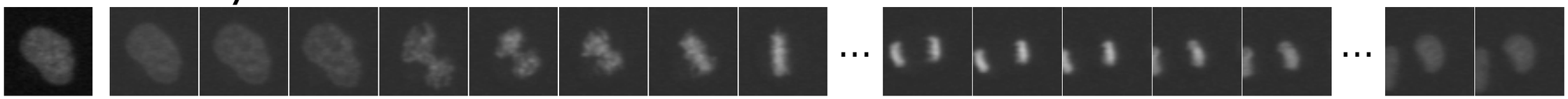
Hidden Markov Models

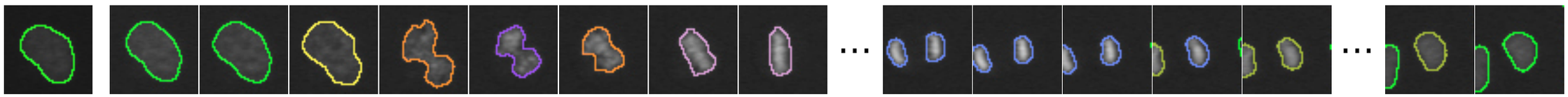
Thomas Walter

Sequential data

H2B-mCherry



Classification results



The problem

- Independent classification of each single nucleus at each single time point (only based on texture and shape at that timepoint).
- The history is thus completely neglected. Also, it is possible to have "non-sense" transitions between morphologies.
- While the number of single classification errors might be low, the number of tracks containing at least one error is actually still high.
- Classical example of sequential data: there is a need for a statistical framework for such data.

Overview

- Basic notions for graphical models
- Markov Models
- Hidden Markov Models
- Algorithms for Hidden Markov Models:
  - evaluation
  - decoding
  - estimation / learning
- Applications

Basic notions

- We limit ourselves to discrete probability distributions.
- $P(X) \in [0, 1]$ is the probability distribution of random variable X, $P(X, Y)$ the joint probability distribution of two random variables X and Y and $P(X|Y)$ is the conditional probability distribution of X given Y.
- There are two important rules:

Sum rule:	$P(X) = \sum_Y P(X, Y)$
Product rule:	$P(X, Y) = P(X\|Y)P(Y)$

Summation in the sum rule is made over all possible values Y can take.

Independence and conditional independence

- Two variables X and Y are independent, when the joint probability factorizes as the product of the marginals.

$$X \perp\!\!\!\perp Y \quad \Leftrightarrow \quad P(Y|X) = P(Y) \quad \Leftrightarrow \quad P(Y, X) = P(Y)P(X)$$

- Equivalently, we can say that the conditioning on X does not change the probability, i.e. knowing X will not teach us anything about Y and vice versa.
- X is conditionally independent on Y given Z, if:

$$X \perp\!\!\!\perp Y|Z \quad \Leftrightarrow \quad P(Y|X, Z) = P(Y|Z) \quad \Leftrightarrow \quad P(Y, X|Z) = P(Y|Z)P(X|Z)$$

- Given that we know Z, knowledge of X does not teach us anything in addition about Y.

Examples

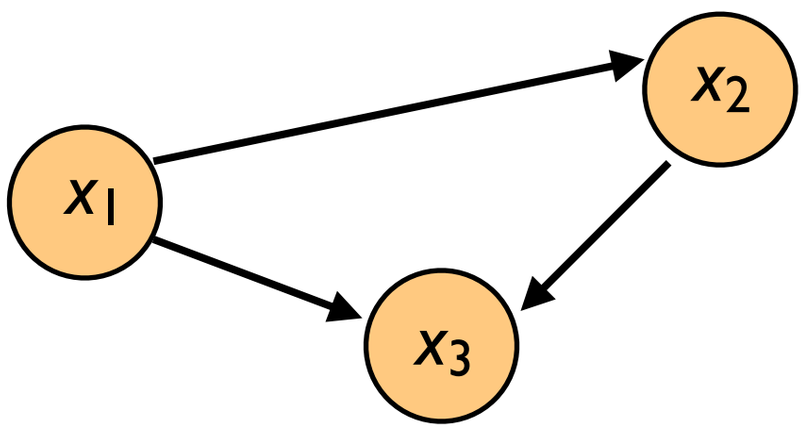
- Independent events:
  - Y: getting promoted
  - X: finding an excellent novel to read
  - Y is probably independent from X
- Dependent events:
  - Y: getting promoted
  - X: successful project
  - Here it is reasonable to assume dependence, as the success on your last project might have an impact on your promotion.
- Conditionally independent events:
  - Y: getting promoted
  - X: successful project
  - Z: favorable report on past activities.
  - Here, X and Y are conditionally independent knowing Z, as knowing X will not teach me anything more than Z.

Graphical models

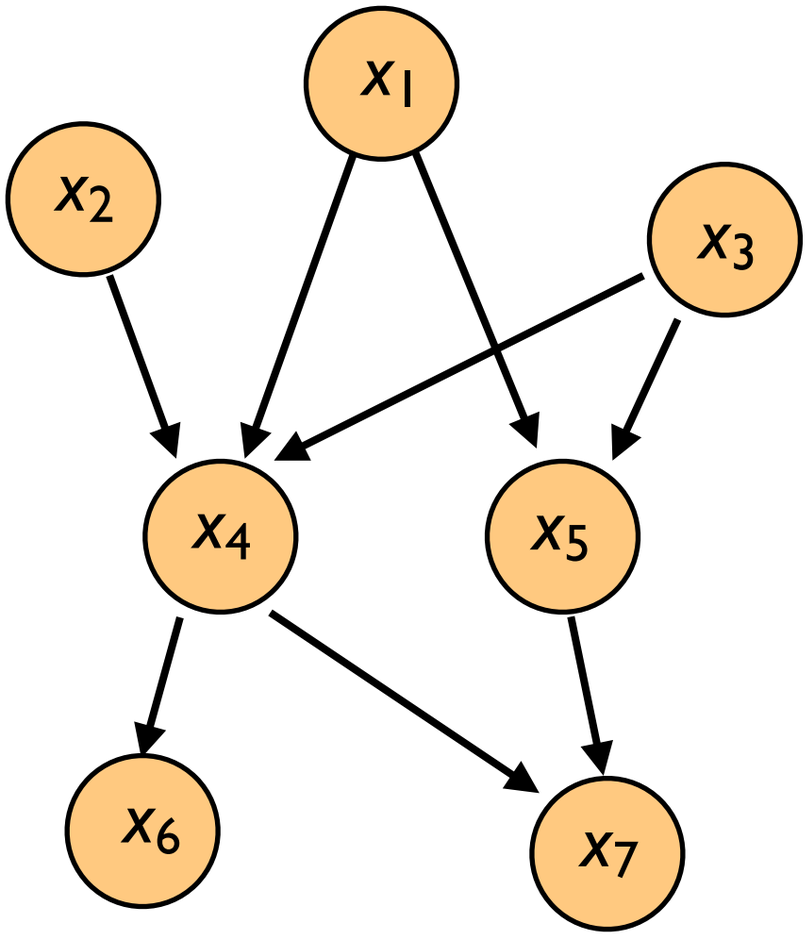
- We consider three variables x_1, x_2, x_3. The joint probability distribution $p(x_1, x_2, x_3)$can be factorized (using the product rule):

$$\begin{aligned} p(x_1, x_2, x_3) &= p(x_1, x_2)p(x_3|x_1, x_2) \\ &= p(x_2|x_1)p(x_1)p(x_3|x_1, x_2) \end{aligned}$$

- This factorization can be represented by a graph:
  - The random variables are the nodes. Each of these random variables follows a certain conditional distribution $p(x_i|x_1, x_2, \ldots)$.
  - The edges represent the random variables on which these distributions are conditioned.



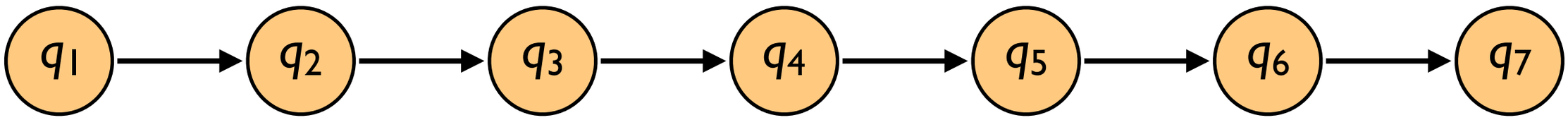
Graphical models



- We start with the marginals (probability distributions without conditioning).
- We can now write successively the conditional probability distribution for the variables, whose incoming edges have already been taken into account.
- Doing this for all nodes in the graph leads to the factorization symbolized by the graph.

$$p(x_1, x_2, x_3, x_4, x_5, x_6, x_7) =$$
$$p(x_1)p(x_2)p(x_3)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_3)p(x_6|x_4)p(x_7|x_4, x_5)$$

A particular graphical model

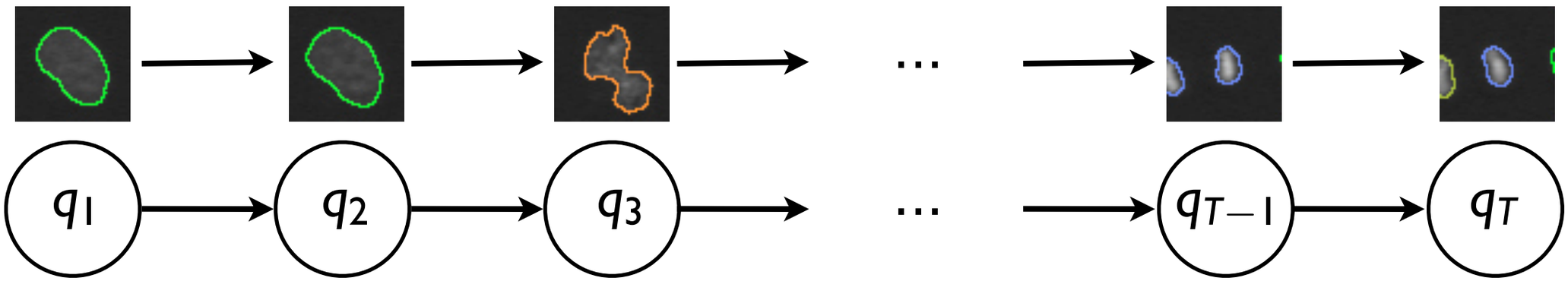


- Each random variable is conditioned on exactly one other variable (except for the first). The graphical model becomes a chain.
- The joint probability can thus be written:

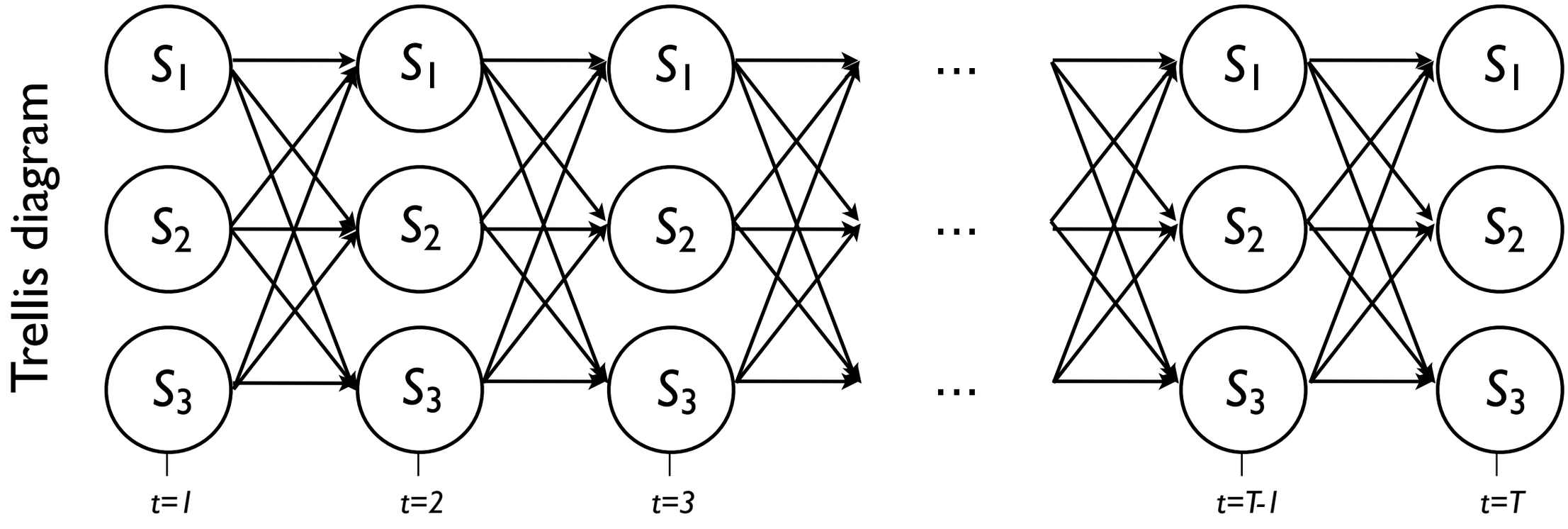
$$\begin{aligned} p(q_1, \ldots, q_T) &= p(q_1)p(q_2|q_1)p(q_3|q_2)\ldots p(q_T|q_{T-1}) \\ &= p(q_1)\prod_{t=2}^{T} p(q_t|q_{t-1}) \end{aligned}$$

- Such a graphical model is called a Markov chain.

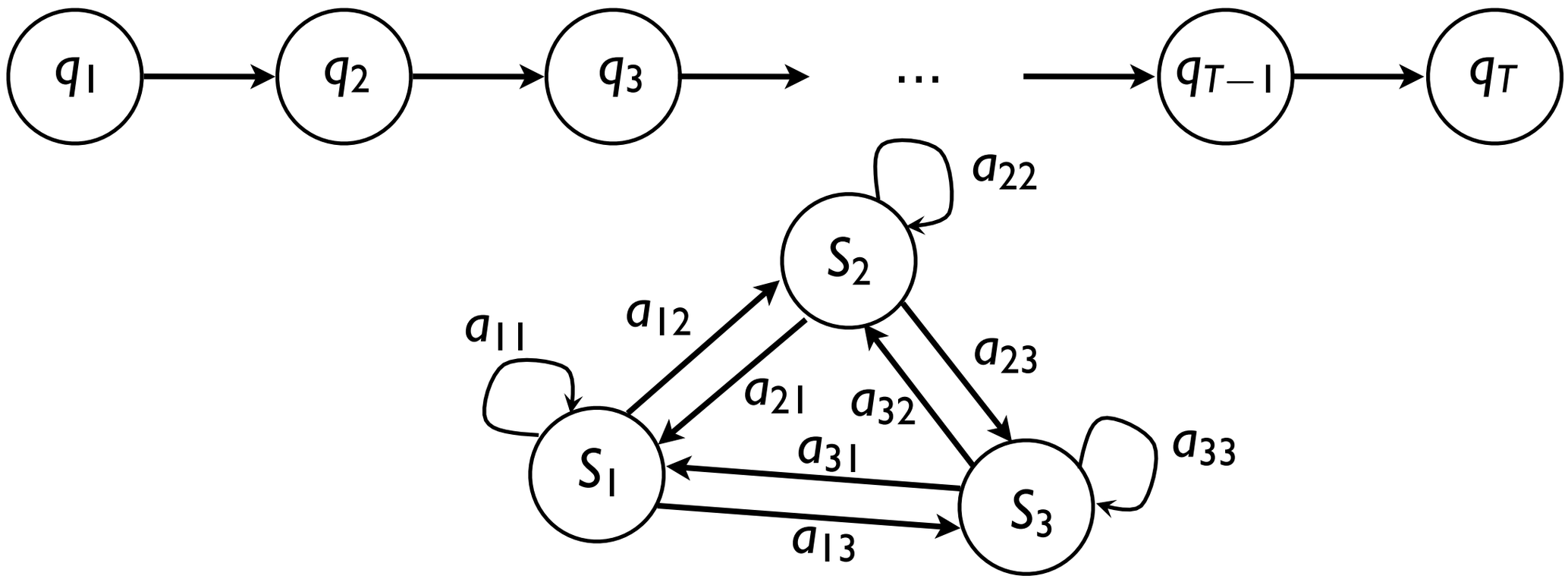
Markov Models



- Markov Models are a probabilistic framework to deal with sequential data.
- At each time point $t = 1, \ldots, T$, the system is in a state q_t, drawn from a set of possible states $\{S_1, S_2, \ldots, S_N\}$.

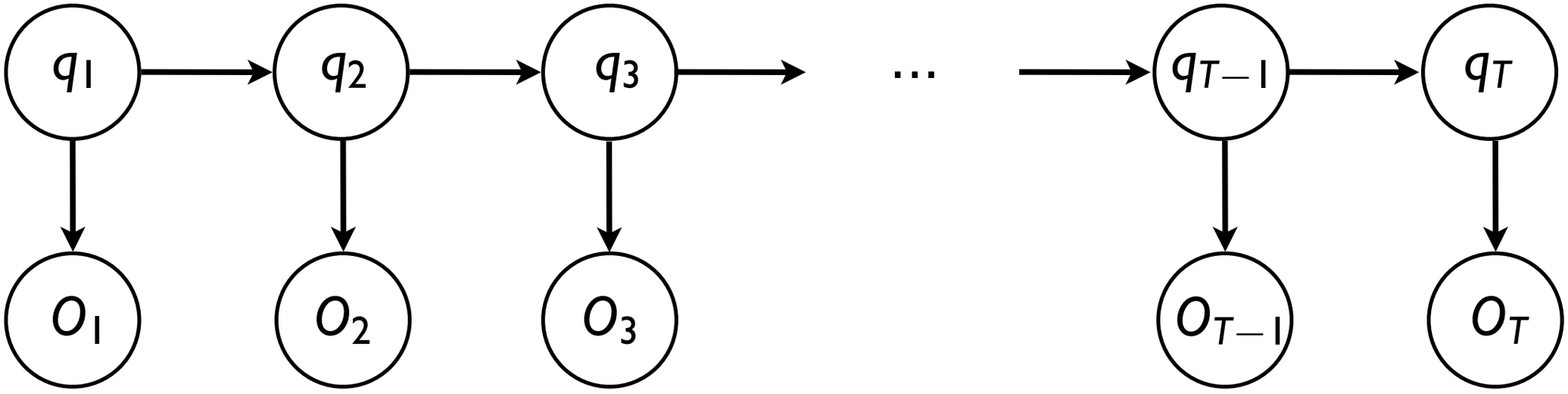


Transition probabilities



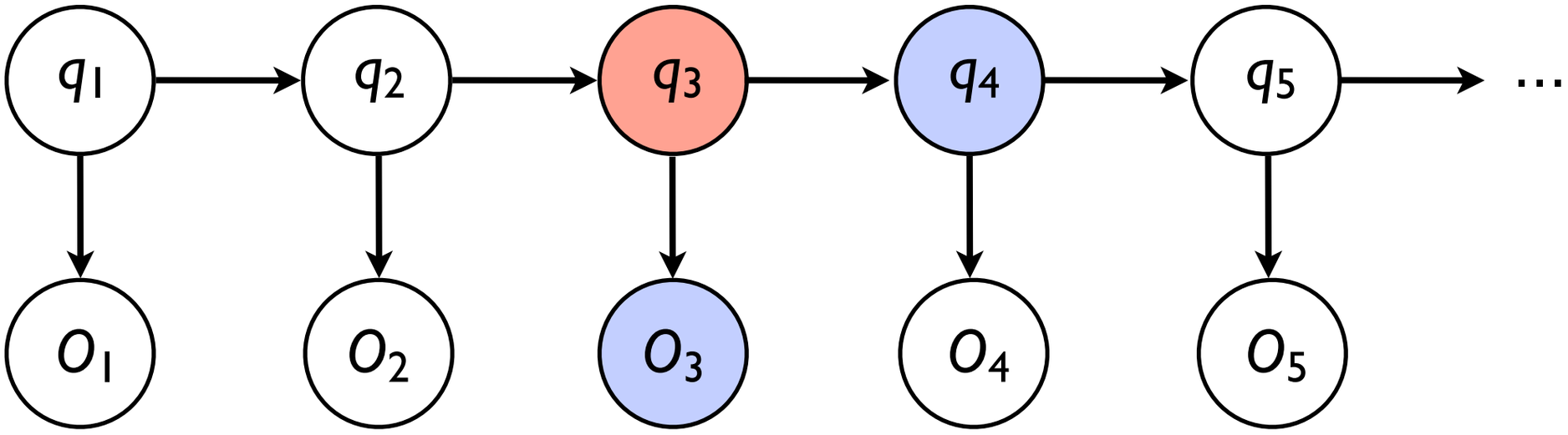
- At any given timepoint, the system can go from one state S_i to another state S_j.
- In first order Markov chains, the probability for this transition is only dependent on the previous state, i.e. $a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i, q_{t-2} = S_k \ldots) = P(q_t = S_j \mid q_{t-1} = S_i)$. This property is called "Markov property.
- Transition probabilities can be represented in the transition graph, or equivalently by the transition matrix $A = (a_{ij})_{i=1\ldots N, j=1\ldots N}$.

Emission probabilities



- Hidden Markov Models (HMM) explicitly model the uncertainty of observations.
- The observed sequence $O = (O_1, O_2, \ldots, O_T)$, where each O_t is drawn from $\{V_1, \ldots, V_M\}$, is linked to the true state sequence by the emission probabilities $b_{ij} = P(O_t = V_j | q_t = S_i)$.
- Together with the initial state distribution $\pi_i = P(q_1 = S_i)$ for $i = 1, \ldots, N$, the complete parameter set of the model is $\lambda = (A, B, \pi)$.
- For **homogeneous** Markov models, the model parameters are constant.

Independence statements



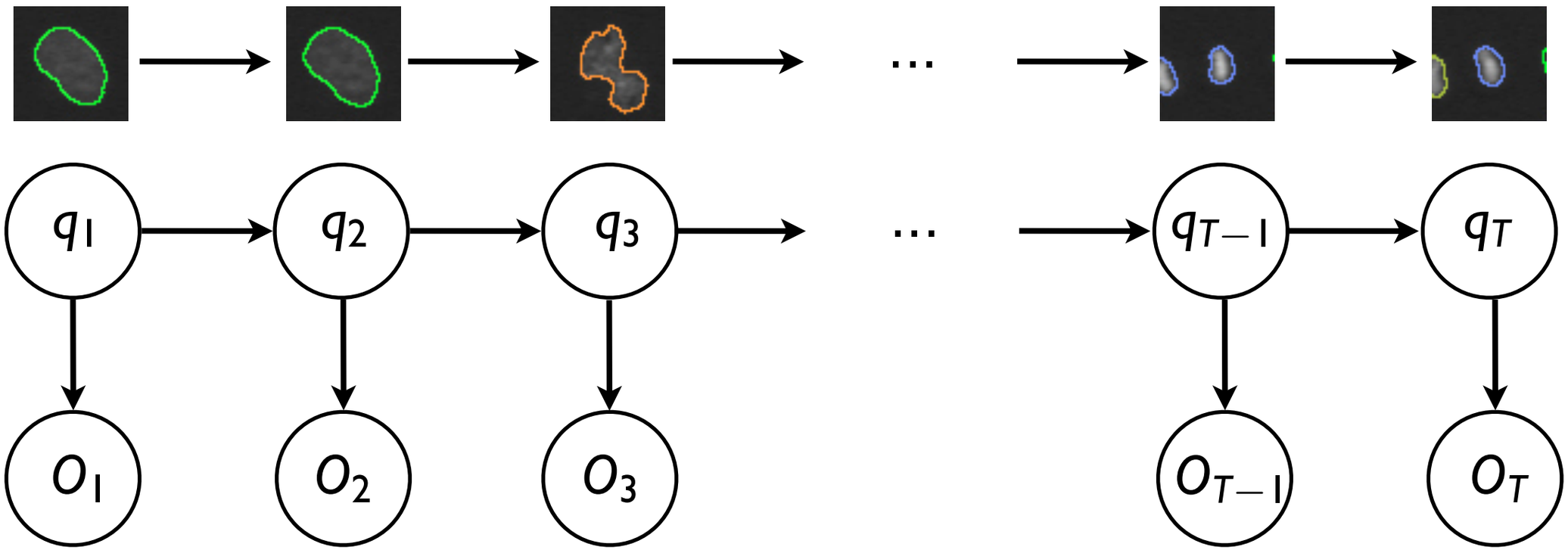
- q_{t+1} is conditionally independent from q_{t-1} knowing q_t, i.e. if I know q_t, knowledge of q_{t-1} will not teach me anything about q_{t+1}. More formally:

$$P(q_{t+1}|q_t, q_{t-1}, \ldots) = P(q_{t+1}|q_t)$$

- O_t is conditionally independent from O_{t-1}, knowing q_t. In other words, if I know q_t, observation of O_{t-1} will not teach me anything about O_t.

$$P(O_t|q_t, O_{t-1}, O_{t-2} \ldots) = P(O_t|q_t)$$

The 3 problems for HMM



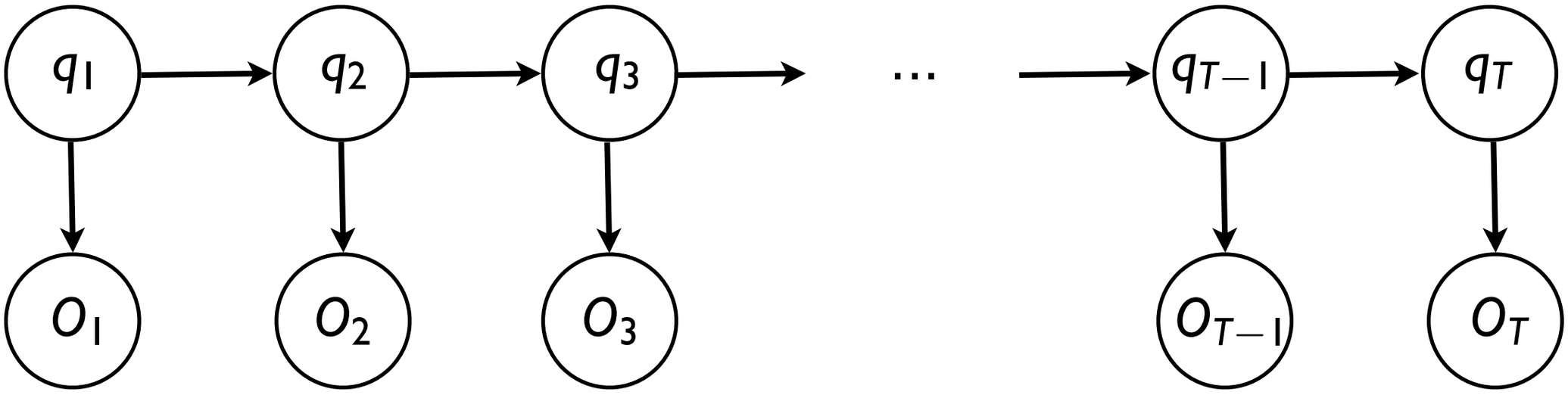
There are three classical problems for HMMs:

1. Given the observation sequence $O = O_1 O_2 \ldots O_T$ and a model $\lambda = (A, B, \pi)$: how do we efficiently compute $P(O|\lambda)$?

2. Given the observation sequence $O = O_1 O_2 \ldots O_T$ and the model λ, what is the state sequence $Q = Q_1 Q_2 \ldots Q_T$ which best explains the observations?

3. How can we estimate the model parameters λ to maximize $P(O|\lambda)$?

Problem 1: direct calculation



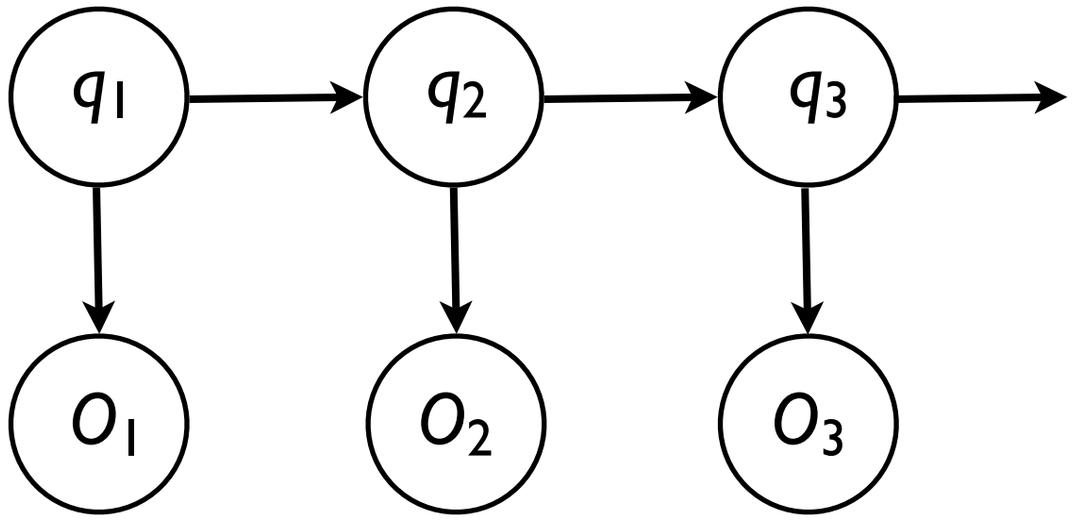
We suppose that the sequence $O = (O_1, O_2, \ldots, O_T$ was observed. Let us consider a fixed state sequence $Q = (Q_1, Q_2, \ldots, Q_T)$. We can then calculate the joint probability given the model $P(O, Q|\lambda)$:

$$\begin{aligned} P(O, Q|\lambda) &= \pi_{q_1} b_{q_1}(O_1) a_{q_1,q_2} b_{q_2}(O_2) a_{q_2,q_3} b_{q_3}(O_3) \ldots a_{q_{T-1},q_T} b_T(O_T) \\ P(O, Q|\lambda) &= \pi_{q_1} b_{q_1}(O_1) \prod_{t=2\ldots T} a_{q_{t-1},q_t} b_{q_t}(O_t) \end{aligned}$$

We can then calculate $P(O|\lambda)$ by marginalization:

$$P(O|\lambda) = \sum_{\text{all } Q} P(O, Q|\lambda)$$

Problem 1: direct calculation



Problem:
This is extremely time consuming.
There are N^T sequences over which we have to sum, which makes this approach unfeasible in practice.

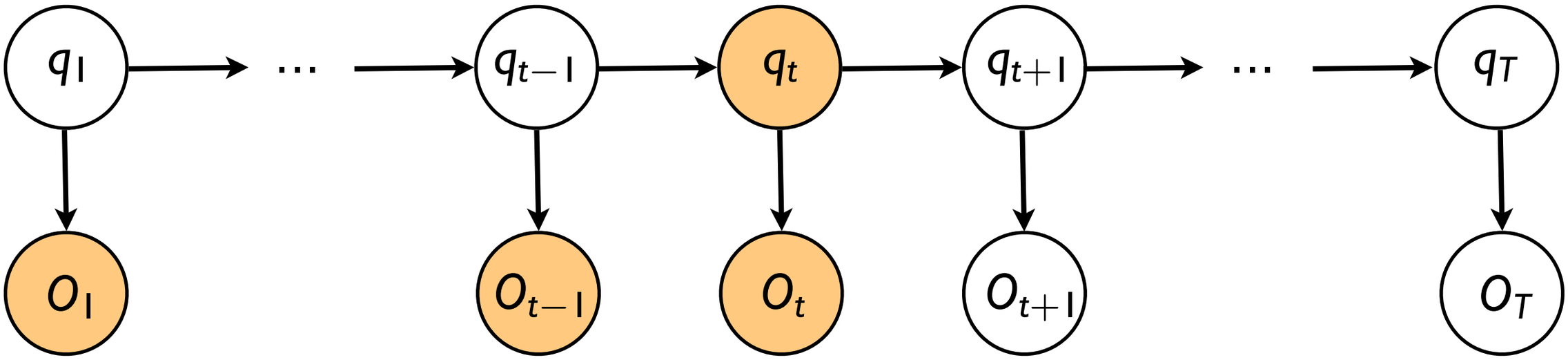
We suppose that the sequence $O = (O_1, O_2, \ldots, O_T$ was observed. Let us consider a fixed state sequence $Q = (Q_1, Q_2, \ldots, Q_T)$. We can then calculate the joint probability given the model $P(O, Q|\lambda)$:

$$\begin{aligned} P(O, Q|\lambda) &= \pi_{q_1} b_{q_1}(O_1) a_{q_1,q_2} b_{q_2}(O_2) a_{q_2,q_3} b_{q_3}(O_3) \ldots a_{q_{T-1},q_T} b_T(O_T) \\ P(O, Q|\lambda) &= \pi_{q_1} b_{q_1}(O_1) \prod_{t=2\ldots T} a_{q_{t-1},q_t} b_{q_t}(O_t) \end{aligned}$$

We can then calculate $P(O|\lambda)$ by marginalization:

$$P(O|\lambda) = \sum_{\text{all } Q} P(O, Q|\lambda)$$

Forward procedure



We consider the following joint probability:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda) \tag{1}$$

$\alpha_t(i)$ is the joint probability of the observation sequence up to timepoint t and the system being in state i at time t. This probability can be recursively calculated.

In the next slides, we drop the explicit condition on λ for better readability.

Forward procedure

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$$

Initialization:

$$\begin{aligned}\alpha_1(i) &= P(O_1, q_1 = S_i) \\ &= P(q_1 = S_i)P(O_1|q_1 = S_i) \\ &= \pi_i b_i(O_1)\end{aligned}$$

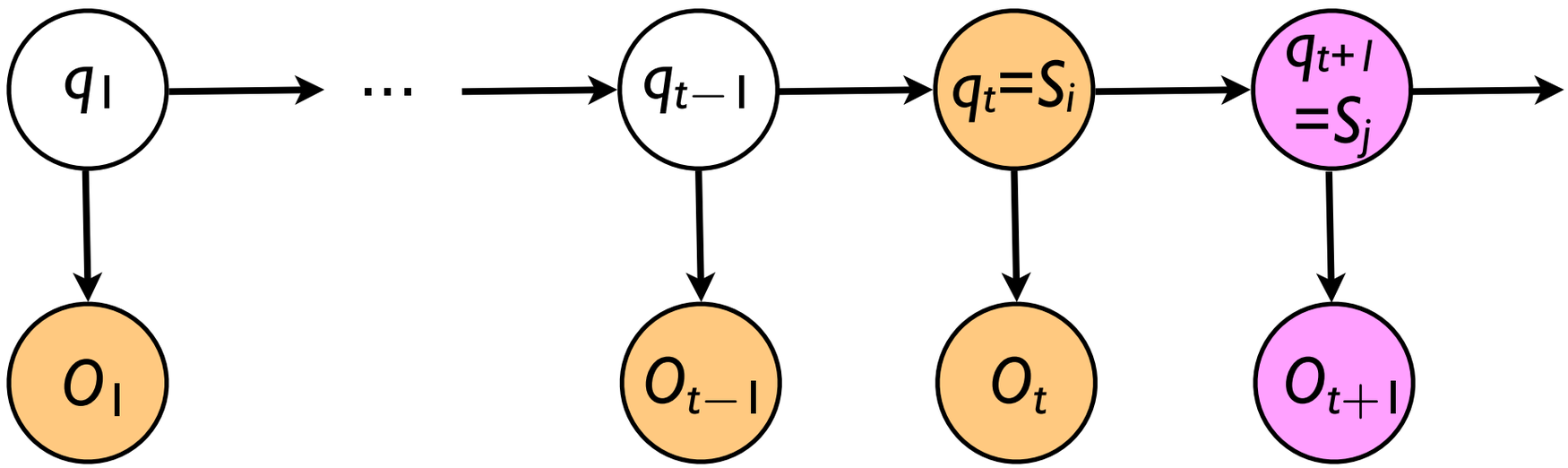
Induction:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)a_{ij}\right] b_j(O_{t+1})$$

Termination:

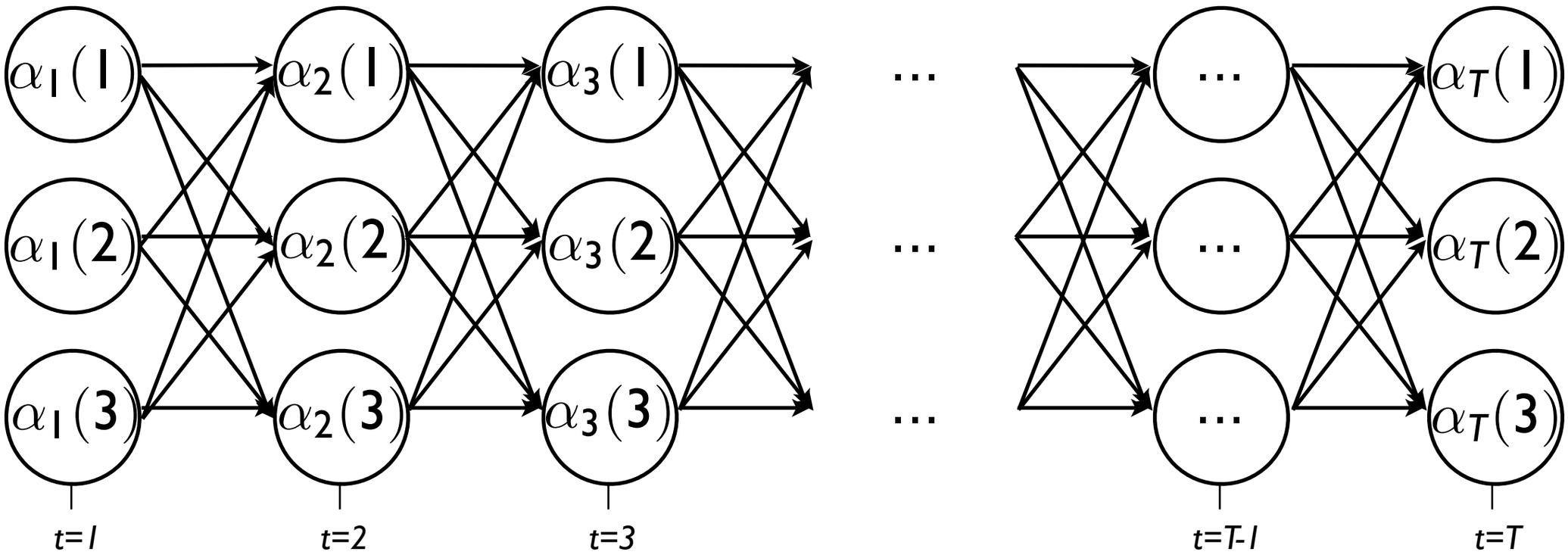
$$\begin{aligned}\alpha_T(i) &= P(O_1, \ldots, O_T, q_T = S_i) \\ P(O) &= \sum_i \alpha_T(i)\end{aligned}$$

Forward procedure: induction step



$$
\begin{aligned}
\alpha_{t+1}(j) &= P(O_1, \ldots, O_t, O_{t+1}, q_{t+1} = S_j) \\
&= \sum_{i=1}^{N} P(O_1, \ldots, O_t, q_t = S_i, O_{t+1}, q_{t+1} = S_j) \\
&= \sum_{l=1}^{N} P(O_1, \ldots, O_t, q_t = S_i) P(q_{t+1} = S_j | q_t = S_i) P(O_{t+1} | q_{t+1} = S_j) \\
&= \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1})
\end{aligned}
$$

Forward procedure



- The probabilities $\alpha_t(i)$ are stored for all *i* and *t* in a trellis structure.
- The entire set of αs can be calculated in N^2T calculations and is thus much more efficient than direct calculation.

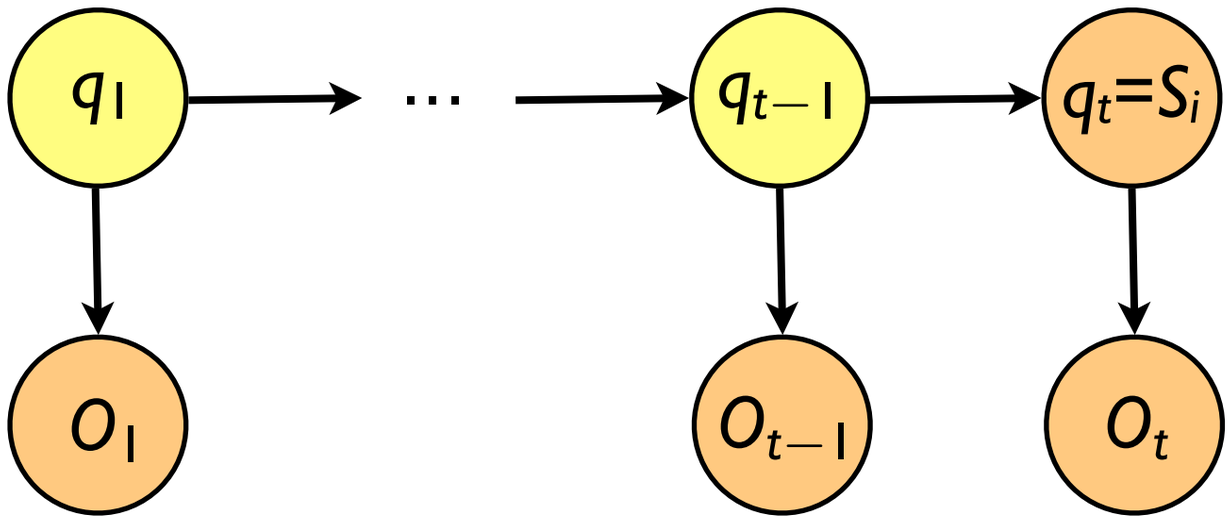
Problem 2: finding the sequence of hidden states

- Given a model and an observation sequence, we wish to find the sequence of true (hidden) states that best explains the observation (given the model).
- For this we maximize the posterior probability:

$$\begin{aligned} Q^* &= \arg\max_Q P(Q|O) \\ &= \arg\max_Q \frac{P(Q,O)}{P(O)} \\ &= \arg\max_Q P(Q,O) \end{aligned}$$

- Again, for better readability, we drop the explicit dependence of the model parameters: obviously, all calculations are done conditioned on the model parameters (i.e. with model parameters fixed).

Viterbi algorithm

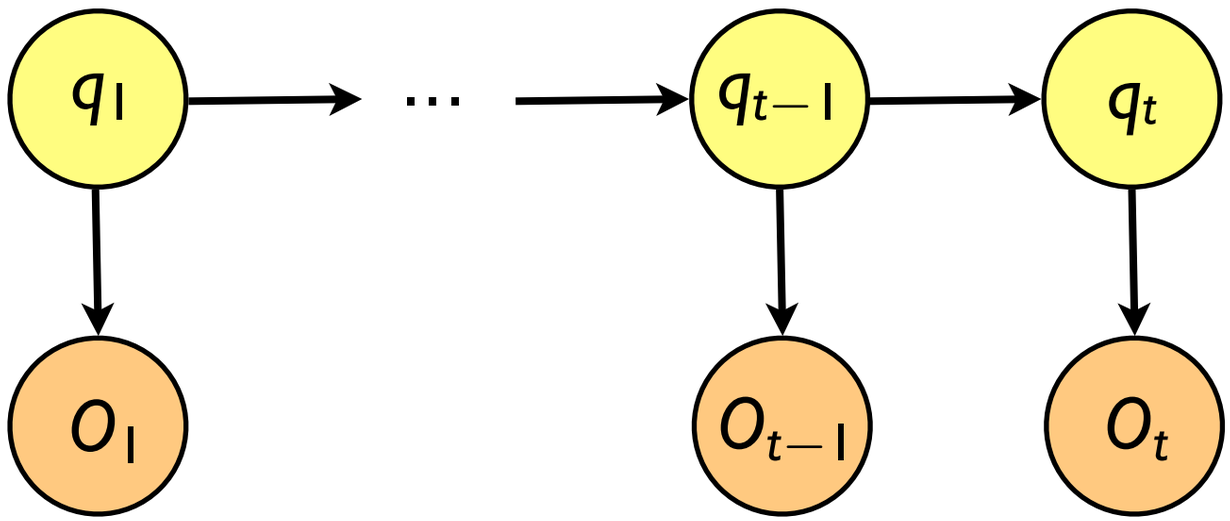


States over which maximization is done.

- We define the probability $\delta_t(i)$ as the maximal joint probability of the observation sequence O up to time t and the state sequence Q up to time $t-1$ and the state S_i at time t.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Viterbi algorithm: the recursion



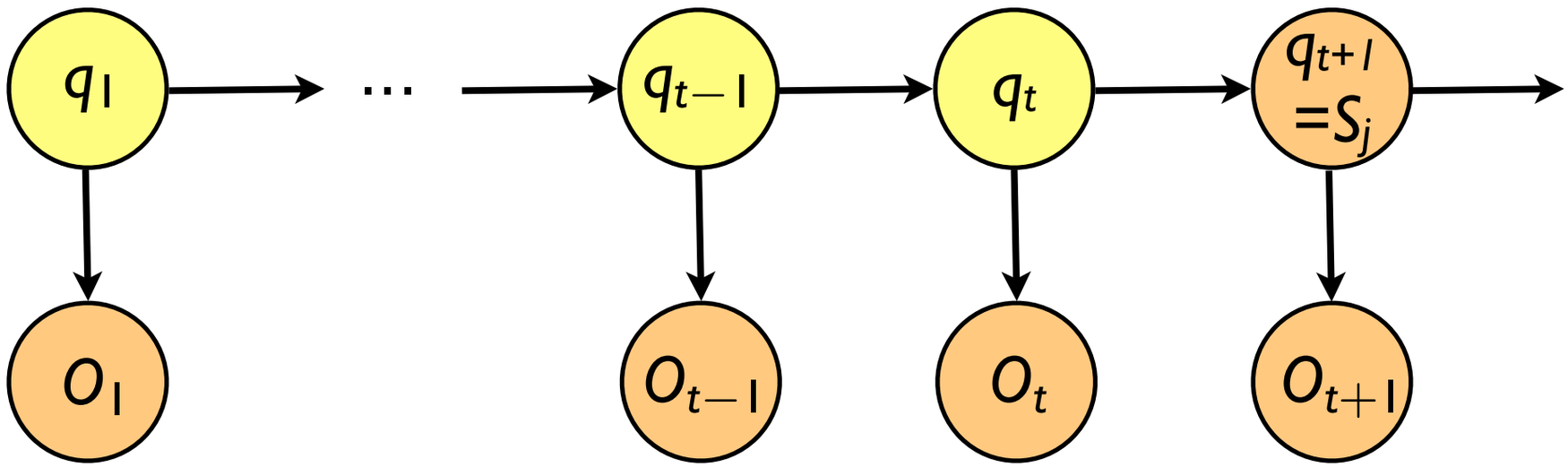
- We define the probability $\delta_t(i)$ as the maximal joint probability of the observation sequence O up to time t and the state sequence Q up to time $t-1$ and the state S_i at time t.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

- In the induction step, we have the following expression for $\delta_{t+1}(j)$:

$$\begin{aligned} \delta_{t+1}(j) &= \max_i P(O_1, \ldots, O_t, q_1, \ldots, q_t = S_i, q_{t+1} = S_j, O_t + 1) \\ &= \max_i \delta_t(i) a_{ij} b_j(O_{t+1}) \end{aligned}$$

Viterbi algorithm: the recursion



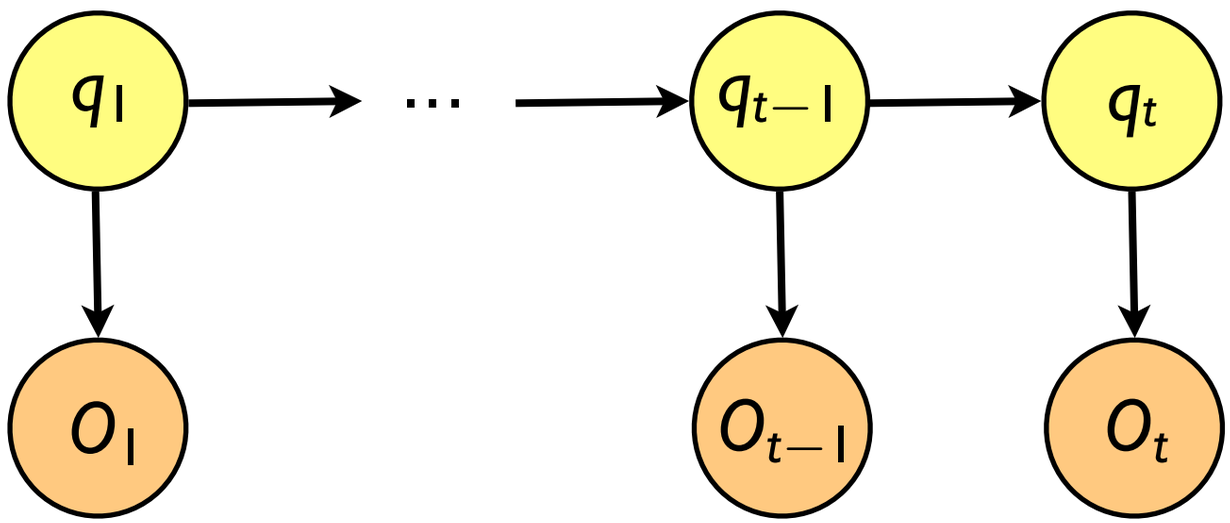
- We define the probability $\delta_t(i)$ as the maximal joint probability of the observation sequence O up to time t and the state sequence Q up to time $t-1$ and the state S_i at time t.

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

- In the induction step, we have the following expression for $\delta_{t+1}(j)$:

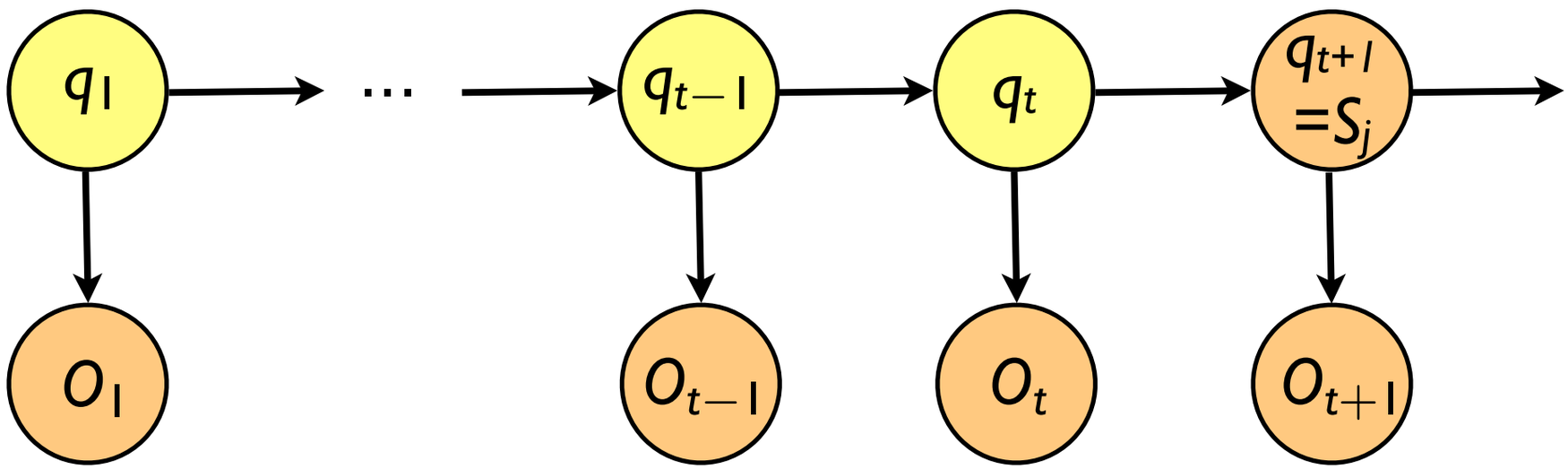
$$\begin{aligned} \delta_{t+1}(j) &= \max_i P(O_1, \ldots, O_t, q_1, \ldots, q_t = S_i, q_{t+1} = S_j, O_t + 1) \\ &= \max_i \delta_t(i) a_{ij} b_j(O_{t+1}) \end{aligned}$$

Viterbi algorithm: the induction step



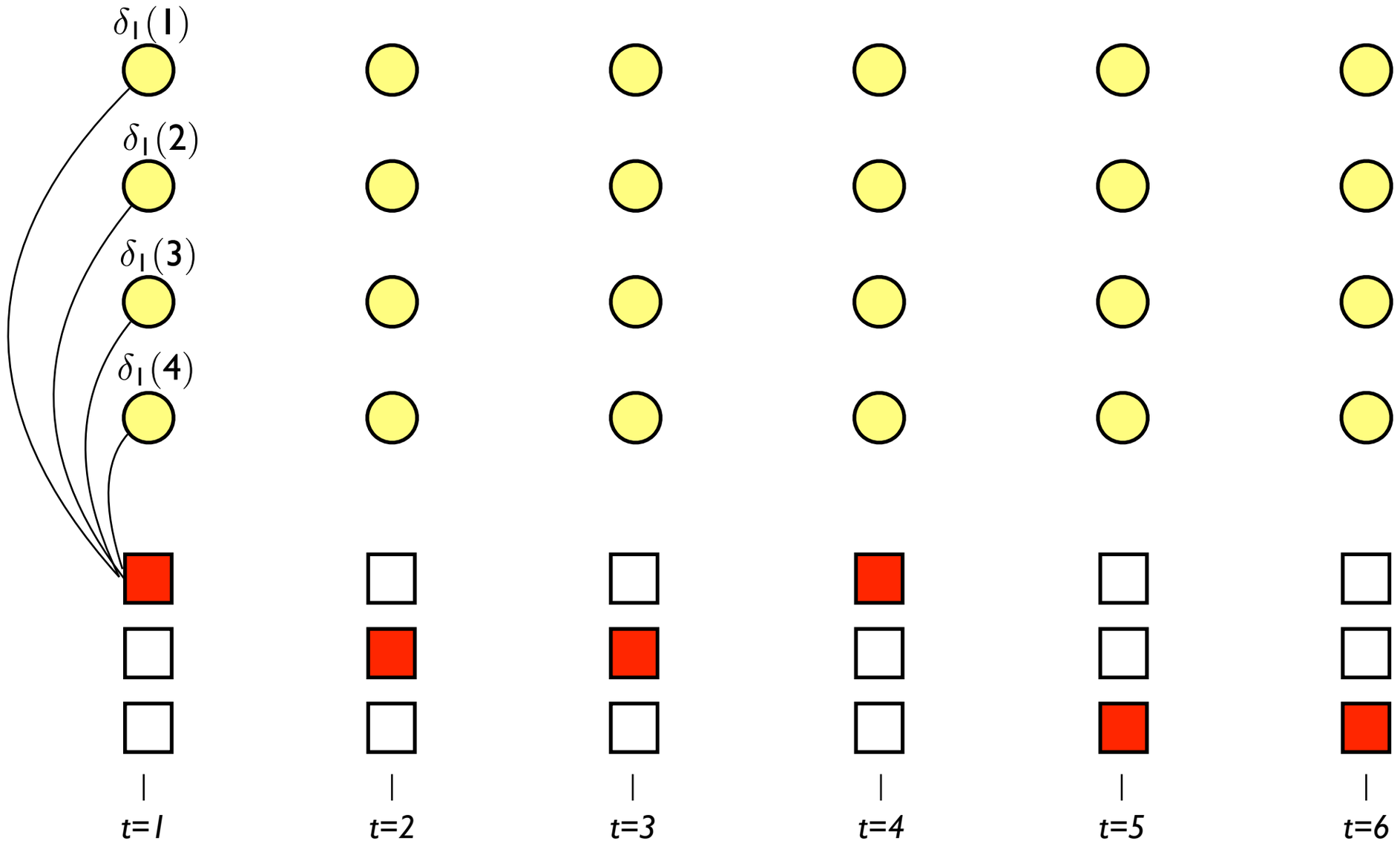
$$
\begin{aligned}
\delta_{t+1}(j) &= \max_{q_1,q_2,\ldots,q_t} P(O_1,\ldots,O_t,q_1,\ldots,q_t,q_{t+1}=S_j,O_{t+1}) \\
&= \max_{q_1,\ldots,q_{t-1}} \max_i P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i,q_{t+1}=S_j,O_{t+1}) \\
&= \max_i \max_{q_1,\ldots,q_{t-1}} P(q_{t+1}=S_j,O_{t+1}|O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i)P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \\
&= \max_i \max_{q_1,q_2,\ldots,q_{t-1}} P(q_{t+1}=S_j,O_{t+1}|q_t=S_i)P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \\
&= \max_i \left[P(q_{t+1}=S_j,O_{t+1}|q_t=S_i) \max_{q_1,q_2,\ldots,q_{t-1}} P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \right] \\
&= \max_i P(q_{t+1}=S_j|q_t=S_i)P(O_{t+1}|q_{t+1}=S_j)\delta_t(i) \\
&= \max_i \delta_t(i)a_{ij}b_j(O_{t+1})
\end{aligned}
$$

Viterbi algorithm: the induction step



$$
\begin{aligned}
\delta_{t+1}(j) &= \max_{q_1,q_2,\ldots,q_t} P(O_1,\ldots,O_t,q_1,\ldots,q_t,q_{t+1}=S_j,O_{t+1}) \\
&= \max_{q_1,\ldots,q_{t-1}} \max_i P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i,q_{t+1}=S_j,O_{t+1}) \\
&= \max_i \max_{q_1,\ldots,q_{t-1}} P(q_{t+1}=S_j,O_{t+1}|O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i)P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \\
&= \max_i \max_{q_1,q_2,\ldots,q_{t-1}} P(q_{t+1}=S_j,O_{t+1}|q_t=S_i)P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \\
&= \max_i \left[P(q_{t+1}=S_j,O_{t+1}|q_t=S_i) \max_{q_1,q_2,\ldots,q_{t-1}} P(O_1,\ldots,O_t,q_1,\ldots,q_{t-1},q_t=S_i) \right] \\
&= \max_i P(q_{t+1}=S_j|q_t=S_i)P(O_{t+1}|q_{t+1}=S_j)\delta_t(i) \\
&= \max_i \delta_t(i) a_{ij} b_j(O_{t+1})
\end{aligned}
$$

Viterbi algorithm

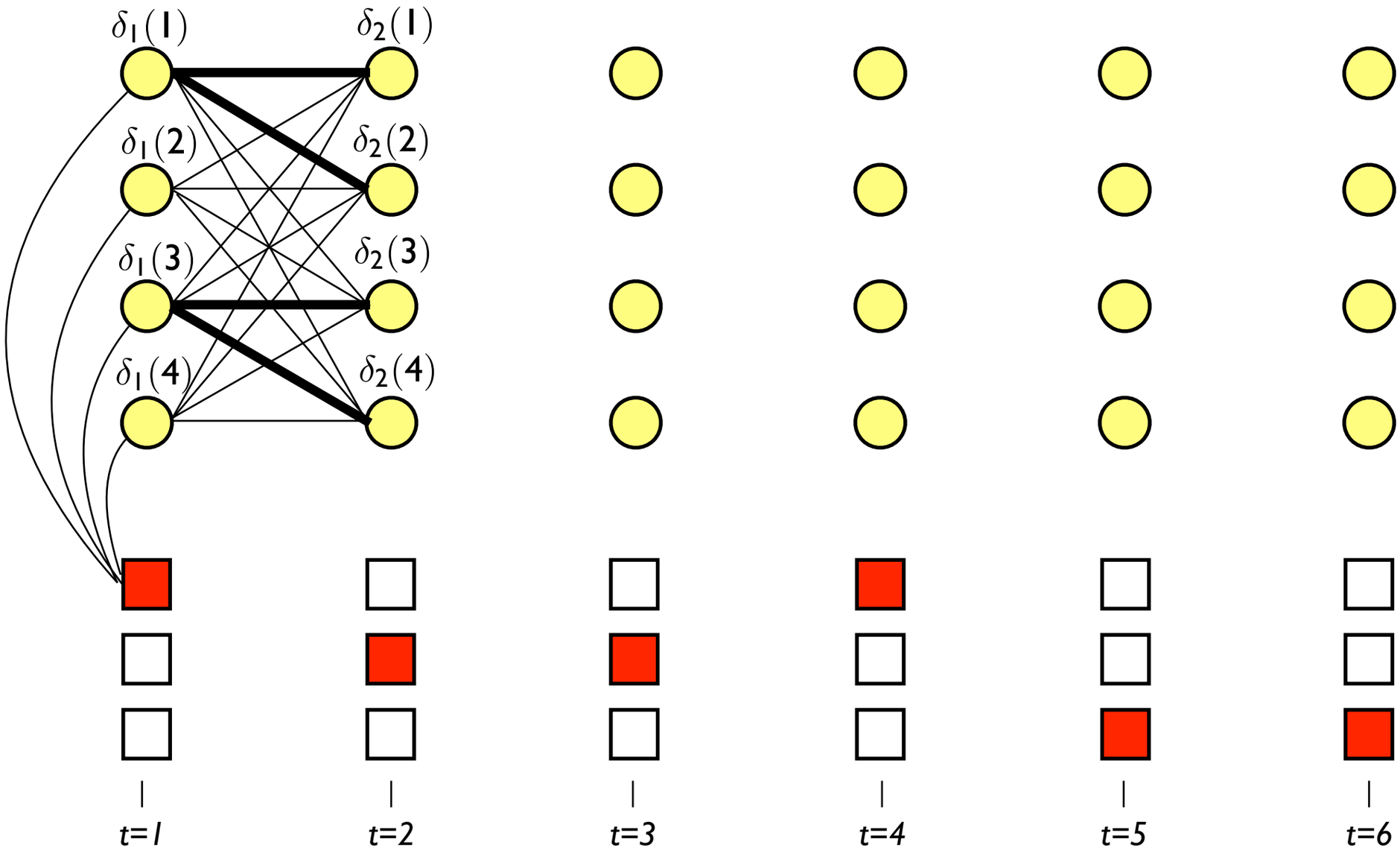


$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Initialization:

$$\delta_1(i) = \pi_i b_i(O_1)$$
$$\psi_1(i) = 0$$

Viterbi algorithm



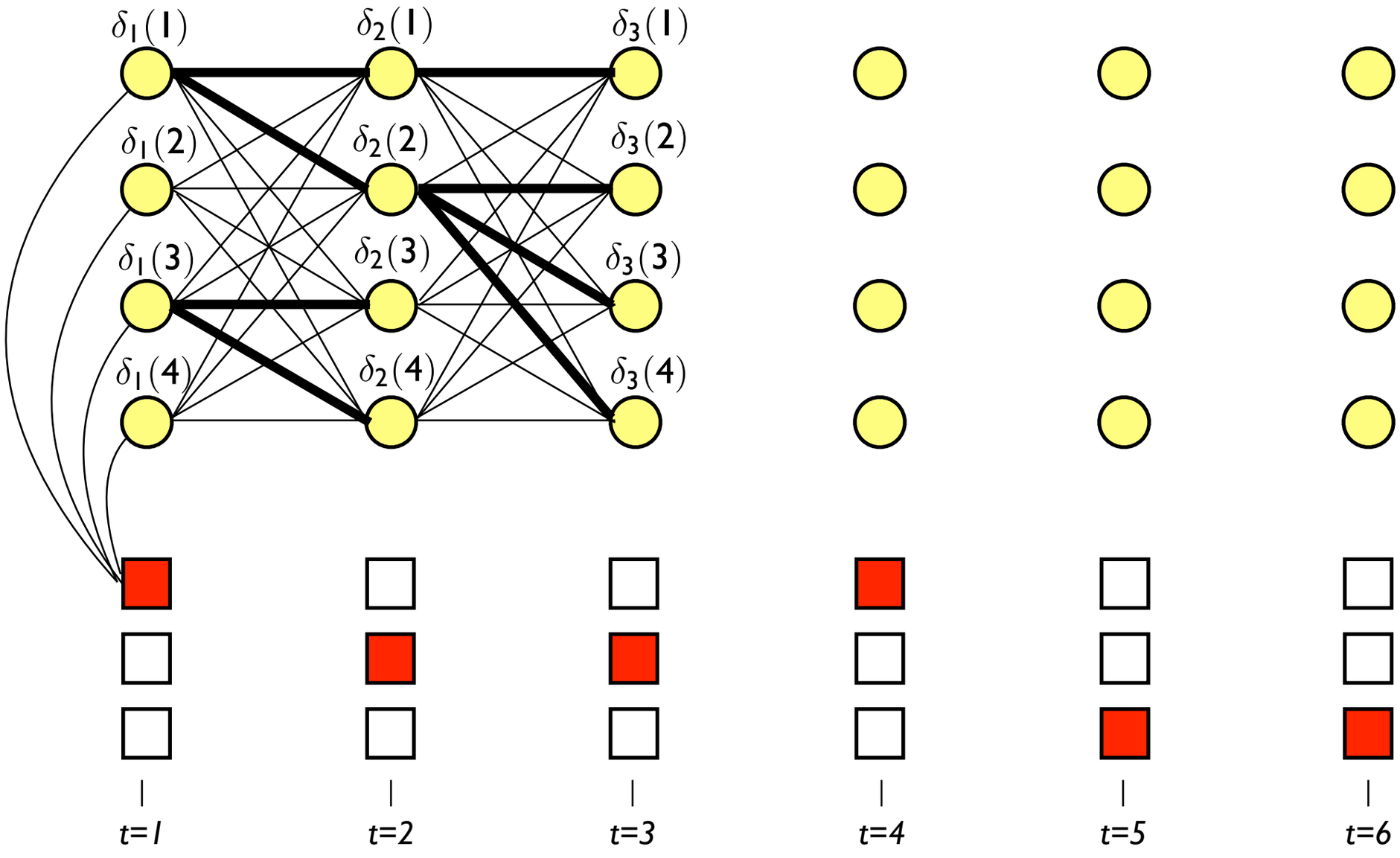
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Recursion:

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

Viterbi algorithm



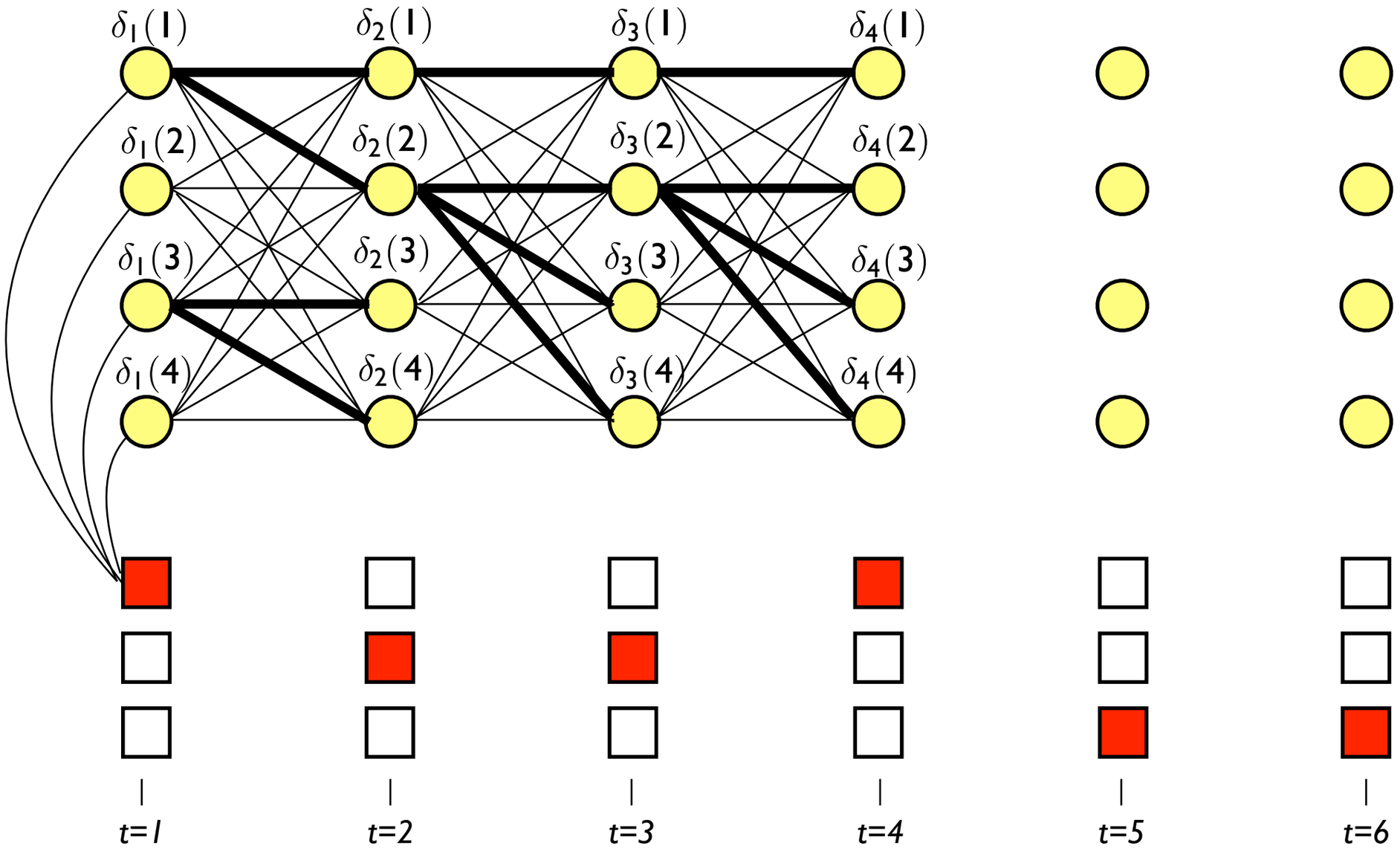
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Recursion:

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

Viterbi algorithm



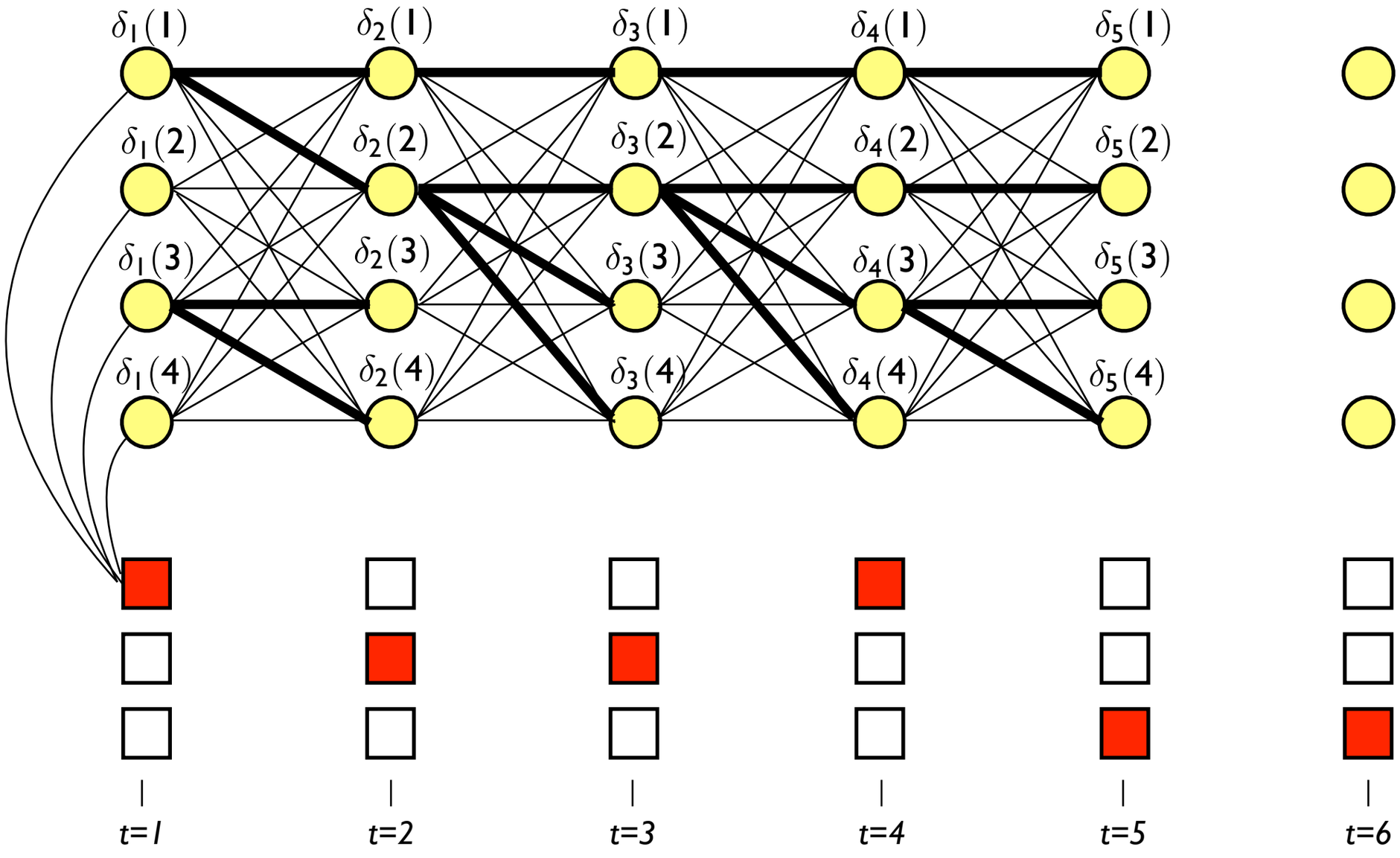
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Recursion:

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

Viterbi algorithm



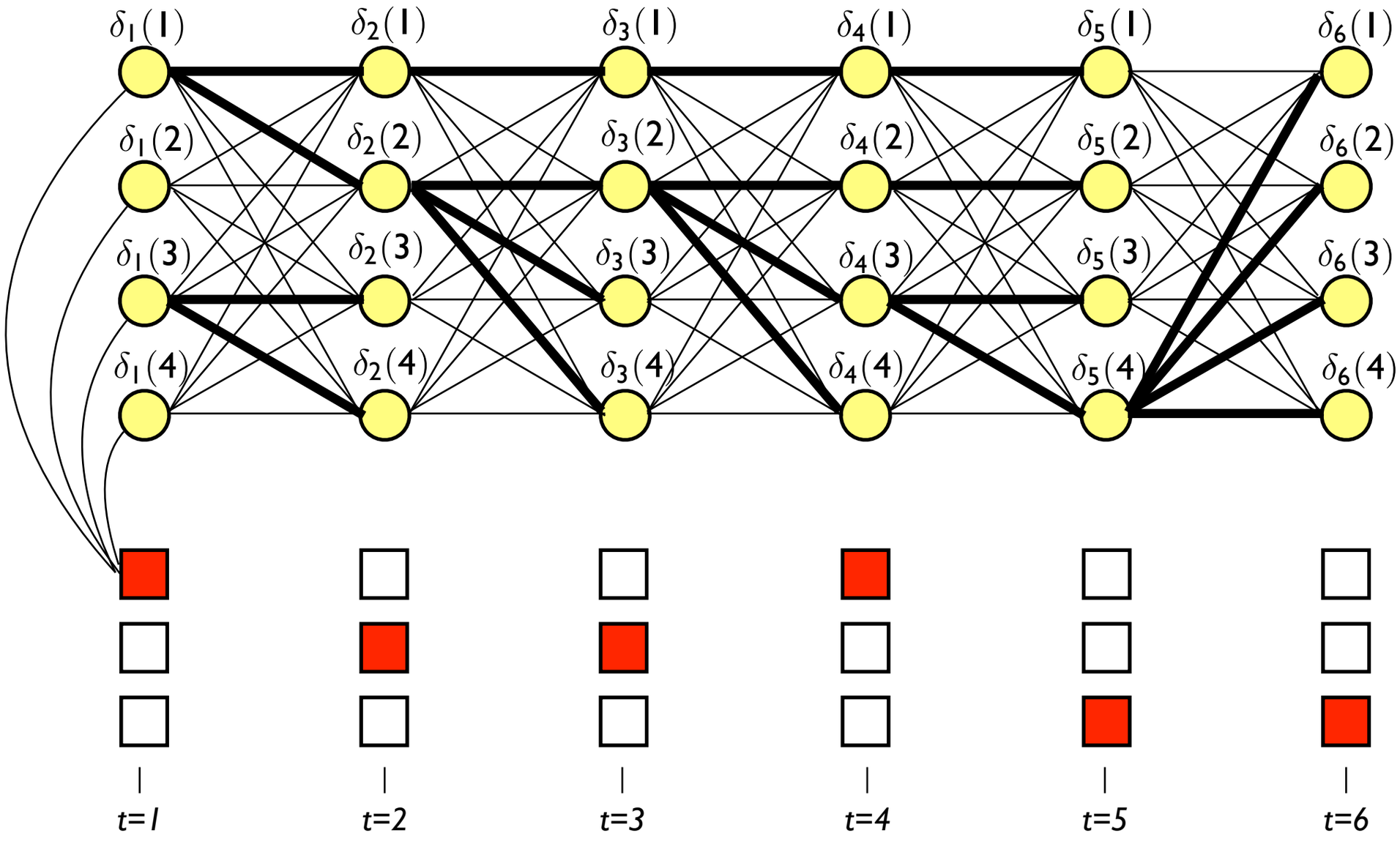
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Recursion:

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

Viterbi algorithm



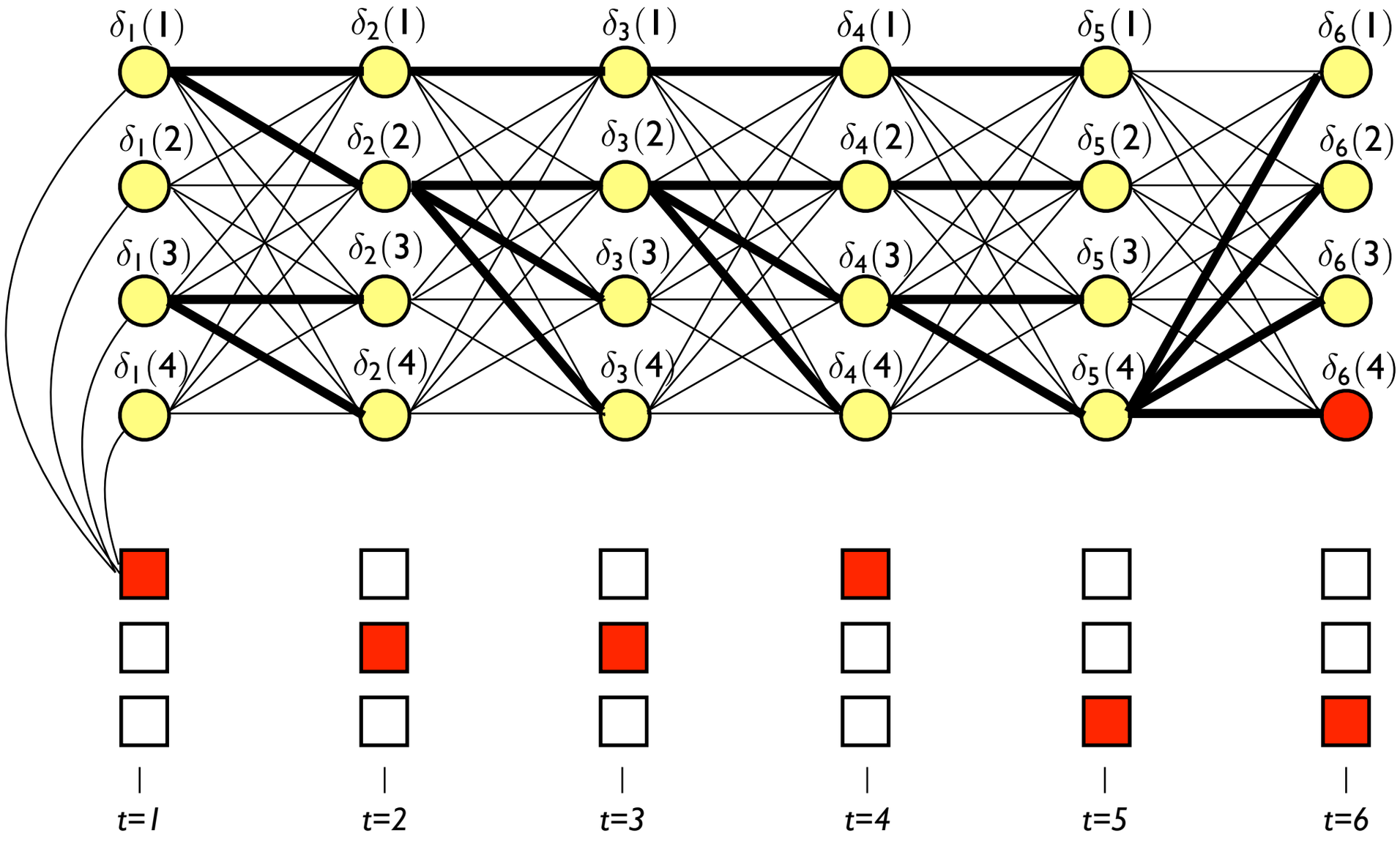
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Recursion:

$$\delta_t(j) = \max_i \delta_{t-1}(i) a_{ij} b_j(O_t)$$

$$\psi_t(j) = \arg\max_i \delta_{t-1}(i) a_{ij}$$

Viterbi algorithm



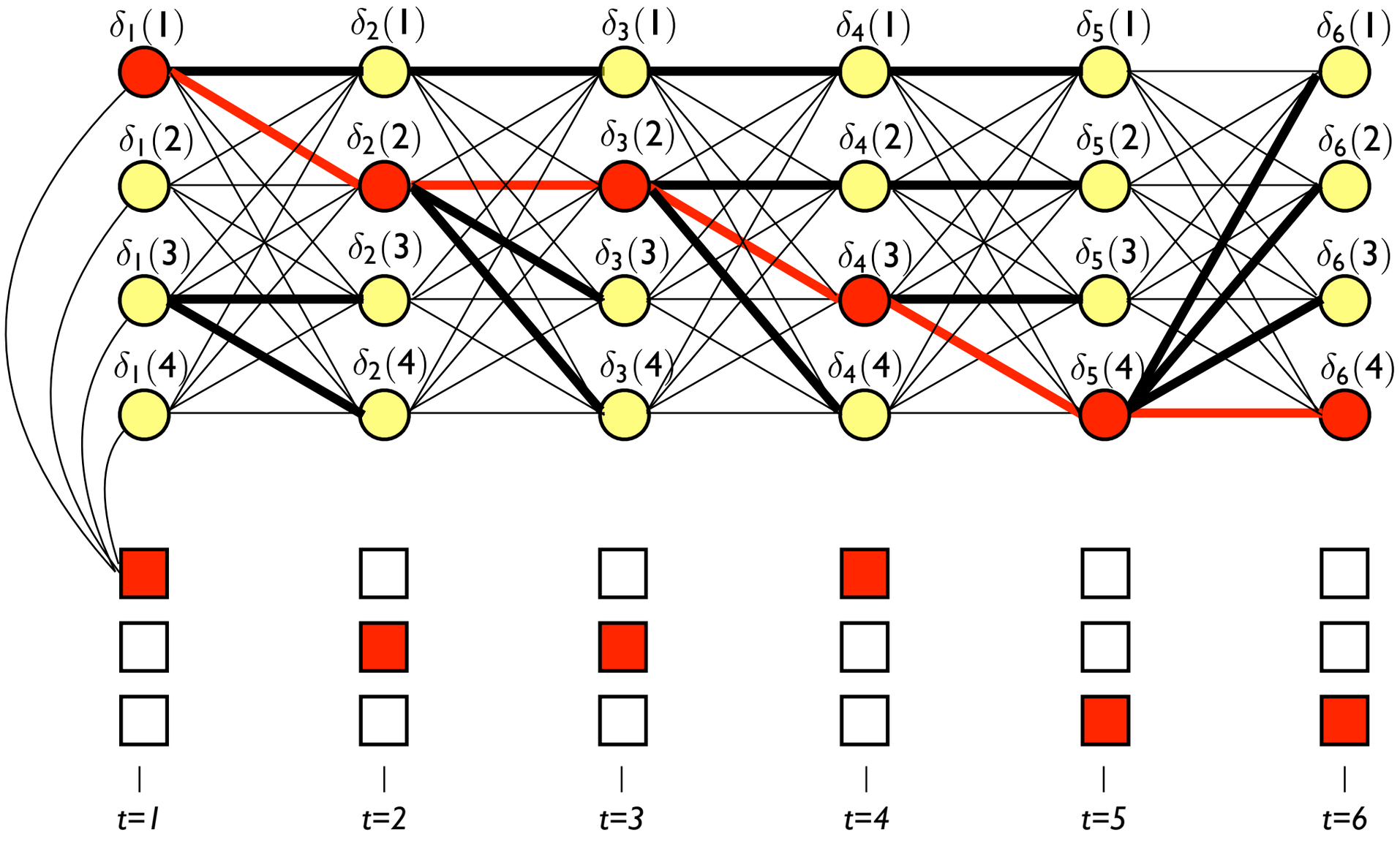
$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Termination:

$$P^* = \max_i \delta_T(i)$$

$$q_T^* = \arg\max_i \delta_T(i)$$

Viterbi algorithm



$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Backtracking: $q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \ldots, 1$

Viterbi algorithm at a glance

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1, q_2, \ldots, q_t = S_i, O_1, O_2, \ldots, O_t)$$

Initialization:
$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_1) \\ \psi_1(i) &= 0 \end{aligned}$$

Recursion:
$$\begin{aligned} \delta_t(j) &= \max_i \delta_{t-1}(i) a_{ij} b_j(O_t) \\ \psi_t(j) &= \arg\max_i \delta_{t-1}(i) a_{ij} \end{aligned}$$

Termination:
$$\begin{aligned} P^* &= \max_i \delta_T(i) \\ q_T^* &= \arg\max_i \delta_T(i) \end{aligned}$$

Backtracking: $q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1, T-2, \ldots, 1$

Problem 3: estimation of model parameters

- The objective is to estimate the model parameters $\lambda = (A, B, \pi)$ from a series of observed sequences $\{O^{(d)}\}$.
- The most classical approach to this is the *Baum-Welch algorithm*. The underlying principle is a maximum likelihood approach, i.e. we seek the model that maximizes the probabilities of observation given the model:

$$\lambda^* = \arg\max_{\lambda} P(\{O^{(d)}\}|\lambda)$$

- There is no analytical solution to this problem. The strategy is to start with chosen model parameters and then to adapt them in an iterative procedure until convergence is reached.
- For simplicity, we only consider one single observed sequence. It will become obvious that the procedure can be easily extended to a set of observations.

Forward-backward procedure

- The forward procedure provides us with a recursive algorithm to calculate:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$$

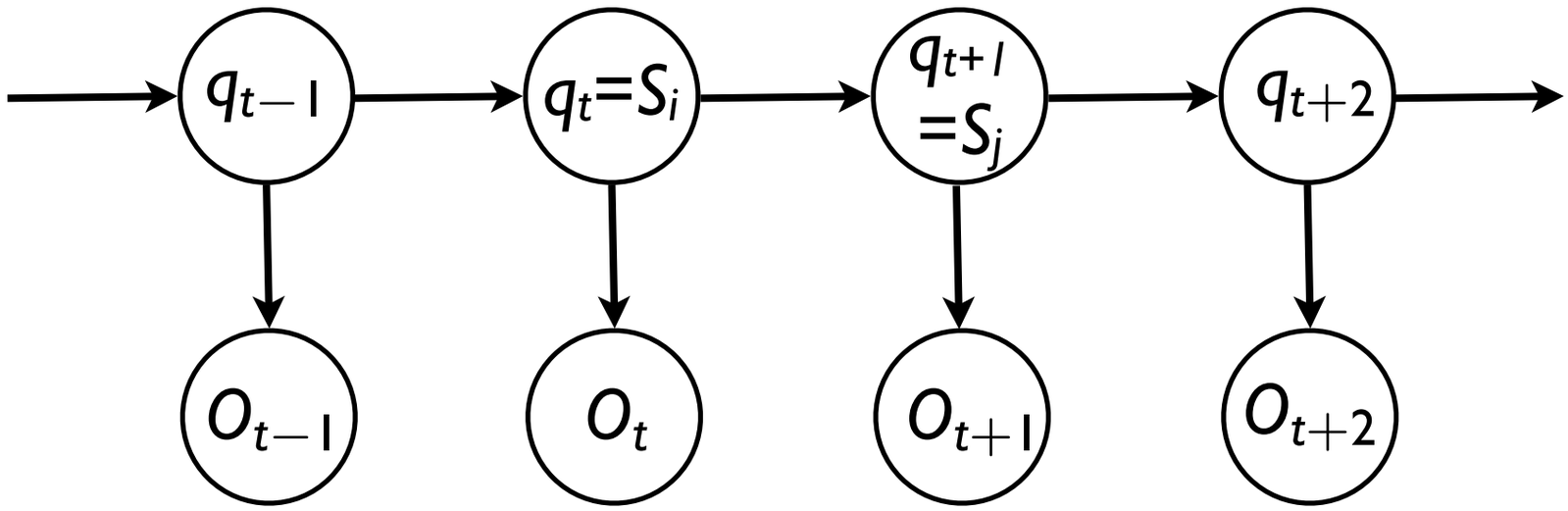
- The backward procedure is a very similar recursive procedure that allows us to calculate:

$$\beta_t(i) = P(O_{t+1}O_{t+2}\ldots O_T|q_t = S_i, \lambda)$$

Initialization: $\beta_T(i) = 1$

Recursion: $\beta_t(i) = \sum_j a_{ij}b_j(O_{t+1})\beta_{t+1}(j)$

Recursion of the backward procedure



$$
\begin{aligned}
\beta_t(i) &= \sum_j P(O_{t+1}, \ldots, O_T, q_{t+1} = S_j | q_t = S_i) \\
&= \sum_j P(q_{t+1} = S_j | q_t = S_i) P(O_{t+1}, \ldots, O_T | q_{t+1} = S_j, q_t = S_i) \\
&= \sum_j a_{ij} P(O_{t+1}, \ldots, O_T | q_{t+1} = S_j) \\
&= \sum_j a_{ij} P(O_{t+1} | q_{t+1} = S_j) P(O_{t+2} \ldots, O_T | q_{t+1} = S_j) \\
&= \sum_j a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)
\end{aligned}
$$

Forward-backward procedure

- The forward procedure provides us with a recursive algorithm to calculate:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = S_i|\lambda)$$

- The backward procedure is a very similar recursive procedure that allows us to calculate:

$$\beta_t(i) = P(O_{t+1}O_{t+2}\ldots O_T|q_t = S_i, \lambda)$$

- The product of these two probabilities is:

$$\begin{aligned}
\alpha_t(i)\beta_t(i) &= P(O_1 \ldots O_t, q_t = S_i)P(O_{t+1}O_{t+2}\ldots O_T|q_t = S_i) \\
&= P(O_1 \ldots O_t|q_t = S_i)P(q_t = S_i)P(O_{t+1}O_{t+2}\ldots O_T|q_t = S_i) \\
&= P(O|q_t = S_i)P(q_t = S_i) \\
&= P(O, q_t = S_i)
\end{aligned}$$

- From this, we can now find the probability of being in state S_i at time t given an observed sequence O:

$$\begin{aligned}
\gamma_t(i) &= P(q_t = S_i|O,\lambda) \\
&= \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \\
&= \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}
\end{aligned}$$

- We can also find the joint probability of being in state S_i at time t and S_j at time $t+1$, given the observed sequence O and the model:

$$\begin{aligned}
\xi_t(i,j) &= P(q_t = S_i, q_{t+1} = S_j|O,\lambda) \\
&= \frac{P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} \\
&= \frac{P(O_1 \ldots O_t, q_t = S_i, q_{t+1} = S_j, O_{t+1} \ldots O_T|\lambda)}{\sum_{i=1}^{N}\sum_{j=1}^{N} P(q_t = S_i, q_{t+1} = S_j, O|\lambda)}) \\
&= \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}
\end{aligned}$$

Baum-Welch algorithm

- To summarize, we have efficient ways to calculate two important probabilities for a given model:

$$\gamma_t(i) = P(q_t = S_i|O, \lambda) \quad (1)$$

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|O, \lambda) \quad (2)$$

- From this, it is easy to see that $\sum_{t=1}^{T} \gamma_t(i)$ corresponds to the expected number of times that S_i is visited, and with this also to the expected number of times of transitions from S_i, while $\sum_{t=1}^{T} \xi_t(i,j)$ is the expected number of transitions from S_i to S_j.

- With this we can now calculate estimations for:

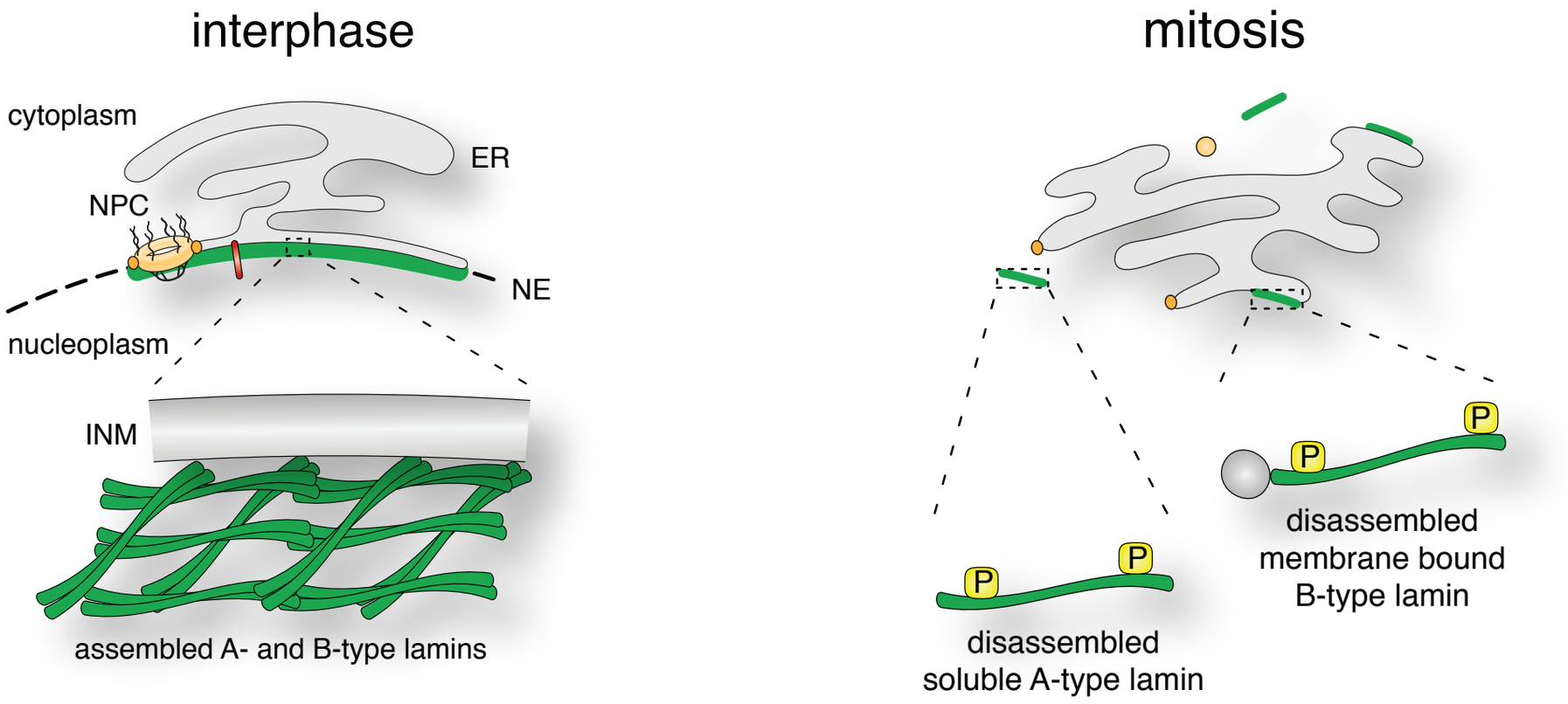
  1. The initial probabilities: $\widehat{\pi}_i = \gamma_1(i)$
  2. The transition probabilities $\widehat{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)}$
  3. The emission probabilities $\widehat{b}_i(k) = \frac{\sum_{t=1, o_t=v_k}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)}$

Baum-Welch algorithm

- We start with a chosen model (parameters are set to random values or correspond to some assumptions).
- Then, we calculate the probabilities $\gamma_t(i)$ and $\xi_t(i,j)$, conditioned on the model.
- With these probabilities, we can now calculate the new model parameters.
- This is iterated until we reach convergence.

Application

Lamin Disassembly during Nuclear Envelope Breakdown (NEBD)

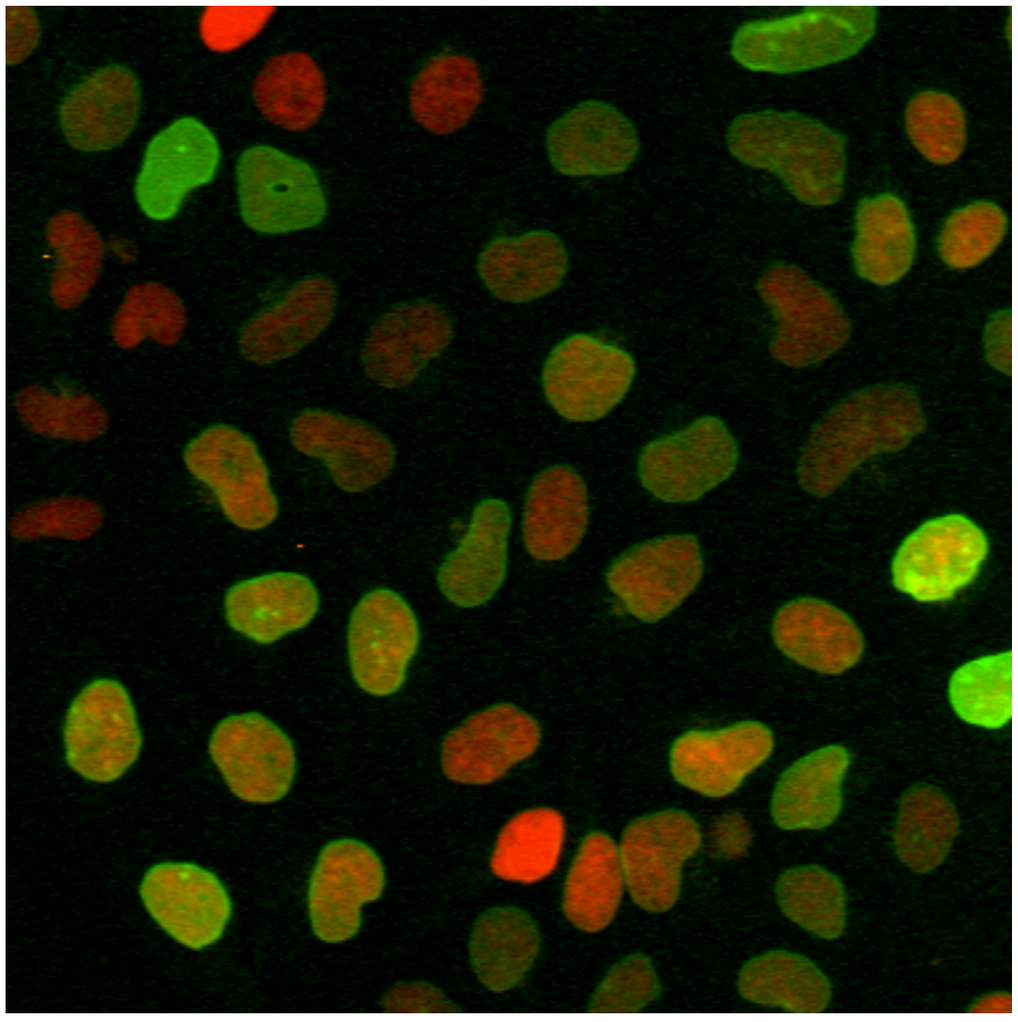


How is the disassembly of the nuclear lamina regulated?

Candidates known to phosphorylate Lamin: CDK1, PKC

Moritz Mall

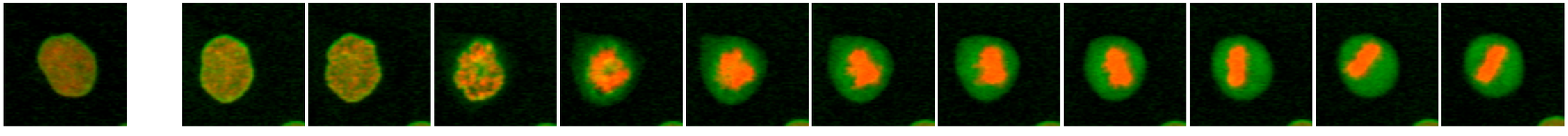
Assaying the disassembly of the nuclear lamina by life cell imaging



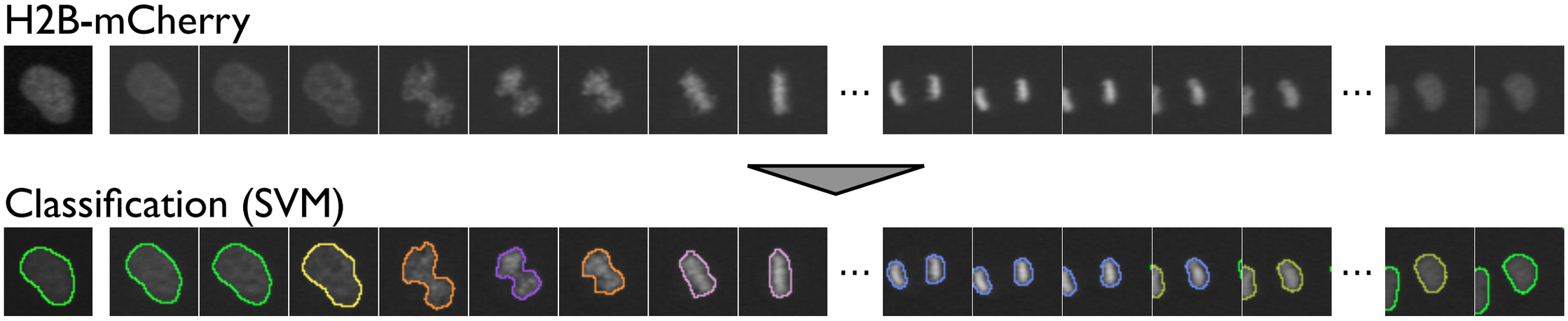
negative control, background subtracted

Assay:

- Cell line stably expressing H2B-mCherry and eGFP-LAMINB1.
- coated 8-well labteks.
- 10x widefield microscope (Olympus)
- time-lapse: 2 min



Monitoring of mitotic progression

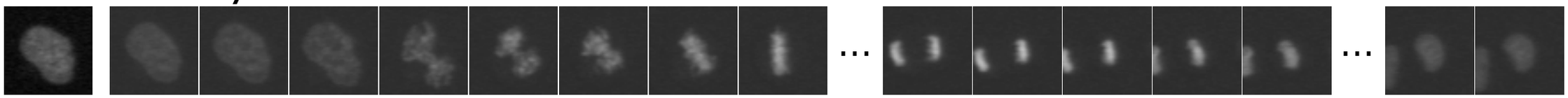


Measure mitotic phase lengths:

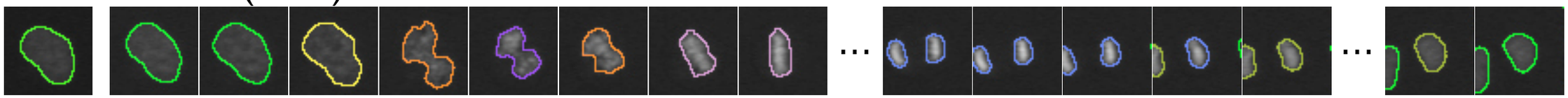
- Classification of each single nucleus at each single time point
- Follow individual cells by tracking
- This allows us to measure directly the lengths of the mitotic phases.
- Problem: a single classification error can invalidate an entire trajectory, i.e. even though classification accuracy is high for each nucleus, the number of correct tracks is low.

Quantification of the lamina disassembly

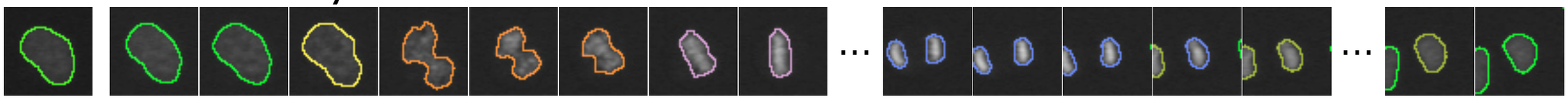
H2B-mCherry



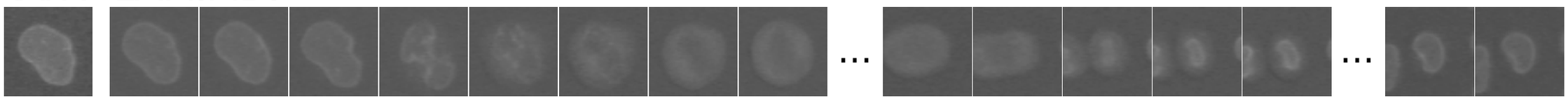
Classification (SVM)



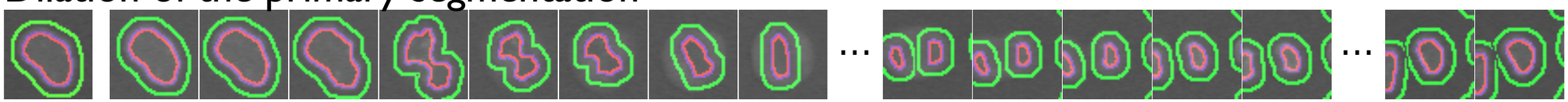
Error Correction by HMM



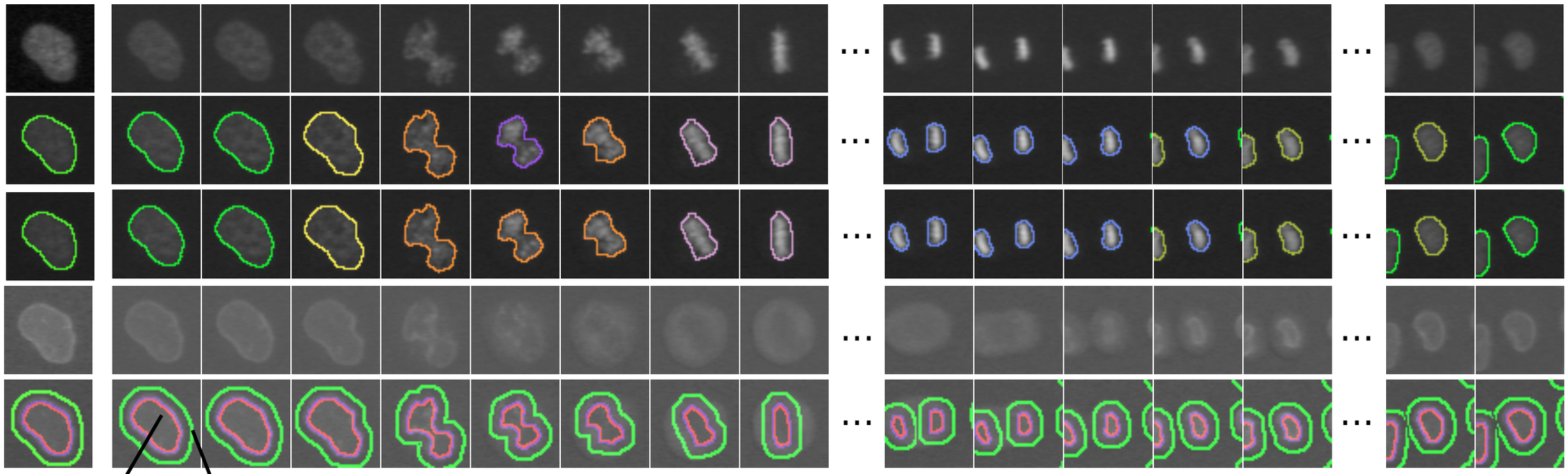
eGFP-LAMINB1



Dilation of the primary segmentation



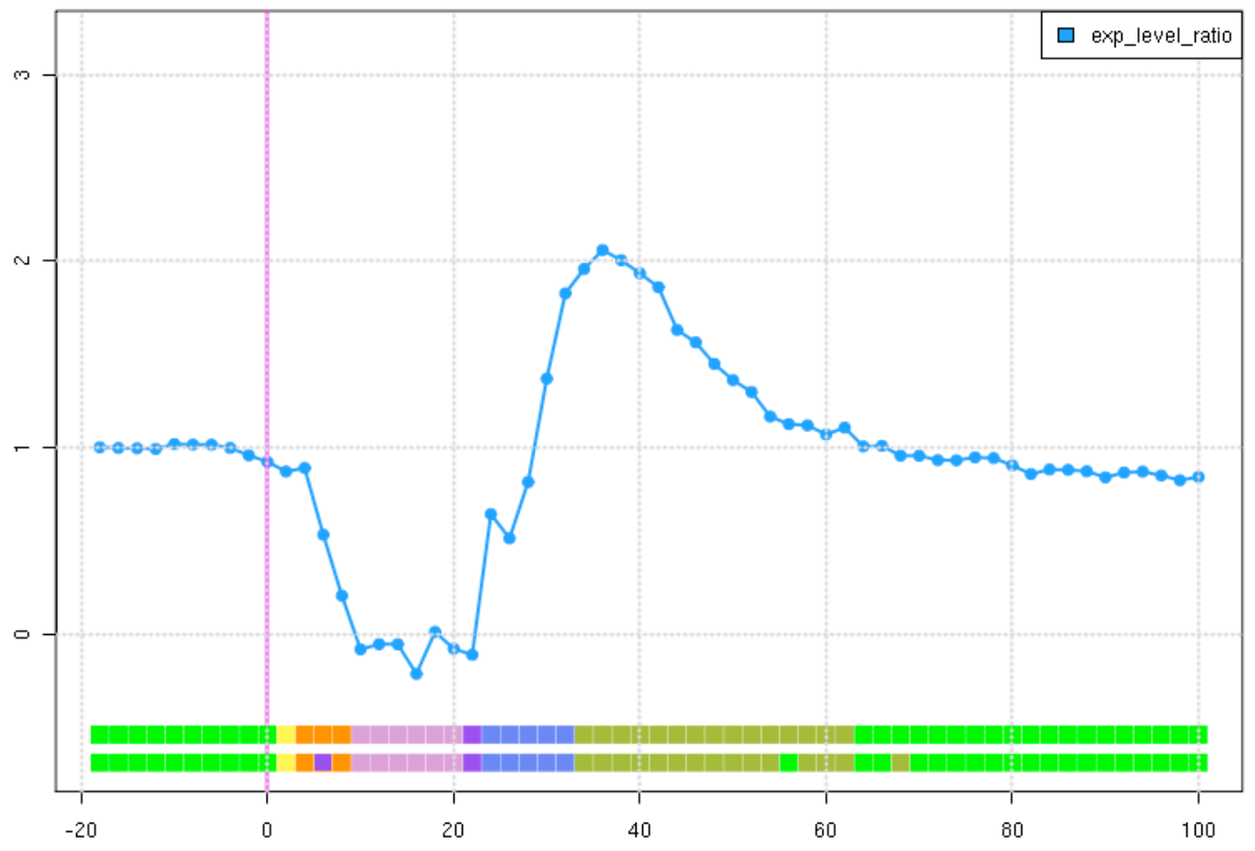
Quantification of the lamina disassembly



$IN(t)$ $OUT(t)$

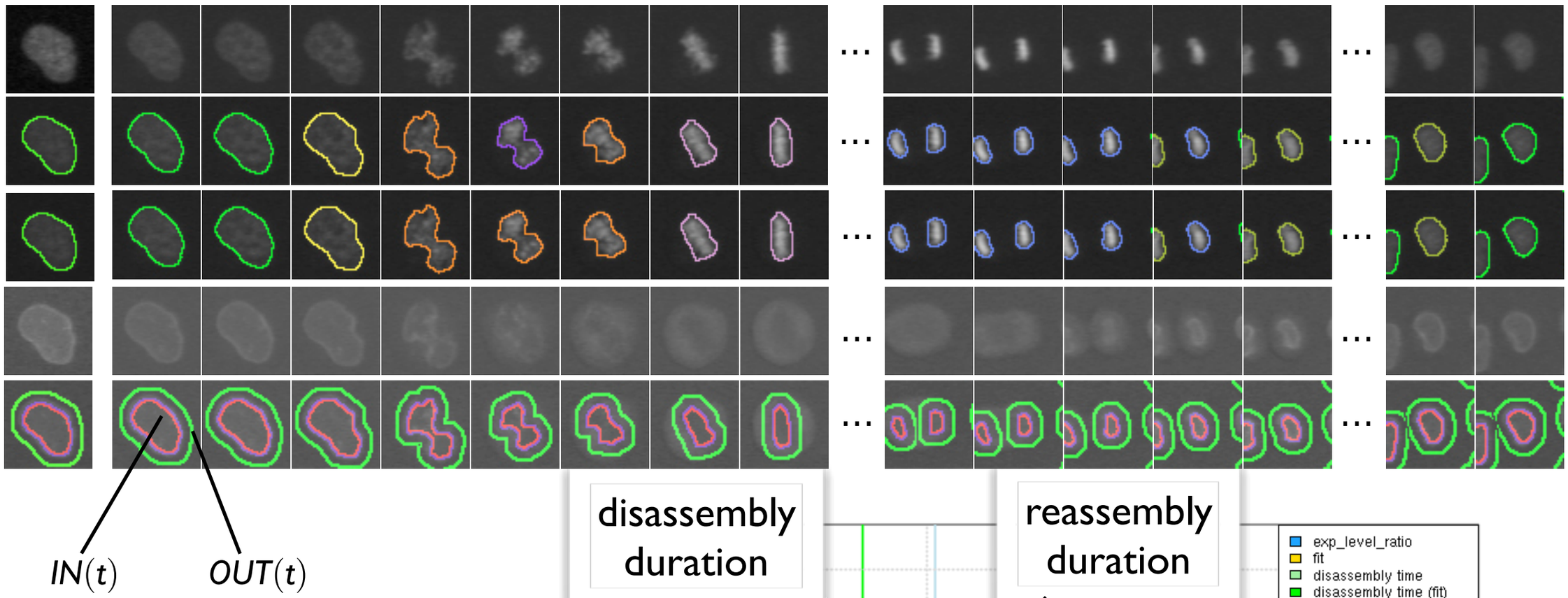
Quantification of depolymerization:

$$x(t) = \frac{IN(t) - OUT(t)}{avg(IN(\tau) - OUT(\tau))|_{\text{Interphase}}}$$



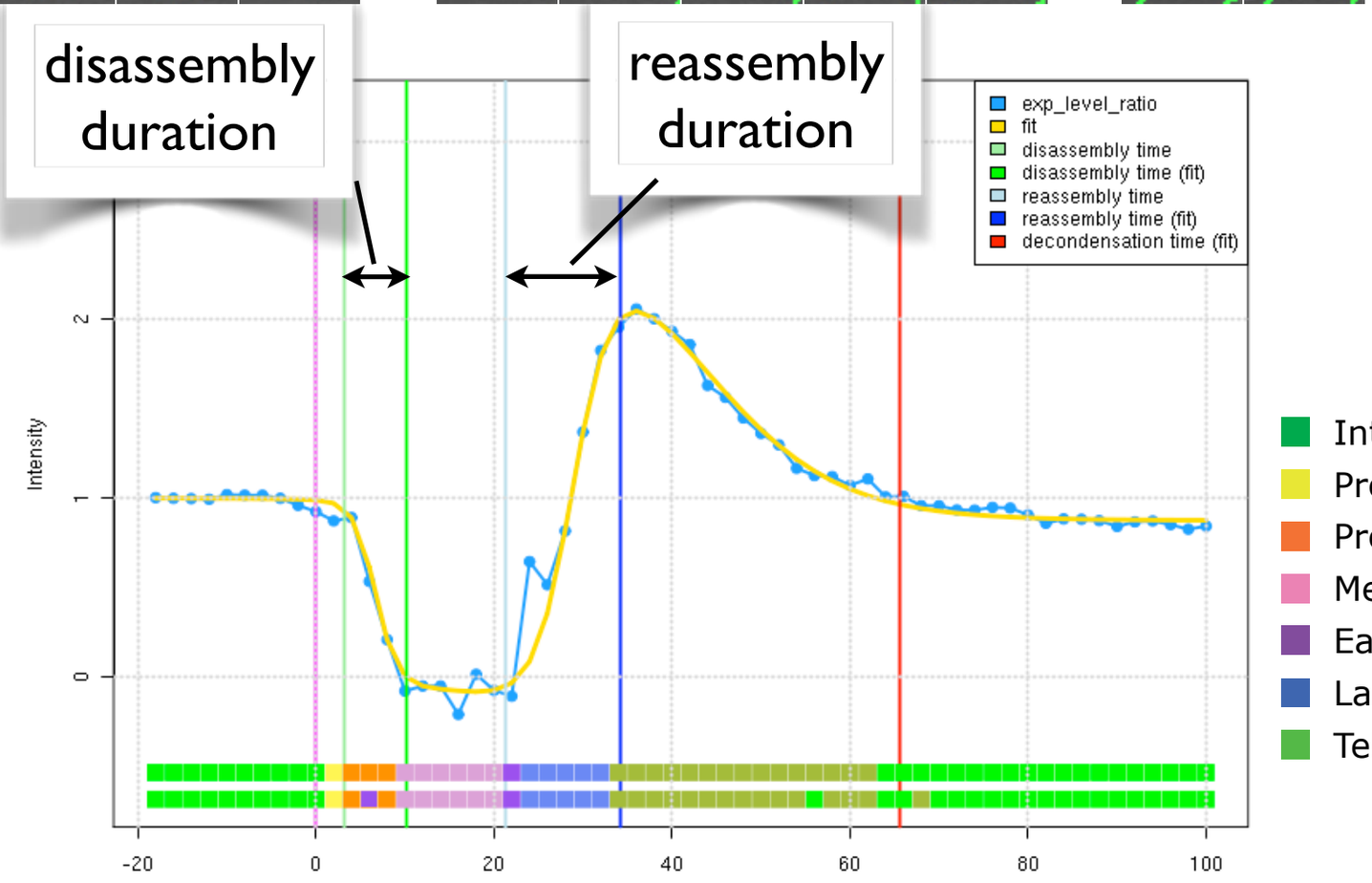
- Interphase
- Prophase
- Prometaphase
- Metaphase
- Early anaphase
- Late anaphase
- Telophase

Quantification of the lamina disassembly

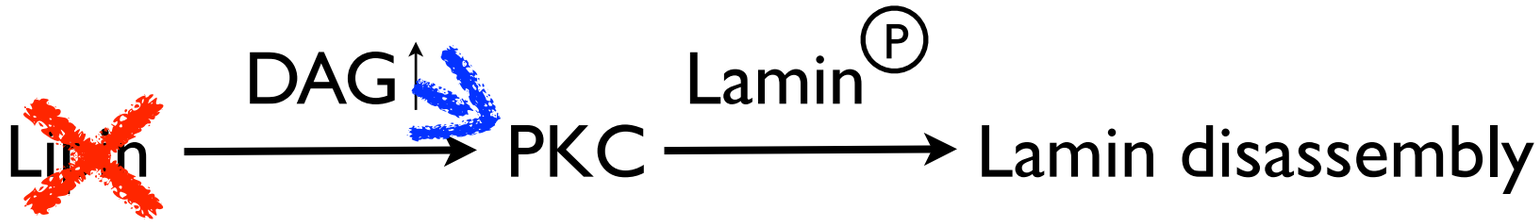


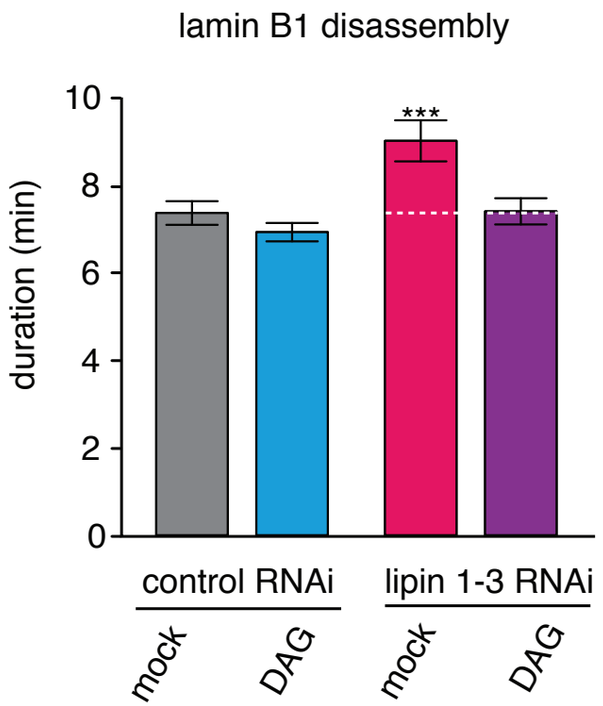
Quantification of depolymerization:

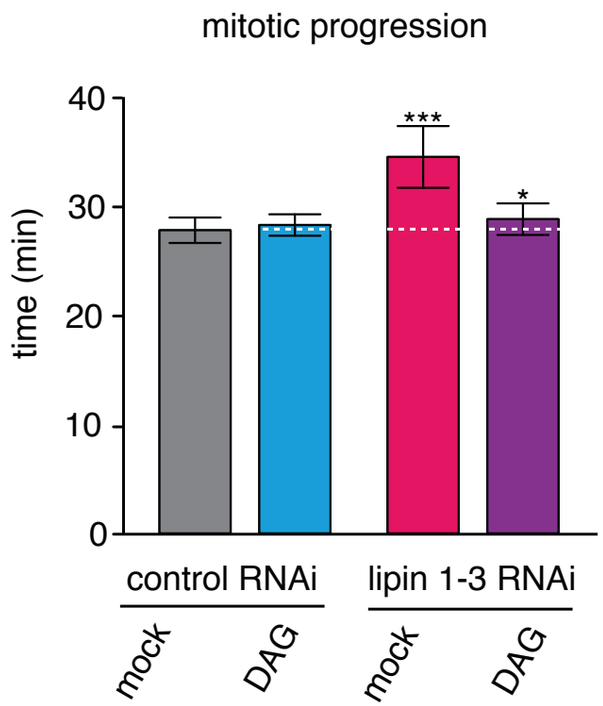
$$x(t) = \frac{IN(t) - OUT(t)}{avg(IN(\tau) - OUT(\tau))|_{\text{Interphase}}}$$

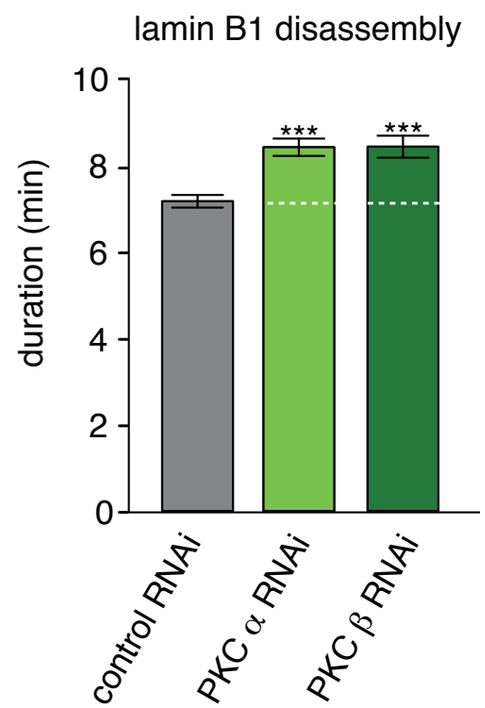


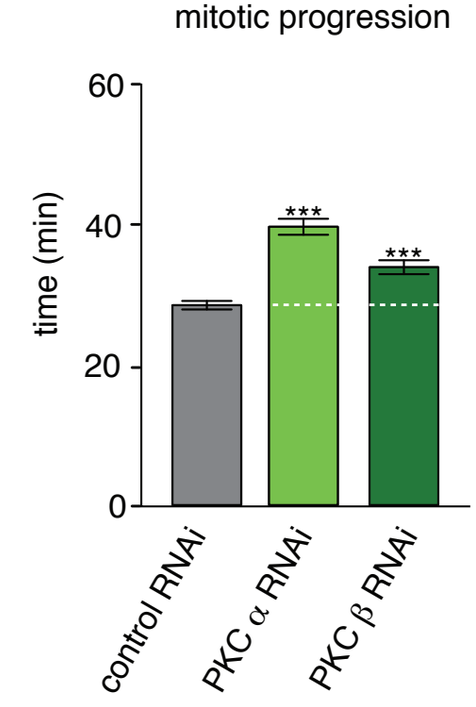
Lipins trigger mitotic progression and lamin B1 disassembly

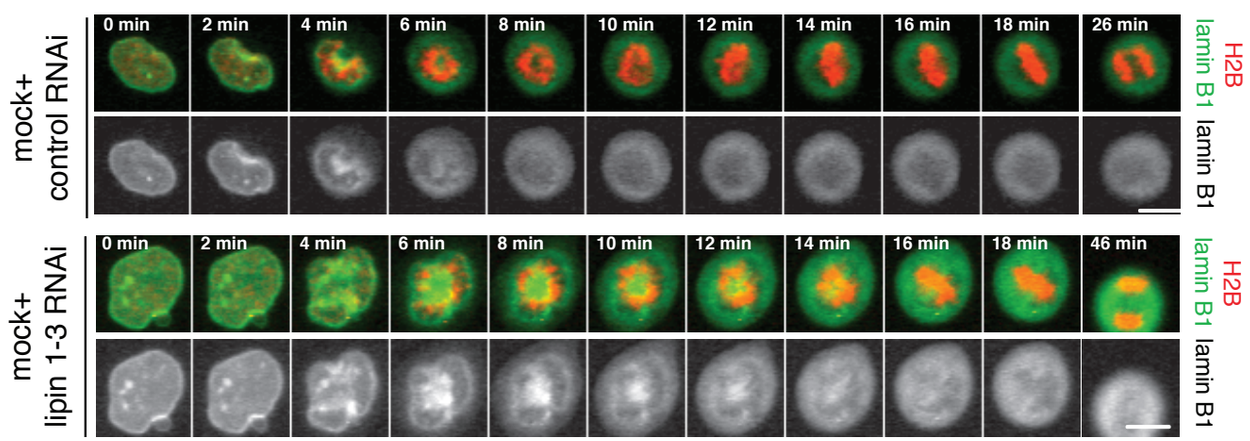


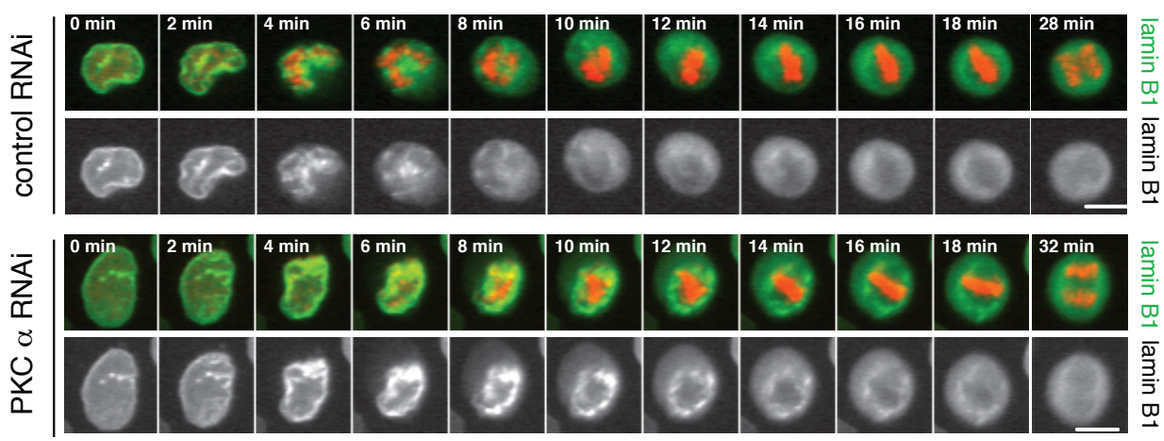




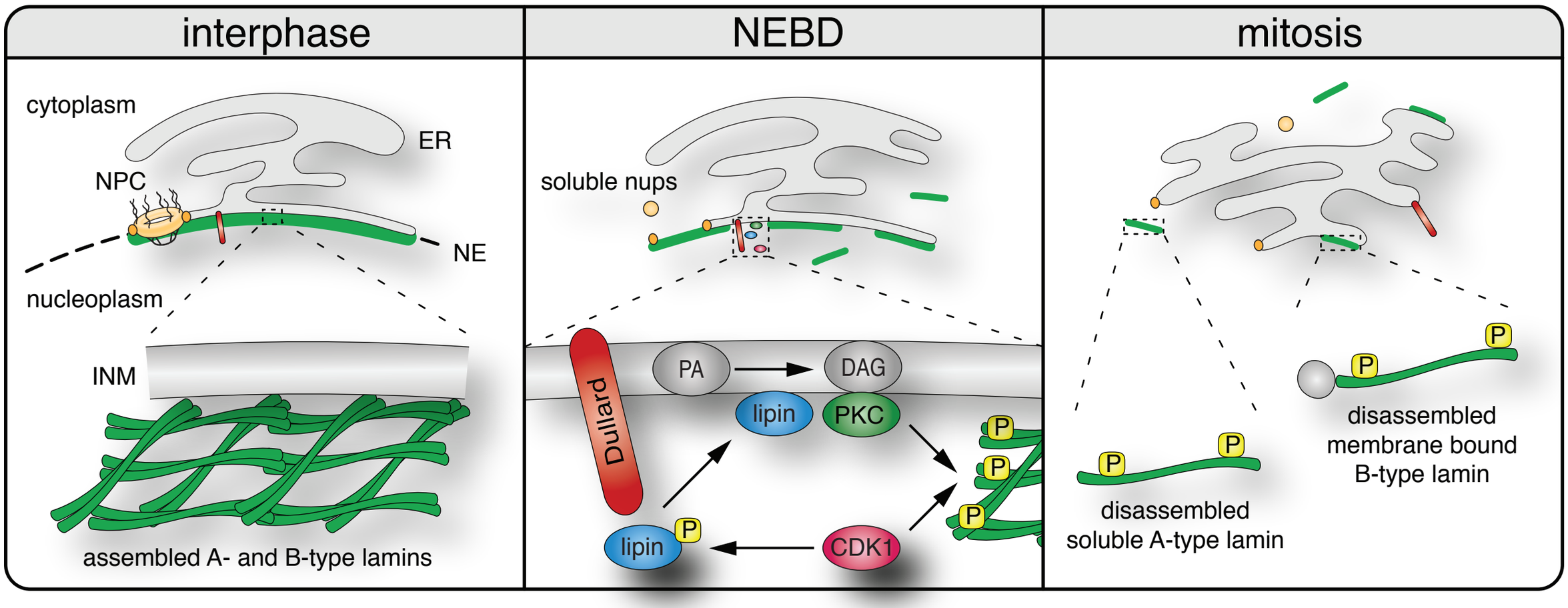








Proposed Model for the regulation of lamin disassembly at mitotic onset



Mall, Walter et al., Journal of Cell Biology. Sep 2012.

Conclusions

- Hidden Markov Models provide a statistical framework to deal with sequential data.

- 3 problems: evaluation, decoding, learning

- Applications in Bioinformatics:
  - Gene prediction
  - Bioimaging : sequence of phenotypes