Tufts
UNIVERSITY

MARKOV MODELS 2

ARTIFICIAL INTELLIGENCE | COMP 131

TODAY ON AI

- Inference for classification
- Inference in temporal models
- Likelihood with the Forward algorithm
- Decoding with the Viterbi algorithm
- Learning with the Baum-Welch algorithm
- Questions?

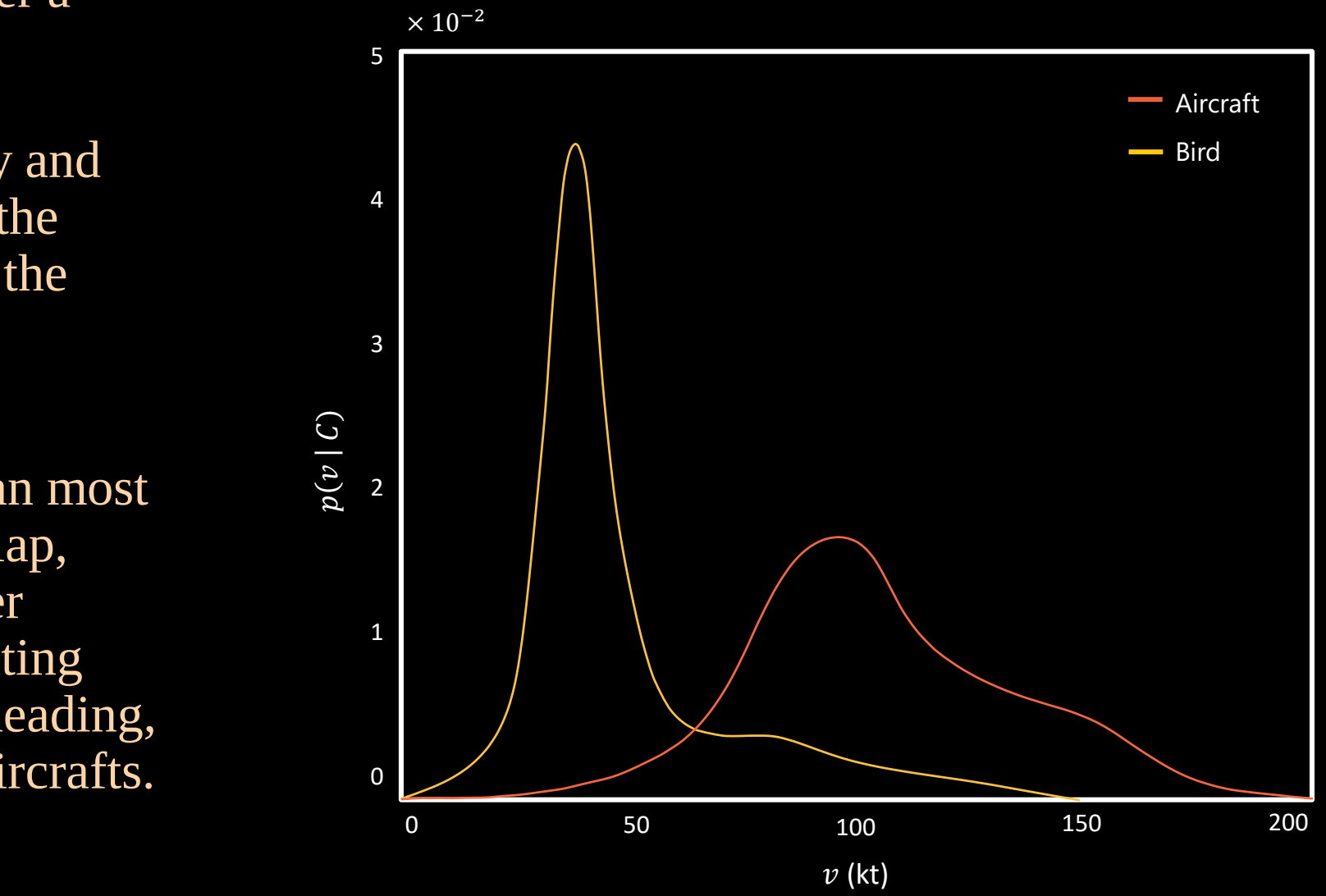
- **Conditional probability**: $P(X|Y) = \frac{P(X,Y)}{P(Y)}$
- **Product rule**: $P(X,Y) = P(X|Y)P(Y)$
- **Chain rule**: $P(X_1, \dots, X_n) = \prod_i P(X_i|X_1, \dots, X_{i-1})$
- **$\mathbf{X}$ and $\mathbf{Y}$ are independent iff** $P(X,Y) = P(X)P(Y)$
- **$\mathbf{X}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{Z}$ iff** $X \perp Y|Z$: $P(X,Y|Z) = P(X|Z)P(Y|Z)$
- **Bayes rule**: $P(Cause|Effect) = \frac{P(Effect|Cause)\,P(Cause)}{P(Effect)}$
- **Law of total probability**: $P(A) = \sum_n P(A|B_n)\,P(B_n)$

SECTION 01

Inference for classification

Suppose we want to determine whether a radar target is either a bird or an aircraft, and the observations might include measurements of the velocity and the amount of fluctuation in the heading over the duration of the track.

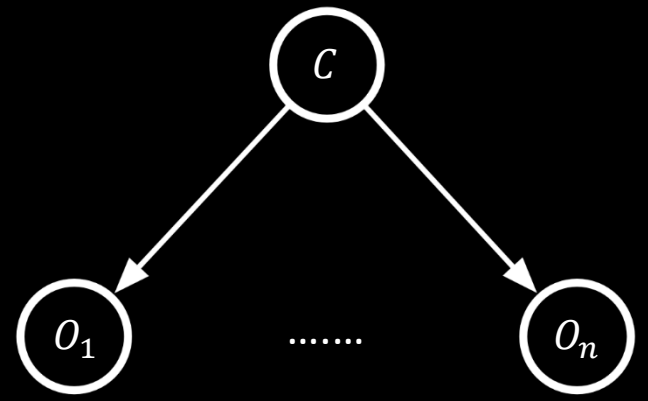
Most aircraft travel faster than most birds, but there is some overlap, especially with smaller, lower performance aircrafts. Migrating birds tend to maintain their heading, in contrast to maneuvering aircrafts.



Inference can be used for **classification** tasks.

A simple probabilistic model often used in classification tasks is the **Naïve Bayes model**.

In the naïve Bayes model, the class C is the query variable, and the **observed features** $O_1 \ldots O_n$ are the **evidence variables**.



The naïve Bayes model is called **naïve** because it assumes **conditional independence** between the evidence variables given the class.

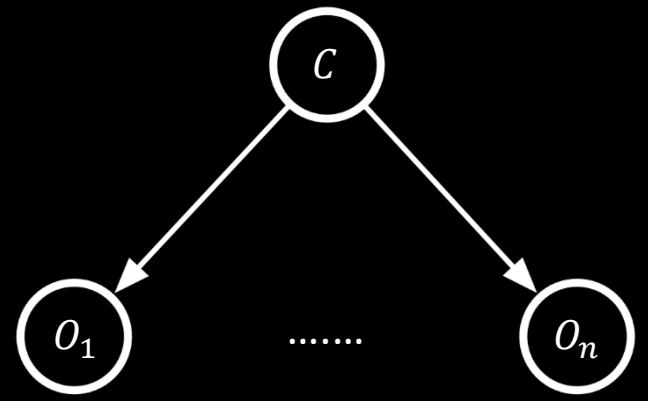
In the naïve Bayes model we must specify the prior $P(C)$ and the class-conditional distribution $P(O_i|C)$:

$$P(c, o_{1:n}) = P(c)\prod_{i=1}^{n} P(o_i|c)$$

$$P(c \mid o_{1:n}) = \frac{P(c, o_{1:n})}{P(o_{1:n})} \qquad P(o_{1:n}) = \sum_{c} P(c, o_{1:n})$$

$$P(c \mid o_{1:n}) = \chi P(c, o_{1:n})$$

$$P(c \mid o_{1:n}) \propto P(c, o_{1:n})$$



SECTION 01

Inference in temporal models

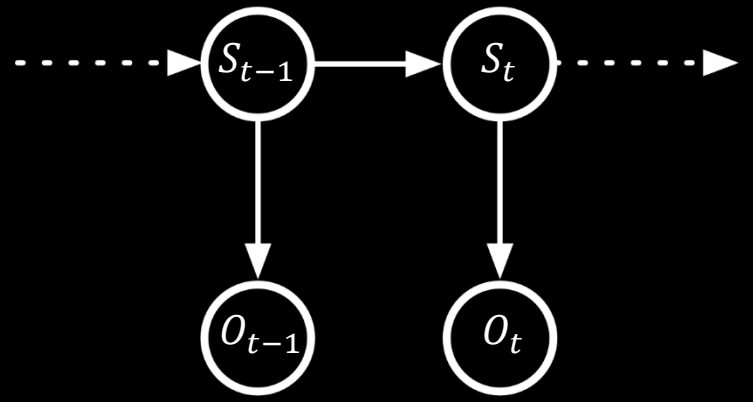
Many important applications involve performing inference in temporal models:

- **Filtering**: $P(S_t | O_{0:t})$
- **Prediction**: $P(S_{t'} | O_{0:t})$ where $t' > t$
- **Smoothing**: $P(S_{t'} | O_{0:t})$ where $t' < t$
- **Most likely explanation**: $\underset{S_{0:t}}{\operatorname{argmax}} P(S_{0:t} | O_{0:t})$

We can the problem as a Hidden Markov structure:

By the Bayes' rule: $P(s_t|o_{0:t}) \propto P(o_t|s_t, o_{0:t-1})\, P(s_t|o_{0:t-1})$

Because of the **Hidden Markov** properties and the **total law of probability**:



$O_t \perp O_{0:t-1} \mid S_t$ $\qquad P(s_t|o_{0:t}) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t, s_{t-1}|o_{0:t-1})$

Applying the definition of **conditional probability** to $P(s_t, s_{t-1}|o_{0:t-1})$

$$P(s_t|o_{0:t}) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, o_{0:t-1}) P(s_{t-1}|o_{0:t-1})$$

Because of the **conditional independence** $s_t \perp o_{0:t-1} \mid s_{t-1}$

$$P(s_t|o_{0:t}) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}) P(s_{t-1}|o_{0:t-1})$$

$$P(s_t|o_{0:t}) \propto \underbrace{P(o_t|s_t)}_{\text{KNOWN}} \sum_{s_{t-1}} \underbrace{P(s_t|s_{t-1})}_{\text{KNOWN}} P(s_{t-1}|o_{0:t-1})$$

We **know** $P(o_t\,|s_t)$ and $P(s_t\,|s_{t-1})$ directly from the **model**.

The probability $P(s_{t-1}|o_{0:t-1})$ can be evaluated recursively in a process called **recursive Bayesian estimation**:

```
1  function RecursiveBayesianEstimation()
2  b_0(s) = P(o_0| s)P(s_0) for all s
3  Normalize b_0
4  for t = 1 to ∞
5    b_t(s) = P(o_t| s) Σ_s' P(s | s') b_t_1(s') for all s
6    Normalize b_t
```

Recursive Bayesian estimation
ALGORITHM

SECTION 01

Likelihood or Forward algorithm

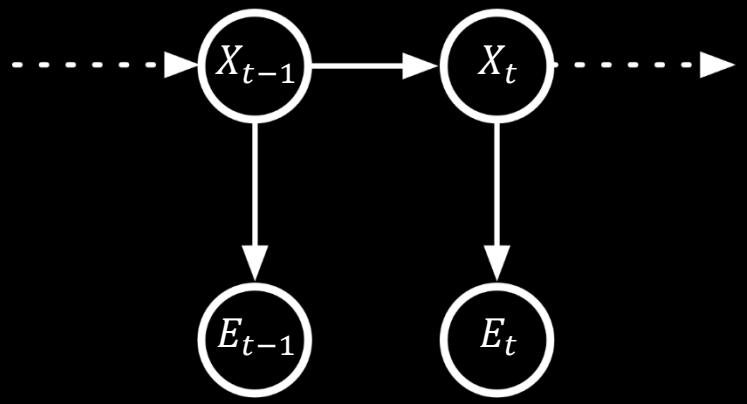
The likelihood of a sequence, or Forward algorithm, given an HMM $H = (P(X_1), T, E)$ and a sequence of observations E, determines the likelihood $P(E|H)$ of E.

We want to find $P(X_t, E_{1:t})$?

$$P(X_t, E_{1:t}) = \sum_{X_{t-1}} P(X_t, X_{t-1}, E_{1:t}) \qquad l_t(X_t) = \sum_{X_{t-1}} P(X_t, X_{t-1}, E_{1:t})$$

$$= \sum_{X_{t-1}} P(E_t|X_t, X_{t-1}, E_{1:t-1})\, P(X_t|X_{t-1}, E_{1:t-1})\, P(X_{t-1}, E_{1:t-1})$$

$$= \sum_{X_{t-1}} \underbrace{P(E_t|X_t)}_{E}\, \underbrace{P(X_t|X_{t-1})}_{T}\, l_{t-1}(X_{t-1})$$



where $l_1(X_1) = P(E_1|X_1)\, P(X_1)$

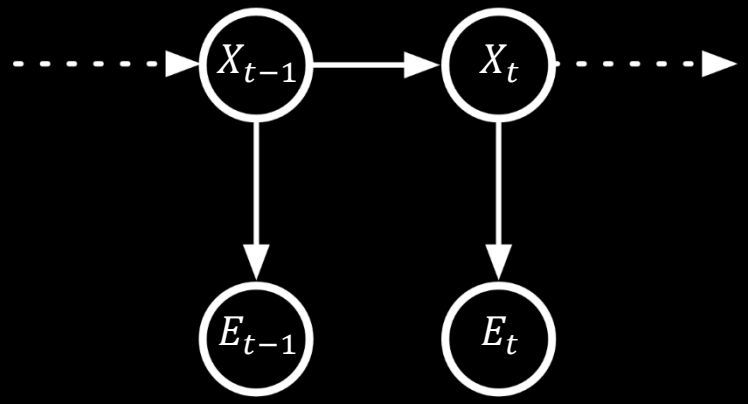
The complexity of this algorithm is $O(tm^2)$ where t is the length of the evidence sequence and m is the cardinality of X.

SECTION 01

Decoding or Viterbi algorithm

The decoding or Viterbi algorithm, given an HMM $H = (P(X_1), T, E)$ and a sequence of observations E discovers the best hidden state sequence Q that generated the sequence.

We want to find $X^*_{1:t} = \underset{X_{1:t}}{\operatorname{argmax}} P(X_{1:t}|E_{1:t})$



$$X^*_{1:t} = \underset{X_{1:t}}{\operatorname{argmax}} P(X_{1:t}, E_{1:t}) \qquad \mu_t(X_t) = \max_{X_{1:t}} P(X_{1:t}, E_{1:t})$$

$$\mu_t(X_t) = \max_{X_{1:t-1}} P(E_t|X_t)\, P(X_t|X_{t-1})\, P(X_{t-1}, E_{1:t-1})$$

If $f(x) \geq 0\ \forall x$ and $h(x,y) \geq 0\ \forall x,y$ then $\max_{x,y}[f(x)\, h(x,y)] = \max_x \left[f(x) \max_y h(x,y)\right]$

$$\mu_t(X_t) = \max_{X_{1:t-1}} P(E_t|X_t)\, P(X_t|X_{t-1}) \max_{X_{1:t-2}} P(X_{t-1}, E_{1:t-1})$$

$$\mu_t(X_t) = \max_{X_{1:t-1}} \underbrace{P(E_t|X_t)}_{E}\, \underbrace{P(X_t|X_{t-1})}_{T}\, \mu_{t-1}(X_{t-1}) \qquad \text{where } \mu_1(X_1) = P(E_1|X_1)\, P(X_1)$$

SECTION 01

Learning or Baum-Welch algorithm

The Baum-Welch algorithm, given a set of states in an HMM $H = (P(X_1), T, E)$ and a sequence of observations E, learn the HMM parameters T and E.

The Baum-Welch algorithm is the specific implementation for HMMs of a family of algorithms called **Expectation Maximization** or **EM**.

Assuming θ as the missing transition and emission probabilities of an HMM, with want to find:

$$\theta^*(E) = \max_{\theta} \sum_{X} P_{\theta}(E, X)$$

The Baum-Welch algorithm is the specific implementation for HMMs of a family of algorithms called **Expectation Maximization** or **EM**.

It is based on the repetition of two steps until convergence:

- Initialization: Give $\theta = \theta_0$
- For a finite number of steps:
  - **E-step**: $Q(\theta, \theta_t) = \sum_E \log\left[P_{\theta_t}(X, e)\right] P_{\theta_t}(e|X)$
  - **M-step**: $\theta_{t+1} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta, \theta_t)$

GOOD $P_{\theta_{t+1}}(E) \geq P_{\theta_t}(E)$ and it works well in most cases

BAD It might get stuck in local maxima. Convergence might be slow. It might overfit the observed data.

READINGS

Chapter 15, 16, and 17

QUESTIONS ?

Tufts
UNIVERSITY

ARTIFICIAL INTELLIGENCE

COMP 131

FABRIZIO SANTINI

VERSION 3.1