Decision Trees

Claude Monet, “The Mulberry Tree”
Orange vs Lemon?

What classifiers can we use to classify fruit as oranges or lemons?

- kNN
- Logistic Regression
- Neural Networks
Decision Trees
Decision Trees

Yes: orange
No: lemon
Decision Trees

- **Internal nodes**
  - test the value of particular features $x_j$
  - branch according to the results of the test
- **Leaf nodes** specify the class $h(x)$
- Simpler Example: Predicting whether we’ll play tennis (outputs: Yes, No)

*Features: Outlook ($x_1$), Temperature ($x_2$), Humidity ($x_3$), and Wind ($x_4$).*
  - $x = (\text{Sunny, Hot, High, Strong})$ will be classified as No
  - The **Temperature** feature is irrelevant
Decision Trees

• As close as it gets to an “off-the-shelf” classifier

• Random forests – averages of multiple decision trees classifiers – often perform the best on Kaggle
  – Kaggle.com: a website that hosts machine learning “bake-offs”
  – Though carefully-engineered neural networks and other methods win as well
Decision Trees: Continuous Features

• If the features are continuous, internal nodes may test the value of a feature against a threshold
Decision Trees: Decision Boundaries

- Decision trees divide the feature space into axis-parallel rectangles
- Each rectangle is labelled as one of the K classes
Decision Trees: Model Capacity

- Any Boolean function can be represented
- Might need exponentially many nodes in order to represent the function
Decision Trees: Model Capacity

• As the number of nodes in the tree/the depth of the tree increases, the hypothesis space grows
  – Depth 1 ("decision stump"): can represent any Boolean function of one feature
  – Depth 2: Any Boolean function of two features + some Boolean functions of three features (e.g. \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)\))
  – Etc.

Hypothesis space: the set of all possible functions \(h_\theta(x)\)
Learning the Parity Function

• Suppose we want to learn to distinguish strings of parity 0 and strings of parity 1 — # of 1’s in the string mod 2

• All splits will look equally good!

• Need $2^n$ examples to learn the function correctly

• If there are extra random features, cannot do anything
Learning Decision Trees

• Learning the simplest (smallest) decision tree is an NP-complete problem

• So use a greedy heuristic to construct the tree
  – Start from an empty decision tree
  – Select the best attribute/feature to split on
  – Recurse

But what does “best” mean?
We’ll come back to this.
Learning Decision Trees (Binary Features)

GrowTree(S)
    if (y=0 for all <x,y> ∈ S)
        return new leaf(0)
    else if (y=1 for all <x,y> ∈ S)
        return new leaf(1)
    else
        choose the best attribute $x_j$
        $S_0 = \{<x,y> ∈ S \text{ s.t. } x_j = 0\}$
        $S_1 = \{<x,y> ∈ S \text{ s.t. } x_j = 1\}$
        return new node($x_j$, GrowTree($S_0$), GrowTree($S_1$))
Threshold splits

• For continuous features, need to decide on a threshold $t$
  – Branches: $(x_j < t), (x_j \geq t)$

• Want to allow repeated splits along a path
  – Why?
Set of all possible thresholds

• Branches: \((x_j < t), (x_j \geq t)\)
• Can’t try all real \(t\)
• But only a finite number of \(t\)’s are important

Sort the values of \(x_j\) into \(z_1, \ldots, z_m\), consider split points of the form \(z_i + (z_{i+1} - z_i)/2\)
• Only splits between different examples of different classes matter
Choosing the Best Attribute

- Most straightforward idea: do a 1-step lookahead, and choose the attribute such that if we split on it, we get the lowest error rate on the training data
  - Do a majority vote if not all y’s agree at a leaf

ChooseBestAttribute(S)

Choose \( j \) s.t. \( J_j \) is minimized

\[
S_0 = \langle x, y \rangle \in S \text{ s.t. } x_j = 0
\]

\[
S_1 = \langle x, y \rangle \in S \text{ s.t. } x_j = 1
\]

\( y_0 \): the most common value of \( y \) in \( S_0 \)

\( y_1 \): the most common value of \( y \) in \( S_1 \)

\[
J_0 = \#\{< x, y > \in S_0, y \neq y_0 \}, \quad J_1 = \#\{< x, y > \in S_1, y \neq y_1 \}
\]

\( J_j = J_0 + J_1 \) #total number of errors if we split on \( x_j \)
Choosing the Best Attribute (example)

Splitting on \( x_1 \) produces just two errors, splitting on other attributes produces four errors.
Choosing the Best Attribute

• The number of errors won’t always tell us that we’re making progress
Choosing the Best Attribute

• The number of errors won’t always tell us that we’re making progress

Same number of errors as before the split
A digression on Information Theory

• Suppose $V$ is a random variable with the probability distribution

<table>
<thead>
<tr>
<th>$P(v = 0)$</th>
<th>$P(v = 1)$</th>
<th>$P(v = 2)$</th>
<th>$P(v = 3)$</th>
<th>$P(v = 4)$</th>
<th>$P(v = 5)$</th>
<th>$P(v = 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.002</td>
<td>0.52</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• The surprise $S(V = v)$ for each value of $v$ is defined as

$$S(V = v) = -\log_2 P(V = v)$$

- The smaller the probability of the event, the larger the surprise if we observe the event
- 0 surprise for events with probability 1
- Infinite surprise for events with probability 0
Surprise and Message Length

• Suppose we want to communicate the value of \( v \) to a receiver. It makes sense to use longer binary codes for rarer values of \( V \)

  – Can use \(-\log_2 P(V = v)\) bits to communicate \( v \)

    • Check that this makes sense if \( P(V = 0) = 1 \) (no need to transmit any information) and \( P(V = 0) = P(V = 1) = \frac{1}{2} \) (need one bit to transmit \( v \))

    • Fractional bits only make sense for longer messages

    • Example: UTF-8 uses more bytes for rare symbols

    • “Amount of information”

• We won’t go into this in further detail. We need to make sure that the receiver can decode the message even though different symbols take up different amounts of bits, for example
Entropy: Average/Expected Surprise

• The entropy of $V$, $H(V)$ is defined as

$$H(V) = \sum_{\nu} -P(V = \nu) \log_2 P(V = \nu)$$

• The average surprise for one “trial” of $V$
  – The average message length when communicating the outcome $\nu$

• The average amount of information we get by seeing one value of $V$ (in bits)
Entropy: How “Spread Out” the distribution is

• High entropy of $V$ means we cannot predict what the value of $V$ might be

• Low entropy means we are pretty sure we know what the value of $V$ is every time

The entropy of a Bernoulli variable is maximized when $p = 0.5$
Entropy of Coin Flips

Sequence 1:
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 \ldots \ ?

Sequence 2:
0 1 0 1 0 1 1 1 1 0 1 0 0 1 1 0 1 0 1 1 \ldots \ ?

\[\begin{array}{c}
16 \\
0 \\
1 \\
2 \\
\end{array}\]

versus

\[\begin{array}{c}
8 \\
0 \\
10 \\
1 \\
\end{array}\]
Entropy of Coin Flips

\[ H(V) = \sum_v -P(V = v) \log_2 P(V = v) \]

Higher Entropy; more uncertainty about the outcome
Three views of Entropy

We are considering a random variable $V$, and a sample $v$ from it.

The Entropy is

1. Average Surprise at $v$
2. Average message length when transmitting $v$ in an efficient way
3. Measure of the "spread-out"-ness of the distribution $V$
Conditional Entropy

- The amount of information needed to communicate the outcome of $B$ given that we know $A$

$$H(B|A) = \sum_a P(A = a)H(B|A = a)$$

$$= \sum_a P(A = a)\left[-\sum_b P(B = b|A = a)\log_2 P(B = b|A = a)\right]$$

$H(B)$ if $A$ and $B$ are indep.
0 if $A = B$ always.
Mutual Information

- The amount of information we learn about the value of $B$ by knowing the value of $A$

$$I(A; B) = H(B) - H(B|A) = H(A) - H(A|B)$$

# of bits of information we know about $B$ if we know $A$ = # of bits need to communicate the value of $B$ - # of extra bits need to communicate the value of $B$ if we know $A$

Also called “Information Gain”
Mutual Information

- Suppose the class $Y$ of each training example and the value of feature $x_1$ are random variables. The mutual information quantifies how much $x_1$ tells us about the value of $Y$.

\[
I(Y; X) = H(Y) - H(Y|X) = H(Y) - \sum_x P(X = x)H(Y|X = x)
\]
Mutual Information

• What is the mutual information of this split? (Exercise)
Mutual Information Heuristic

• Pick the attribute $x_j$ such that $I(x_j; Y)$ is as high as possible
Mutual Information Heuristic

- If we had a correct rate of 0.7, and split the data into two groups where the correct rates were 0.6 and 0.8, we will not make progress on the number of errors, but we will make progress on the average $H(Y|X)$.

- We could use any concave function of $p$ instead of computing the conditional entropy in

$$I(Y;X) = H(Y) - H(Y|X) = H(Y) - \sum_x P(X = x)H(Y|X = x)$$
Learning Decisions Trees: Summary
Learning Decisions Trees: Summary
Learning Decisions Trees: Summary
Learning Decisions Trees: Summary

- Graph showing the distribution of oranges (red dots) and lemons (blue triangles) based on height and width.
- Decision tree for sorting oranges from lemons based on height and width criteria: width > 6.5cm?, height > 9.5cm?, height > 6.0cm?.
Aside: Cross Entropy

• The cost function we used when training classifiers was called the Cross Entropy

\[ H(P, Q) = -\sum_{v} P(V = v) \log Q(V = v) \]

• The amount of information we need to transmit if we are using a coding scheme optimized for distribution \( Q \), when the actual distribution over \( V \) is \( P \)
Aside: Cross Entropy

- \( H(P, Q) = -\sum_y P(X = y) \log Q(X = y) \)
- When used as a cost function:
  - P: the observed distribution (we know the answer)
  - Q: what the classifier actually outputs
  - Smaller \( H(P, Q) \) means the distributions P and Q are more similar
    - Different conditional distribution for every value of \( x^{(i)} \)
    - Note: we previously explained the function as the negative log-likelihood of the datapoint. That also works
Missing Attribute Values

• Can use examples with missing attribute values
  – If node \( n \) tests missing attribute \( A \), assign the most common value of attribute \( A \) among the other examples in node \( n \)
  – Assign the most common value of \( A \) among examples with the same target value
  – Assign probability \( p_i \) to each possible value \( v_i \) of \( A \). Assign fraction \( p_i \) of example to each descendent in the tree
Avoiding Overfitting

• Stop growing the tree early
• Or grow full tree, then prune

• The “best” tree:
  – Measure performance on the validation set
  – Measure performance on the training data, but add a penalty term that grows with the size of the tree
Reduced-Error Pruning

• Repeat
  – Evaluate the impact on the validation set of pruning each possible node (and all those below it)
  – Greedily remove the node such that removing the node improves validation set accuracy the most
Effect of Reduced-Error Pruning
Converting a Tree to Rules

IF (Outlook = Sunny) AND (Humidity = High)  
THEN  PlayTennis = No

IF (Outlook = Sunny) AND (Humidity = Normal)  
THEN  PlayTennis = Yes

...
Rule Post-Pruning

• Convert the tree into an equivalent set of rules
  – “If sunny and warm, there will be a tennis match”
  – “If rainy, there will not be a tennis match”
  – ...

• Prune each rule independently of the others
  – Is removing the rule improving validation performance

• Sort the rules into a good sequence for use
Scaling Up

• Decision trees algorithms like ID3 and C4.5 assume random access to memory is fast
  – Good for up to hundreds of thousands of examples

• SPRINT, SLIQ: multiple sequential scans of data
  – OK for millions of examples

• VDFT: at most one sequential scan
  – “stream mode”