Parallel Exercises for Project D

The coding part of Project D consists of completing questions Q1 . . . Q6 of UC Berkeley's Pac-Man **Project 5 (Classification)**. In parallel, you are required to write a lab-report addressing the answers to the exercises below. General information about expectations and grading can be found on the projects page of the course website.

Most exercises correspond to the questions Q1 . . . Q6 and can only be answered upon implementation of those questions. Some exercises cover multiple questions, so make sure to read all of them before proceeding with the first one!

Lab-report

Write a report with answers to the exercises below. Be sure to state your answers clearly and to explain and/or motivate all your answers, choices and ideas. Your answers should be a convincing proof of understanding and include, where relevant, concrete experimental results. The projects page on the course website gives an explanation of what we expect concerning the questions that require experiments; please review this information carefully prior to starting the experiments!
You can obtain a maximum of 90 points for the lab-report.

Do not forget to include your names and studentnumbers in the report!

The purpose of the first question is to get further acquainted with the classification framework; answer the questions in enough detail to convince us that your understanding suffices for successful completion of the project.

Exercise 1

Study the files `dataClassifier.py`, `mostFrequent.py`, and `naiveBayes.py`.

a. Run the `dataClassifier` code on the two above-mentioned classifiers using the digit data (= default). Write down for each classifier the validation and testing accuracy, as well as the value of the smoothing parameter used by naive Bayes. (These results will be your baseline for further comparison.) Give an explanation for the difference in performance of the two classifiers.

b. The smoothing parameter in naive Bayes training is used to perform *Laplace smoothing*. Explain in your own words what the smoothing does and why it is used (Google 'laplace smoothing and naive bayes').

c. Train the naive Bayes classifier with the `-- autotune` option. Explain what this option does. In addition, report the validation and test accuracy you find and the value of k to which they correspond. Finally, compare the performance of this naive Bayes classifier with that of part a. (is it better/worse/comparable?) and explain the difference or similarity.

d. In class we assumed that a naive Bayes classifier determines the most likely class based upon a posterior distribution $p(C \mid F_1, \ldots F_n)$. In the file `naiveBayes.py` you can see that the naive Bayes classifier computes the $\log$ of the posteriors rather than pure posteriors. Use the number of features in the digit data to explain why computing the $\log$ is a wise thing to do.

e. Note that upon training the above-mentioned classifiers, a validation set is used in addition to the training set. Subsequently the classifier is tested on a test set. Explain, in your own words, the general difference between accuracy on a validation set and on a test set.

Points: a: 5; b: 4; c: 5; d: 2; e: 4

Exercise 2

For Q1:

a. Report validation and testing accuracy of the perceptron on the digit data after i training iterations, for $i = 1, 2, 3, 4, 5, 6, 10, 50$. Also mention in which order your code goes through the training examples. Is your perceptron better than the naive Bayes classifier? Provide a clear argument to support your claim.

b. Based upon the above numbers, when would it be safe to stop training (if safe at all)? Clearly motivate your answer.

c. Give a drawing or clear description of the multi-class perceptron learnt for the digit data. Your description or image should show, if applicable: input nodes, output nodes, hidden nodes, their connections, weights, activation functions, thresholds/biases, etc. (you just don't have to provide actual numbers for the weights: those are only available after training).

Points: a. 4; b. 3; c. 5

Exercise 3

In Q2 you analyse your learnt perceptron on digits data.

a. Clearly motivate your choice for weight sequence 'a' or 'b' in `answers.py`.

b. Given the type of functions a perceptron is capable of learning, is it in theory a suitable instrument for classifying digits? Argue why or why not.

Points: a. 2; b. 2

Exercise 4

In Q3 you train, tune and analyse a MIRA classifier on the digits data.

a. In class we included a learning rate α in the perceptron learing rule; in the perceptron learning rule implemented for Q1 you in fact used $\alpha = 1$. The MIRA classifier uses a *dynamic* 'margin-dependent' learning rate, called τ, which limits the amount of change applied to the weights. Explain why limiting the weight-change is a wise thing to do, especially for domains with a large number of features.

b. Clearly explain what the option `--autotune` does in relation to the constant C used by MIRA. In addition, report the validation and test accuracy you find and the value of C to which these correspond. Finally, compare the performance of your MIRA classifier with the perceptron and the (best) naive Bayes classifier: is it better/worse/comparable? Provide clear arguments to support your claims.

c. Similar to the perceptron in Q2 you can determine the 100 highest weight features for each label and visualize them. Present images of the results for the MIRA classifier, and compare them to those of the perceptron in Q2. What kind of differences and similarities do you see? Can you explain these? (Note that you have to copy a piece of code from the perceptron; this adjustment of mira.py is allowed.)

Points: a. 2; b. 6 c. 3

Exercise 5

In Q4 you reconsider a naive Bayes classifier on the digits data.

a. Is a naive Bayes classifier with its underlying independence assumptions in theory a suitable instrument for classifying digits? Provide clear arguments to support your claim.

b. Execute `python dataClassifier.py -d digits -c naiveBayes -o -1 4 -2 2`. Clearly explain what the command does, what *odds* are and how to interpret the output.

c. Design and describe (at least) four features that can help distinguish between (and therefore classify) digits (note that you do not necessarily have to implement them all!). Clearly explain why you believed at design time that your features could improve classification.

d. Which features did you in fact implement? And what is the test performance of your classifier upon executing

   ```
   python dataClassifier.py -d digits -c naiveBayes -f -a -t 1000?
   ```

   Has the performance of your naiveBayes classifier improved as a result of the implemented features? Provide clear arguments to support your claim.

Points: a. 2; b. 2; c. 12; d. 3

Exercise 6

In Q5 you train a perceptron for pacman and in Q6 you attempt to improve the perceptron by trying to clone the behaviour of a number of predefined agents.

a. In Q5, what is the validation and test performance of your baseline perceptron classifier given by executing

   ```
   python dataClassifier.py -d pacman -c perceptron?
   ```

b. For each of the following agents, design and describe at least the number of indicated features (note that you do not necessarily have to implement them all!). In addition, clearly explain why you believed at design time that these features would capture the intented behaviour of the corresponding agents (to at least some extent).

   1. **StopAgent**: 1 feature
   2. **FoodAgent**: 1 feature
   3. **SuicideAgent**: 2 features
   4. **ContestAgent**: 3 features

c. Which features did you in fact implement? Discuss, for each of the agents individually, whether or not the use of the implemented feature(s) increases the performance of your classifier. Provide clear arguments to support your claims.

Points: a. 0 (-1 if missing); b. 12; c. 12

Exercise 7

List the *autograder* scores for all the project questions:

Q1	Q2	Q3	Q4	Q5	Q6	Total
x/4	x/1	x/6	x/6	x/4	x/4	$\sum_{\mathrm{Qi}}$ x / 25 = . . .

Note that we will run the autograder on your code to verify these scores!

Points: 0 (-1 if unanswered)