Assignment Goals

- Use mutual information to reconstruct gene expression networks
- Control for multiple hypothesis testing with *q*-values
- Gain experience working with human SNPs and variant prioritization strategies
- Understand the relationship between convolutional layers and traditional hidden layers in neural networks

Part 1: Mutual Information in Regulatory Networks

In class, we saw how FIRE uses mutual information to detect relationships between sequence motifs and gene expression levels. Mutual information is also a popular technique for reconstructing transcriptional regulatory networks from gene expression datasets[1]. After measuring gene expression levels for all genes in a sufficient number of biological conditions, mutual information can detect some types of pairwise dependencies that may suggest one gene is a regulator and another is its target. Fluctuations in the regulator's expression can influence the expression levels of the target gene. We can create an undirected gene-gene network by computing mutual information for all pairs of genes and applying a threshold.

For this assignment, we will use the DREAM3[2] network inference challenge dataset. The file **`data.txt`** has a 21 time point simulation of the gene expression levels for ten genes over four trials (simulated replicates). The first line of the file shows the labels of the dataset columns, where each column, other than the first one, represents the expression level of a particular gene during different time courses. The data for each trial is separated by an empty line. The file is in a tab-delimited format.

1A: Mutual Information Implementation

Write a program **`CalcMI`** that takes as input the expression data for a set of genes over a number of trials and reconstructs pairwise gene-gene dependencies using mutual information. The program will output the list of gene-gene dependencies and annotate those that pass a specified mutual information threshold as positive edge predictions. It should only consider the dependencies between unique genes, not the mutual information of a gene and itself (the entropy of that gene).

A pseudocount of **`0.1`** should be used when creating the count matrix for the mutual information.

Your program should be callable from the command line as follows:

```
CalcMI <dataset> <bins> <th> <out> <tpn>
```

[1] http://link.springer.com/article/10.1186%2F1471-2105-7-S1-S7

[2] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0009202

where

- **`<dataset>`** is the name of the text file that contains the gene expression data formatted according to the description above. Because the file contains multiple trials, you need to average the expression level for each gene during each time course over the different trials. One would typically account for the variability in expression across the replicates, but our testing shows that using the mean expression level yields the best results on the DREAM3 data.
- **`<bins>`** is an integer number that represents the number of bins that should be used when calculating the mutual information. You should use a *uniform* binning of the gene expression range instead of equal density binning (equal number of points per bin) for this assignment. In practice equal density is often preferable, but uniform binning yields better results on the DREAM3 dataset.
- **`<th>`** is a decimal number that represents the mutual information threshold. A dependency is considered a *positive prediction* if its mutual information is greater than or equal to this value.
- **`<out>`** is the name of the text file into which the program will print *all* dependencies and their mutual information values in decreasing order. Each line in the file should contain the dependency (undirected gene-gene edge) and its mutual information value. Use a double dashed line (constructed using the equal '=' sign) to separate the positively predicted edges from the rest. The file should be formatted as follows:

```
(6,9)      0.81709597900947597583747816 21
(3,10)     0.63256891833957902661133044 04

...        ...

(5,8)      0.47990469878356364667017051 31
========================================
(3,7)      0.46260625802055212777077571 15

...        ...

(8,9)      0.42654635911311815216642744 94
```

- **`<tpn>`** is the file into which the program will output the number of positive predictions.

***1B**: Varying Number of Bins*

Here you will use **`CalcMI`** to see how the number of bins affects the number of positive predictions. Fix the mutual information threshold to **`0.4`** and run **`CalcMI`** on each of the following number of bins: **`{3,5,7,9}`**. In your writeup, for each value of the bin parameter report the total number of positively predicted dependencies as well as the maximum mutual information in a table.

1C: Calculating TPR and FPR

In this part, you will calculate the True Positive Rate (TPR) and the False Positive Rate (FPR) for your predicted edges based on the known, gold standard interactions between the genes of the **`<dataset>`** provided in **1A**. The gold standard edges are tabulated in a file called **`net.txt`** as two columns. Each line represents an undirected edge between two different genes. Each gene is represented by its index (e.g., 'G1' is represented by 1). We will not model the direction of the regulation in this assignment, so the order of the genes in each line is not significant.

The formulas for calculating TPR and FPR are defined in class. The positive interactions in the gold standard are the edges listed in the file **`net.txt`**. The negative interactions in the gold standard network are all the undirected gene-gene interactions between pairs of unique genes that are not in the file **`net.txt`** (i.e., ignore self-edges).

Run **`CalcMI`** on the **`<dataset>`** provided in **1A** with a threshold of **`0.5`** and the number of bins equal to **`8`**. Use the output files to (manually) calculate TPR and FPR. In your writeup report the confusion matrix (number of true positives, false positives, true negatives, and false negatives) followed by the TPR and FPR.

Input files and sample scripts can be downloaded from
`https://www.biostat.wisc.edu/bmi776/hw/hw2_files.zip`

If you are using a language that is not compiled into machine code such as Java, then you should make a small script called **`CalcMI`** that accepts the command line arguments and invokes the appropriate source code and interpreter. Sample scripts are contained in the **`Hw2SampleScripts`** subdirectory.

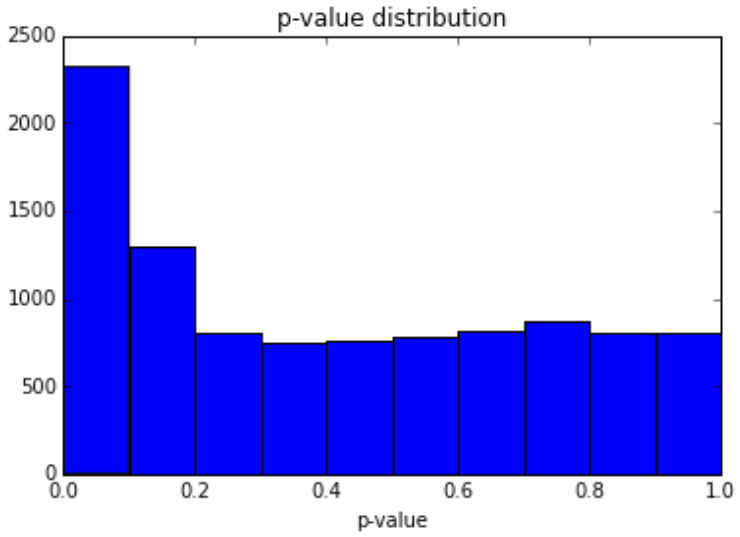
Recall that only standard libraries for the chosen language may be used. Third-party libraries must be approved by the instructor or TA on the Piazza forum.

Part 2: Calculating *q*-Values

We will manually calculate *q*-values from a *p*-value distribution.

2A: Estimating λ

First, use the histogram of the *p*-value distribution below to estimate λ visually as in Storey and Tibshirani 2003. The distribution contains *p*-values for 10000 features. Estimate λ to the nearest 0.1 and report the value you estimated.



***2B**: Estimating* $\hat{\pi}_0(\lambda)$

Use the table below to estimate $\hat{\pi}_0(\lambda)$ for the value of λ that you selected. Report the $\hat{\pi}_0$ you estimated.

λ	$\#\{p_i > \lambda; i = 1 \ldots m\}$
0.0	**10000**
0.1	**7680**
0.2	**6382**
0.3	**5582**
0.4	**4838**
0.5	**4074**
0.6	**3290**
0.7	**2474**
0.8	**1603**
0.9	**805**

***2C**: Calculating* q*-values*

Although 10000 features have been tested, only the top 10 features ranked by *p*-value are listed below:

Rank	p-value
1	**0.000024**
2	**0.000050**
3	**0.000091**
4	**0.000162**
5	**0.000217**
6	**0.000220**
7	**0.000221**
8	**0.000238**
9	**0.000248**
10	**0.000289**

Calculate the *q*-value for these 10 features. You may assume that the *q*-values for the remaining features not show in the list above do not affect the *q*-values of these 10 features.

Part 3: Prioritizing Noncoding Variants

We will use RegulomeDB and a trained DeepSEA classifier to prioritize a group of noncoding human SNPs on chromosome 1, one of which has been mechanistically linked to risk of myocardial infarction.

***3A**: RegulomeDB Scoring*

Use RegulomeDB (http://regulomedb.org/) to score the following SNPs:

- **`rs12740374`**
- **`rs599839`**
- **`rs14000`**

These are identifiers from the dbSNP database. Report the RegulomeDB Score for each SNP. Note that a score of 1a is the strongest evidence for functional significance and 6 is the weakest evidence.

***3B**: Variant Call File*

The RegulomeDB 'Summary of SNP Analysis' page provides links to the dbSNP database for each SNP. We will use the dbSNP entry to construct a minimal Variant Call Format (**`.vcf`**) file, which can be provided as input to DeepSEA. The **`.vcf`** file is a tab-separated text file with one line per SNP. DeepSEA requires the following five **`.vcf`** file columns:

```
Chromosome #    Coordinate    dbSNP id      Ref allele    Alt allele
```

An example is:

```
16              52599188      rs4784227     C             T
```

To obtain this information use the following fields from the dbSNP entry:

- *Chromosome #* – 'Chr' of the 'Integrated Maps' section.
- *Coordinate* – 'Chr Pos' of the 'Integrated Maps' section. These are 1-based coordinates. *Be sure to use the GRCh37.p13 genome assembly.*
- *dbSNP id* – already provided.
- *Ref allele* – 'Contig allele' of the 'Integrated Maps' section.
- *Alt allele* – the other (non-reference) allele reported in the 'RefSNP Alleles' field of the 'Allele' section.

Note that these simple rules for creating a **`.vcf`** file are not universally applicable to all variants but work for this special case.

Save your **`.vcf`** file for the three SNPs as **`hw2.vcf`** in your handin directory.

***3C**: DeepSEA Scoring*

Use the DeepSEA web server (http://deepsea.princeton.edu/) to score the **`.vcf`** file. Upload **`hw2.vcf`** and designate the file type as VCF. Report the DeepSEA functional significance score, which appears above the chromatin feature table, for each SNP in your writeup.

***3D**: Comparing Scores*

Are there any SNPs that are scored highly by both web servers? Are there any SNPs that are scored highly in only one of the web servers? For RegulomeDB, consider score categories 1a through 1f to be a high score. For DeepSEA, consider a functional significance score < 0.05 to be a high score.

For the top-ranked SNP in RegulomeDB, what features contribute to its high score? Use the RegulomeDB help page to see which types of supporting data (biological evidence) contribute to each score category.

For the top-ranked SNP in DeepSEA, what chromatin feature has the greatest effect size ($\log_2$ fold change) in the results table? Based on the P(Reference) and P(Alternative) columns in the results table, why would RegulomeDB not be able to detect this particular type of epigenetic evidence for prioritizing the SNP?

Part 4: Convolutional Neural Networks

***4A**: Convolutional Layer*

Consider the first convolutional layer in the DeepSEA network. How many parameters (weights) are needed in the convolutional layer when the input sequences are of length 1000, the window size is 8, and the hidden layer depth is 320? Include the w_0 parameters needed for the bias term at each node.

***4B**: Standard Hidden Layer*

Now suppose we replace the convolutional layer with a similar layer that is composed of standard hidden nodes (perceptrons). The number of hidden nodes is the same as the number in the convolutional layer version of the network. How many parameters are needed for the same input length, window size, and depth?

***4C**: Comparing Two Types of Layers*

What is one advantage of the network in **4A** relative to the network in **4B**? What assumptions are made in the network in **4A** but not in **4B**, and how could they be violated?

Submission Instructions

a) Login to the biostat server **`mi1.biostat.wisc.edu`** or **`mi2.biostat.wisc.edu`** using your BMI (biostat) username and password.

b) Copy any relevant files to the directory **`/u/medinfo/handin/bmi776/hw2/<USERNAME>`** where **`<USERNAME>`** is your BMI (biostat) username. For the programming part, make sure to submit the source code as well as any required files to run the program on the biostat server. For the rest of the assignment, if not otherwise instructed, compile your answers in a single file and submit as **`solution.pdf`**.