Homework 4

NLP/ML/Web - Spring 2015 - Andrew Rosenberg

Due Monday, May 11 at 11:59pm

Problem 1) Ensemble Methods for Web Page Classification

The task in this homework is to explore the use of ensemble methods for Web Page Classification. This work assumes the successful completion of Homework 1. Web page classification.

1a. [From HW1] Identify a class label with at least five classes for web documents. Include clear definitions of each of these. Possible categories include "author", "date published", "product type based on reviews", "politics vs. sports vs. etc.".

1b. [From HW1 – you need not download additional instances of these categories.] Download at least 150 instances of each category.
Identify 100 as training, and 50 as testing. Generate labels for these documents these documents for each category. These labels may be generated by hand, or automatically (i.e. all web pages that contain "by Adam Platt" are categorized as "Adam Platt". NB: if you are doing something like this, make sure that you remove this identifier before the classification routine.)

Details:

- Store each of these webpages in a single directory.
- Create two plain text files one for training, train_labels.txt and one for testing, test_labels.txt.
- These text files should contain the filenames and associated label for all of the downloaded pages. One per line, separated by a comma.

E.g.

```
http:__cheezburger.com_7790385920,CAT
http:__www.dailypuppy.com_puppies_spud-the-labrador-retriever_2013-09-16,DOG
http:__en.wikipedia.org_wiki_Giant_panda,PANDA
```

1c. Write a training program that, given these web pages and labels produces a set of classifiers in an ensemble and aggregation parameters.

You may use scikit learn, weka or other classification tools. You may also write your own classifier.

Most important component here is that you write feature extraction routines to generate a **document representation**.

Generating an ensemble using sampled data.

Identify the best feature representation from Homework 1. Using this feature representation, train $N \geq 2$ classifiers each using $1/N$ of your available training data.

Construct an ensemble learner from these N classifiers aggregating predictions using 1) min, 2) max and 3) weighted majority voting.

Evaluate the performance of these classifiers with at least 2 values of N.

Early vs. Late Fusion.

In homework 1, K distinct feature representations were evaluated.

Early Fusion: Train a classifier using a feature vector that is the combination of all K feature representations.

Late Fusion: Using these K representations to train K classifiers. Construct an ensemble learner from these K classifiers aggregating predictions using 1) min, 2) max, and 3) weighted majority voting.

Evaluate the performance of these classifiers.

DETAILS:

- This program should be called with (minimally) three command line parameters in this order: 1) the training label file, train_labels.txt, and 2) the data directory, something like '/home/andrew/nlpmlweb/hw1/data/' and 3) a filename for the generated classifier. If additional files need to be read, they should be included in the command line after these three. A description of these files must be included in the README.

1d. Write a testing program that evaluates a set of testing files using a trained ensemble classifier.

This program should generate two outputs. 1. A measure (or measures) of overall classification performance. Accuracy is an excellent choice here. 2. A listing of predictions for each data point.

DETAILS:

- This program should be called with (minimally) four command line parameters in this order: 1) the testing label file, test_labels.txt, and 2) the data directory, something like '/home/andrew/nlpmlweb/hw1/data/' and 3) a filename for the trained classifier, and 4) a filename to write predictions to with an indication of if it is correct or not.

If additional files need to be read, they should be included in the command line after these four. A description of these files must be included in the README.

- The overall classification performance should be written to the command line.
- The format of the prediction file should be a comma separated value file containing, the file stem and true label (from the labels file), the predicted label, and an indicator to indicate whether it was correct or not ('+' for correct, '-' for incorrect).

E.g.

```
http:__cheezburger.com_7790385920,CAT,DOG,-
http:__www.dailypuppy.com_puppies_spud-the-labrador-retriever_2013-09-16,DOG,DOG,+
http:__en.wikipedia.org_wiki_Giant_panda,PANDA,DOG,-
```

Deliverables

- All source code and libraries (or pointers to download) required for your project.
- A README/Report file whose contents are described below.
- All required configuration files (including training_labels.txt and testing_labels.txt).

The README can be in any electronic format (txt, pdf, doc, google docs, open office) and should minimally include the following

- A list and description of every file included in the submission – including which matrix library (if any) you use.
- A description of how your code is compiled (if it is compiled)
- A description of how to run your code.
- A high level description of the task – here: A description of the data and the types of document representations you are exploring. This should be just a few paragraphs long, and doesn't necessarily need a lot of mathematical notation. It should be understandable to a reasonably informed reader.
- A report of the experiments. How do the ensemble methods compare with the homework 1 experiments?Is one better than the others? Do they have different strengths that are observable by its performance on particular files? In particular settings? For your experiment, is early fusion better than late fusion? Is data or feature ensemble generation.
- Any other points of interest – running time, complexity, unique qualities of your implementation, etc.

Written Response to a Paper: Read "PPDB: The Paraphrase Database" by Juri Ganitkevitch, Benjamin Van Durme and Chris Callison-Burch. It is linked to from the course webpage. Write a short 1-2 page response to the paper. This response should 1) summarize the core aspects of the paper, 2) thoughtfully critique the work and 3) provide any contextualization of this work to either other papers, research you've read, or material we've discussed in class.

Grading:

- **10 points - Compilation** Each file must compile without error or warning into an executable as described above. (Note: for python, no points are awarded for compilation, but execution is worth 30 points.)
- **10 points - Execution** Each executable must run without error or warning on valid input using the command line parameters described above. (Note: for python, no points are awarded for compilation, but execution is worth 30 points.)
- **15 points - README** Does your README documentation completely and accurately describe the task and approach taken? Does it satisfy the content requirements (i.e. how to compile and run your project, file listing, etc.)
- **7 points - Within Code Documentation** Every function should include a comment minimally describing 1) what it does, 2) what its inputs are and 3) what its output is. Are there effective other comments throughout the code? You may use a javadocs, or pydoc, or other standard. For a good read check out the google style guides: `https://code.google.com/p/google-styleguide/`.
- **8 points - Style** Is the structure of your program clear and coherent? Are functions and variables given self-explanatory names? Are functions used to aid intelligibility of the code? Are functions used to reduce repeated blocks of code? Is indentation, spacing, use of parentheses, use of braces consistent, and sensible? For example, if you use brackets on the same line at the start of a block, always do so. If you place a brace on the line following the start of the brace, always do so. If you put a space between variables and operators, e.g. `if (i == j)`, always do so. So, `if (i == j) i = j+k;` is bad. It should be `if (i == j) i = j + k;` or if you prefer `if (i==j) i=j+k;`. You will be graded on consistency in these decisions, not on any particular style.
- **25 points - Correctness** Is/Are the algorithm(s) implemented correctly? Have an appropriate number of word/document representations been used and used correctly?
- **5 points - Instructor's Discretion** Has this assignment gone beyond the minimal requirements in a substantive way? Is it especially clear? Is the code especially well written? Is the response particularly thoughtful or insightful? Have non-trivial representations been examined?

- **20 points - Written Response** Does the written response to the paper reflect 1) Understanding of the paper 2) Thoughtful assessment and reasonable criticism and 3) Appropriate contextualization to other work or ideas discussed in class?