dot{\lambda}_i - \ddot{\lambda}_i), \text{ where } f(t) = \frac{1}{\sum_{j=1}^{m} \lambda_j \theta_j^2(t)}
\]

Note that \(f(t) > 0\) for all \(t\). Therefore, it can be easily shown using Lyaponov stability theorems that, with \(\dot{\lambda}_i > \dot{\lambda}_2 > \dot{\lambda}_1 > \ldots > \dot{\lambda}_m > 0\), \(\alpha_i(t) \to 0\) as \(t \to \infty\) for \(k > 1\)

This is the global asymptotic value for \(\alpha_i(t)\) for \(1 < k \leq m\).

**Case II**: \(k = 1\)

\[
\frac{d\theta_1(t)}{dt} = \frac{\lambda_1 \theta_1(t)}{\sum_{j=1}^{m} \lambda_j \theta_j^2(t)} - \theta_1(t)
\]

Using (14), (15) can be written as,

\[
\frac{d\theta_1(t)}{dt} = \frac{\lambda_1 \theta_1(t)}{\left( \lambda_1 \theta_1^2(t) + \sum_{j=1}^{m} \lambda_j \theta_j^2(t) \right)} - \theta_1(t) = \frac{\left(1 - \theta_1^2(t)\right)}{\theta_1(t)}
\]

We have to analyze this differential equation. As a test of stability, choose a Lyaponov function \(V(t)\) as,

\[
V(t) = \left[\theta_1^2(t) - 1\right]
\]

- \(V(t) > 0\) for all \(t\)
- \(\frac{dV(t)}{dt} = 4\theta_1^2(t)\left[\theta_1(t) - 1\right] \frac{d\theta_1(t)}{dt} = 4\left[\theta_1^2(t) - 1\right] - \theta_1(t)] < 0\)

Hence (16) is stable and has a minimum given by

\[
\frac{dV(t)}{dt} = 0 \Rightarrow \theta_1(t) = \pm 1
\]

Therefore, as \(t \to \infty\)

\[
\theta(t) = \pm \eta_i
\]

Thus (19) clearly states that the stable stationary point of the system described by (7) is the maximum eigenvector. It is easy to solve the non-linear differential equation in (16). Solving it gives,
\( \theta_i(t) = \pm e^{-t} \sqrt{e^{2t} + K} \)  \hspace{1cm} (20)

where, \( K \) is an arbitrary constant. \( \frac{d\theta_i(t)}{dt} \to 0 \) as \( t \to \infty \). Therefore, the minimum value of \( \theta_i(t) \) is given by

\[ \theta_i(t) \big|_{t=\infty} = \pm \lim_{t \to \infty} e^{-t} \sqrt{e^{2t} + K} = \pm \lim_{t \to \infty} \sqrt{1 + Ke^{-2t}} = \pm 1 \]  \hspace{1cm} (21)

However, simulations show that the iterative algorithm in (3) reaches a scaled value of the principal eigenvector and then oscillates between two weight vectors. This is due to time discretization. We have analyzed this peculiar behavior and the details will be given in a later paper.

REFERENCES