Plotting 3-D Orthogonal Vectors in Realtime

Sandipan Dey,
Consultant,
Wipro KDC,
Kolkata, India,
sandipan.dey@gmail.com

March 6, 2012

Abstract

In this paper we present a novel technique of plotting a set of vectors that change over time (both in magnitude and direction). The changes are captured by the change in the color by which they are plotted in 3D space. We apply this technique to plot the dominant (orthogonal) eigenvectors obtained by doing Principal Component Analysis (PCA) on a realtime dataset. We ran our experiments on synthetically generated dataset and on realtime astronomy, traffic and stock datasets, each having multiple attributes (dimensions). By visualizing only the changes in the most dominant eigenvectors one could apprehend the change in the underlying realtime data.

Keywords

PCA, eigenvectors, visualization.

Introduction

We used two different pseudo-random generators to synthetically generate mutually orthogonal 3-D vectors for our proof of concept purpose. These vectors actually converge to their corresponding mean vectors that is easily shown by our color-coded plotting. Also, it can immediately be seen that they form right angles to each other in 3-D space. Next we apply this visualization technique to the orthogonal eigenvectors obtained from PCA.

Principal Component Analysis (PCA) [5,6,8,10] is a very popular technique primarily used to capture the most varying directions in the dataset and to reduce the dimensionality of the data, since few of the most dominant eigenvectors of the underlying covariance matrix (also call principal components) can be used to describe the variance in the entire data. We choose the most two dominant 3-D eigenvectors to monitor the

changes in the underlying data. We use a simple color-code mapping to visualize the change in the direction of the eigenvectors immediately. A simple 3-D to 2-D mapping technique [3] has been used to plot the 3-D eigenvectors. Similar work was done by [2] in their distributed eigenstate monitoring algorithm but visualization of eigenvectors was never used to detect the changes in data / convergence of the eigenvectors towards a mean vector. Also, we applied this visualization technique on synthetically generated orthogonal vectors and on the orthogonal eigenvectors obtained by PCA from realtime stock datasets. This technique can be applied on any realtime data such as traffic data (recorded by sensors) or astronomy data and this technique can also be easily extended to distributed settings.

Problem Definition

Given a source that continuously generates 3-D vectors in such a way that each of these vectors belong to either of two sets (V_1 and V_2) and any two vectors belonging to different sets mutually orthogonal 3-D vectors ($\vec{v_1} \in V_1 \wedge \vec{v_2} \in V_2 \Rightarrow < \vec{v_1}, \vec{v_2} >= 0$, [1, 9]), plot those vectors using color coding. Also, use color coding to represent the convergence of any vector $\vec{v_1} \in V_1$ to a mean vector $\vec{v_{1m}}$ (if at all they converge) and similarly for $\vec{v_2} \in V_2$ to a mean vector $\vec{v_{2m}}$.
The above visualization technique can be applied to problems like the following one: Given a realtime dataset $\{X(t) : t = 1, 2, 3, \ldots\}$, visualize the change in the dataset (over time) by plotting the principal components. Also, visualize the convergence of these dominant eigenvectors, if they converge (to some mean vectors) at all.

Approach

Synthetically generating 3-D orthogonal Vectors by the Pseudorandom generator

Two sets (V_1 and V_2) of orthogonal vectors are generated in the following manner:

$$
\begin{aligned}
v_{1x} &= C_{1x} + dx_1 \times rand() \\
v_{1y} &= C_{1y} + dy_1 \times rand() \\
v_{1z} &= C_{1z} + dz_1 \times rand() \\
v_{2x} &= C_{2x} + dx_2 \times rand() \\
v_{2y} &= C_{2y} + dy_2 \times rand() \\
v_{2z} &= C_{2z} + dz_2 \times rand() \\
\vec{v_1} &= (v_{1x})\hat{i} + (v_{1y})\hat{j} + (v_{1z})\hat{k} \in V_1 \\
\vec{v_2} &= (v_{2x})\hat{i} + (v_{2y})\hat{j} + (v_{2z})\hat{k} \in V_2
\end{aligned}
$$

where Cs are constants such that $C_{1x} \times C_{2x} + C_{1y} \times C_{2y} + C_{1z} \times C_{2z} = 0$ and $dx_1, dx_2, dy_1, dy_2, dz_1, dz_2 \rightarrow 0$ and we have $< \vec{v_1}, \vec{v_2} > \approx 0$.

Generating 3-D orthogonal eigenvectors thorugh PCA

Given a realtime dataset $\{X(t) : t = 1, 2, 3, \ldots\}$ with n attributes (columns), the last (most recent) m data rows $\{X(t) : t = t_i, t_i + 1, \ldots t_i + m - 1\}$ are stored (Initially $t_i = 1$, so that at every $t_i = k.m+1$, m new rows $\{X(k.m+1), \ldots, X((k+1).m)\}$ are obtained), where $k = 0, 1, 2, \ldots$ Also, the most relevant (significant) 3 columns from the dataset are chosen (by the domain experts) so that at every time instant $t = k.m+1$, a new dataset $X_{m\times 3}(t)$ is obtained. Then PCA is performed on this dataset and 2 dominant 3-D eigenvectors ($\vec{v_1}(t)$ and $\vec{v_2}(t)$) are chosen, Now, 3-D to 2-D mapping [3] is used to visualize these eigenvectors. If the underlying data changes abruptly then that change will be captured by changes in these dominant eigenvectors. Otherwise, if the changes are subtle (mean values of the attributes do not change much), these dominant vectors will gradually converge to their corresponding mean vectors. The problem that we address here is to *visualize* the *dominant* (and changing) eigenvectors dynamically .

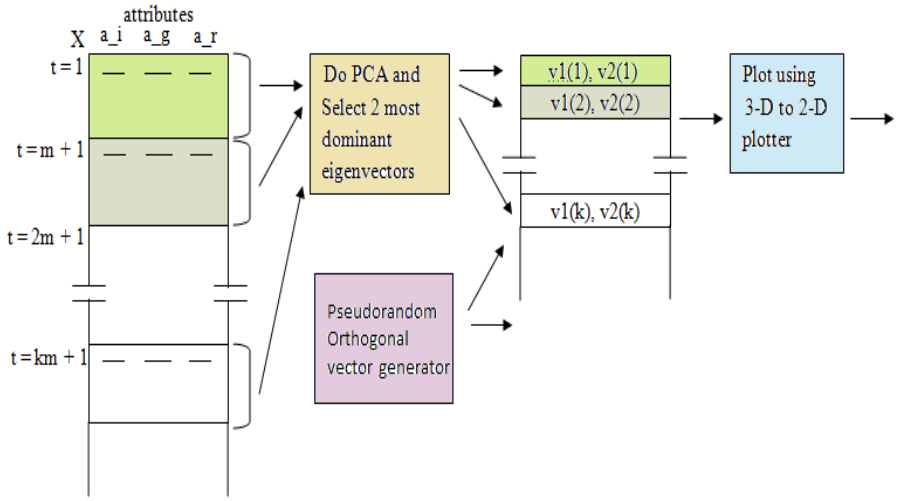

Figure 1: Schematic diagram for plotting 3-D orthogonal vectors

Vizualization

Both the pseudorandom orthogonal vector generation algorithm and the PCA algorithm generates outputs in form of ($\vec{v_1}$,$\vec{v_2}$) tuples where they represent the dominant (orthogonal, i.e., $v_1^T.v_2 = 0$) eigenvectors, i.e., each output tuple generated by the algorithms is of the form $(v_{1x}, v_{1y}, v_{1z}), (v_{2x}, v_{2y}, v_{2z})$.
In graphical representation, each eigenvector will represent a direction in 3D space and moreover since they are orthogonal to each other they will represent two mutually perpendicular directions. It's always good visual representation if we distinguish the two distinct orthogonal eigenvectors (namely 1st most dominant and 2nd most dominant

eigenvector) by different color codes. Also, it will be good and easily understandable if we can show how the distributed algorithm convergences to each of the eigenvectors using color codes. We are going to use 3D to 2D mapping technique proposed in [3]. We use properties of *inner product* of two vectors to assign different color codes to the eigenvectors. The following steps describe the technique we used for color code mapping:

1. We have two distinct mutually perpendicular directions for the two orthogonal vectors. Now, additionally each vector will have its own radius of convergence that is defined by the respective *solid angles* $\delta\theta$ or $\delta\phi$, as shown in . As it can be seen, if the variation / fluctuation in data is small (as is the case for the synthetically generated data, the orthogonal vectors quickly converge to their corresponding means.

2. Let the i^{th} row of the output generated by PCA / our orthogonal vector generation algorithm be denoted by $(\vec{v_1}^{(i)},\ \vec{v_2}^{(i)})$. Such output tuples are continuously generated. We use a backend thread to asynchronously read the stream and update the frontend UI (applet).

3. Also, let's denote the set of 1st dominant eigenvectors computed by the algorithm by V_1 and set of 2nd dominant eigenvectors by V_2. Hence,

   (a) $\vec{v}_1 \in V_1 \wedge \vec{v}_2 \in V_2 \Rightarrow < \vec{v}_1, \vec{v}_2 >= v_1^T.v_2 \approx 0$, (by orthogonality)

   (b) $< \vec{v}_1, \vec{v}_1 >=< \vec{v}'_1, \vec{v}'_1 >= 1$, (if we have orthonormality as well)

   (c) $\vec{v}_1, \vec{v}'_1 \in V_1 \Rightarrow < \vec{v}_1, \vec{v}'_1 >= v_1^T.v'_1 \approx 1$. (Since the inner product is a similarity measure, if two vectors are similar, their product will be close to 1. Also, $< \vec{v}_1, \vec{v}'_1 > \leq < \vec{v}_1, \vec{v}_1 > . < \vec{v}'_1, \vec{v}'_1 >= 1$, by the ***Cauchy-Shwartz inequality***, [4, 7])

   (d) $\vec{v}_2, \vec{v}'_2 \in V_2 \Rightarrow < \vec{v}_2, \vec{v}'_2 >= v_2^T.v'_2 \approx 1$ (Similarly)

   (e) Hence, for any two vectors u and w, $\langle u, w \rangle \approx 1 \Rightarrow$ they belong to the same set. But $\langle u, w \rangle \approx 0 \Rightarrow$ they belong to different sets.

4. Initially, $V_1 = V_2 = \Phi$. Iteratively compute $V_1 = V_1 \cup \{\vec{v_1}^{(i)}\}$ and $V_2 = V_2 \cup \{\vec{v_2}^{(i)}\}$, by adding the corresponding vectors obtained from the i^{th} output tuple generated by the algorithm.

5. At any given instant, we compute the mean dominant eigenvector directions $\vec{e_1}$ and $\vec{e_2}$ from the already-computed eigenvectors from the set V_1 and V_2 respectively in the following manner (by taking online average):

   $\vec{e}_1 = \{e_{1x}, e_{1y}, e_{1z}\}$, with $e_{1x} = \frac{\sum\limits_{v_1 \in V_1} v_{1x}}{|V_1|}$, $e_{1y} = \frac{\sum\limits_{v_1 \in V_1} v_{1y}}{|V_1|}$, $e_{1z} = \frac{\sum\limits_{v_1 \in V_1} v_{1z}}{|V_1|}$,

   $\vec{e}_2 = \{e_{2x}, e_{2y}, e_{2z}\}$, with $e_{2x} = \frac{\sum\limits_{v_2 \in V_2} v_{2x}}{|V_2|}$, $e_{2y} = \frac{\sum\limits_{v_2 \in V_2} v_{2y}}{|V_2|}$, $e_{2z} = \frac{\sum\limits_{v_2 \in V_2} v_{2z}}{|V_2|}$.

   Simplifying the above, we get $\vec{e}_1 = \frac{1}{|V_1|} \sum\limits_{v_1 \in V_1} \vec{v_1}$ and $\vec{e}_2 = \frac{1}{|V_2|} \sum\limits_{v_2 \in V_2} \vec{v_2}$.

6. As soon as we get a new vector $\vec{v}$ from the backend, we perform the following steps:

   (a) Normalize eigenvector $\vec{v} = \{v_x, v_y, v_z\}$ to obtain $\vec{\vartheta} = \left\{ \frac{v_x}{\sqrt{v_x^2+v_y^2+v_z^2}}, \frac{v_y}{\sqrt{v_x^2+v_y^2+v_z^2}}, \frac{v_z}{\sqrt{v_x^2+v_y^2+v_z^2}} \right\}$.

   (b) Find if $\vec{\vartheta} \in V_1$ or $\vec{\vartheta} \in V_2$.

   (c) Use colors gradients (C_1, C_1') to color the vectors in set V_1 and colors gradients (C_2, C_2') to color the vectors in set V_2.

   (d) Use orthonormality to find set membership, e.g., if $\vec{\vartheta} \in V_1$, the inner product $< \vec{\vartheta}, \vec{e}_1 > \approx 1$ and $< \vec{\vartheta}, \vec{e}_2 > \approx 0$.

   (e) Use gradient colors to represent the dispersion of the eigenvectors in the same set. To find the solid angles, again use the inner products (as similarity measure), e.g., if $\vec{\vartheta} \in V_1$, use the inner product $< \vec{e_1}, \vec{\vartheta} >$ to find how far $\vec{v_1}$ is from the centroid of the set V_1 (i.e., measure the solid angle $\delta\theta$) and accordingly assign a *convex combination* of the color codes C_1 and C_1' to the vector $\vec{\vartheta}$. The vector $\vec{v_1}$ will have more C_1 components in its color if it is closer to $\vec{e_1}$. Consequently it will have more C_1' components if the solid angle between $\vec{e_1}$ and $\vec{v_1}$ is larger. As seen from figure, the eigenvectors closer to $\vec{e_1}$ are yellower, while the ones comparatively away from the convergence are redder.

   (f) Similarly the 2nd dominant eigenvector is colored.

   (g) The algorithms for construction of the sets and color mapping are described in the following section.

Algorithms

Algorithm 1 Construct sets V_1 and V_2

$V_1 \leftarrow \Phi$
$V_2 \leftarrow \Phi$
$\vec{e_1} \leftarrow 0$
$\vec{e_2} \leftarrow 0$
for all $(\vec{v_1}^{(i)},\ \vec{v_2}^{(i)})$ **do**
    $\vec{e_1} \leftarrow \frac{|V_1|.\vec{e_1}+\vec{v_1}^{(i)}}{|V_1|+1}$
    $\vec{e_2} \leftarrow \frac{|V_2|.\vec{e_2}+\vec{v_2}^{(i)}}{|V_2|+1}$
    $V_1 \leftarrow V_1 \cup \{\vec{v_1}^{(i)}\}$
    $V_2 \leftarrow V_2 \cup \{\vec{v_2}^{(i)}\}$
end for

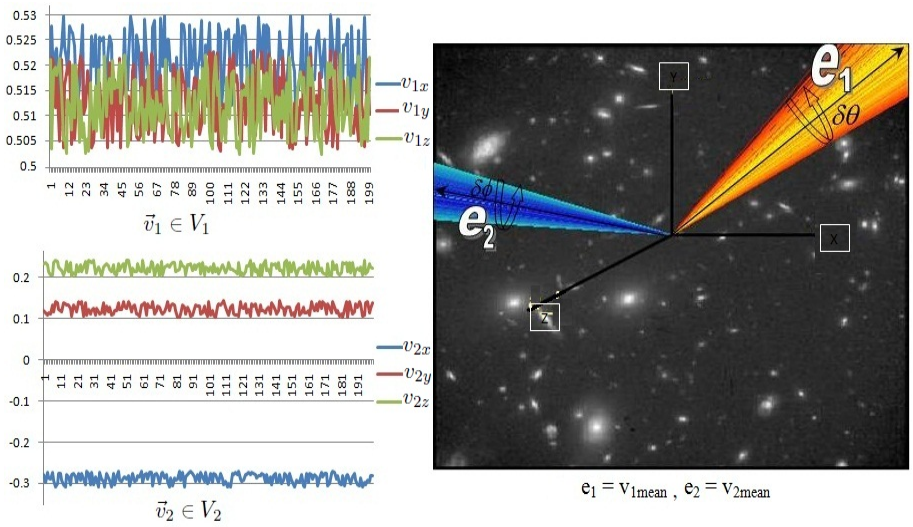

Figure 2: The mapping of color codes in the graph applet: the solid angles of convergence with synthetically generated vectors

Algorithm 2 Draw a new (normalized) vector $\vec{\vartheta}$ with proper color

if $< \vec{e_1}, \vec{\vartheta} > \approx 1$ **then** {equivalently $< \vec{e_2}, \vec{\vartheta} > \approx 0$}
  $V_1 \leftarrow V_1 \cup \{\vec{\vartheta}\}$
else if $< \vec{e_2}, \vec{\vartheta} > \approx 1$ **then** {equivalently $< \vec{e_1}, \vec{\vartheta} > \approx 0$}
  $V_2 \leftarrow V_2 \cup \{\vec{\vartheta}\}$
end if
if $\vec{\vartheta} \in V_1$ **then**
  Use $< \vec{e_1}, \vec{\vartheta} >$ to find the solid angle $\delta\theta$ and accordingly assign a convex combination of the colors $\lambda C_1 + (1-\lambda)C_1'$ to $\vec{\vartheta}$, where $0 \leq \lambda \leq 1$, λ being directly proportional to the inner product $< \vec{e_1}, \vec{\vartheta} >$
end if
if $\vec{\vartheta} \in V_2$ **then**
  Use similar logic to color $\vec{\vartheta}$
end if

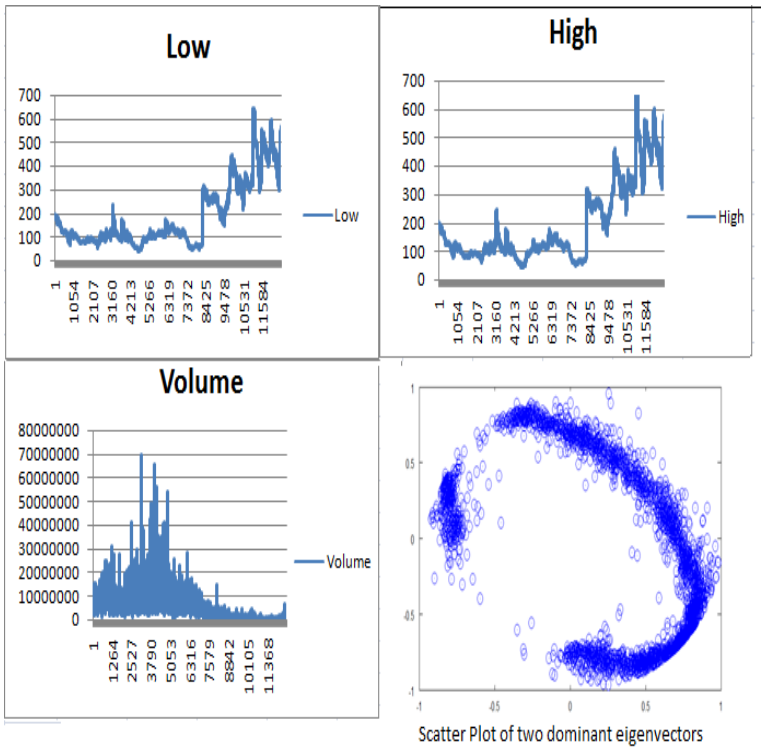

Figure 3: Dominant Eigenvectors for Realtime Stock data

References

[1] Sheldon Axler. *Linear Algebra Done Right (2nd ed.)*. Springer-Verlag, Berlin, New York, 1997.

[2] Kamalika Das, Kanishka Bhaduri, Sugandha Arora, Wesley Griffin, Kirk D. Borne, Chris Giannella, and Hillol Kargupta. Scalable distributed change detection from astronomy data streams using local, asynchronous eigen monitoring algorithms. In *SDM*, pages 245–156, 2009.

[3] Sandipan Dey, Ajith Abraham, and Sugata Sanyal. A very simple approach for 3-d to 2-d mapping. *Image Processing & Communications*, 11(2):75–82, 2006.

[4] S.S. Dragomir. A survey on cauchy-bunyakovsky-schwarz type discrete inequalities. *Journal of Inequalities in Pure and Applied Mathematics*, 4(63), 2003.

[5] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press Baltimore, MD, USA, 1996.

[6] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques, Third Edition*. The Morgan Kaufmann Series in Data Management Systems, 2011.

[7] G. H. Hardy, J. E. Littlewood, and G. Plya. *Inequalities*. Cambridge University Press, Cambridge, England, 1952.

[8] I.T. Jolliffe. *Principal Component Analysis (2nd ed.)*. Springer, 2002.

[9] David C. Lay. *Linear Algebra and Its Applications (3rd ed.)*. AddisonWesley, 2006.

[10] Lindsay I Smith. A tutorial on principal components analysis, 2002. (http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf).