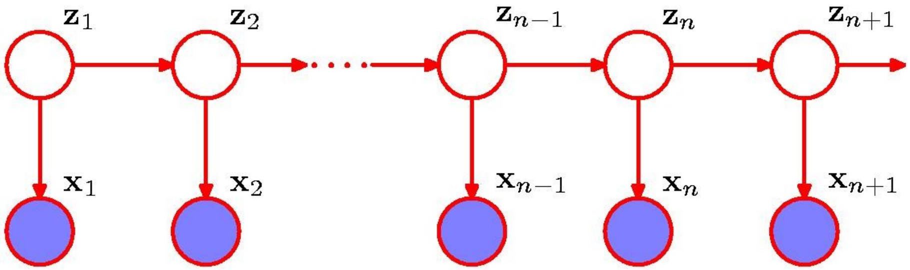
Hidden Markov Models

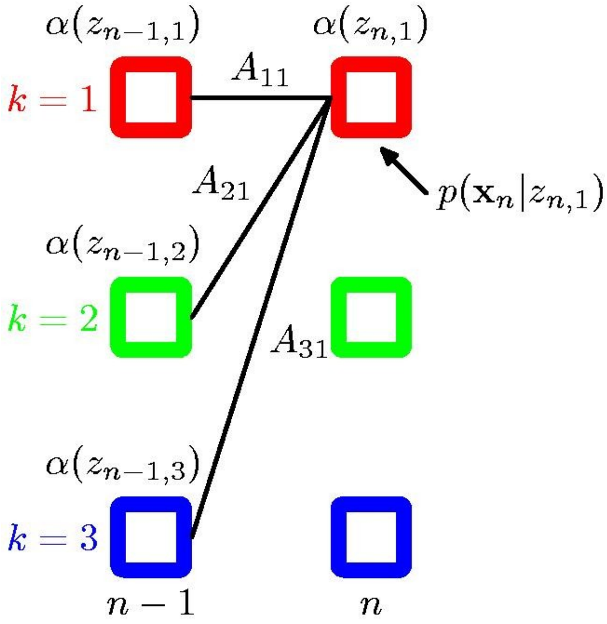
Implementing the forward-, backward- and Viterbi-algorithms using log-space and scaling



The Viterbi Algorithm

$\omega(\mathbf{z}_n)$ is the probability of the most likely sequence of states $\mathbf{z}_1,\ldots,\mathbf{z}_n$ generating the observations $\mathbf{x}_1,\ldots,\mathbf{x}_n$

$$\omega(\mathbf{z}_n) = \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}} p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{z}_1,\ldots,\mathbf{z}_n)$$



Recursion:

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\max_{\mathbf{z}_{n-1}}\omega(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

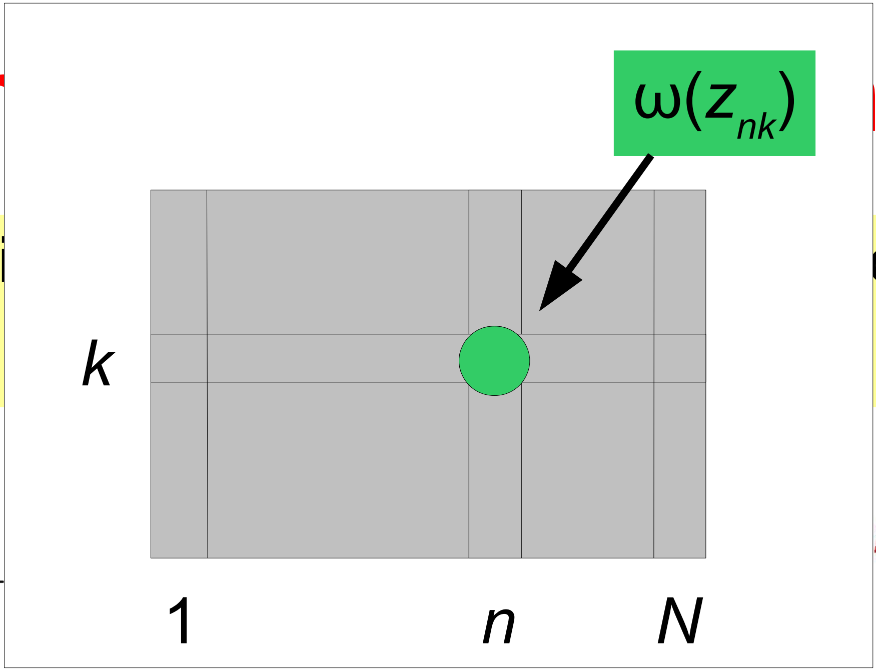
Basis:

$$\omega(\mathbf{z}_1) = p(\mathbf{x}_1,\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K}\{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}}$$

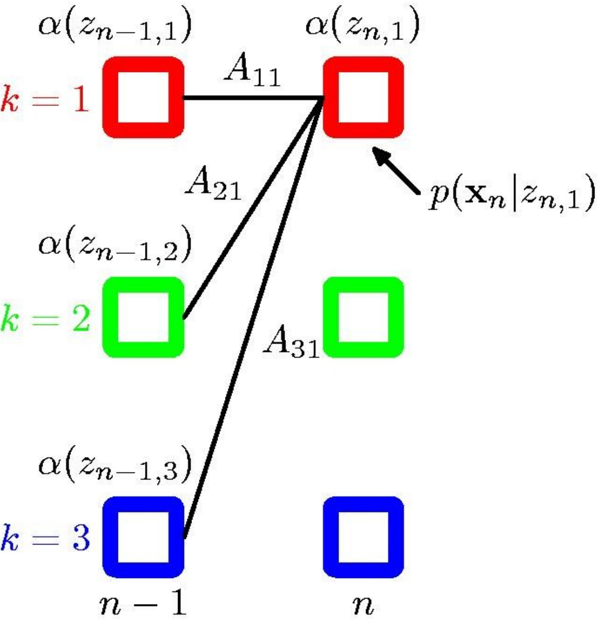
Takes time O(K^2N) and space O(KN) using memorization

Th m

$\omega(\mathbf{z}_n)$ is the probabili of states $\mathbf{z}_1,\ldots,\mathbf{z}_n$ generating the obse



$$\omega(\mathbf{z}_n) \equiv \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-}}$$



Recursion:

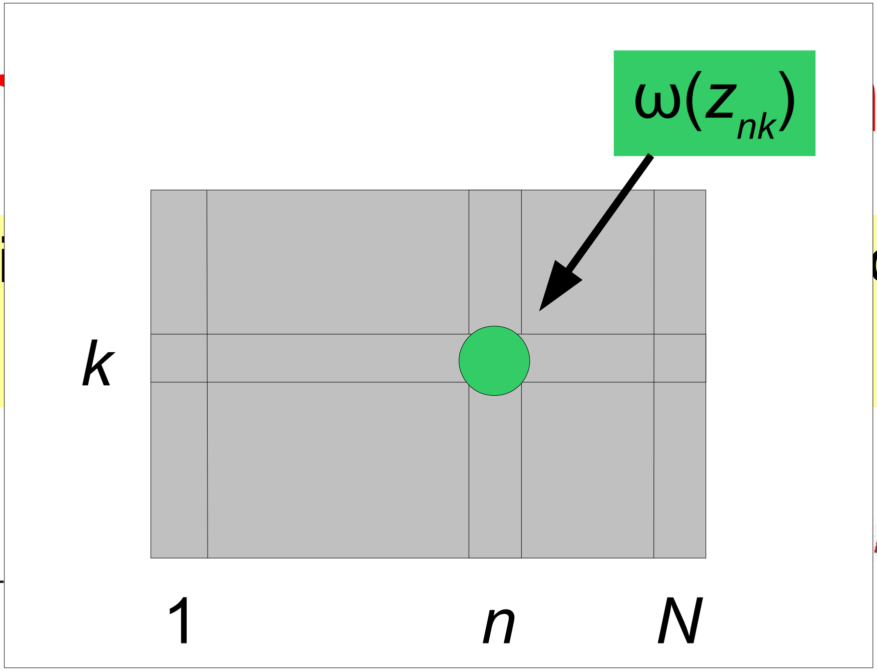
$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\max_{\mathbf{z}_{n-1}}\omega(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Basis:

$$\omega(\mathbf{z}_1) = p(\mathbf{x}_1,\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K}\{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}}$$

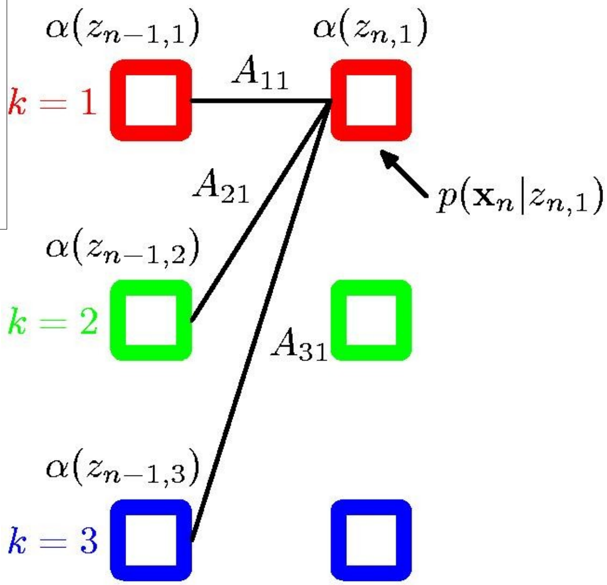
Takes time $O(K^2N)$ and space $O(KN)$ using memorization

Th m



$\omega(\mathbf{z}_n)$ is the probabili of states $\mathbf{z}_1,\ldots,\mathbf{z}_n$ generating the obse

$$\omega(\mathbf{z}_n) \equiv \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-}}$$



Recursion:

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\max_{\mathbf{z}_{n-1}}\omega(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Basis:

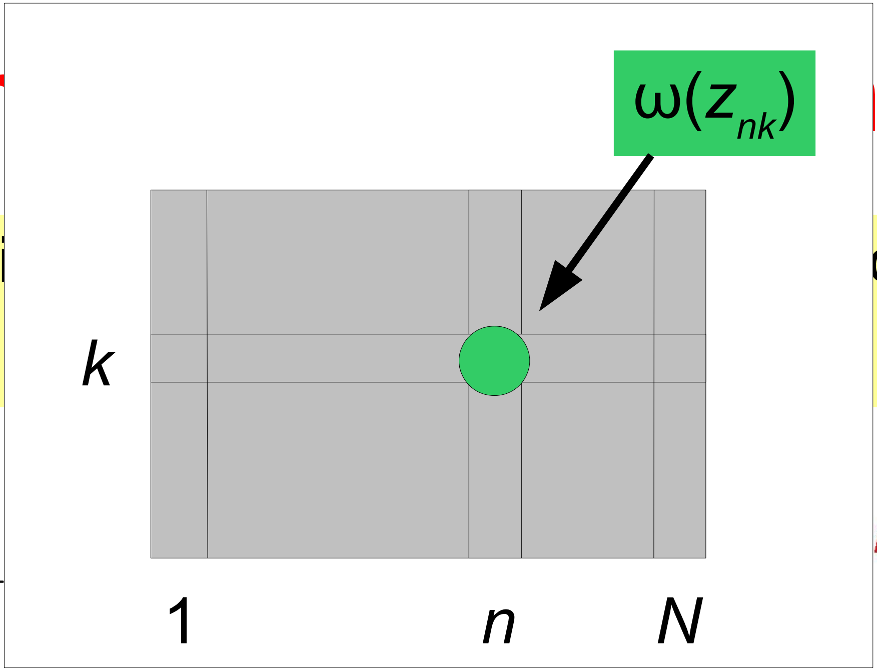
$\omega(\mathbf{z}$

Problem: The values ω(z_{nk}) can come very close to zero, by multiplying them we potentially exceed the precision of double precision floating points

$n-1$

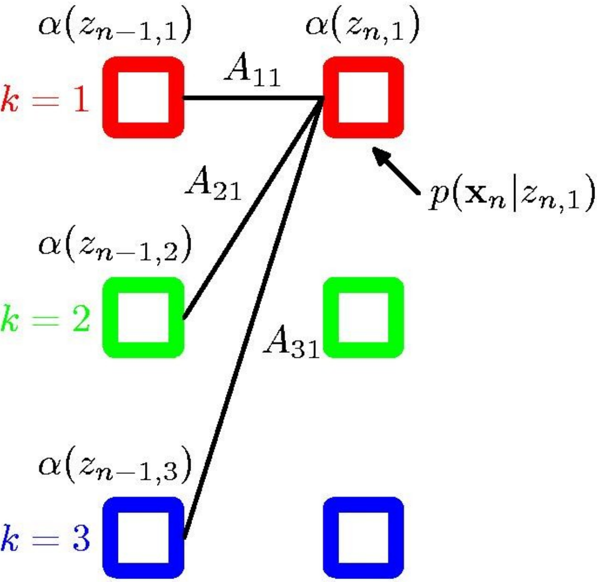
Takes time O(K^2N) and space O(KN) using memorization

Th m



ω(**z**$_n$) is the probabili of states **z**$_1$,...,**z**$_n$ generating the obse

$$\omega(\mathbf{z}_n) \equiv \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-}}$$



Recursion:

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\max_{\mathbf{z}_{n-1}}\omega(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Basis:

$$\omega(\mathbf{z} \quad n-1$$

Problem: The values ω(z$_{nk}$) can come very close to zero, by multiplying them we potentially exceed the precision of double precision floating points

Solution: Because *log* (max f) = max *log* f, we can work in "log-space" which turns multiplications into additions ...

The Viterbi Algorithm in log-space

ω(**z**$_n$) is the probability of the most likely sequence of states **z**$_1$,...,**z**$_n$ generating the observations **x**$_1$,...,**x**$_n$

$$\log \omega(\mathbf{z}_n) = \max_{\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}} \log p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{z}_1,\ldots,\mathbf{z}_n)$$

Recursion:

$$\log \omega(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\log \omega(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)$$

$$\hat{\omega}(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\hat{\omega}(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)$$

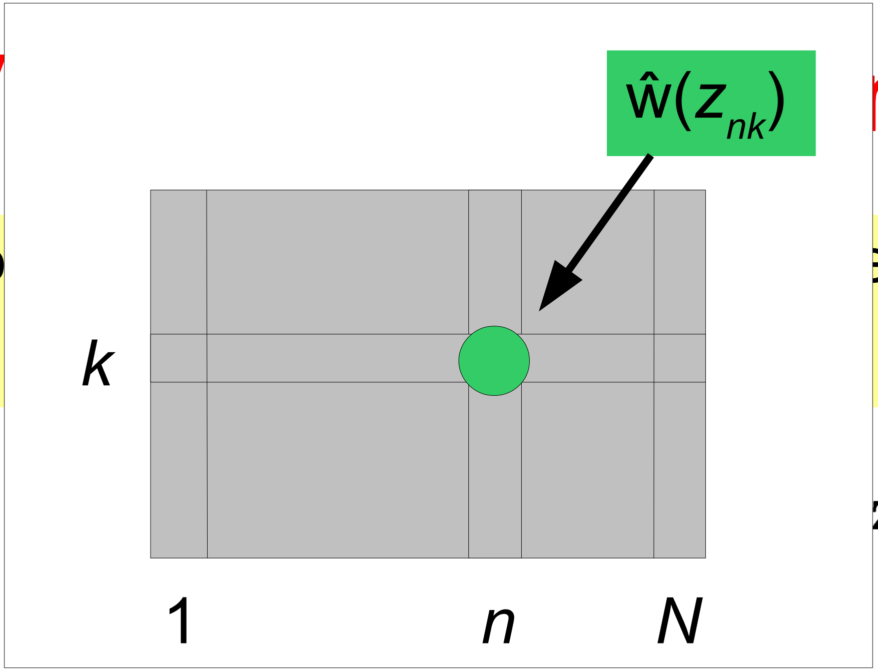
Basis:

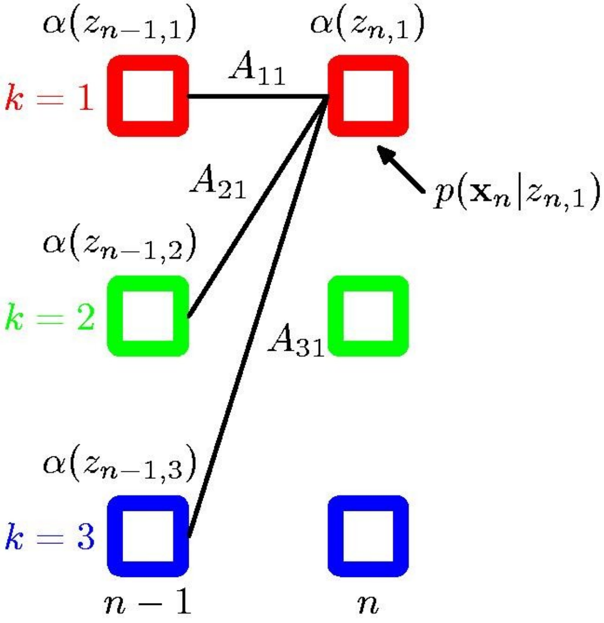
$$\hat{\omega}(\mathbf{z}_1) = \log \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}} = \sum_{k=1}^{K} z_{1k}(\log \pi_k + \log p(\mathbf{x}_1|\phi_k))$$

The V n log-space

$\omega(\mathbf{z}_n)$ is the pro ence of states z z
generating the

$$\log \omega(\mathbf{z}_n) = \quad {}_{\mathbf{z}_1} \quad \mathbf{z}_n)$$





Recursion:

$$\log \omega(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\log \omega(\mathbf{z}_{n-1}) + \log\right.$$

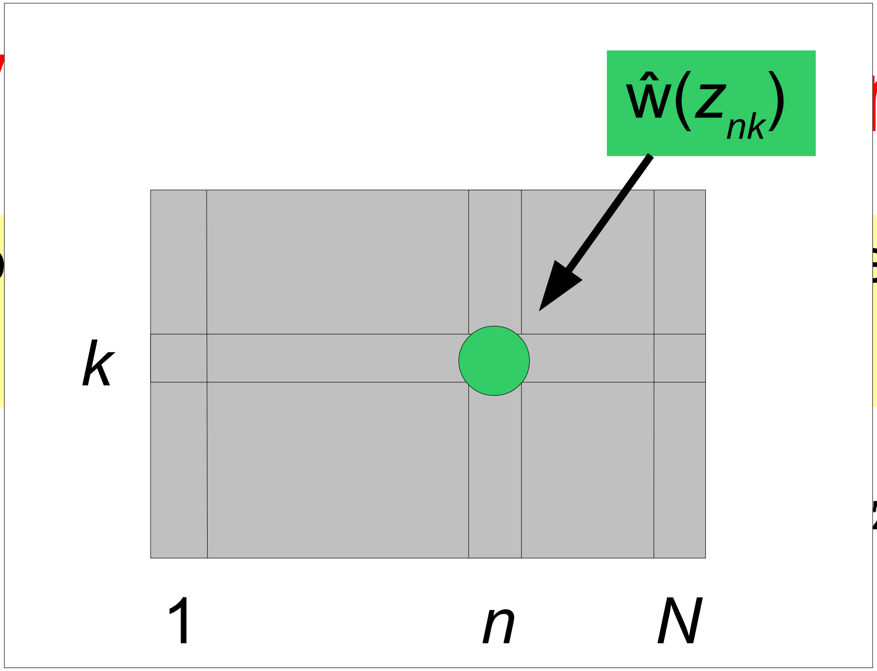
$$\boxed{\hat{\omega}(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\hat{\omega}(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)}$$

Basis:

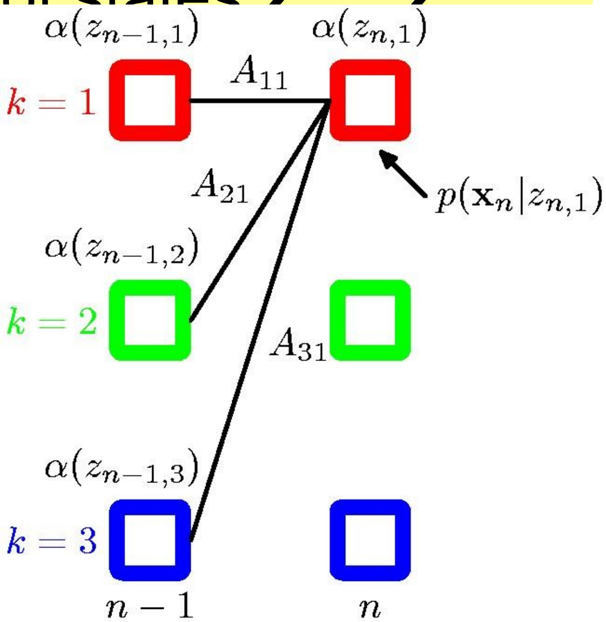
$$\hat{\omega}(\mathbf{z}_1) = \log \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}} = \sum_{k=1}^{K} z_{1k}(\log \pi_k + \log p(\mathbf{x}_1|\phi_k))$$

The V n log-space

$\omega(\mathbf{z}_n)$ is the pro ence of states $\mathbf{z}$ $\mathbf{z}$
generating the



$$\log \omega(\mathbf{z}_n) = \ \mathbf{z}_1 \ \mathbf{z}_n)$$



Recursion:

$$\log \omega(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\log \omega(\mathbf{z}_{n-1}) + \log\right.$$

$$\boxed{\hat{\omega}(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\hat{\omega}(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)}$$

Basis:

$$\hat{\omega}(\mathbf{z}_1) = \log \prod^{K} \{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}} = \sum^{K} z_{1k} (\log \pi_k + \log p(\mathbf{x}_1|\phi_k))$$

Still takes time O(K^2N) and space O(KN) using memorization, and the most likely sequence of states can be found be *backtracking*

Backtracking

Pseudocode for backtracking not using log-space:

```
$z[1..N]$ = undef

$z[N] = \arg\max_k \omega[k][N]$

for $n = N-1$ to $1$:
    $z[n] = \arg\max_k ( p(x[n+1] \mid z[n+1]) * \omega[k][n] * p(z[n+1] \mid k ) )$

print $z[1..N]$
```

Pseudocode for backtracking using log-space:

```
$z[1..N]$ = undef

$z[N] = \arg\max_k \omega\text{^}[k][N]$

for $n = N-1$ to $1$:
    $z[n] = \arg\max_k ( \log p(x[n+1] \mid z[n+1]) + \omega\text{^}[k][n] + \log p(z[n+1] \mid k ) )$

print $z[1..N]$
```

Takes time O(NK) but requires the entire ω- or ω^-table in memory

A problem with “log-space”?

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \max_{\mathbf{z}_{n-1}} \omega(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\hat{\omega}(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\hat{\omega}(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)$$

What if $p(\mathbf{x}_n|\mathbf{z}_n)$ or $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ is 0? Then the product of probabilities becomes 0, but what should it be in log-space?

A problem with “log-space”?

$$\omega(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \max_{\mathbf{z}_{n-1}} \omega(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

$$\hat{\omega}(\mathbf{z}_n) = \log p(\mathbf{x}_n|\mathbf{z}_n) + \max_{\mathbf{z}_{n-1}} \left(\hat{\omega}(\mathbf{z}_{n-1}) + \log p(\mathbf{z}_n|\mathbf{z}_{n-1})\right)$$

What if $p(\mathbf{x}_n|\mathbf{z}_n)$ or $p(\mathbf{z}_n|\mathbf{z}_{n-1})$ is 0? Then the product of probabilities becomes 0, but what should it be in log-space?

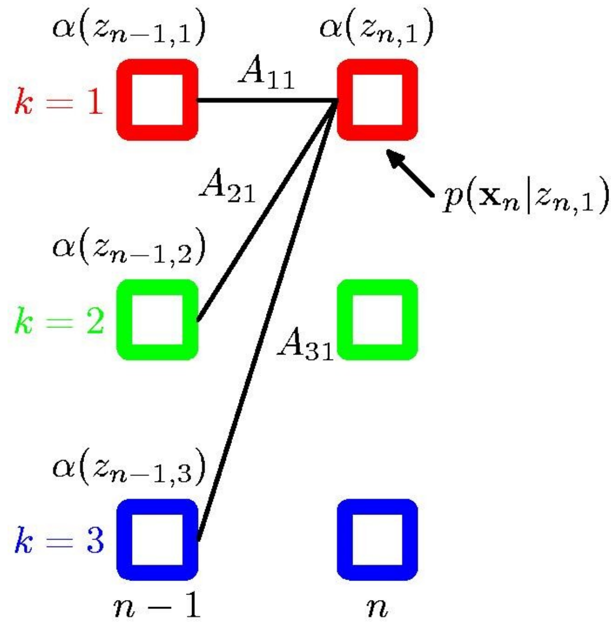
It should be some representation of “minus infinity”

```
// Pseudo code for computing w^[k][n] for some n>1
w^[n][k] = undef
if p(x[n] | k) != 0:
    for j = 1 to K:
        if p(k | j) != 0:
            w^[n][k] = max(w^[k][n], log(p(x[n] | k)) + w^[j][n-1] + log(p(k | j)))
```

The Forward Algorithm

$\alpha(\mathbf{z}_n)$ is the joint probability of observing $\mathbf{x}_1,\ldots,\mathbf{x}_n$ and being in state $\mathbf{z}_n$

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathbf{z}_n)$$



Recursion:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}}\alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

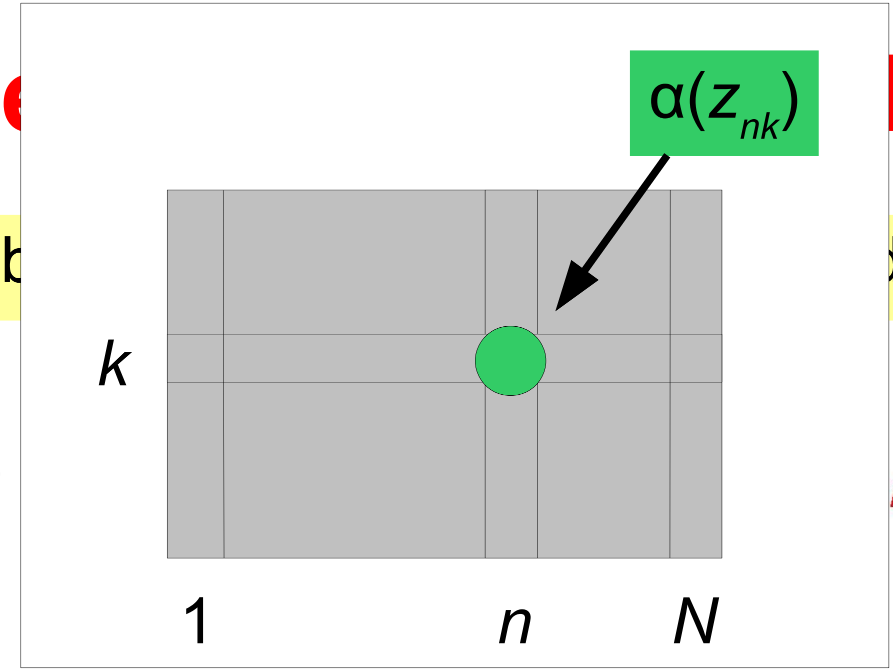
Basis:

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1,\mathbf{z}_1) = p(\mathbf{z}_1)p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K}\{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}}$$

Takes time O(K^2N) and space O(KN) using memorization

The hm

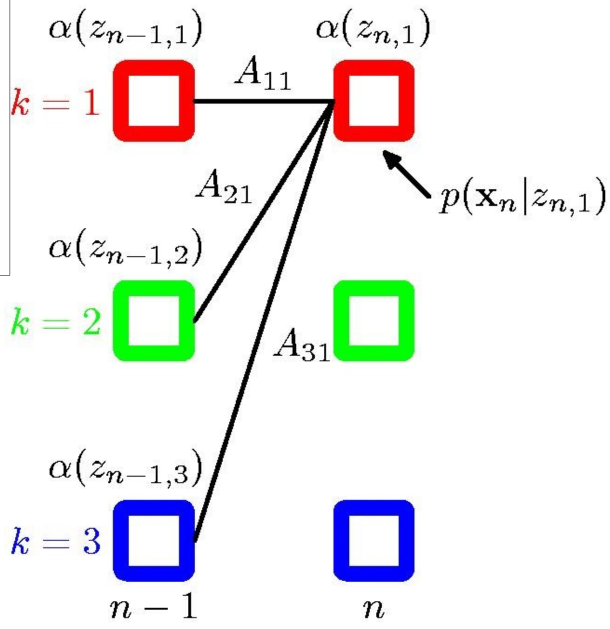
$\alpha(\mathbf{z}_n)$ is the joint prob being in state $\mathbf{z}_n$



$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, \ldots$$

Recursion:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

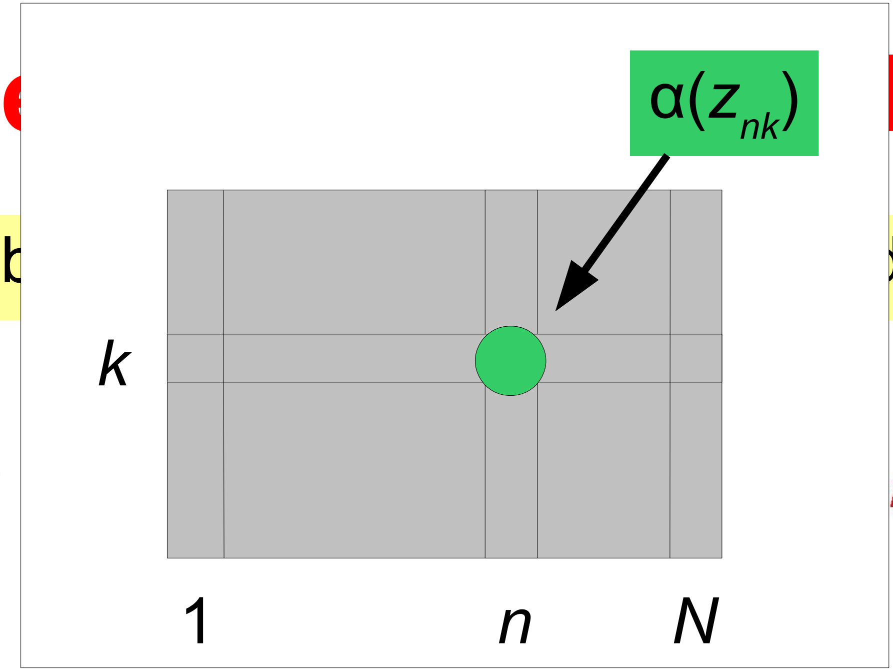


Basis:

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) = \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1|\phi_k)\}^{z_{1k}}$$

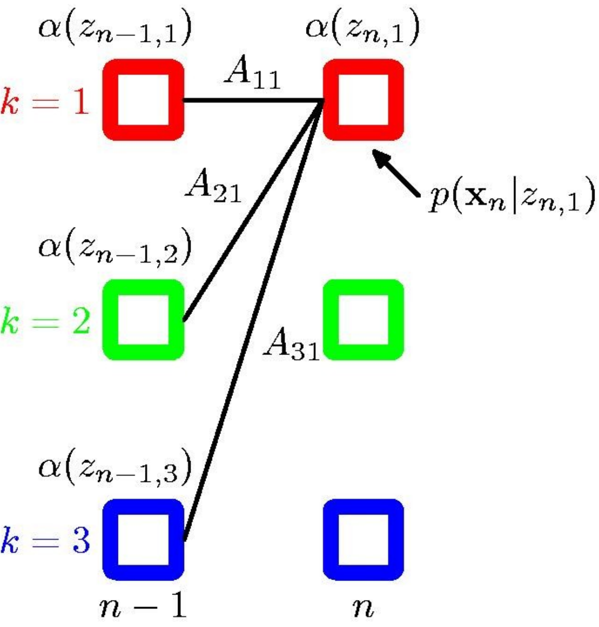
Takes time O(K^2N) and space O(KN) using memorization

The hm



$\alpha(\mathbf{z}_n)$ is the joint prob being in state $\mathbf{z}_n$

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, \ldots$$



Recursion:

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Basis:

$\alpha(\mathbf{z}$

Problem: The values $\alpha(z_{nk})$ can come very close to zero, by multiplying them we potentially exceed the precision of double precision floating points

Another problem: Because *log* (Σ f) ≠ Σ (*log* f), we cannot use the “log-space” trick ...

Forward algorithm using scaled values

α(**z**$_n$) is the joint probability of observing **x**$_1$,...,**x**$_n$ and being in state **z**$_n$

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n) p(\mathbf{z}_n | \mathbf{x}_1, \ldots, \mathbf{x}_n)$$

$$\boxed{\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n | \mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_n)} = \frac{\alpha(\mathbf{z}_n)}{\prod_{m=1}^{n} c_m}}$$

$$c_n = p(\mathbf{x}_n | \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) \qquad p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{m=1}^{n} c_m$$

This "normalized version" *α*(**z**$_n$) is a probability distribution over *K* variables, and we expect it to "behave numerically well" because

$$\sum_{k=1}^{K} \hat{\alpha}(z_{nk}) = 1$$

The normalized values can not all become arbitrary small ...

Forward algorithm using scaled values

We can modify the forward-recursion to use scaled values

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \left(\prod_{m=1}^{n-1} c_m\right) \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\boxed{c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})}$$

$$\boxed{\alpha(\mathbf{z}_n) = \left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n)}$$

Forward algorithm using scaled values

We can modify the forward-recursion to use scaled values

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \left(\prod_{m=1}^{n-1} c_m\right) \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\boxed{c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})}$$

If we know *c*$_n$ then we have a recursion using the normalized values

$$\boxed{\alpha(\mathbf{z}_n) = \left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n)}$$

Forward algorithm using scaled values

We can modify the forward-recursion to use scaled values

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \left(\prod_{m=1}^{n-1} c_m\right) \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1}) \Leftrightarrow$$

$$\boxed{c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})}$$

If we know c_n then we have a recursion using the normalized values

$$\boxed{\sum_{k=1}^{K} c_n \hat{\alpha}(z_{nk}) = c_n \sum_{k=1}^{K} \hat{\alpha}(z_{nk}) = c_n \cdot 1}$$

$$\boxed{\alpha(\mathbf{z}_n) = \left(\prod_{m=1}^{n} c_m\right) \hat{\alpha}(\mathbf{z}_n)}$$

Forward algorithm using scaled values

We can modify the forward-recursion to use scaled values

Recursion:

In step *n* compute and store temporarily the *K* values δ(z$_{n1}$), ..., δ(z$_{nK}$)

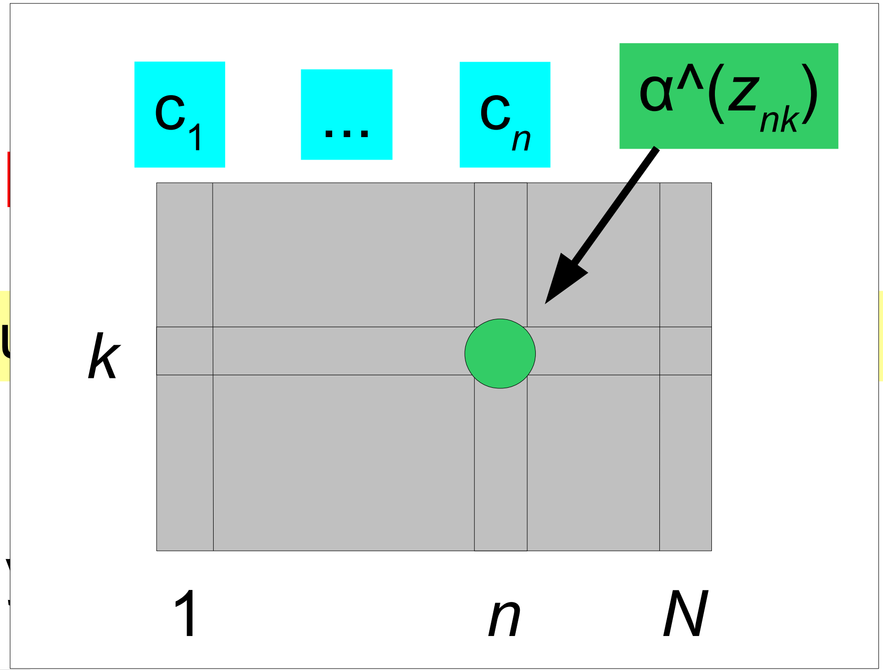
$$\delta(\mathbf{z}_n) = c_n\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}}\hat{\alpha}(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Compute and store c_n as

$$\sum_{k=1}^{K}\delta(z_{nk}) = \sum_{k=1}^{K}c_n\hat{\alpha}(z_{nk}) = c_n\sum_{k=1}^{K}\hat{\alpha}(z_{nk}) = c_n$$

Compute and store $\hat{\alpha}(z_{nk}) = \delta(z_{nk})/c_n$

Forward algorithm usi



We can modify the forward-recursion to u

Recursion:

In step n compute and store temporaril

$$\delta(\mathbf{z}_n) = c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Compute and store c_n as

$$\sum_{k=1}^{K} \delta(z_{nk}) = \sum_{k=1}^{K} c_n \hat{\alpha}(z_{nk}) = c_n \sum_{k=1}^{K} \hat{\alpha}(z_{nk}) = c_n$$

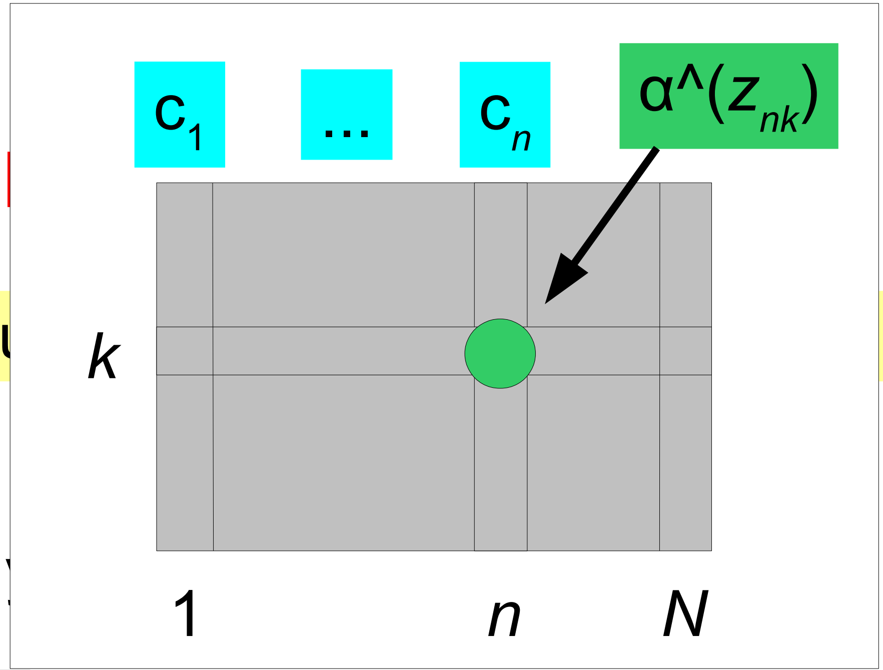
Compute and store $\hat{\alpha}(z_{nk}) = \delta(z_{nk})/c_n$

Basis:

$$\hat{\alpha}(\mathbf{z}_1) = \frac{\alpha(\mathbf{z}_1)}{c_1} \qquad c_1 = p(\mathbf{x}_1) = \sum_{\mathbf{z}_1} p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_1|\phi_k)$$

Forward algorithm usi

We can modify the forward-recursion to u



Recursion:

In step *n* compute and store temporaril

$$\delta(\mathbf{z}_n) = c_n \hat{\alpha}(\mathbf{z}_n) = p(\mathbf{x}_n|\mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \hat{\alpha}(\mathbf{z}_{n-1}) p(\mathbf{z}_n|\mathbf{z}_{n-1})$$

Compute and store c_n as

$$\sum_{k=1}^{K} \delta(z_{nk}) = \sum_{k=1}^{K} c_n \hat{\alpha}(z_{nk}) = c_n \sum_{k=1}^{K} \hat{\alpha}(z_{nk}) = c_n$$

Compute and store $\hat{\alpha}(z_{nk}) = \delta(z_{nk})/c_n$

Takes time O(K^2N) and space O(KN) using memorization

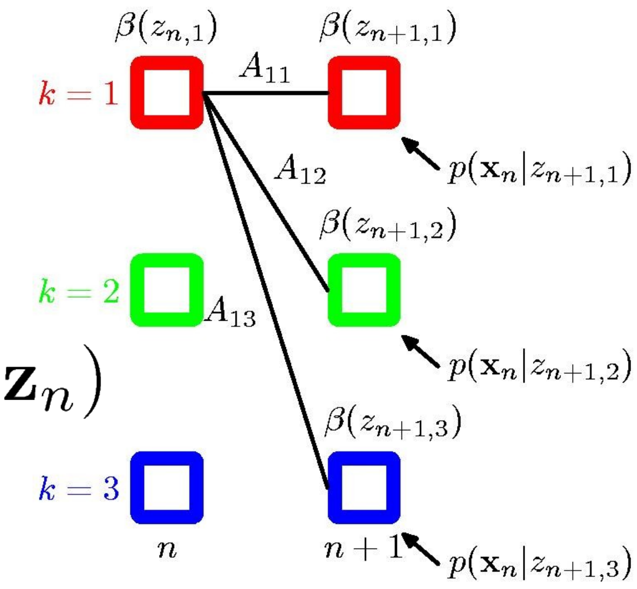
Basis:

$$\hat{\alpha}(\mathbf{z}_1) = \frac{\alpha(\mathbf{z}_1)}{c_1} \qquad c_1 = p(\mathbf{x}_1) = \sum_{\mathbf{z}_1} p(\mathbf{z}_1) p(\mathbf{x}_1|\mathbf{z}_1) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}_1|\phi_k)$$

The Backward Algorithm

$\beta(\mathbf{z}_n)$ is the conditional probability of future observation $\mathbf{x}_{n+1},\ldots,\mathbf{x}_N$ assuming being in state $\mathbf{z}_n$

$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1},\ldots,\mathbf{x}_N|\mathbf{z}_n)$$



Recursion:

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

Basis:

$$\beta(\mathbf{z}_N) = 1$$

Takes time O(K^2N) and space O(KN) using memorization

Backward algorithm using scaled values

We can modify the backward-recursion to use scaled values

Recursion:

In step *n* compute and store temporarily the *K* values ε(z$_{n1}$), ..., ε(z$_{nK}$)

$$\epsilon(\mathbf{z}_n) = c_{n+1}\hat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \hat{\beta}(\mathbf{z}_{n+1})p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

Using *c$_{n+1}$* computed during the forward-recursion, compute and store

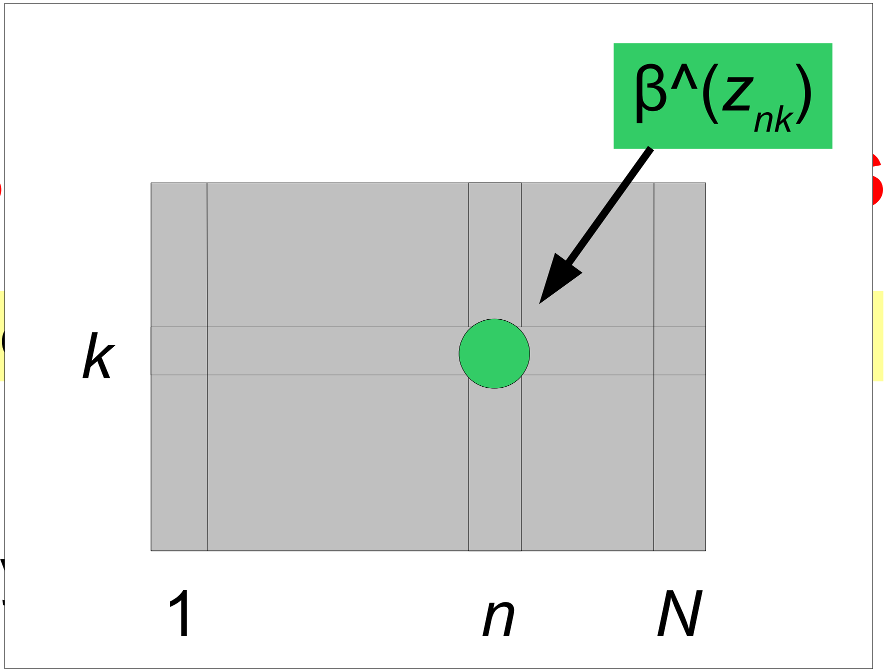
$$\hat{\beta}(z_{nk}) = \epsilon(z_{nk})/c_{n+1}$$

Basis:

$$\hat{\beta}(\mathbf{z}_N) = 1$$

Backward algorithm us

We can modify the backward-recursion t



Recursion:

In step *n* compute and store temporaril

$$\epsilon(\mathbf{z}_n) = c_{n+1}\hat{\beta}(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \hat{\beta}(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}) p(\mathbf{z}_{n+1}|\mathbf{z}_n)$$

Using c_{n+1} computed during the forward-recursion, compute and store

$$\hat{\beta}(z_{nk}) = \epsilon(z_{nk})/c_{n+1}$$

Basis:

$$\hat{\beta}(\mathbf{z}_N) = 1$$

Takes time O(K^2N) and space O(*KN*) using memorization