HMM Example – Viterbi Algorithm

Forward Recursion

Transition Probabilities: P(To|From)

From→To	Sunny	Cloudy	Rainy
Sunny	0.25	0.50	0.25
Cloudy	0.33	0.33	0.33
Rainy	0.33	0.67	0.00

Emission Probabilities: P(Behavior|Weather)

Weather→Behavior	Walk	Umbrella
Sunny	1	0
Cloudy	0.67	0.33
Rainy	0.33	0.67

Day	Observation	? →	Sunny	Cloudy	Rainy
		V0(?)	0.333	0.333	0.333
1	Walk	P(W|?)	1.000	0.670	0.330
		V1(?) = V0(?)*P(W|?)	0.333	0.223	0.110
2		V1(S)*P(?|S)		0.167	0.083
		V1(C)*P(?|C)		0.074	0.074
		V1(R)*P(?|R)		0.074	0.000
	Umbrella	P(U|?)	0.000	0.330	0.670
		V2(?) = max(?)*P(U|?)	0.000	0.055	0.056
3		V2(S)*P(?|S)	0.000	0.000	0.000
		V2(C)*P(?|C)	0.018	0.018	0.018
		V2(R)*P(?|R)	0.018	0.037	0.000
	Walk	P(W|?)	1.000	0.670	0.330
		V3(?) = max(?)*P(W|?)	0.018	0.025	0.006

Backtracking

Quiz 2: HMM and MDP

1. Complete the HMM Forward/Backtracking table (reference in the previous slide) using these transition and emission probabilities.

From → To	Sunny	Cloudy	Rainy
Sunny	0.33	0.67	0.00
Cloudy	0.33	0.00	0.67
Rainy	0.33	0.33	0.33

Weather → Behavior	Walk	Umbrella
Sunny	4/4 = 1.0	0/3 = 0.0
Cloudy	2/3 ≈ 0.67	1/3 ≈ 0.33
Rainy	1/3 ≈ 0.33	2/3 ≈ 0.67

2. Complete the MDP process from Canvas → Modules → Jupyter Notebooks → MDP.todo.ipynb

- Construct a table (similar to but not the same as the reference HMM table in the previous slide) that shows the progress of the MDP process.

Submission:

- Follow Canvas/Assignments/Q2: HMM-MD for more detail
- Due: 2/25/2025 before class

Probabilistic Model → Decision Tree

MATH/CSCI 485 Sp25 Advanced Topics in Data Science, Week 5 Session 2

HMM vs MDP

Similarities:

- Both rely on **probabilistic state transitions**.
- Both use the **Markov property**.
- Both can be solved using **dynamic programming techniques**.

Key Insight:

- **HMMs model uncertainty and inference**, while **MDPs model decision-making under uncertainty**.
- MDPs can be thought of as a generalization of HMMs, adding an **action space** and a **reward function**.

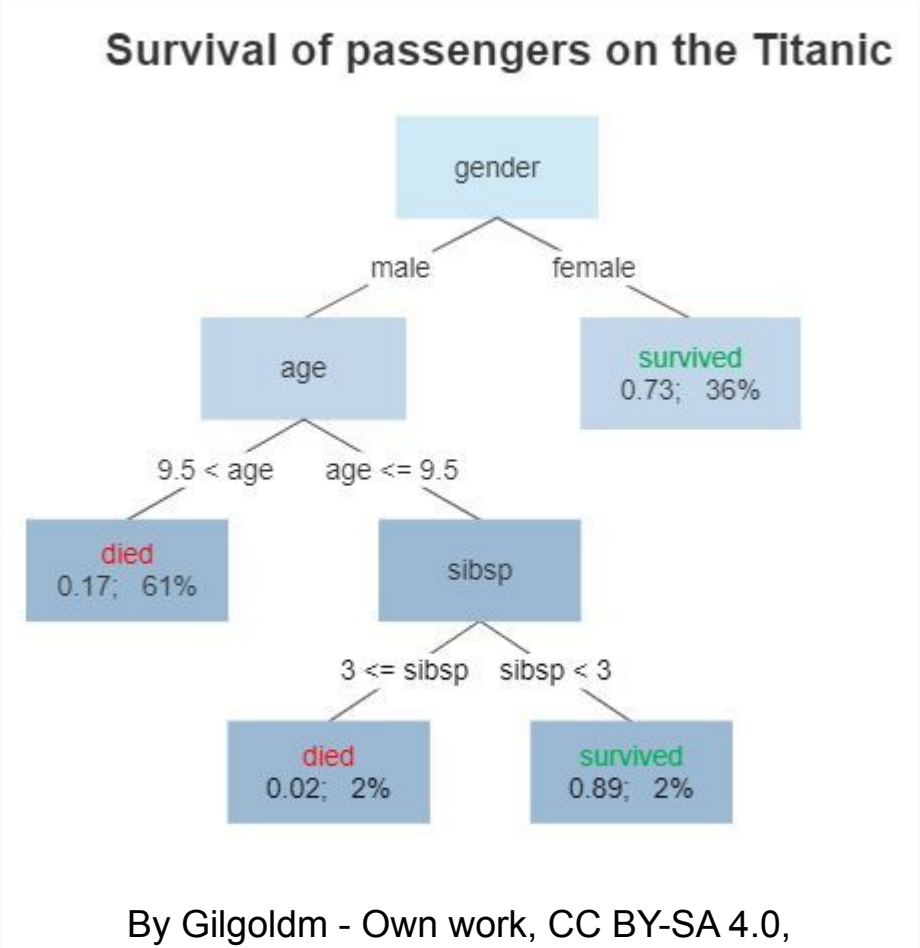
Feature	HMM	MDP
Observability	Partial (Hidden States)	Fully Observable
Decision-Making?	No (purely probabilistic)	Yes (agent optimizes actions)
Goal	State inference from observations	Optimal policy for rewards
Mathematical Tool	Viterbi Algorithm, Forward-Backward	Bellman Equations, Reinforcement Learning

More Comparison

Feature	HMM	MDP
Observability	**Partial (Hidden States)** – You cannot observe the true state directly but only a probabilistic observation.	**Fully Observable** – The agent knows exactly what state it is in.
Actions	No actions – The system evolves naturally without agent intervention.	Has **actions** (A) – The agent **chooses** an action at each step, affecting state transitions.
Goal	Estimate the most likely **hidden state sequence** given observations (e.g., Viterbi algorithm).	Find an **optimal policy** ($\pi(s)$) that maximizes expected reward over time.
Transition Model	($P(S_t \mid S_{t-1})$) – The next state depends only on the previous state via a probability distribution.	($P(S_{t+1} \mid S_t, A_t)$) – The next state depends on both the current state and the action taken.
Rewards	No rewards; purely probabilistic modeling.	Has a **reward function** R(S,A) that defines the objective.

Decision Tree

A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard). The figures under the leaves show the probability of survival and the percentage of observations in the leaf. Summarizing: Your chances of survival were good if you were (i) a female or (ii) a male at most 9.5 years old with strictly fewer than 3 siblings.



What is Decision Tree?

Definition: A decision tree is a flowchart-like structure where each internal node represents a "test" on an attribute, each branch represents the outcome of the test, and each leaf node represents a decision or classification.

Importance: Widely used in machine learning for classification and regression tasks.

Key Features:

- Simple to understand and interpret.
- Can handle both categorical and numerical data.
- No need for extensive data preparation (scaling, normalization).

Anatomy of a Decision Tree

- **Root Node**: Represents the entire dataset and the first decision to make.
- **Internal Nodes**: Represent tests on specific features or attributes.
- **Branches**: Outcomes of a decision that lead to further nodes or leaves.
- **Leaf Nodes**: The final decision or classification.

Decision Tree Construction

1. **Splitting**:
   - The dataset is split based on an attribute that leads to the highest "information gain."
2. **Recursive Partitioning**:
   - The process continues recursively, splitting each subset of data further.
3. **Stopping Criteria**:
   - The algorithm stops when all data points are perfectly classified, or other predefined stopping rules are met (e.g., max depth, minimum samples per leaf).
4. **Prediction**:
   - Once built, the tree can predict the output by navigating from the root to a leaf node for new input data.

Information Gain – Entropy

1. **Entropy**:
   - Measures the uncertainty or impurity in a dataset.
   - Formula: $H(S) = -\sum p(x) \log_2 p(x)$, where $p(x)$ is the proportion of each class in the dataset.
2. **Information Gain**:
   - Represents the reduction in entropy after a dataset is split on an attribute.
   - Formula:

$$IG(S, A) = H(S) - \sum \frac{|S_v|}{|S|} H(S_v)$$

   - We choose the attribute that provides the highest information gain to split the data.

Information Gain – Gini Index

The **Gini Index** is a measure of impurity or disorder used in decision trees. It represents how often a randomly chosen element from the set would be incorrectly classified if it was randomly labeled according to the distribution of labels in the set. The Gini index is defined for a binary classification as:

$$Gini(S) = 1 - \sum_{i=1}^{n} p_i^2$$

Where:

- p_i is the proportion of elements in class i (e.g., how many belong to one class vs. the other).
- n is the total number of classes (usually 2 in a binary problem).

The Gini index is 0 when all items belong to a single class (pure) and maximum (near 0.5) when the items are equally distributed among classes.

The **Weighted Gini Index** takes into account the size of the groups (or "splits") when calculating impurity. It's used to evaluate a split, where the impurity of each group formed by the split is weighted by the size of that group relative to the original set.

$$Weighted\ Gini(Split) = \sum_{i=1}^{k} \left(\frac{|S_i|}{|S|} \times Gini(S_i) \right)$$

Where:

- S is the original set before the split.
- S_i is one of the k groups formed by the split.
- $|S_i|$ is the size (number of elements) in group S_i.
- $|S|$ is the size of the original set.

Initial Gini

Student	Study Hours	Previous Score	Result
1	2	65	F
2	8	85	P
3	3	55	F
4	7	90	P
5	2	75	F
6	6	80	P
7	3	60	F
8	6	62	F
9	1	45	F
10	7	88	P
11	5	82	P
12	4	78	F

Total samples: 12
Pass (P): 5 students
Fail (F): 7 students
Initial Gini = $1 - [(7/12)^2 + (5/12)^2$
$= 1 - [0.34 + 0.17] = 0.49$

Next Step: Compute Gini index for “all” possible split, and pick the smallest one.

Student	Study Hours	Previous Score	Result
1	2	65	F
2	8	85	P
3	3	55	F
4	7	90	P
5	2	75	F
6	6	80	P
7	3	60	F
8	6	62	F
9	1	45	F
10	7	88	P
11	5	82	P
12	4	78	F

Left Branch (Study Hours ≤ 5)

Students: 1, 3, 5, 7, 9, 11, 12
Results: F, F, F, F, F, P, F (6 Fail, 1 Pass)
Gini_left = 1 - [(6/7)² + (1/7)²]
= 0.245

Right Branch (Study Hours > 5)

Students: 2, 4, 6, 8, 10
Results: P, P, P, F, P (1 Fail, 4 Pass)
Gini_right = 1 - [(1/5)² + (4/5)²]
= 0.32

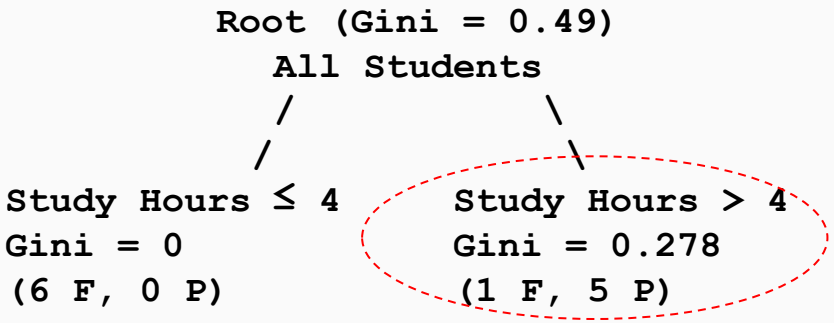
Weighted Gini for this split:

- Weighted Gini = (7/12 × 0.245) + (5/12 × 0.32) = 0.276
- Gini Improvement = 0.49 - 0.276 = 0.214

T	Study Hours ≤ T	F	P	Total	Gini_left	Study Hours > T	F	P	Total	Gini_right	Weighted Gini
3	1,3,5,7,9	5	0	5	0.000	2,4,6,8,10,11,12	2	5	7	0.408	0.238
4	1,3,5,7,9,12	6	0	6	0.000	2,4,6,8,10,11	1	5	6	0.278	0.139
5	1,3,5,7,9,11,12	6	1	7	0.245	2,4,6,8,10	1	4	5	0.320	0.276
6	1,3,5,6,7,8,9,11,12	7	2	9	0.346	2,4,10	0	3	3	0.000	0.259

Pick the smallest

First Level Split Done ⇒ Second Level Decision



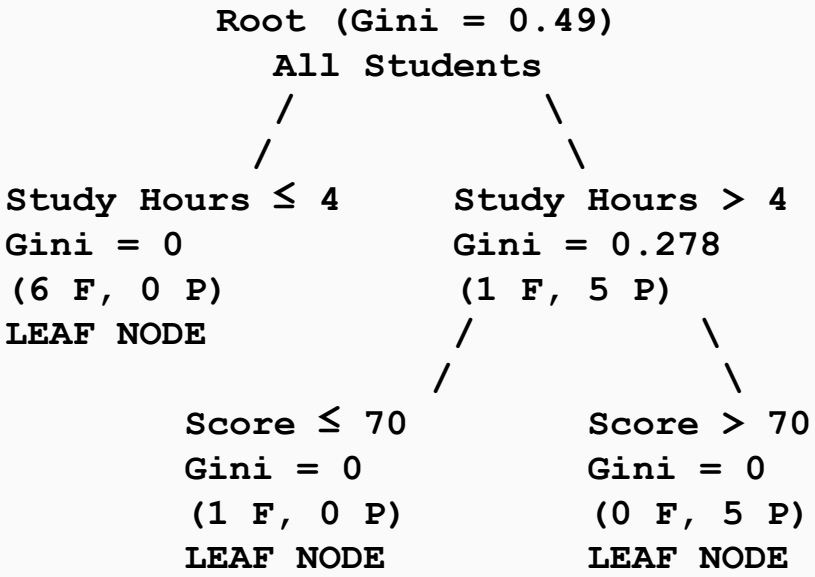
Student	Study Hours	Previous Score	Result
2	8	85	P
4	7	90	P
6	6	80	P
8	6	62	F
10	7	88	P
11	5	82	P

Left (Score ≤ 70):
- Students: 8
- Results: F (1 Fail, 0 Pass)
- Gini_left = 0 (Pure!)

Right (Score > 70):
- Students: 2, 4, 6, 10, 11
- Results: P, P, P, P, P (0 Fail, 5 Pass)
- Gini_right = 0 (Pure!)

Weighted Gini = (1/6 × 0) + (5/6 × 0) = 0
Gini Improvement = 0.278 - 0 = 0.278

Final Decision Tree



Decision Tree Algorithms

ID3 (Iterative Dichotomiser 3):

- Focuses on maximizing information gain.

C4.5:

- Successor of ID3, handles both categorical and continuous data.

CART (Classification and Regression Trees):

- Supports both classification (for categorical targets) and regression (for continuous targets).

Random Forests:

- An ensemble method that builds multiple trees and combines them to improve accuracy.

Avoid Overfitting in Decision Trees

1. **Pruning**:
    - Removes branches that have little importance in classification.
    - Two types: Pre-pruning (set limits on tree depth) and post-pruning (remove after the tree is fully grown).
2. **Setting Maximum Depth**:
    - Restricts the tree from growing too deep and becoming overly complex.
3. **Minimum Samples per Leaf**:
    - Ensures that leaf nodes have a minimum number of data points.

Q&A