CSE 353– Homework II

Dr. Ritwik Banerjee

Due by Apr 24, 2019 [11:59 pm]

1 Theory

I: Logistic regression and naïve Bayes **12 points**

Suppose in a binary classification problem, the input variable $\mathbf{x}$ is n-dimensional and the ouput is a binary class label $y \in \mathcal{Y} = \{0, 1\}$. In this situation, there is an interesting connection between two learners: *logistic regression* and *naïve Bayes classifier*.

(a) Write down the expressions for the class conditional probability for each class, i.e., $P(y = 1|\mathbf{x})$ and $P(y = 0|\mathbf{x})$, for logistic regression. (2 points)

(b) Using Bayes' rule, derive the posterior probabilities for each class, i.e., $P(y = 1|\mathbf{x})$ and $P(y = 0|\mathbf{x})$, for naïve Bayes. (2 points)

(c) Assuming a Gaussian likelihood function in each of the n dimensions, write down the full likelihood function $f(\mathbf{x}|d)$ for naïve Bayes. (2 points)

(d) Assuming a uniform prior on the two classes and using the results from parts (b) and (c) above, derive a full expression for $P(y = 1|\mathbf{x})$ for naïve Bayes. (3 points)

(e) Show that with appropriate manipulation and parameterization, $P(y = 1|\mathbf{x})$ in naïve Bayes from part (d) is equivalent to $P(y = 1|\mathbf{x})$ for logistic regression in part (a). (3 points)

II: Failure of Cross-Validation **4 points**

Cross-validation works well in practice, but there are some pathological cases where it might fail. Suppose that the label is chosen at random according to $P[y = 1] = P[y = 0] = 1/2$. Consider a learning algorithm that outputs the constant predictor $h(x) = 1$ if the number of 1s in the training set labels is odd, and $h(x) = 0$ otherwise. Prove that the difference between the leave-one-out estimate and the true error in such a case is always $1/2$.

III: Decision Tree **4 points**

Show that any binary classifier $h : [0, 1]^d \to \{0, 1\}$ can be implemented as a decision tree of height at most $d + 1$, with internal nodes of the type "Is $x_i = 0$?" for some $i \in \{1, 2, \ldots, d\}$.

2 Experiment

I: Decision Tree **40 points**

In this section, you will be working with a somewhat morbid dataset. This is a dataset of passengers on the Titanic. Each row in this dataset has 12 columns, where the second column, "Survived", is what we would like to estimate using a machine learning algorithm.

Some of the features are obvious from their names (i.e., column headers), the others are:

- `pclass` – ticket class (1st, 2nd, or 3rd).
- `sibsp` – the number of siblings/spouses aboard the Titanic.
- `parch` – the number of parents/children aboard the Titanic (some children traveled with a nanny, so `parch = 0` for them).
- `ticket` and `cabin` are just the ticket number and the cabin number, respectively.
- `age` – age in years (fractional age indicates that age was less than 1 year).
- `embarked` – three different ports of embarkation were (S)outhampton, (Q)ueenstown, and (C)herbourg.

Your task is write your own code in Java or Python to implement the decision tree algorithm we learnt in class (ID3). Instead of training and testing on the same data, however, you must split the data in a 60-40 ratio to train on the 60% sub-dataset and test on the remaining 40%. You must also calculate the accuracy of prediction on the training data ***and*** the test data, and plot the accuracy numbers as the tree depth increases. This should lead to a plot that looks like the one in the slide on overfitting when we covered "Decision Trees" in class.

Note that in this code, you will have to

- handle continuous variables, and
- avoid overfitting

Your submission for this task should be your code for learning and testing the decision tree. This code should be able to perform the training and testing tasks separately, and the user should be simply able to train on the 60% sub-dataset by doing something like

```
$ python id3.py --dataset /path/to/data/filename.csv --train
```

and test on the remaining 40% by

```
$ python id3.py --dataset /path/to/data/filename.csv --test
```

You may assume that the number 60 and 40 are fixed, and the user will not change them around.

II: Final Report **20 points**

Along with your code, you are also required to submit a short report (no more than 1 page)[1]. The report must contain the following:

- The accuracy plot of the decision tree model (the tree depth and accuracy on the x- and y-axes, respectively. The plot must be clear enough the reader to distinguish between, say, 0.85 and 0.9. Otherwise, include a table with the accuracy numbers in addition to the plot.
- A brief discussion (at most a half-page) about what you observed regarding (i) the depth at which the tree started overfitting, (ii) the most discriminatory features, and (iii) the least discriminatory features.
- Also include a brief discussion about whether some features that you expected to be completely useless (e.g., ticket or cabin number) surprised you by exhibiting predictive power (or vice versa . . . i.e., some features that you thought would be very predictive actually turned out to be useless).

[1]For the report, please stick to single line spacing.

- A **README** section explaining how to run your code. Keep in mind that the grader will only run the code, and not do any manual work (e.g., placing the dataset in the same folder, etc.). So, make sure that your code runs "as is" once the path to the entire dataset[2] has been specified.

The report will be graded based on (a) performance details, (b) replicability of experiments, (c) explanation of how to run your code, and (d) the quality of the discussions about each algorithm.

Notes

What programming languages are allowed?

1. Java (JDK 1.8 or above, but don't use Java 11), Python (3.x).
2. You are NOT allowed to use any data science and/or machine learning library for this assignment. If you have doubts about whether the use of a library is acceptable, please ask before assuming that it can be used!
3. Calculations for entropy, information gain, etc. must be your own code.
4. You are NOT allowed to use any libraries that provide decision tree data structures off-the-shelf.
5. For the accuracy plot, you may use any library you want as long as the use of that library (e.g., `pandas`) is strictly restricted to the generation of the plot as an image file, and has nothing to with the rest of your code.

What should we submit, and how?

Submit a single `.zip` archive containing (i) one folder simply called "code", containing all your code, (ii) a PDF document for your report, and (iii) a PDF document for the theory part of your submission. Please do **NOT** handwrite the solutions and scan or take pictures. For the PDF documents, use either LaTeXor MS Word to write the solutions, and export to PDF. Anything else will not be graded.

[2] Remember that just like the previous assignment, the grader will also not perform the 60-40 split in any way; this should be done by your code.