Deterministic Dynamic Programming

- Recall that one key component of DP is the dynamics function:

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_k), \forall k \in \{0, ..., N-1\}$$

- In the **deterministic** setting, we will assume that there is no uncertainty and the DP problem can written as a “single-shot” optimization problem:

$$J_0^*(x_0) = \min_{u_0,\dots,u_{N-1}} g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k)$$

$$\begin{aligned} \text{s.t.: } & x_{k+1} = f_k(x_k, u_k), \forall k \in \{0, ..., N-1\} \\ & x_k \in \mathcal{X}_k, \forall k \in \{0, ..., N-1\} \\ & u_k \in U_k(x_k), \forall k \in \{0, ..., N-1\} \end{aligned}$$

- This problem can be solved via optimization algorithms, such IPM, if the state-space is continuous. But what if it’s not?

Deterministic DP as a graph problem

- Let's focus on the discrete state space. Then every possible transition from states i to j can be "predicted" with certainty by:

$$j = f_k(i, u), \text{for some } u \in U_k(i)$$

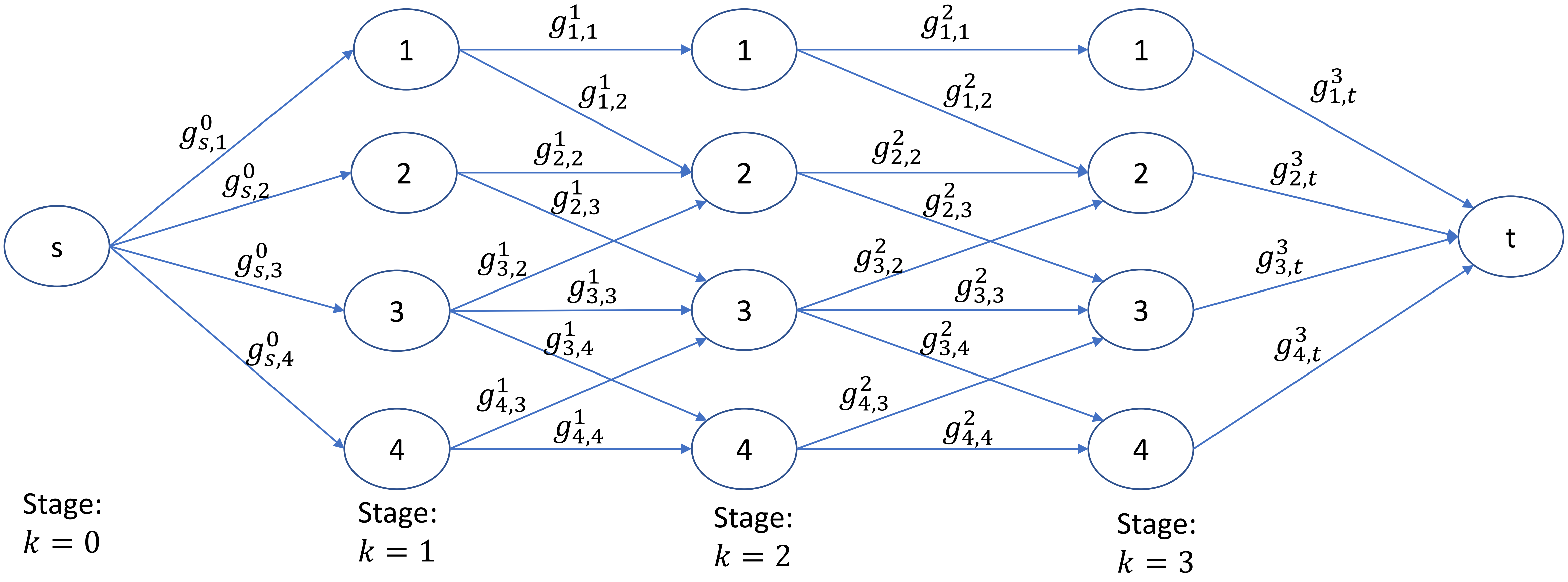
- Now, consider the following graph $G = (V, E)$:
  - V is the set of nodes: We will have one node for every possible state in every time period.
  - E is the set of arcs: We will have an arc linking node i and j if and only if there exists a control u such that the transition from i to j is possible (via the dynamics).
  - We add dummy nodes s and t to link the initial state and final state nodes, respectively.

- Moreover we can assign arc-lengths matching the associated cost:

$$g_{i,j}^k = g_k(i, u), \text{ for some } u \in U_k(i) \text{ such that } j = f_k(i, u)$$

Deterministic DP as a graph problem

- Then we obtain the following graph, for $N = 3$:



Key Properties of Deterministic DP

- This graph representation allows to make two important observations:

1. Obtaining a sequence of controls $(u_0, \dots, u_{N-1})$ is equivalent to selecting a single *path* from the source s to the terminal t. Hence, the problem of finding the sequence of controls with the smallest cost is equivalent of finding the path with the smallest weight, a.k.a. the *Shortest Path.*

2. Since the DP is deterministic, we can always map the optimal policy $\pi^* = \{\mu_0^*, \dots, \mu_{N-1}^*\}$ to a sequence $(u_0^*, \dots, u_{N-1}^*)$, such that $u_i^* = \mu_i^*(x_i)$. Hence, the closed-loop optimal policy and the open-loop optimal control sequence are equivalent.

Deterministic DP Algorithm

- Then, the usual **backwards** DP algorithm is given by:

$$J_N(i) = g_{i,t}^N, \forall i \in S_N$$

$$J_k(i) = \min_{j \in S_{k+1}} \{g_{i,j}^k + J_{k+1}(j)\}, \forall i \in S_k, k \in \{0, ..., N-1\}$$

- But now, we can also define the **forward** DP algorithm:

$$\tilde{J}_N(j) = g_{s,j}^0, \forall j \in S_1$$

$$\tilde{J}_k(j) = \min_{i \in S_{N-k}} \{g_{i,j}^{N-k} + \tilde{J}_{k+1}(i)\}, \forall j \in S_{N-k+1}, k \in \{1, ..., N-1\}$$

$$\tilde{J}_0(t) = \min_{i \in S_N} \{g_{i,t}^N + \tilde{J}_1(i)\}$$

- And due to the 2nd observation, we can conclude that: $J_0(s) = \tilde{J}_0(t)$

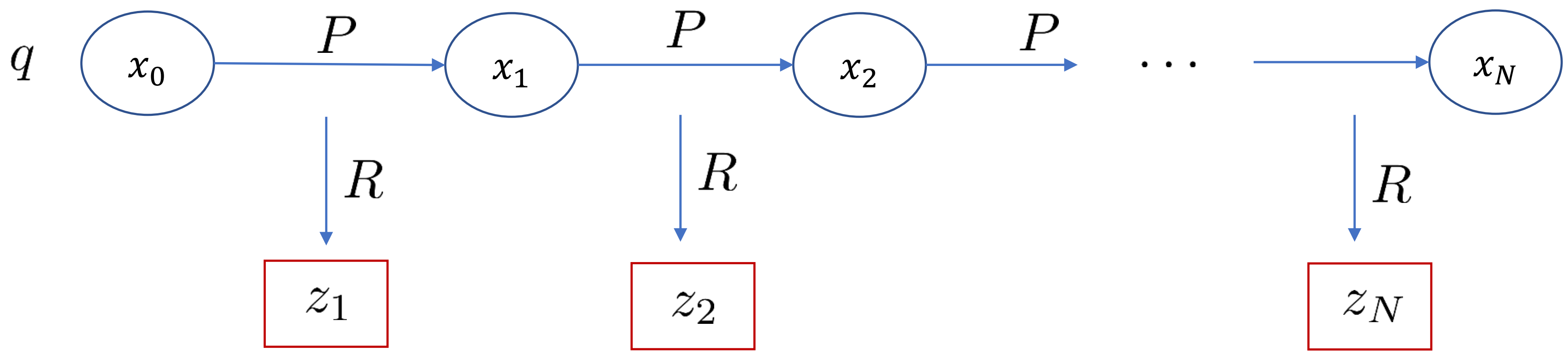
Hidden Markov Models (HMM)

- Consider a Markov Chain with finite number states and transition probabilities:

$$p_{i,j} = \Pr(x_{k+1} = j | x_k = i), \forall k = \{0, ..., N-1\}$$

$$q_i = \Pr(x_0 = i)$$

- A Hidden Markov Model (HMM) can be view schematically as follows:



- Where:

$$r(z|i,j) = r(z_{k+1} = z, x_k = i, x_{k+1} = j)$$

Hidden Markov Models (HMM)

- Suppose we have obtained our imperfectly observed sequence $Z_N = (z_1, \dots, z_N)$. Our goal select a sequence $(\hat{x}_0, \dots, \hat{x}_N)$ that maximizes the following probability:

$$\Pr((x_0, ..., x_N)|(z_1, ..., z_N)) = \Pr(X_N|Z_N)$$

- But it is easier to maximize the following (by Bayes Rule):

$$\Pr(X_N|Z_N) = \frac{\Pr(X_N, Z_N)}{\Pr(Z_N)}$$

- And applying the Markov Property that: $\Pr(z_k|x_{k-2}, x_{k-1}, x_k, z_{k-1}) = r(z_k|x_{k-1}, x_k)$

- We get:

$$\Pr(X_N, Z_N) = q_{x_0} \prod_{k=1}^{N} p_{x_{k-1}, x_k} r(z_k|x_{k-1}, x_k)$$

Hidden Markov Models (HMM)

- In order to simplify, we can take the logarithm and write the following (equivalent) minimization problem:

$$\min -\ln(q_{x0}) - \sum_{k=1}^{N} \ln(p_{x_{k-1},x_k} r(z_k|x_{k-1}, x_k))$$

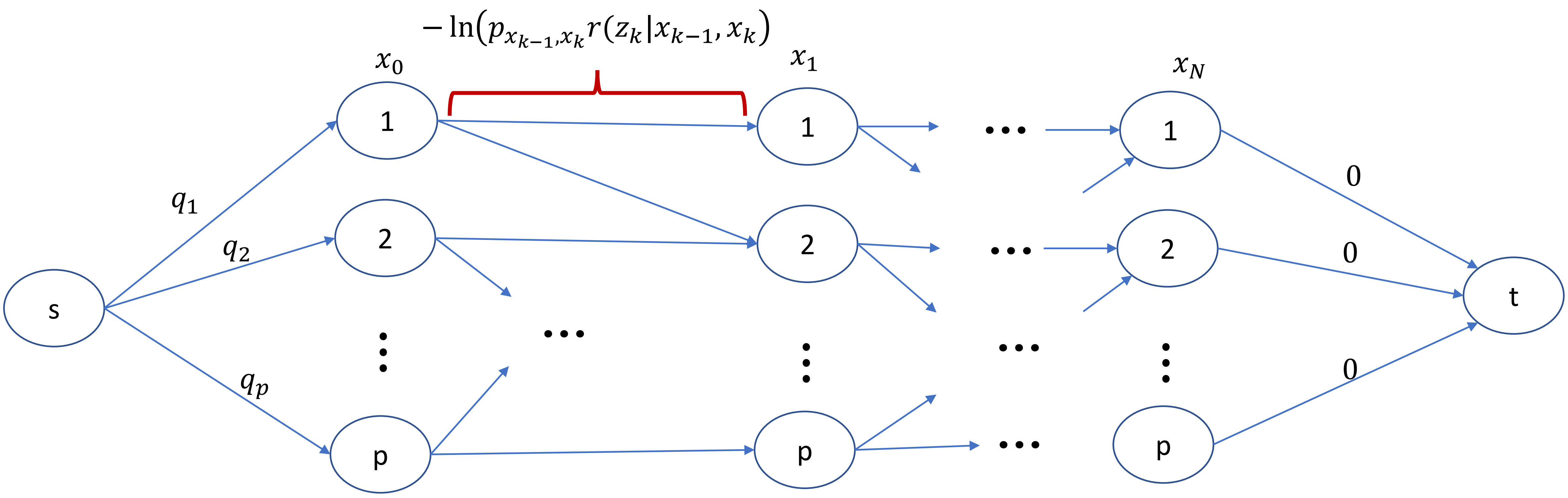
$$\text{over all possible sequences } \{x_0, x_1, ..., x_N\}$$

- And our estimate of the most likely sequence of transitions is the optimal solution of the above problem.

- And it turns out this is actually a Shortest Path Problem. The related graph is called the *Trellis Diagram*.

Hidden Markov Models (HMM)

- In fact, it is this problem:



The Viterbi Algorithm

- The famous Viterbi Algorithm is just the forward DP algorithm applied to the Trellis Diagram:

$$D_{k+1}(x_{k+1}) = \min_{\text{all } x_k \,:\, p_{x_k, x_{k+1}} > 0} \left\{ D_k(x_k) - \ln\left(p_{x_k, x_{k+1}} r(z_{k+1} | x_k, x_{k+1})\right) \right\}$$

$$D_0(x_0) = -\ln(q_{x_0})$$

- Where $D_k(x_k)$ is shortest distance from s to node x_k given the observation sequence $(z_1, \ldots, z_k)$.

- And the optimal sequence $(\hat{x}_0, \ldots, \hat{x}_N)$ corresponds to the shortest path from s to t.

Convolutional Coding Example

- Suppose we would like to send a binary data vector containing sensitive information to someone via a noisy communication channel.

- The main idea is to convert your data vector:

$$(w_1, w_2, ...),\ \ w_k \in \{0, 1\}, k \in \{1, 2, ...\}$$

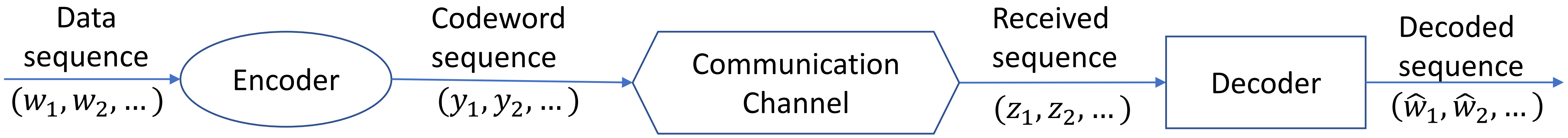
- Into a coded sequence of "words":

$$y_k = \begin{bmatrix} y_k^1 \\ \vdots \\ y_k^n \end{bmatrix},\ \ y_k^i \in \{0, 1\}, k \in \{1, 2, ...\}$$

- However we obtain the noisy transmission $(z_1, z_2, ...)$ and we have to decode it into a sequence $(\hat{w}_1, \hat{w}_2, ...)$.

Convolutional Coding Example

- Schematically we can draw:



- One way to address the encoding part is to use convolutions:

$$y_k = Cx_{k-1} + dw_k, \ \ , k \in \{1, 2, \ldots\}$$

$$x_k = Ax_{k-1} + bw_k, \ \ , k \in \{1, 2, \ldots\}, \ x_0 : \ \text{given}$$

- Where the elements of $A, \mathrm{C}, \mathrm{d}$ and b are all binary elements (0 or 1), and x_k is some embedded state space.

Convolutional Coding Example

- For example:

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, \; A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, d = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

- Suppose the initial state is $x_0 = 00$ and the data sequence is:

$$(w_1, w_2, w_3, w_4) = (1, 0, 0, 1)$$

- Then the generate state sequence is:

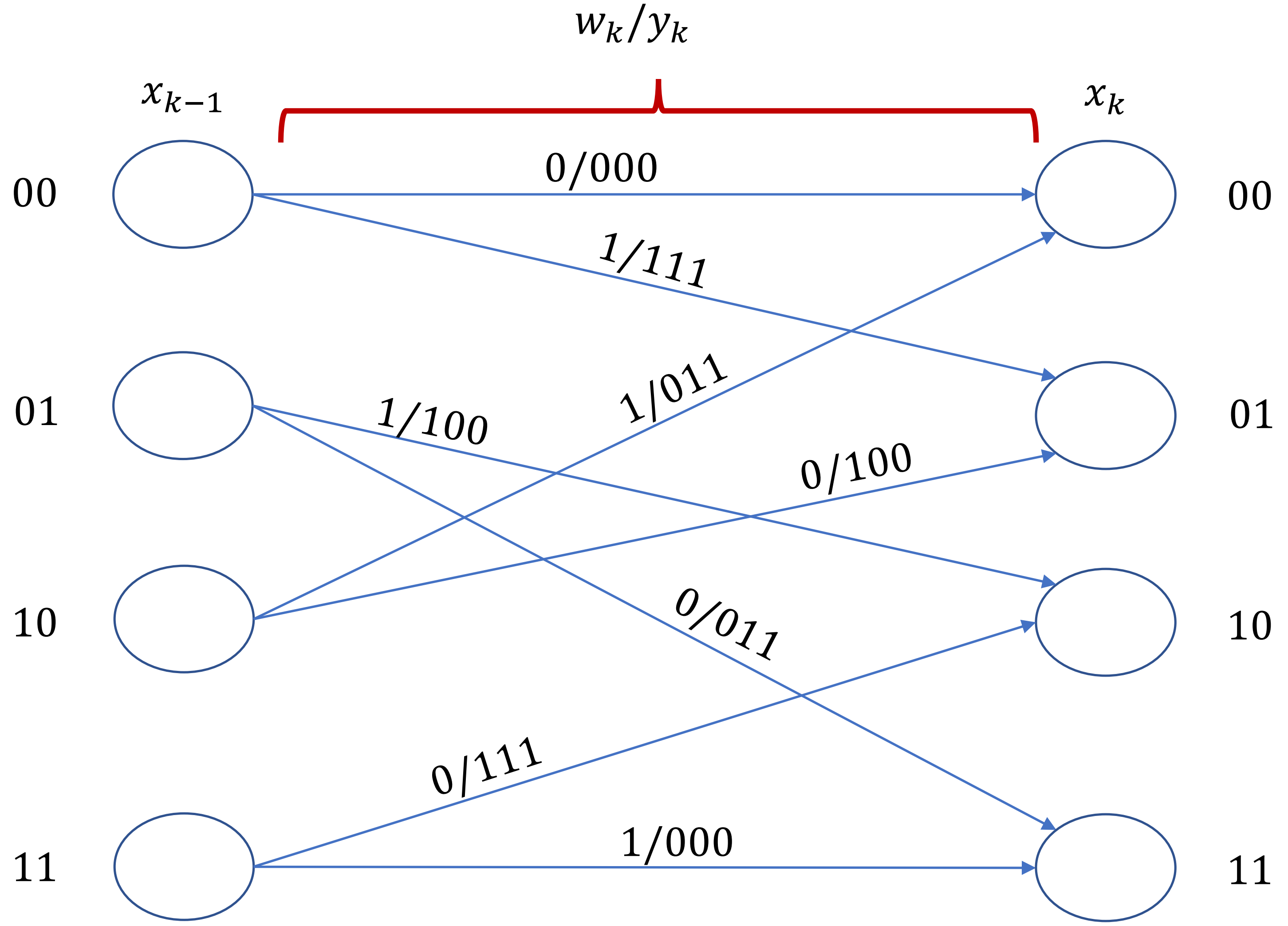
$$(x_0, x_1, x_2, x_3, x_4) = (00, 01, 11, 10, 00)$$

- And the coded sequence is:

$$(y_1, y_2, y_3, y_4) = (111, 011, 111, 011)$$

Convolutional Coding Example

- We can illustrate the encoding in the following diagram:



Convolutional Coding Example

- Similarly as before we wish to maximize:

$$\Pr(Z_N|Y_N) = \prod_{k=1}^{N} \Pr(z_k|y_k)$$

- Then, the most likely sequence $\hat{Y}_N = (\hat{y}_1, \ldots, \hat{y}_N)$ solves:

$$\Pr(Z_N|\hat{Y}_N) = \max_{Y_N} \{ \Pr(Z_N|Y_N) \}$$

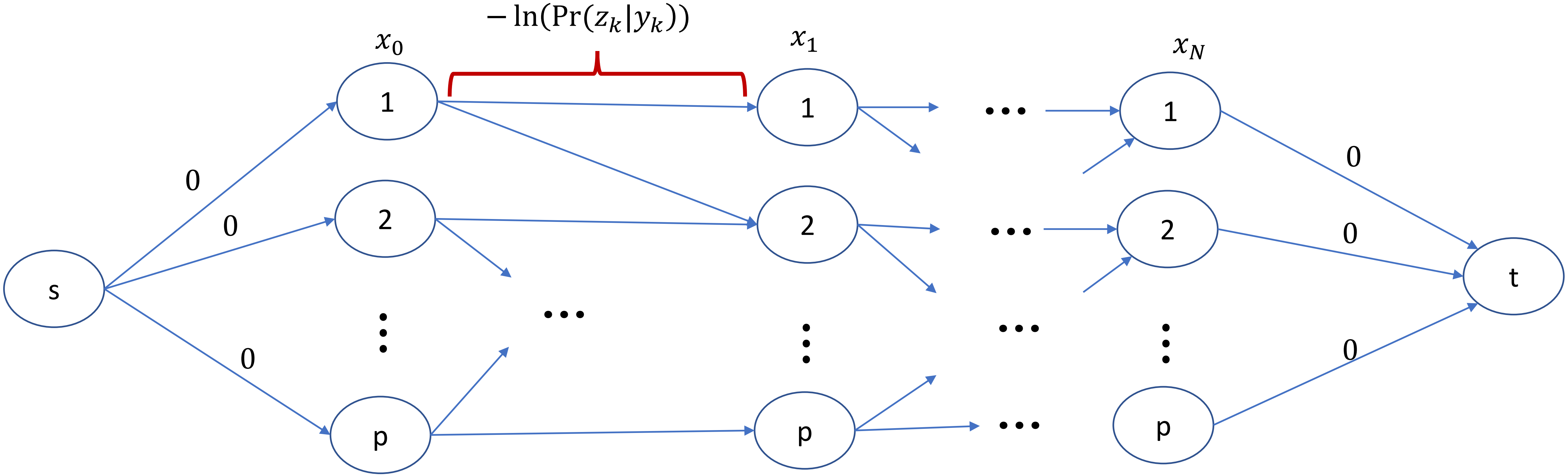
- We use the log-trick again to write:

$$\Pr(Z_N|\hat{Y}_N) = \min \{ \sum_{k=1}^{N} -\ln(\Pr(z_k|y_k)) \}$$

$$\text{over all binary sequences } (y_1, \ldots, y_N)$$

Convolutional Coding Example

- The problem then reduces to finding the Shortest Path in Trellis Diagram



Convolutional Coding Example

- Then the most likely sequence is obtained by the Viterbi Algorithm:

$$D_{k+1}(x_{k+1}) = \min_{\text{all } x_k \,:\, (x_x, x_{k+1}) \text{ is an arc}} \left\{ D_k(x_k) - \ln(\Pr(z_{k+1}|y_{k+1})) \right\}$$

- The shortest path $(\hat{x}_0, \dots, \hat{x}_N)$ can then be easily mapped to the corresponding data sequence $(\widehat{w}_0, \dots, \widehat{w}_N)$, which is the actual *decoded sequence*.