Calvin Quach
CSCI 493.71
Professor Di He
25 May 2018

# Project 2

## Part 1

The idea behind feature clustering is find some underlying labels that group the genes into clusters. By grouping the genes into clusters we can reduce the dimension of our training data.

**1.** Transforming ROSMAP for Discovering Clusters

In order to find the clusters of our features we would have transform our ROSMAP data into samples to train our clustering model. To do this, I would have created a gene matrix. The feature vector would be the mean, variance, max and min of each gene. The rows would be the different genes.

**2.** Train with KMeans to find cluster centers

Once the data has been transformed, plug the data into KMeansModel. The number of clusters to be generated will be tunable hyperparameter. Ideally, it will be a small number. The output will be a set of K centers.

**3.** Mapping Genes to K Clusters

We then use the KMeans model to predict the labels of the gene ids. This will assign the cluster membership to each of the genes. The RDD of this operation will be <e_id, cluster_id>. Alternatively, if the cluster membership of each entrez id was provided, the following operation below will separate these ids to to their clusters.

$$f :< clusterId : \ [eID_1, \ eID_2, .....] > \ \Rightarrow \ (< eID_1, clusterID >, \ < eID_2, clusterID >)$$

**4.** Update the Training Samples

Now we have to update the training sample to use the clusters to represent them. This requires use to generate a mapping that will map each sample to multiple values. The **mapping** operation will emit <e_id, [patient id, diagnosis, value]> for each patient.

$$f :< patientID, diagnosis, \ (eID_1, val), (eID_2, val), \ ... > \ \Rightarrow \ (< eID_1, \ [patientID, \ diagnosis, \ val] >, ....)$$

**5.** Now we will perform a **join** operation between these two RDDs on the gene ID. This combine the information of cluster membership and gene values.

$$f : (< eID_1, [patientID, diag, val] >, ...) \cup (< eID_1, clusterID >) \ \Rightarrow$$
$$< (C1, \ patientID, diag), (val) >$$

**6.** Now we perform a **groupByKey** using the (C1 and PatientID) tuple as the key. This will group together all the gene ids into their clusters

$$f : (< C1, patientID, diag), (val1) >, < (C1, patientID, diag), (val2) >, ....) \ \Rightarrow \ < (C1, patientID, diag), (val1, val2) >$$

7. Then we **reduce** the groups to get the sum. To go further, you may replace this with some other statistics

$$f : (< (C1, patientID, diag), (val1, val2, ...) >, ...) \ \Rightarrow \ < patientID, \ (C1, \ diag, val2 \ + \ val3) >$$

8. Finally we want to combine the clusters together, using **groupByKey**(patientID) This will piece together the sample.

$$f : (< patientID,\ (C1,\ diag, val) >, < patientID,\ (C2, diag, val >, ...) \ \Rightarrow\ < patientID,\ [diag, (C1, val),\ (C2, val), ...] >$$

9. We have the option to perform a T significance test. The purpose of doing so is to further reduce our dimension. First we need to separate our data into two sets. One for AD and the other for NCI. We can complete this using **RDD filter operation.**

10. To calculate the t score we need the mean and the variances of the two sets. We need to **map and emit** the cluster values into their own pairs.

$$f :< patientID,\ [diag, (C1, val),\ (C2, val), ...] > \ \Rightarrow\ (< C1,\ (val, 1) >,\ < C1,\ (val, 1) >,\ < C2, (val, 1) >,\ < C2, (val, 1) >,\ ...)$$

11. To sum up the values and retrieve the counts, use **a reduce**

$$f : (< C1,\ val >,\ < C1,\ val >,\ < C2, val >,\ < C2, val >,\ ...) \ \Rightarrow\ (< C1,\ (val + val, 1 + 1) >, < C2,\ (val + val, 1 + 1) >, ...)$$

12. Just a **mapper** to calculate the average.

$$f : (< C1,\ (val, 2) >, < C2,\ (val, 2) >, ...) \ \Rightarrow\ (< C1,\ val/2 >,\ < C2,\ val/2 >)$$

13. Don't know how to calculate the Std without redundant work….. but I am going to assume that we have the pairs below.

$$The\ AD\ set\ (< C1,\ avg,\ std, count >, ...)$$
$$The\ NCI\ set\ (< C1,\ avg,\ std, count >, ...)$$

14. To calculate the T score we first we need to perform a **join** to combine the two sets. Then we perform a **groupByKey(cluserId).** Followed by a **reduce**. The **reduce operation performs the t score calculation**

$$f :\ (< C1,\ avg,\ std, count >, ...)\ AND\ (< C1, avg, std, count >) \Rightarrow$$
$$f :\ (< C1,\ avg,\ std, count >, < C1, avg, std, count >, .....) \Rightarrow$$
$$(< C1, [(avg1,\ std1), (avg2,\ std2] >) \Rightarrow$$
$$(< C1, t - score >)$$

15. Now we have the t-score for each cluster. Now to select the top K clusters, we can **sort** these clusters by their t-score and select the topK t scores.

# Part 2

## Introduction

In this project I will be generating a classifier for diagnosing patients with Alzheimer's disease. There are two classes, either the patient has Alzheimer's Disease (AD) or they have No Cognitive Impairment (NCI). Since there are two labels in this scenario, we are addressing a binary classification problem.

Before heading straight my approach in generating a classifier, I first want to discuss about the data that was provided. The data that I will be using to train and evaluate my model is the ROSMAP_RNASeq_entrez.csv. This file contains information about patient. More specifically they contain Patient ID, Diagnosis, and 16380

additional features. These 16380 features are continuous variables that represent the gene expression values for the patient. The diagnosis are a discrete set of labels from 1 - 6. This dataset contains patients who were diagnosed NCI, AD, and MCI (Mild Cognitive Impairment), Other Cognitive Impairments (OCI) and no answers (NA). For the diagnosis label in the data, 1 denotes for patients with NCI, 2-3 means the patients have MCI, 4-5 means the patient has AD and 6 means that the patient has OCI. There are a total of 568 instances in the ROSMAP dataset. The patient id are an alphanumeric identifier for the patient. Another file provided for this problem is the patients.csv. This dataset contains additional patient information, their age, gender and education and patient id.

The instances in the ROSMAP_RNASeq_entrez dataset have a large dimension. This emphasizes the problem of the curse of dimensionality. Since our training instances are represented by a large feature vector, it will definitely impact what the training model that will be outputted. It may cause the model not to fit the training data well. And since we are dealing with higher dimensions the sample space has increased. This means that the number of instances we have needs to increase because it may not represent that population well. Some other prior analysis that may further our understanding of the problem is the class distribution of the dataset.

## Method

I will be training a Support Vector Machine model to diagnose patients. The input vector for this model will be the gene expressions for the patient. And the output will be either a 0 or 1. 0 represents that patient is diagnosed No Cognitive Impairment and 1 represents that the patient is diagnosed with Alzheimer's Disease. The tools I will be using are Spark for Python to train the model and perform transformations on the data.

Before training the model, I have to perform some transformations because at its current state, there is some information that I have to remove. Since the ROSMAP dataset contains patient information on those diagnosed with MCI or OCI, I will have to remove those samples from the dataset. I first load the csv into the Spark Context. This creates an RDD which I use to filter out instances with MCI, OCI and those with NA for their diagnosis. Once that is completed I had to map each sample to a Label Points. Labeled Points is a data type that associated a feature vector to a label. In our case it will pair the diagnosis label to the gene expression list. I also have to remove the patient ids. The reason being that it provides no benefit to training the model.

I will be using 3-Fold Cross Validation to evaluate my training model. This means that I will be running three tests and splitting the dataset into three equally sized sets. After the three test have been completed I will average out the metrics. At each test I will be training the model with two of the three sets and evaluating its performance with the third set. This is risky because the number of samples we currently have is low because our dimensions are very high.

## Analysis

| Fold # | 1 | 2 | 3 | Average |
|---|---|---|---|---|
| Area Under PR | 0.55558408 | 0.55072463 | 0.55102040 | 0.55086416 |
| Area Under ROC | .492424242 | 0.5 | 0.5 | 0.49747474 |
| Accuracy | 0.55084741 | 0.55072463 | 0.55102040 | 0.55086416 |

The Area Under the Precision Recall Curve (AUPRC) tells us how well a classifier is performing. It measure the area under the precision and recall rates. Having a higher AUPRC means that there is a high true positive rate and lower false positive and false negatives rates from our classifier . The Area Under the ROC (AUROC) measures the area between the true positive rates and false positive rates. Having a higher area means that our true positive rate is higher than our false positive rate.

From our cross fold evaluation, the AUPRC is slightly above .5 and the AUROC is slightly below .5. Our accuracy is also slightly above .5. An accuracy higher than .5 implies that our classifier is better than a classifier that randomly assigns labels.

### Retrospective

Some possible ways to improve the classifier is to address the problem with our datasets dimensions. Since our instances contained 1680 features and the number of instances we had were only 580 (even less, when filtering out NA,MCI, and OCI), by reducing our dimension size there may be a significant improvement on our models performance. Some methods to reduce dimensionality of our dataset would be to perform feature clustering and PCA. Once we have performed these transformations on the dataset we can perform student T test to see if there were any significant differences in their performance metrics.

Aside from transforming our data, another approach would be to select more models and compare them to the SVM used for this project. We can also tune the SVM parameters to see when hyperparameter settings would produce a better model. I did briefly test out increasing the number of iterations for the SVM and there was a significant decrease in accuracy. My guess would be that the model was overfitting the data. But I will not include much about this in this report because no significance test was performed.

A third option that may improve our classifiers performance would be to incorporate more instances for training. Since the dimension of our instances were higher than the number of samples, more AD/NCI samples would help the training.

## To Run

To run this project:

1. You first need to have python 3.6 installed, virtualenvwrapper and make (optional)
2. mkvirtualenv ad_classifier
3. workon ad_classifier (if not activated)
4. pip install -r requirements.txt
5. make train (OPTION 1)
    a. This will call the training script. It will generate the output model into the models directory. The model will be saved under the name **test**
    b. It will also display the test results
6. python trainModel.py <output: model dirname>
    a. This will call the training script. It will generate the output model into the models directory. The model will be saved under the **name you provided**
    b. It will also display the test results