ENGS/QBS 108 Fall 2017 Assignment 2

Due October 3, 2017

Instructors: George Cybenko and Saeed Hassanpour

1 K means and K nearest neighbors clustering

Problem: Clustering [13 points]. In this problem, you will solve a clustering task using the k-Means and k-NN algorithms you learned in class. The dataset for this problem has been provided in the *clustering* folder as csv files. There are 3,100 examples in the dataset. Each entry has two features (x_1, x_2) and a target cluster y. The dataset has been split into two sets: *train_data.csv* and *test_data.csv*. Use the training set solely for all analysis, but report performance results on the test set.

1. A reasonable first step in every machine learning task is to understand the dataset at hand. Proceed to explore this problem's dataset by addressing the following:

   (a) Choose a suitable type of plot and visualize the training data.

   (b) From your plot, how many clusters, k, are in the dataset?

   (c) What is the dataset distribution across the k clusters? Use a histogram to illustrate this.

   (d) Is the training data balanced?

2. Using the k-Means algorithm, implement a clustering model for the training data. You can use Matlab, Python (Scikit-learn) or any other tools at your disposal. Be sure to include code or screenshots of the implementation. Report the clustering accuracy of your model on the test set.

3. Repeat the above but using the k-Nearest Neighbors algorithm. Note that k here refers to the number of neighbors, not clusters. By repeated training, find the optimal k that produces a similar number of clusters as you first estimated. Report the clustering accuracy of this model on the test set.

2 Decision Tree and SVM classifiers

Problem: Classification [25 points]. In this problem you will use Decision Trees and SVM methods to classify the quality of red and white vinho verde wine samples based on their physicochemical properties. The datasets for this problem are in the *classification* folder as csv files. There are training and test datasets for both white and red wine samples (total of 4 .csv files.)

1. First let's explore the datasets through the following exercises. Note that we cannot plot the data in a meaningful way given that number of features exceed the physical dimensions:

   (a) How many datapoints are in the *red_train.csv* and *white_train.csv* files.

   (b) How many features are available for each datapoint?

   (c) What are the average *alcohol* and *pH* values for red and white training samples?

2. Decision Trees:

   (a) Implement a decision tree model for the *red_train* data and report the mean accuracy or score of your model on the test dataset.

   (b) There are a lot of parameters that can be tuned to improve your model, one of which is the criteria for ending the splitting process. Two common ways of terminating the splitting process are *maximum depth* of the tree or *minimum number of samples* left. Repeat the process in *part a* using different values for maximum depth of the tree and plot a graph of accuracy vs. tree depth.

3. Support Vector Machines (SVM)

   (a) Using the same dataset for red wine samples, build a SVM classifier and report its mean accuracy on the test dataset.

   (b) Depending on your data, a linear classifier might not be able to separate your datapoints in that case you would want to increase the complexity of your classifier by choosing different *kernels*. This can be done by selecting different kernels such as *Linear, Polynomial or Radial Basis Function(RBF)*. Redo the previous part with "Linear" and "RBF" kernels and report the accuracy in each case. Does using a more complex classifier improve the accuracy of your model?

3 Logistic Regression

[25 points + 10 additional implementation points] This assignment offers you two choices, one is an applied implementation that follows the class material, and the other one is an complete implementation for those interested in deeper understanding of logistic regression. The complete implementation will be awarded with extra-credit points which will be counted towards a grade-boost over the course of the term.

For this problem, you will need to use the logistic regression. You will evaluate your algorithm on the so-called *parkinsons* dataset contained in *parkinsons.mat*, which we obtained from the UCI Machine Learning Repository. The description of the dataset can be found here (*hint*: in MATLAB, you can check the description of the features using *parkinsons.names* command). The objective is to recognize healthy people from those with Parkinson's disease using a series of biomedical voice measurements.

This assignment offers two options, however **only one needs to be solved for the full credit**.

1. **applied implementation** that follows the class material
2. **complete implementation** for those interested in deeper understanding of logistic regression and gradient ascent algorithm. Should it turn out successful, this implementation will be awarded with 10 additional extra-credit points

3.1 Applied implementation

Problem: Programming: Naive Logistic Regression [13 points]. Train the logistic regression via gradient ascent using a small fixed learning rate $\alpha = 10^{-6}$. Your solution should report both the training and the text error, the number of iterations needed to reach convergence, and you should plot a curve of log likelihood as a function of the number of iterations (*hint:* if you have implemented the functions correctly, the log likelihood curve should be monotonically increasing).

Problem: Programming: Line search optimization [8 points]. Train the logistic regression using line search algorithm (aka newton line-search) to refine the step size α adaptively at each iteration. The line search method requires an initial value α_0. This value should be chosen fairly large, e.g., $\alpha_0 = 10^{-4}$. Your solution should report, for both the version using the fixed α (as coded for the previous question) and the one using line search, the following information: the training and the test errors, the number of iterations needed to reach convergence, and the log likelihood as a function of the number of iterations.

Based on the results, answer the questions below:

Problem: Writing [2 points]. Compare the training and test errors of the two variants of gradient ascent. Are the errors different in the two cases? Explain why or why not.

Problem: Writing [2 points]. Now compare the two log likelihood curves. Does the method using line search converge faster or slower than the version using a fixed step size? Explain the result.

3.2 Complete implementation

Problem: Programming: gradient ascent [13 points]. Implement a gradient ascent algorithm maximizing the logistic regression log-likelihood function (you are not allowed to use MATLAB built-in functionalities for gradient ascent optimization). The method requires selecting a value for the learning rate : for now, use a fixed small value (e.g., $\alpha = 10^6$).

To do so, you should implement following functions:

- *initialize* initializes the weights for maximum log likelihood training. Your task is to code for the purpose of the assignment, this can be random, although there are better heuristics.
- *predict* computes the label predictions and the class posterior probabilities for the input examples in matrix X using the parameter vector θ.
- *loglik* computes the log likelihood of the training data given the model θ. Add the Matlab constant eps (i.e., a tiny positive number) to the probabilities before taking the log in order to avoid numerical issues. [Hint: you can call *predict* inside this function].
- *gradient* computes the gradient of the log likelihood at the current θ. [Hint: use *predict* inside this function].
- *error* calculates the misclassification rate by comparing the pre- dicted labels to the true labels Y. The misclassification rate is given by the number of misclassified examples over the total number of examples.

After implementing the above optimization functions, you should be able to implement the training for your own logistic regression algorithm. It should report both the training and the test error, the number of iterations needed to reach convergence, and will plot a curve of the log likelihood as a function of the number

Problem: Programming: Line search optimization [8 points]. Train the logistic regression using line search algorithm (aka newton line-search) to refine the step size α adaptively at each iteration. The line search method requires an initial value α_0. This value should be chosen fairly large, e.g., $\alpha_0 = 10^{-4}$. Your solution should report, for both the version using the fixed α (as coded for the previous question) and the one using line search, the following information: the training and the test errors, the number of iterations needed to reach convergence, and the log likelihood as a function of the number of iterations.

Based on the results, answer the questions below:

Problem: Writing [2 points]. Compare the training and test errors of the two variants of gradient ascent. Are the errors different in the two cases? Explain why or why not.

Problem: Writing [2 points]. Now compare the two log likelihood curves. Does the method using line search converge faster or slower than the version using a fixed step size? Explain the result.