Lecture 4: Probabilistic models 2

Helen Yannakoudakis and Oana Cocarascu

Department of Informatics
King's College London

(Version 1.2)

(Adapted from Simone Teufel)

Today

- Introduction
- Inductive Learning
- Probabilistic models 1
- Probabilistic models 2
- Kernel Methods
- Neural Networks
- Evolutionary Algorithms
- Reinforcement Learning 1
- Reinforcement Learning 2
- Learning from Demonstration

- So far we've looked at (statistical) classification and clustering.
- Experimented with underwear, tennis, and aircraft engines.
- Let us now talk about . . .

- So far we've looked at (statistical) classification and clustering.
- Experimented with underwear, tennis, and aircraft engines.
- Let us now talk about . . . the weather!

Weather prediction

- Two types of weather: rainy and cloudy
- The weather doesn't change within the day

Weather prediction

- Two types of weather: rainy and cloudy
- The weather doesn't change within the day
- Can we guess what the weather will be like tomorrow?

Weather prediction

- Two types of weather: rainy and cloudy
- The weather doesn't change within the day
- Can we guess what the weather will be like tomorrow?
- We can use a history of weather observations:

$$P(w_t = Rainy \mid w_{t-1} = Rainy, w_{t-2} = Cloudy, w_{t-3} = Cloudy, w_{t-4} = Rainy)$$

Weather prediction

- Two types of weather: rainy and cloudy
- The weather doesn't change within the day
- Can we guess what the weather will be like tomorrow?
- We can use a history of weather observations:

$$P(w_t = Rainy \mid w_{t-1} = Rainy, w_{t-2} = Cloudy, w_{t-3} = Cloudy, w_{t-4} = Rainy)$$

- Markov Assumption (first order):

$$P(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_1) \approx P(w_t \mid w_{t-1})$$

Weather prediction

- Two types of weather: rainy and cloudy
- The weather doesn't change within the day
- Can we guess what the weather will be like tomorrow?
- We can use a history of weather observations:

$$P(w_t = \mathit{Rainy} \mid w_{t-1} = \mathit{Rainy}, w_{t-2} = \mathit{Cloudy}, w_{t-3} = \mathit{Cloudy}, w_{t-4} = \mathit{Rainy})$$

- Markov Assumption (first order):

$$P(w_t \mid w_{t-1}, w_{t-2}, \ldots, w_1) \approx P(w_t \mid w_{t-1})$$

- The joint probability of a sequence of observations / events is then:

$$P(w_1, w_2, \ldots, w_t) = \prod_{t=1}^{n} P(w_t \mid w_{t-1})$$

Markov Chains

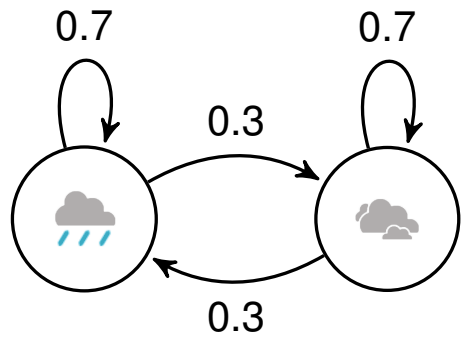
		Tomorrow	
		Rainy	*Cloudy*
Today	*Rainy*	0.7	0.3
	Cloudy	0.3	0.7

Transition probability matrix

Markov Chains

		Tomorrow: *Rainy*	Tomorrow: *Cloudy*
Today	*Rainy*	0.7	0.3
	Cloudy	0.3	0.7

Transition probability matrix

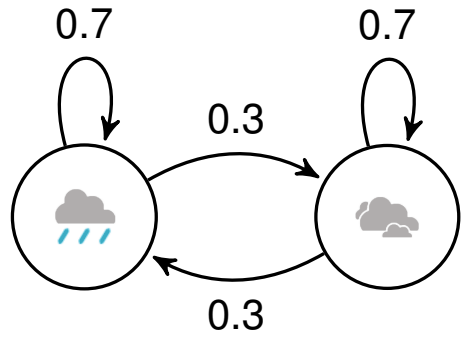


Two states: rainy and cloudy

Markov Chains

		Tomorrow	
		Rainy	*Cloudy*
Today	*Rainy*	0.7	0.3
	Cloudy	0.3	0.7

Transition probability matrix



Two states: rainy and cloudy

- A Markov Chain is a stochastic process that embodies the Markov Assumption.
- Can be viewed as a probabilistic finite-state automaton.
- States are fully observable, finite and discrete; transitions are labelled with transition probabilities.
- Models sequential problems – your current situation depends on what happened in the past

Markov Chains

- Useful for modeling the probability of a sequence of events
  - Valid phone sequences in speech recognition
  - Sequences of speech acts in dialog systems (eg, answering, ordering, opposing)
  - Predictive texting

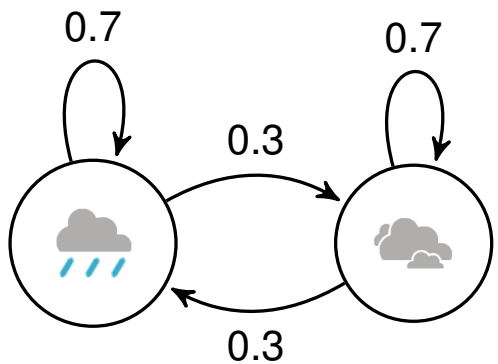
Markov Chains

- Useful for modeling the probability of a sequence of events *that can be unambiguously observed*
  - Valid phone sequences in speech recognition
  - Sequences of speech acts in dialog systems (eg, answering, ordering, opposing)
  - Predictive texting

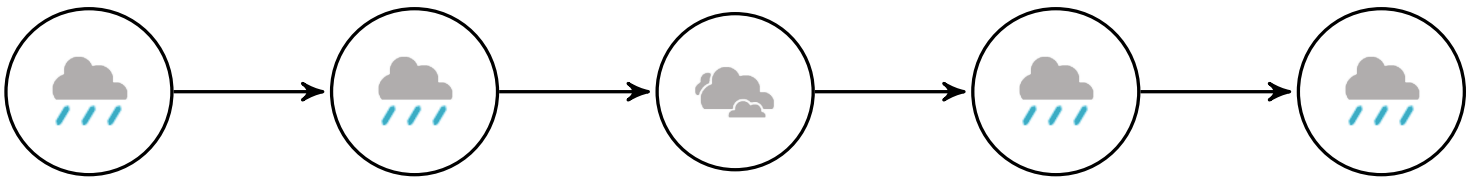
Markov Chains

- Useful for modeling the probability of a sequence of events *that can be unambiguously observed*
  - Valid phone sequences in speech recognition
  - Sequences of speech acts in dialog systems (eg, answering, ordering, opposing)
  - Predictive texting
- What if we are interested in events that are not unambiguously observed?

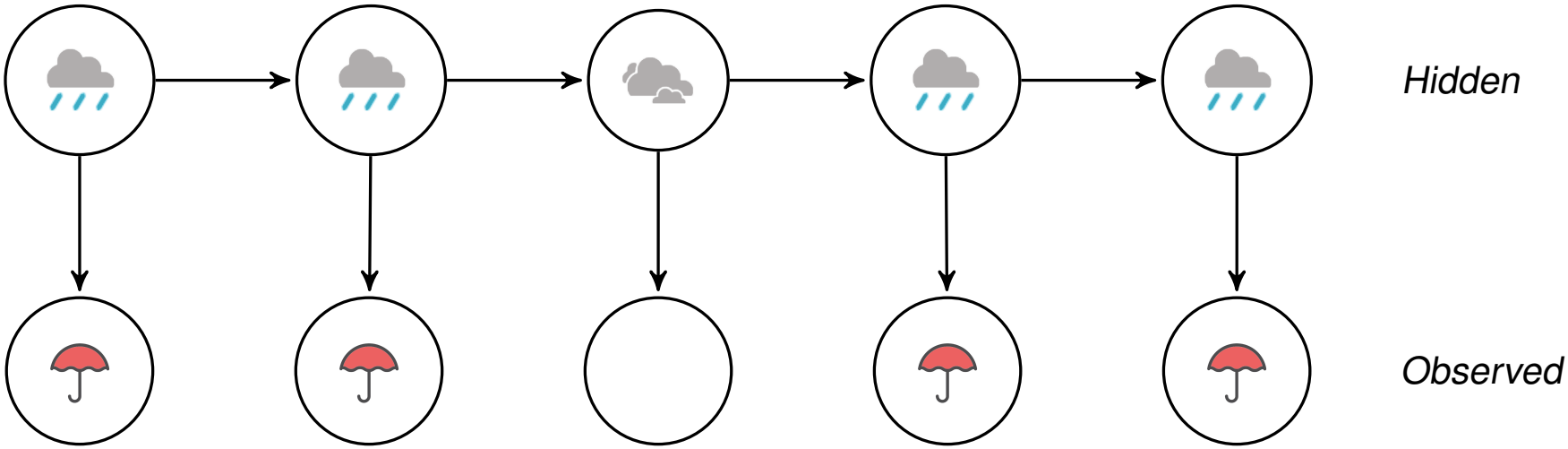
Markov Model



Markov Model: A Time-elapsed view

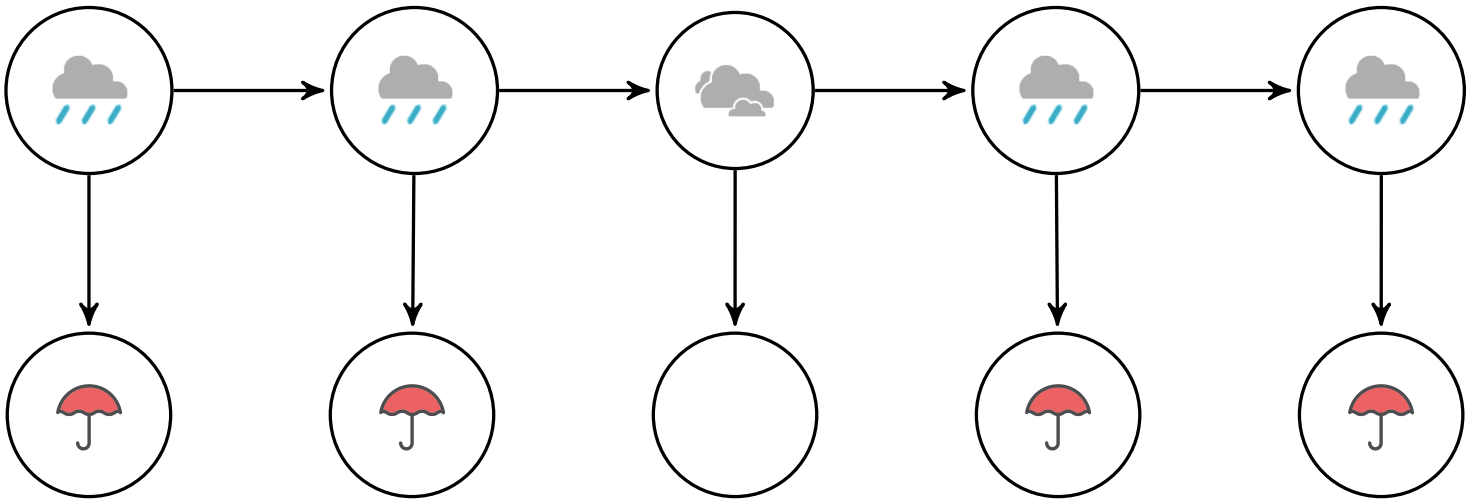


Hidden Markov Model: A Time-elapsed view



- Underlying Markov Chain over hidden states.
- We only have access to the observations at each time step.
- There is no 1:1 mapping between observations and hidden states.
- A number of hidden states can be associated with a particular observation, but the association of states and observations is governed by statistical behaviour.
- We now have to *infer* the sequence of hidden states that correspond to a sequence of observations.

Hidden Markov Model: A Time-elapsed view



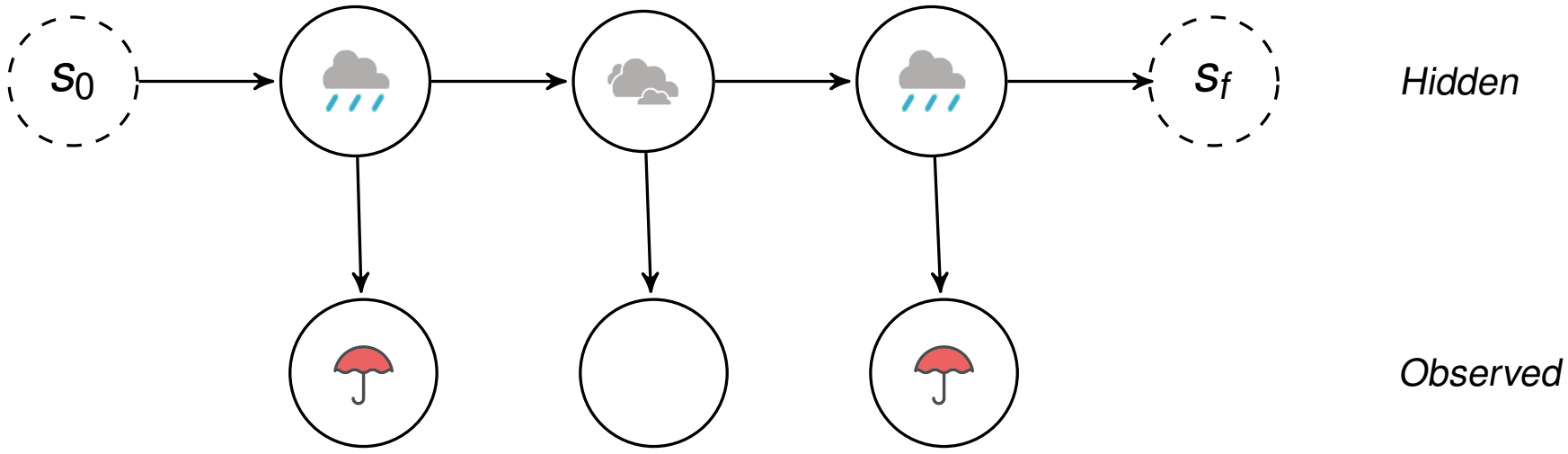
	Rainy	*Cloudy*
Rainy	0.7	0.3
Cloudy	0.3	0.7

Transition probabilities $P(w_t|w_{t-1})$

	Umbrella	*No umbrella*
Rainy	0.9	0.1
Cloudy	0.2	0.8

Emission probabilities $P(o_t|w_t)$
(Observation likelihoods)

Hidden Markov Model: A Time-elapsed view – start and end states



- Could use initial probability distribution over hidden states.
- Instead, for simplicity, we will also model this probability as a transition, and we will explicitly add a special start state.
- Similarly, we will add a special end state to explicitly model the end of the sequence.
- Special start and end states not associated with “real” observations.

More formal definition of Hidden Markov Models; States and Observations

$S_e = \{s_1, \ldots, s_N\}$	a set of N emitting hidden states,
s_0	a special start state,
s_f	a special end state.
$K = \{k_1, \ldots k_M\}$	an output alphabet of M observations ("vocabulary").
k_0	a special start symbol,
k_f	a special end symbol.
$O = O_1 \ldots O_T$	a sequence of T observations, each one drawn from K.
$X = X_1 \ldots X_T$	a sequence of T states, each one drawn from S_e.

More formal definition of Hidden Markov Models; First-order Hidden Markov Model

1. **Markov Assumption (Limited Horizon):** Transitions depend only on current state:

$$P(X_t|X_1...X_{t-1}) \approx P(X_t|X_{t-1})$$

2. **Output Independence:** Probability of an output observation depends only on the current state and not on any other states or any other observations:

$$P(O_t|X_1...X_t, ..., X_T, O_1, ..., O_t, ..., O_T) \approx P(O_t|X_t)$$

More formal definition of Hidden Markov Models; State Transition Probabilities

A: a state transition probability matrix of size $(N+2) \times (N+2)$.

$$A = \begin{bmatrix} - & a_{01} & a_{02} & a_{03} & . & . & . & a_{0N} & - \\ - & a_{11} & a_{12} & a_{13} & . & . & . & a_{1N} & a_{1f} \\ - & a_{21} & a_{22} & a_{23} & . & . & . & a_{2N} & a_{2f} \\ - & . & . & . & & & & . & . \\ - & . & . & . & & & & . & . \\ - & . & . & . & & & & . & . \\ - & a_{N1} & a_{N2} & a_{N3} & . & . & . & a_{NN} & a_{Nf} \\ - & - & - & - & - & - & - & - & - \end{bmatrix}$$

a_{ij} is the probability of moving from state s_i to state s_j:

$$a_{ij} = P(X_t = s_j | X_{t-1} = s_i)$$

$$\forall_i \sum_{j=0}^{N+1} a_{ij} = 1$$

More formal definition of Hidden Markov Models; State Transition Probabilities

A: a state transition probability matrix of size $(N+2) \times (N+2)$.

$$A = \begin{bmatrix} - & a_{01} & a_{02} & a_{03} & . & . & . & a_{0N} & - \\ - & a_{11} & a_{12} & a_{13} & . & . & . & a_{1N} & a_{1f} \\ - & a_{21} & a_{22} & a_{23} & . & . & . & a_{2N} & a_{2f} \\ - & . & . & . & & & & . & . \\ - & . & . & . & & & & . & . \\ - & . & . & . & & & & . & . \\ - & a_{N1} & a_{N2} & a_{N3} & . & . & . & a_{NN} & a_{Nf} \\ - & - & - & - & - & - & - & - & - \end{bmatrix}$$

a_{ij} is the probability of moving from state s_i to state s_j:

$$a_{ij} = P(X_t = s_j | X_{t-1} = s_i)$$

$$\forall_i \sum_{j=0}^{N+1} a_{ij} = 1$$

More formal definition of Hidden Markov Models; Start state s_0 and end state s_f

- Not associated with “real” observations.
- a_{0i} describe transition probabilities out of the start state into state s_i.
- a_{if} describe transition probabilities into the end state.
- Transitions into start state (a_{i0}) and out of end state (a_{fi}) undefined.

More formal definition of Hidden Markov Models; Emission Probabilities

B: an emission probability matrix of size $(M+2) \times (N+2)$.

$$B = \begin{bmatrix} b_0(k_0) & - & - & - & - & - & - & - & - \\ - & b_1(k_1) & b_2(k_1) & b_3(k_1) & . & . & . & b_N(k_1) & - \\ - & b_1(k_2) & b_2(k_2) & b_3(k_2) & . & . & . & b_N(k_2) & - \\ - & . & . & . & & & & . & - \\ - & . & . & . & & & & . & - \\ - & . & . & . & & & & . & - \\ - & b_1(k_M) & b_2(k_M) & b_3(k_M) & . & . & . & b_N(k_M) & - \\ - & - & - & - & - & - & - & - & b_f(k_f) \end{bmatrix}$$

$b_i(k_j)$ is the probability of emitting vocabulary item k_j from state s_i:

$$b_i(k_j) = P(O_t = k_j | X_t = s_i)$$

Our HMM is defined by its parameters $\mu = (A, B)$.

More formal definition of Hidden Markov Models; Emission Probabilities

B: an emission probability matrix of size $(M+2) \times (N+2)$.

$$B = \begin{bmatrix} b_0(k_0) & - & - & - & - & - & - & - & - \\ - & b_1(k_1) & b_2(k_1) & b_3(k_1) & . & . & . & b_N(k_1) & - \\ - & b_1(k_2) & b_2(k_2) & b_3(k_2) & . & . & . & b_N(k_2) & - \\ - & . & . & . & & & & . & - \\ - & . & . & . & & & & . & - \\ - & . & . & . & & & & . & - \\ - & b_1(k_M) & b_2(k_M) & b_3(k_M) & . & . & . & b_N(k_M) & - \\ - & - & - & - & - & - & - & - & b_f(k_f) \end{bmatrix}$$

$b_i(k_j)$ is the probability of emitting vocabulary item k_j from state s_i:

$$b_i(k_j) = P(O_t = k_j | X_t = s_i)$$

Our HMM is defined by its parameters $\mu = (A, B)$.

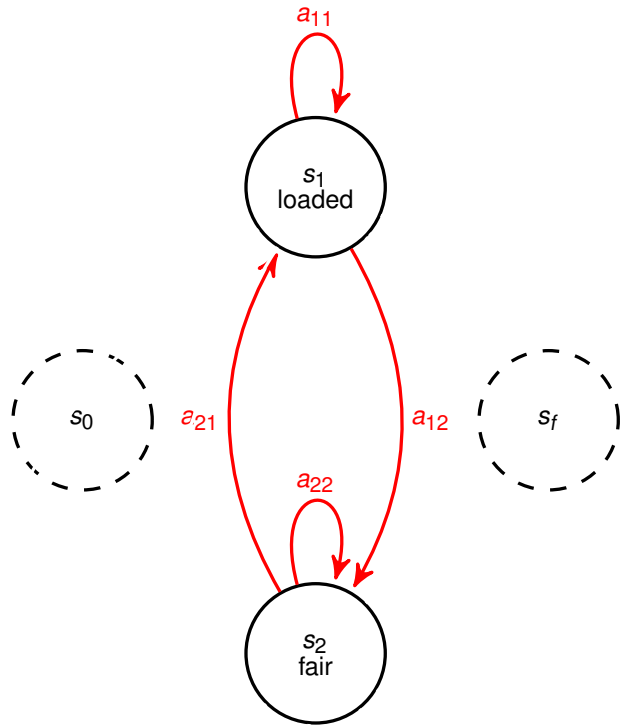
Examples where states are hidden

- Speech recognition
  - Observations: audio signal
  - States: phonemes
- Part-of-speech tagging (assigning tags like Noun and Verb to words)
  - Observations: words
  - States: part-of-speech tags
- Machine translation
  - Observations: target words
  - States: source words

The dice HMM

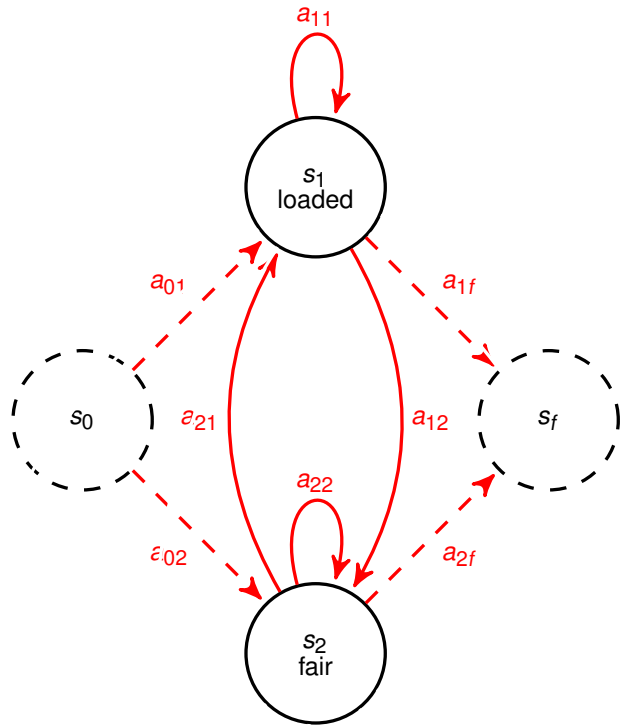
- Imagine a fraudulous croupier in a casino where customers bet on dice outcomes.
- She has two dice – a fair one and a loaded one.
- The fair one has the normal distribution of outcomes – $P(O) = \frac{1}{6}$ for each number 1 to 6.
- The loaded one has a different distribution.
- She secretly switches between the two dice.
- You don't know which dice is currently in use. You can only observe the numbers that are thrown.

The dice HMM



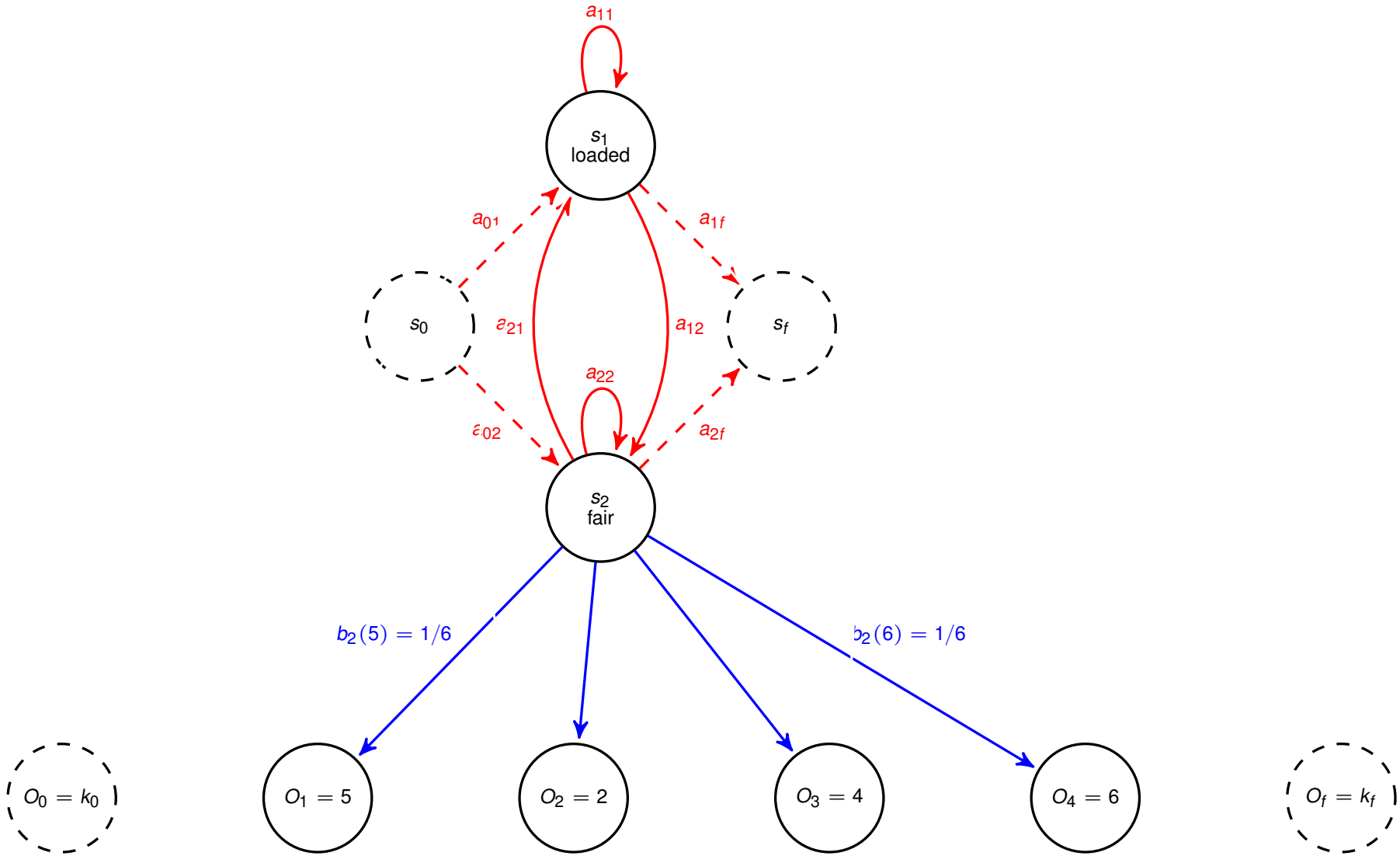
- There are two states (fair and loaded), and two special states (start s_0 and end s_f).
- Distribution of observations differs between the states.

The dice HMM



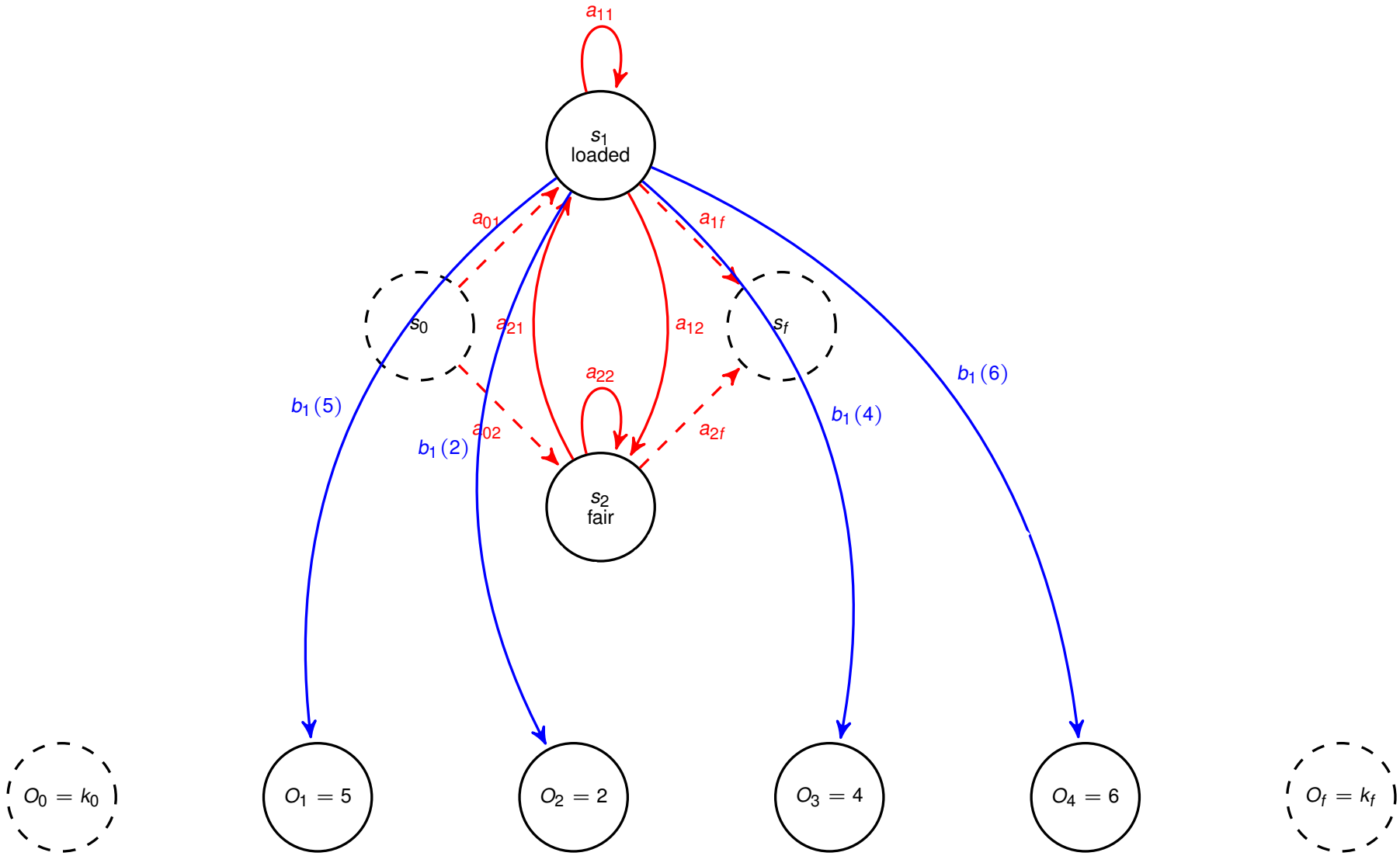
- There are two states (fair and loaded), and two special states (start s_0 and end s_f).
- Distribution of observations differs between the states.

The dice HMM



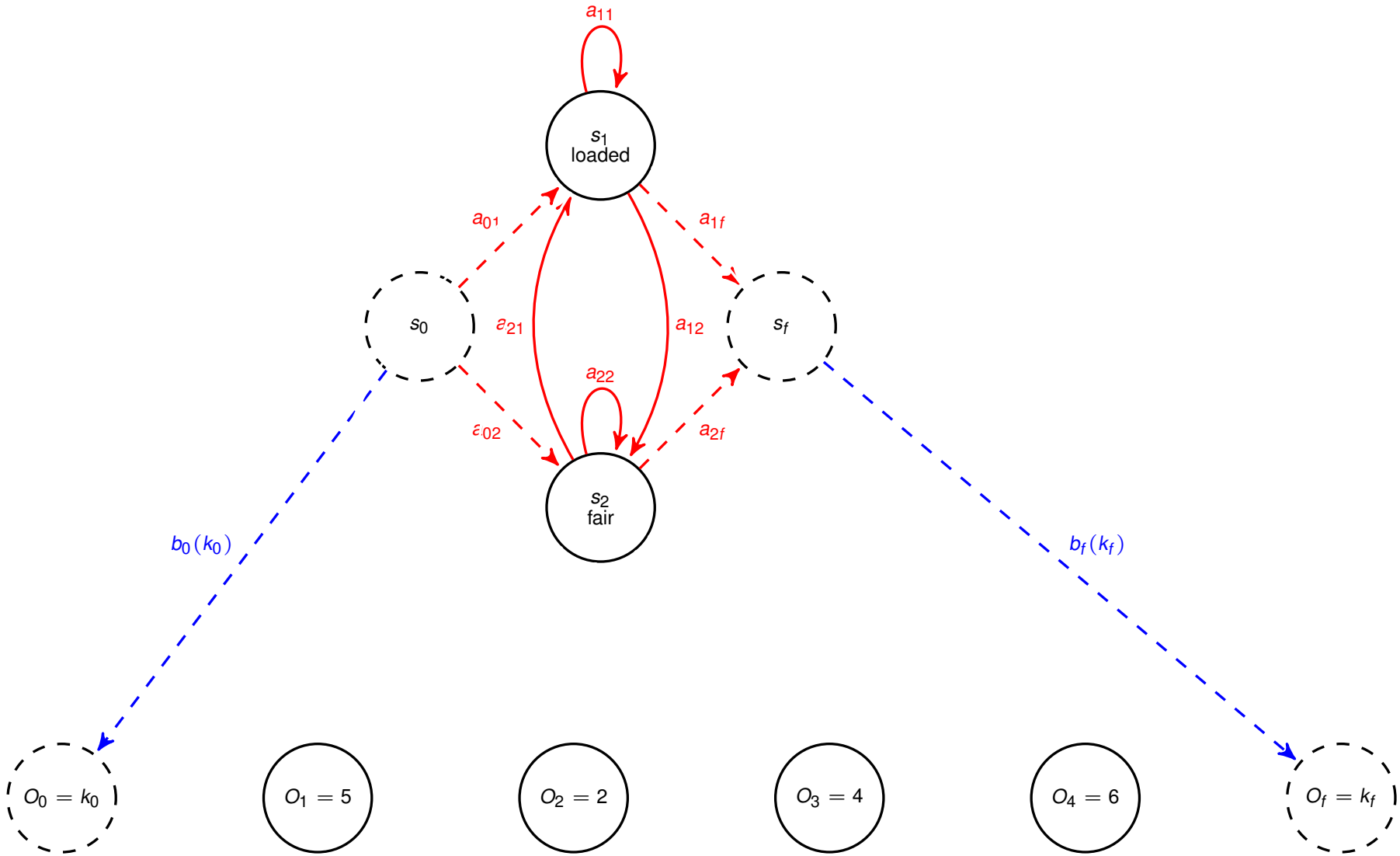
- There are two states (fair and loaded), and two special states (start s_0 and end s_f).
- Distribution of observations differs between the states.

The dice HMM



- There are two states (fair and loaded), and two special states (start s_0 and end s_f).
- Distribution of observations differs between the states.

The dice HMM



- There are two states (fair and loaded), and two special states (start s_0 and end s_f).
- Distribution of observations differs between the states.

The dice HMM

- Labelled HMM learning, i.e. it has
  - Input: dual tape of state and observation (dice outcome) sequences X and O.

(s_0)	F	F	F	F	L	L	L	F	F	F	F	L	L	L	L	F	F	F	L	L	(s_f)
(k_0)	1	3	4	5	6	6	5	1	2	3	1	4	3	5	4	1	2	6	1	2	(k_f)

  - Output: HMM parameters A, B.

Parameter estimation of HMM parameters A, B

- Transition matrix A consists of transition probabilities a_{ij}

$$a_{ij} = P(X_{t+1} = s_j | X_t = s_i) \sim \frac{count_{trans}(X_t = s_i, X_{t+1} = s_j)}{count_{trans}(X_t = s_i)}$$

- Emission matrix B consists of emission probabilities $b_i(k_j)$

$$b_i(k_j) = P(O_t = k_j | X_t = s_i) \sim \frac{count_{emission}(O_t = k_j, X_t = s_i)}{count_{emission}(X_t = s_i)}$$

- (Add-one smoothed versions of these)

The dice HMM

(s_0)	F	F	F	F	L	L	L	F	F	F	F	L	L	L	L	F	F	F	L	L	(s_f)
(k_0)	1	3	4	5	6	6	5	1	2	3	1	4	3	5	4	1	2	6	1	2	(k_f)

- Transition matrix A consists of transition probabilities a_{ij}

$$a_{ij} = P(X_{t+1} = s_j | X_t = s_i) \sim \frac{count_{trans}(X_t = s_i, X_{t+1} = s_j)}{count_{trans}(X_t = s_i)}$$

- Emission matrix B consists of emission probabilities $b_i(k_j)$

$$b_i(k_j) = P(O_t = k_j | X_t = s_i) \sim \frac{count_{emission}(O_t = k_j, X_t = s_i)}{count_{emission}(X_t = s_i)}$$

Emission probabilities	k0	1	2	3	4	5	6	kf
F	0.00	0.36	0.18	0.18	0.09	0.09	0.09	0.00
L	0.00	0.11	0.11	0.11	0.22	0.22	0.22	0.00
s0	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
sf	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00

Transition probabilities	to F	to L	to s0	to sf
From s0	1.00	0.00	0.00	0.00
From F	0.73	0.27	0.00	0.00
From L	0.22	0.67	0.00	0.11
From sf	0.00	0.00	0.00	0.00

Fundamental tasks with HMMs

- Problem 1 (Labelled Learning)
  - Given a parallel observation and state sequence O and X, learn the HMM parameters A and B. → looked at this
- Problem 2 (Unlabelled Learning)
  - Given an observation sequence O (and only the set of emitting states S_e), learn the HMM parameters A and B.
- Problem 3 (Likelihood)
  - Given an HMM $\mu = (A, B)$ and an observation sequence O, determine the likelihood $P(O|\mu)$.
- Problem 4 (Decoding)
  - Given an observation sequence O and an HMM $\mu = (A, B)$, discover the best hidden state sequence X. → now to this

Literature

- Manning and Schutze (2000). Foundations of Statistical Natural Language Processing, MIT Press. Chapters 9.1, 9.2.
  - We use state-emission HMM instead of arc-emission HMM
  - We avoid initial state probability vector π by using explicit start and end states (s_0 and s_f) and incorporating the corresponding probabilities into the transition matrix A.
- (Jurafsky and Martin, 2nd Edition, Chapter 6.2 (but careful, notation!))
- Fosler-Lussier, Eric (1998). Markov Models and Hidden Markov Models: A Brief Tutorial. TR-98-041.
- Smith, Noah A. (2004). Hidden Markov Models: All the Glorious Gory Details.
- Bockmayr and Reinert (2011). Markov chains and Hidden Markov Models. Discrete Math for Bioinformatics WS 10/11.

So far: estimating parameters of an HMM

- The dishonest casino, dice edition.
- Two hidden states: L (loaded dice), F (fair dice).
- You don't know which dice is currently in use. You can only observe the numbers that are thrown.
- You estimate transition and emission probabilities (Problem 1).

- We are now turning to Problem 4: Decoding.
- We want the HMM to find out when the fair dice was out, and when the loaded dice was out.
- We need a decoder.

Decoding: finding the most likely path

- Definition of decoding: Finding the most likely hidden state sequence X that explains the observation O given the HMM parameters μ.

$$\begin{aligned}\hat{X} &= \operatorname*{argmax}_X P(X, O|\mu) \\ &= \operatorname*{argmax}_X P(O|X,\mu)P(X|\mu) \\ &= \operatorname*{argmax}_{X_1 \ldots X_T} \prod_{t=1}^{T} P(O_t|X_t)P(X_t|X_{t-1})\end{aligned}$$

- Search space of possible state sequences X is O(N^T); too large for brute force search.

Viterbi is a Dynamic Programming Application

(From Algorithms)
We can use Dynamic Programming if two conditions apply:

- Optimal substructure property
  - An optimal state sequence $X_1 \ldots X_j \ldots X_T$ contains inside it the sequence $X_1 \ldots X_j$, which is also optimal
- Overlapping subsolutions property
  - If both X_t and X_u are on the optimal path, with $u > t$, then the calculation of the probability for being in state X_t is part of each of the many calculations for being in state X_u.

The intuition behind Viterbi

- Here's how we can save ourselves a lot of time.
- Because of the Limited Horizon of the HMM, we don't need to keep a complete record of how we arrived at a certain state.
- For the first-order HMM, we only need to record one previous step.
- Just do the calculation of the probability of reaching each state once for each time step.
- Then memoise this probability in a Dynamic Programming table
- This reduces our effort to $O(N^2 T)$.
- This is for the first order HMM, which only has a memory of one previous state.

Viterbi: main data structure

- Memoisation is done using a *trellis*.
- A trellis is equivalent to a Dynamic Programming table.
- The trellis is $(N+2) \times (T+2)$ in size, with states j as rows and time steps t as columns. (T: length of observation / time steps)
- Each cell j, t records the Viterbi probability $\delta_j(t)$, the probability of the most likely path that ends in state s_j at time t:

$$\delta_j(t) = \max_{1 \leqslant i \leqslant N} [\delta_i(t-1)\, a_{ij}\, b_j(O_t)]$$

- This probability is calculated by maximising over the best ways of going to s_j for each s_i.
- a_{ij}: the transition probability from s_i to s_j
- $b_j(O_t)$: the probability of emitting observation O_t from destination state s_j

Viterbi algorithm, initialisation

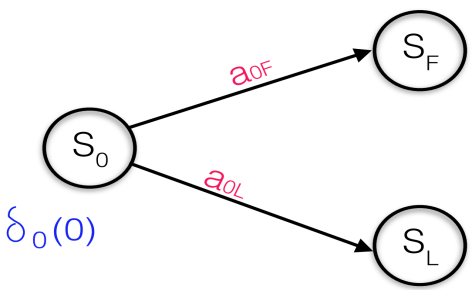
Note: the probability of a state starting the sequence at $t = 0$ is just the probability of it emitting the first symbol.

Viterbi algorithm, initialisation

Viterbi algorithm, initialisation

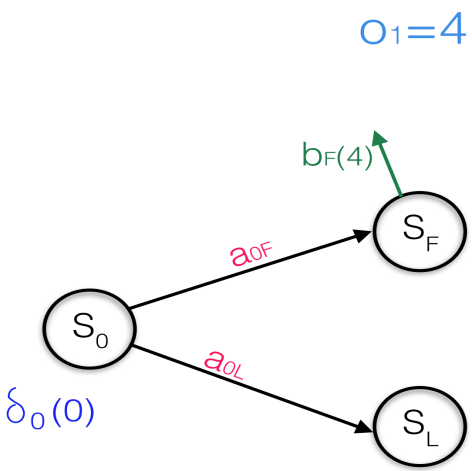
Viterbi algorithm, initialisation

Viterbi algorithm, main step



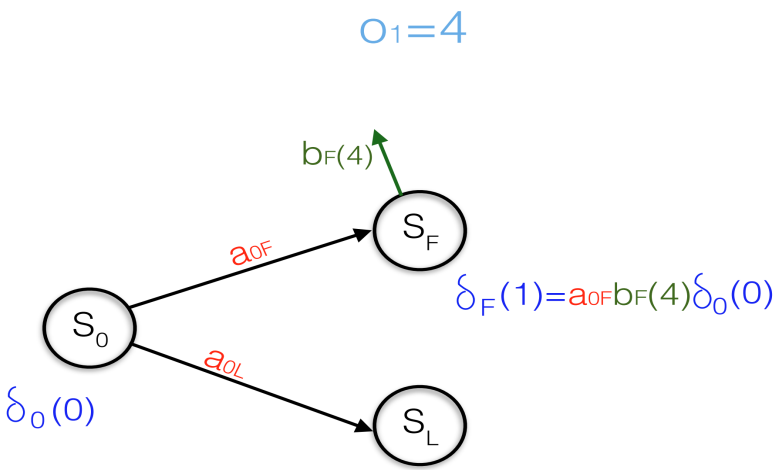
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



$X_1 = F$ $X = L$ $X = L$

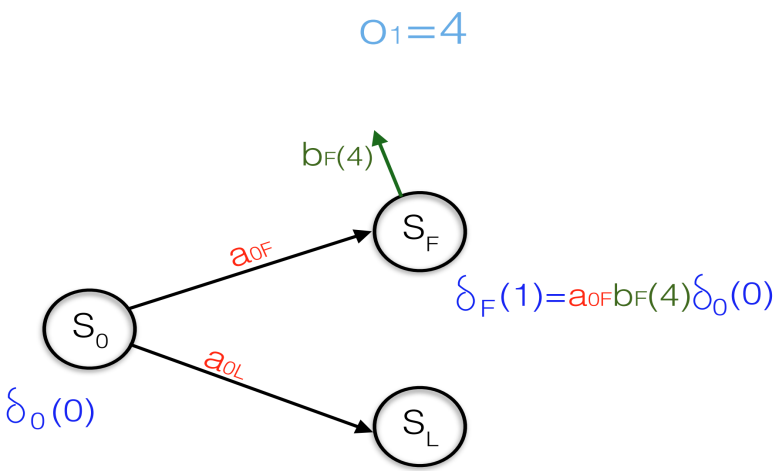
Viterbi algorithm, main step, ψ

- $\psi_j(t)$ is a helper variable that stores the $t-1$ state index i on the highest probability path.

$$\psi_j(t) = \underset{1 \leqslant i \leqslant N}{\operatorname{argmax}} [\delta_i(t-1)\, a_{ij}\, b_j(O_t)]$$

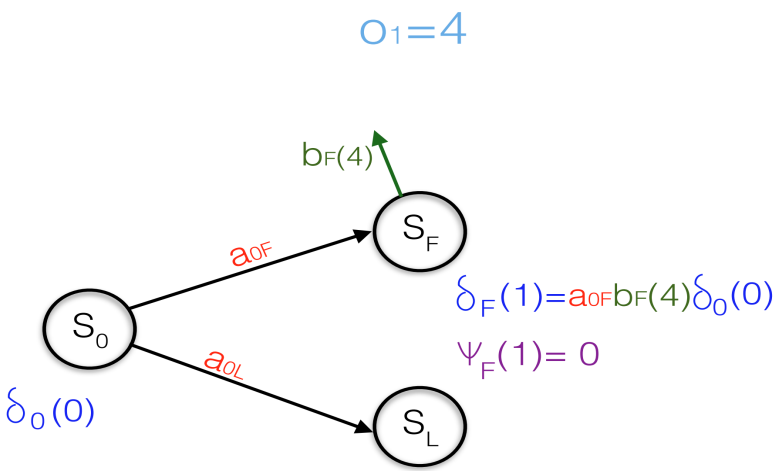
- In the backtracing phase, we will use ψ to find the previous cell/state in the best path.

Viterbi algorithm, main step: observation is 4



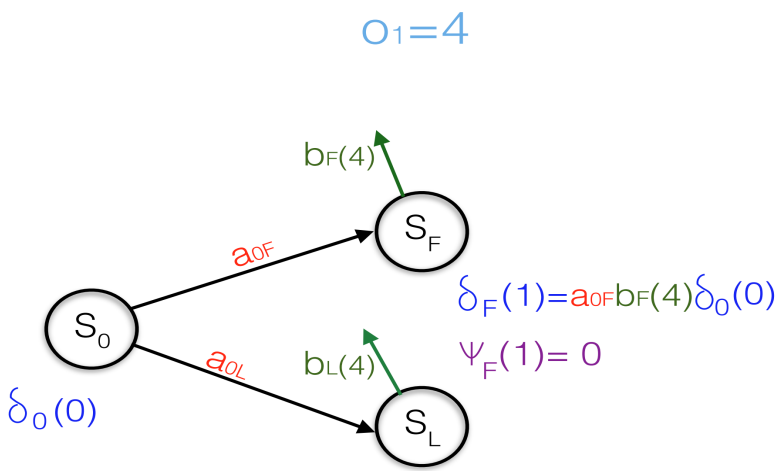
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4



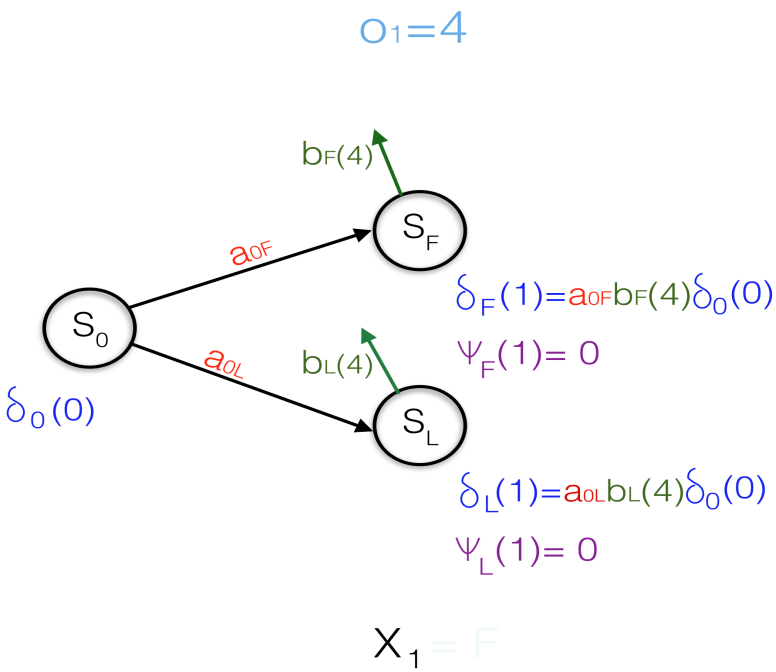
$X_1 = F$ $X = L$ $X = L$

Viterbi algorithm, main step: observation is 4

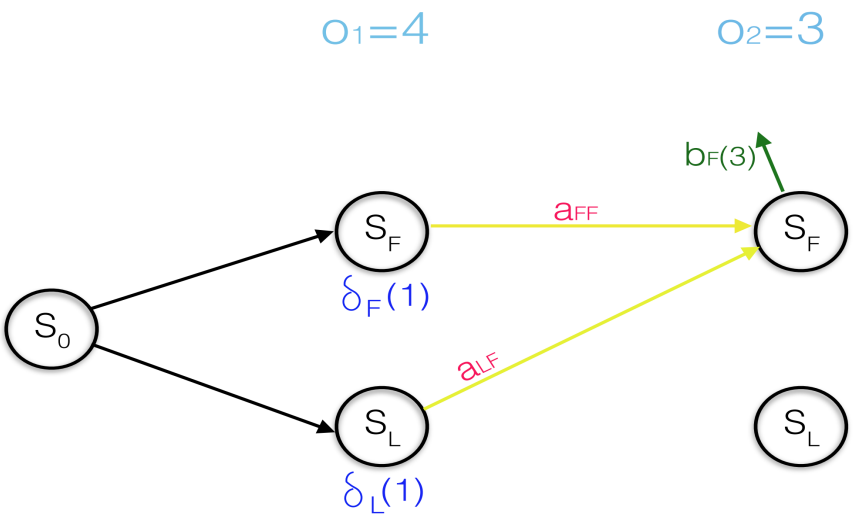


$X_1 = F$ $X = L$ $X = L$

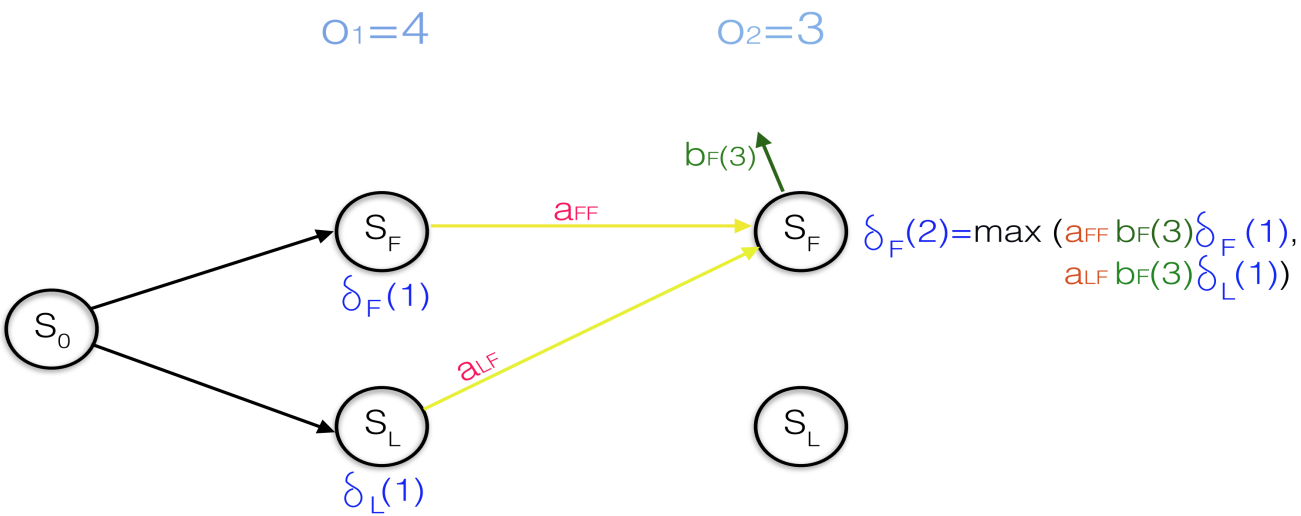
Viterbi algorithm, main step: observation is 4



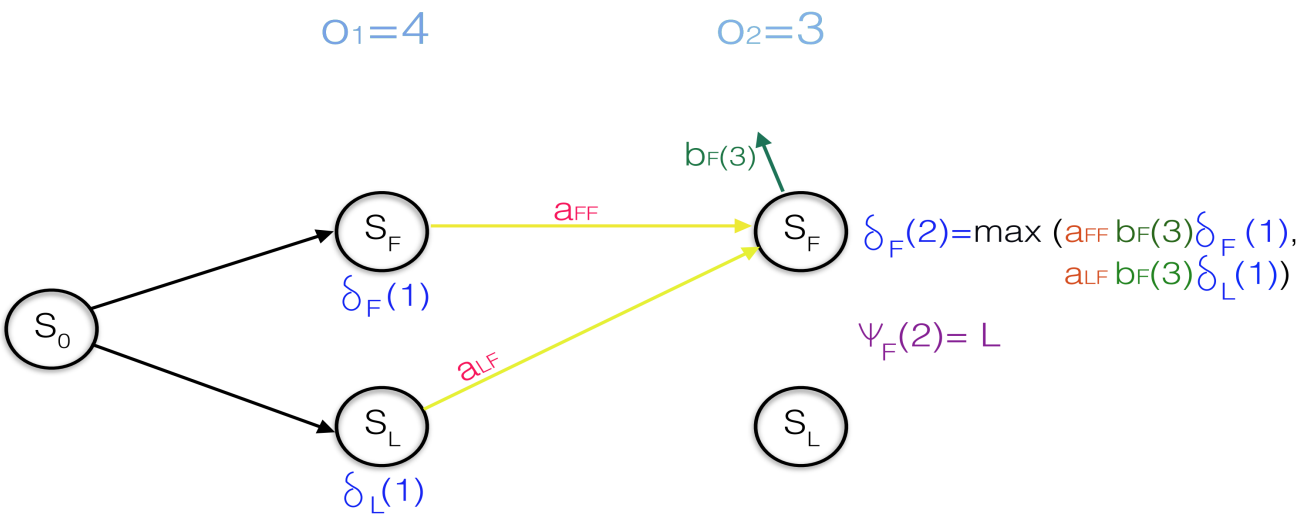
Viterbi algorithm, main step: observation is 3



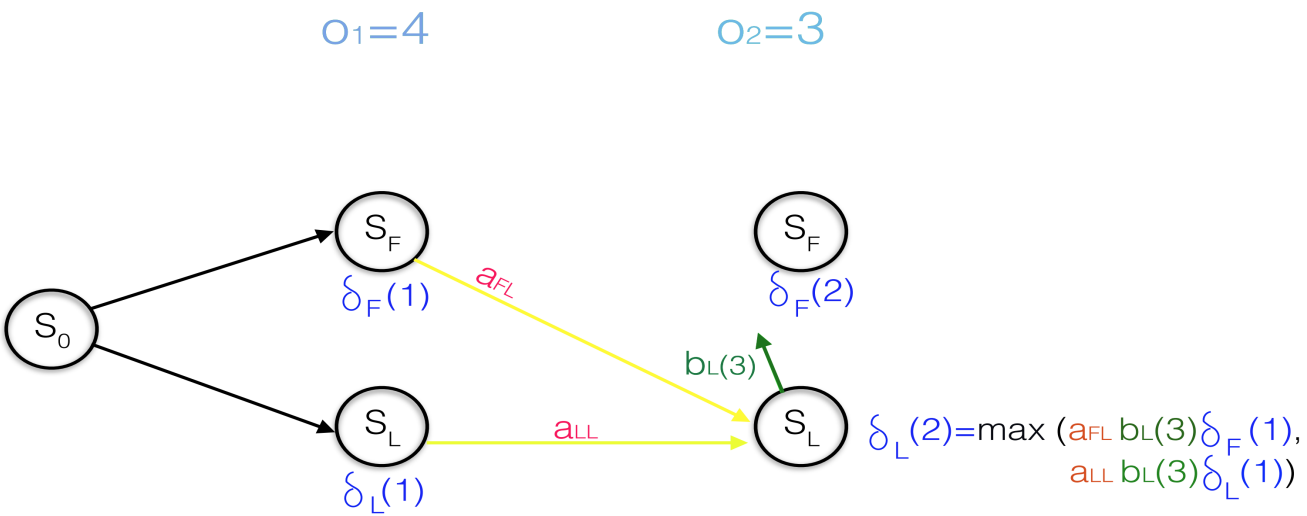
Viterbi algorithm, main step: observation is 3



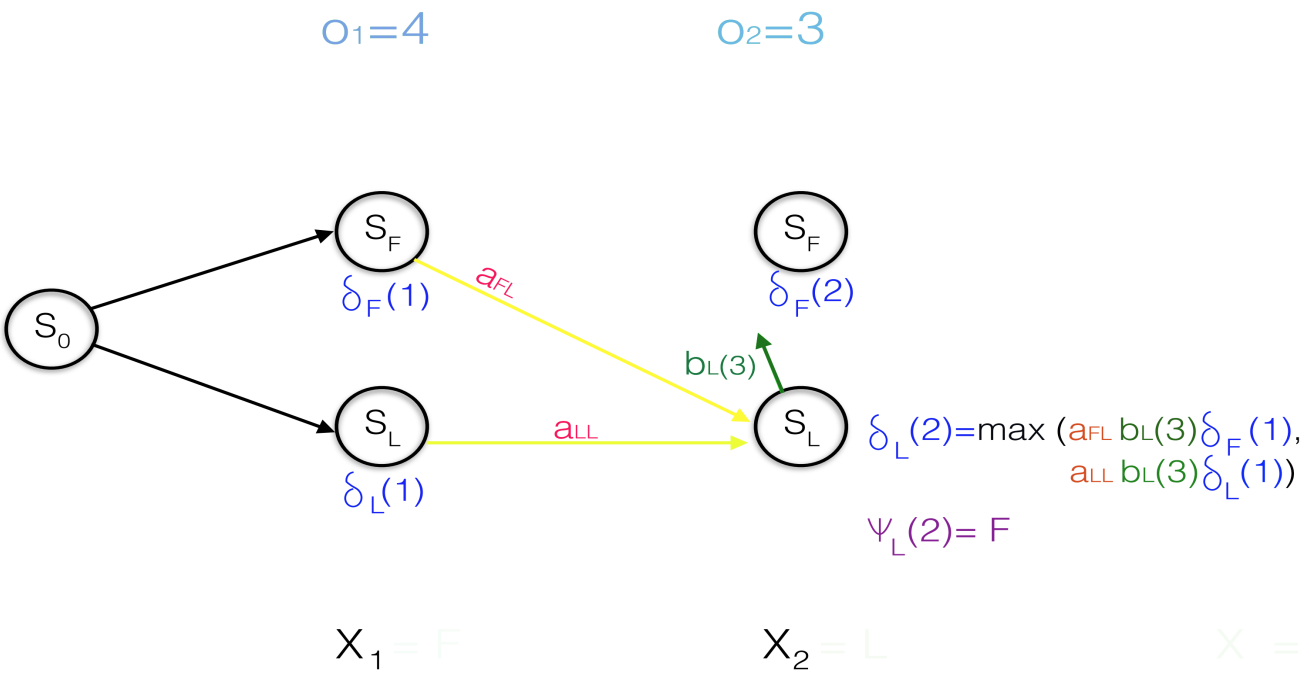
Viterbi algorithm, main step: observation is 3



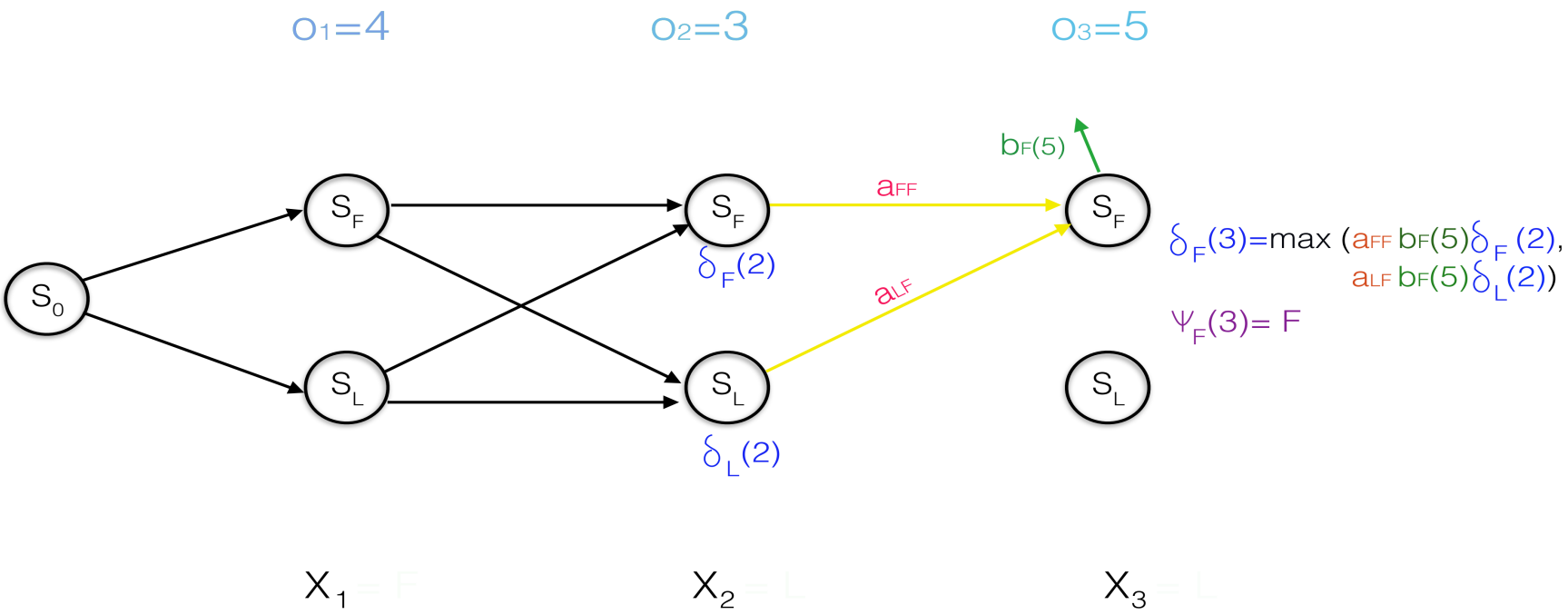
Viterbi algorithm, main step: observation is 3



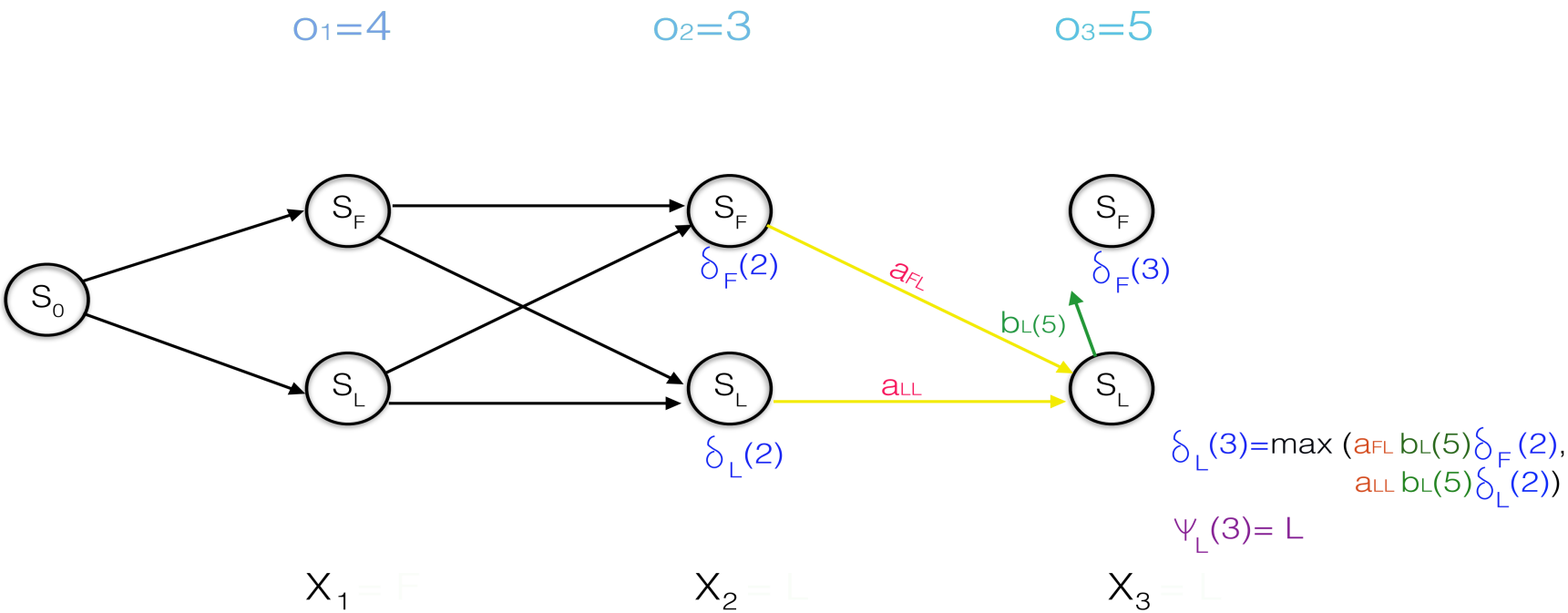
Viterbi algorithm, main step: observation is 3



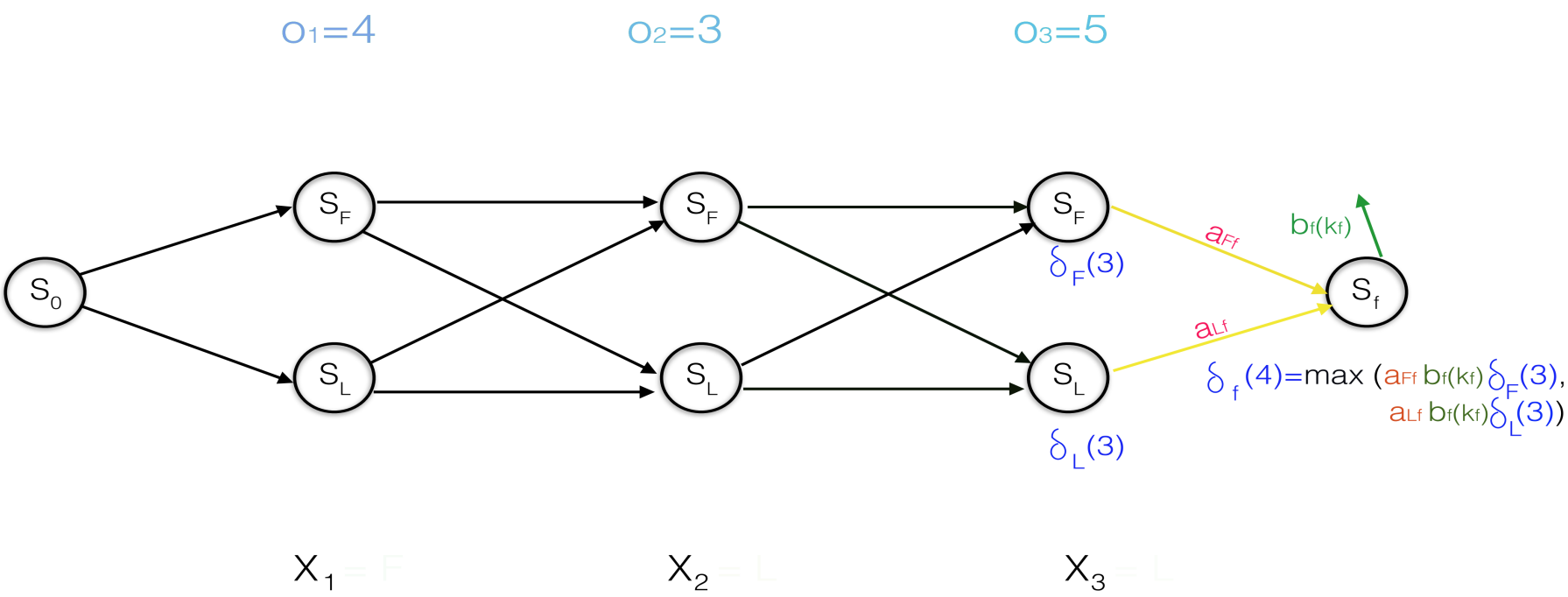
Viterbi algorithm, main step: observation is 5



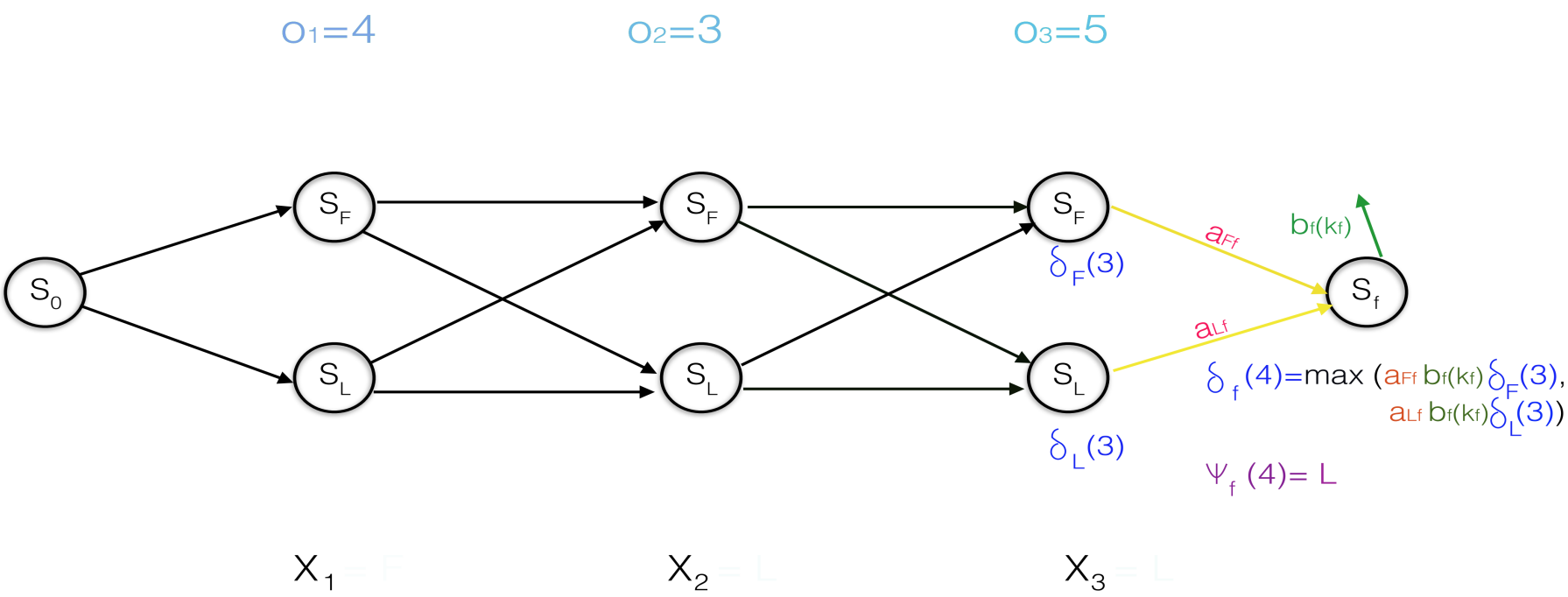
Viterbi algorithm, main step: observation is 5



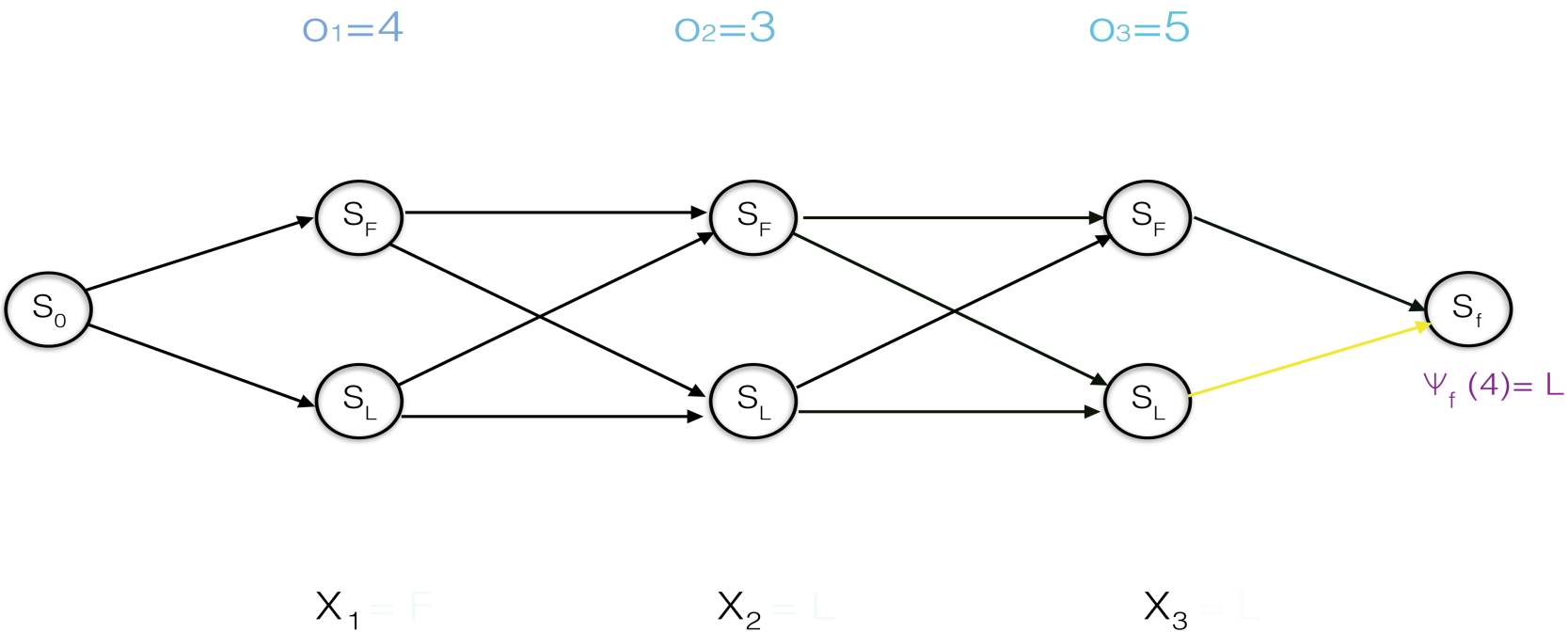
Viterbi algorithm, termination



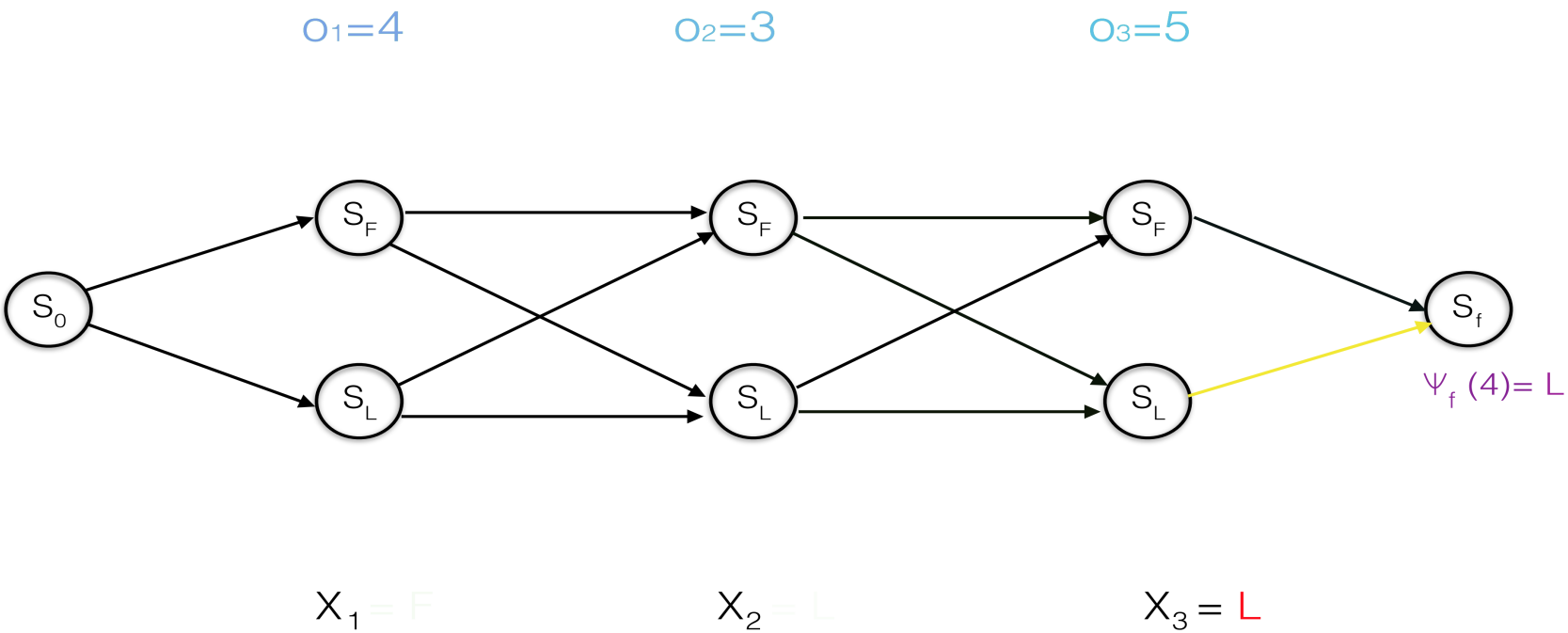
Viterbi algorithm, termination



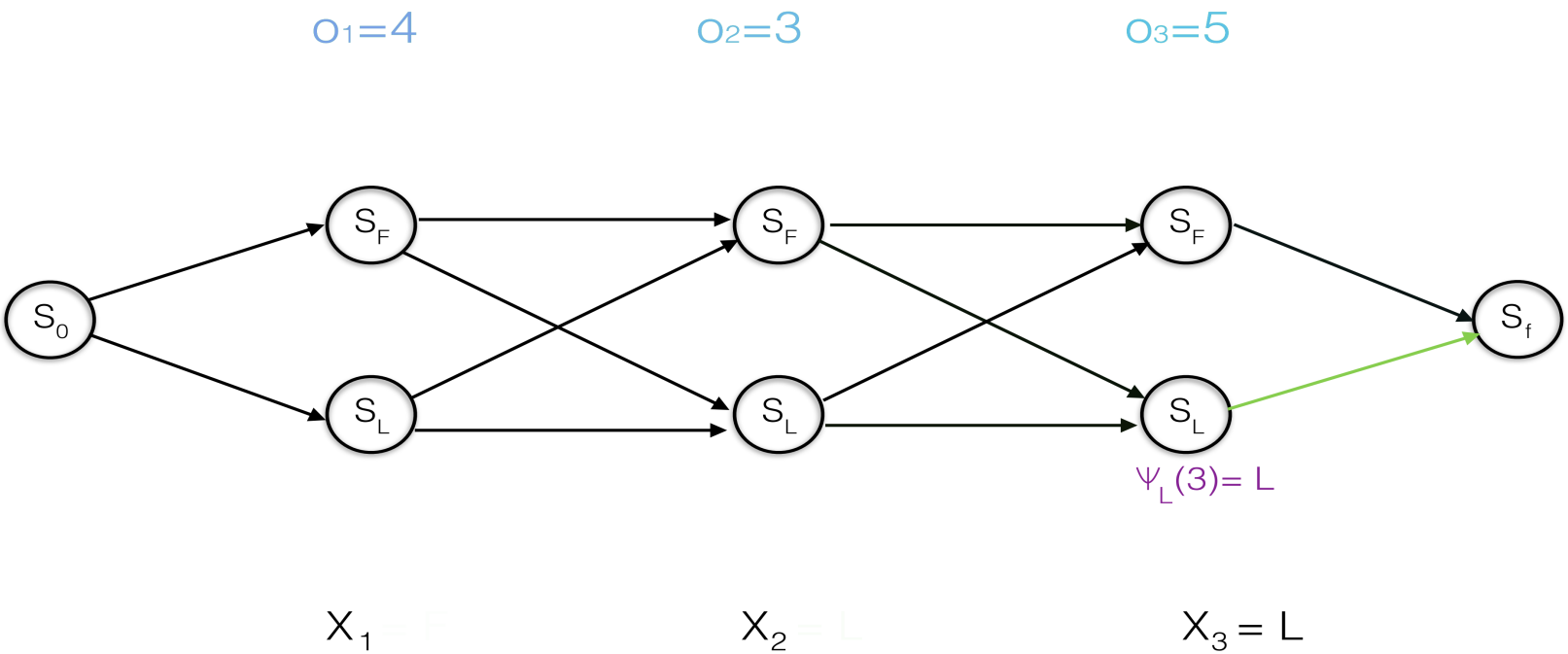
Viterbi algorithm, backtracing



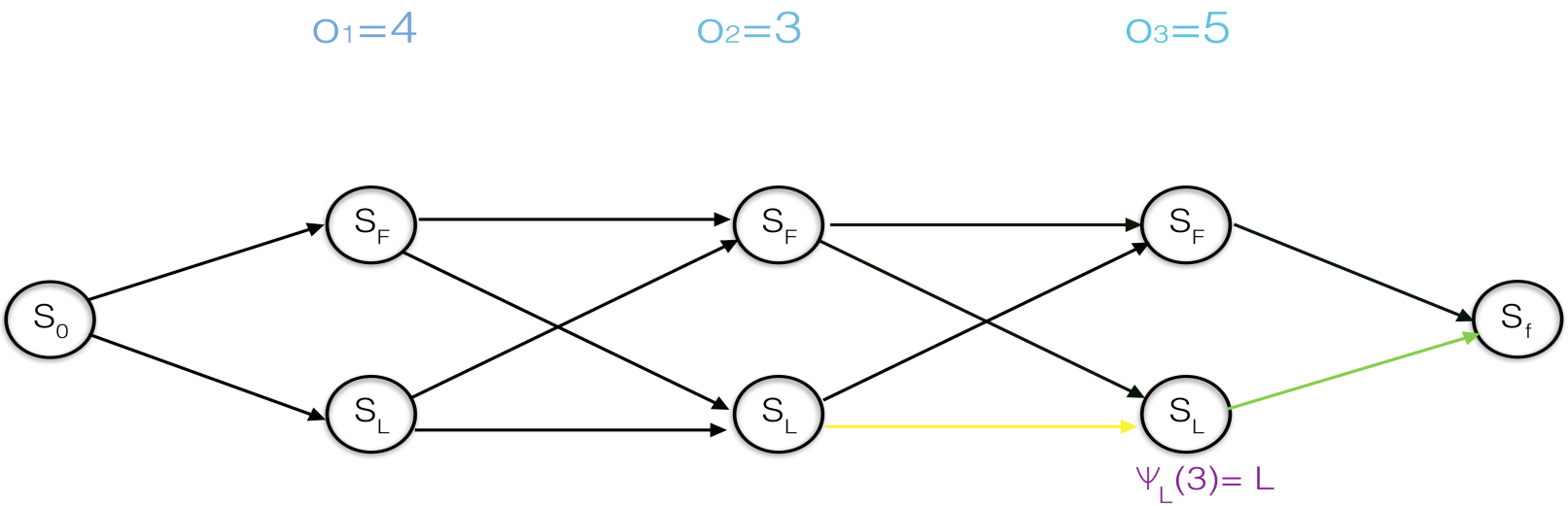
Viterbi algorithm, backtracing



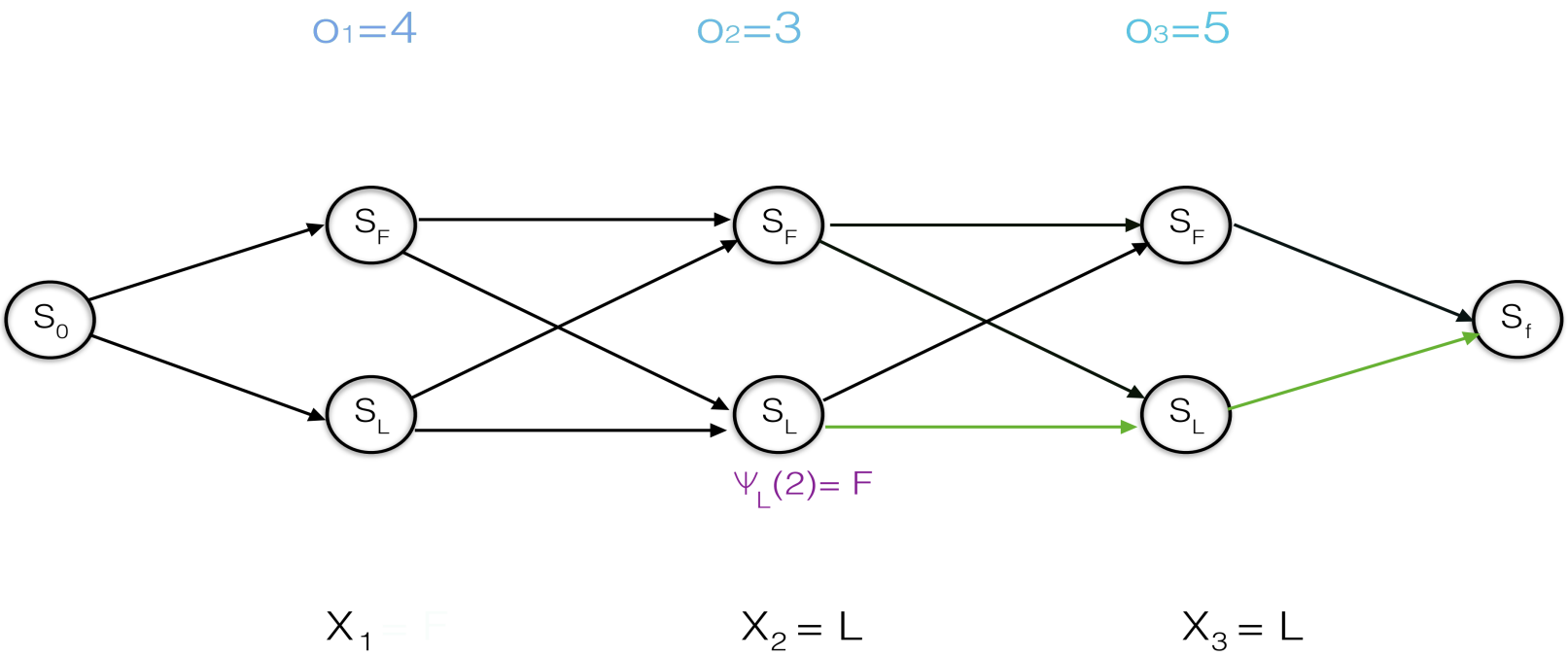
Viterbi algorithm, backtracing



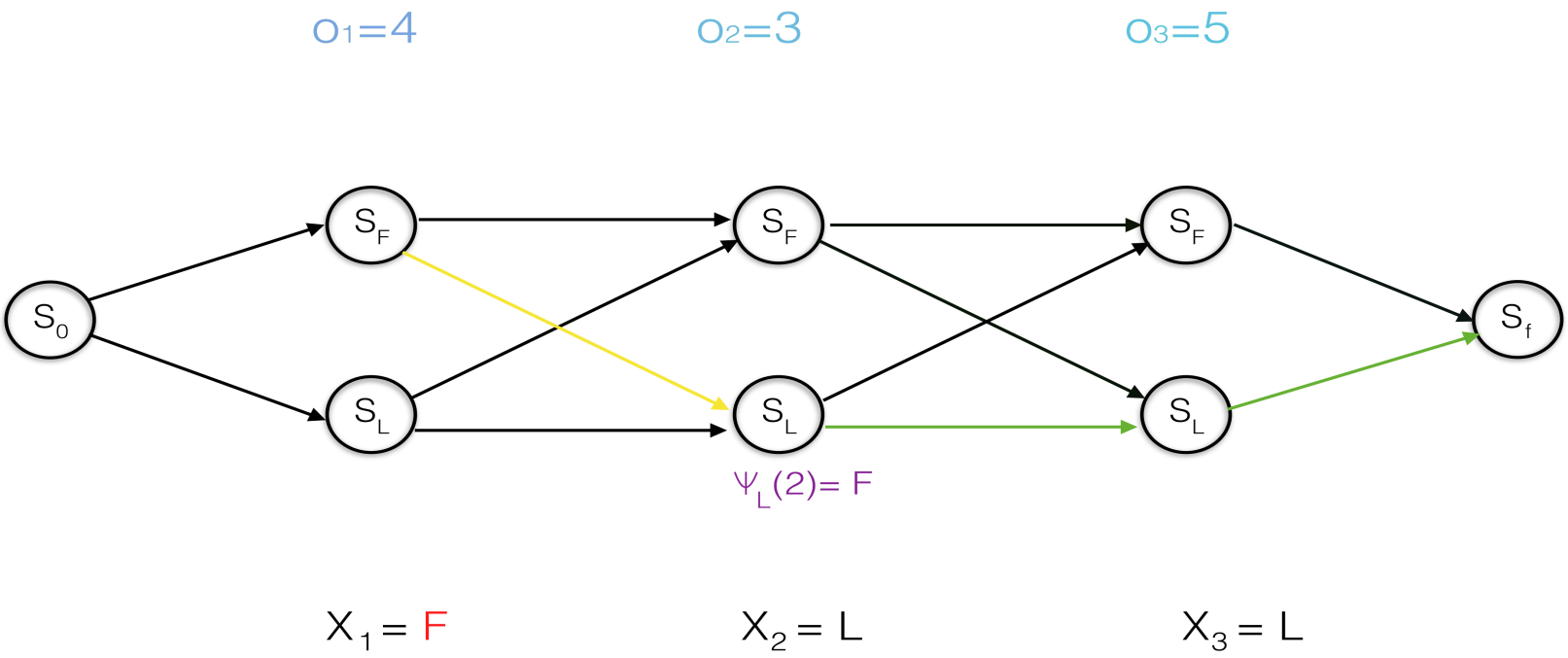
Viterbi algorithm, backtracing



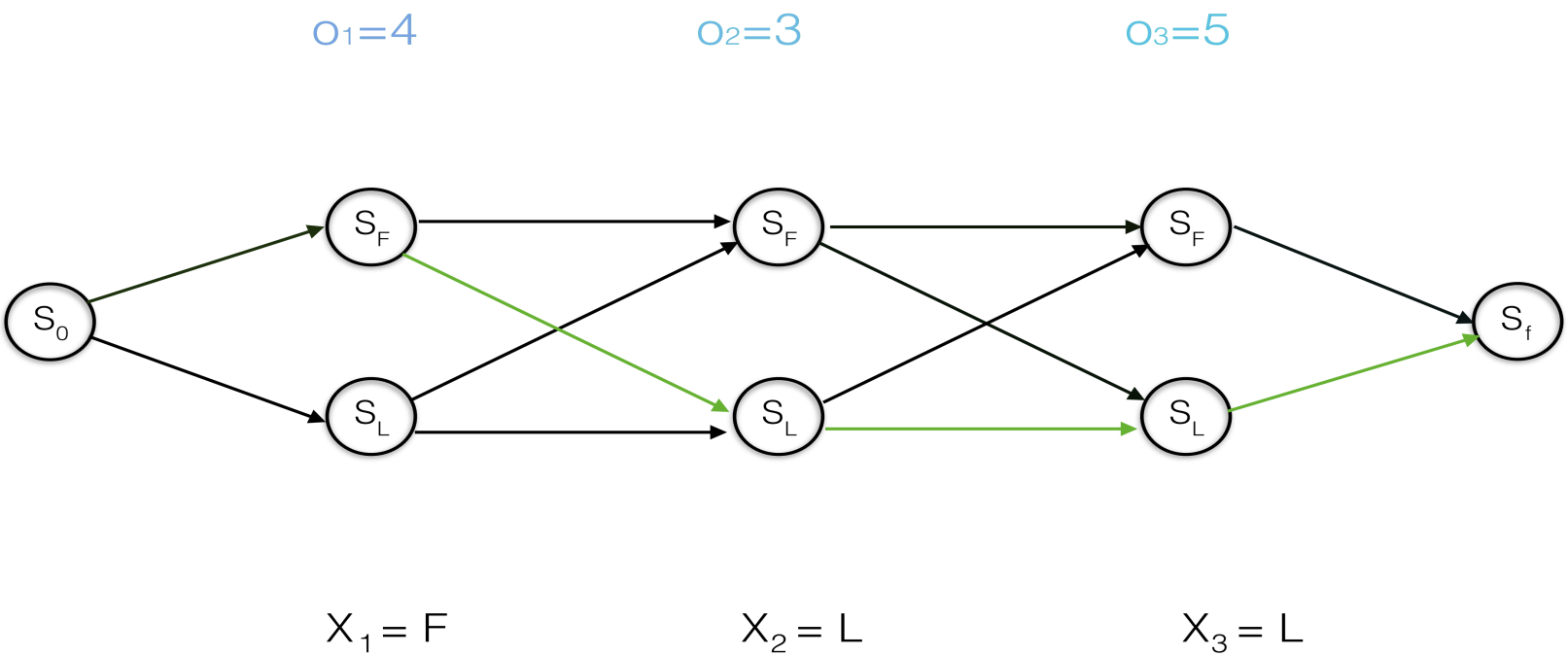
Viterbi algorithm, backtracing



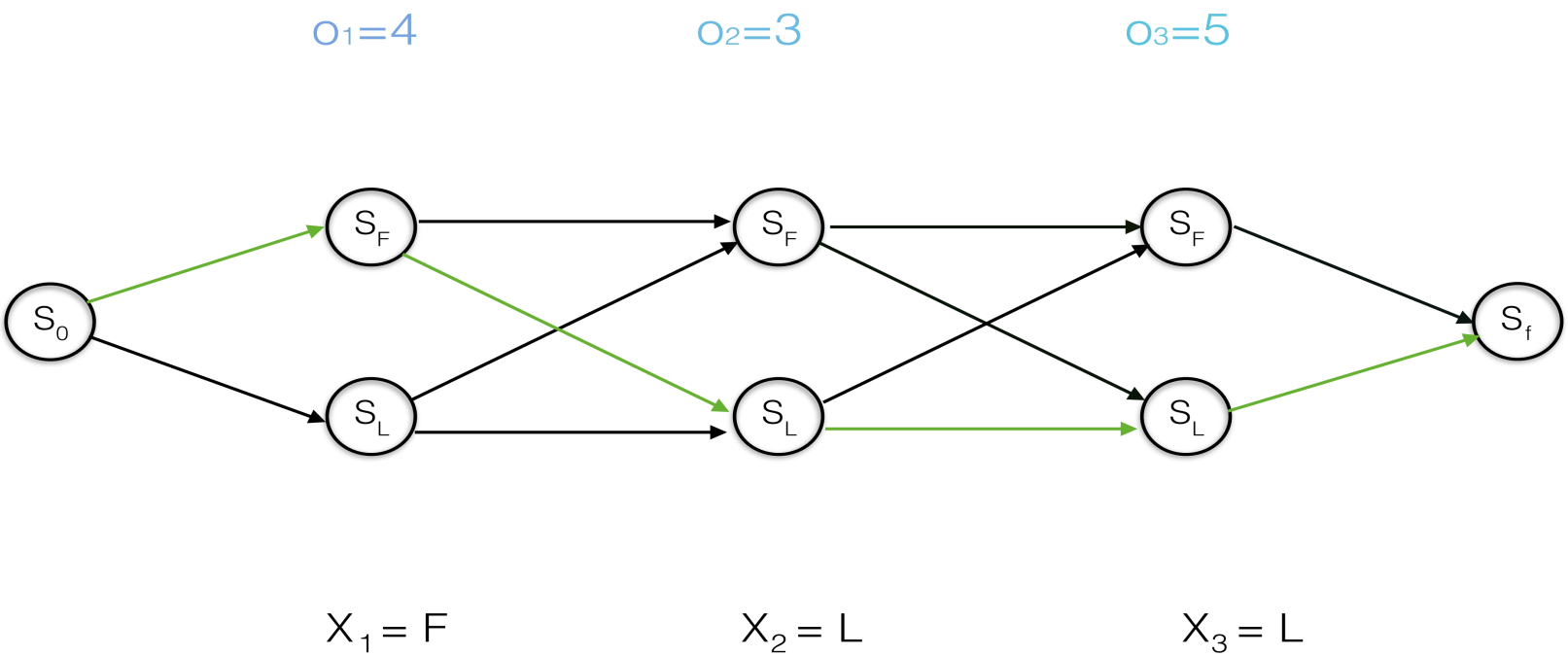
Viterbi algorithm, backtracing



Viterbi algorithm, backtracing



Viterbi algorithm, backtracing



Why is it necessary to keep N states at each time step?

- We have convinced ourselves that it's not necessary to keep more than N ("real") states per time step.
- But could we cut down the table to just a one-dimensional table of T time slots by choosing the probability of the best path overall ending in that time slot, in any of the states?
  - This would be the greedy choice
  - But think about what could happen in a later time slot.
  - You could encounter a zero or very low probability concerning all paths going through your chosen state s_j at time t.
  - Now a state s_k that looked suboptimal in comparison to s_j at time t becomes the best candidate.
  - As we don't know the future, this could happen to any state, so we need to keep the probabilities for each state at each time slot.
- But thankfully, no more.

Precision and Recall

- We can measure system success using accuracy.
- But sometimes it's only one type of instances that we find interesting.
- We don't want a summary measure that averages over interesting and non-interesting instances, as accuracy does.
- In those cases, we use precision, recall and F-measure.
- These metrics are imported from the field of information retrieval, where the difference beween interesting and non-interesting examples is particularly high.
- Accuracy doesn't work well when the types of instances are unbalanced.

Precision and Recall

Measure precision of L (P_L), recall of L (R_L) and F-measure of L (F_L).

		System says: F	L	Total
Truth is:	F	a	b	a+b
	L	c	d	c+d
	Total	a+c	b+d	a+b+c+d

- Precision of L: $P_L = \frac{d}{b+d}$
- Recall of L: $R_L = \frac{d}{c+d}$
- F-measure of L: $F_L = \frac{2P_L R_L}{P_L + R_L}$
- Accuracy: $A = \frac{a+d}{a+b+c+d}$

Literature

- Manning and Schutze (2000). Foundations of Statistical Natural Language Processing, MIT Press. Chapter 9.3.2.
  - We use a state-emission HMM, but this textbook uses an arc-emission HMM. There is therefore a slight difference in the algorithm as to in which step the initial and final $b_j(k_t)$ are multiplied in.
- Jurafsky and Martin, 2nd Edition, chapter 6.4
- Smith, Noah A. (2004). Hidden Markov Models: All the Glorious Gory Details.
- Bockmayr and Reinert (2011). Markov chains and Hidden Markov Models. Discrete Math for Bioinformatics WS 10/11.

A word on baseline algorithms / systems

- Let's say you've implemented Viterbi, and measured its performance.
- How "good" an algorithm's performance is depends on what the alternatives are.
- A baseline is defined as a simpler alternative that serves as a comparison.
- We are often only interested in the difference between our system over the baseline.
- Creating a convincing and informative baseline is an art.

Conditions for baselines

- A baseline should represent considerably less effort than your proposed solution
- It should be easily understandable to other researchers
  - Always picking the most frequent category (MFC)
  - A random process
  - An existing implementation (by somebody else)
- The strength of the baseline is important: the stronger the baseline the better for you, provided you still beat the baseline
  - It should not be too easy, otherwise you will be be accused of using a “strawman”.
  - it should have “fair” access to a reasonable amount of information.

Baseline for HMM

- Random guess of hidden states
  - only knows how often the HMM is in each state on average (the observed distribution of states)
  - does not know anything about sequences or observations
- This tests how much HMM modelling helps in this highly ambiguous setup.

Observed system improvement

- Your model produced a better system.
- Or at least, you observed higher performance (eg, higher F measure).
- We use a statistical test to gather evidence that one system is really better than another system.

Variation in the data

- Some examples will make it easier for your system to score well, some will make it easier for the other system.
- Maybe you were just lucky and *all* examples in the test set are in your favour?
  - This could be the case if you don't have enough data.
  - This could be the case if the difference in performance is small.
- Maybe both systems perform equally well in reality?

Statistical Significance Testing

- Null Hypothesis: two result sets come from the same distribution
  - System 1 is (really) equally good as System 2.
- First, choose a significance level (α), e.g., $\alpha = 0.01$ or 0.05.
- We then try to reject the null hypothesis with confidence $1 - \alpha$ (99% or 95% in this case)
- Rejecting the null hypothesis means showing that the observed result is very unlikely to have occurred by chance.

Reporting significance

- If we successfully pass the significance test, and only then, we can report:

  "The difference between System 1 and System 2 is statistically significant at $\alpha = 0.01$."

- Any other statements based on raw accuracy differences alone are strictly speaking meaningless.

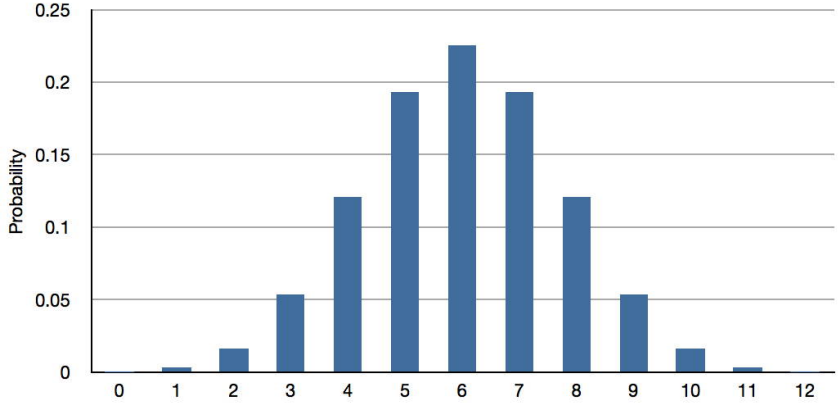
Sign Test (non-parametric, paired)

- The sign test uses a binary event model.
- Events have binary outcomes:
  - Positive: System 1 beats System 2 on this example.
  - Negative: System 2 beats System 1 on this example.
  - (Tie: System 1 and System 2 do equally well on this example / have identical results – more on this later).
- Null hypothesis → baseline and our system are equally good
  - prob of negative outcome = prob of positive outcome
  - Counts obey a binomial distribution with a mean of 0.5
- How likely is it that our observations could arise in that situation?
  - The probability that (at most) 753 out of 2,000 are negative.

Binomial Distribution $B(N, q)$

- Call the probability of a negative outcome q (here $q = 0.5$)
- Probability of observing $X = k$ negative events out of N:

$$P_q(X = k|N) = \binom{N}{k} q^k (1 - q)^{N-k}$$



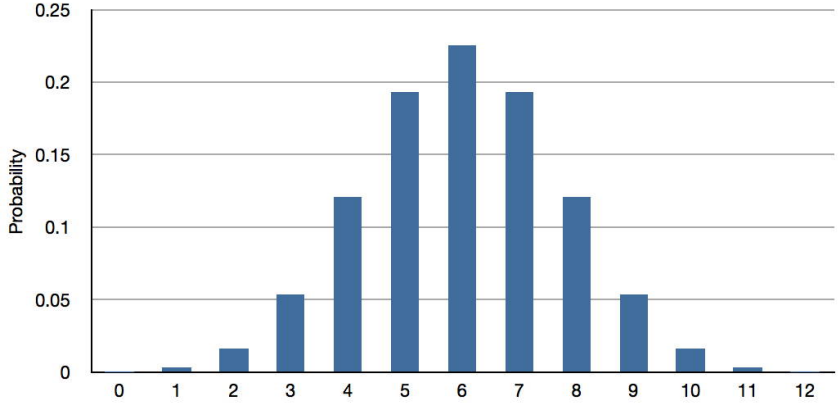
Binomial Distribution $B(N, q)$

- Call the probability of a negative outcome q (here $q = 0.5$)
- Probability of observing $X = k$ negative events out of N:

$$P_q(X = k|N) = \binom{N}{k} q^k (1-q)^{N-k}$$

- At most k negative events:

$$P_q(X \leqslant k|N) = \sum_{i=0}^{k} \binom{N}{i} q^i (1-q)^{N-i}$$



Binary Event Model and Statistical Tests

- If the probability of observing our events under the Null Hypothesis is very small (smaller than our pre-selected significance level α, e.g., 0.01), we can safely reject the Null hypothesis.
- The $P(X \leqslant k)$ we just calculated directly gives us the probability p we are after.
- This means there is only a 1% chance that System 1 does not beat System 2.

Two-Tailed vs. One-Tailed Tests

- Testing difference in a specific direction:
  - System 1 better than System 2 [One-tailed test]
- A more conservative, rigorous test would be a non-directional one (though some debate on this!)
  - Testing for statistically significant difference regardless of direction [Two-tailed test]
  - This is given by $2P(X \leqslant k)$ (because $B(N, 0.5)$ is symmetric).

Treatment of Ties

- When comparing two systems in classification tasks, it is common for a large number of ties to occur.
- Disregarding ties will tend to affect a study's statistical power.
- Can treat ties by adding 0.5 events to the positive and 0.5 events to the negative side (and round up at the end).

Literature

- Siegel and Castellan (1988). *Non-parametric statistics for the behavioral sciences*, McGraw-Hill, 2nd. Edition.
  - Chapter 2: The use of statistical tests in research
  - Sign test: p. 80–87