Page 1:

Brain imaging data allows for distinguishing between the brains of patients with various brain disorders. Graph classification methodology uses edge weights to predict the classification of the disorder, as well as the number of nodes, overall structure, and selection of edges. This project aims to classify networks whose nodes are not interchangeable and are not singular i.e. more than one network is being analyzed.

Page 2:

Each patient is represented by their own network depending on the pathology of their condition. Each brain region is treated as a node and are registered to a common atlas. The focus is how do we predict the kind of network and why is that the methodology for making our prediction. This project codifies connectivity with functional connectivity, which is a measure of the association between a pair of locations in the brain with statistical inference. Works with any sample of weighted networks with labeled nodes. Measures dependence between different voxels.

Page 3:

Two data sets used: one for constructing the analytic software and the other for testing. Each data set is preprocessed and then ROIs are chosen, then an adjacency matrix made. Pearson correlation coefficient used to define connectivity, a marginal correlation. Then the matrix is standardized to account for subject to subject variability and global signal regression.

Page 4:

Standardized marginal correlation performed better than regular marginal correlation. From a computational standpoint, it is not possible to use discriminative patterns in graphs as features for training a classification method for any kind of data set besides small binary networks. Other methods stem from graph kernels, which define a similarity measure between two networks. These are used with support vector machines (SVMs) on small networks and achieve positive results. Large scale networks cannot be analyzed with these methods due to dimensionality once again.

Page 5:

In the data sets used, graph kernel methods performed no better than random guessing. In terms of classifying large scale brain networks, there are two primary ways of doing it. One utilizes graph theory parameters, including average degree, weighted degree, clustering coefficient, average path length. Find all of these for the network and use them to train a classification method. These have shown promise, but are not as valued due to their lack of attention to the local characteristics of a network. The other approach is to go the complete opposite route, focusing entirely on local edges. Treat each edge weight as a component of a vector for each node. This allows for vector classification methods to be used and for edge level interpretation. What defines all the nodes in the data set determines how effective the correlation between these nodes is at classification. Each edge can also be tested in two different data sets, and the overall results can be used to describe the differences between the two populations. This does not account for the overall structure of the network nor does it provide interpretation. It only quantifies the presence of differences and interpretation of the differences in edges. What is desired is analysis of nodes as well. The network structure is considered more so in grouping of connective edges. One method conducts univariate testing at each edge, then counts the number of connections out of each cell, where a cell is a group of edges where all the edges connect two nodes in different functional systems (in this case ROIs). Then statistical inference is run on each cell. Power can be improved with network-based multiple dependence testing correction.

Page 6:

From a classification standpoint, better interpretability and possibly accuracy can be obtained if we focus on which brain regions, or interactions between them, are responsible for the differences between the data sets. Other methods exist to look at individual nodes and edges, however those are only computationally feasible for small networks. Goal is to create a classifier that uses all individual edge weights yet looks at the entire network structure as well and provides interpretable results. All graphs can be represented by N nodes and an n by n adjacency matrix. Graphs are undirected, thus all of the adjacency matrices are symmetric. There are no self-loops, thus the main diagonal is all zeros, making the only eigenvalue zero. The problem has been framed into a binary classification problem, thus the class label of the graph is either -1 or 1.

Page 7:

Entry-wise norm just treats the matrix as a giant n2-dimensional vector and computes standard magnitude. Linear classifier creates a matrix B with coefficients as entries that is transposed and multiplied by the adjacency A and the diagonal of the result is summed. Coefficients that are in B are estimated from data by minimizing the sum of a loss function and a penalty. Edges are organized into subnetworks called brain systems, which have specific functionality. The goal is to find nodes or any of these subnetworks that have good discriminative power. The main focus is medium to large network method that allow for efficient and scalable implementations that also have convergence guarantee.

Page 8:

A typical linear classifier includes the B-matrix of coefficients and the loss function. The loss function can be defined as any function, including those beyond classification such as least squares or generalized linear models. Logistic loss function is used in this paper, which includes a threshold b that needs to be estimated. Convex structured sparsity penalties encourage a small number of active nodes. Spatial smoothness penalties were not utilized, however due to the flexibility of the algorithm those can be added in.

Page 9:

The penalty algorithm that implements sparsity ensures that an edge would only be selected if both nodes it is attached to are active. Requiring symmetry in the matrix B yields more accurate classifiers.

Page 10:

The analogue for directed graphs is far more complex, as each edge now has a backwards and forwards component. Every edge would have to be treated as a vector with two components and the magnitude of the entire thing is the magnitude of the entries squared, again treating as a large vector with n squared entries.