Rethinking Linear Algebra for Computer Science: Applying Vygotsky's Theory of Learning[*]

Abbas Attarwala
Computer Science Department
California State University, Chico
Chico, CA 95973
aattarwala@csuchico.edu

Abstract

This paper explores the application of Vygotsky's educational theories within the teaching of applied linear algebra for computer science students. The key point of this pedagogical study is a case study on principal component analysis (PCA), illustrated through noisy image compression, which serves as a representative example of the comprehensive teaching methodology applied throughout the course. This case study highlights the integration of key linear algebra concepts—eigenvectors, eigenvalues, covariance matrices, dot products, and change of basis matrices—demonstrating their application in a tangible real-world scenario. Employing MATLAB as a mediational tool, the teaching approach is scaffolded in accordance with Vygotsky's theory of learning, which progressively builds upon students' existing knowledge. The significance of aligning teaching practices with Vygotsky's theories lies in their proven ability to enhance conceptual understanding and student engagement, ultimately creating a deeper learning experience.

[*]Copyright ©2024 by the Consortium for Computing Sciences in Colleges. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the CCSC copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Consortium for Computing Sciences in Colleges. To copy otherwise, or to republish, requires a fee and/or specific permission.

1 Introduction

The modernization of teaching applied linear algebra through computational tools is essential for equipping students in engineering and computer science with practical problem-solving skills. At the University of Illinois Urbana-Champaign, a refreshed applied linear algebra course featuring these tools saw a rise in popularity and enrollment, showcasing the pivotal role of such tools in making linear algebra education more engaging and relevant [5]. This trend is indicative of a broader need across disciplines for computational proficiency moving away from theory-laden approaches and empowers students to apply linear algebra concepts to diverse professional fields [5, 4, 2].

Incorporating Vygotsky's educational theory, my pedagogical approach at Boston University and soon at California State University, Chico, positions both the teacher and the student as active agents in the learning process. Vygotsky's emphasis on the social aspects of learning aligns with the collaborative environment I encourage in my classroom. Here, the teacher's role transcends providing assistance; it involves cultivating high-quality, meaningful interactions that significantly contribute to students' learning experiences [6]. This method, deeply rooted in social constructivism, leverages the dynamics of social interaction to facilitate the evolution of students' understanding, epitomizing Vygotsky's concept of good learning within the Zone of Proximal Development (ZPD)—a space where students thrive under skilled guidance and scaffolding [8].

The paper's structure is as follows: Section 2 outlines Vygotsky's educational theories. Section 3 applies these theories to the teaching of PCA with noisy image compression. Finally, Section 4 provides concluding remarks.

2 Integrating Vygotsky in Linear Algebra Teaching

Lev Vygotsky's social development theory [8] and its application in modern education have gained increasing relevance, particularly in the integration of technology in learning environments. The ZPD and the role of the More Knowledgeable Other (MKO), typically the instructor in a classroom, are central to this theory. Contemporary educational research validates the significance of these concepts in enhancing collaborative and socialized learning [1, 3]. Recent trends in educational research, as noted by [7], show a growing interest in how cultural contexts and social dynamics influence computer technology's role in education. This aligns with the increasing acknowledgment of Vygotsky's socio-cultural theory in educational practices.

In my teaching of linear algebra, I employ Vygotsky's principles by using MATLAB as a mediational tool, bridging theory with practical application.

This approach, emphasizing scaffolding, allows students to progress within their ZPD from basic to advanced linear algebra concepts. As students apply linear algebra concepts like eigenvectors to new contexts such as image compression, they assimilate new information into their existing cognitive schema, modifying their understanding to accommodate new insights. Interactive lectures and team-based problem-solving create a collaborative learning environment, encouraging students to actively construct knowledge. This method not only improves computational proficiency but also aims to deepen conceptual understanding, underscoring the real-world relevance of linear algebra.

This paper presents a case study that exemplifies the application of these principles: using PCA in noisy image compression as a practical teaching example. This case study is not just a representation of how linear algebra can be taught through Vygotsky's lens but also demonstrates why such an approach is vital in today's educational landscape. It addresses the need for teaching methods that promote critical thinking, adapt to diverse learning styles, and prepare students for challenges in a technologically advanced society. By integrating Vygotsky's educational theories, the goal is to equip students with not just academic knowledge but also with skills essential for lifelong learning and professional success.

3 Teaching of PCA

Students begin their study of eigenvectors and eigenvalues by employing both manual calculations and MATLAB. By this point in the course, they have developed proficiency in computing the eigenvectors and eigenvalues of 2×2 and 3×3 matrices. However, the broader significance and practical applications of eigenvectors and eigenvalues remain unclear to them. As a result, these concepts currently exist in isolation, appearing as standalone topics without apparent connection to their broader mathematical or real-world relevance.

To enhance intuition, I employ MATLAB's `eigshow` command, allowing students to visually grasp eigenvectors and eigenvalues of a certain 2×2 matrix $\mathbf{A}$. This command visualizes how each unit vector $\mathbf{x}$ is transformed into the corresponding eigenvector $\mathbf{Ax}$, highlighting the scaling relationship between $\mathbf{Ax}$ and $\mathbf{x}$. This concept is depicted in Figure 1.

At this stage, my lecture becomes interactive, inviting student questions while tracing the unit circle with the green vector $\mathbf{x}$, and guiding them to identify when the blue vector $\mathbf{Ax}$ becomes an eigenvector of the matrix $\mathbf{A}$.

MATLAB serves as a mediational tool, aligning with Vygotsky's theory which underscores the importance of tool usage and integration in students' developmental processes. As students advance in their linear algebra studies, their engagement with such tools transitions from mere eigenvector computa-

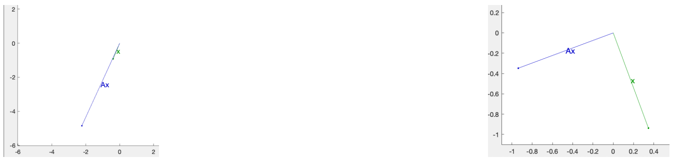

Figure 1: The left image demonstrates **x** (green vector) as an eigenvector of matrix **A**, transforming into **Ax** (blue vector) as a scaled version of **x**. The right image shows that the green vector is not an eigenvector of **A**, as its transformation **Ax** (blue vector) does not maintain the same direction as **x**.

tions to a more profound understanding of the underlying mechanics of these computations.

After introducing students to eigenvectors, I present three 10×2 matrices (referred to as **data** and as seen in Figure 2 where I have plotted them) to demonstrate two-dimensional data compression: one matrix with data along the $y = x$ line, another along the x axis, and the third along the y axis. The complete MATLAB code is available in the Appendix A and Appendix B and Appendix C of this paper.

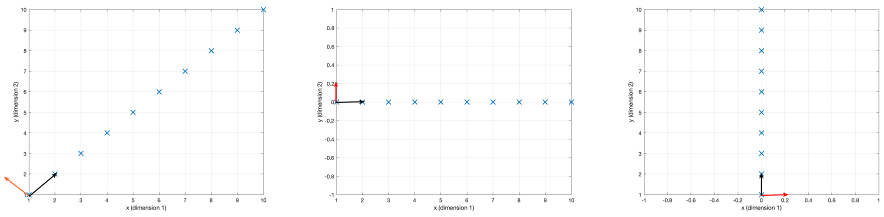

Figure 2: First image shows data along the y=x line, the second image displays data along the x-axis, and the third image represents data along the y-axis.

The students' task involves identifying the optimal vector for compressing two-dimensional data, as depicted in Figure 2. This figure clearly shows that the best compression vector coincides with the direction of the maximum data spread. In the three scenarios presented, this corresponds to along the $y = x$ line, the x-axis, and the y-axis. Through this exercise, students learn that the optimal compression vector not only aligns with the maximum spread but also captures the entire variance of the original data. This is visually represented in each plot of Figure 2 by a black arrow, indicating the direction of maximum variance. Specifically, the variance is 18.33 for data along the $y = x$ line and 9.1667 for data along the x and y axes. To quantify this variance, students calculate it either manually or using the *var* function in MATLAB.

I then encourage my students to create a 2×2 covariance matrix for each of the three **data** matrices. In these matrices, the diagonal elements represent the variance along the x and y axes. This exercise aids in understanding that the total variance of the **data** matrix, which they previously calculated manually or using MATLAB's *var* function, is equal to the sum of the diagonal elements in the covariance matrix. The off-diagonal elements are determined by the relationship between the x and y dimensions in each data set. The covariance matrix is $\begin{bmatrix} 9.1667 & 9.1667 \\ 9.1667 & 9.1667 \end{bmatrix}$ when the data points lie along the $y = x$ line. Conversely where data points are aligned along the y-axis and x-axis respectively, the covariance matrices are $\begin{bmatrix} 0 & 0 \\ 0 & 9.1667 \end{bmatrix}$ and $\begin{bmatrix} 9.1667 & 0 \\ 0 & 0 \end{bmatrix}$.

Computing eigenvectors for these matrices using MATLAB's `eig` command, students discover that the primary eigenvector (black arrow) aligns with the direction of maximum spread, and its eigenvalue represents the total variance. Additionally, they observe a secondary eigenvector (red arrow), perpendicular to the first and indicating zero variance, as evident from its eigenvalue of 0. This observation reinforces the understanding that the primary eigenvector captures all the variance of the original data matrix. The collaborative approach I adopt in my lectures aligns with Vygotsky's scaffolding concept and ZPD. ZPD is the gap between what a student can do without help and what a student can do with help. The initial exercise where students draw the black vector indicating maximum spread now seamlessly integrates with their newfound understanding gained from computing eigenvectors of the covariance matrix. This progression illustrates the scaffolding process and helping students through the ZPD, where early, simpler tasks lay the foundation for grasping more complex concepts.

Students often raise two key questions at this juncture: (1) Why does computing the eigenvectors of a covariance matrix indicate the direction of maximum spread? (2) How does this method apply to randomly generated or multi-dimensional data? In addressing students' queries regarding the covariance matrix and its application to multi-dimensional data, I engage in a collaborative problem-solving process. To address the first query, instead of the 10×2 **data** matrix from Figure 2, I now consider a mean-adjusted $n \times d$ matrix referred to as **data**. Compressed data is defined as $\mathbf{data_c} = \mathbf{data} \times \mathbf{c}$, where $\mathbf{c}$ is the compression vector and $\mathbf{data_c}$ is the compressed data along the $\mathbf{c}$ vector. The goal is to find the compression vector $\mathbf{c}$ that maximize the variance of $\mathbf{data_c}$, given by $\mathrm{Var}(\mathbf{data_c}) = \frac{\mathbf{data_c}^T \times \mathbf{data_c}}{n}$. This is similar to the exercise that my students engaged previously with Figure 2 in finding the black vector pointing in the direction of the maximum variance. The optimization problem then is to maximize $\mathbf{c^T} \times \mathbf{data^T} \times \mathbf{data} \times \mathbf{c}$, where $\mathbf{data^T} \times \mathbf{data}$ represents the covariance matrix **C**. This translates to aligning the vector $\mathbf{C} \times \mathbf{c}$ with the vector **c**,indicating that **c** is an eigenvector of **C**. This is because in order to maximize the dot product of the vectors **c** and $\mathbf{C} \times \mathbf{c}$ the vector $\mathbf{C} \times \mathbf{c}$ must be in the same direction as **c**. Through this interactive and joint exploration, I aim to transition students' understanding from a social plane, where learning is collaborative and external, to a cognitive plane, creating internalization of these concepts. This internalization process will be assessed later through individual assignments, allowing me to evaluate each student's independent grasp and application of the learned material.

For the second question, I use a 360×640 grayscale image as the **data** matrix. The complete MATLAB code is available in Appendix D of this paper.

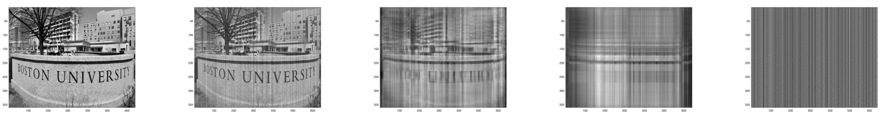

Figure 3: Grayscale Image Compression: Original image and its compressed versions using the 100 largest, 30 largest, 1 largest, and 500 smallest eigenvectors.

Students observe from Figure 3 the effects of image compression using different numbers of eigenvectors. What my students are truly amazed by is that there is very little to no variance captured by the 500 smallest eigenvectors (based on their eigenvalues). These visual examples highlight the importance of variance preservation and demonstrate that discarding eigenvectors with minimal variance has a negligible impact on image quality and can lead to higher compression without losing much of the information captured in the picture. This hands-on MATLAB experience provides students with a concrete understanding of linear algebra concepts, as they visualize the practical effects of image compression with different numbers of eigenvectors.

As students observe and discuss the outcomes of different compression levels, they are engaging in a social learning process, which then transitions into individual understanding and cognitive development. The use of MATLAB as a mediational tool in this experiment not only simplifies the complex concept of dimensional reduction in PCA but also makes the learning experience more engaging and relatable. This process, aligning with Vygotsky's educational theory, transitions learning from a social context – collaborative discussions and guided exploration in the classroom – to internal cognitive processing.

4 Conclusions

In conclusion, integrating Vygotsky's theory into linear algebra teaching fosters an engaging, collaborative, and effective learning environment, crucial for students' mastery and application of the subject.

Acknowledgement

I gratefully acknowledge the use of OpenAI's ChatGPT for proofreading, grammatical checks, and other text editing tasks.

References

[1] Megan Cicconi. "Vygotsky meets technology: A reinvention of collaboration in the early childhood mathematics classroom". In: *Early Childhood Education Journal* 42 (2014), pp. 57–65.

[2] Robert M Corless, David J Jeffrey, and Azar Shakoori. "Teaching Linear Algebra in a Mechanized Mathematical Environment". In: *arXiv preprint arXiv:2306.00104* (2023).

[3] Karen Gooch and Paula Saine. "Integration of the visual arts and Web 2.0 technologies in the classroom". In: *New England Reading Association Journal* 47.1 (2011), p. 92.

[4] Geoffrey R Hutchison. "Integrating python into an undergraduate mathematics for chemists course". In: *Teaching Programming across the Chemistry Curriculum.* ACS Publications, 2021, pp. 123–134.

[5] Mariana Silva et al. "Innovating and modernizing a Linear Algebra class through teaching computational skills". In: *2022 ASEE Annual Conference & Exposition.* 2022.

[6] Roland G Tharp and Ronald Gallimore. *Rousing minds to life: Teaching, learning, and schooling in social context.* Cambridge University Press, 1991.

[7] Irina Verenikina. "Vygotsky in twenty-first-century research". In: *EdMedia+ innovate learning.* Association for the Advancement of Computing in Education (AACE). 2010, pp. 16–25.

[8] Lev S Vygotsky. *Mind in society (M. Cole, V. John-Steiner, S. Scribner, & E. Souberman, Eds.)* 1978.

Appendix A MATLAB Code for compression along the y=x line

```
%compression on Y=X line
x = 1:10;
y = x;
plot(x, y, 'X', 'MarkerSize', 10, 'LineWidth', 2)
xlabel('x (dimension 1)')
ylabel('y (dimension 2)')
grid on
data=[x' y']
[V,D]=eig(cov(data))
% Compute the mean of data
meanData = mean(data);

% Scale factor for the eigenvectors
scaleFactor = 5;

% Plotting the scaled eigenvectors
hold on; % Keep the current plot
quiver(meanData(1), meanData(2), scaleFactor * V(1,1),
    scaleFactor * V(2,1), 'r'); % First eigenvector
quiver(meanData(1), meanData(2), scaleFactor * V(1,2),
    scaleFactor * V(2,2), 'k'); % Second eigenvector
hold off;
```

Appendix B MATLAB Code for compression along the X axis

```
%compression on X axis
x = 1:10;
y = zeros(1,10);
plot(x, y, 'X', 'MarkerSize', 10, 'LineWidth', 2)
xlabel('x (dimension 1)')
ylabel('y (dimension 2)')

grid on
data=[x' y']
[V,D]=eig(cov(data))
% Compute the mean of data
meanData = mean(data);
```

```
% Scale factor for the eigenvectors
scaleFactor = 5;

% Plotting the scaled eigenvectors
hold on; % Keep the current plot
quiver(meanData(1), meanData(2), scaleFactor * V(1,1),
    scaleFactor * V(2,1), 'r'); % First eigenvector
quiver(meanData(1), meanData(2), scaleFactor * V(1,2),
    scaleFactor * V(2,2), 'k'); % Second eigenvector
hold off;
```

Appendix C MATLAB Code for compression along the Y axis

```
%compression on Y axis
y = 1:10;
x = zeros(1,10);
plot(x, y, 'X', 'MarkerSize', 10, 'LineWidth', 2)
xlabel('x (dimension 1)')
ylabel('y (dimension 2)')
grid on
data=[x' y']
[V,D]=eig(cov(data))
% Compute the mean of data
meanData = mean(data);

% Scale factor for the eigenvectors
scaleFactor = 5;

% Plotting the scaled eigenvectors
hold on; % Keep the current plot
quiver(meanData(1), meanData(2), scaleFactor * V(1,1),
    scaleFactor * V(2,1), 'r'); % First eigenvector
quiver(meanData(1), meanData(2), scaleFactor * V(1,2),
    scaleFactor * V(2,2), 'k'); % Second eigenvector
hold off;
```

Appendix D Noisy Image Compression

```
%compression on BU.jpg example
img_in=double((rgb2gray(imread('BU.jpg'))));
numberOfDataPoints=size(img_in,1);
```

```
numberOfDimensions=size(img_in,2);
numberOfEigenVectors=100
imagesc(img_in);
colormap(gray);  % Set the colormap to grayscale
[V,D]=eig(cov(img_in))
imagesc((V(:,numberOfDimensions-numberOfEigenVectors:
    numberOfDimensions)*V(:,numberOfDimensions-
    numberOfEigenVectors:numberOfDimensions)'*img_in')')
```