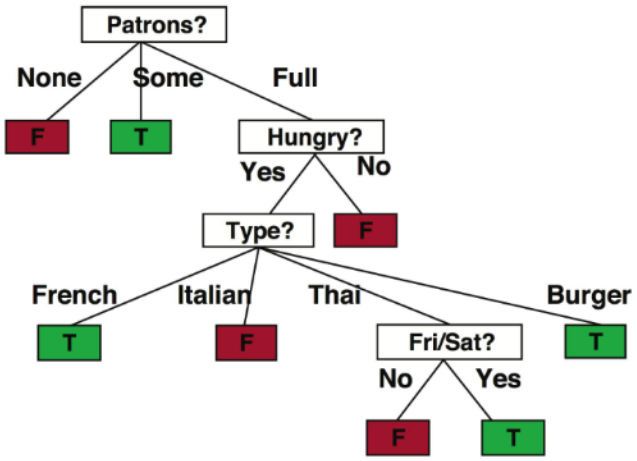
CS 188
Spring 2021

Regular Discussion 10 Solutions

Decision Trees

Decision trees partition an input space and assign a label to each partition. Given a set of training data of a list of attributes and labels, our goal is to find the smallest decision tree that fits the training data. An example of a decision tree is shown below:



- **Entropy:** The expected code length of information in bits. The average level of "surprise" in a variable's possible outcomes.

$$H(\langle p_1, \cdots, p_n \rangle) = \sum_{i=1}^{n} -p_i \log_2(p_i)$$

- **Information Gain:** The amount by which we reduce entropy after transforming a dataset. We use this for training decision trees by selecting the best attribute that maximizes information gain from performing a split.

$$H(before) - H(after)$$

  Where $H(after)$ is a weighted average of the entropy of each subset created after splitting.

Binary Perceptron

A simple linear classifier used to find a decision boundary that perfectly separates the training data into one of two classes. Note that we can generalize the algorithm for multiple classes.

The perceptron algorithm works as follows:

1. Initialize all weights to 0: $\mathbf{w} = \mathbf{0}$

2. For each training sample, with features $\mathbf{f}(x)$ and true class label $y^* \in \{-1, +1\}$, do:

   (a) Classify the sample using the current weights, let y be the class predicted by your current $\mathbf{w}$:

$$y = \text{classify}(x) = \begin{cases} +1 & \text{if activation}_w(x) = \mathbf{w}^T\mathbf{f}(x) > 0 \\ -1 & \text{if activation}_w(x) = \mathbf{w}^T\mathbf{f}(x) < 0 \end{cases}$$

   (b) Compare the predicted label y to the true label y^*:

   - If $y = y^*$, do nothing
   - Otherwise, if $y \neq y^*$, then update your weights: $\mathbf{w} \leftarrow \mathbf{w} + y^*\mathbf{f}(x)$

3. If you went through **every** training sample without having to update your weights (all samples predicted correctly), then terminate. Else, repeat step 2.

Q1. Decision Trees and Information Gain

(a) Suppose we are training a decision tree classifier to predict the price of a widget based on its material and color. Our training data is shown in the table below.

Material	Color	Price
Plastic	Red	$
Ceramic	Blue	$$$
Plastic	Blue	$
Plastic	Blue	$$
Glass	Red	$$$
Ceramic	Blue	$$$$
Glass	Red	$$$$
Plastic	Red	$$

(i) At the root of the tree, what is the information gain (in bits) of splitting on the Material attribute?

1

For notational convenience, we label our classes 1, 2, 3, 4 corresponding to $, $$, $$$, $$$$ respectively. First we calculate the entropy of our training data before we split, estimating p_i as $\frac{n_i}{total}$ where n_i is the number of training points corresponding to class i. Since each price label has exactly two training points, $p_i = \frac{1}{4}$:

$H(\langle p_1, p_2, p_3, p_4 \rangle) = -\sum_{i=1}^{4} p_i \log_2(p_i) = -\sum_{i=1}^{4} \frac{1}{4} \log_2(\frac{1}{4}) = 2$

Now we calculate the entropy of each split and take the weighted average.

1. Split 1 (Plastic): 2 training points from 1 and 2 training points from 2.
   $H(\langle p_1, p_2, p_3, p_4 \rangle) = -\frac{1}{2} \log_2(\frac{1}{2}) - \frac{1}{2} \log_2(\frac{1}{2}) = 1$
2. Split 2 (Ceramic): 1 training point from 3 and 1 training point from 4.
   $H(\langle p_1, p_2, p_3, p_4 \rangle) = -\frac{1}{2} \log_2(\frac{1}{2}) - \frac{1}{2} \log_2(\frac{1}{2}) = 1$
3. Split 3 (Glass): 1 training point from 3 and 1 training points from 4.
   $H(\langle p_1, p_2, p_3, p_4 \rangle) = -\frac{1}{2} \log_2(\frac{1}{2}) - \frac{1}{2} \log_2(\frac{1}{2}) = 1$

Taking the weighted average of the entropy of each split based on the number of training points per split gives us the entropy after splitting: $\frac{1}{2}(1) + \frac{1}{4}(1) + \frac{1}{4}(1) = 1$

Finally we get the information gain by subtracting the entropy after from the entropy before: $2 - 1 = 1$

(ii) At the root of the tree, what is the information gain (in bits) of splitting on the Color attribute?

0

Using the same training set, the original entropy as calculated from the previous part is 2. Now we calculate the entropy of each split and take the weighted average.

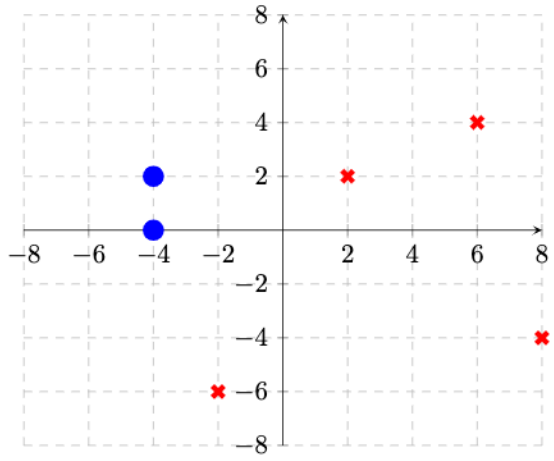
1. Split 1 (Red): 1 training points from each class.
   $H(\langle p_1, p_2, p_3, p_4 \rangle) = -\frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) = 2$
2. Split 2 (Blue): 1 training points from each class.
   $H(\langle p_1, p_2, p_3, p_4 \rangle) = -\frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) - \frac{1}{4} \log_2(\frac{1}{4}) = 2$

Taking the weighted average of the entropy of each split based on the number of training points per split gives us the entropy after splitting: $\frac{1}{2}(2) + \frac{1}{2}(2) = 2$

Finally we get the information gain by subtracting the entropy after from the entropy before: $2 - 2 = 0$

(b) Suppose that we have a dataset and have successfully separated our data points into two distinct classes: **circles and x's**.

(i) The graph depicting the same dataset separated into classes is shown below. Each coordinate corresponds to the values of two features f_1 and f_2 of a datapoint, plotted on the x and y axes respectively:

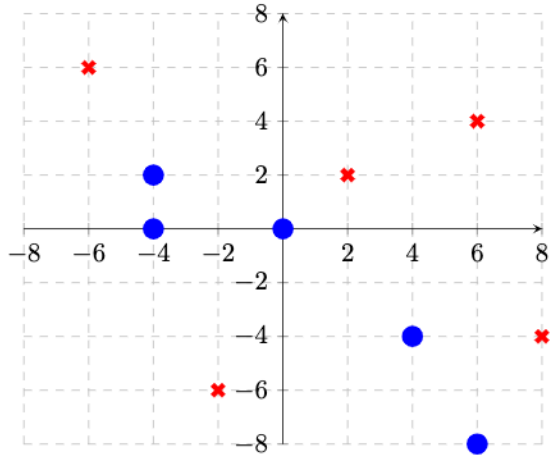


What is the minimum depth of the decision tree that completely separates the two classes? Assume that each decision involves comparing a **single** feature value against a chosen threshold.

- (●) 1
- ( ) 2
- ( ) 3
- ( ) 4
- ( ) 5
- ( ) > 5
- ( ) Cannot be determined

The data is linearly separable by a single line, so we only need to query once to determine if a point is red or blue, which can be drawn as a decision tree of depth 1.

(ii) We recovered some additional data points that were missing from the graph! We've plotted them alongside the original data, and labeled their classes accordingly.



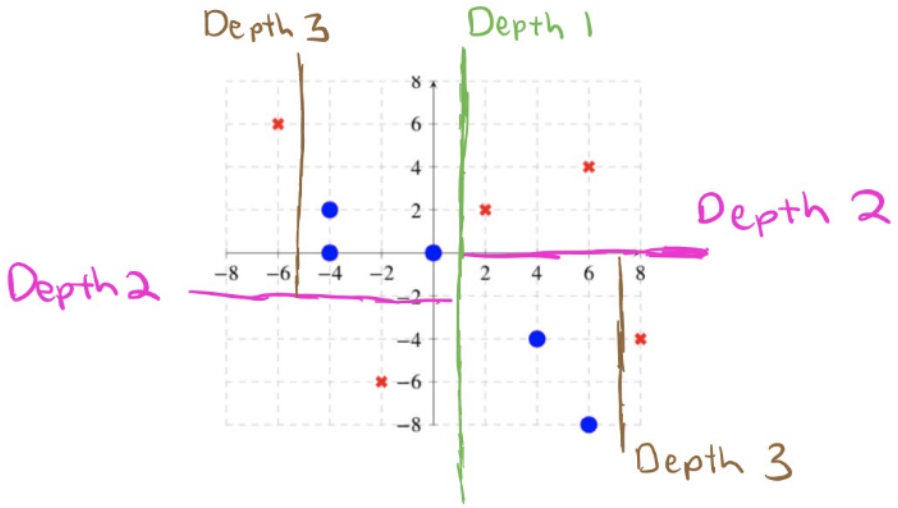
What is the minimum depth of the decision tree that completely separates the two classes now? Assume that each decision involves comparing a **single** feature value against a chosen threshold.

- ( ) 1
- ( ) 2
- (●) 3
- ( ) 4

- ○ 5
- ○ > 5
- ○ Cannot be determined

Intuitively, we want to separate the dataset above using as few horizontal and vertical lines as possible such that our separation will correspond to the minimal depth of the decision tree. The key is to segment off as much of the dataset that is of the same class together that we can; however, the way that the data overlaps on x and y axes make it not so straightforward.

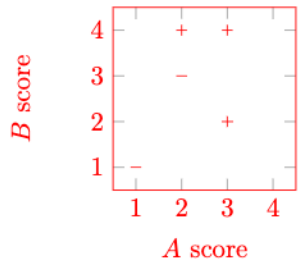
At depth 1 and depth 2, no matter how we section off the dataset into two parts, we will be unable to perform a separation that perfectly separates the data, because of how points of different classes share identical x and y coordinates. Only at depth 3 can we completely separate the data. An example of this is seen below:



2 Perceptron

You want to predict if movies will be profitable based on their screenplays. You hire two critics A and B to read a script you have and rate it on a scale of 1 to 4. The critics are not perfect; here are five data points including the critics' scores and the performance of the movie:

#	Movie Name	A	B	Profit?
1	Pellet Power	1	1	-
2	Ghosts!	3	2	+
3	Pac is Bac	2	4	+
4	Not a Pizza	3	4	+
5	Endless Maze	2	3	-



(a) First, you would like to examine the linear separability of the data. Plot the data on the 2D plane above; label profitable movies with + and non-profitable movies with − and determine if the data are linearly separable.
The data are linearly separable.

(b) Now you decide to use a perceptron to classify your data. Suppose you directly use the scores given above as features, together with a bias feature. That is $f_0 = 1$, $f_1 =$ score given by A and $f_2 =$ score given by B.

Run one pass through the data with the perceptron algorithm, filling out the table below. Go through the data points in order, e.g. using data point #1 at step 1.

step	Weights	Score	Correct?
1	[-1, 0, 0]	$-1 \cdot 1 + 0 \cdot 1 + 0 \cdot 1 = -1$	yes
2	[-1, 0, 0]	$-1 \cdot 1 + 0 \cdot 3 + 0 \cdot 2 = -1$	no
3	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 2 + 2 \cdot 4 = 14$	yes
4	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 3 + 2 \cdot 4 = 17$	yes
5	[0, 3, 2]	$0 \cdot 1 + 3 \cdot 2 + 2 \cdot 3 = 12$	no

Final weights: [-1, 1, -1]

(c) Have weights been learned that separate the data? With the current weights, points will be classified as positive if $-1 \cdot 1 + 1 \cdot A + -1 \cdot B \geq 0$, or $A - B \geq 1$. So we will have incorrect predictions for data points 3:

$$-1 \cdot 1 + 1 \cdot 2 + -1 \cdot 4 = -3 < 0$$

and 4:

$$-1 \cdot 1 + 1 \cdot 3 + -1 \cdot 4 = -2 < 0$$

Note that although point 2 has $w \cdot f = 0$, it will be classified as positive (since we classify as positive if $w \cdot f \geq 0$).

(d) More generally, irrespective of the training data, you want to know if your features are powerful enough to allow you to handle a range of scenarios. Circle the scenarios for which a perceptron using the features above can indeed perfectly classify movies which are profitable according to the given rules:

(a) Your reviewers are awesome: if the total of their scores is more than 8, then the movie will definitely be profitable, and otherwise it won't be. Can classify (consider weights $[-8, 1, 1]$)

(b) Your reviewers are art critics. Your movie will be profitable if and only if each reviewer gives either a score of 2 or a score of 3. Cannot classify

(c) Your reviewers have weird but different tastes. Your movie will be profitable if and only if both reviewers agree. Cannot classify