Hidden Markov Models

Theory, Algorithms, and Applications

Alexey Kravatskiy

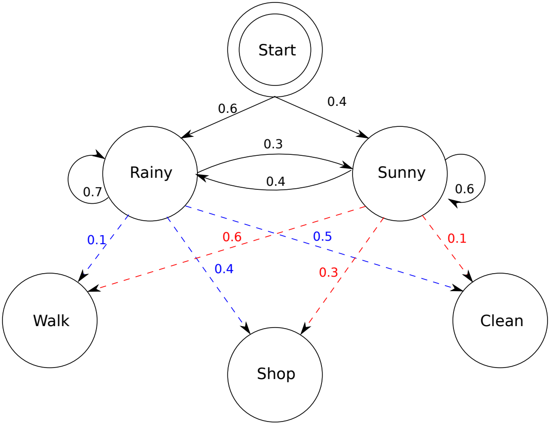
May 31, 2025

Outline

What is a Hidden Markov Model?

- A statistical model that represents a system with hidden states
- Based on Markov processes where future states depend only on the current state
- Consists of:
  - Hidden states X_t (not directly observable)
  - Observable outputs Y_t
  - Transition probabilities between states
  - Emission probabilities for observations

Weather example



Weather prediction. Sunny and Rainy are hidden states, while the activities are observations.

Mathematical Definition

A discrete HMM is defined by:

- N hidden states: $S = \{1, 2, ..., N\}$
- M possible observations: $V = \{v_1, v_2, ..., v_M\}$
- Transition matrix $A = \{a_{ij}\}$ where $a_{ij} = P(X_{t+1} = j | X_t = i)$
- Emission matrix $B = \{b_j(k)\}$ where $b_j(k) = P(Y_t = v_k | X_t = j)$
- Initial state distribution $\pi = \{\pi_i\}$ where $\pi_i = P(X_1 = i)$

Baum-Welch Algorithm for hidden structure

- Also known as the Forward-Backward algorithm
- Used to find the maximum likelihood estimate of HMM parameters
- Given observation sequence $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$
- Finds $\theta^* = \arg\max_\theta P(Y|\theta)$ (or a stationary point)
- Parameters $\theta = (A, B, \pi)$ where:
  - $A = \{a_{ij}\}$: transition probabilities
  - $B = \{b_j(y_i)\}$: emission probabilities
  - π: initial state distribution

Baum-Welch as Expectation Maximization

EM Principle

- Iterative method for finding maximum likelihood estimates
- Handles incomplete data by treating hidden states as missing data
- Each iteration consists of two steps:
  - E-step: Compute expected value of log-likelihood
  - M-step: Maximize this expectation

Forward-Backward Procedure

Forward Variable $\alpha_i(t)$

Probability of observing sequence $y_1, ..., y_t$ and being in state i at time t:

$$\alpha_i(t) = P(Y_1 = y_1, ..., Y_t = y_t, X_t = i|\theta)$$

- Initialization:

$$\alpha_i(1) = \pi_i b_i(y_1)$$

- Recursion:

$$\alpha_j(t) = b_j(y_t) \sum_{i=1}^{N} \alpha_i(t-1) a_{ij}$$

- Termination:

$$P(Y|\theta) = \sum_{i=1}^{N} \alpha_i(T)$$

Forward-Backward Procedure

Backward Variable $\beta_i(t)$

Probability of observing sequence $y_{t+1}, ..., y_T$ given state i at time t:

$$\beta_i(t) = P(Y_{t+1} = y_{t+1}, ..., Y_T = y_T | X_t = i, \theta)$$

- Initialization:

$$\beta_i(T) = 1$$

- Recursion:

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(y_{t+1}) \beta_j(t+1)$$

E-step: Computing Intermediate Variables

- Using Bayes' theorem and forward-backward variables:
  - Probability of being in state i at time t:

$$\gamma_i(t) = P(X_t = i | Y, \theta) = \frac{\alpha_i(t)\beta_i(t)}{P(Y|\theta)}$$

  - Probability of transition from i to j:

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta) = \frac{\alpha_i(t) a_{ij} b_j(y_{t+1}) \beta_j(t+1)}{P(Y|\theta)}$$

- These probabilities are used to compute expected counts:
  - Expected time spent in state i: $\sum_{t=1}^{T} \gamma_i(t)$
  - Expected transitions from i to j: $\sum_{t=1}^{T-1} \xi_{ij}(t)$

M-step: Parameter Updates

- Update parameters to maximize expected log-likelihood:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_j(t) \cdot \delta(y_t, v_k)}{\sum_{t=1}^{T} \gamma_j(t)}$$

$$\bar{\pi}_i = \gamma_i(1)$$

- Each update increases the likelihood:

$$P(Y|\theta_{new}) \geq P(Y|\theta_{old})$$

Convergence Properties

- Baum-Welch is guaranteed to converge
- However, it may converge to:
  - Local maximum (most common)
  - Saddle point (rare)
  - Global maximum (not guaranteed)
- Quality of solution depends on:
  - Initial parameter values
  - Model structure
  - Amount of training data

Viterbi Algorithm for the hidden sequence

- Dynamic programming algorithm
- Finds most likely sequence of hidden states (Viterbi path)
- Given observation sequence $Y = (Y_1 = y_1, Y_2 = y_2, ..., Y_T = y_T)$
- Uses two matrices of size $T \times N$:
  - $P_{t,s}$: maximum probability of ending at state s at time t
  - $Q_{t,s}$: previous state in the maximum probability path

Viterbi Algorithm Steps

1. Initialization ($t = 0$):

$$P_{0,s} = \pi_s \cdot b_s(y_0)$$

$$Q_{0,s} = 0$$

2. Recursion ($t > 0$):

$$P_{t,s} = \max_{r \in S}(P_{t-1,r} \cdot a_{r,s} \cdot b_s(y_t))$$

$$Q_{t,s} = \arg\max_{r \in S}(P_{t-1,r} \cdot a_{r,s})$$

3. Termination:

$$P^* = \max_{s \in S} P_{T-1,s}$$

$$X^*_{T-1} = \arg\max_{s \in S} P_{T-1,s}$$

4. Path backtracking:

$$X^*_t = Q_{t+1}(X^*_{t+1})$$

Applications

- Speech recognition
- Natural language processing
- Bioinformatics
- Financial time series analysis
- Weather forecasting

NLP: Part-of-speech tagging

Brown corpus: 1M tagged words, 15 categories. Universal tagset:

Tag	Meaning	Examples
NOUN	noun	dog, time
VERB	verb	run, is
ADJ	adjective	blue, big
ADV	adverb	quickly, very
PRON	pronoun	I, you, they
DET	determinative	the, a, some
ADP	preposition	in, on, under
NUM	numeral	one, two
CONJ	conjunction	and, but
PRT	particle	up, not
X	unknown	foreign words, typos
.	punctuation	. ! ? ,

Supervised vs unsupervised

Supervised:

- maximum likelihood estimation for training and Viterbi for predicting
- all train dataset ($\approx$ 45 000 sentences), 1.6 sec training, 1.5 min predicting: 73% test accuracy
- random 1000 sentences: 22%
- random 10 000 sentences: 52%.
- random 30 000 sentences: 67%.

Unsupervised:

- Baum-Welch takes much more time (105 min)
- First 1 000 sentences with tagging, supervised, then all unsupervised. Accuracy = 67% after 10 iterations.

Additive Smoothing

Lidstone smoothing:

$$P_{\text{lidstone}}(w) = \frac{\text{count}(w) + \gamma}{N + \gamma \times V}$$

where:

- $\text{count}(w)$ is the raw count of outcome w,
- $N = \sum_x \text{count}(x)$ is the total number of observations in `freqdist`,
- $V =$ `bins` is the total number of distinct possible outcomes (e.g., vocabulary size).

Supervised:

- $\gamma = 0.1$ for the estimator
- all train dataset: 95% test accuracy

Unsupervised:

- First 1 000 sentences with tagging, supervised, then all unsupervised. Accuracy = 57% after 10 iterations.

Thank You!

Questions?