Indian Institute of Technology Kanpur
CS771 Introduction to Machine Learning

Instructor: Purushottam Kar
Date: 05 March, 2023
Total: 40 marks

ASSIGNMENT 2

1 What should I submit, where should I submit and by when?

Your submission for this assignment will be one PDF (.pdf) file and one ZIP (.zip) file. Instructions on how to prepare and submit these files is given below.

Assignment Package:
`https://www.cse.iitk.ac.in/users/purushot/courses/ml/2022-23-w/material/assn/assn2.zip`
Deadline for all submissions: 27 March, 2023, 9:59PM IST
Code Validation Script: `https://colab.research.google.com/drive/1_qlxLBXcauDRwHGIZUcppUEMtV92SR9j?usp=sharing`
Code Submission: `https://forms.gle/yxK4Psa5sxpmCgbz9`
Report Submission: on Gradescope
There is no provision for "late submission" for this assignment

1.1 How to submit the PDF report file

1. The PDF file must be submitted using Gradescope in the *group submission mode*. Note that this means that auditors may not make submissions to this assignment.

2. Make only one submission per assignment group on Gradescope, not one submission per student. Gradescope allows you to submit in groups - please use this feature to make a group submission.

3. Ensure that you validate your submission files on Google Colab before making your submission (validation details below). Submissions that fail to work with our automatic judge since they were not validated will incur penalties.

4. Link all group members in your group submission. If you miss out on a group member while submitting, that member may end up getting a zero since Gradescope will think that person never submitted anything.

5. You may overwrite your group's submission (submitting again on Gradescope simply overwrites the old submission) as many times as you want before the deadline.

6. Do not submit Microsoft Word or text files. Prepare your report in PDF using the style file we have provided (instructions on formatting given later).

1.2 How to submit the code ZIP file

1. Your ZIP file should contain a single Python (.py) file and nothing else. The reason we are asking you to ZIP that single Python file is so that you can password protect the ZIP

file. Doing this safeguards you since even after you upload your ZIP file to your website, no one can download that ZIP file and see your solution (you will tell the password only to the instructor). If you upload a naked Python file to your website, someone else may guess the location where you have uploaded your file and steal it and you may get charged with plagiarism later.

2. We do not care what you name your ZIP file but the (single) Python file sitting inside the ZIP file must be named “submit.py”. There should be no sub-directories inside the ZIP file – just a single file. We will look for a single Python (.py) file called “submit.py” inside the ZIP file and delete everything else present inside the ZIP file.

3. Do not submit Jupyter notebooks or files in other languages such as C/Matlab/Java. We will use an automated judge to evaluate your code which will not run code in other formats or other languages (submissions in other languages may simply get a zero score).

4. Password protect your ZIP file using a password with 8-10 characters. Use only alphanumeric characters (a-z A-Z 0-9) in your password. Do not use special characters, punctuation marks, whitespaces etc in your password. Specify the file name properly in the Google form.

5. Remember, your file is not under attack from hackers with access to supercomputers. This is just an added security measure so that even if someone guesses your submission URL, they cannot see your code immediately. A length 10 alphanumeric password (that does not use dictionary phrases and is generated randomly e.g. 2x4kPh02V9) provides you with more than 55 bits of security. It would take more than 1 million years to go through all $> 2^{55}$ combinations at 1K combinations per second.

6. Make sure that the ZIP file does indeed unzip when used with that password (try
   `unzip -P your-password file.zip`
   on Linux platforms).

7. Upload the password protected ZIP file to your IITK (CC or CSE) website (for CC, log on to `webhome.cc.iitk.ac.in`, for CSE, log on to `turing.cse.iitk.ac.in`).

8. Fill in the Google form linked above to tell us the exact URL for the file as well as the password.

9. Make sure that when we visit the URL you have given, there is actually a file to be downloaded. Test your URL on a browser window to verify that you have not submitted a corrupted URL.

10. Do not host your ZIP submission file on file-sharing services like GitHub or Dropbox or Google drive. Host it on IITK servers only. We will autodownload your submissions and GitHub, Dropbox and Google Drive servers often send us an HTML page (instead of your submission) when we try to download your file. Thus, it is best to host your code submission file locally on IITK servers.

11. While filling in the form, you have to provide us with the password to your ZIP file in a designated area. Write just the password in that area. For example, do not write “Password: helloworld” in that area if your password is “helloworld”. Instead, simply write “helloworld” (without the quotes) in that area. Remember that your password should contain only alphabets and numerals, no spaces, special or punctuation characters.

12. While filling the form, give the complete URL to the file, not just to the directory that contains that file. The URL should contain the filename as well.

    (a) Example of a proper URL:
    `https://web.cse.iitk.ac.in/users/purushot/mlassn1/my_submit.zip`

    (b) Example of an improper URL (file name missing):
    `https://web.cse.iitk.ac.in/users/purushot/mlassn1/`

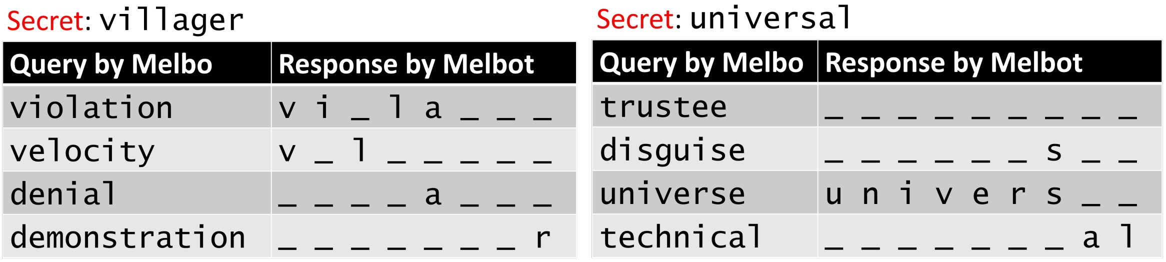
    (c) Example of an improper URL (incomplete path):
    `https://web.cse.iitk.ac.in/users/purushot/`

13. We will use an automated script to download all your files. If your URL is malformed or incomplete, or if you have hosted the file outside IITK and it is difficult for us to download automatically.

14. Make sure you fill in the Google form with your file link before the deadline. We will close the form at the deadline.

15. Make sure that your ZIP file is actually available at that link at the time of the deadline. We will run a script to automatically download these files after the deadline is over. If your file is missing, we will treat this as a blank submission.

16. We will entertain no submissions over email, Piazza etc. All submissions must take place before the stipulated deadline over the Gradescope and the Google form. The PDF file must be submitted on Gradescope at or before the deadline and the ZIP file must be available at the link specified on the Google form at or before the deadline.



Secret: villager

Query by Melbo	Response by Melbot
violation	v i _ l a _ _ _
velocity	v _ l _ _ _ _ _
denial	_ _ _ _ a _ _ _
demonstration	_ _ _ _ _ _ _ r

Secret: universal

Query by Melbo	Response by Melbot
trustee	_ _ _ _ _ _ _ _ _
disguise	_ _ _ _ _ _ s _ _
universe	u n i v e r s _ _
technical	_ _ _ _ _ _ _ a l

Figure 1: Some examples of the response Melbot would give to Melbo's queries on the words villager and universal. Note that if the query word is longer than the secret word, the extra characters are ignored. If the query word is shorter than the secret word, the remaining positions are padded with underscores. A character is revealed only if both the query word and the secret word have that character at the same location.

Problem 2.1 (Playing with the Melbot)**.** Melbo has purchased a new toy to improve his vocabulary called the *Melbot*. The toy lets Melbo play a word-guessing game. There is a dictionary of N words that is known to both Melbo and Melbot. In each round of this game, the following steps are taken:

1. Melbot chooses one of the words from the dictionary as secret say `villager` for this round.
2. Melbot tells Melbo the number of characters in that word by sending Melbo the string `"_ _ _ _ _ _ _ _"` (without the quotes)
3. The following steps are repeated until the round is terminated by either Melbo or Melbot.
   (a) Melbo guesses a word from the dictionary by taking an index i between 0 and $N-1$ and sending it to Melbot as a query. For example, Melbo chooses the index 4972 that corresponds to the word `violation`.
   (b) Melbot check's Melbo's query to see if it is a valid one. If the query index is invalid i.e. $i \notin [0, N-1]$, then Melbot assumes that Melbo no longer wants to continue and terminates the round.
   (c) If the query index is valid, Melbot checks if Melbo's guess is indeed the secret word and if so, the round is terminated and the win count is incremented by 1.
   (d) If the query word is not the secret word and Melbo has made too many queries in this round (the limit is $Q = 15$ queries per round), then Melbot terminates the round.
   (e) If the query word is not the secret word but Melbo has not reached the query limit, then Melbot reveals to Melbo all characters that are common to the query word and the secret word (only if those characters are in the correct location as well). For example, if the secret word is `villager` and the query word is `violation`, then Melbot would return the string `"v i _ l a _ _ _"` (without the quotes). See the figure for more examples.

Melbo plays N rounds of this game, once with each word in the dictionary. Melbo's performance is judged based on the number of words guessed correctly within the query limit (the win count divided by N), the average number of queries asked per round and some other measures described below. Note that at any point, Melbo can ask any word as a query so long as it is in the dictionary. It is not necessary that if Melbo is at a certain node in the decision tree, then only one of the words that reached that node must be asked – words that did not reach that node may also be asked if they help discriminate between words that reached that node.

Before proceeding, **please take a look at the Google Colab validation code and the dummy submission file** `dummy_submit.py` provided with the assignment package to get an idea of how the above protocol works and how your evaluation would be done.

Note that Melbo can also terminate a round by setting the `is_done` flag but that is automatically done when Melbo reaches the leaf of the decision tree. There is no way for you to specify this flag in your solution. When you are writing a code and wish to terminate a round, you should instead send an illegal index to signal termination of the round.

Your Data. We have provided you with a dictionary of $N = 5167$ words. Each word in the dictionary is written using only lower-case Latin characters i.e. `a - z`.

Your Task. You need to develop a decision tree learning algorithm that can play this game. However, note that your algorithm will be tested on a secret dictionary that we have not revealed to you. More on this later. The following enumerates the 2 parts to the question. Part 1 needs to be answered in the PDF file containing your report. Part 2 needs to be answered in the Python file.

1. Give detailed calculations explaining the various design decisions you took to develop your decision tree algorithm. This includes the criterion to choose the splitting criterion at each internal node (which essentially decides the query word that Melbo asks when that node is reached), criterion to decide when to stop expanding the decision tree and make the node a leaf, any pruning strategies and hyperparameters etc. (10 marks)

2. Write code implementing your decision tree learning algorithm. You are not allowed to use any library other than numpy. This means that even use of scikit-learn is prohibited. Use of other libraries such as scipy, skopt, etc is also forbidden. Submit code for your chosen method in `submit.py`. Your code must implement a `my_fit()` method that takes a dictionary as a list of words and returns a trained decision tree as a model. The trained decision tree as a model should be a tree object. Every node in that tree should be a node object. There is no restriction on what attributes the tree object or the node objects may have and what methods those classes implement (i.e. feel free to implement your own Tree and Node classes) but the Node class must implement at least 2 methods:

   (a) Every non-leaf node should implement a `get_child()` method that takes a response and decides which child node to move to.

   (b) Every node (leaf as well as non-leaf) should implement a `get_query()` method that tells what query Melbo should ask when at that node. For leaf nodes, this would be the final query in that round after which the round will be terminated.

   We will evaluate your method on a different dictionary than the one we have given you and check how good is the algorithm you submitted (see below for details). **Please go over the Google Colab validation code and the dummy submission file** `dummy\_submit.py` to clarify any doubts about data formats, protocol etc. (30 marks)

Part 1 needs to be answered in the PDF file containing your report. Part 2 needs to be answered in the Python file.

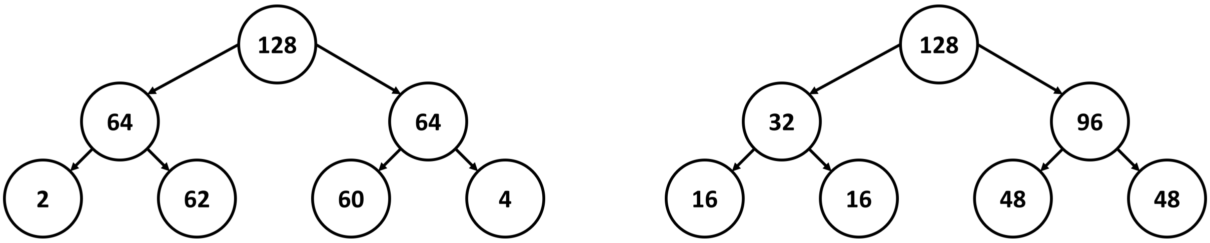


Figure 2: Greedy splitting can be suboptimal. In the example on the left, the split at the root was very nice (entropy 6.0) but it eventually led to a bad set of grandchildren (entropy 5.73). In the example on the left, the split at the root was not that nice (entropy 6.19 ¿ 6.0) but the set of grandchildren it produced had lower entropy (entropy 5.19).

Evaluation Measures and Marking Scheme. We have two dictionaries with us, a public one and a secret one. The public one has been given to you in the assignment package but the secret one is safe with us. The query limit per round will be $Q = 15$ for both dictionaries i.e. Melbo can make upto 15 queries in each round before Melbot terminates the round. Both dictionaries will be composed of words that are written using only lower-case Latin characters i.e. `a - z`. However, whereas the public dictionary is composed of English words, the secret dictionary may be in another language e.g. Spanish, French, Italian, Hindi, Bangla, etc but there will be no accents or special characters – only lower-case Latin characters will be used. Thus, do not assume that character frequencies will be the same e.g. the character `e` may not be the most common in the secret dictionary. However, your `my_fit()` function can do operations to find out which is the most common character etc. The number of words in the secret dictionary will be between 5000 and 6000 and the length of each word will be between 4 and 15 characters.

1. How fast is your `my_fit` method able to finish training (5 marks)
2. What is the on-disk size of your learnt model (after a pickle dump) (5 marks)
3. How many queries does your model make per round on average (15 marks)
4. How high is the win rate offered by your model (on what fraction of the secret dictionary is your model able to guess the word correctly within 15 queries)? (5 marks)

Thus, the total marks for the code evaluation is 30 marks. For more details, please check the evaluation script on Google Colab (linked below). Once we receive your code, we will execute the evaluation script to award marks to your submission.

You would have noticed the relatively less marks for win rate. This is because it is pretty simple to get a high win rate simply by growing the tree larger and larger. The challenge is to get a high win rate will as few queries as possible.

Possible Solution Strategies. You may of course follow standard information-theoretic entropy reduction algorithms such as we have discussed in class but you will need to implement these yourself. However, there is much more you can experiment with:

1. A slightly expensive but solid strategy is to implement *lookahead.* Notice that instead of choosing the split that gives the maximum entropy reduction right now (as we saw in class), a better strategy is too choose a split that will result in maximum entropy reduction two levels or three levels from now. The figure above shows an example where a greedy split may end up giving poorer result. However, implementing large lookahead can quickly

blow up training time. If you at all wish to experiment with lookahead, do so once the number of words in the nodes has reduced to smaller numbers i.e. don't do lookahead at the root – do so a few levels down the root.

2. You may ditch entropy altogether and instead directly aim for minimizing the number of queries it takes to correctly guess the word and win rate. Suppose you have 100 words in a node at depth 5. Find out all subtrees that can be built from this node onward by trying out all possible queries at this node, then all possible queries at this node's children and so on for 10 levels (we have to stop at 10 levels since we cannot make more than 15 queries and the node at which we started was already at level 5). Of all these subtrees, choose the one that has the highest value of average win rate across the 100 words or lowest average number of queries asked for those 100 words before success, or some combination thereof. Again, do not experiment with this technique close to the root. Try this only when the node sizes have become manageably small.

3. You may experiment with splits that are not just based on entropy reduction or expected time to success but that also offer a balanced split. Think of the balance of a split being a regularization term that prevents you from choosing an imbalanced split that simply reduces entropy a lot.

Validation on Google Colab. Before making a submission, you must validate your submission on Google Colab using the script linked below.

Link: `https://colab.research.google.com/drive/1_qlxLBXcauDRwHGIZUcppUEMtV92SR9j?usp=sharing`

Validation ensures that your `submit.py` does work with the automatic judge and does not give errors due to file formats etc. Please use IPYNB file at the above link on Google Colab and the dummy secret dictionary (details below) to validate your submission.

Please make sure you do this validation on Google Colab itself. **Do not download the IPYNB file and execute it on your machine – instead, execute it on Google Colab itself.** This is because most errors we encounter are due to non-standard library versions etc on students personal machines. Thus, running the IPYNB file on your personal machine defeats the whole purpose of validation. You must ensure that your submission runs on Google Colab to detect any library conflict. **Please note that there will be penalties for submissions which were not validated on Google Colab and which subsequently give errors with our automated judge.**

Dummy Submission File and Dummy Secret Dictionary. In order to help you understand how we will evaluate your submission using the evaluation script, we have included a dummy secret dictionary and a dummy submission file in in the assignment package itself (see the directory called `dummy`). However, note that the dummy secret dictionary is just a copy of the public dictionary. However, the dummy submission file may prove useful to you since it contains code to implement a fully functional decision tree for the problem. However, please note the following points

1. You are allowed to use a different Tree and Node class implementation in your submission so long as your Node class implements the `get_child()` and `get_query()` methods as mentioned above.

2. The dummy implementation stores a lot of information at each node e.g. node history etc. Feel free to discard this information if you are not using it to decide your query since it will just increase your model size.

3. The dummy implementation makes very poor choices e.g. expanding the tree to upto 20 levels, paying no heed to balance in the tree, and using random queries at each node. You must make much better choices in your submission to get good marks.

The reason for providing the dummy secret dictionary is to allow you to check whether the evaluation script is working properly on Google Colab or not. Be warned that the secret dictionary on which we actually evaluate your submission will be a different one. We have also included a dummy submission file `dummy_submit.py` to show you how your code must be written to return a model with `my_fit()`. Note that the model used in `dummy_submit.py` is a very bad model which will give poor accuracies. However, this is okay since its purpose is to show you the code format.

Using Internet Resources. You are allowed to refer to textbooks, internet sources, research papers to find out more about this problem and for specific derivations e.g. the XORRO PUF problem. However, if you do use any such resource, cite it in your PDF file. There is no penalty for using external resources so long as they are acknowledged but claiming someone else's work (e.g. a book or a research paper) as one's own work without crediting the original author will attract penalties.

Restrictions on Code Usage. You are allowed to use the numpy module in its entirety. However, you **may not use** any other library including **sklearn, scipy, libsvm, keras, tensorflow is also prohibited**. Use of prohibited modules and libraries for whatever reason will result in penalties. For this assignment, you should also not download any code available online or use code written by persons outside your assignment group (we will relax this restriction for future assignments). Direct copying of code from online sources or amongst assignment groups will be considered an act of plagiarism for this assignment and penalized according to pre-announced policies.

(60 marks)

2 How to Prepare the PDF File

Use the following style file to prepare your report.
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.sty`

For an example file and instructions, please refer to the following files
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.tex`
`https://media.neurips.cc/Conferences/NeurIPS2022/Styles/neurips_2022.pdf`

You must use the following command in the preamble

```
\usepackage[preprint]{neurips_2022}
```

instead of `\usepackage{neurips_2022}` as the example file currently uses. Use proper LaTeX commands to neatly typeset your responses to the various parts of the problem. Use neat math expressions to typeset your derivations. Remember that all parts of the question need to be answered in the PDF file. All plots must be generated electronically - no hand-drawn plots would be accepted. All plots must have axes titles and a legend indicating what the plotted quantities are. Insert the plot into the PDF file using proper LaTeX `\includegraphics` commands.

3 How to Prepare the Python File

The assignment package contains a skeleton file `submit.py` which you should fill in with the code of the method you think works best among the methods you tried. You must use this skeleton file to prepare your Python file submission (i.e. do not start writing code from scratch). This is because we will autograde your submitted code and so your code must have its input output behavior in a fixed format. Be careful not to change the way the skeleton file accepts input and returns output.

1. The skeleton code has comments placed to indicate non-editable regions. **Do not remove those comments**. We know they look ugly but we need them to remain in the code to keep demarcating non-editable regions.

2. The code file you submit should be self contained and should not rely on any other files (e.g. other .py files or else pickled files etc) to work properly. Remember, your ZIP archive should contain only one Python (.py) file. This means that you should store any hyperparameters that you learn inside that one Python file itself using variables/functions.

3. You are allowed to freely define new functions, new variables, new classes in inside your submission Python file while not changing the non-editable code.

4. The use of machine learning libraries such as sklearn, scipy, libsvm, keras, tensorflow is prohibited.

5. Do take care to use broadcasted and array operations as much as possible and not rely on loops to do simple things like take dot products, calculate norms etc otherwise your solution will be slow and you may get less marks.

6. Before submitting your code, make sure you validate on Google Colab to confirm that there are no errors etc.

7. You do not have to submit the evaluation script to us – we already have it with us. We have given you access to the Google Colab evaluation script just to show you how we would be evaluating your code and to also allow you to validate your code.