homework 2: Some Linear Classification Models

CSE 142: Machine Learning

University of California, Santa Cruz

Please do this assignment on your own. Discussing concepts with your classmates is fine, but please do not share any code.

For this assignment, the primary goal is to get comfortable working with some of the linear classification models that we've discussed in class so far, as well as thinking a little bit about feature engineering.

For your code, you can either do your implementation in a Jupyter notebook, or standalone Python code.

The deliverables for this assignment will be your code, as well as a written report of your findings from your programming. Half of your grade will depend on the quality of the report, which you ought to submit as a PDF. For the full experience, you might like to typeset your scientific writing using LATEX. Some free tools that might help: Overleaf (online), TexLive (cross-platform), MacTex (Mac), and TexStudio (Windows).

Datasets and problems to solve

For this homework, you'll be working with the Spambase dataset from the UCI Machine Learning Repository (available at https://archive.ics.uci.edu/dataset/94/spambase), as well as textual data in the form of the Universal Declaration of Human Rights, the text of which is included in the zip file for the assignment, in English and Dutch.

- For the Spambase dataset, you are working with a binary classification problem; use the given features to determine whether an email was spam or not spam.
- For the Universal Declaration of Human Rights dataset, you are also doing a binary classification problem, distinguishing between English and Dutch text.

Programming: implementing the Perceptron algorithm

Similarly to how we implemented the Perceptron training algorithm in class, implement Perceptron training, such that...

- Given a training set X, y (a matrix where the rows are example instances and a vector of corresponding labels) and a maximum number of training iterations, you return the appropriate trained weights and biases.

- Be able to run that trained model on a test set, and be able to report accuracy and a confusion matrix, given a specific test set.

Programming: getting comfortable with linear classifiers in scikit-learn

Learn how to use the Scikit-learn library, specifically the SVM class `sklearn.svm.LinearSVC`, and the logistic regression class `sklearn.linear_model.LogisticRegression`.

Take some time to read through the documentation for these classes and look at their related examples in the docs.

Feature engineering and feature extraction

In class on Friday the 18th, we did a demo on language identification to tell the difference between English and Dutch text by turning each line of the file (one sentence) into a vector of length 26, where each entry in the vector simply counted the number of times the corresponding letter ('a' or 'A' for the initial entry in the vector, etc), and initial experiments seemed to indicate that this worked well.

But! People in class immediately had some intuitive sense about how Dutch text differs from English text just by looking at it – perhaps you can come up with some good features to extract! Count up how many times certain digraphs show up, perhaps? Is "th" very common in English? How about repeated vowels in Dutch? Maybe you could even look for whole words. Come up with some hypotheses and see if you can come up with features that are useful for this classification problem.

Experiments

For the Spambase dataset, set aside 10% of the dataset to be your test set, 10% to be your development set, and the rest to be your training set.

For the Universal Declaration of Human Rights text, treat the entire available text as your training set, but go collect 20 more sentences for both Dutch and English to be your development set, and another 20 sentences (for each language) to be your test set. You should include these sentences in your turn-in. You'll want to report results with the feature set that you came up with, and optionally also with the letter-count features we tried in class.

Then, for both of the datasets, try out your perceptron implementation, SVMs, and logistic regression. You may want to try out some different hyperparameters – how many iterations of Perceptron training work well on your development

set? Should you shuffle your training data? What kinds of regularization seem sensible for Logistic Regression? How about setting `C` for your SVM?

Report

Your report should include an explanation of the code that you wrote, your rationale for the features you've designed, and an explanation for what happened in your experiments. Describe your results, and explain which methods worked the best for which problems.

In the writeup, be sure to describe your models and experimental procedure. Provide some results on your test sets, ideally with some tables or graphs. Organize your report as neatly as possible, and articulate your thoughts as clearly as possible. We prefer quality over quantity here. When discussing the experimental results, do not copy and paste the entire system output directly to the report – make some nice tables or figures to report your results.

Deliverables

You should turn in your code along with a README, explaining how how to run each of the variants you developed, as well as your report as a PDF.

Submission Instructions

Submit a zip file (hw2.zip) on Canvas, containing the following:

- Code: You code should be implemented in Python 3, and needs to be runnable. Submit your code together with a neatly written README file explaining how to run your code with different settings. Follow good software engineering practice here – code is meant to be read!
- Report: As noted above, your writeup should be in PDF; please put your name at the top.

References

1. https://archive.ics.uci.edu/dataset/94/spambase
2. https://unric.org/en/human-rights-75/
3. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
4. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html