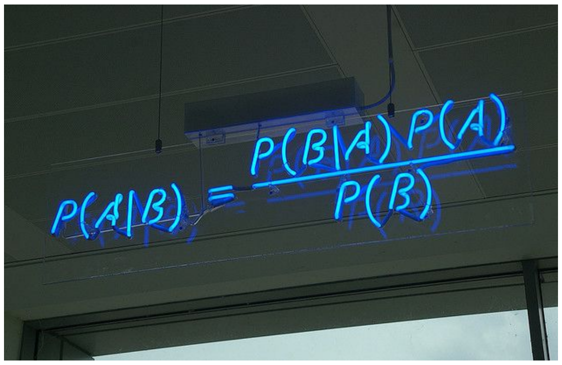
Homework Programming Assignment 3: Incremental Bayesian Learning

Due Time: October 21, 2019, 3:00PM



Introduction

Incremental learning is a method of machine learning in which input data is continuously used to extend the existing model's knowledge. It can be applied when training data becomes available gradually over time or its size is out of system memory limits. In this assignment, you will implement a Naive Bayes classifier which can learn incrementally (i.e. without seeing all the instances at once).

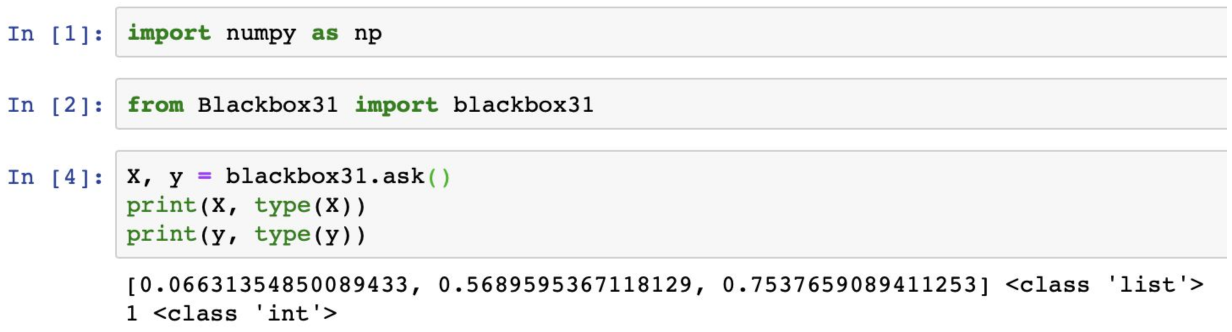
1. Homework Description

Please use ***Python 3*** to implement your homework assignment.

1.1 Blackbox31

In this assignment, you are given `Blackbox31.pyc` instead of ready-made data, and you need to generate your own data from that blackbox to simulate data stream.

Example of getting data from `Blackbox31.pyc,` in your `NaiveBayes.py:`

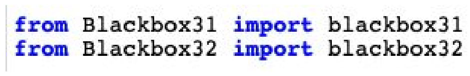


```
In [1]: import numpy as np

In [2]: from Blackbox31 import blackbox31

In [4]: X, y = blackbox31.ask()
        print(X, type(X))
        print(y, type(y))

        [0.06631354850089433, 0.5689595367118129, 0.7537659089411253] <class 'list'>
        1 <class 'int'>
```

Each time you `ask` the blackbox, it will randomly return a tuple (X, y) back. As you can see in the above image, X is a list and y is an integer, together X and y form ***one*** single sample.

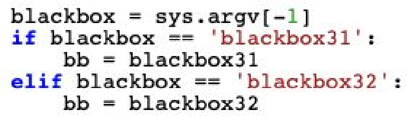
Observations ***X*** has 3 attributes, all attributes have continuous value and are in the range [0.0, 1.0), and target ***y*** has 3 possible values: **0, 1 and 2**.

Your python script submitted to Vocareum **must import both** blackbox31 and blackbox32 even though blackbox32 is not provided to you.



```
from Blackbox31 import blackbox31
from Blackbox32 import blackbox32
```

Remember that your code will be run with 2 different boxes. One way to parse the input argument is



```
blackbox = sys.argv[-1]
if blackbox == 'blackbox31':
    bb = blackbox31
elif blackbox == 'blackbox32':
    bb = blackbox32
```

and then use `bb` as data source. When you develop your algorithm, you only need to care about blackbox31, but you also need to import and add additional parsing logic for blackbox32 since blackbox32 will be used to test your code on Vocareum.

1.2 Noisy data

This time the data includes noises, that means even the features of a sample(X) shows that the sample should belongs to class C_i(y), it might still return a wrong class C_j when you ask the blackbox. However, Bayesian model is not quite sensitive to noises, so that should not be a big program (hopefully). You are also encouraged to try other models like decision tree and Neural Network to compare the differences.

1.3 Task Description

Your task is to classify this categorical data into ***3*** classes **incrementally** and keep track of the testing accuracy statistics.

You need to hold out 200 samples as testing data, and then repeatedly generate training data one at a time during the learning process for 1000 times. Note that not all training data examples are known at once, it comes one at a time.

The logic of your program should be something like this:

```
# hold out 200 examples to estimate accuracy
Repeat 200 times:
    Ask the blackbox for one data point

# do incremental training
Repeat 1000 times:
    X, y = blackbox31.ask()
    adapt model to this new X, y
    accumulate test accuracy stats per 10 samples

# output accuracy stats
```

Your program will be run in the following way:

```
python3 NaiveBayes.py blackbox31
=> results_blackbox31.txt
```

When we grade your model with hidden blackbox32, it should be run:

```
python3 NaiveBayes.py blackbox32
=> results_blackbox32.txt
```

The `results.txt` contains accuracy stats, it should have the following format:

```
 10, 0.545
 20, 0.545
 30, 0.595
 40, 0.66
 50, 0.66
    …
970, 0.81
980, 0.82
990, 0.82
1000, 0.815
```

The first column indicates the number of training data that have been seen so far, and the second column is the corresponding test accuracy (rounded to 3 decimals).

Since the data will be randomly generated, it is acceptable that sometimes your classifier does not give very ideal results and you will not lose points for that.

In your implementation, ***please do not use any existing machine learning library call***. You must implement the algorithm yourself. Please develop your code yourself and do not copy from other students or from the Internet.

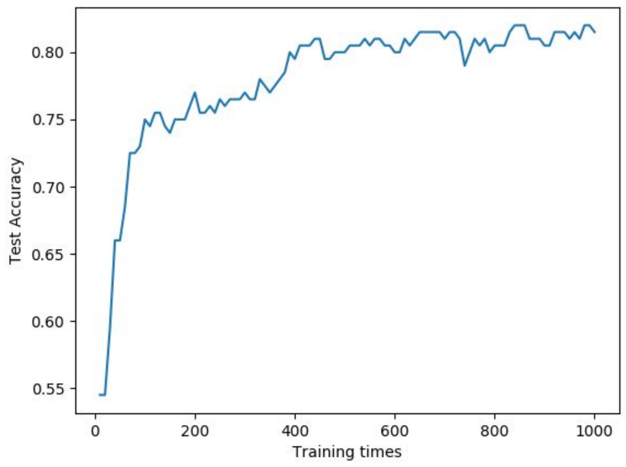
When we grade your algorithm, we will use a different blackbox. Your code will be autograded for technical correctness. Please name your file correctly, or you will wreak havoc on the autograder. **The maximum running time is 3 minutes.**

2. Submission:

2.1 Submit your code to Vocareum

- Submit `NaiveBayes.py` to ***Vocareum***
- After your submission, Vocareum would run two scripts to test your code, a submission script and a grading script. The submission script will test your code with ***only blackbox31***, while the grading script will test your code with another ***blackbox32***.
- The program will terminate and fail if it exceeds the ***3 minutes*** time limit.
- After the submission script/grading script finishes, you can view your submission report immediately to see if your code works, while the grading report will be released after the deadline of the homework.
- You don't have to keep the page open while the scripts are running.

2.2 Submit your report to Blackboard

- Create a single **`.zip`** (`Firstname_Lastname_HW3.zip`) which contains:
  - `NaiveBayes.py`
  - `Report.pdf`, a ***brief*** report contains your ***testing accuracy graph*** for blackbox31***.*** Example graph:
  - 

- Submit your zipped file to the blackboard.

3. Rubric:

100 points in total

- Program correctness(60 points): program always works correctly and meets the specifications within the time limit
- Documentation(20 points): code is well commented and submitted graph is reasonable
- Performance (20 points): the classifier has reasonable performance