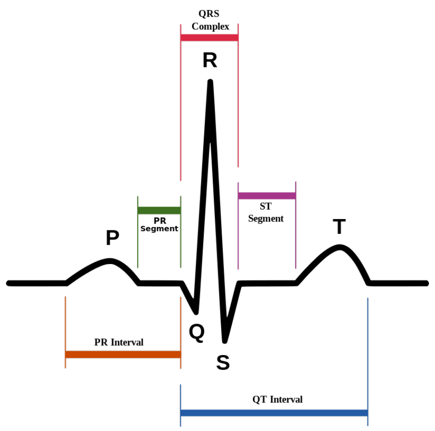
CS589: Machine Learning - Spring 2017

Homework 4: Clustering

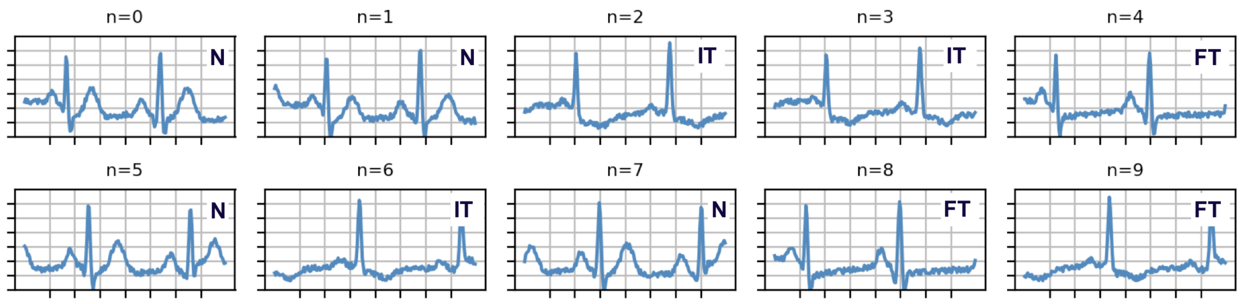
Getting Started: Download the assignment archive from Moodle and unzip the file. This will create the directory structure shown below. The data files for the data set are under the 'Data' directory. You will write your code under the Submission/Code directory. Make sure to put the deliverables (explained below) into their respective directories.

```
HW04
--- Data
     |--ECG
--- Submission
     |--Code
     |--Figures
```

Data Set: This assignment will focus on applications of clustering to processing Electrocardiogram (ECG) signals. An ECG signal records the electrical activity of the heart. Each heart beat in an ECG recording is characterized by five sub-waves labeled P, Q, R, S, and T with R typically being the highest magnitude sub-wave (see figure at right). As heart rate goes up, the distance between the peaks of subsequent sub-waves of the same type decreases. Specifically, instantaneous heart rate is typically computed as the inverse of the time interval between successive R peaks.



ECG recordings are often used to diagnose different heart-related conditions including the presence of a variety of arrhythmias (irregular heart beats). In this assignment, you will work with a data set containing 100Hz ECG data. Each data case corresponds to a 2 second trace represented as a 200-long vector of ECG wave amplitudes. The data include both normal (N) and two types of abnormal heart beats (IT and FT). Some example recordings are shown below.



In addition to basic clustering, you will also explore the combination of clustering and classification as another method for achieving a non-linear classifier. To support these tasks, the data have been split into training and test sets stored in NumPy binary format. The provided Submission/Code/run_me.py file contains example code for reading in both data sets. Class labels are provided for both the train and test data. These will be used for classification, but not during clustering. Note that Kaggle will not be used for this assignment.

Deliverables: This assignment has two types of deliverables: a report and code files.

- **Report:** The solution report will give your answers to the homework questions (listed below). The maximum length of the report is 5 pages in 11 point font, including all figures and tables. You can use any software to create your report, but your report must be submitted in PDF format.
- **Code:** The second deliverable is the code that you wrote to answer the questions, which will involve training and evaluating models. Your code must be written in Python 2.7. You may use any clustering methods from SciKit Learn. Your code for clustering must be runable from *run_me.py* and should reproduce your results without the need for any manual intervention (make sure any plots are saved to files in your final code and not displayed on screen, which halts your code until they are manually closed). To help organize your code, two helper files are provided *cluster_utils.py* and *cluster_class.py*. Code stubs and/or starter code are provided in each file.

Submitting Solutions: When you complete the assignment, you will upload your report and all of your code using Gradescope.com. If you used Python to generate report figures, place them in Submission/Figures. Create a zip file of your submission directory, Submission.zip. Upload this single zip file to Gradescope as your solution to the HW04-Programming assignment. Gradescope will run checks to determine if your submission contains the required files in the correct locations, and will run preliminary correctness checks on some of the code. Upload your report as your solution to the HW04-Report assignment. The submission time for your assignment is considered to be the later of the submission timestamps for your code and report.

Academic Honesty Statement: Copying solutions from external sources (books, web pages, etc.) or other students is considered cheating. Sharing your solutions with other students is considered cheating. Posting your code to public repositories like GitHub is also considered cheating. Collaboration indistinguishable from cheating will be treated as cheating. Any detected cheating will result in a grade of 0 on the assignment for all students involved, and potentially a grade of F in the course.

1. (*30 points*) **Clustering:** In this question, you will select, describe, and experiment with a single clustering method/model. Read all parts of this question and the following questions before selecting a method.

1.1. (*5 pts*) Name the clustering method you selected (e.g. hierarchical clustering, k-means, mixture model, etc.), and explain how it works in your own words. Include equations as appropriate to support your description. Explain the strengths and weaknesses of the method and why you selected it for this task.

1.2. (*5 pts*) Given a data matrix $\mathbf{X}$, describe a method for measuring the quality of a clustering represented by a given vector of cluster indicators $\mathbf{Z}$. Include equations as appropriate to support your description. For example, you might use the sum of the pairwise within-cluster distances. There are many possibilities, and some are better suited to specific clustering methods/models. Explain why you chose the function you selected.

1.3. (*10 pts*) Implement your cluster quality method by completing the *cluster_quality(X,Z,K)* function in *cluster_utils.py*. This function takes a data array of shape (N,D) as its first input, a cluster indicator array Z of shape $(N,)$ as its second argument, and the number of clusters K as its final input. Assume that the clusters are numbered $0, ..., K-1$. Do not assume that every cluster has at least one data point assigned to it. The method should produce a scalar float as output. Include a listing of this function in your report as the answer to this question.

1.4. (*5 pts*) Add code to *run_me.my* to run your clustering method on the ECG training data (Xtr) while varying the number of clusters from 1 to 40. Produce a line plot showing the value of your cluster quality method versus the number of clusters. Include this figure in your report as the answer to this question.

1.5. (*5 pts*) Select and describe a method for determining an optimal number of clusters for the data set. Include equations as appropriate to support your description. Implement and/or apply your method for selecting the optimal number of clusters to the data set using the training data. Your method does not need to be fully automatic (it can rely on manual inspection), in which case there may be nothing more to implement. If your approach does require an additional implementation, add it to *cluster_utils.py* as a function, call it from *run_me.my*, and add the code listing to your report. Report the optimal number of clusters found by your method.

2. (*30 points*) **Cluster Analysis:** In this question, you will implement and apply basic methods to analyze the clustering that you determined to be optimal in the previous question.

2.1. (*5 pts*) Complete the function *cluster_proportions(Z,K)* in *cluster_utils.py*. This function should accept a cluster indicator array Z of shape $(N,)$ and the number of clusters K, and return an array p of shape $(K,)$ such that $p[k]$ gives the proportion of data cases in cluster k. Assume that the clusters are numbered $0, ..., K-1$, but that some clusters may be empty. Include a code listing for this function in your report as the answer to this question.

2.2. (*5 pts*) Using the best clustering that you found in Question 1, add code to *run_me.py* to apply your *cluster_proportions(Z,K)* function to compute the cluster proportions. Produce a bar chart showing the cluster proportions and include it in your report as your answer to this question.

2.3. (*5 pts*) Complete the function *cluster_means(X,Z,K)* in *cluster_utils.py*. This function should accept a data array X of shape (N,D) as its first input, a cluster indicator array Z of shape $(N,)$ as its second argument, and the number of clusters K as its final input. As output, it should produce an array mu of shape (K,D) such that $mu[k,:]$ represents the average (mean) of the data cases (rows of X) assigned to cluster k by Z. Include a code listing for this function in your report as the answer to this question.

2.4. (*5 pts*) Using the best clustering that you found in Question 1, add code to *run_me.py* to apply your *cluster_means(X,Z,K)* functions to compute the cluster means. Use the function *show_means(mu,p)* to display the mean of each cluster. This function will plot the cluster means as waveforms and list their proportions. Include this figure in your report.

2.5. (*10 pts*)Analyze the cluster means and proportions produced in the previous step. Explain what aspects of the data seem to be reflected in the clustering and what might be ignored. To support your answer, you may want to produce clusterings with more or fewer clusters and visualize their cluster means to see how stable different properties of your clustering are with respect to K. You may also want to inspect some of the data cases assigned to each cluster in the optimal clustering, as well a sub-sample of all of the data to understand the variation in the data (you can plot individual cases as waveforms and inspect). You may include any additional figures in your report that are helpful to explain your answer.

3. (*30 points*) **Cluster-Based Classification:** In this question, you will leverage the clustering method you investigated in Questions 1 and 2 to build a simple non-linear classification model implemented according to the Sklearn classifier API. This classifier will associate each cluster in a clustering model with a class label. Given a data case, the classifier will first determine which cluster the data case belongs to. It will then predict the class the label associated with the cluster. If you haven't seen how to write classes in Python previously, Google for a tutorial before starting to write code for this question.

3.1. (*5 pts*) Explain how this classifier relates to other non-linear classifiers that we covered in Unit 1 of the course. What do you think its strengths and weaknesses are?

3.2. (*5 pts*) Complete the *init* method in *cluster_class.py*. This method takes the number of clusters K as an input and should initialize your clustering model. It should also initialize any data structures that you need to support classification. You can add any additional imports needed to the top of *cluster_class.py*. Add the code listing to your report.

3.3. (*5 pts*) Complete the *fit(X,Y)* function in *cluster_class.py*. The method takes as input a numpy feature array X of shape (N, D) and a label array Y of shape $(N,)$ as input. This method must first fit your clustering model using the number of clusters specified during initialization. For each cluster, you must then compute the most frequent label among the data cases assigned to that cluster (this label will be predicted for any data case assigned to the cluster). If no cases are assigned to a given cluster, select a class label at random. If two classes are tied, break the tie at random. You must then store both the fit clustering model and the mapping between clusters and predicted class labels for use during prediction. Add the code listing to your report.

3.4. (*5 pts*) Complete the *predict(X)* function in *cluster_class.py*. The method takes as input a numpy feature array X of shape (N, D). This method should use your fit clustering model to assign each data case in X to a cluster, and then use the learned mapping between clusters and class labels to predict a class label for each data case. It should output the corresponding predictions in an array Y of shape $(N,)$. Note that some clustering methods do not include a *predict()* method for assigning new data cases to clusters (see the documentation for individual clustering methods in the sklearn.cluster module for details). In this case, you will have to implement such predict method yourself. Add the code listing for all components of this question to your report.

3.5. (*5 pts*) When your implementation is complete, add code to *run_me.py* to train your model on Xtr and Ytr using between 1 and 40 clusters and compute the prediction error using Xte and YTe as test data. Plot the prediction error as a function of the number of clusters and include the plot in your report.

3.6. (*5 pts*) How does this plot compare to your cluster quality plot? How does the optimal number of clusters according to error compare to what you found for the optimal number of clusters in Question 1? Explain any differences in the results between the two approaches.

3.7. (*10 pts*) **Bonus:** This same basic idea can be applied to construct more advanced cluster-based classifiers by applying more powerful classifiers like logistic regression or SVMs within clusters. Experiment with other choices of the within-cluster classifier and plot their prediction error curves. Discuss your findings. Include the code for your best performing model in *cluster_class_bonus.py*.

4. (*10 points*) **Reproducibility and Code Quality:** You should aim to design your code so that all experimental results shown in your report can be reproduced by running the *run_me.py* file (except results that rely on manual interpretation). You are strongly encouraged to use the Python plotting library Matplotlib to create figures from your results programatically. Your code must also be sufficiently documented and commented that someone else can easily understand what each method is doing. You will be scored on how reproducible your results are, and how well documented and structured your code is. For this assignment, we will focus the reproducibility assessment on the *fit(X,Y)* and *predict(X)* functions you write in *cluster_class.py*. All code will be checked for quality.