2. Basis Vectors versus Orthogonal Vectors

2.1 Issue

When students are performing the standard pattern associator assignments involving the training sets basis8.net (the set of the 8 basis vectors for an 8-dimensional space) and ortho8.net (a set of 8 mutually orthogonal vectors in 8-dimensional space) they are often puzzled by the differences between the two. How can the two sets of patterns, whose numbers look dramatically different from one another, be very similar?

2.2 Task

We will use a graphing demonstration in a smaller space (only three dimensions) to show that geometrically a set of basis vectors is very similar to a set of mutually orthogonal vectors. We will also use this demonstration to show how one might construct a set of mutually orthogonal vectors by 1) taking a set of basis vectors and then 2) rotating these vectors randomly about the x, y, and z axes of a three dimensional space.

2.3 Materials

This activity involves using an R script that are loaded into the R statistical package. It will be performed on the computers in BSB P-116 because it is not assumed that students will have R on their own computers. The script is available from the course website for those who do have R on their own computer.

2.4 Procedure

2.4.1 On a computer in the classroom start the R programming language. Use the top menu to open a script, and load the script BasisVsOrtho.R into the R environment.

2.4.2 Examine the script. It defines three basis vectors. It assigns three different "degrees of rotation" about the x, y, and z axis of a three-dimensional space. It creates a transformation matrix that performs this desired rotation on each of the three basis vectors. It graphs the three basis vectors in black, and then graphs the three new vectors – created by rotating the basis vectors – in green. It then prints out the values of these different vectors to the R console. These general operations will be explained in class.

2.4.3 Using the mouse, click on the window that holds the script. Press <CTL>A to select the entire script. Press <CTL>R to execute the entire script.

2.4.4 After executing the script, a graph that shows three black vectors (the original) and three green vectors (the rotated) will appear. Click this graph with the mouse. You can use the mouse to spin the graph to look at it from different angles. *What are the similarities and differences between the two sets of three vectors?*

2.4.5 Close the graph. Scroll to the bottom of the R console, looking at the printouts of each basis vector and its transformed version. *What are the similarities and differences between the different pairs of vectors?*

2.4.6 Click on the script. Change the three angles (in degrees) to whatever angles you like. Then repeat steps 2.4.3, 2.4.4, and 2.4.5. *What are the similarities and differences between a set of basis vectors, and a set of mutually orthogonal vectors? How would one create a random set of mutually orthogonal vectors in eight-dimensional space?*

2.4.7 A short class discussion about basis8.net and ortho8.net would now be in order!

3. Geometry of Hebb Learning

3.1 Issue

When a distributed associative memory is trained using the Hebb rule, and makes mistakes, what is the nature of these mistakes? This activity attempts to provide insight to issues like this one by graphing network responses during learning. By observing this graph develop over time, we can develop some concrete intuitions about the successes and failures of Hebb learning.

3.2 Task

Students will use the statistical programming language R to read in a script that brings Hebb learning to life. They will run this script, modifying it at times, and observe Hebb learning in action. It is expected that there will be an ongoing class discussion about what is being observed, and about how to play with the scripts to try and answer specific questions that come to mind when observing the script in action.

3.3 Materials

This activity uses the R statistical programming language, as well as the script HebbGeometry.R. The script is available from Week 2 of the website. The activity will be conducted using the computers in BSB P-116, but if students want to install R on their own computers and use them instead that will be acceptable.

3.4 Procedure

3.4.1 On a computer in the classroom start the R programming language. Use the top menu to open a script, and load the script HebbGeometry.R into the R environment.

3.4.2 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from the first line until the comment box that begins "Learn three different associations each epoch". Press <CTL>R to execute the selected part of the script. Look at the output of the console – it provides information about the starting state of the network.

3.4.3 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from the comment box that begins "Learn three different associations each epoch" to the comment box that begins with "With space drawn, let us do some learning". Press <CTL>R to execute the selected part of the script. R will draw three vectors in the space; these are the three vectors that serve as correct answers during learning. Use the mouse to spin the graph around so that you can see these patterns from different angles.

3.4.4 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from comment box that begins "With space drawn, let us do some learning" until the end of the script. Press <CTL>R to execute the selected part of the script. Learning will commence; network responses will be graphed in colour. Watch how these responses change over time, and look at their relationship to the three correct responses drawn in black.

3.4.5 Repeat step 3.4.4, and watch how additional learning affects network responses. *Are the responses correct, or not? In what way are they correct? In what way are they incorrect?*

3.4.6 Easy edits can be made to the script to alter numerous properties, including:

- the patterns involved in learning
- the initial weights
- the learning rate
- the amount of training

In discussion with an instructor, determine how such changes can be made, make them, and observe the results. One can select the entire script by clicking in its window and then typing <CTL>A; typing <CTL>R will run the entire script. (A good idea would be to close graphics windows before running new scripts, unless you want to compare different outputs from different runs.)

4. Geometry of Delta Rule Learning

4.1 Issue

When a distributed associative memory is trained using the delta rule, what is the nature of this learning in comparison with Hebb learning? This activity attempts to provide insight to issues like this one by graphing network responses during learning. By observing this graph develop over time, we can acquire some concrete intuitions about the dynamics of delta rule learning.

4.2 Task

Students will use the statistical programming language R to read in a script that brings the delta rule to life. They will run this script, modifying it at times, and observe the delta rule in action. It is expected that there will be an ongoing class discussion about what is being observed, and about how to play with the scripts to try and answer specific questions that come to mind when observing the script in action.

4.3 Materials

This activity uses the R statistical programming language, as well as the script DeltaGeometry.R. The script is available from Week 2 of the website. The activity will be conducted using the computers in BSB P-116, but if students want to install R on their own computers and use them instead that will be acceptable.

4.4 Procedure

4.4.1 On a computer in the classroom start the R programming language. Use the top menu to open a script, and load the script DeltaGeometry.R into the R environment.

4.4.2 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from the first line until the comment box that begins "Learn three different associations each epoch". Press <CTL>R to execute the selected part of the script. Look at the output of the console – it provides information about the starting state of the network.

4.4.3 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from the comment box that begins "Learn three different associations each epoch" to the comment box that begins with "With space drawn, let us do some learning". Press <CTL>R to execute the selected part of the script. R will draw three vectors in the space; these are the three vectors that serve as correct answers during learning. Use the mouse to spin the graph around so that you can see these patterns from different angles.

4.4.4 Using the mouse, click on the window that holds the script. Use the mouse to select all of the lines of the script from comment box that begins "With space drawn, let us do some learning" until the end of the script. Press <CTL>R to execute the selected part of the script. Learning will commence; network responses will be graphed in colour. Watch how these responses change over time, and look at their relationship to the three correct responses drawn in black. *How do they compare to what you saw in the previous exercise?*

4.4.5 Repeat step 4.4.4, and watch how additional learning affects network responses. *Are the responses correct, or not? In what way are they correct? In what way are they incorrect? How do they compare to Hebb rule responses?*

4.4.6 Easy edits can be made to the script to alter numerous properties, including:

- the patterns involved in learning
- the initial weights
- the learning rate
- the amount of training

In discussion with an instructor, determine how such changes can be made, make them, and observe the results. One can select the entire script by clicking in its window and then typing <CTL>A; typing <CTL>R will run the entire script. (A good idea would be to close graphics windows before running new scripts, unless you want to compare different outputs from different runs.)