Visualizing Low-Rank Tensors With Software

Nam Nguyen, Rob Vanyo

Department of Mathematics
Tensors: A tensor is a set of values arranged in a $d$-dimensional array, much like a matrix. In fact, a matrix is an example of a 2-dimensional tensor. If the tensor has the same number of rows, or entries, in every dimension, then we call that number the resolution, $m$. Thus, an $n \times n$ matrix has resolution-$n$. The following is an example of a tensor with dimension-3, resolution-2.

\[
\begin{bmatrix}
a_{111} & a_{121} \\
a_{211} & a_{221}
\end{bmatrix},
\begin{bmatrix}
a_{112} & a_{122} \\
a_{212} & a_{222}
\end{bmatrix}
\]

The Tensor Product: The tensor product, denoted $\otimes$, is analogous to the outer product of vectors. Thus, a tensor like the one above may have been achieved by:

\[
\begin{bmatrix}
a_1 \\
a_2
\end{bmatrix} \otimes \begin{bmatrix}
b_1 \\
b_2
\end{bmatrix} \otimes \begin{bmatrix}
c_1 \\
c_2
\end{bmatrix} = \begin{bmatrix}
a_1 b_1 c_1 & a_1 b_2 c_1 \\
a_2 b_1 c_1 & a_2 b_2 c_1
\end{bmatrix},
\begin{bmatrix}
a_1 b_1 c_2 & a_1 b_2 c_2 \\
a_2 b_1 c_2 & a_2 b_2 c_2
\end{bmatrix}
\]
The Curse of Dimensionality: A tensor with $d$ dimensions and resolution-$m$ takes a computer $m^d$ operations just to access. When $m$ and $d$ become large, computational cost becomes astronomical. This is called the Curse of Dimensionality.

The Separated Representation: The separated representation of a tensor is a potential answer to this problem. Here, a $d$-dimensional tensor with resolution-$m$ is represented with error $\epsilon$ by a sum of $r$ tensor-products of $d$ one-dimensional vectors $V$ with $m$ entries in each vector. So, if $s_l$ is a scalar, the separated representation of a tensor is given by

$$\sum_{l=1}^{r} s_l V_1^l \otimes V_2^l \otimes \ldots \otimes V_d^l$$

Now, a computer may carry out operations with a tensor with only $rdm$ values to keep track of. However, this structure is not yet well understood.
To plot tensors on a 2-dimensional plane, we pick 3 basepoints that define the plane. Then a projection of the tensor onto the span of the basepoints is found and scaled into an \( xy \)-point on the plane. This \( xy \)-point represents the tensor in our 2-d image.

The angle \( \theta \) is used to color-code the points plotted. A smaller \( \theta \) means the tensor was closer to the plane, and a darker color is plotted.
In order to implement our plotting scheme and explore these separated representations quickly and easily, we wrote a program. It was written in the Python language, and included the following features:

- Plotting of colored points into a PostScript document.

- Option to plot a number of random points.

- Option to plot a random walk of points.

- Option to plot homotopies.

- Option to plot an alternating least squares fitting on a rectangular region.
Designing a GUI

The program was functional, but unwieldy. To make it easier to run, and user-friendly, we designed a GUI to go with it. This was the main focus of our work so far this year. Some details considered were:

- A viewing window that shows up-to-date plots.
- The ability to zoom and pan plots.
- The ability to change any and all pertinent parameters on the fly.
- The ability to save and load previous work sessions.
The latest version includes many of the desired features, and more. It has:

- A GUI built with Tkinter, and featuring items one would expect from any software, such as clickable buttons.

- A viewing window built with the Visualization Tool Kit, that allows for the zooming, panning, and rotating of plots.

- A project save and load feature.

- A printable image generator.

- The ability to easily change any parameter, such as rank, dimension, etc.

- The easy choosing of basepoints, either randomly or by direct input.

- Customizable and easy-to-use functions, such as linepoints, alternating least squares, and random walk.

- Plotting and viewer options to trim the image or reset the camera.

- A status box that tracks and displays recent user activity.
Overview of Program
**Linepoints:** This function plots points in the viewing plane along one of the three lines defined by the basepoints. It makes it easier to see the basepoints in convoluted plots.

**Alternating Least Squares:** This function uses the alternating least squares method to find the closest tensors that can be represented in a given rectangular region of the viewing plane. This is the function we implement most often.

**Random Walk:** This function approximates and plots a random walk of tensors. Each new random tensor is within a given radius of the previous tensor. The new tensor is then plotted.
\(G_3\) is a representative tensor of a class of \(3 \times 3 \times 3\) tensors in which each of them is truly rank 3 and cannot be approximated by rank-2 tensors. 

\(G_3\) is a combination of the three basepoints \(B_1, B_2, B_3\) where 

\[
B_1 = (e_1 + e_2) \otimes e_2 \otimes e_2, \quad B_2 = (e_1 - e_2) \otimes e_1 \otimes e_1, \quad B_3 = e_2 \otimes (e_1 + e_2) \otimes (e_1 - e_2)
\]

, i.e \(G_3 = B_1 + B_2 + B_3\)

By using the SetTools program, we can see how the picture of \(G_3\) looks like:
$D_3$ is a representative tensor of a class of $3 \times 3 \times 3$ tensors in which each of them is rank 3 and can be approximated by rank-2 tensors.

$D_3$ is a combination of the three basepoints $I_1$, $I_2$, $I_3$ where

$I_1 = e_1 \otimes e_1 \otimes e_1$, \quad $I_2 = e_1 \otimes e_2 \otimes e_2$, \quad $I_3 = e_2 \otimes e_1 \otimes e_2$

, i.e $D_3 = I_1 + I_2 + I_3$

By using the SetTools program, we can see how the picture of $D_3$ looks like:
• We successfully made a tool that allows us to visualize the images of low rank tensor approximation.

• We had good pictures of $3 \times 3 \times 3$ tensors that support the theorem proposed by Vin De Silva and Lek-heng Lim.

• We are studying the pictures of tensors of higher rank to make some conjectures.

• We hope to draw some solid conclusions concerning the separated representation, now that the program is easy to use.

• We hope to add a help feature or user’s guide to the program.