HOMEWORK 1
DECISION TREES & kNNS[1]

CMU 10-701: INTRODUCTION TO MACHINE LEARNING (FALL 2023)

https://machinelearningcmu.github.io/F23-10701/
OUT: Wednesday, Sep 6th, 2023

DUE: Wednesday, Sep 20th, 2023, 11:59pm

Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., "Jane explained to me what is asked in Question 2.1"). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only.

  See the Academic Integrity Section on our course syllabus for more information:
  `https://machinelearningcmu.github.io/F23-10701/#Syllabus`.

- **Late Submission Policy:** You will have **4 total grace days** for use across all of the homework assignments. We will automatically keep a tally of these grace days for you; they will be applied greedily. You may not use more than **2 grace days** on any single assignment.

  See the General Policies on our course syllabus for more information:
  `https://machinelearningcmu.github.io/F23-10701/#Syllabus`.

- **Submitting your work:**
  - **Gradescope:** There will be two submission slots for this homework on Gradescope: Written and Programming.
    For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional page at the end of the homework and guide us there with a 'See page xx for the rest of the solution'.
    You are also required to upload your code, which you wrote to solve the final question of this homework, to the Programming submission slot.
    Regrade requests can be made after the homework grades are released, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For LaTeXusers, use ■ and ●for shaded boxes and circles, and don't change anything else. If an answer box is included for showing work, **you must show your work**!

---

[1] Compiled on Wednesday 6th September, 2023 at 21:33

1 Decision Trees [20 pts]

1. Consider the dataset in Table 1 for this problem. Given the four attributes on the left, we want to predict if the student got an A in the course. The following questions involve some computation, so you may want to solve them programmatically. You can find this dataset in *data.txt*

Wakes Up Early	**Finished All Homework**	**Completed Pre-requisites**	**Likes Coffee**	**A**
1	1	0	0	1
1	1	1	0	1
0	0	1	0	0
0	1	1	0	0
0	1	1	0	1
0	0	1	1	1
1	0	0	0	0
0	1	0	1	1
0	0	1	0	1
1	0	0	0	0
1	1	1	0	1
0	1	1	1	0
0	0	0	0	0
1	0	0	1	1

Table 1: decision tree features and labels

Create a decision tree of depth 2 using the ID3 algorithm from class. The tree should have the same structure as the diagram in Figure 1. Note that 0 leads to the left branch and 1 leads to the right branch.

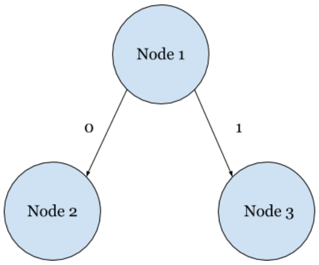

Figure 1: Decision tree structure for question 1

(i) **[2 pts]** What is the attribute at Node 1? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Attribute to split on: ☐ Mutual information: ☐

(ii) **[2 pts]** What is the attribute at Node 2? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Attribute to split on: ☐ Mutual information: ☐

(iii) **[2 pts]** What is the attribute at Node 3? What is the information gain of this attribute? Please round your answer to 4 decimal places.

Attribute to split on: ☐ Mutual information: ☐

2. **[4 pts]** Consider learning a decision tree for some binary classification task. Assume that we do not restrict the size of the tree and that the tree can branch multiple times on the same attribute. Under what condition(s) would we be unable to construct a tree that perfectly classifies the dataset?

3. **[4 pts]** Given enough time, is the ID3 algorithm guaranteed to output the smallest decision tree that is *consistent* with the training dataset i.e., achieves zero training error rate? Briefly explain why or why not.

4. **[6 Points]** Recall that the information gain of some variable A is the reduction in entropy of a target variable Y in some dataset S due to partitioning on A. Prove that information gain is always non-negative. To receive full credit, you must show all steps of your work. **Hint:** You may find the result proved in Recitation 1 regarding the *KL divergence* between two distributions useful for this question.

2 Nearest Neighbors and Decision Trees [6 Points]

Consider a multiclass classification problem with 2 real-valued features. Suppose you are given n data points $P_1, P_2, \ldots, P_n$ and the corresponding label for each data point $C_1, C_2, \ldots, C_n$ (where $C_1, C_2, \ldots, C_n$ take values from the set of all possible class labels). Under the k nearest neighbors classification scheme, each new element Q is simply categorized by a majority vote among its k nearest neighbors. The 1-NN is a simple variant of this which divides up the input space for classification purposes into convex regions (see Figure 2), each corresponding to a point in the dataset.

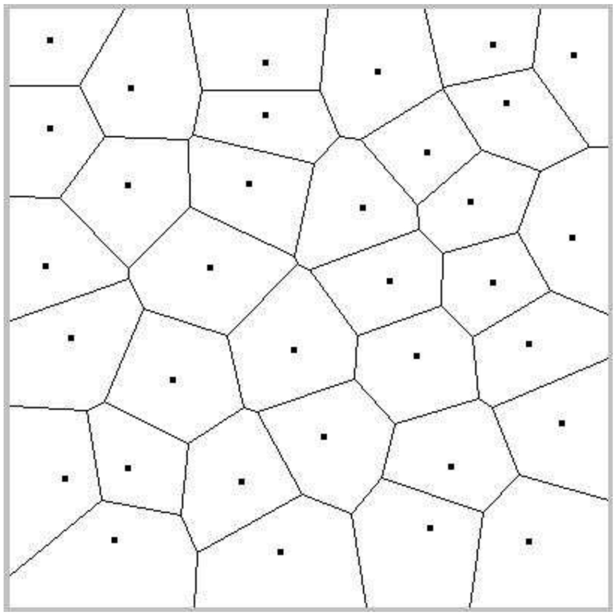

Figure 2: 1-NN decision boundaries using the Euclidean distance metric.

1. **[6 Points]** Is it possible to build a decision tree (with decisions at each node of the form "is $x_1 > a$", "is $x_1 < b$", "is $x_2 > c$", or "is $x_2 < d$" for any real constants a, b, c, d) which behaves identically to a 1-NN classifier using the Euclidean distance metric? If so, how and if not, why not.

3 k-Nearest Neighbors [19 Points]

1. [**6 pts**] In a kNN classification problem, assume that the distance measure is not explicitly specified to you. Instead, you are given a "black box" where you input a set of data points $P_1, P_2, \ldots P_n$ and an unlabelled data point Q, and the black box outputs the nearest neighbor of Q, say P_i and its corresponding label C_i. Is it possible to construct a kNN classification algorithm (w.r.t the unknown distance metrics) based on this black box alone? If so, how and if not, why not?

2. [**4 pts**] Now suppose the black box returns the j nearest neighbors (and their corresponding labels) instead of the single nearest neighbor (assume $j \neq k$). Is it still possible to construct a kNN classification algorithm based on the black box? If so, how and if not, why not?

   Note: The black box operates with a *set* of data points, which assumes they are distinct.

3. [**3 pts**] Assume we have a dataset with binary labels and we know that the leave-one-out cross validation accuracy (LOOCV) of a 1-NN classifier on this dataset is 1. Now suppose that we introduce some noise to the dataset: each label has a 10% chance of being flipped to the other class. What is the *expected* LOOCV accuracy of a 1-NN classifier in this setting?

4. **[3 pts]** Suppose that in the dataset from the previous question, exactly 10% of the labels got flipped. What is the *maximum* possible LOOCV accuracy? Justify your answer using a formal proof or by constructing an example.

5. **[3 pts]** How should the noise level (i.e., the probability of a label being flipped) affect your choice of k for the k-NN algorithm, if at all? You do not need to provide a formal proof but you should give a brief justification.

4 Programming (55 points)

In this assignment, you will implement a decision tree learning algorithm for binary classification. In addition, we will ask you to run some end-to-end experiments on two tasks (predicting whether or not a patient has heart disease / predicting the final grade for high school students) and report your results. Please confine all code you write to a single, self-contained file titled `decision_tree.py`.

4.1 The Tasks and Datasets

Materials Download the zip file from the course website. The zip file will have a handout folder that contains all the data that you will need in order to complete this assignment.

Datasets The handout contains three datasets. Each one contains attributes and labels and is already split into training and validation data. The first line of each `.tsv` file contains the name of each attribute, and *the class label is always the last column.*

1. **heart:** The first task is to predict whether a patient has been (or will be) diagnosed with heart disease, based on available patient information. The attributes (aka. features) are:
   (a) `sex`: The sex of the patient—1 if the patient is male, and 0 if the patient is female.
   (b) `chest_pain`: 1 if the patient has chest pain, and 0 otherwise.
   (c) `high_blood_sugar`: 1 if the patient has high blood sugar (>120 mg/dl fasting), and 0 otherwise.
   (d) `abnormal_ecg`: 1 if the patient had an abnormal resting electrocardiographic (ECG) reading, and 0 otherwise.
   (e) `angina`: 1 if exercise induced angina in the patient, and 0 otherwise. Angina is a type of severe chest pain.
   (f) `flat_ST`: 1 if the patient's ST segment (a section of an ECG) was flat during exercise, and 0 if it had some slope.
   (g) `fluoroscopy`: 1 if a physician used fluoroscopy, and 0 otherwise. Fluoroscopy is an imaging technique used to see the flow of blood through the heart.
   (h) `thalassemia`: 1 if the patient is known to have thalassemia, and 0 otherwise. Thalassemia is a blood disorder that may impair the oxygen-carrying capacity of the patient's red blood cells.
   (i) `heart_disease`: 1 if the patient was diagnosed with heart disease, and 0 otherwise. This is the class label you should predict.

   The training data is in `heart_train.tsv`, and the validation data in `heart_valid.tsv`.
2. **education:** The second task is to predict the final grade for high school students. The attributes are student grades on 5 multiple choice assignments *M1* through *M5*, 4 programming assignments *P1* through *P4*, and the final exam *F*. Values of 1 indicate that a student received an A, and 0 indicates that the student did not receive an A. The training data is in `education_train.tsv`, and the validation data in `education_valid.tsv`.
3. **small:** We also include `small_train.tsv` and `small_valid.tsv`—a small, purely for demonstration version of the **heart** dataset, with *only* attributes `chest_pain` and `thalassemia`.

4.2 Program: Decision Tree

Your code should learn a decision tree with a specified maximum depth, print the decision tree in a specified format, predict the labels of the training and validation examples, and calculate training and validation errors.

Your implementation must satisfy the following requirements:

- Use mutual information to determine which attribute to split on.
- Be sure you're correctly weighting your calculation of mutual information. For a split on attribute X, $I(Y;X) = H(Y) - H(Y|X) = H(Y) - P(X=0)H(Y|X=0) - P(X=1)H(Y|X=1)$.
- As a stopping rule, only split on an attribute if the mutual information is > certain threshold. By default, you should set the threshold to 0. Certain questions in 4.3 may require you to experiment with different threshold.
- Do not grow the tree beyond a certain max-depth, if specified in the question. For example, for a maximum depth of 3, split a node only if the mutual information is > 0 and the current level of the node is < 3.
- Use a majority vote of the labels at each leaf to make classification decisions. If the vote is tied, choose the label that is numerically larger (i.e., 1 should be chosen before 0)
- It is possible for different columns to have equal values for mutual information. In this case, you should split on the ***first column to break ties*** (e.g. if column 0 and column 4 have the same mutual information, use column 0).

4.2.1 Getting Started

Careful planning will help you to correctly and concisely implement your decision tree learning algorithm. Here are a few *hints* to get you started:

- Write helper functions to calculate entropy and mutual information.
- It is best to think of a decision tree as a collection of nodes, where nodes are either leaf nodes (where final decisions are made) or interior nodes (where we split on attributes). It is helpful to design a function to train a single node (i.e. a depth-0 tree), and then recursively call that function to create sub-trees.
- In the recursion, keep track of the depth of the current tree so you can stop growing the tree beyond the max-depth.
- Implement a function that takes a learned decision tree and data as inputs, and generates predicted labels. You can write a separate function to calculate the error of the predicted labels with respect to the given (ground-truth) labels.
- Be sure to correctly handle the case where the specified maximum depth is greater than the total number of attributes.
- Be sure to handle the case where max-depth is zero (i.e. a majority vote classifier).

4.2.2 Output: Printing the Tree

You should also write a function to pretty-print your learned decision tree. **Your function should print your tree only *after* you are done generating the fully-trained tree.** Each row should correspond to a node in the tree. They should be printed using a *pre-order depth-first-search* traversal (but you may print left-to-right or right-to-left, i.e. your answer does not need to have exactly the same order as the reference below). Print the attribute of the node's parent and the attribute value corresponding to the node. Also include the sufficient statistics (i.e. count of positive / negative examples) for the data passed to that node. The row for the root should include *only* those sufficient statistics. A node at depth d, should be prefixed by d copies of the string '| '.

Below, we have provided the recommended format for printing the tree. You can print it directly rather than to a file.

```
$ python decision_tree.py small_train.tsv small_valid.tsv 2

[14 0/14 1]
| chest_pain = 0: [4 0/12 1]
| | thalassemia = 0: [3 0/4 1]
| | thalassemia = 1: [1 0/8 1]
| chest_pain = 1: [10 0/2 1]
| | thalassemia = 0: [7 0/0 1]
| | thalassemia = 1: [3 0/2 1]
```

However, you should be careful that the tree might not be full. For example, with a different subset of the small dataset, there may be no nodes under `chest_pain = 0` if all labels are the same.

The following pretty-print shows the education dataset with max-depth 3. Use this example to check your code before submitting your pretty-print of the heart dataset.

```
$ python decision_tree.py education_train.tsv education_valid.tsv 3

[65 0/135 1]
| F = 0: [42 0/16 1]
| | M2 = 0: [27 0/3 1]
| | | M4 = 0: [22 0/0 1]
| | | M4 = 1: [5 0/3 1]
| | M2 = 1: [15 0/13 1]
| | | M4 = 0: [14 0/7 1]
| | | M4 = 1: [1 0/6 1]
| F = 1: [23 0/119 1]
| | M4 = 0: [21 0/63 1]
| | | M2 = 0: [18 0/26 1]
| | | M2 = 1: [3 0/37 1]
| | M4 = 1: [2 0/56 1]
| | | P1 = 0: [2 0/15 1]
| | | P1 = 1: [0 0/41 1]
```

The numbers in brackets give the number of positive and negative labels from the training data in that part of the tree.

At this point, you should be able to answer questions 1-5 in the "Empirical Questions" of this handout. Write your solutions in the template provided.

4.3 Written: Empirical Questions

1. [**5 Points**] Report the validation accuracy of the decision tree classifiers trained on the three included datasets (with no limitation on maximum depth and all configurations like information gain threshold etc. set to the default value):
   - `small` dataset:
   - `heart` dataset:
   - `education` dataset:
2. [**5 Points**] In the box below, paste the decision tree learned on `heart_train.tsv` under max depth limitation of 4. Be sure to follow the formatting requirement in section 4.2.2.
3. [**10 Points**] Now use `heart_val.tsv` to perform reduced error pruning on the tree you learned in the previous question; break ties in favor of shorter trees. In the box below, paste the decision tree learned after pruning. Be sure to follow the formatting requirement in section 4.2.2.

In the following questions, we will explore the performance of learned decision trees on validation data and introduce the problem of "overfitting". **All questions below are asked on the `heart` dataset (`heart_train.tsv` and `heart_val.tsv`)**.

3. **[15 Points]** Iterate the maximum depth limitation in the range $[0, 8]$, collect the training accuracy and validation accuracy. Plot both metrics with respect to the maximum depth in a single figure. Label your axes as well as your training and validation accuracy plots.

   Is the validation accuracy monotonically increasing as the maximum depth limit increases? What about the training accuracy? Report the maximum validation accuracy observed and the maximum depth limit that at which it is achieved.

The phenomenon you have (hopefully) witnessed, wherein the model struggles to extrapolate effectively from the training dataset to the validation dataset, is widely referred to as *overfitting*. One approach to mitigating the overfitting dilemma involves constraining the complexity of the classifier.

Constraining the maximum depth of the decision tree is perhaps the most straightforward method for addressing the issue of overfitting but there are many alternatives.

4. **[10 Points]** Among these alternatives is the imposition of a restriction on the size of splitting nodes. Specifically, if a node comprises *fewer than* M data points during the training process, further splitting is prohibitted.

   Train a decision tree under different minimum splitting sizes, with C ranging from 0 to the number of data points in the `heart` dataset: what is the optimal minimum splitting size, and what is the validation accuracy under this value?

   Note: If there are multiple minimum splitting sizes that achieve the optimal validation accuracy, report the smallest of them.

5. **[10 Points]** Another technique for mitigating overfitting involves changing the mutual information threshold. In this approach, at each node, if the best possible mutual information from a subsequent split falls below a designated threshold τ, the node is not divided any further.

   Train a decision tree with mutual information thresholds of $\{0.00, 0.01, \cdots, 0.99, 1.00\}$: what is the optimal information gain threshold, and what is the validation accuracy under this value?

   Note: If there are multiple mutual information thresholds that achieve the optimal validation accuracy, report the smallest of them.

4.4 Submission Instructions

Programming Please ensure you have completed the following file(s) for submission.

`decision_tree.py`

When submitting your solution, make sure to select and upload the file(s) shown above. **Any other files will be deleted.** Ensure the files have the exact same spelling and letter casing as above. We may manually grade your code for the purposes of assigning partial credit if your solutions to the empirical questions differ from our reference solutions.

Written Questions Make sure you have completed all questions from Written component (including the collaboration policy questions) in the template provided. When you have done so, please submit your document in **PDF format** to the corresponding assignment slot on Gradescope.

5 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?

   (b) If you answered ‘yes’, give full details (e.g. “Jane Doe explained to me what is asked in Question 3.4”)

2. (a) Did you give any help whatsoever to anyone in solving this assignment?

   (b) If you answered ‘yes’, give full details (e.g. “I pointed Joe Smith to section 2.3 since he didn’t know how to proceed with Question 2”)

3. (a) Did you find or come across code that implements any part of this assignment?

   (b) If you answered ‘yes’, give full details (book & page, URL & location within the page, etc.).