Brian Randall & Arjan Puniani
ECE 2556
10/28/2021

Project 1 Report

(1) Method Description

We want to use network features to predict the statuses to which a patient belongs.

Using 18 selected BCT methods, we extracted global features from some of them and local features from others. The local features are averaged across the nodes to become a single feature.

GLOBAL features		LOCAL features
Assortativity		Betweenness
Charpath	Lambda (network characteristic path length)	Clustering coefficient
	Network Efficiency	Community Louvain
	Nodal eccentricity	Core Periphery (local version)
	Network radius	Degrees (# of links connecting to each node)
	Network diameter	Modularity
Density		Strength
Efficiency		Eigenvector Centrality
Transitivity		
Core Periphery (global)		

We use 10-fold cross-validation (85% for training and 15% for testing) to classify each patient (observation) into Status 1 vs. 2, Status 2 vs. 3, and Status 3 vs. 4

However, the datasets are imbalanced, i.e., there are more patients with Status 2 than statuses 1 and 3:

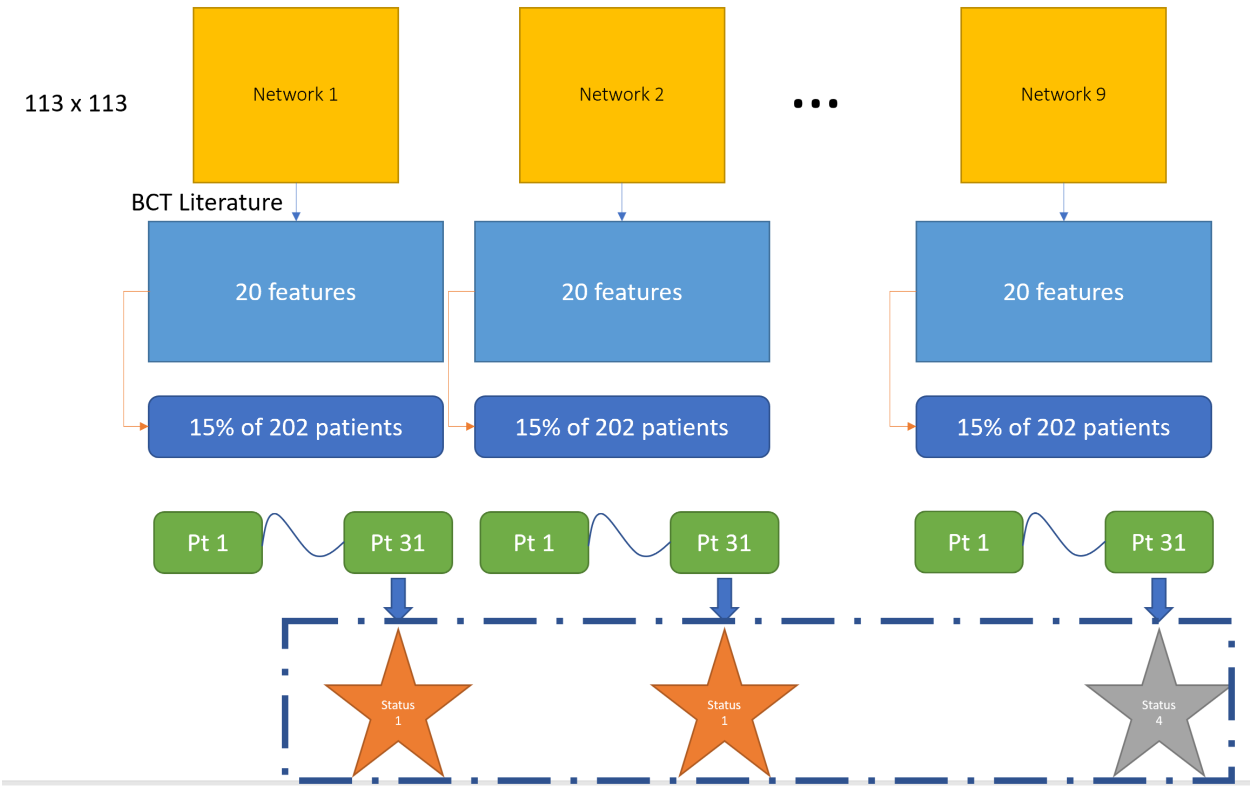
```
Status: 202×1 categorical

    Values:

        Status 1       51
        Status 2       73
        Status 3       39
        Status 4       39
```

To accommodate unbalanced datasets, we re-sampled the smaller dataset. The method randomly draws observations from the smaller dataset and grows it with those drawn observations until the smaller set matches the length of the larger.

Our method is known as Consensus and is based on a conversation we had with Prof. Zhan:



We partitioned the observations (patients) into training (85%) and testing (15%) sets. Our algorithm applies a Gaussian SVM classifier to the patients assigned to the testing set for each network. Then, a "consensus" is taken of the 9 Gaussian SVM classifiers. Because there are an odd number of networks, and therefore classifiers, there will be no ties, so a winner must be found.

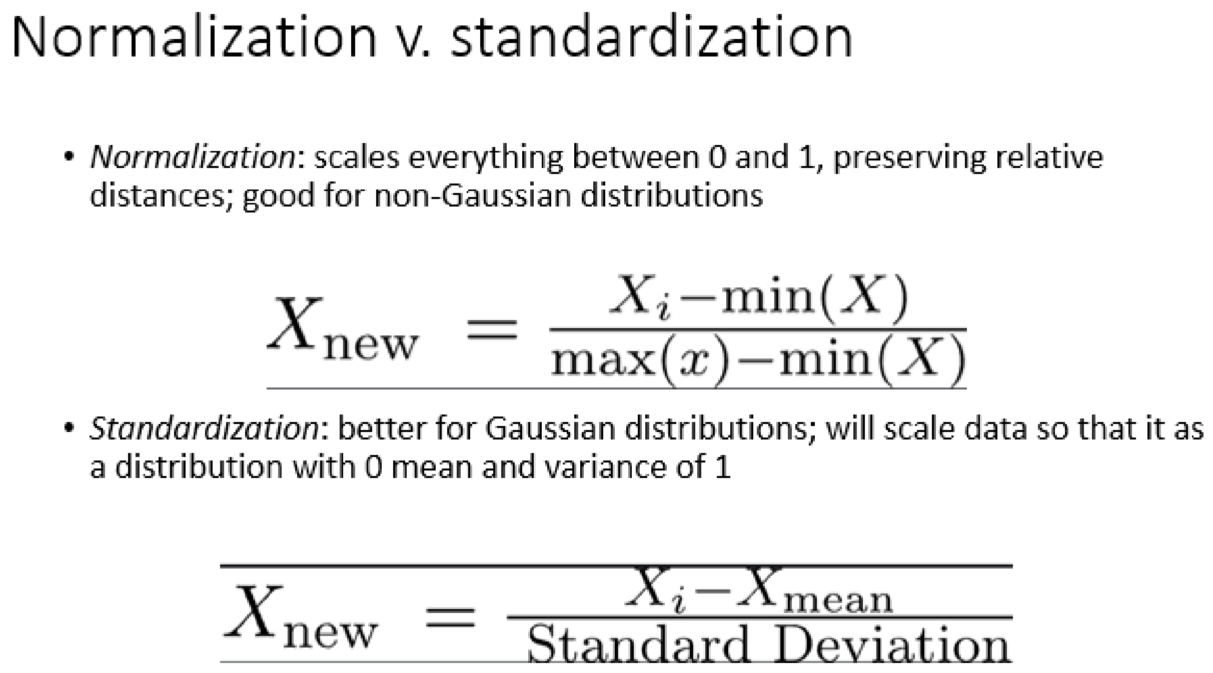
We also ran the consensus algorithm as a reduced data set: omitting GENDER and AGE from the features matrix. We call this the reduced-data model and found that our classification accuracies did not improve, absolutely or significantly, with a reduced features matrix.

(2) Baseline Method Description

We initially looked at Naïve Bayes, binary trees, and K-nearest neighbors to compare against our support vector machines (SVM) algorithm for accuracy. However, we decided to compare the SVM consensus algorithm against the "averaged" network features.

Our baseline for comparison uses what we call the mean network model. Essentially, we collapse the local and global features of the nine brain graph networks into a mean model, m. In other words, the elements-wise average of the subject's nine brain graph network models is that subject's "mean model," m.

Depending on if the distribution appeared Gaussian, we would normalize or standardize the extracted features from the patient network.



Normalization v. standardization

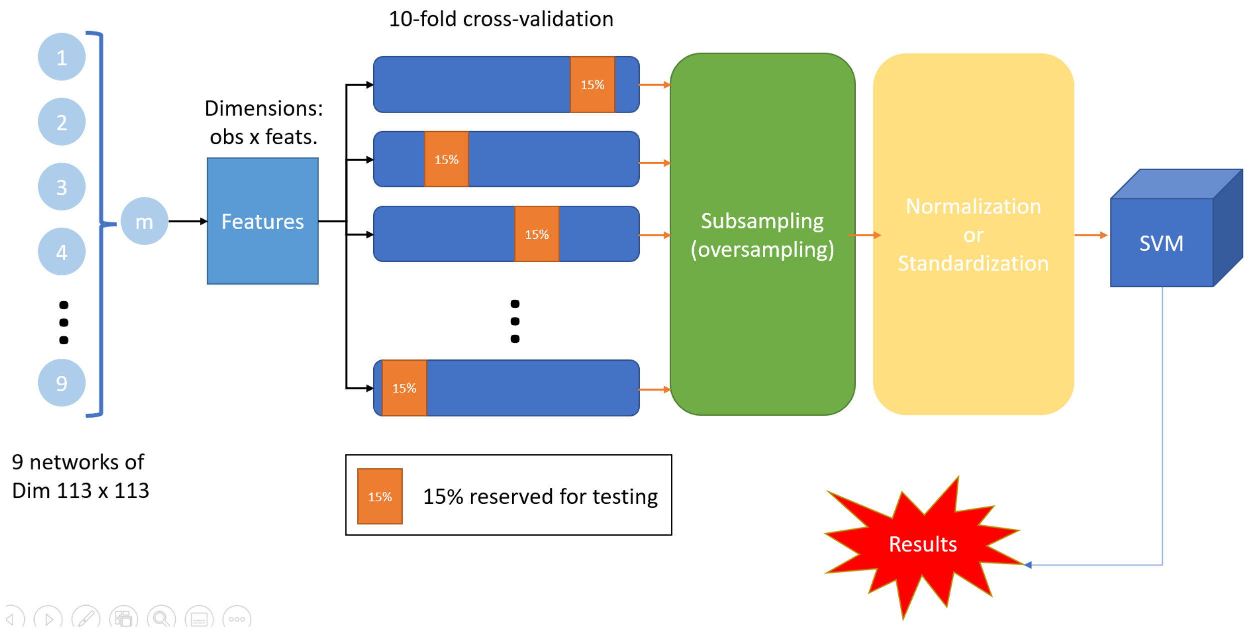
- *Normalization*: scales everything between 0 and 1, preserving relative distances; good for non-Gaussian distributions

$$X_{\text{new}} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

- *Standardization*: better for Gaussian distributions; will scale data so that it as a distribution with 0 mean and variance of 1

$$X_{\text{new}} = \frac{X_i - X_{\text{mean}}}{\text{Standard Deviation}}$$

The entire process is summarized here:



(3) Results

In this section, we discuss the results of testing the above methods.

Accuracy Results			
	Average Accuracy	TPR	FPR
Mean 1v2	0.7045	0.4583	1.0000
Mean 2v3	0.6682	1.0000	0.0000
Mean 3v4	0.5750	0.2759	0.7049
Consensus 1v2	0.7628	1.0000	0.5739
Consensus 2v3	0.7773	0.0000	0.9942
Consensus 3v4	0.4083	0.2759	0.5645
Reduced 1v2	0.7500	0.9910	0.5229
Reduced 2v3	0.7545	0.0000	0.9759
Reduced 3v4	0.4750	0.3167	0.5167

Our algorithm obtained a higher average accuracy than the baseline model and the reduced-data model in the 1v2 and 2v3 classification tasks. However, the results for the 3v4 classification show that the consensus model performs worse than the baseline model and reduced-data model, although all the models perform poorly in this task.

Observing the TPR and FPR results, there are many 100%'s or 0%'s. This may indicate that the model is heavily bias towards a certain outcome, as it assigns most of the testing data to the same class. All the 2v3 classification models have a TPR/FPR split like this.

A possible conclusion from this is that the classes in 2v3 classification are very similar, and difficult for our model to distinguish.

T-Test Results

	P-value
Mean 1v2 vs. Consensus 1v2	0.0997
Mean 2v3 vs. Consensus 2v3	0.0181
Mean 3v4 vs. Consensus 3v4	0.9234
Reduced 1v2 vs. Consensus 1v2	0.6476
Reduced 2v3 vs. Consensus 2v3	0.6474
Reduced 3v4 vs. Consensus 3v4	0.2100

The t-test results show that the difference between the consensus model and the mean model is significant for tasks 1v2 and 2v3, and not significant for all other comparisons. This means that our model did improve on the baseline for two out of the three tasks. It also means that the data we removed from the dataset in the reduced-data model was not important in classification.

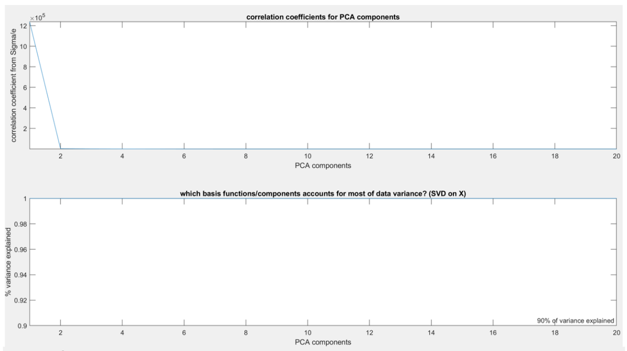
Appendix

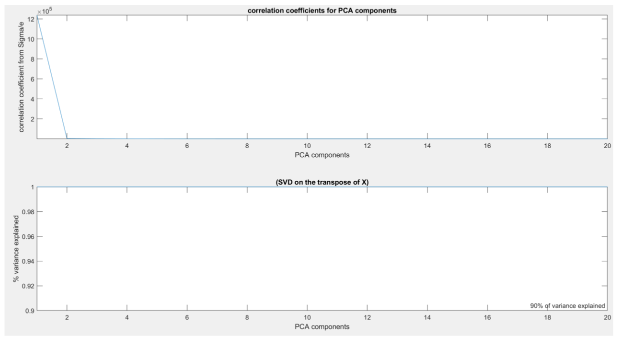
Challenges

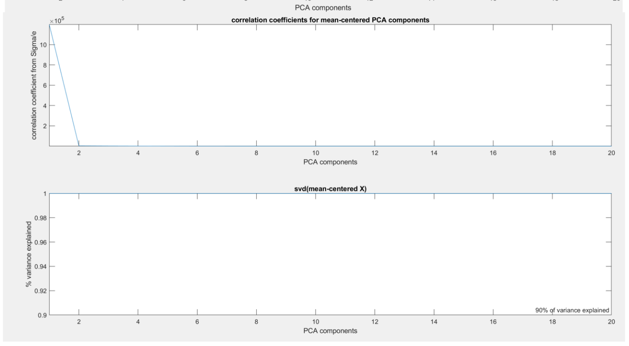
How do we reconcile local features that are particular to a patient's specific indexing? We "depersonalize" the 113 x 113 network matrices by applying a distance/shortest path metric to each network. This removes any patient-specific bias from the nature of the indices.

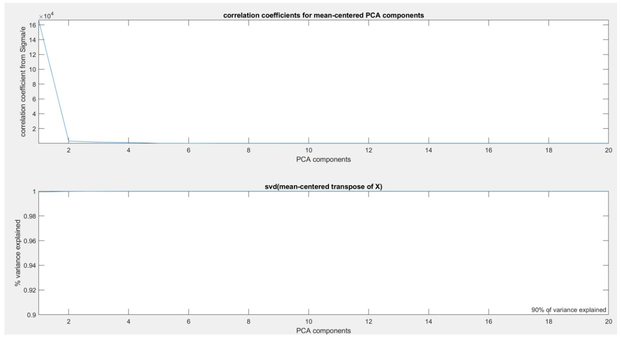
Features need to be standardized or normalized, depending on how Gaussian it looks. We wrote our own function before finding MATLAB's `normalization` function and option to standardize with a 'zscore' flag.

Principal Component Analysis in our first pass did not yield eigenvectors that explained the variance distributed across multiple columns. We plotted the principal components against the % variance explained and found the first eigenvector tended to explain 99% of the data variance.









We shifted our attention to `fitclinear` to examine the most important predictor variables based on the magnitude of the beta coefficients. When setting the learner to "logistical regression," we can save a lot of time needing to find likelihood functions by hand. However, we didn't have a reason to prefer one component over another unless the difference was 99% , which it never was:

```
>> Mdlslr.Beta
ans =
    0.1274
    0.2898
    0.4201
    0.7166
    0.1870
   -0.0212
    0.1176
   -0.3248
    0.2901
    0.0900
   -0.3410
   -0.0651
    0.2568
    0.1952
    0.1952
   -0.3964
   -0.4778
    0.3244
    0.7558
    0.0696
```