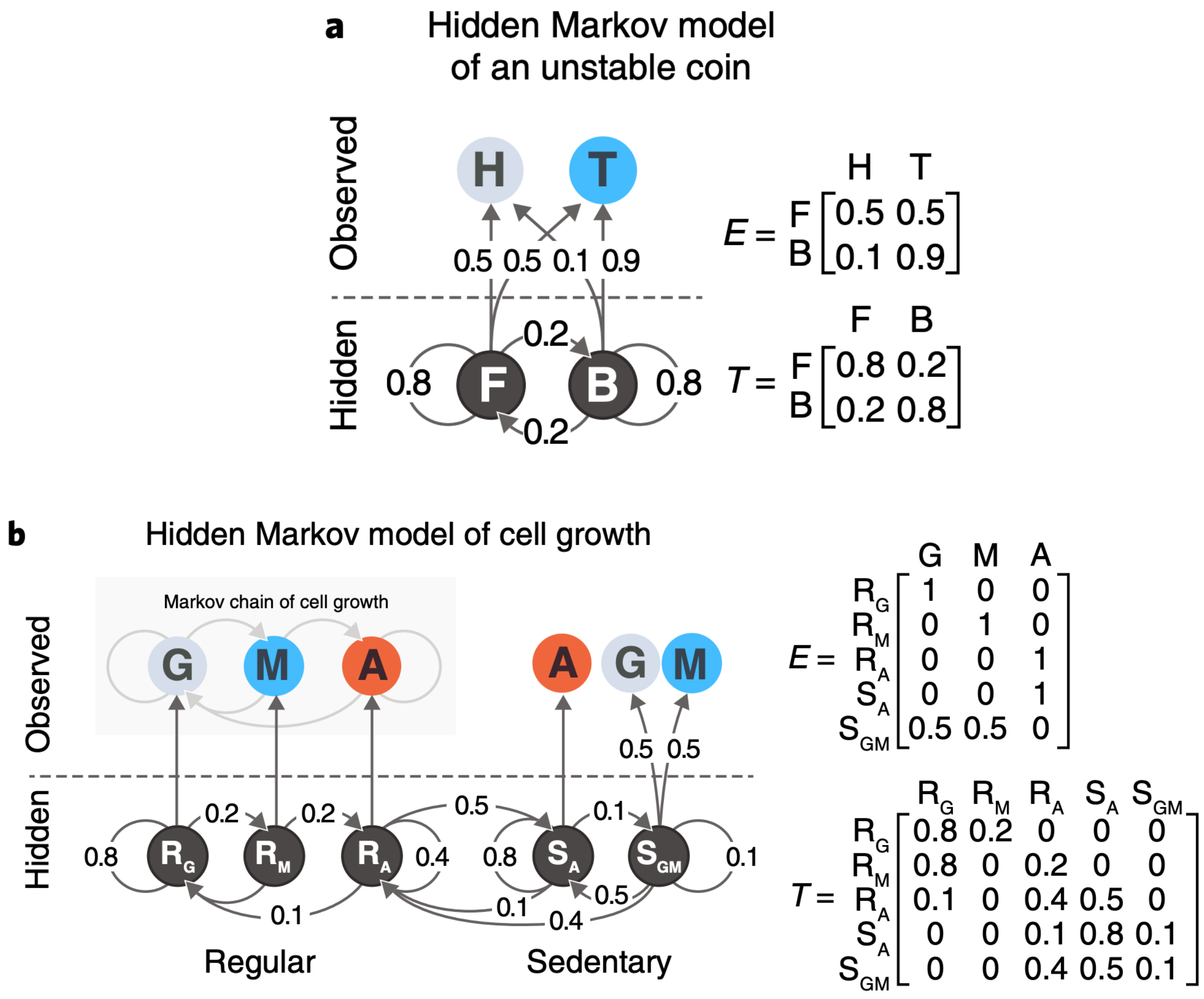
BE 175/275: Machine learning & data-driven modeling in bioengineering

Discussion 7

Topics

- Hidden Markov Model
- Homework: Implementation of HMMs on real-world heart rate data.

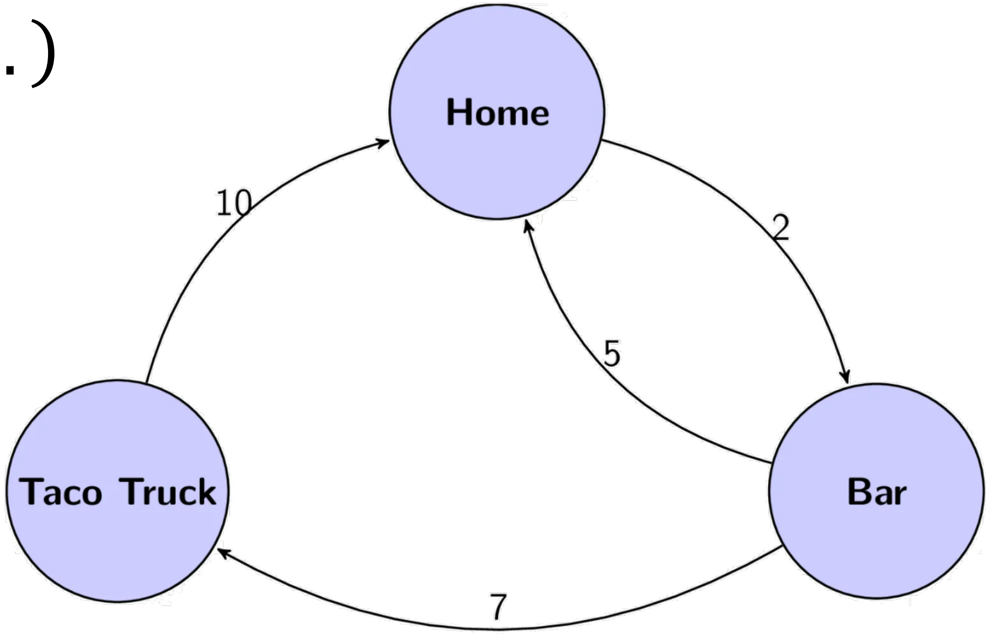


From the [Nature paper]

So, what is so "Markov" about it?

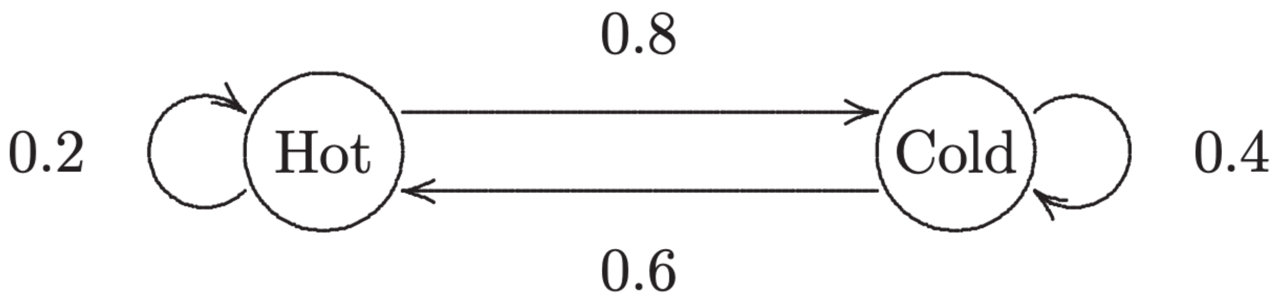
- **Markov Property**: The future depends only on the present state and does not depend on past history.
- $P(X_n|X_{n-1}) = P(X_n|X_{n-1}, X_{n-2}, X_{n-3}, \dots)$

	State Observable	**State not fully observable**
Autonomous	Markov Chain	Hidden Markov Model
Controlled	Markov Decision Process	Partially observable MDP



Markov Chain

- State at time t is the value of X_t
- State space is the set of values that each X_t can take
- Trajectory is a particular set of values for X_0, X_1, X_2, …
- Transition probability $p_{ij} = \mathbb{P}(X_{t+1} = j \mid X_t = i).$
- Transition matrix P = (p_{ij})



$$
\begin{array}{cc}
 & \overbrace{\begin{array}{cc}\text{Hot} & \text{Cold}\end{array}}^{X_{t+1}} \\
X_t \left\{ \begin{array}{c}\text{Hot} \\ \text{Cold}\end{array} \right. & \begin{pmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \end{pmatrix}
\end{array}
$$

Example from [Rachel Fewster]

What is hidden in the Hidden Markov Model?

- HMM as a 5-tuple
  - λ(S, O, p, q , π)
- States (S)
- Observations (O)
- Transition probabilities (p)

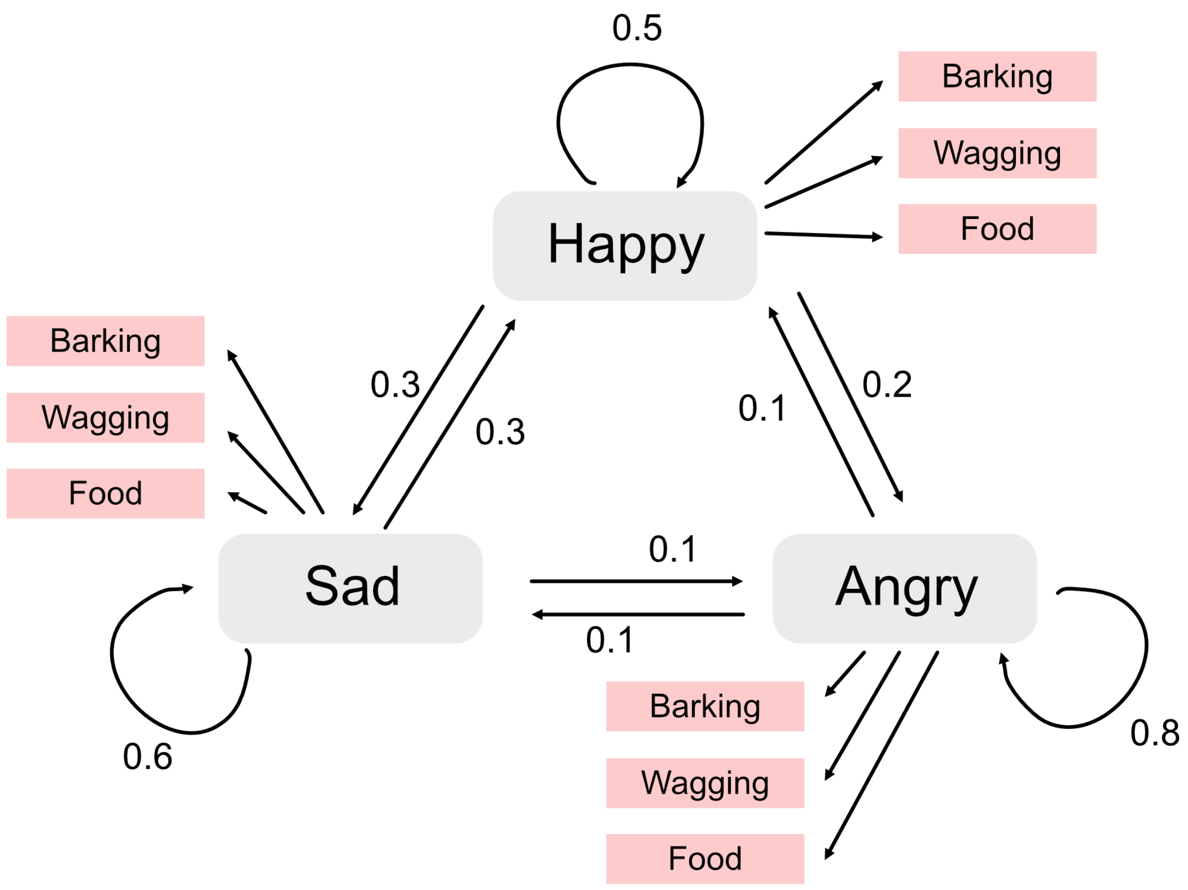
$$p(S_t = j \mid S_{t-1} = i) = p_{ij}$$

- Emission probabilities (q)

$$p(O_t = y \mid S_t = i) = q_i^y$$

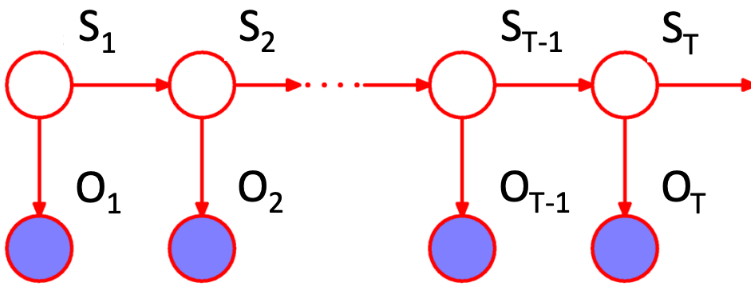
- Initial probabilities (π)

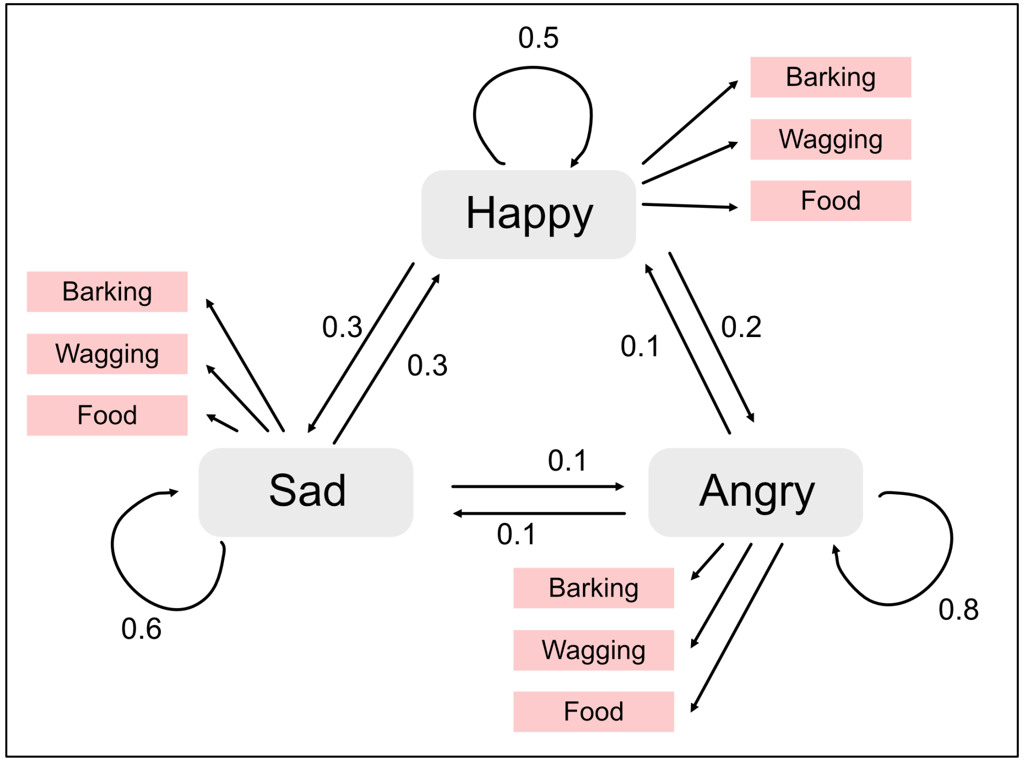
$$p(S_1 = i) = \pi_i$$

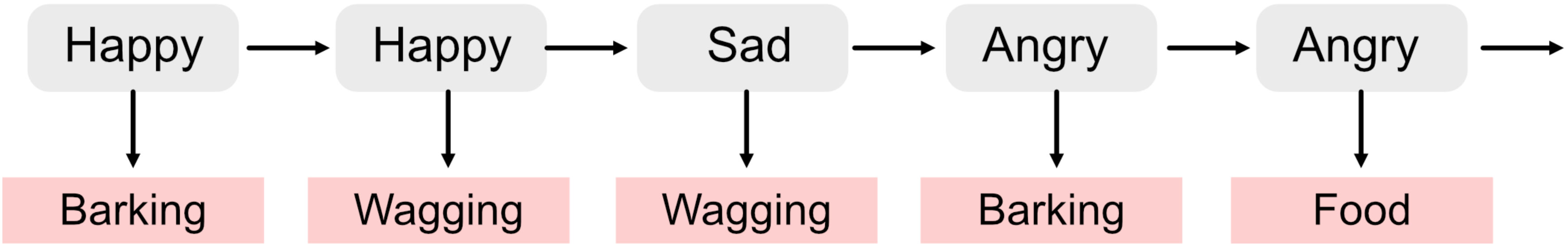


Many diagrams of HMM

- States and observations



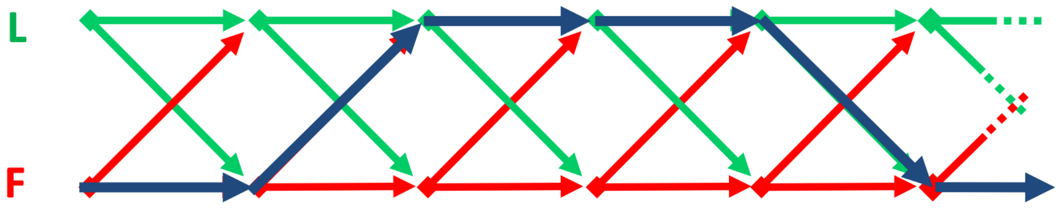


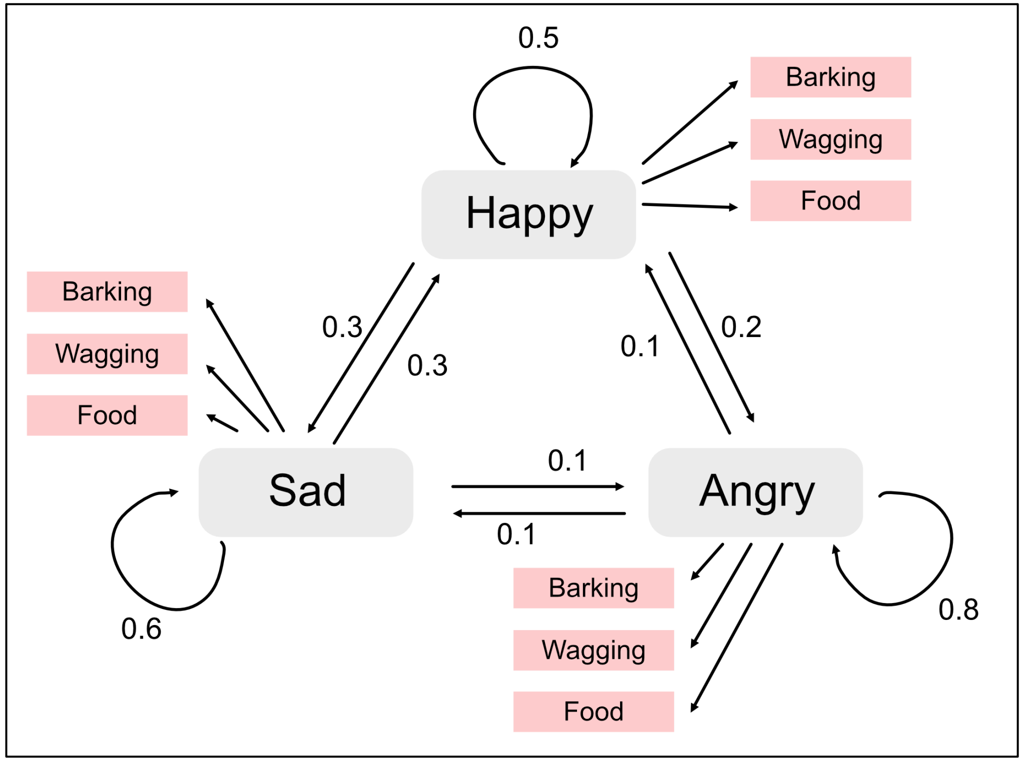


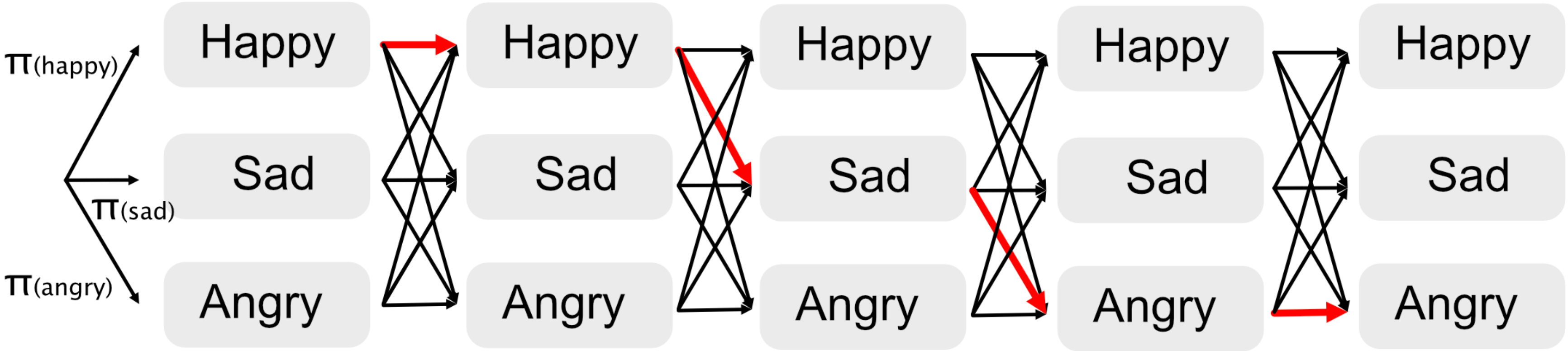
- State sequence, observation sequence

Many diagrams of HMM

- State transition



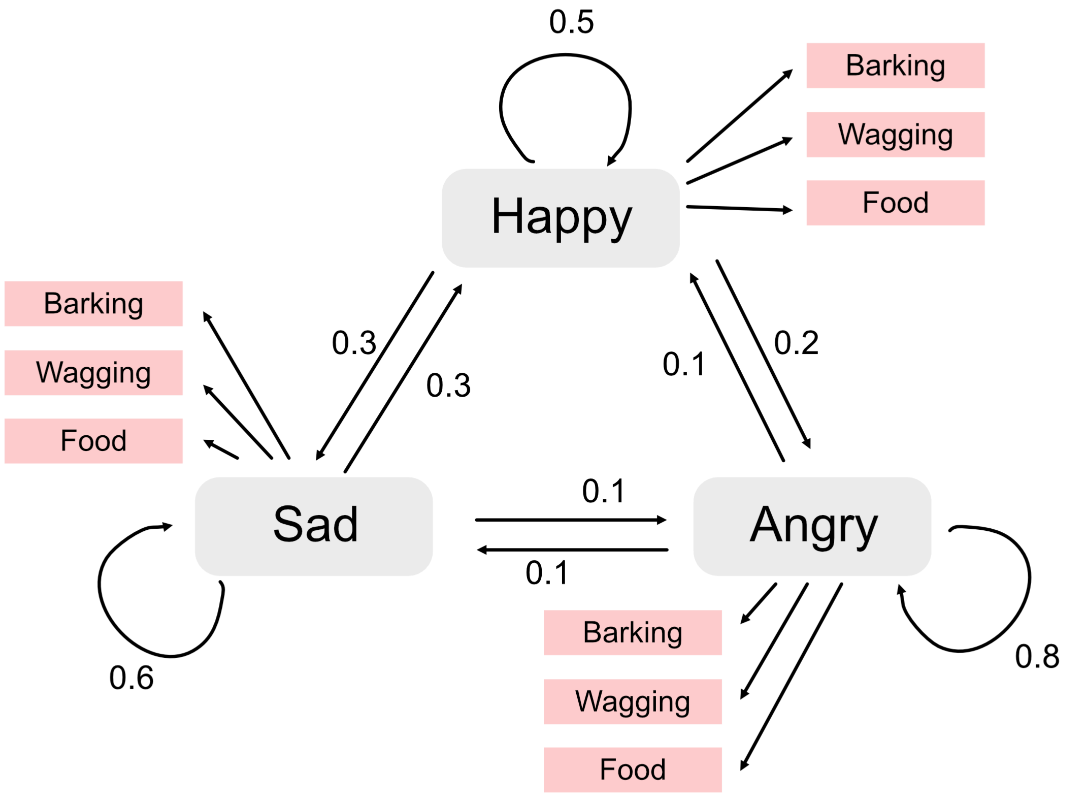




Three fundamental problems of HMM

- **Evaluation** Given an HMM λ(p, q) and an observation sequence O, determine the likelihood P(O | λ).
  - **Forward Algorithm**
- **Decoding** Given an observation sequence O and an HMM λ(p, q), discover the best hidden state sequence S.
  - One time point: **Forward Backward Algorithm**
  - Sequence: **Viterbi Algorithm**
- **Learning** Given an observation sequence O and the set of states in the HMM, learn the HMM parameters p, q.
  - Expectation Maximization – **Baum-Welch Algorithm**

In class example



- States (S): H, S, A
- Observations (O): B, W, F
- Transition probabilities (p_{SS})

t

p	H	S	A
H	0.5	0.3	0.1
S	0.3	0.6	0.1
A	0.2	0.1	0.8

t+1

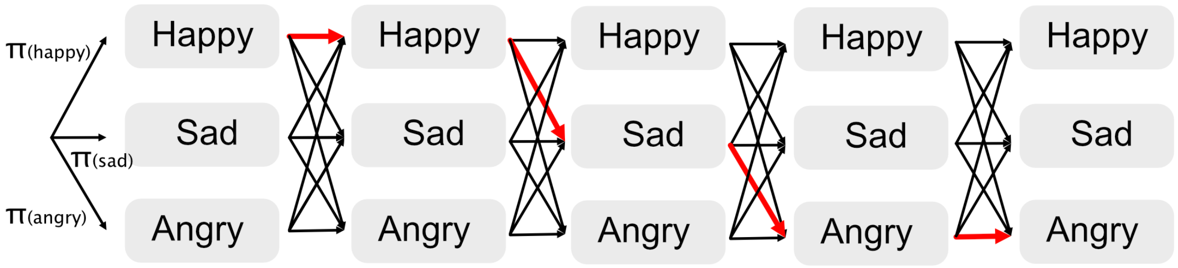
- Emission probabilities (q_S^O)

q	H	S	A
B	0	0	0.9
W	0.8	1	0
F	0.2	0	0.1

- Initial probability (π_S)

π	H	S	A
	1	0	0

Dynamic Programming



- Expedite computation by storing the result of previous calculations (Memoization)
- In Forward-Backward algorithm
  - The probability of observing this series before/after current time point AND the current state is k

$$
\begin{aligned}
p(S_t = k, \{O_t\}_{t=1}^{T}) &= p(O_1, \ldots, O_t, S_t = k, O_{t+1}, \ldots, O_T) \\
&= \underbrace{p(O_1, \ldots, O_t, S_t = k)}_{\alpha_t^k}\underbrace{p(O_{t+1}, \ldots, O_T | S_t = k)}_{\beta_t^k}
\end{aligned}
$$

Compute recursively

- In Viterbi algorithm

Forward Algorithm

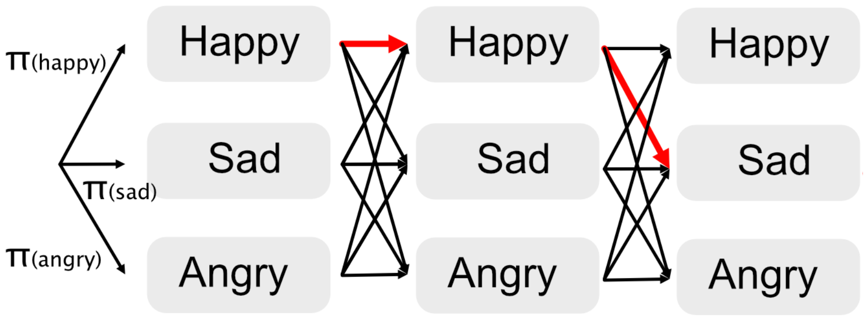
p	H	S	A
H	0.5	0.3	0.1
S	0.3	0.6	0.1
A	0.2	0.1	0.8
q	H	S	A
B	0	0	0.9
W	0.8	1	0
F	0.2	0	0.1
π	H	S	A
	1	0	0

- Find the probability of observing "wagging, food, barking" in the first three days?
  - Initialize: $\alpha_1^k = p(O_1 \mid S_1 = k)p(S_1 = k) \;\; \forall k$
  - Iterate: for $t = 2, \ldots, T$

$$\alpha_t^k = p(O_t \mid S_t = k) \sum_i \alpha_{t-1}^i p(S_t = k \mid S_{t-1} = i) \;\; \forall k$$

  - Termination:

$$p\left(\{O_t\}_{t=1}^T\right) = \sum_k \alpha_T^k$$



- Saving α along the way

Forward-Backward Algorithm

p	H	S	A
H	0.5	0.3	0.1
S	0.3	0.6	0.1
A	0.2	0.1	0.8

q	H	S	A
B	0	0	0.9
W	0.8	1	0
F	0.2	0	0.1

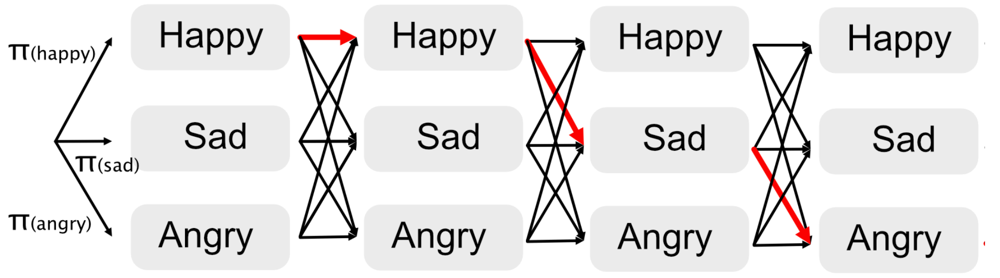
π	H	S	A
	1	0	0

- If you observe “wagging, food, barking, food” in the first four days, what is the probability that it is **angry** at day 3?
  - Initialize: $\beta_T^k = 1 \quad \forall k$
  - Iterate: for t = T-1, . . . , 1

$$\beta_t^k = \sum_i p(S_{t+1} = i \mid S_t = k) p(O_{t+1} \mid S_{t+1} = i) \beta_{t+1}^i \quad \forall k$$

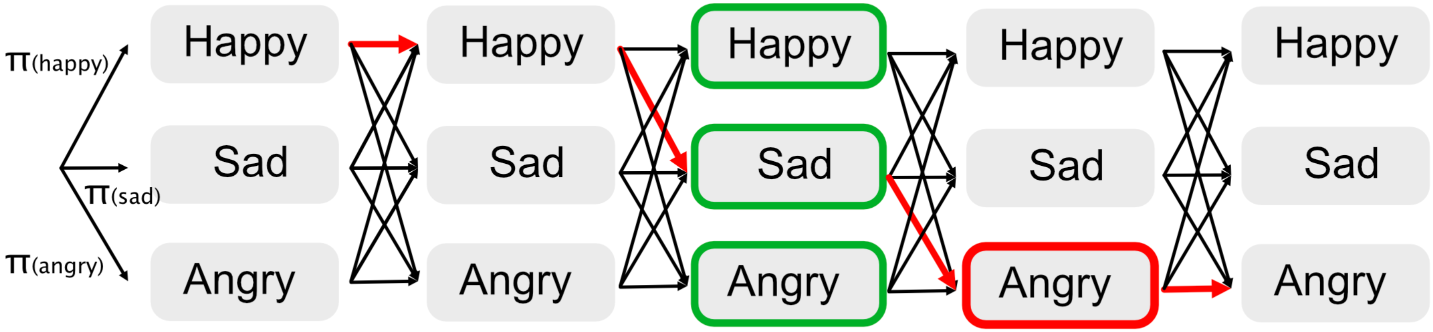
  - Termination: $p(S_t = k, \{O_t\}_{t=1}^T) = \alpha_t^k \beta_t^k$

$$p(S_t = k \mid \{O_t\}_{t=1}^T) = \frac{p(S_t = k, \{O_t\}_{t=1}^T)}{p(\{O_t\}_{t=1}^T)} = \frac{\alpha_t^k \beta_t^k}{\sum_i \alpha_t^i \beta_t^i}$$



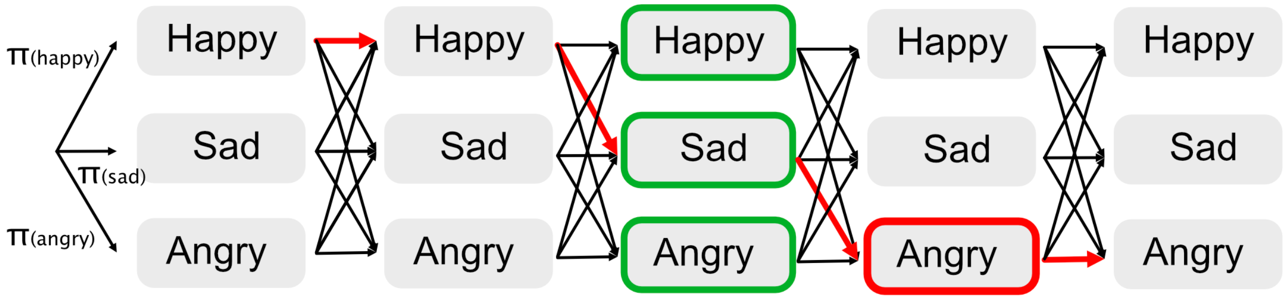
Viterbi algorithm

- If you observe "wagging, food, barking" in the first three days, what are the most likely emotion states of your dog these days?
- Translate to math: $$\arg\max_{\{S_t\}_{t=1}^T} p(\{S_t\}_{t=1}^T | \{O_t\}_{t=1}^T)$$
- Intuition: How do we find the "shortest" (optimal) path from beginning to the red node?



- Viterbi path

Viterbi algorithm



The optimal path from source to state k at t = The optimal path from source to state i at t-1 × Transition from state l to state k

- when we choose the best i

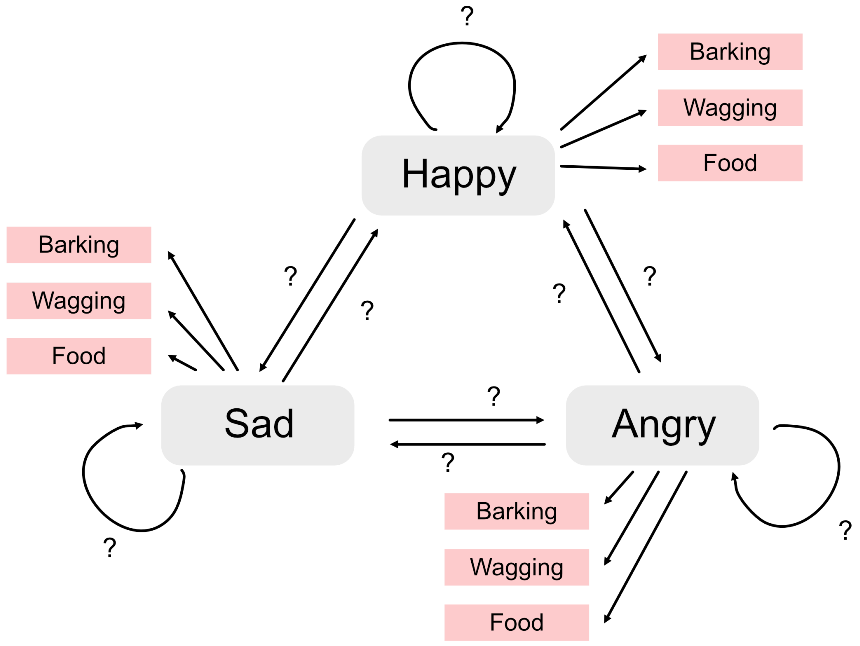
- Define V_t^k as the probability of the most probable Viterbi path of $O_1O_2\ldots O_t$ ends at state k
  then we have

$$V_1^k = p(O_1|S_1=k)p(S_1 = k)$$

$$V_t^k = p(O_t|S_t = k) \max_i p(S_t = k|S_{t-1} = i) V_{t-1}^i$$

- Eventually, traceback each step to find the path

Baum-Welch Algorithm



- We have a series of observation for our dog, what are the parameters of our HMM?
- Challenge: we have two unknowns
  - How does the HMM assign states
    E-step If we know the parameters, we can use forward-backward to assign states given observation
  - The parameters in this HMM
    M-step If we know the states, we can estimate the parameters by MLE (counting!)

$$p_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \qquad q_i^k = \frac{\sum_{t=1}^{T} \delta_{O_t=k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

- Expectation-Maximization (EM) algorithm!
  - Recursive, local optimization

Homework: Implementation of HMMs on real-world heart rate data

- Take heart rate data from one day of an individual and predict which state the person is in over the course of a day.
- Install the HMM package using pip install hmmlearn
- Set 3 states: model = hmmlearn.hmm.GaussianHMM(n_components = 3)
- (3) How do we know a dataset is normally distributed?
- (6) Fit HMM. Use the transformation you validated in (3)
  - model.fit(data)
  - states = model.predict(data)
- (7) Use HMM to predict on your moving window averaged data