Problem Set 2

Issued: Monday, September 24, 2018 **Due:** Sunday, October 7, 2018

Part I - Spatial organization of the DNA

Problem 2.1: [10pts] Hi-C data analysis.

As discussed in class, it is becoming clear that the spatial organization of the DNA is crucial for making the different cell-type specific gene expression patterns possible. In this problem, we analyze the data collected by Rao et al (2014), in particular the Hi-C data for IMR90 (fibrobloast cells). The data for this problem is separated into tab-separated value files, `chrX_chrY.txt`, representing a sparse matrix of interactions between chromosome X and chromosome Y. The first column corresponds to a location on chromosome X (in base pairs). The second column corresponds to a location on chromosome Y (in base pairs). The third column gives an observed interaction frequency between these two regions, averaged over both copies of chromosomes. The resolution of this data is 250kb, so the locations are all multiples of 250k. (You should divide all of the locations by 250k to build your matrices.)

(a) For the purposes of this question, we will model $\log(1 + \text{interaction\_frequency})$ as a Gaussian random variable for each interaction site. Compute the mean and standard deviation of $\log(1 + \text{interaction\_frequency})$ across all inter-chromosome sites (i.e. don't include intra-chromosome interaction matrices like `chr7_chr7.txt`). Recall each file represents a sparse matrix, so there are many more entries in the matrix (with value 0) that are not listed. Hint: you do not need to simultaneously store all of the matrices in memory.

(b) Next, we will look at interactions between chromosomes 19 and 20. Our goal is to identify intermingling portions of these two chromosomes, as it has been shown experimentally that these regions are particularly active with respect to gene expression. These regions show up as contiguous sub-blocks of our interaction matrix with high average interaction values. To begin, plot a heat map of the interaction matrix (using the $\log(1 + x)$ transformation). What do you see? Do you see any interacting regions?

(c) To identify regions with high interaction frequencies, we will perform a series of hypothesis tests. Under our null hypothesis, we assume all entries of the (transformed) interaction matrix are independent and identically distributed according to a Gaussian distribution with mean and standard deviation computed in part (a) (μ and σ). Given a $k \times l$ submatrix with mean value m, we compute its adjusted p-value as:

$$N_{\text{submatrices}} \left(1 - \Phi \left(\frac{(m - \mu)\sqrt{kl}}{\sigma} \right) \right)$$

where $\Phi(\cdot)$ represents the standard normal CDF and $N_{\text{submatrices}}$ is the number of possible contiguous submatrices in the interaction region. Note: we have adjusted our p-value because we are testing multiple hypotheses when we look for submatrices with high interaction values. Explain where this formula comes from and compute the value of $N_{\text{submatrices}}$ for the chromosome 19-20 interaction matrix.

(d) As you saw in part (c), there are too many submatrices for us to compute the p-value for each one. Instead, we will use a greedy algorithm to identify regions with small p-values. To identify a submatrix with low p-value, we perform the following procedure, which we call `greedy_search()`:

1. Randomly pick an entry in the interaction matrix. Compute the p-value of the 1×1 submatrix with this entry. We call this initial submatrix M.
2. Compute the p-value of the submatrix that consists of M joined with the column to the right of it. Repeat, but with M joined with the column to the left of M. Repeat again with M joined with the row above. Repeat once more with M joined with the row below.
3. If all four of these transformation led to increases in p-value, stop. Otherwise, proceed to step 4.
4. Choose which of the four transformations led to the smallest p-value and add the appropriate row/column to M. Return to step 2.

Explain why this proceedure works. Implement this procedure and run it a few times to verify it works. Since we use a random initialization, we can get very different results each time. Therefore, we repeat `greedy_search()` several hundred times with different initializations and pick the result with the smallest p-value. Our overall procedure to identify interaction regions in a given (transformed) interaction matrix, Z, is as follows.

1. Run `greedy_search()` (multiple times) on Z to identify a submatrix with near-minimal p-value.
2. If the p-value of this submatrix is greater than 0.01, stop. Otherwise, proceed to step 3.
3. Store the identified submatrix as an interacting region. Subtract the mean of this submatrix from each entry of the submatrix in Z. Return to step 1 with this updated Z.

Implement this procedure and run it on the interaction matrix for chromosomes 19 and 20. Plot a heat map of the (transformed) interaction matrix and highlight the interacting regions.

(e) Run the procedure you developed in part (d) on all pairs of chromosomes. Count the number of intermingling 250kb regions for each pair, i.e. the number of entries in the interaction matrix that are contained in any of the identified interaction regions. Plot a heat map of the inter-chromosome interaction counts (i.e. a 22×22 matrix with these interaction counts).

Part II - Cell differentiation and gene expression

Problem 2.2: [10pts] Single-cell RNA-seq analysis.

In this problem, we analyze single-cell RNA-seq data and determine structure in this high-dimensional data. The data set consists of 272 cells, each row corresponds to the RNA-seq measurements of a particular gene. Each entry corresponds to the normalized transcript compatibility count (TCC) of an equivalence class. An equivalence class is a set of short RNA sequences. The TCC counts the number of reads of sequences which are compatible with each equivalence class for a given cell. The entries have been normalized so that each row in the matrix sums to 1. The data for this problem can be found in "Trapnell.csv." Note: this data has 1,065,024 columns, so the built-in csv-reader functions in R/numpy may be very slow. You may want to look for alternatives.

(a) Determine cell clusters by applying k-means clustering to the data. Hint: it may be helpful to first apply a dimension reduction method such as tSNE or PCA. This will help you determine the correct number of clusters to use and can speed up computations.

(b) Which genes are good markers for each cluster? Using the clusters calculated in the previous part as labels for each cell, train a classifier on the original data. For example, you could use logistic regression (make sure you include a regularization term because the data is very high-dimensional!). Which genes have the largest coefficient values for each cluster?

(c) Map the cells to a differentiation tree. (To do this, you can, for example, compute a minimum spanning tree for the cluster centers from part (a). It may be helpful to compute a distance matrix.)

Part III - Theory

Problem 2.3: [10pts]
In problem set 1, we looked at penalized linear regression in order to obtain a sparse solution, when we believe that only a few features are important. We can similarly apply regularization to logistic regression. Here, consider the effects of different norm penalties on the coefficient vector β. We begin by running regression on a single variable X and get an output coefficient $\hat{\beta}$.

(a) After the initial regression, we now duplicate the variable X and rerun our regression with an ℓ_1 penalty. How does the new $\hat{\beta_{\ell 1}}$ compare to $\hat{\beta}$.

(b) Now, instead of an ℓ_1 penalty, we duplicate the variable X and rerun our regression with an ℓ_2 penalty. How does the new $\hat{\beta_{\ell 2}}$ compare to $\hat{\beta}$.

(c) Each of the ℓ_1 and ℓ_2 penalties have their advantages. In order to get the best of both worlds, one commonly applied option is the elastic net penalty, which applies both an ℓ_1 and ℓ_2 penalty. The penalty term that is applied to the unregularized objective is shown below (where $\alpha > 0$ is some constant).

$$P_\alpha(\beta) = (1-\alpha)||\beta||_2 + \alpha||\beta||_1$$

What shortcomings/advantages of the ℓ_1 and ℓ_2 penalties does the elastic net penalty help to overcome? In what applications may this penalty be useful?

Next, we will see how to estimate the parameters of a logistic regression model. Recall, logistic regression is a generalized linear model for predicting a binary response variable.

$$(Y \mid X = x) \sim \text{Bernoulli}\left(\frac{1}{1+\exp(-\beta_0 - \beta^T x)}\right)$$

(d) (*optional for undergraduates*) Suppose you observe n iid, labeled data points, $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$. What is the log-likelihood function for these observations under the logistic regression model? (This optimization problem does not have a closed form solution as in the linear regression case, but it is in fact a concave function, so it is easy to maximize numerically.)

(e) (*optional for undergraduates*) Recall that in LDA, we assume

$$\begin{aligned}(X \mid Y = c) &\sim \mathcal{N}(\mu_c, \Sigma) \quad c = 0, 1 \\ P(Y = 1) &= \eta\end{aligned}$$

Suppose, we again observe n iid, labeled data points, $(x_1, y_1), \ldots, (x_n, y_n)$. What is the log likelihood function for these observations?

(f) (*optional for all*) What are the decision boundaries for logistic regression and for LDA? Compare the two decision boundaries. (The decision boundary corresponds to the function $P(Y = 1 \mid X = x)$. If this is greater than $\frac{1}{2}$, we give x the label 1, otherwise, we give x the label 0.) How does LDA relate to logistic regression?