CSCI 5525: Machine Learning (Fall 2020)

Homework 4

Due 12/10/2020 11:59 PM CST

This homework consists of 3 programming problems. The first two will focus on using boosting and random forests for classification and the last will focus on using k-means for image segmentation.

In the first two problems, we will use a cancer dataset[1]. This dataset consists of $n = 699$ samples, $p = 10$ categorical attributes, and 2 classes, benign and malignant, which are in the last column and represented by the values 2 and 4 respectively. Often in practice, the data we have is incomplete. To get some experience with that, this dataset has some missing values represented by "?". It is up to you to decide how to handle this missing data. (Two options are to: (1) entirely remove the features with missing data; or (2) fill in the values with the mode value of the feature. However, you are not limited to these options and may choose to deal with the missing features in a different way. Please indicate how you dealt with missing values in your summary.)

1. **(35 points)** In this problem, we consider Adaboost. Implement the Adaboost algorithm with 100 weak learners and apply it to the cancer dataset described above. For the weak learners, use decision stumps (1-level decision trees). You must implement the decision stumps from scratch. Use information gain as the splitting measure.

   Please submit (a) **summary of methods and results** report and (b) **code**:

   (a) **Summary of methods and results**: Briefly describe the approaches used above, along with relevant equations. Report a plot of the classification error on both the train and test sets as the number of weak learners increase. (One plot where the x-axis is the number of weak learners from 1 to 100 and the y-axis is the classification error.)

   (b) **Code**: Submit the file `adaboost.py` which contains the function `def adaboost(dataset: str) -> None:`. The function takes in a string of the dataset filename and does not return anything but must print out to the terminal (stdout) the train and test classification error rates as the number of weak learners increase.

2. **(35 points)** In this problem, we consider Random Forests. Implement the Random Forest algorithm with 100 decision stumps. You must implement the decision stumps from scratch. Use information gain as the splitting measure. Apply your Random Forest implementation to the cancer dataset described above and do the following:

   (i) Use $m = 3$ random attributes to determine the split of your decision stumps. Learn a model for an increasing number of decision stumps in the ensemble. Compute the train and test set classification error as the number of decision stumps increases.

[1] `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+%28original%29`

(ii) Vary the number of random attributes m from $\{2, \ldots, p = 10\}$ and fit a model using 100 decision stumps. Compute the train and test set classification error as the number of random features m increases.

Please submit (a) **summary of methods and results** report and (b) **code**:

(a) **Summary of methods and results**: Briefly describe the approaches used above, along with relevant equations. Report a plot of the classification error on both the train and test sets for both (i) and (ii) above. (A total of 2 plots where for (i) the x-axis is the number of decision trees and y-axis is the classification error, and (ii) the x-axis is the number of random features m and y-axis is the classification error.)

(b) **Code**: Submit the file `rf.py` which contains the function `def rf(dataset: str) -> None:`. The function takes in a string of the dataset filename and does not return anything but must print out to the terminal (stdout) the train and test classification error rates for (i) and (ii) above.

3. **(30 points)** In this problem, we consider k-means for image segmentation. We can use k-means to cluster pixels with similar (color) values together to generate a segmented or compressed version of the original image. Implement the k-means algorithm and apply it to the provided image "umn_csci.png". For each $k = \{3, 5, 7\}$, generate a segmented image and compute the cumulative loss (i.e., distortion measure from the lecture notes). (Note, it may be helpful to test on a smaller version of the image "umn_csci.png" to ensure your code works but report final results on the full version.)

   Please submit (a) **summary of methods and results** report and (b) **code**:

   (a) **Summary of methods and results**: Briefly describe the approaches used above, along with relevant equations. For each value of $k = \{3, 5, 7\}$, report the final (i.e., after k-means has converged) segmented image and a plot of the cumulative loss during training. (This will be 3 segmented images and 3 plots of the loss where the x-axis is the training iteration number and y-xais is the loss value.)

   (b) **Code**: Submit the file `kmeans.py` which contains the function `def kmeans(image: str) -> None:`. The function takes in a string of the image to segment and does not return anything but must print out to the terminal (stdout) the cumulative loss at each iteration during training.

Additional instructions: Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Please specify instructions on how to run your program in the README file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures should be included in the report.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. In your code, you **cannot** use machine learning libraries such as those available from scikit-learn for learning the models. However, you may now use scikit-learn for cross validation and computing misclassification errors. You may also use libraries for basic matrix computations and plotting such as numpy, pandas, and matplotlib. Put comments in your code so that one can follow the key parts and steps in your code.

Your code must be runnable on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). One option is to SSH into a machine. Learn about SSH at these links: `https://cseit.umn.edu/knowledge-help/learn-about-ssh`, `https://cseit.umn.edu/knowledge-help/choose-ssh-tool`, and `https://cseit.umn.edu/knowledge-help/remote-linux-applications-over-ssh`.

Instructions

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**

  1. hw4.pdf: The report that contains the solutions to Problems 1, 2, and 3 including the summary of methods and results.
  2. `adaboost.py`: Code for Problem 1.
  3. `rf.py`: Code for Problem 2.
  4. `kmeans.py`: Code for Problem 3.
  5. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
  6. Any other files, except the data, which are necessary for your code.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.