CS 486/686
Decision Trees

Yuntian Deng

Lecture 18

RN 19.3 · PM 7.3.1

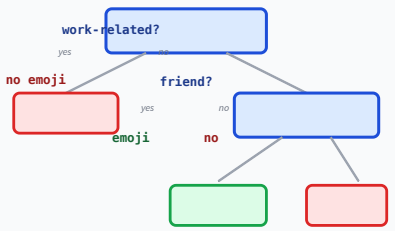
Learning goals

- Describe the **components** of a decision tree.
- **Construct** a decision tree given a feature-testing order.
- Compute a tree's **prediction accuracy** on a test set.
- Compute the **entropy** of a distribution.
- Compute the **expected information gain** of a feature.
- Trace + implement the **DT learning algorithm**.

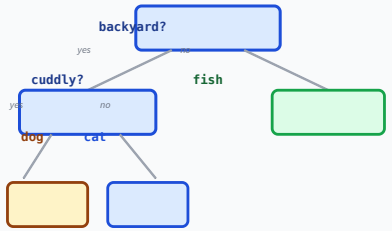
Decision trees in the wild

A decision tree is nothing more than a **nested if-then-else** — learned automatically from data.

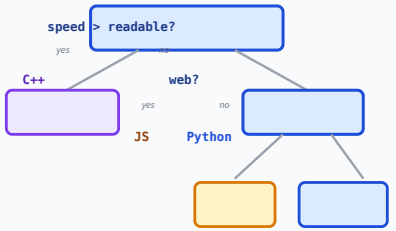
Should I use an emoji?



What pet?



Which language?



Simple to read, fast to evaluate, and surprisingly effective in practice.

Running example: Jeeves the valet

Jeeves wants to predict whether Bertie will play tennis today. He has been recording the weather and Bertie's behavior every morning for 14 days.

Features and target

- **Input features:** Outlook (Sunny/Overcast/Rain), Temp (Hot/Mild/Cool), Humidity (High/Normal), Wind (Weak/Strong).
- **Target:** Tennis? (Yes / No).

14-day training set + 14-day held-out test set. Goal: build a classifier from training, evaluate on test.

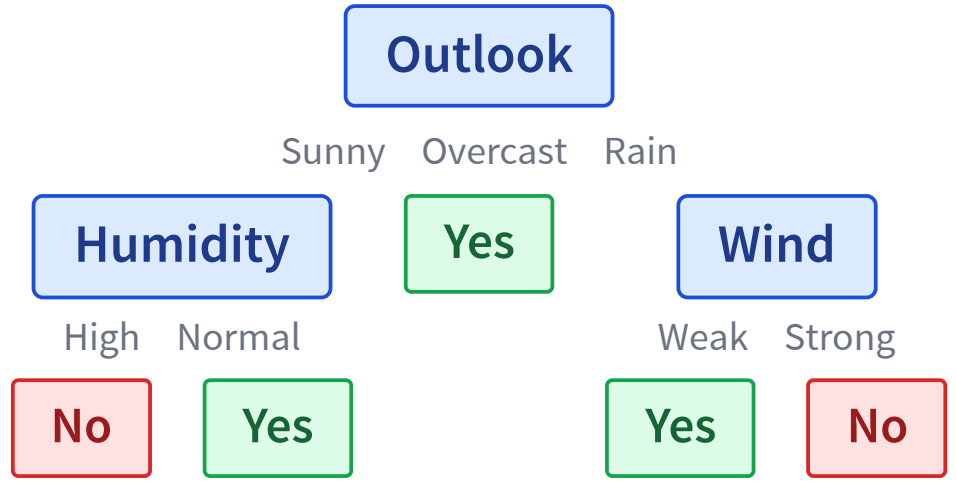
The Jeeves training set

Day	Outlook	Temp	Humidity	Wind	Tennis?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

9 Yes, 5 No. We'll learn a tree from these 14 rows.

Anatomy of a decision tree

- A supervised classifier with a **single discrete target**.
- Each **internal node** tests an input feature.
- **Edges** are labeled with feature values.
- Each **leaf** predicts a target class.
- **Classify**: follow edges from the root down to a leaf.



Sketch only — we'll build the real tree shortly.

Classify an example: walk down the tree

Given a new example, follow the tests from the root to a leaf:

> Example: **Outlook = Overcast, Temp = Hot, Humidity = High, Wind = Weak.**
>
> Test Outlook $\longrightarrow$ value is Overcast $\longrightarrow$ leaf **Yes**. Done.

Observation: the path itself is a nested if-then-else program. Converting a tree to code is straightforward.

Build a tree given an order

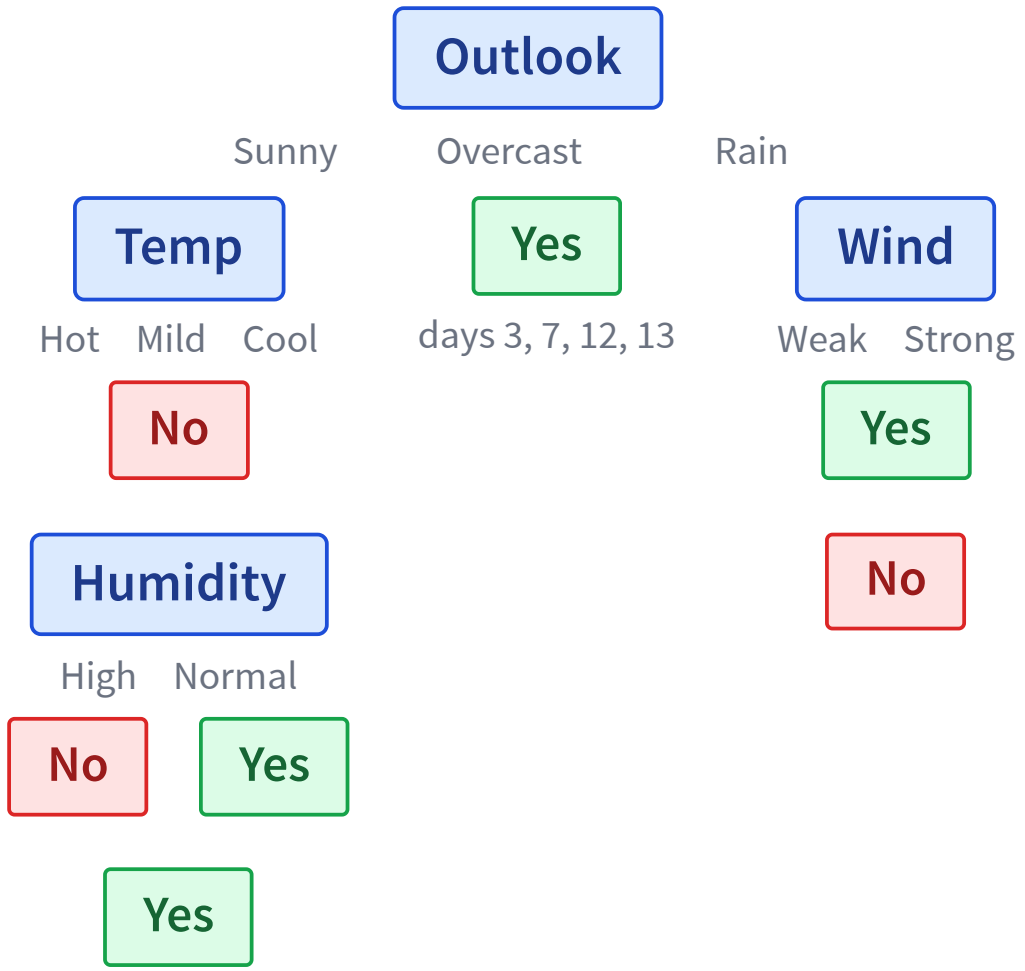
Pick a feature-testing order; recursively split the training examples by that feature's value.

An order for the Jeeves data

- First, test **Outlook**.
- For Outlook = Sunny, test **Temp**.
- For Outlook = Rain, test **Wind**.
- (For other branches, test Humidity before Wind.)

Each example flows down the tree according to its feature values; leaves end up with subsets of the training set.

The Jeeves decision tree



Built from the 14 training rows by following the feature order above.

When do we stop? Three cases

All same class

Every example at this node has the same label.

Return that label.

No features left

We've tested every feature, but examples still disagree (noisy data).

Return the **majority class** at this node.

No examples left

A feature value never appeared in training data — no examples reach this leaf.

Return the **majority class at the parent node**.

"No features left" handles noisy labels; "no examples left" handles unseen feature combinations.

The decision-tree learner

`DT-Learner(examples, features)`

1. If all examples are in the same class, return that class.
2. Else if no features left, return the majority class of *examples*.
3. Else if no examples left, return the majority class at the *parent*.
4. Else: choose a feature f; for each value v of f,
   - build an edge labeled v;
   - recurse on examples with $f = v$ and features $\setminus \{f\}$.

The only open question: **which feature do we pick to test?**

Which feature should we test?

Pick the feature whose split makes the children most **class-pure**.

Split on Outlook

Overcast cleanly predicts **Yes** — one branch is pure!

Split on Temp

Every Temp value still mixes Yes/No — messier.

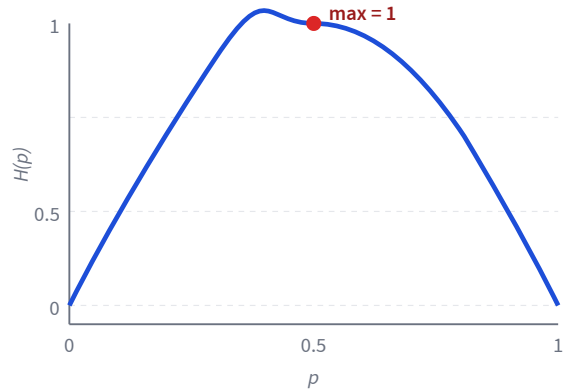
We need a metric to score "purity". Enter **entropy**.

Entropy: measuring uncertainty

Entropy

$$I\big(P(c_1), \ldots, P(c_k)\big) = -\sum_{i=1}^{k} P(c_i)\, \log_2 P(c_i)$$

Bits of uncertainty. Bigger = less certain.



- $I(0.5, 0.5) = 1$: maximally uncertain.
- $I(0.01, 0.99) \approx 0.08$: nearly certain the second outcome.
- $I(1, 0) = I(0, 1) = 0$: no uncertainty at all.

Expected information gain

If we split on a feature with k values, weighted-average the children's entropies:

The three formulas

$$H_{\text{before}} = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right)$$

$$H_{\text{after}} = \sum_{i=1}^{k} \frac{p_i + n_i}{p + n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

$$IG = H_{\text{before}} - H_{\text{after}}$$

Pick the feature with the **largest** information gain at every node.

Jeeves: gain at the root (Outlook)

Training set has 9 Yes, 5 No $(p = 9,\ n = 5)$.

Q3. $H_{\text{before}} = I(9/14, 5/14) = ?$

$\Rightarrow$ **0.940** bits.

Q4. Split on **Outlook**: Sunny (2+/3−), Overcast (4+/0−), Rain (3+/2−).

$$H_{\text{after}} = \tfrac{5}{14}(0.971) + \tfrac{4}{14}(0) + \tfrac{5}{14}(0.971) = 0.694.$$

$$\Rightarrow IG(\text{Outlook}) = 0.940 - 0.694 = \mathbf{0.247}.$$

Jeeves: gain at the root (Humidity) and winner

Same $H_{\text{before}} = 0.940$. Now try splitting on Humidity instead.

Q5. Split on **Humidity**: High (3+/4−), Normal (6+/1−).

$$H_{\text{after}} = \tfrac{7}{14}(0.985) + \tfrac{7}{14}(0.591) = 0.788.$$

$$\Rightarrow IG(\text{Humidity}) = 0.940 - 0.788 = \mathbf{0.151}.$$

Q6. Which feature do we pick as the root?

Outlook — higher IG: $0.247 > 0.151$.

ID3: the canonical algorithm

`ID3(examples, features)`

1. Apply the three base cases from DT-Learner (same-class / no-features / no-examples).
2. For each feature f in features, compute $IG(f)$ on the current examples.
3. Choose $f^\star = \arg\max_f \ IG(f)$; make an internal node testing $f^\star$.
4. For each value v of $f^\star$: recurse on examples with $f^\star = v$ and features $\setminus \{f^\star\}$.

Greedy: pick the locally best feature at each step. Not always globally optimal — but fast and effective.

Learning goals (recap) — Next: neural networks

✓ Describe the **components** of a decision tree.

✓ **Construct** a tree given a feature-testing order.

✓ Compute **test-set accuracy**.

✓ Compute the **entropy** of a distribution.

✓ Compute the **expected information gain**.

✓ Trace + implement the **DT learning algorithm**.

L19: from hand-engineered trees to **neural networks** — the building block of deep learning.