Homework 1

Graded

Student

Jinzhi Shen

Total Points

31 / 55 pts

Question 1

P1 **26** / 50 pts

1.1 **a** **5** / 5 pts

✔ **– 0 pts** Correct

1.2 **b** **7** / 10 pts

A simpler way is to directly prove that the entropy of each child node should not be greater than or equal to 1

✔ **– 2 pts** Insufficient proof of the principle of maximum entropy.

✔ **– 1 pt** Minor mistake - see comment.

You need to explicitly state to which function you are applying Jensen's inequality. And prove it's convex.

1 E over what variables?

1.3 **c** **2** / 5 pts

✔ **– 3 pts** Wrong formula of information gain

1.4 **d** **0** / 5 pts

✔ **– 2 pts** Wrong answer

✔ **– 3 pts** Missing/Wrong analysis

1.5 **e** **0** / 5 pts

✔ **– 3 pts** Wrong answer

✔ **– 2 pts** Wrong reason/split

1.6 **f** **10** / 10 pts

✔ **– 0 pts** Correct

Points are given. But next time, please give a specific example for each decision tree.

1.7 **g** **2** / 10 pts

✔ **– 2 pts** Incorrect (or missing) calculation for the tree with zero split

✔ **– 2 pts** Incorrect calculation for the tree with one split

✔ **– 2 pts** Incorrect calculation for the tree with two split

✔ **– 4 pts** Incorrect answer for the MAP tree

+ 2 pts You are on the right track but your calculations are incorrect.

Question 2

P2 **5** / 5 pts

2.1 **d** **5** / 5 pts

✔ **– 0 pts** Correct

No questions assigned to the following page.

CS 446 / ECE 449 — Homework 1

jinzhis2

Version 2.0

Instructions.

- Homework is due **Wednesday, September 14, at 11:59 AM CST**; you have **3** late days in total for **all Homeworks**.
- The template for coding problems are available at this link.
- Everyone must submit individually at gradescope under `Homework 1` and `Homework 1 Code`.
- The "written" submission at `hw1` **must be typed**, and submitted in any format gradescope accepts (to be safe, submit a PDF). You may use LaTeX, markdown, google docs, MS word, whatever you like; but it must be typed!
- When submitting at `Homework 1`, gradescope will ask you to **mark out boxes around each of your answers**; please do this precisely!
- Please make sure your NetID is clear and large on the first page of the homework.
- Your solution **must** be written in your own words. Please see the course webpage for full **academic integrity** information. You should cite any external reference you use.
- We reserve the right to reduce the auto-graded score for `hw1code` if we detect funny business (e.g., your solution lacks any algorithm and hard-codes answers you obtained from someone else, or simply via trial-and-error with the autograder).
- When submitting to `Homework 1 Code`, only upload `hw1.py` and `hw1_utils.py`. Additional files will be ignored.

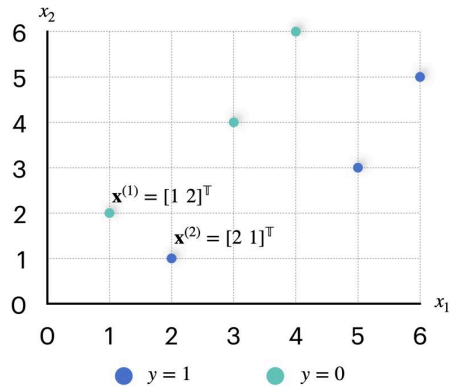
Version History.

1. Initial version.
2. Add MLE/MAP and Bayes parts.

No questions assigned to the following page.

1. Decision Tree, MLE and MAP.

The figure below shows a dataset $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{6}$ (where $\boldsymbol{x}^{(i)} \in \mathbb{R}^2$, $y^{(i)} \in \mathbb{R}$), containing six data points with two features x_1 and x_2. For example, $\boldsymbol{x}^{(1)} = [1\ 2]^\intercal$ and $\boldsymbol{x}^{(2)} = [2\ 1]^\intercal$. The label $y^{(i)}$ can take on the values 1 or 0.



The decision tree defined in class can only support discrete-valued instances. Here, we extend this concept to general continuous spaces. A continuous-valued decision attribute need to be represented using a comparison operator $(\geq, <)$. Specifically, for each round, unlike discrete-valued tree building that chooses only which feature to use as the current decision attribute, we will specify a feature (x_1 or x_2) and also **a threshold** ϵ, and then create **two descendant nodes** at the current node. For all data points in the current node, those below the threshold will be put into child node "$x_j < \epsilon$", and those above the threshold will be put into child node "$x_j \geq \epsilon$"($j = 0$ or 1).

Note: We assume ϵ can only be integer values. Please describe the split rule as "$x_j \geq \epsilon$", such as $x_1 \geq 1$ or $x_2 \geq 2$. Do not use the answers like $x_1 \geq 2.5$ or $x_2 > 2$. And also make sure to describe what the predicted label is in two child nodes "$x_j < \epsilon$" and "$x_j \geq \epsilon$" ($j = 0$ or 1).

(a) What is the entropy at the root?

(b) What is the minimum information gain if we split the root into two child nodes? Please prove the lower bound of information gain, then find a split that reaches the bound.

Hint: Consider the mathematical definition of information gain and Jensen's Inequality.

Jensen's inequality for two point a_1 and a_2 in a convex function f: For any $t \in [0, 1]$, $f(ta_1+(1-t)a_2) \leq tf(a_1) + (1-t)f(a_2)$.

(c) What is the maximum information gain if we split the root into two child nodes? what is the rule for this split? You only need to give this maximum information gain, no proof is required.

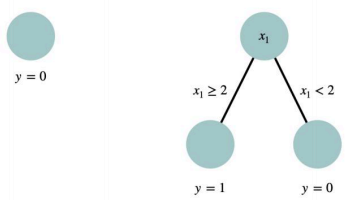
(d) After the first split in (c), how do we further split child nodes based on maximum information gain?

(e) Let's return to the beginning of the tree building. Suppose we define $\boldsymbol{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(6)}\right)^\top$. Is there a matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$, with which we can preprocess the dataset to $\boldsymbol{X}' = \boldsymbol{X}\mathbf{A}$, so that the decision tree building algorithm on $\boldsymbol{X}'$ and same labels $\{y^{(i)}\}_{i=1}^{6}$ ends up with only one split at the root? Explain your answer.

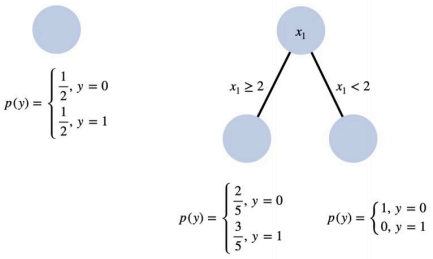
Remark. Regular univariate decision trees can only split data parallel to the coordinate axis, generating axis-aligned splits. A multivariate decision tree is a generalization of univariate decision trees, where more than one attribute can be used in the decision rule for each split. In other words, it selects not the best attribute but the best combination of the attribute, generating non-axis-aligned splits.

Hint. In other word, consider if we can transform(rotate) $\boldsymbol{X}$ so that it can be classified by one axis-aligned split.

Question assigned to the following page: 1.1



Continuous-valued decision trees



Continuous-valued probabilistic decision trees

Now consider a new **probabilistic decision tree** where each leaf node has a probabilistic distribution $P(y)$ instead of the label $y = 0$ or 1. The probability is the fraction of data points at that leaf node with that label. See the figure above, a probabilistic decision tree from the dataset $\mathcal{D}$ with only root node would say $P(y = 1) = P(y = 0) = \frac{3}{6} = \frac{1}{2}$. A probabilistic decision tree with one split ($x_1 \geq 2$) would say $P(y = 1) = 0$ and $P(y = 0) = 1$ for the leaf node "$x_1 < 2$", $P(y = 1) = \frac{3}{5}$ and $P(y = 0) = \frac{2}{5}$ for the leaf node "$x_1 \geq 2$".

(f) What is a decision tree that maximizes the likelihood of the data $\mathcal{D}$? What is this maximum likelihood? What is a decision tree that minimizes the likelihood of the data $\mathcal{D}$? What is this minimum likelihood?
Hint: Note such decision trees may not be unique, just answer one. Remember the likelihood is the probability of observed data $\mathcal{D}$. And now the probability is given by the leaf node in the probabilistic decision tree.

(g) Consider a prior probability distribution $P(T)$ over trees that penalizes the number of nodes in the tree.

$$P(T) = \left(\frac{1}{2}\right)^{2^{(S(T)+1)/2}},$$

where T is a decision tree, $S(T)$ is the **number of nodes** in T. Please give the MAP decision tree with the prior $P(T)$ and the same dataset $\mathcal{D}$.
Hint. Consider the trees with only root node ($S(T) = 1$), one split at the root ($S(T) = 3$), and splits at the root and the child node(s) ($S(T) > 3$). The tree building is still based on maximum information gain principle. Please calculate and compare the posterior probabilities of all decision trees during tree building.
Remark. The MAP decision tree is equivalent to a decision tree with **pruning**.

Solution.

Your solution here.
a)At the beginning, nothing really separate the data which according to the original data we have ,we could easily get the points are in half and half which gives P(0) and P(1) =

$$\frac{1}{2}$$

which give entropy =

$$P(1) * \log_2 P(1) + P(0) * \log_2 P(0) = \frac{1}{2} * \log_2 P(1/2) + \frac{1}{2} * \log_2 P(1/2) = 1$$

b)

Questions assigned to the following page: 1.2 and 1.3

The minimum of information gain would be theoretically zero. And just to achieve this, we just split the info into one empty and another one with all the original data, that way entropy remain the same and so for entropy which gives information gain zero.
Prove of lower bounds of information gain: By the definition of information gain, we could get

$$Gain(S, X) = I_S(X, Y) = H_S(Y)H_S(Y|X)$$

— this is what we get from the lecture note Let's rewrite this formula into:
I(X; Y) = H(X) + H(Y) - H(X; Y)

$$= E[\log \frac{p(X, Y)}{p(X)p(Y)}]$$

Now let's apply Jensen's inequality to a convex function

$$log(x) and \frac{q(x)}{p(x)}$$

We will have

$$E[\log \frac{q(X)}{p(X)}]$$

$$\geq -\log E[\frac{q(X)}{p(X)}]$$

$$= -\log \sigma_x \frac{q(X)p(x)}{p(X)}$$

= 0 – this is actually according to another way of writing Jensen's law considering more with the value of it

$$f(E[X]) \leq E[f(X)]$$

, the result equal to zero mostly because addition of all possibility give 1 and log 1 = 0. Now we could say:

$$H(X|Y) \leq H(X)$$

and combine this info, we could find original function follow the log formula we have above (just flip fraction inside we could make it a negative log). And thus we know that

$$= E[\log \frac{p(X, Y)}{p(X)p(Y)}]$$

must be bigger or equal to zero and thus our information gain
c)
the maximum is something that really would vary by cases, as some data are just random so bad by themselves that if we want to just split into two, they have a best case. After calculating all the cases out, I found out that the base cases is when split on

$$x_1$$

$$\epsilon = 5$$

which will give you 5 correct value in total and 1 wrong with

$$x_1 < 5$$

have [3+, 1-] and the other [2+,0] with H(child) =

$$\frac{1}{2} * [-\frac{3}{4} * log_2 \frac{3}{4} - \frac{1}{4} log_2 \frac{1}{4}] + 0$$

Questions assigned to the following page: 1.3, 1.4, 1.5, 1.6, and 1.7

(another side is completely correct, so zero)

$$\approx 0.4056$$

And this is minimum H for child helping us achieving maximum info gain : 1 - 0.4056 = 0.5944

d)
After the first split, the next step is very easy, we only need to find a number in

$$x_1$$

or

$$x_2$$

where is separate that child have one data wrong into two parts (probably just the 3 right as one case and wrong one as one case). As wither way would achieve this goal, I feel both of them are possible way.

e)
This multiply on the old data is just just stretching, rotating, and moving the axis according to the original graph. We could see that there is a clear line that could separate the 0 and 1 data perfectly. This will give it the chance be linear separable. And now we just need a A that rotate one of the axis into position parallel to that straight line and that will make the value at that line one split to the data.

f)
Same as question before, the one minimize the data is just left it there with no split which means we choose

$$\epsilon$$

greater than 6 likelihood:

$$(\frac{1}{2})^6$$

unlike before, the one maximize the data is when there is that best split, where everything will have probability of 1 which with likelihood: 1

g)
We could always achieve the split of the complete data in 2 spites. However, with penalty, we need to reconsider which is better: 1. with only one split : likelihood =

$$\frac{1}{4} * (\frac{1}{4})^3 * (\frac{1}{2})^{2\frac{7}{2}}$$

2.with two split: we have three cases: 1) split a 1,5 again: we get likelihood = 1*

$$(\frac{1}{2})^{2^{2/2}} + (\frac{1}{2})^{2^{6/2}} \approx 0.000977$$

2) split a 2,4 again we get likelihood = 1*

$$(\frac{1}{2})^{2^{3/2}} + (\frac{1}{2})^{2^{5/2}} \approx 0.00279$$

3)split a 3,3 again we get likelihood = 1*

$$(\frac{1}{2})^{2^{4/2}} + (\frac{1}{2})^{2^{4/2}} \approx 0.003906$$

Question assigned to the following page: 1.7

where that split by 3,3 will be the best case and in the data it is when we split by

$$\epsilon = 4$$

on

$$x_1$$

for first and then split with

$$x_2$$

on the later two child tree (one with

$$\epsilon = 2$$

another child

$$\epsilon = 6$$

)

No questions assigned to the following page.

2. Programming - Naive Bayes.

Recall the Naive Bayes classifier mentioned in lecture 4. We can derive the classification rule as follows:

$$
\begin{aligned}
\hat{Y} &= \arg\max_y P(Y=y|X) && \text{(Definition)}\\
&= \arg\max_y \frac{P(X|Y=y)P(Y=y)}{P(X)} && \text{(Bayes rule)}\\
&= \arg\max_y P(X|Y=y)P(Y=y) && \text{(Denominator does not depend on } y\text{)}\\
&= \arg\max_y P(X_1,\ldots,X_d|Y=y)P(Y=y)\\
&= \arg\max_y \left(\prod_{j=1}^{d} P(X_j|Y=y)\right)P(Y=y) && \text{(Conditional independence)}\\
&= \arg\max_y \left(\left(\sum_{j=1}^{d} \log P(X_j|Y=y)\right) + \log P(Y=y)\right) && \text{(Logarithmic operation).}
\end{aligned}
$$

In this problem, you will implement a simple Naive Bayes classifier for objects from 2 classes. For simplicity, we assume that there exists at least one object that belong to each class.

- There are d properties to describe each object.
- We use pair $(\boldsymbol{x}^{(i)}, y^{(i)})$ to represent object i, where $\boldsymbol{x}^{(i)}$ is a length-d vector that describes its properties, $y^{(i)}$ is a scalar representing its label, and $i \in \{1, 2, \cdots, N\}$.
- For $j \in \{1, 2, \cdots, d\}$, $x_j^{(i)} \in \{0, 1\}$. $x_j^{(i)} = 1$ means object i has property j, and $x_j^{(i)} = 0$ otherwise; $y^{(i)} = 0$ means object i is in class 0, and $y^{(i)} = 1$ means object i is in class 1.
- We use $(\boldsymbol{X}, \boldsymbol{y})$ to represent a dataset of size N, where $\boldsymbol{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^\top$ is a $N \times d$ matrix, and $\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, \cdots, y^{(N)}\right)$ is a length-N vector.

You are given two datasets in this homework: $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ and $(\boldsymbol{X}_{\text{test}}, \boldsymbol{y}_{\text{test}})$, which can be obtained by calling `bayes_dataset("train")` and `bayes_dataset("test")` in `hw1_utils.py`. Your tasks are:

(a) Implement the function `bayes_MAP(X, y)` in `hw1.py`.
The input is the training dataset $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$. The output $\boldsymbol{\Theta}$ is a $2 \times d$ matrix in PyTorch tensor, that $\theta_{y,j}$ is the MAP estimation of $P(X_j = 1 \mid Y = y)$.

(b) Implement the function `bayes_MLE(y)` in `hw1.py`.
The input is the label part $\boldsymbol{y}_{\text{train}}$ of the training dataset. The output p is a scalar, which is the MLE for $P(Y = 0)$.

(c) Implement the function `bayes_classify(theta, p, X)` in `hw1.py`.
The input is the output $\boldsymbol{\Theta}, p$ from the two functions above, and the features $\boldsymbol{X}_{\text{test}}$ of the testing dataset of size N. The output $\hat{\boldsymbol{y}}$ is an length-N vector, where $\hat{\boldsymbol{y}}^{(i)}$ is the predicted label (0 or 1) for the object with properties $\boldsymbol{x}_{\text{test}}^{(i)}$, or the i-th row of $\boldsymbol{X}_{\text{test}}$. For simplicity, it's guaranteed that there will be no ties (so the arg max will be unique).
Hint. Please use the logarithmic form of $\hat{Y}$ in your computation to avoid precision issues.

(d) In your **written submission**, for each class label $y \in \{0, 1\}$, list the top 5 properties that the model regards as the most likely to occur in an object of class y in the given training dataset $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$.

Library routines: `torch.log`, `torch.sum`. **Solution.**

Question assigned to the following page: 2.1

Your solution here.
d) y=0: feature14, feature2, feature11, feature19, feature8 y=1: feature1, feature7, feature15, feature11, feature10 (index is from 0 to n)

No questions assigned to the following page.

3. Programming - Gaussian Naive Bayes.

Recall the Gaussian Naive Bayes classifier mentioned in lecture 5. From the derivations of Problem 2, we have:

$$\hat{Y} = \arg\max_y \left(\left(\sum_{j=1}^{d} \log P(X_j | Y = y) \right) + \log P(Y = y) \right)$$

In a Gaussian Naive Bayes, the features X_j are continuous variables, and the probability $P(X_j|Y = y)$ is modeled as a Gaussian distribution

$$P(X_j|Y = y) \sim d(X_j \mid \mu_j, \sigma_j^2)$$

In this problem, similar to Problem 2, you will implement a simple Gaussian Naive Bayes classifier for objects from 2 classes. For simplicity, we assume that there exists at least one object that belong to each class.

- There are d metrics to describe each object.
- We use pair $(\boldsymbol{x}^{(i)}, y^{(i)})$ to represent object i, where $\boldsymbol{x}^{(i)}$ is a length-d vector that describes its properties, and $y^{(i)}$ is a scalar representing its label.
- For $j \in \{1, 2, \cdots, d\}$, $x_j^{(i)} \in \mathbb{R}$ is the value of metric j of object i; $y^{(i)} = 0$ means object i is in class 0, and $y^{(i)} = 1$ means object i is in class 1.
- We use $(\boldsymbol{X}, \boldsymbol{y})$ to represent a dataset of size N, where $\boldsymbol{X} = \left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \cdots, \boldsymbol{x}^{(N)}\right)^\top$ is a $N \times d$ matrix, and $\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, \cdots, y^{(N)}\right)$ is a length-N vector.

You are given two datasets in this homework: $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ and $(\boldsymbol{X}_{\text{test}}, \boldsymbol{y}_{\text{test}})$, which can be obtained by calling `gaussian_dataset("train")` and `gaussian_dataset("test")` in `hw1_utils.py`. Your tasks are:

(a) Implement the function `gaussian_MAP(X, y)` in `hw1.py`.
The input is the training dataset $(\boldsymbol{X}_{\text{train}}, \boldsymbol{y}_{\text{train}})$. The output is $(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$. Both of them are $2 \times d$ matrices in PyTorch float tensor, where $\mu_{y,j}$ and $\sigma_{y,j}^2$ are Gaussian distribution parameters of the MAP estimation of $P(X_j = 1 \mid Y = y)$.

(b) Implement the function `gaussian_MLE(y)` in `hw1.py`.
The input is the label part $\boldsymbol{y}_{\text{train}}$ of the training dataset. The output p is a scalar, which is the MLE for $P(Y = 0)$.

(c) Implement the function `gaussian_classify(mu, sigma2, p, X)` in `hw1.py`.
The input is the output $\boldsymbol{\mu}, \boldsymbol{\sigma^2}, p$ from the two functions above, and the label part $\boldsymbol{X}_{\text{test}}$ of the testing dataset of size N. The output $\hat{\boldsymbol{y}}$ is an length-N vector, where $\hat{\boldsymbol{y}}^{(i)}$ is the predicted label (0 or 1) for the object with properties $\boldsymbol{x}_{\text{test}}^{(i)}$, or the i-th row of $\boldsymbol{X}_{\text{test}}$. For simplicity, it's guaranteed that there will be no ties (so the arg max will be unique).
Hint. Please use the logarithmic form of both $\hat{Y}$ and Gaussian distribution's PDF in your computation to avoid precision issues.

Library routines: `torch.log`, `torch.sum`, `torch.mean`, `torch.var` (Please use `unbiased=False`).

Solution.

Your solution here.