**Guide to using SimilarityMatrix and MatrixComparison in MATLAB**

SimilarityMatrix and MatrixComparison are two MATLAB classes to perform calculations related to finding the similarity between nodes in neural networks based on their connectivity. These classes are constructed using an underlying AdjacencyMatrix object which describes the network being analyzed. SimilarityMatrix will compare nodes to other nodes within the same network, while MatrixComparison will compare nodes between two networks. Both classes were designed to allow for polymorphism: these classes implement various methods for calculating similarity. SimilarityMatrix implements the following methods:

* Cosine Similarity
* Algorithm of Blondel et al. from SIAM Review, 2004
* Weight Similarity (the method that I created)
* Tanimoto Similarity (Similar to Cosine Similarity, but using the Tanimoto coefficient instead of the cosine)

The MatrixComparison class currently only implements the Weight Similarity algorithm, but I can implement the other methods if they would be useful.

**SimilarityMatrix:**

Adam Bloniarz September, 2010

1. **Creating a SimilarityMatrix object**
   1. **Specifying the underlying AdjacencyMatrix object and similarity algorithm**

To create a SimilarityMatrix object, use the SimilarityMatrix constructor, which acts on an Adjacency Matrix object (which is made first). This constructor takes several parameters, which specify the underlying AdjacencyMatrix object, the algorithm to use, grouping of nodes, and direction of connection to consider. The simplest way to construct a SimilarityMatrix object is to simply specify the AdjacencyMatrix object and the similarity algorithm. Thus, for example, if you want to calculate similarity of nodes in the network described by the AdjacencyMatrix object ‘adjMat’ using cosine similarity, enter the following command:

>> simMat = SimilarityMatrix(adjMat, ‘similarityMethod’, ‘cosine’);

To use another algorithm, simply replace ‘cosine’ with one of the following:

* ‘blondel’
* ‘tanimoto’
* ‘weightSimilarity’
  1. **Specifying node groupings**

When using cosine similarity, weight similarity, and Tanimoto similarity, you can group together nodes during similarity calculation. This can be used to allow for L/R homologs to be considered equivalent outputs or inputs to a neuron, for example. Before instantiating a SimilarityMatrix object, create the groupings as a cell array of cell arrays, alternating group names with group members (just as in making groups in AdjacencyMatrix objects). For example, to group together the LUA neurons, enter the following command:

>> groupLUAL = {‘LUA’, {‘LUAL’, ‘LUAR’}};

Then if you provide this to the constructor of SimilarityMatrix, the similarities will be computed, considering LUAL and LUAR identical:

>> simMat = SimilarityMatrix(adjMat, ... ‘similarityMethod’,’tanimoto’,’groups’,groupLUAL);

* 1. **Specifying the direction of connections to consider**

You can restrict similarity calculations to consider only outgoing or incoming connections. The default is to consider both directions. This should only be specified when using the cosine similarity, weight similarity, or Tanimoto similarity algorithms (restricting directionality doesn’t really make sense when using the Blondel algorithm).

The SimilarityMatrix constructor takes the parameter ‘direction’ to specify directionality. To only consider outgoing connections when constructing the SimialrityMatrix using Tanimoto similarity, for example, enter the command:

>> simMat = SimilarityMatrix(adjMat, ...

‘similarityMethod’, ‘tanimoto’, ‘direction’, ...

‘outgoing’);

1. **Displaying Similarity Values**
   1. **Displaying Similarities of pairs of neurons**

Individual values from a SimilarityMatrix object can be retrieved just like you would retrieve values from a normal Matlab matrix. Thus, for example, if you are interested in seeing the similarity between CP07 and CP08 in the SimilarityMatrix object ‘simMat’, enter the command:

>> simMat(‘CP07’,’CP08’);

This will display the value.

* 1. **Displaying a list of the most similar pairs in a SimilarityMatrix object**

To display a list of the most similar pairs in a SimilarityMatrix object, use the function ‘showMostSimilarPairs’. You must enter the number of pairs you would like to see as the second argument to the function. Thus, for example, if you would like to see the 50 most similar pairs in the SimilarityMatrix object chemSim, enter the command:

>> showMostSimilarPairs(chemSim,50)

* 1. **Displaying the similarities of L/R pairs in a SimilarityMatrix object**

To display the similarities of all L/R pairs within the AdjacencyMatrix being analyzed by a SimilarityMatrix object, use the command LRSimilarities. This command will also display a histogram of the L/R similarity values (I can get rid of this feature if you would like). Thus, to see the L/R similarities in the SimilarityMatrix object chemSim, enter the command:

>> LRSimilarities(chemSim);

This function can handle L/R pairs where one neuron name ends in ‘L’ and the other ends in ‘R’. It can also handle muscle L/R pairs like ‘vBWML23’ and ‘vBWMR23’. Other kinds of L/R pairs may not be displayed, and would have to be accessed individually.

* 1. **Displaying a list of neurons most similar to a given neuron**

If you would like to see a list of the neurons which have the highest similarity to a given neuron, use the function showSimilarNeurons. You must provide the name of the neuron you are interested in as well as the number of results you would like to see as arguments. Thus, for example, if you would like a list of the 20 neurons most similar to PVX according to the SimilarityMatrix object chemSim, enter the command:

>> showSimilarNeurons(chemSim,’PVX’,20);

This will display a list of neurons and their similarity to PVX.

* 1. **Displaying the histogram of similarity values**

To display a histogram of similarity values for all pairs of neurons, use the command showHistogram. The first argument should be the SimilarityMatrix object, and the second argument should be the number of bins to use in the histogram. Thus, for example:

>> showHistogram(chemSim,50)

will show a histogram of all similarity values from chemSim, grouped into 50 different bins.

**MatrixComparison:**

1. **Creating a MatrixComparison object**

A MatrixComparison object can compute similarity values between nodes in different networks. Thus, it requires 2 AdjacencyMatrix objects as arguments to the constructor. Currently the MatrixComparison class only implements the weightSimilarity algorithm in computing similarity values. If you would like me to add the cosine, Tanimoto, and Blondel similarity algorithms, let me know.

For example, if you have 2 AdjacencyMatrix objects, called N2Ychem and JSE chem, you can create a MatrixComparison object as following:

>> N2YtoJSE = MatrixComparison(N2Ychem, JSEchem);

Similar to SimilarityMatrix, you can specify groups when constructing a MatrixComparison object. If ‘LRgroups’ is a cell array describing LR groupings, for example, enter the command:

>> N2YtoJSE = MatrixComparison(N2Ychem, JSEchem, ‘groups’,LRgroups);

This will lump together L/R pairs (or whatever groups are described in LRgroups) when calculating similarities.

1. **Displaying similarity values**
   1. **Displaying the similarity of a pair of neurons**

Individual values from a MatrixComparison object can be retrieved just like you would retrieve values from a normal Matlab matrix. The first index gives the neuron name in the AdjacencyMatrix that was provided first to the MatrixComparison constructor, and the second index gives the name in the second AdjacencyMatrix object. Thus, for example, if you are interested in seeing the similarity between CP07 in N2Ychem and CP07 in JSEchem in the MatrixComparison object created above, enter the command

>> N2YtoJSE(‘CP07’,’CP07’);

This will display the value.

* 1. **Displaying a list of the most similar pairs in a MatrixComparison object**

To display a list of the most similar pairs within a MatrixComparison object, use the function showMostSimilarPairs. Again, this function takes an argument specifying the number of pairs you would like to see:

>> showMostSimilarPairs(N2YtoJSE,20)

* 1. **Displaying a list of similarity values between the same neurons in different networks**

To display a list of the similarity values between neurons with the same name in two different networks, use the function sameNodeSimilarity. Thus, for instance, if you would like to see how all the neurons in N2Y compare with their homologs in JSE, use the syntax:

>> sameNodeSimilarity(N2YtoJSE);

(this assumes you have already created the object N2YtoJSE as described above.)

* 1. **Displaying a histogram of similarity values**

To display a histogram of all similarity values within a MatrixComparison object, use the command showHistogram. This has the same syntax as above, in which you must input the number of bins:

>> showHistogram(N2YtoJSE, 50);

will make a histogram of all similarity values between pairs of neurons from N2Y to JSE, grouped into 50 bins.