homework 3: Naïve Bayes and Decision Trees

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. We've implemented some of these algorithms live in class, but you ought to turn in your own implementation, code that you typed yourself – code that's "in your own words", so to speak.

For this assignment, we're doing some programming – we'll be implementing simple Naïve Bayes and decision tree classifiers, and applying them to two small datasets from the UC Irvine Machine Learning Repository, "Mushroom" and "Congressional Voting Records".

For your code, you can either build 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. One third 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 "Mushroom" dataset [0]
- The "Congressional Voting Records" dataset [1]

For both of these datasets, we'll do a binary classification problem; for the first, the problem is to determine whether a given mushroom is edible or not, and for the second, the goal is to determine the party (democrat, republican) of a specific congressional representative.

Programming: implementing Naïve Bayes and simple Decision Trees

Similarly to how we implemented the Perceptron training algorithm in the previous homework, for this homework we'll implement NB and decision tree classifiers. Don't use any external libraries for this other than numpy and pandas for your classifiers, although feel free to use other libraries for visualizing your datasets and results.

What does it mean to have implemented these classifiers? You ought to implement code (perhaps two distinct Python classes) so that...

- Given a training set X, y (a matrix where the rows are example instances and a column vector of corresponding labels), you should be able to fit NB and decision tree classifiers for that training set.
- Note that for these datasets, all of the features we are considering are categorical (although there are missing values – you can treat a missing value as its own distinct category if you like), so fitting a simple probability distribution ("count and divide") in the NB case and using information gain over categorical variables like we did in class should work well for these.
- Be able to run your trained model on a test set, and be able to report accuracy and a confusion matrix, given a specific test set.

We talked about the algorithm for building a decision tree classifier a bit in class – you're going to have to come up with some representation for what it means to have a decision tree. Perhaps you'll make a Python class that contains a nested structure indicating which feature to inspect at each node in the tree, and what to do if that feature takes on each possible value, linking to nested subtree, or a final result at the leaves. This is up to you to design!

Experiments

For both the Mushroom and Congressional Voting Records datasets, set aside 10% of the available data to be your development set, 10% to be your test set, and the rest to be your training set.

Then, for both of the datasets, try out your implementations of Naïve Bayes and Decision trees. For Decision Trees, you might want to try out some different hyperparameters, in the form of different stopping criteria for recursively splitting your data – you can try different stopping criteria by testing your classifier on the development set.

Does it matter whether you shuffle your training data before fitting your models? Would it help if you do any smoothing, for the probability distributions in the Naïve Bayes model?

How do your implementations compare, in their classification performance, with the implementations in scikit-learn? You might want to consider `sklearn.naive_bayes.CategoricalNB` and `sklearn.tree.DecisionTreeClassifier` here.

Report

Your report should include an explanation of the code that you wrote and how it works 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 (hw3.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

- https://archive.ics.uci.edu/dataset/73/mushroom
- https://archive.ics.uci.edu/dataset/105/congressional+voting+records