Problem Set 2

EN 600.438/638

February 25, 2019

Due date: March 11, 2019 by midnight
Submission: Please submit your write-up pdf files and codes to gradescope.
Reminders: You may discuss problems in small groups but must complete your own write-up and code. You may not copy any of your work or code from others, including but not limited to any resources you may find on the internet. List ALL people with whom you collaborated on your submission. You have 5 total late days for the semester, and can use a maximum of 3 on this assignment. Days will be rounded up; for example, if you submit your assignment 3 hours late, you will use 1 full late day. If you submit your assignment 25 hours after the deadline, you will use 2 full late days.
Programming component: You must submit all source code in addition to answering the questions below (in the template). Instructors should be able to run your code, without exceptions, on gradx or ugradx **with the command specified in each question**. Please note that there will be a text file called `install_modules.txt` that provides all the packages you can use for the programming questions. And we will import only the modules from this text file to run your code. If instructors are not able to run your code you will not receive full credit for the problem in question. If you have questions or trouble setting up your code this way, please get in touch with one of the instructors.

Exercise 1. Heritability (10 points)

1. True or False: A rare variant in a particular individual results in green eye color in that individual. Green eye color is therefor heritable. (2 points)

2. True or False: Heritability of height can change over time. (2 points)

3. True or False: Polygenic risk scores must be computed over SNPs in linkage disequilibrium. (2 points)

4. True or False: Narrow sense heritability assumes all phenotypic variance can be explained by additive genetic and environmental effects. (2 points)

5. Select ALL answers that apply: Which of the following may be the reasons that GWAS results are able to explain only part of heritability in height: (2 points)

   (a) Most GWAS ignore gene by environment (GXE) effects.

(b) Most GWAS do not have the power to detect rare genetic effects.

(c) Most GWAS do not account for non-additive effects of genetic variance.

(d) It is difficult to accurately estimate fixed effect sizes from GWAS given small sample sizes.

Exercise 2. Support Vector Machines (25 points)

In problem set 1, we used logistic regression to do phenotype classification in gene expression and GWAS data. In this problem set, we explore using supervised machine learning for a very different classification problem. We will be using support vector machines (SVMs) for *splice site prediction* using DNA sequence data.

Background on Splicing:

Eukaryotic gene structure is non-continuous. This means that any given sequence of DNA that can be transcribed and translated to a protein contains interspersed patches of non-coding regions called introns. Introns are spliced from pre-mRNAs after transcription. The coding regions of the genes are called exons. One problem in computational gene finding concerns the recognition of splice sites that mark the boundaries between exons and introns in eukaryotes. The majority of splice sites are characterized by the presence of specific dimers on the intronic regions: **GT** for donor and **AG** for acceptor sites. However, only about 0.1-1% of all GT and AG occurrences in the genome represent true splice sites.

Please refer to pg. 35-36 of the Molecular Biology background (from slides of lecture 2) for a visual representation of splicing.

Data:

We provide you with a real splicing dataset. This contains a total of 3175 DNA sequences of which 2175 are training instances (*Exercise2/trainSequence.txt*) and 1000 are test instances (*Exercise2/testSequence.txt*). Additionally, each instance has an associated phenotype label (*Exercise2/trainLabel.txt* and *Exercise2/testLabel.txt*), which is either "S" - spliced or "N" - not spliced. Each instance is a sequence of length 60. Our goal is to train a model which can determine whether a given sequence is likely to undergo splicing ("S) or not ("N").

Notes:

- Please standardize features to have mean 0 and standard deviation 1 before training the model. To do this, for each entry c_i of column C of the dataset you will do the following:

$$c_i = \frac{(c_i - \bar{C})}{\sigma_C}$$

  Please use numpy.std to compute standard deviation, and use 1 degree of freedom by setting option ddof=1. After doing this, use sklearn.svm with default parameters (except the 'kernel' parameter (see 2.2)).

- This problem involves doing 5-fold cross-validation on training data. This involves splitting the training data into 5 equal size blocks. Here, the first block will contain the first 435 training data instances. The second block will contain the next 435 traing data instances. And so on..

Problems:

1. One form of feature encoding can be k-mer frequency counts. A k-mer is a string of k nucleotides. You will compute frequencies for 2-mers, 3-mers and 4-mers of the DNA sequence with 4 bases A, T, G and C. Since the number of unique k-mers depends on the length of the sequence, this gives us 16 + 64 + 256 = 336 features.
   For example, consider the DNA sequence given below:

   (a) ATGCTGTGC

   The set of possible 2-mers in this sequence is {AT, AC, AG, AA, TT, TC, TG, TA, CT, CC, CG, CA, GT, GC, GG, GA}. If we count the number of times each of these features appear in (a), we see:

   {AT = 1, AC = 0, AG = 0, AA = 0, TT = 0, TC = 0, TG = 3, TA = 0, CT = 1, CC = 0, CG = 0, CA = 0, GT = 1, GC = 2, GG = 0, GA = 0}

   Compute all 2-mers, 3-mers, and 4-mer features for all the given training and test DNA sequences. (10 points)

   Code: Please name the code exercise2FeatureExtraction.py. We should be able to run your code as:

```
python3 exercise2FeatureExtraction.py --input trainSequence.txt
                                      --order featureOrder.txt
                                      --output trainFeatures.txt
python3 exercise2FeatureExtraction.py --input testSequence.txt
                                      --order featureOrder.txt
                                      --output testFeatures.txt
```

   The output files created by *exercise2FeatureExtraction.py* should have the same number of rows as the input file and should 336 columns (tab separated, no header or row names needed). The columns should be standardized (See first note). Please order the columns according to *Exercise2/featureOrder.txt*.

2. Using this data, train linear, gaussian and polynomial (with d=3 and 6) kernel SVMs with 5 fold cross-validation (please implement cross-validation manually). Use the default parameters in each case. For each kernel, report the average accuracy across the 5 cross-validation folds. (10 points)

   Code: Please name the code exercise2CrossValidation.py. We should be able to run

your code as:

```
python3 exercise2CrossValidation.py --features trainFeatures.txt
                                     --labels trainLabel.txt
                                      --kernel linear
                                      --output linear_model_cv_accuracy.txt
python3 exercise2CrossValidation.py --features trainFeatures.txt
                                     --labels trainLabel.txt
                                      --kernel gaussian
                                      --output gaussian_model_cv_accuracy.txt
python3 exercise2CrossValidation.py --features trainFeatures.txt
                                     --labels trainLabel.txt
                                      --kernel polynomial_3
                                      --output polynomial_3_model_cv_accuracy.txt
python3 exercise2CrossValidation.py --features trainFeatures.txt
                                     --labels trainLabel.txt
                                      --kernel polynomial_6
                                      --output polynomial_6_model_cv_accuracy.txt
```

The output files should contain a single number: the average accuracy across the 5 cross-validation folds.

3. From part (2), which model gives the best cross-validation accuracy? Use that model on your test data and return test prediction accuracy. (5 points)

   Code: Please name the code exercise2TestPrediction.py. We should be able to run your code (if the gaussian kernel gave best cross-validation accuracy...) as:

```
python3 exercise2TestPrediction.py --features_train trainFeatures.txt
                                   --labels_train trainLabel.txt
                                   --features_test testFeatures.txt
                                   --labels_test testLabel.txt
                                     --kernel gaussian
                                     --output test_accuracy.txt
```

   The output file should contain a single number: the accuracy on the test set.

Exercise 3. PCA (30 points)

Programming

RNA-seq is widely used to measure trancript expression. However, many factors beyond phenotypes of interest may influence gene expression measurements including experimental variables (batch, RNA integrity, etc.) and biological covariates (age, sex, ethnicity, etc.).

These may introduce systematic variability in data. In this exercise, you'll explore how variation in RNA-seq data are correlated with confounding covariates.

We provided a RNA-seq dataset (*Exercise3/counts.txt*) that contains the expression of 17,580 genes 73 human in prefrontal cortex samples. Of the 73 samples, 29 are from subjects with Parkinson's disease, and 44 are normal (control) samples (see *Exercise3/phen.txt*). We also provided data (*Exercise3/cov.txt*) measuring three sample covariates (age, RNA integrity numbers, and post mortem interval) for each subject. You'll compute principal components (PCs) of the gene expression dataset, and then investigate the correlation between the top PCs and these three covariates.
Note: Please use numpy.std to compute standard deviation, and use 1 degree of freedom by setting option ddof=1.
Note: the original data is collected from GEO (dataset id: GSE68719, url: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE68719). You may follow the given URL and read the corresponding paper if you wish to learn more about the data in question.

Written questions
For questions asking for a real number, please keep 3 digits after the decimal, like 0.012, 0.003

1. Preprocessing: (5 points)
   In file *Exercise3/counts.txt*, the rows are genes and columns are the samples.

   (a) Take the log-transform of data. Elements of the matrix

   $$X_{ij}$$

   should be transformed as follows:

   $$L_{ij} = log_2(1 + X_{ij})$$

   where X_{ij} = expression of gene i in sample j.

   (b) Scale the log transformed expression data using the z-score (gene-wise). Transform the elements of the new matrix as follows:

   $$Z_{ij} = (L_{ij} - \mu_i)/\sigma_i$$

   where L_{ij} = log transformed expression of gene i in sample j, μ_i = mean expression of gene i and σ_i = standard deviation of gene i.

   Code: Please name the code exercise3PreProcess.py. We should be able to run your code as:

```
python3 exercise3PreProcess.py --input counts.txt
                               --output log_transformed_and_standardized_expr.txt
```

The output file should be in the same format at *Exercise3/counts.txt*, ie. row labels that are ensemble ids (in same order) and column labels that are sample ids (in same order).

2. Transpose the scaled data you get from Exercise 3.1.b. After transposing, rows will now represent samples and columns will be genes. Compute the principal components of the scaled expression data. Report the percent variance explained by each of the top 10 PCs. (10 points) (NOTE: No submitted code is required for this part)

3. Visualization of data is also very important in genomics. However, it is difficult to visualize high dimensional data. A common practice is use the top 2 principal components for visualization. Make a scatter plot of all the samples after projecting them onto the first 2 principal components. Use different symbols (e.g. + and x) for normal and Parkinson's samples respectively in the plot. (5 points) (NOTE: No submitted code is required for this part)

4. Compute pairwise Pearson correlation values (along with p-values) between each of the top 10 PCs and the provided covariates (30 correlations in total). Report how many PCs are strongly correlated with at least one of the covariates? For this exercise, assume a correlation is strong if $|r| > 0.2$ and raw p-value < 0.05. (3 points) (NOTE: No submitted code is required for this part)

5. Report Pearson correlation p-values between disease status and each of the three covariates. Based on these correlation p-values, do you think the disease status may be confounded by a covariate? In such a scenario, do you think that the principal components fully reflect the disease status of the samples? If not, why not? (3 points) (NOTE: No submitted code is required for this part)

6. Test for genes with significant differences in expression between normal and Parkinson's samples using a t-test (assume two groups have equal variance) using the pre-procesed expression data you generated in Exercise 3.1. Report the number of genes that are significant at $p < 0.05$. Then, using Benjamini Hochberg (at FDR = 0.05), report the number of significant genes after correcting for multiple hypothesis testing. Do you think that these results are affected by the covariates you examined in the previous parts? (4 points) (NOTE: No submitted code is required for this part)

Exercise 4. Clustering (35 points)
In this exercise. you'll separate samples by tissue type using a clustering approach. Your job is to cluster the samples in an unsupervised fashion – without using given tissue type labels – and observe how well the resulting clusters reflect the true tissue types.
Data: The provided dataset (*train_expr.txt* and *test_expr.txt*) includes normalized gene expression of 100 genes in 1816 samples. These samples come from 5 different tissues: Muscle-skeletal, Lung, Thyroid, Adipose-subcutaneous, and Whole blood (encoded as 0,1,2,3, and 4, respectively, in *train_label.txt* and *test_label.txt*).
Note: the data given to you here is a preprocessed subset of the data available from the Genotype-Tissue Expression (GTEx) consortium (v6). If you are interested, you may visit the GTEx website (http://www.gtexportal.org/) and read the relevant papers

(http://www.gtexportal.org/home/news?id=176263).
Code:Please name the code exercise4.py. We should be able to run your code as:

```
python3 exercise4.py --train train_expr.txt --test test_expr.txt
-- Kmin 2 --Kmax 10 --output kmeans_output.txt
```

The goal for the script is to assign data points to clusters. It should be able to read in train data where each row is a feature and each column is a data point. The script then runs clustering using a range of the number of clusters, from `Kmin` to `Kmax`, and decide which k gives the best results. Using the chosen optimal k, the script should run $Kmeans$ clustering on the training data, and use the results to assign each data point in the test data to the clusters, and save the result in `output`. Here the training data and test data can be the same. The output file format should be like this (fields separated by tab):

```
Name_for_data_point_1    cluster_ID
Name_for_data_point_2    cluster_ID
```

...

1. [**EN 638 only**] Implement a function *myKmeans()* to train a k-means clustering model from scratch. (15 points)
   The function should take the following parameters:

   - *data* - n x d numpy matrix. n = number of samples, d = number of features
   - k - int. number of clusters.
   - *centers* - k x d-length numpy matrix. initial value of cluster centers.
   - *maxit*- int. maximum number of iterations.

   The myKmeans() function should return a tuple of two items (in this order):

   - *clusters* - n-length numpy array containing the cluster assignment label of each sample.
   - *centers* - k x d numpy matrix containing the final k d-dimensional cluster centroids.

   You should also implement a function *predict()* that takes the output of the *myKmeans()* function as well as test samples, and predicts the cluster assignment of each sample using the closest centroid. The *predict()* function should take the following inputs:

   - *model* - the tuple returned by the *myKmeans()* function.
   - *test_data* - a q x d numpy matrix where q = number of query samples.

   The *predict()* function should return the following output:

   - *clusters* - A q-length numpy array containing the cluster assignment of each sample.

2. Run k-means with k clusters (k is in the range read in from command line). Use the first k samples in the training data as the initial centers, and run a maximum 10 iterations. (EN 638 - use the *myKmeans()* function you wrote in the previous part, EN 438 - you may use the package available in `installed_modules.txt`.) (5 points)

3. Selection of the number of clusters (k) is an important factor in clustering. There are different approaches for select of k including Bayesian Information Criterion (BIC), the Elbow method, Silhouette, etc. In this exercise, you'll select k using the BIC score defined as:

$$BIC = -2 \log \hat{L} + m \log n \tag{1}$$

   Here, $\hat{L}$ is the estimated maximum likelihood, m is the number of free parameters in the model, and n is number of samples. Generally, the smaller the BIC score, the better the model.

   (a) Free parameters are parameters which are learned by the model. How many free parameters are there in a k-means model with k clusters of d-dimensional points? (2 points)

   (b) Write the equation to compute the log-likelihood for k-means. (3 points)

   (c) Train k-means clustering model using values of k from 2 to 10. Use the first k points as initial centers. Plot BIC scores of each model against k. (5 points)

   (d) Which k will you select here? (2 points)

4. Make a scatter plot of the projection of the training data onto the top 2 principal components. Label the samples by cluster membership from your k-means result (use the k selected in the previous question) using different colors, and also label true tissue types using different symbols (such as + or x).(1.5 point) Make the same plot for test data. (1.5 points)

Exercise 5. Linear Mixture Model in GWAS (10 points)
After we have learned GWAS, we'd like to use it to investigate what genetic variants maybe associated with dog's weight. To do so, we collect data of 5000 dogs from 5 different breeds. The weight of each dog is stored in *Exercise5/phenotype.txt*, genotypes for 100 SNPs are stored in *Exercise5/genotype.txt*, and the breeds information are stored in *Exercise5/breeds.txt*. NOTE: No submitted code is required for this question.

1. Run linear regression of each of the SNPs on the trait. Based on the BH-corrected pvalue, which SNPs are found to be significantly associated with weight at FDR < 0.05? (Hint: use `sm.OLS` from `statsmodels` module.) (2 points)

2. Run linear mixture model (LMM) of each of the SNPs on the trait stratified by breed. Based on the BH-corrected pvalue, which SNPs are found to be significantly associated with weight at FDR < 0.05? (Hint: use `smf.mixedlm` from `statsmodels` module.) (2 points)

3. Are the two results different? What do you think is the main reason? (3 points)

4. If there are SNPs that appear in the result from LMM, but not in the result from linear regression: make a scatter plot of `weight` by `genotype` colored by `breed` for any such SNP. If not, make a scatter plot of `weight` by `genotype` colored by `breed` for any significant SNP. (3 points)