Chapter 6

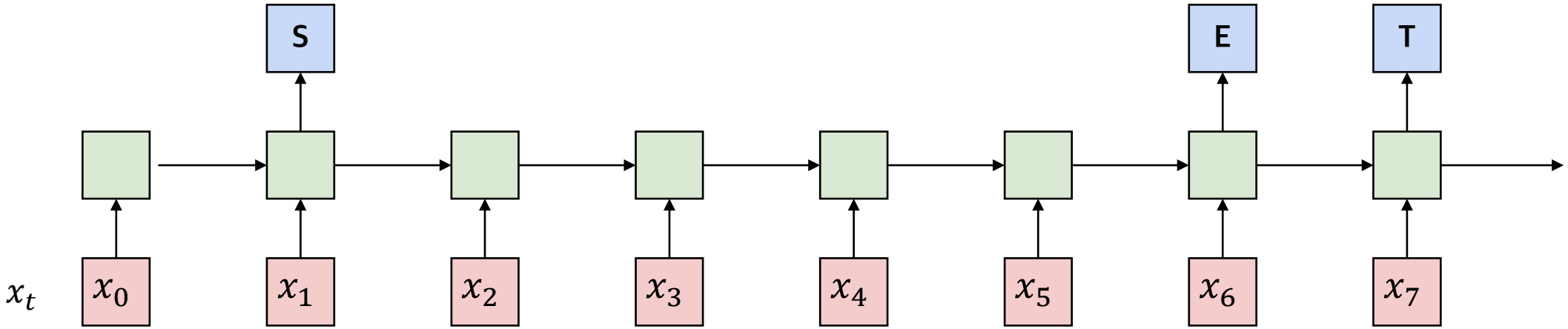
Connectionist Temporal Classification

김지환

서강대학교 컴퓨터공학과

Table of contents

6.1 Sequence-to-Sequence에서의 Inference

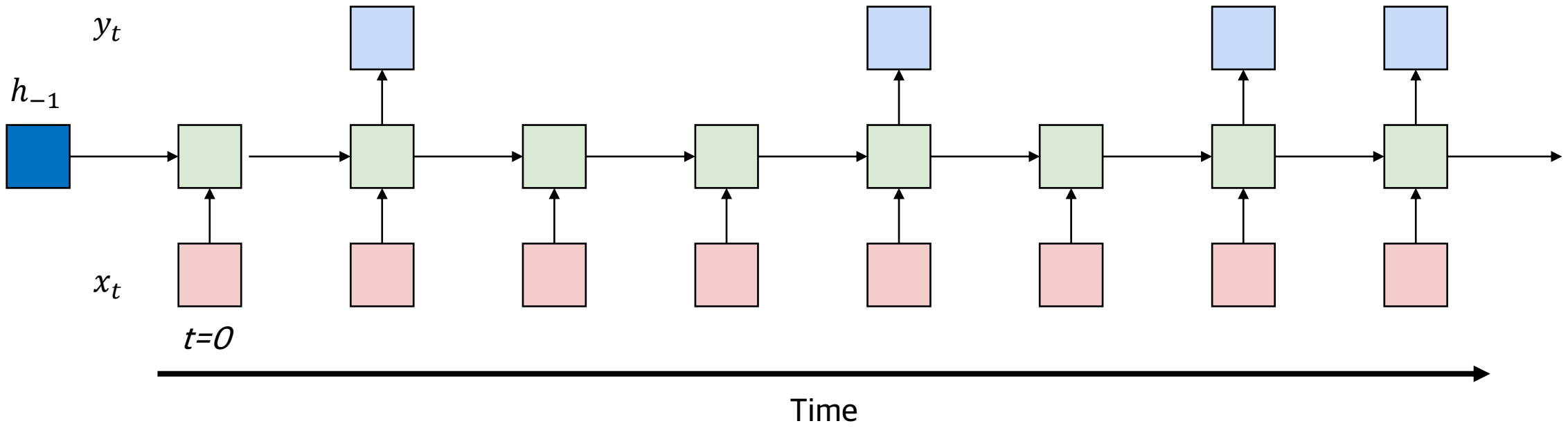


- 입력 sequence가 주어지면 비동기적으로 symbol sequence를 출력
- Decoding
  - 주어진 입력에 가장 가능성이 높은 symbol sequence를 찾아야 함

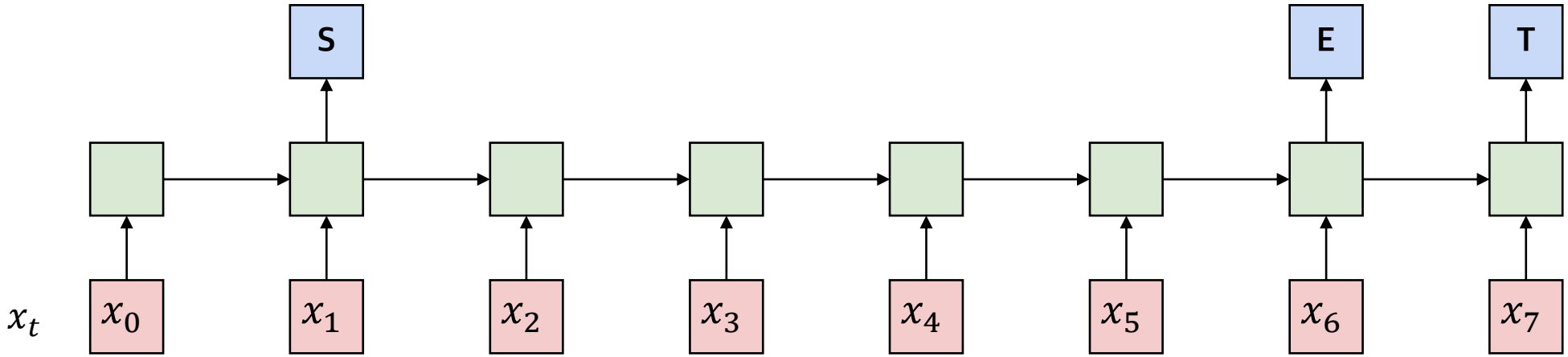
$$S_0 \dots S_{K-1} = \underset{S'_0 \dots S'_{K-1}}{\text{argmax}} \ \text{prob}\left(S'_0 \dots S'_{K-1} \mid X_0 \dots X_{N-1}\right)$$

6.2.1 Order and Alignment Provided

- 입력/출력 sequence는 동일한 순서로 발생
  - 시간 동기(time-synchronous)가 아닐 수 있지만 alignment 가능
  - 음성 인식에서의 예시
    - 입력 음성은 음소 sequence 출력과 alignment 가능

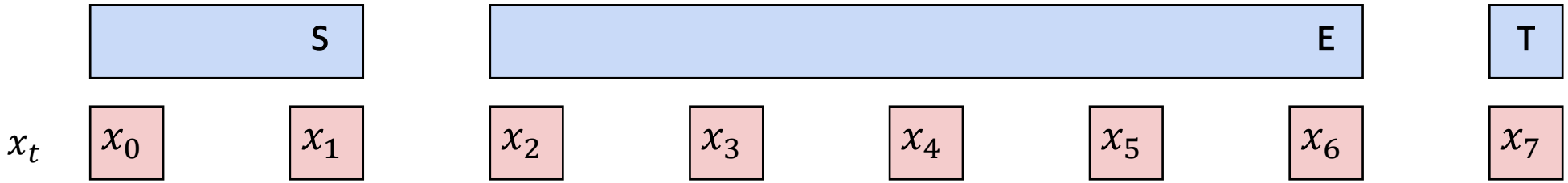


6.2.1.2 Training with Alignment



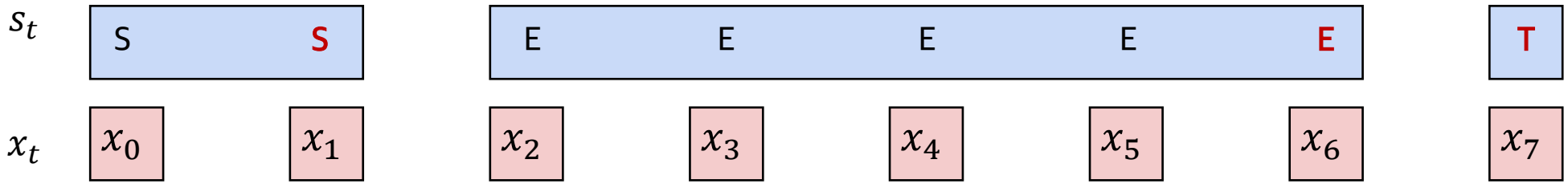
- 학습 데이터 : 입력 sequence + 출력 sequence
  - 출력 sequence의 길이 ≤ 입력 sequence의 길이
- 입력에 대한 출력의 alignment가 제공됨
  - Phoneme S는 x_1 에서, E는 x_6에서, T는 x_7에서 끝남

6.2.1.3 Characterizing an Alignment



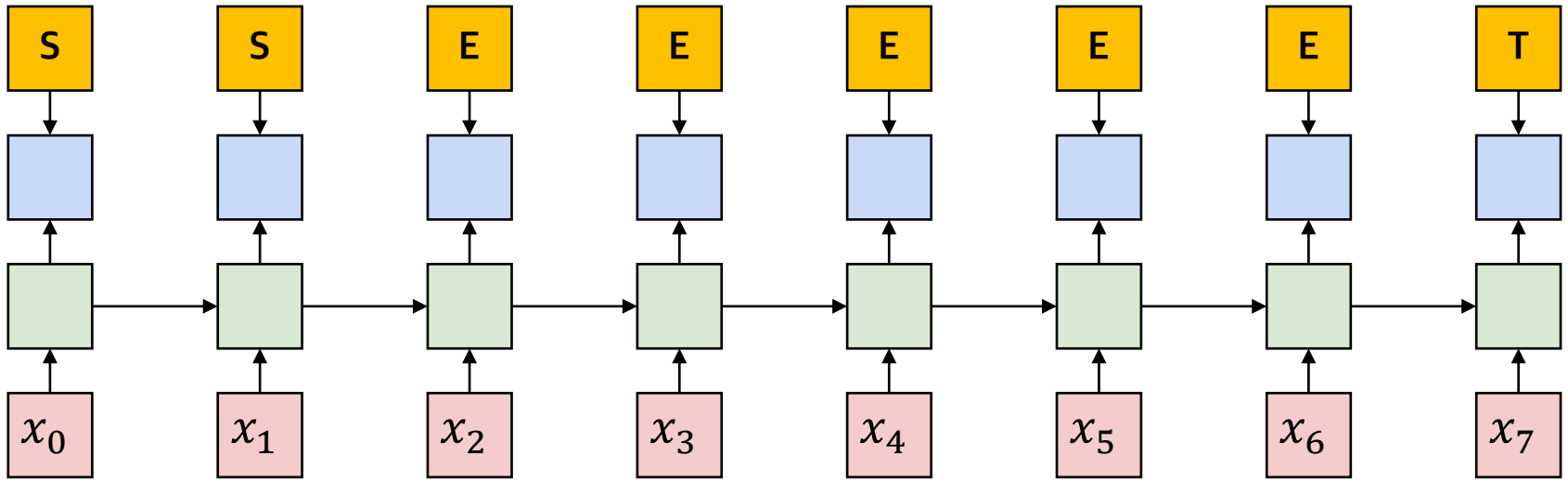
- 입력에 대한 출력의 alignment가 제공됨
  - $S_0(T_0), S_1(T_1), \ldots, S_{K-1}(T_{N-1})$
  - E.g., S_0 =S (1), S_1 =E (6), S_2 =T (7)

6.2.1.3 Characterizing an Alignment



- 입력에 대한 출력의 alignment가 제공됨
  - $S_0(T_0), S_1(T_1), \ldots, S_{K-1}(T_{K-1})$
  - E.g., S_0 =S (1), S_1 =E (6), S_2 =T (7)
- Time-synchronous sequence로 symbol을 반복
  - $S_0, S_1, \ldots, S_{N-1} = S_0, S_0, \ldots, (T_0\ times), S_1, S_1, \ldots (T_1\ times), \ldots, S_{K-1}$
  - E.g., $S_0, S_1, \ldots, S_7$ = S S E E E E E T

6.2.1.4 Training with Alignment



- 각각의 입력에 대하여 time-synchronous 출력이 있고 loss를 구할 수 있으므로 학습 가능

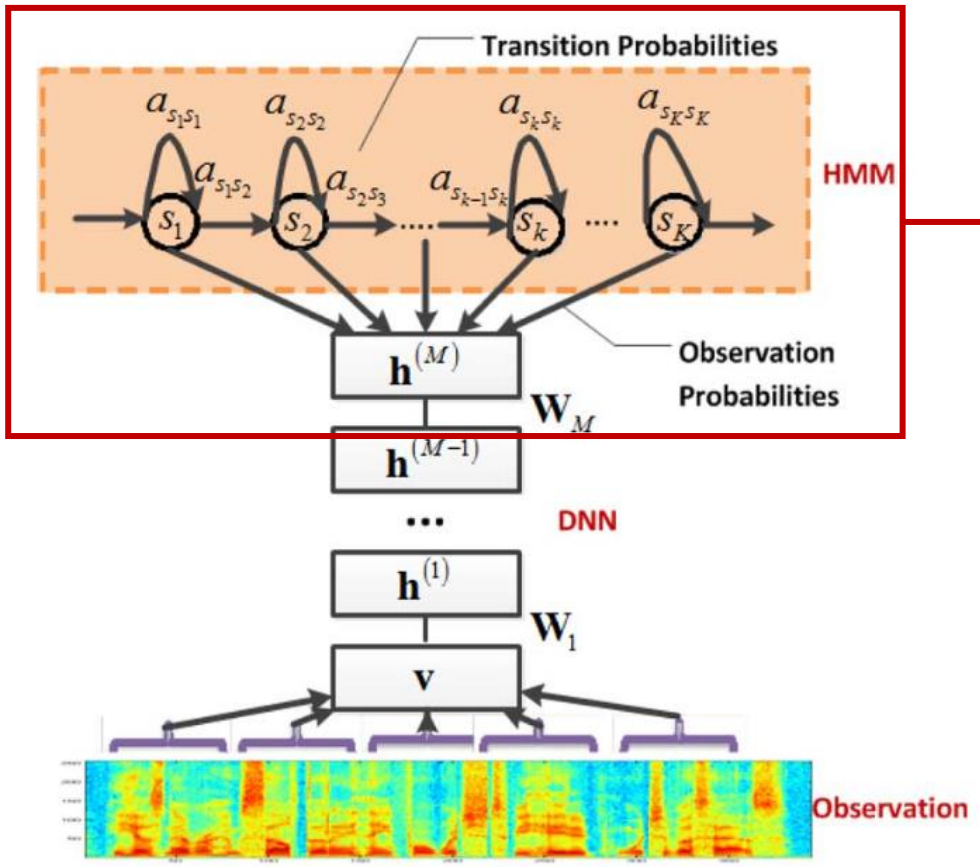
$$L = \sum_t \text{Xent}\left(y_t, \text{symbol}_t\right)$$

- E.g.)

$$L = \text{Xent}(y_0, S) + \text{Xent}(y_1, S) + \text{Xent}(y_2, E) + \text{Xent}(y_3, E) + \text{Xent}(y_4, E) + \text{Xent}(y_5, E) + \text{Xent}(y_6, E) + \text{Xent}(y_7, T)$$

6.2.1.5 참고 : DNN-WFST Model

- GMM의 역할을 DNN이 대체 [Hinton, 2006]
  - DNN 모델로는 DBN을 사용



[Dahl, 2012]

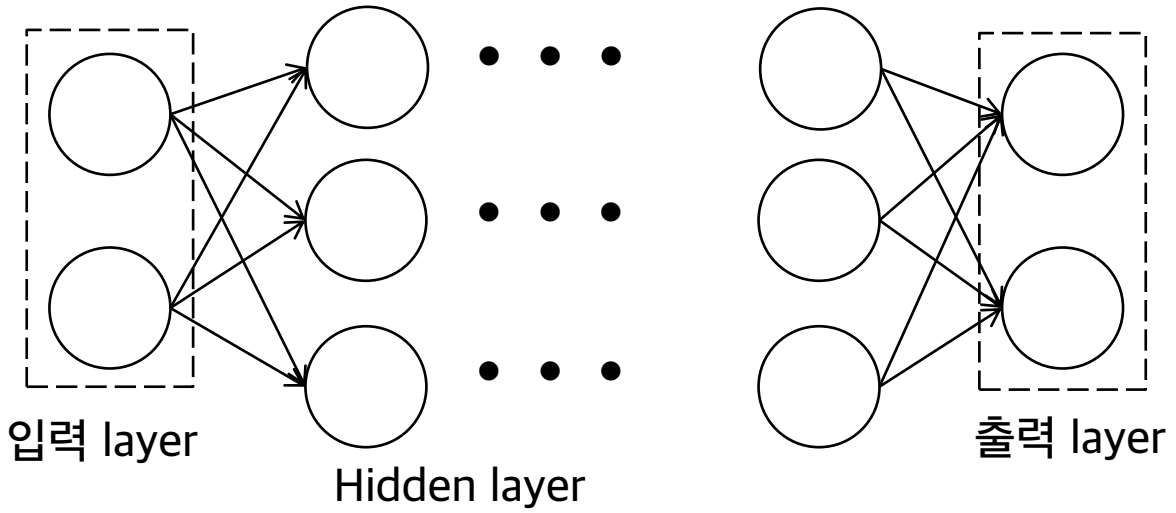
Network output

$$\underbrace{P\left(o_i \mid s^k\right)}_{\text{Emission prob.}} = \frac{P\left(s^k \mid o_i\right) P\left(o_i\right)}{P\left(s^k\right)} \approx \frac{P\left(s^k \mid o_i\right)}{P\left(s^k\right)}$$

6.2.1.6 Recall : Back-propagation Algorithm

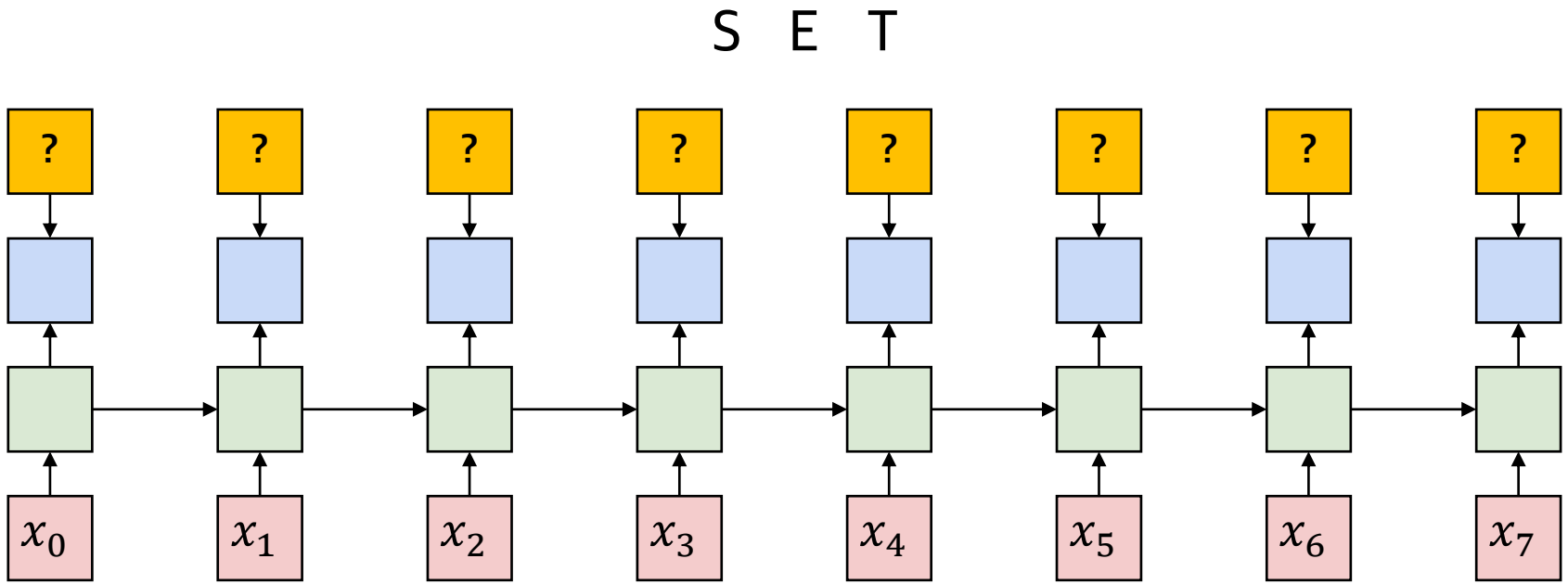
- 학습은 back-propagation 알고리즘으로 수행

$$y = f(\sum_i x_i w_{ij} + b)$$



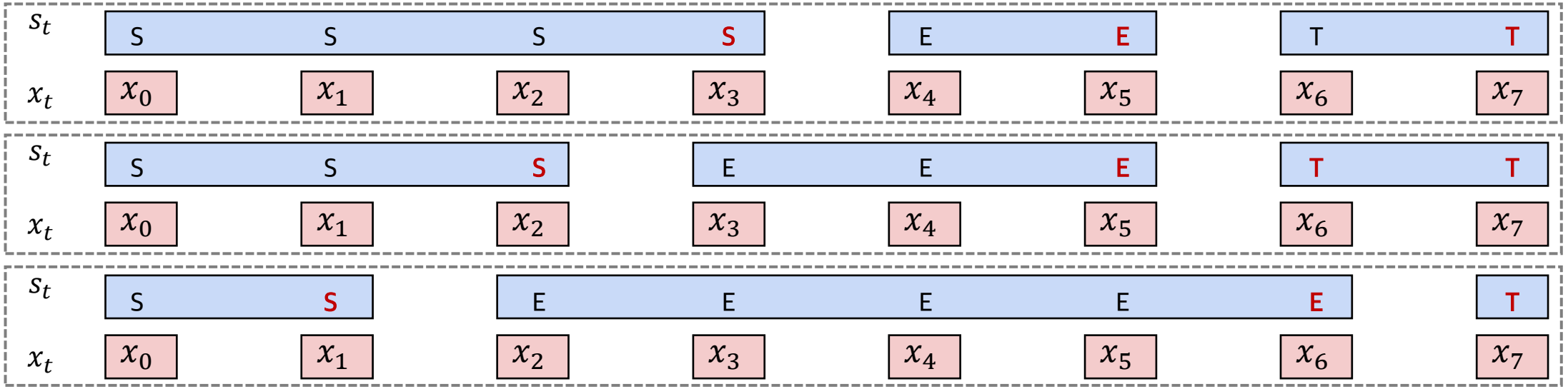
$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

6.2.2 Order Provided But Alignment Not Provided



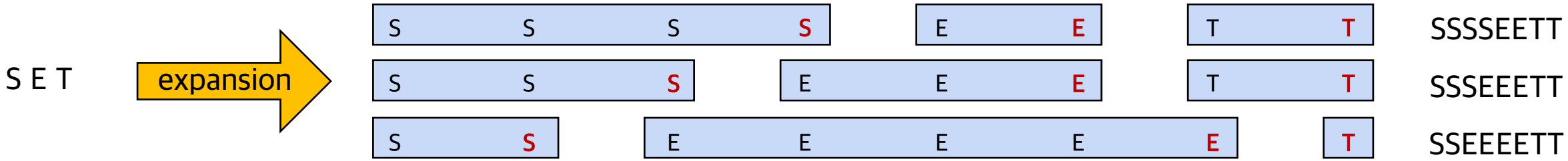
- 학습 데이터의 출력 symbol만이 주어짐
  - 순서는 주어지나 시점에 대한 정보는 주어지지 않음

6.2.2.1 Characterizing an Alignment

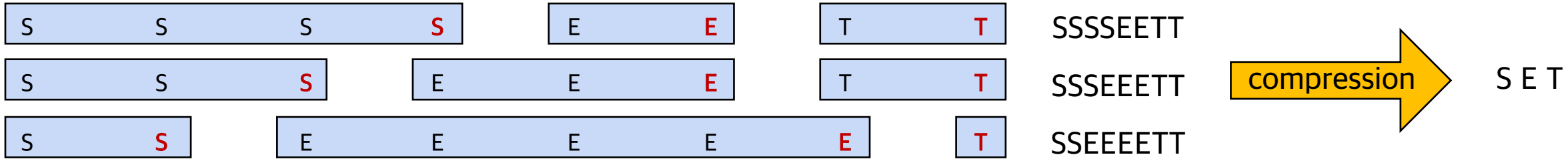


- Alignment : 입력 sequence의 어떤 symbol이 어느 위치에 정렬되는지를 제공
  - 위 그림은 S E T 가 x_0 ... x_7에 대응되는 여러 alignments를 나타냄
- Alignment는 symbols의 반복으로 표현가능
  - "compressed" sequence를 입력의 길이로 "expansion" 가능

6.2.2.2 Expansion and Compression



asynchronous “compressed” sequence는 다양한 time-synchronous “expanded” sequence로 입력의 길이에 맞게 align가능



입력에 대한 다양한 time-synchronous alignments들은 asynchronous unaligned “compressed” sequence로 표현 가능

6.2.2.3 Estimating an Alignment

- Alignment problem
  - Unaligned K-length compressed symbol sequence $S = S_0 \ldots S_{K-1}$
  - N-length input ($N \geq K$) $\mathrm{X} = x_0 \ldots x_{N-1}$

- Find the most likely alignment :

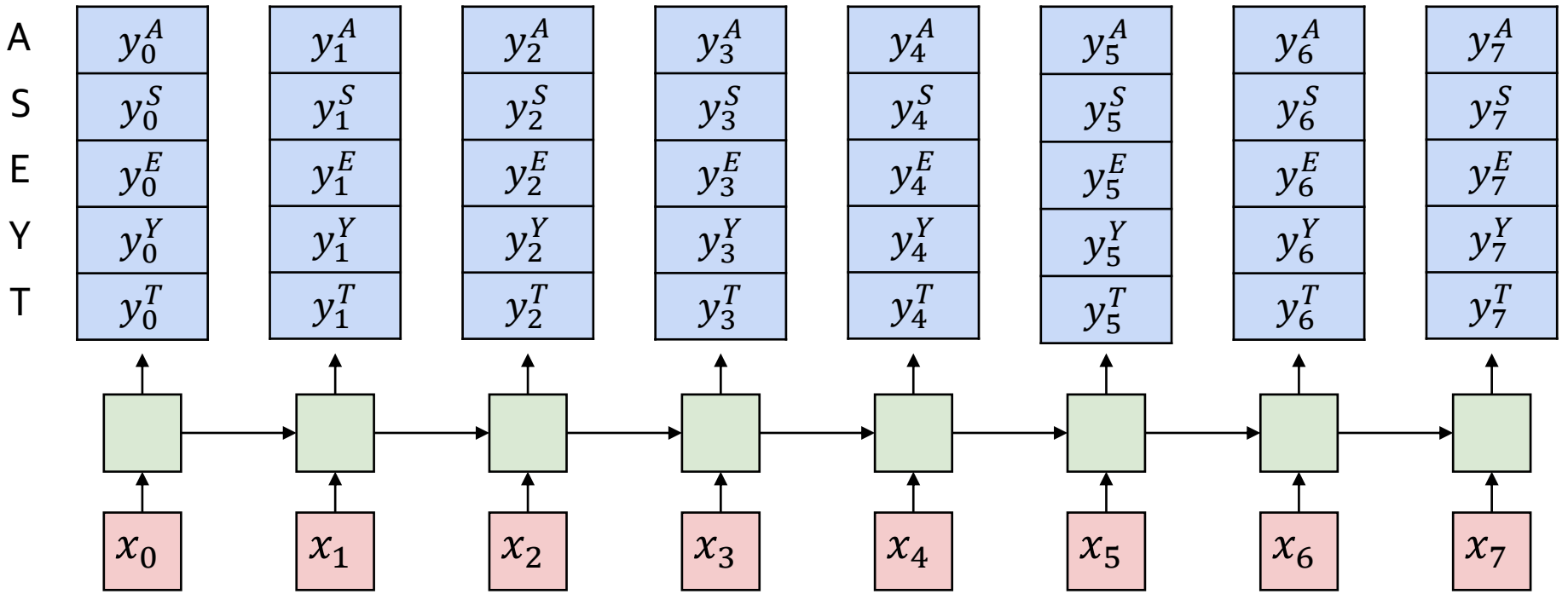
$$\operatorname{argmax} P\left(s_0, s_1, \ldots, s_{N-1} \mid S_0, S_1, \ldots, S_{K-1}, x_0, x_1, \ldots, x_{N-1}\right)$$

  - Such that

$$\operatorname{compress}\left(s_0, s_1, \ldots, s_{N-1}\right) \equiv S_0, S_1, \ldots, S_{K-1}$$

  - compress() is the operation of compressing repetitions into one

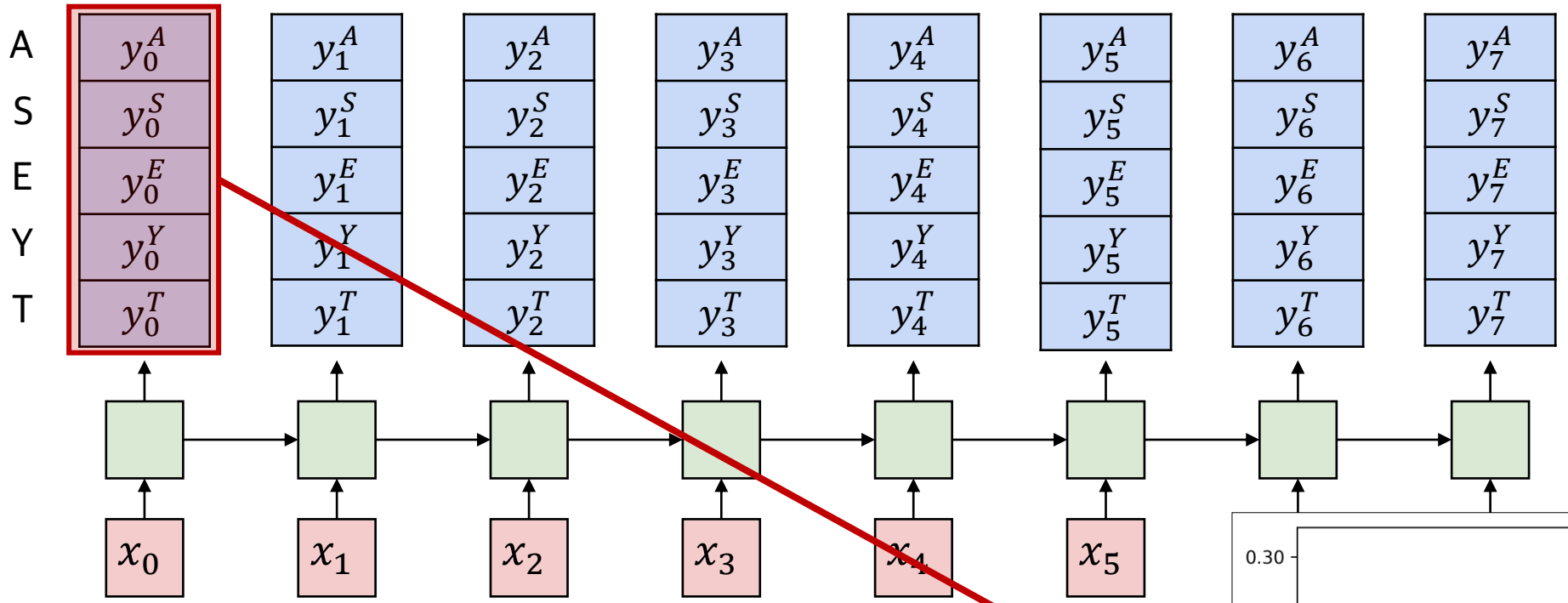
6.2.2.4 Actual Output of The Network

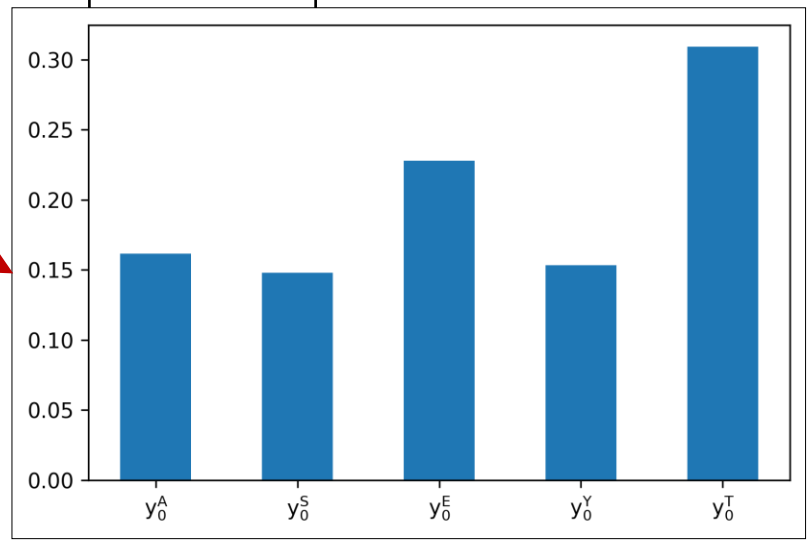


- 가능한 output symbol set {A,S,E,Y,T}
- network의 출력은 출력되는 시점 t까지의 입력 $x_0 \ldots x_t$ 이 주어졌을 때의 확률
  - 예) y_4^E

$$y_4^E = \Pr\left(s_4 = E \mid x_0 \ldots x_4\right)$$

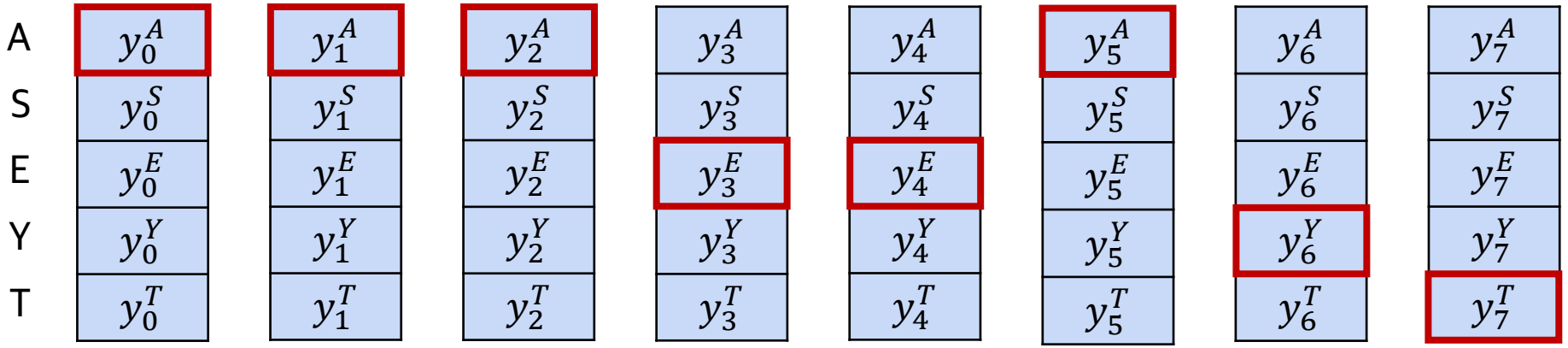
6.2.2.4 Actual Output of The Network





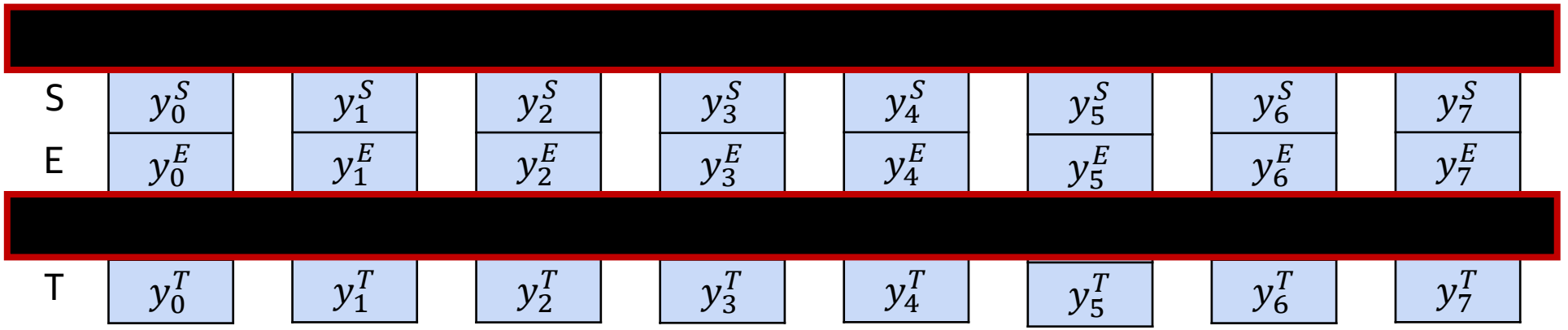
- 가능한 output symbol set {A,S,E,Y,T}
- network는 각 시점 마다의output symbol에 대한 확률을 출력
  - Softmax를 이용하여 추정된 확률

6.2.2.5 Unconstrained Decoding



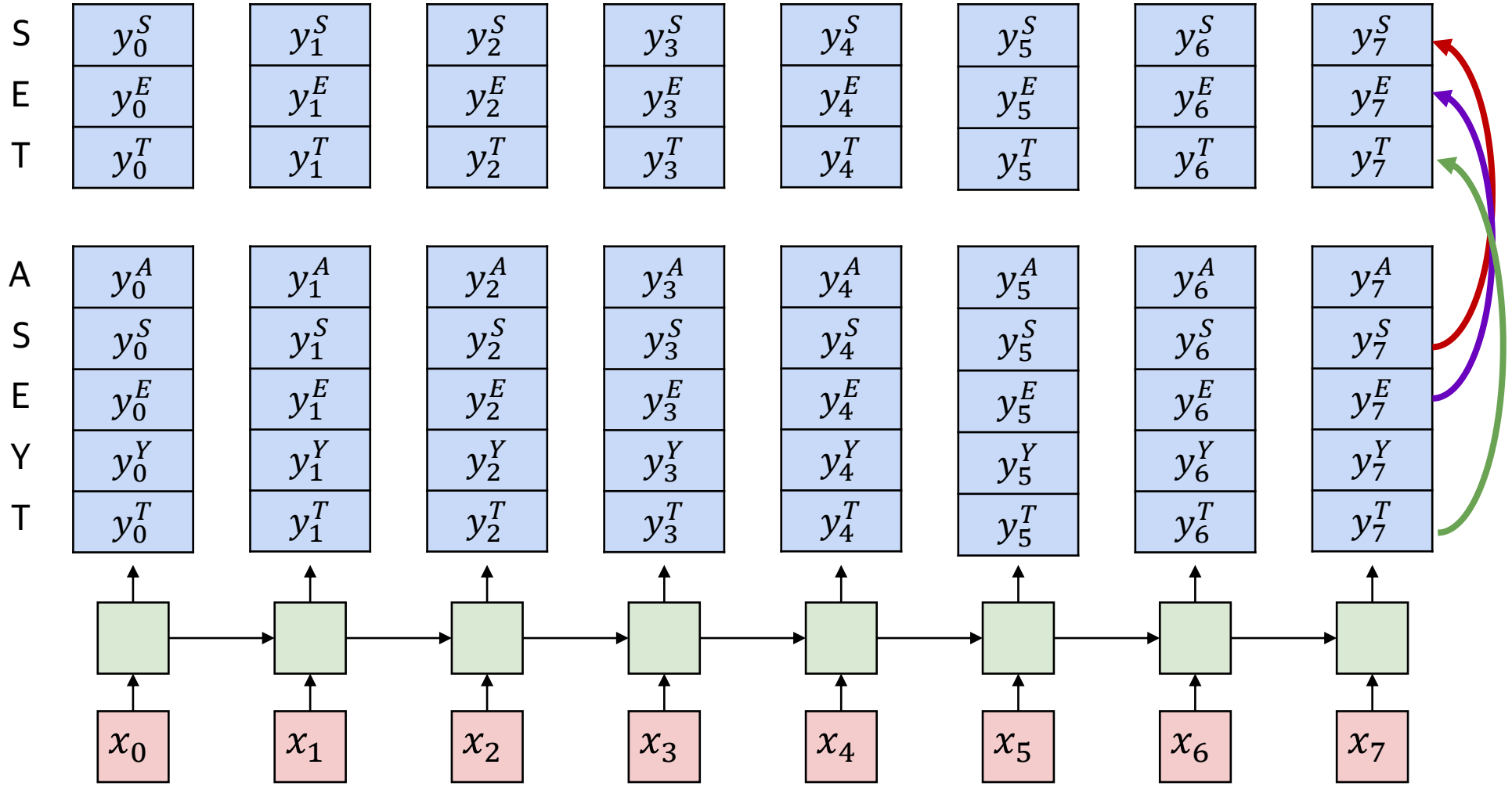
- 각 시점마다 단순히 가장 확률이 높은 symbol을 출력
  - most likely sequence of symbols conditioned on input $x_0 \ldots x_{N-1}$
    - 위 예제에서는 A A A E E A Y T
- compressed sequence : A E Y T
  - 하지만, 우리가 원하는 S E T 의 expansion만을 고려해야함

6.2.2.6 Constraining the Alignment : Try 1



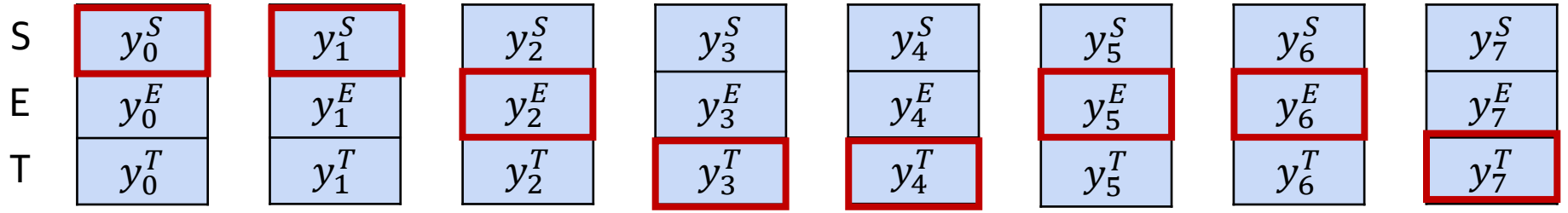
- target sequence(S E T)에 포함되지 않는 모든 행을 제외
  - S,E,T가 아닌 모든 행을 제외
    - 위 그림에서는 A와 Y에 해당하는 행을 제외

6.2.2.7 Blocking Out Unnecessary Outputs



- 모든 출력에 대한 확률을 계산
- Target sequence에 포함되는 행을 축소된 구조로 복사하여 사용

6.2.2.8 Constraining the Alignment : Try 1

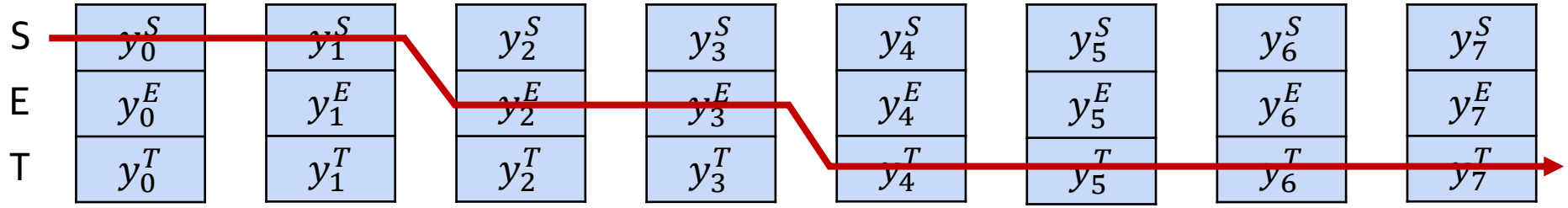


- 축소된 구조에서만 decoding을 진행
  - Target sequence에 존재하는 symbol들만이 decoding결과로 출력 가능

- 문제점 : target sequence의 expansion이 아닌 형태의 sequence가 출력 가능
  - 위의 예제에서의 compressed sequence는 S E T E T

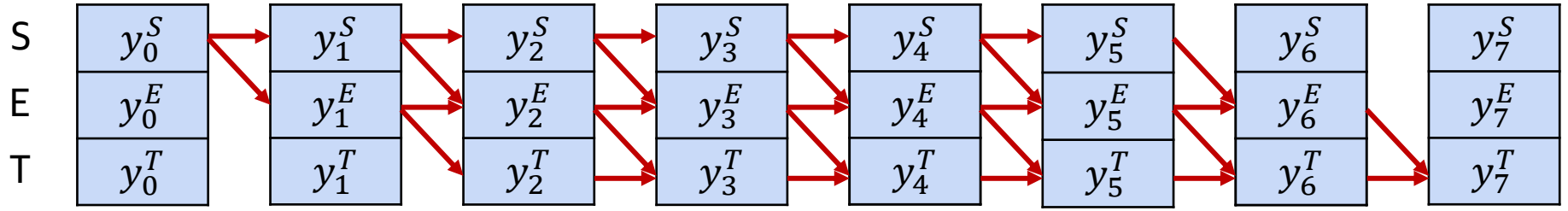
- 추가적인 constraint가 필요

6.2.2.9 Constraining the Alignment : Try 2



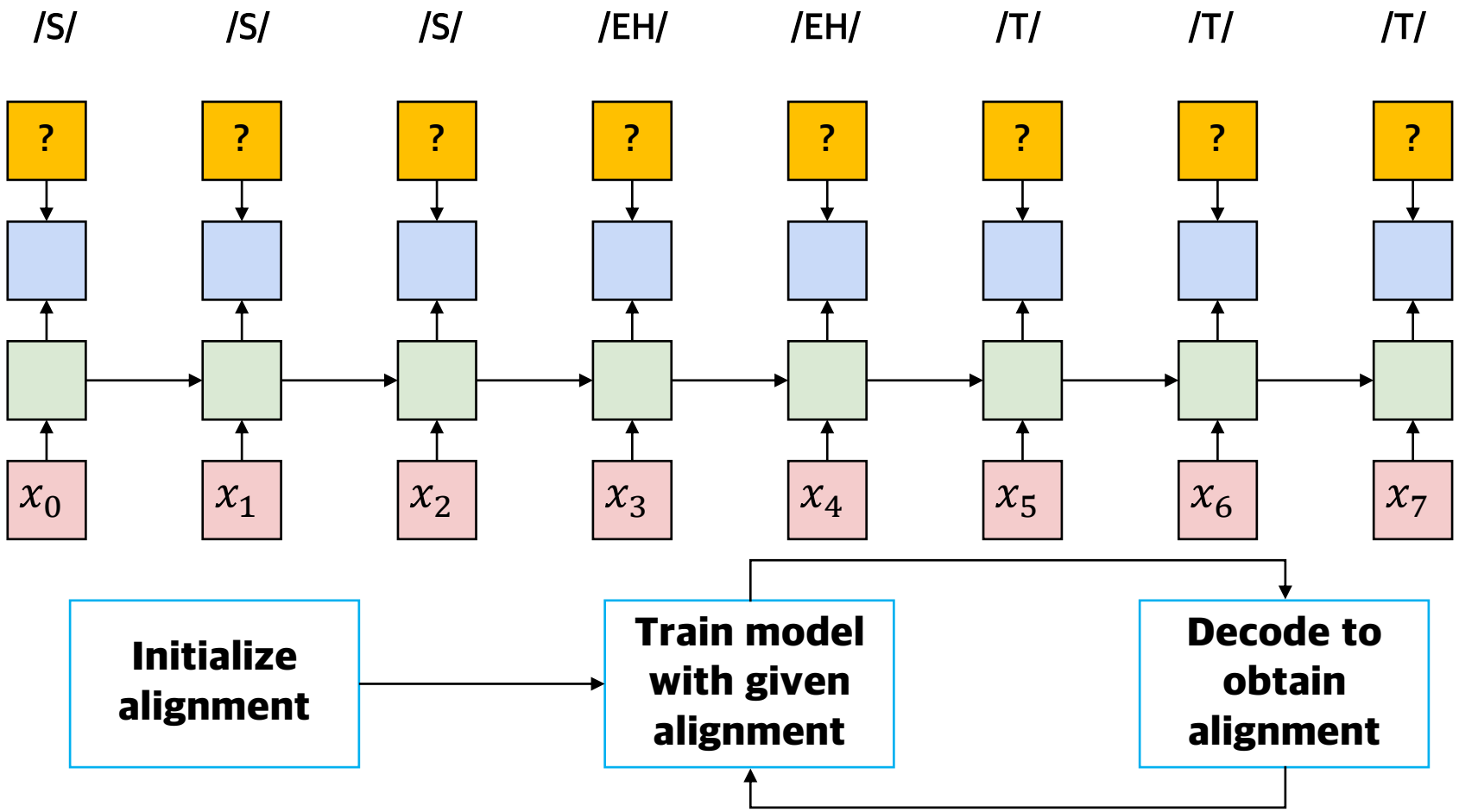
- Constraints
  - 가장 왼쪽 위에 첫번째 symbol이 위치
  - 마지막 symbol은 가장 오른쪽 아래에 위치
  - 나머지 symbol들은 왼쪽 위에서 오른쪽 아래로 monotonically 경로를 가져야 함
    - 즉 t시점에서 t+1시점으로 이동할 때에는 같은 행이나 다음 행으로만 이동가능
- Constraint를 통해서 target sequence의 expansion만이 decoding결과로 출력 가능
  - S E T 의 expansion만 decoding의 결과가 될 수 있음

6.2.2.10 Constraining the Alignment : Try 2



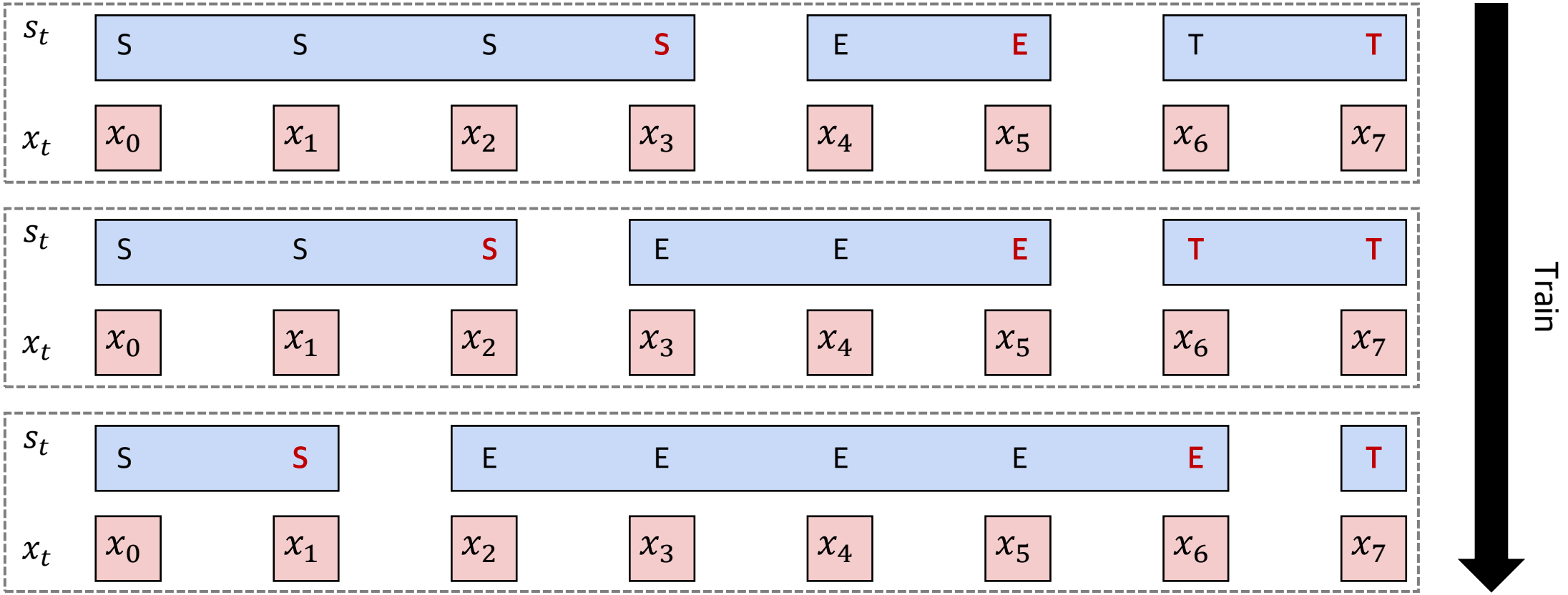
- Constraints를 모두 적용할 경우 위와 같이 생성가능한 경로의 decoding 결과는 모두 target sequence의 expansion
- Neural network를 통해 추정된 확률을 사용하여 그래프에서의 경로의 값을 계산 가능
- Most probable path는 dynamic programming 알고리즘을 통해서 효율적으로 계산 가능
  - HMM의 decoding에 사용된 Viterbi algorithm을 이용하여 most probable path를 decoding 가능

6.2.2.11 Iterative Estimate and Training



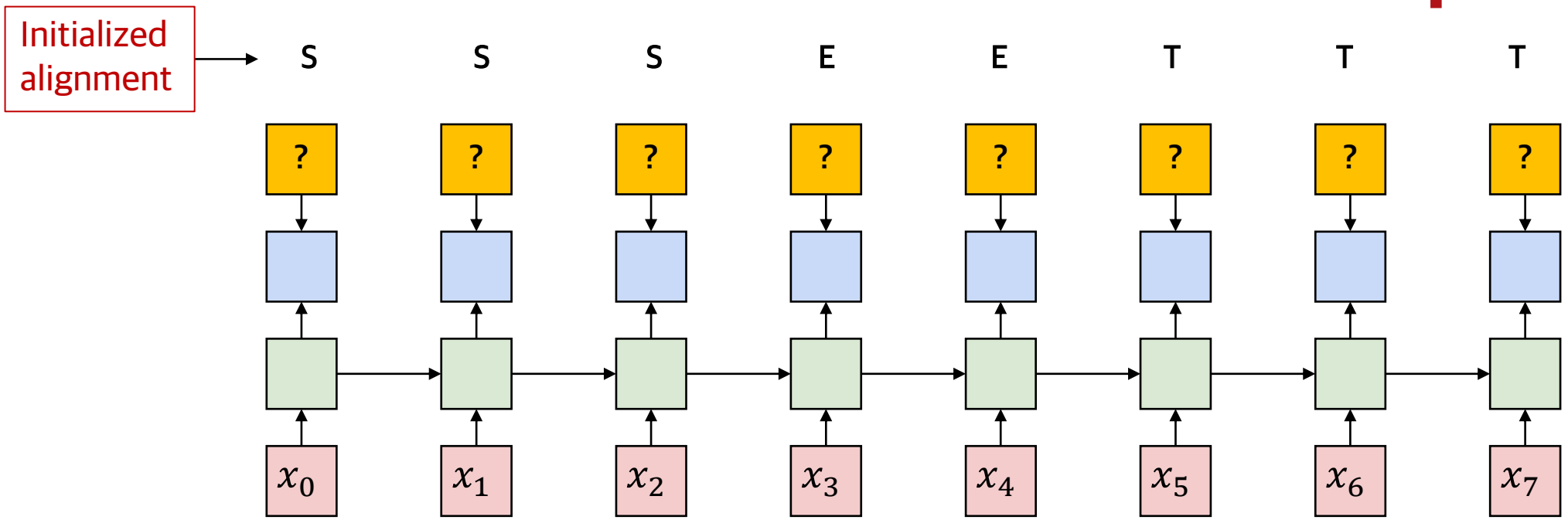
- 하나의 most probable path에 대해서 학습하고 다시 decode하는 과정을 거쳐서 네트워크를 학습

6.2.2.11 Iterative Estimate and Training



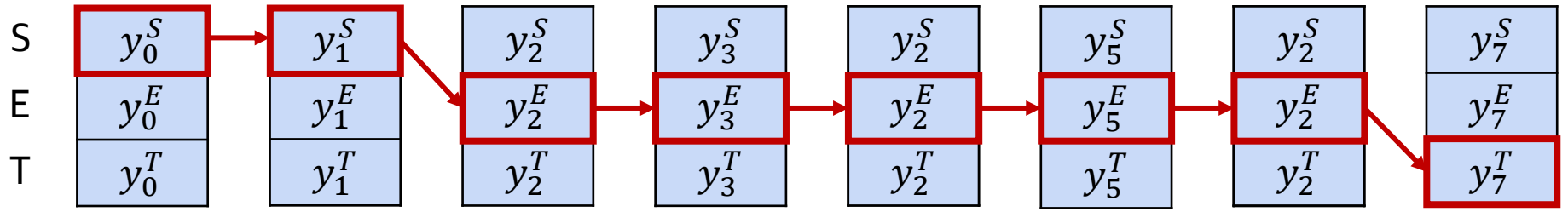
- 초기 alignment로 학습
- 학습을 통해 나온 결과를 decode하여 alignment를 다시 추정
- 이를 반복

6.2.2.12 Problem of Iterative Update



- 초기 alignment에 영향을 많이 받음
- 잘못된 local minima에 빠질 수 있음

6.2.2.12 Problem of Iterative Update



- 하나의 most probable path만을 alignment로 사용할 경우
  - 학습 초기 alignment의 영향을 크게 받음
  - Neural network의 초기 output에 따라 잘못된 학습을 할 수 있음

- Alternative view : target sequence에 대한 확률분포가 존재
  - 하나의 most probable path는 확률 분포에서 확률이 가장 높은 sample로 부터 만들어진 path

6.2.2.13 참고 : HMM Training

- Viterbi training algorithm
  - 현재의 모델로 Viterbi algorithm에 따라 전체 학습자료에 대해 segmentation을 수행
  - Segmentation 결과를 바탕으로 다음 장과 같이 모델 파라미터를 업데이트 함
  - $\prod_{r=1}^{R} \Pr(\mathbf{O}^r \mid M)$이 더 증가하지 않을 때 까지 위의 과정을 반복

6.2.2.13 참고 : HMM Training

- Viterbi training algorithm
  - a_{ij}는 아래와 같이 업데이트

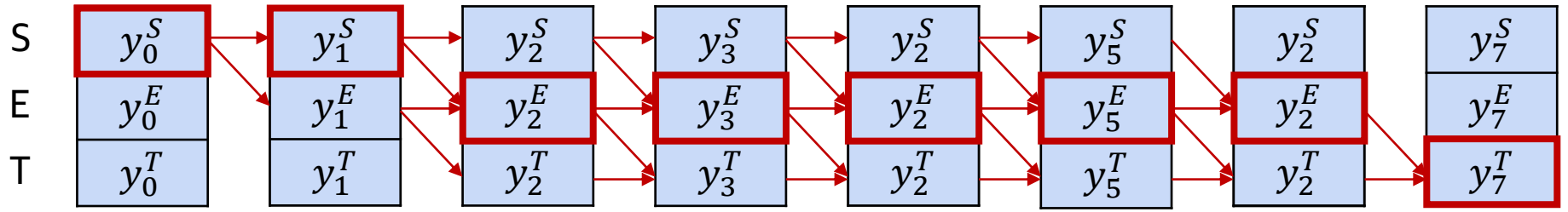
$$\hat{a}_{ij} = \frac{\text{Estimated number of transitions state } i \rightarrow \text{ state } j}{\text{Estimated number of transitions from state } i}$$

  - $b_j(o_t)$가 하나의 정규분포로 모델링 된 경우 mean과 variance는 아래와 같이 업데이트

$$\hat{\mu}_j = \frac{\text{Estimated sum of vectors emitted from } j}{\text{Estimated number of vectors emitted from } j}$$

$$\hat{\Sigma}_j = \frac{\text{Estimated sum of } (\boldsymbol{o}_t - \hat{\mu}_j)(\boldsymbol{o}_t - \hat{\mu}_j)'}{\text{Estimated number of vectors emitted from } j}$$

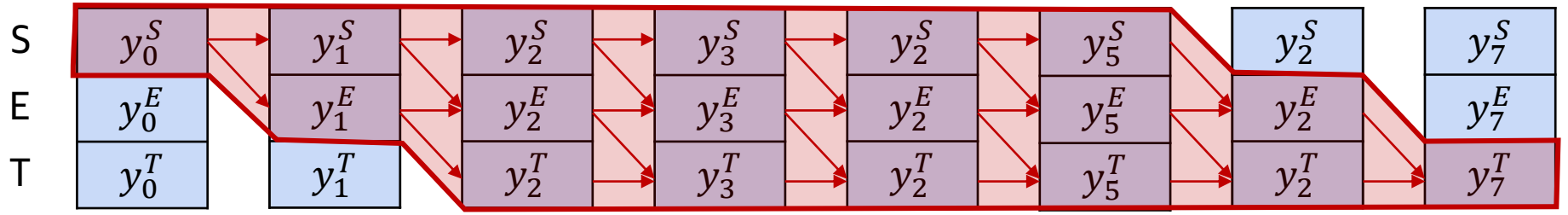
6.2.2.14 Alternative View



- 하나의 most probable path만을 alignment로 사용할 경우
  - 학습 초기 alignment의 영향을 크게 받음
  - Neural network의 초기 output에 따라 잘못된 학습을 할 수 있음

- Alternative view : target sequence에 대한 확률분포가 존재
  - 하나의 most probable path는 확률 분포에서 확률이 가장 높은 sample로 부터 만들어진 path

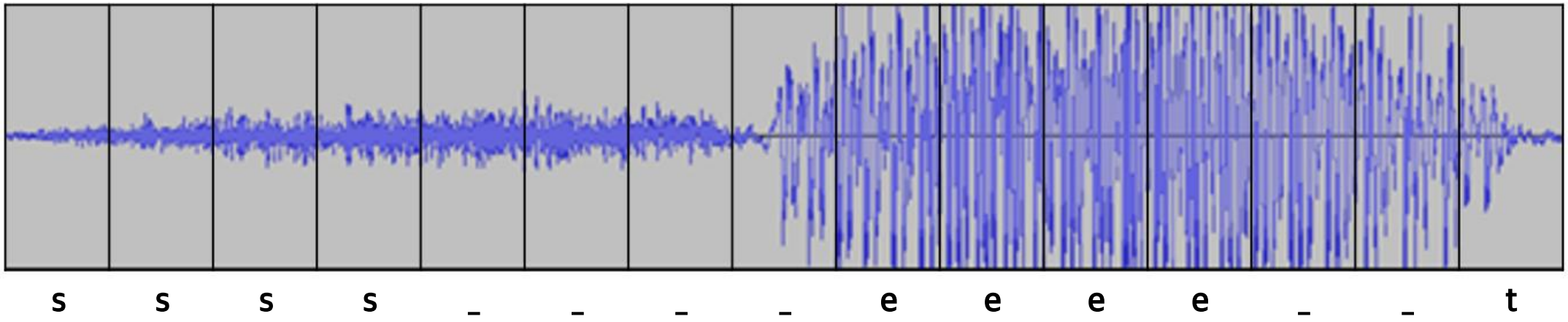
6.2.2.15 Averaging over all alignments



- Most probable path를 alignment로 사용하는 대신, 모든 가능한 경로의 expectation을 사용
  - alignment의 전체 확률 분포를 사용
  - 하나의 most probable path의 alignment만을 사용할 때의 문제점을 완화 가능

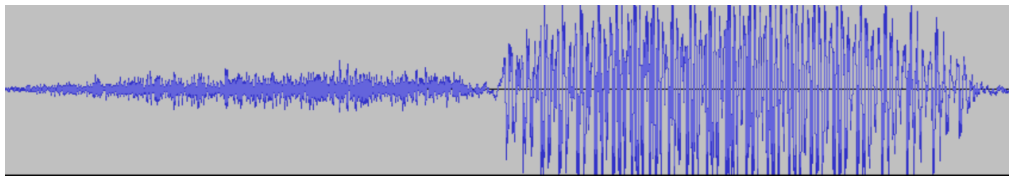
6.3 Connectionist Temporal Classification

- CTC (Connectionist Temporal Classification) (Graves, 2006)
  - 음성에 대하여 사전 정렬되지 않은 알파벳의 열을 학습할 수 있는 최초로 제안된 방법
  - 음성인식 학습에서 주어진 정답은 알파벳의 열 뿐임
    - 알파벳의 열을 하나의 label로 보고 학습할 수 없음
  - CTC는 U길이의 정렬되지 않은 label을 가능한 모든 경우의 T길이로 생성해 forward algorithm을 통해 likelihood를 계산함



CTC의 T길이에 대한 알파벳 생성 예시

6.3.1 Inference Step



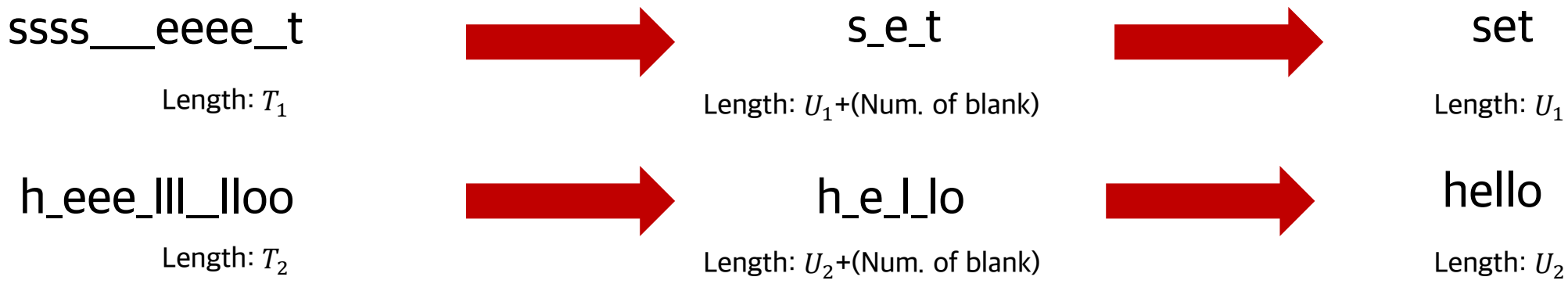
set.pcm

‘set’

transcription

- $\boldsymbol{L}$
  - Transcription을 grapheme(alphabet)의 sequence로 본다면 $L = \{ a, b, c \ \ldots, z, (space)\}$ 27개로 구성
  - Transcription의 길이는 U 로 표현하며, 통상적으로 $U \leq T$ 가 됨
    - 예를 들어, “set’ 을 0.3초간 발성했을 때 T 는 15개의 프레임으로 이루어짐
    - 또한 3개의 알파벳으로 이루어져 있으므로 U 는 3이 됨
      - 이 경우 L^U은 3개의 프레임에 대해 27개의 label의 가능한 경우의 수가 됨 (복잡도: 3^{27})
- $\boldsymbol{l} \in L^{\leq T}$
  - l은 입력된 음성에 대해, 모델에서 생성하는 transcription의 경로
    - 3^{27}개의 경우의 수 중, 음성인식 결과로 나오는 ‘set’을 말함
  - End-to-end로 음성인식을 수행하기 위해서는 T 와 U 의 길이 차이를 고려하여 모델링할 수 있어야 함

6.3.1 CTC : Inference Step



- $\mathcal{B}: L'^T \mapsto L^{\leq T}$
  - Blank를 포함한 sequence을 T 이하의 길이의 blank를 포함하지 않은 sequence로 변환하는 함수
  - 같은 label이 연속되는 경우 하나로 합침
    - 예를 들어, 10개의 프레임에 대해 'ssss___eeee_t'가 출력될 경우, 연속되는 알파벳을 합쳐 's_e_t'가 되고, blank를 제거하여 'set'이 됨
- Blank symbol은 연속되는 알파벳의 열과 실제 단어를 구분하기 위해 존재함
  - 예를 들어, 'hello'의 경우 'l'이 연속해서 나오는 경우 'll'이 정답인지 'l'이 정답인지 구분할 수 없음
  - 이 경우를 방지하기 위해 'l'사이에 blank symbol을 넣어 구분함

6.3.1 CTC : Inference Step

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x}) \quad \text{where } \mathbf{l} \in L^{\leq T}$$

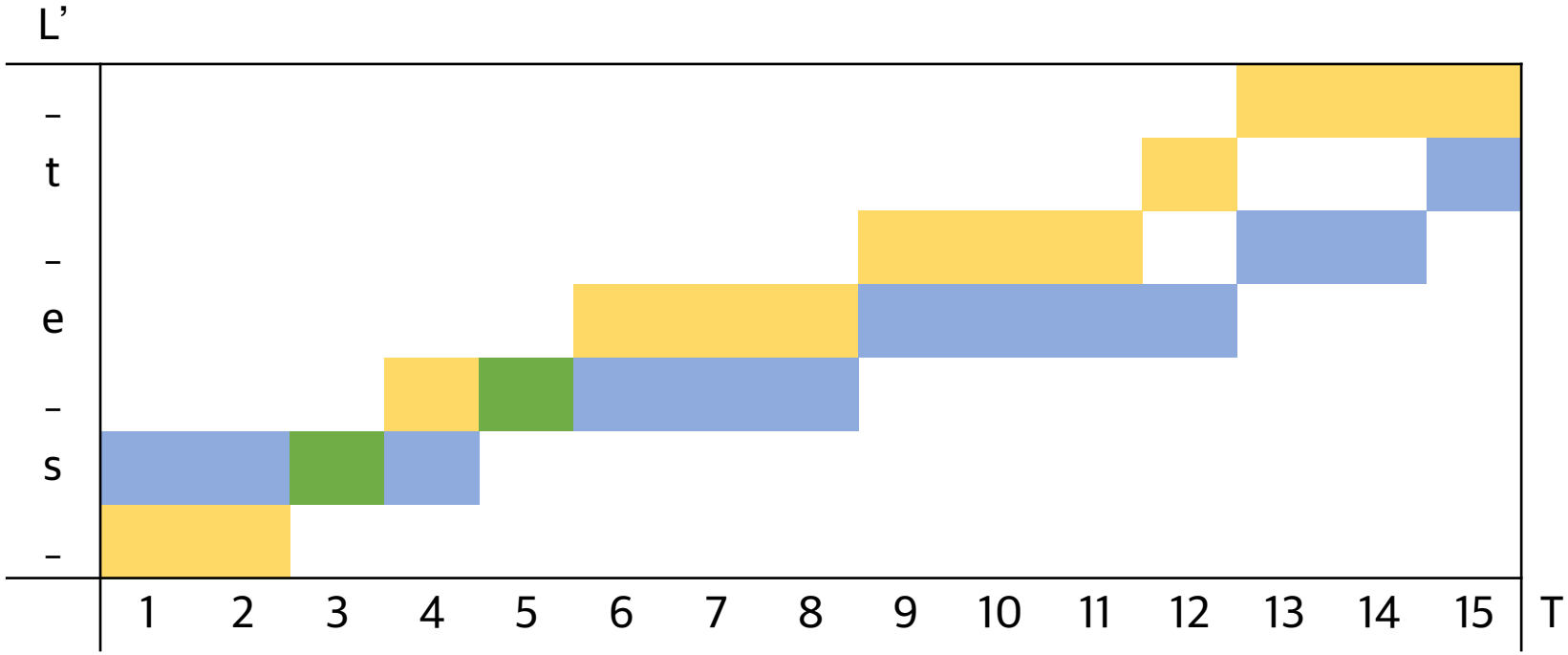
$$p(\pi \mid \mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t$$

- $\sum_{\pi \in B^{-1}(l)} p(\pi|X)$
  - π에 대하여 나타날 수 있는 모든 경우의 수를 더함
  - 음성 입력의 길이 T가 길어질 수록 l의 길이가 길어지고, $B^{-1}(l)$의 경우의 수가 증가(L'^T)
  - Dynamic programming 기법을 이용하여 이 복잡도를 $L' * T$로 감소시킴 (forward & backward algorithm)
    - $L' * T$ 의 2차원 테이블에서 각 grid의 α_n^t를 계산함 (n $\in N$, t $\in T$)

- $Y = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T\}$, $\boldsymbol{y}_t \in \mathbb{R}^{|L'|}$
  - $\boldsymbol{X}$ 에 대한 DNN의 출력 벡터
  - $\boldsymbol{y}_{\pi_t}^t$은 π_t에 대한 time t에서의 벡터를 말함
  - $\boldsymbol{y}_{\pi_t}^t$에 의해 forward algorithm을 계산함

6.3.1 CTC : Inference Step

- Example



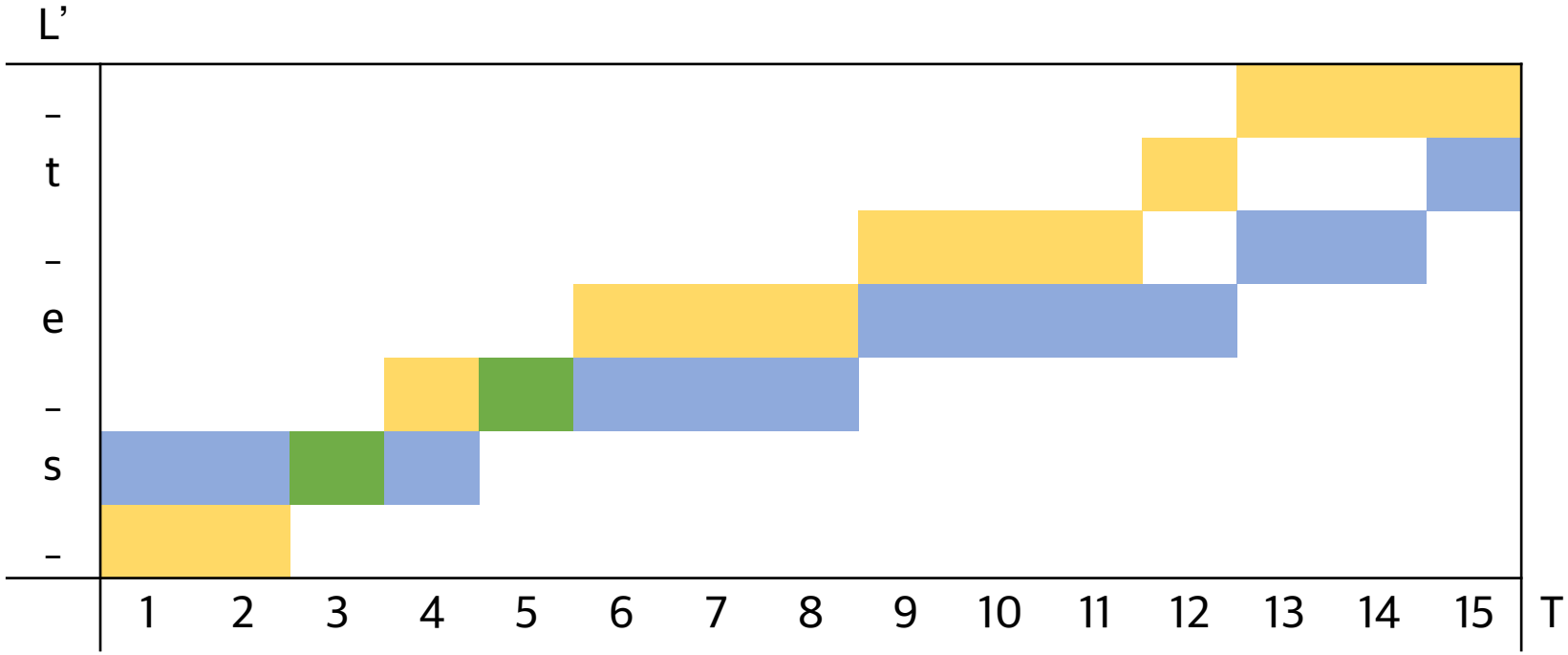
B 를 적용했을 때 'set'이 나올 수 있는 경우의 수 (대표 2개)

π_1 = 'ssss_ _ _ _eeee_ _t'

π_2 = '_ _s_ _eee_ _ _ t _ _ _'

6.3.1 CTC : Inference Step

- Example



Forward algorithm 예제

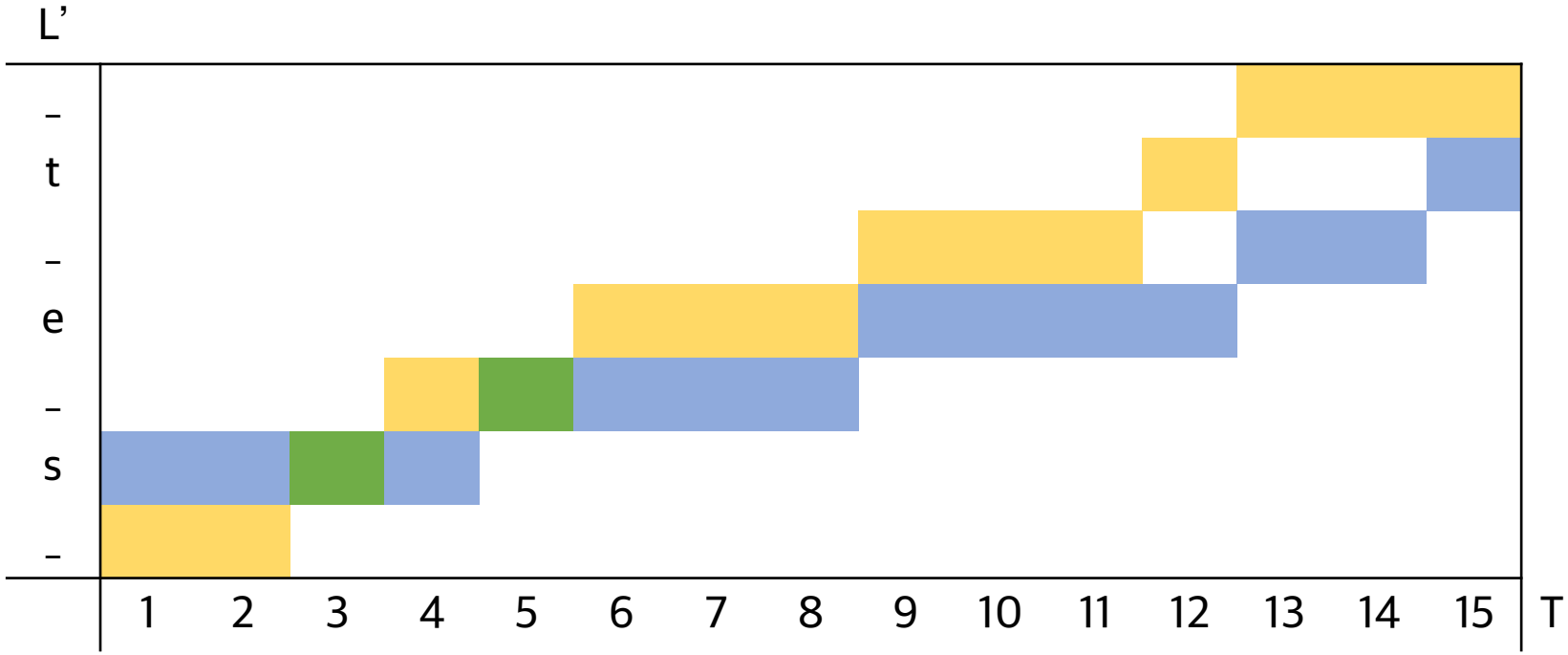
Time 3에서 발생할 수 있는 경로 중 '_ s'에 대한 α인 $\alpha_{_s}^{3}$를 계산

$\mathcal{B}^{-1}(\_s) = \{$'_ _ s', '_ s s', 's s s'$\}$

$\alpha_{\_s}^{3} = p(\_\_s|X) + p(\_ss|X) + p(sss|X) = y_{\_}^{1} * y_{\_}^{2} * y_{s}^{3} + y_{\_}^{1} * y_{s}^{2} * y_{s}^{3} + y_{s}^{1} * y_{s}^{2*} y_{s}^{3}$

6.3.1 CTC : Inference Step

- Example



Forward algorithm 예제

Time 3에서 발생할 수 있는 경로 중 '_e_t_' 혹은 '_e_t' 에 대한 β인 $\beta^3_{_e_t_}, \beta^3_{_e_t}$를 계산

$\mathcal{B}^{-1}(\_e\_t) = \{$'_ _ eee_ _ _ t _ _ _', ….$\}$

$\beta^3_{\_e\_t\_} = p($"_ _ eee_ _ _ t _ _ _" $\mid X) + \cdots = y^4_{\_} * y^5_{\_} * y^6_e * \cdots$

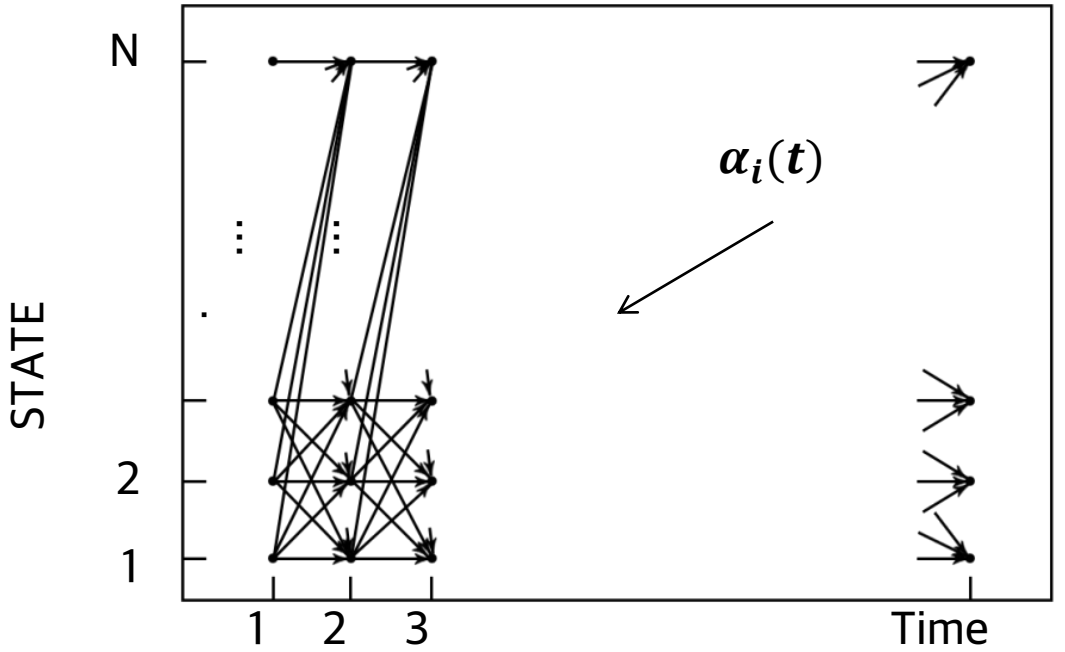
6.3.1.1 참고 : Computing Likelihood - Forward Algorithm

- Dynamic programming기법을 이용
  - Total likelihood method의 시간 복잡도를 $O(N^2T)$로 줄이는 방법
- Idea : 2차원 상에서의 $N*T$개의 각 격자점의 $\alpha_i(t)$값을 저장

$$\alpha_i(t) = \Pr\left(o_1 o_2 \ldots o_t, s(t) = i \mid \mathcal{M}\right)$$

- Algorithm
  - 초기화

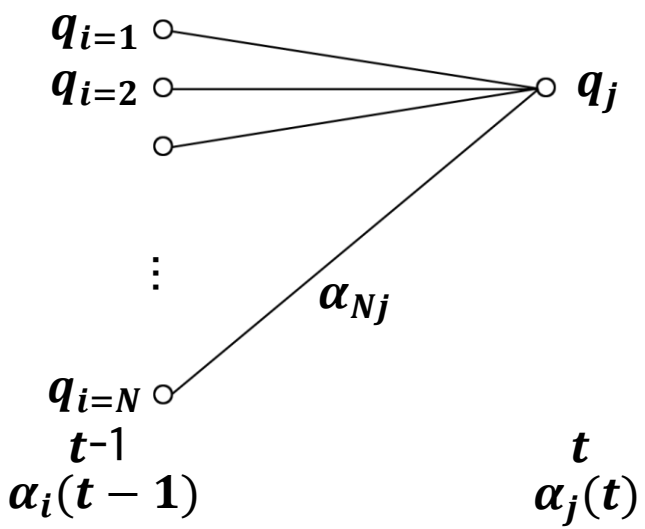
$$\alpha_1(1) = 1$$
$$\alpha_1(t) = 0 \quad t > 0$$
$$\alpha_i(1) = 0 \quad 1 < i \leq N$$



6.3.1.1 참고 : Computing Likelihood - Forward Algorithm

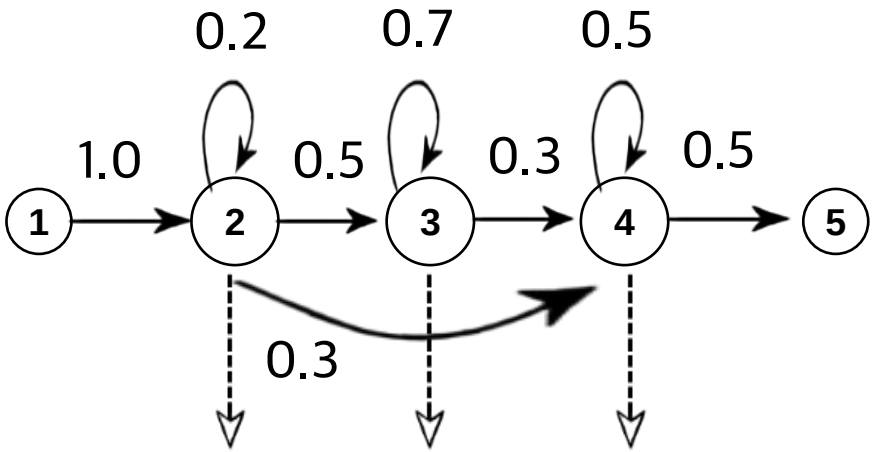
- Recursion

$$\alpha_j(t) = \left[\sum_{i=1}^{N} \alpha_i(t-1)a_{ij}\right] b_j(o_t)$$



6.3.1.1 참고 : Computing Likelihood - Forward Algorithm

- Example
  - HMM likelihood computation



time	o_t	$b_2(o_t)$	$b_3(o_t)$	$b_4(o_t)$
1	0.9	0.397	0.218	0.115
2	1.9	0.266	0.397	0.171
3	2.4	0.150	0.368	0.191
4	3.3	0.028	0.171	0.197

$$\mu_2 = 1.0 \quad \mu_3 = 2.0 \quad \mu_4 = 3.0$$
$$\sigma_2 = 1.0 \quad \sigma_3 = 1.0 \quad \sigma_4 = 2.0$$

6.3.1.2 참고 : Computing Likelihood - Example

- Forward algorithm

state					
4	0.000	0.000	0.0204	0.0077	0.0022
3	0.000	0.000	0.0788	0.0242	0.0029
2	0.000	0.397	0.0211	0.0006	0.000003
1	1.000	0.000	0.000	0.000	0.000
	0	1	2	3	4

Total $\boldsymbol{Pr(\mathbf{O}|\mathcal{M}) = 0.0011}$

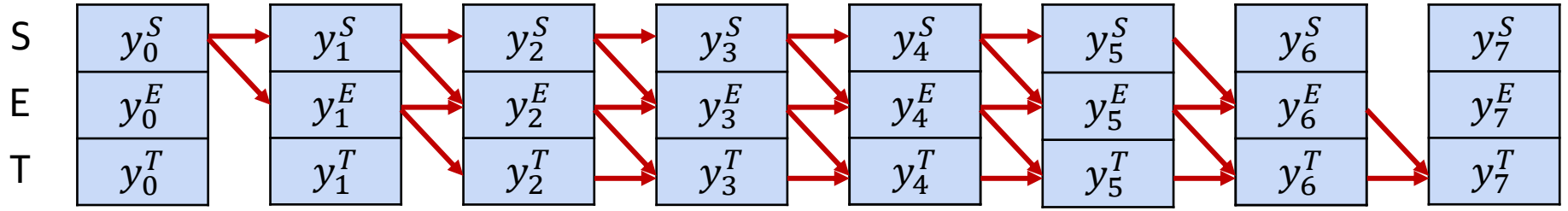
- Viterbi algorithm

state

4	0.000	0.000	0.0204	0.0045	0.0012
3	0.000	0.000	0.0788	0.0203	0.0024
2	0.000	0.397	0.0211	0.0006	0.000003
1	1.000	0.000	0.000	0.000	0.0000
	0	1	2	3	4

Max $\boldsymbol{Pr(\mathbf{O}|\mathcal{M}) = 0.0006}$, 최적의 상태열은= 1, 2, 3, 3, 4, 5.

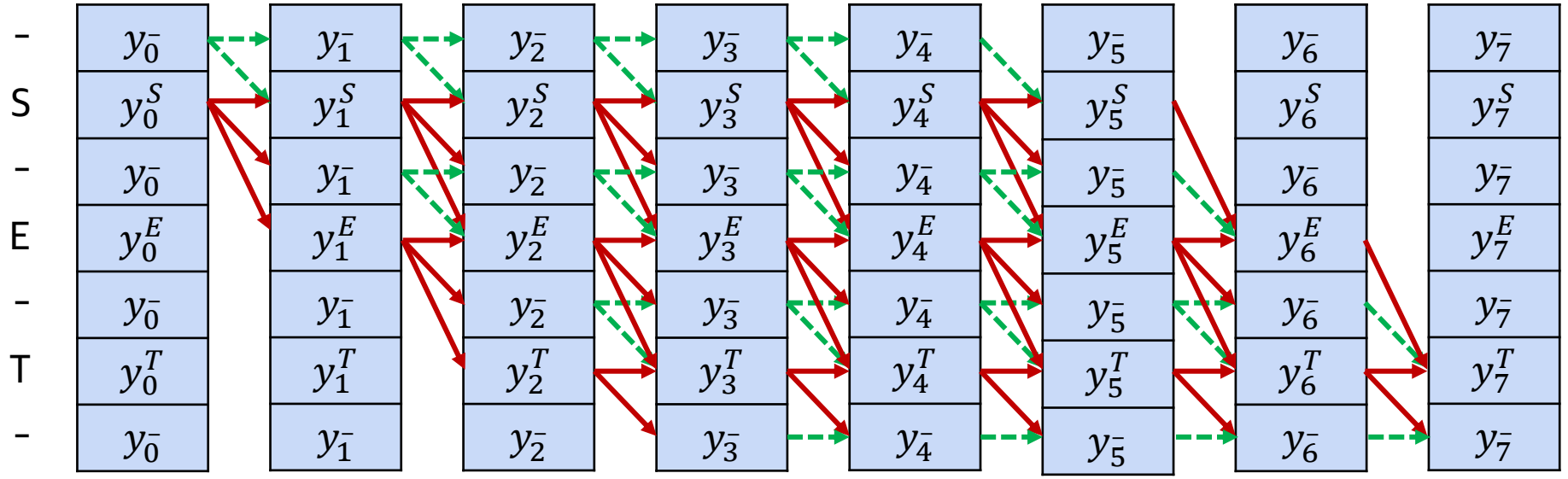
6.3.1.3 Recall : Constraining the Alignment : Try 2



- Constraints를 모두 적용할 경우 위와 같이 생성가능한 경로의 decoding 결과는 모두 target sequence의 expansion
- Neural network를 통해 추정된 확률을 사용하여 그래프에서의 경로의 값을 계산 가능
- Most probable path는 dynamic programming 알고리즘을 통해서 효율적으로 계산 가능
  - HMM의 decoding에 사용된 Viterbi algorithm을 이용하여 most probable path를 decoding 가능

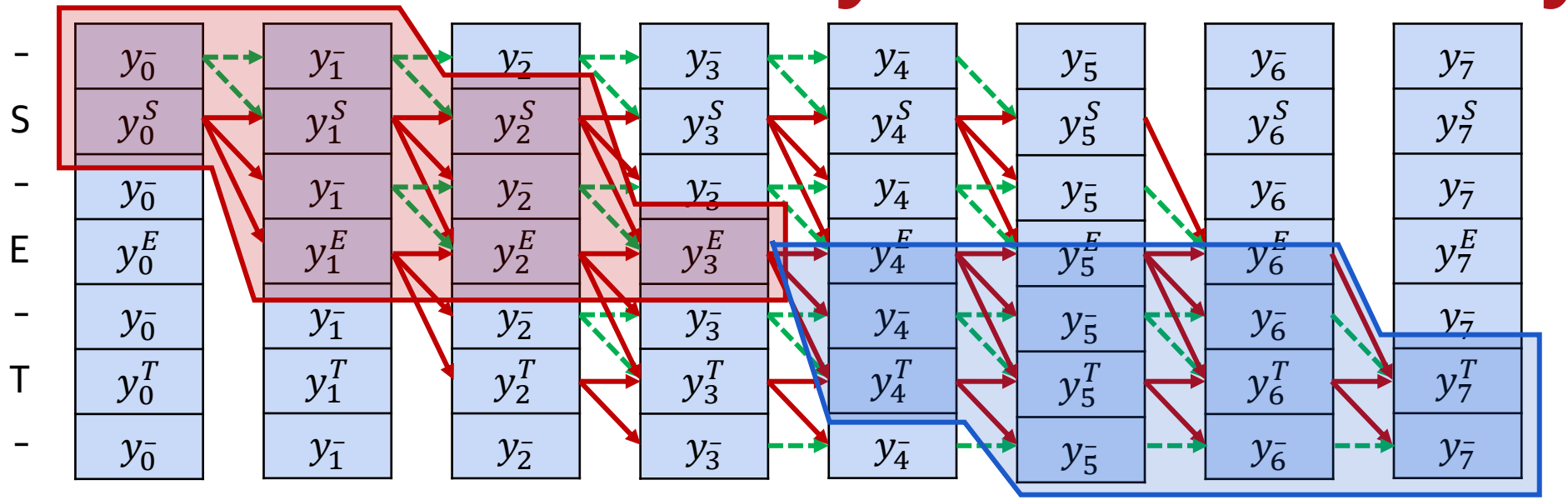
- 문제점
  - SEET, SETT와 같이 동일한 문자가 연속으로 반복 되는 단어 혹은 아무 발성도 없는 부분에 대해서는 적절치 않음

6.3.1.4 Constraining the Alignment : Try 3



- constraint : ‘blank’(-)라 불리는 특수기호를 추가한 구조로 변경
  - path의 시작은 blank 혹은 S_0에서만 가능, path의 끝은 S_T혹은 blank에서만 가능
    - 위 그림의 초록 점선은 blank가 이동가능한 경로를 표시

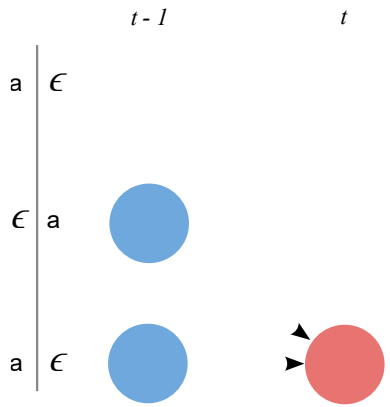
6.3.1.5 Posteriori Symbol Probability



- t=3 시점에서의 symbol E에 대한 확률 y_3^E는 빨간 부분과 파란 부분으로 나뉘어짐
  - 그림의 빨간 부분 : forward probability
  - 그림의 파란 부분 : backward probability

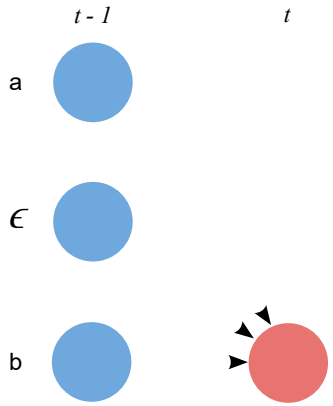
image from : https://distill.pub/2017/ctc/

6.3.2 Forward Probability α



- Case 1 :
  - t 시점의 symbol이 blank인 경우

$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \quad \cdot \quad p_t(z_s \mid X)$$

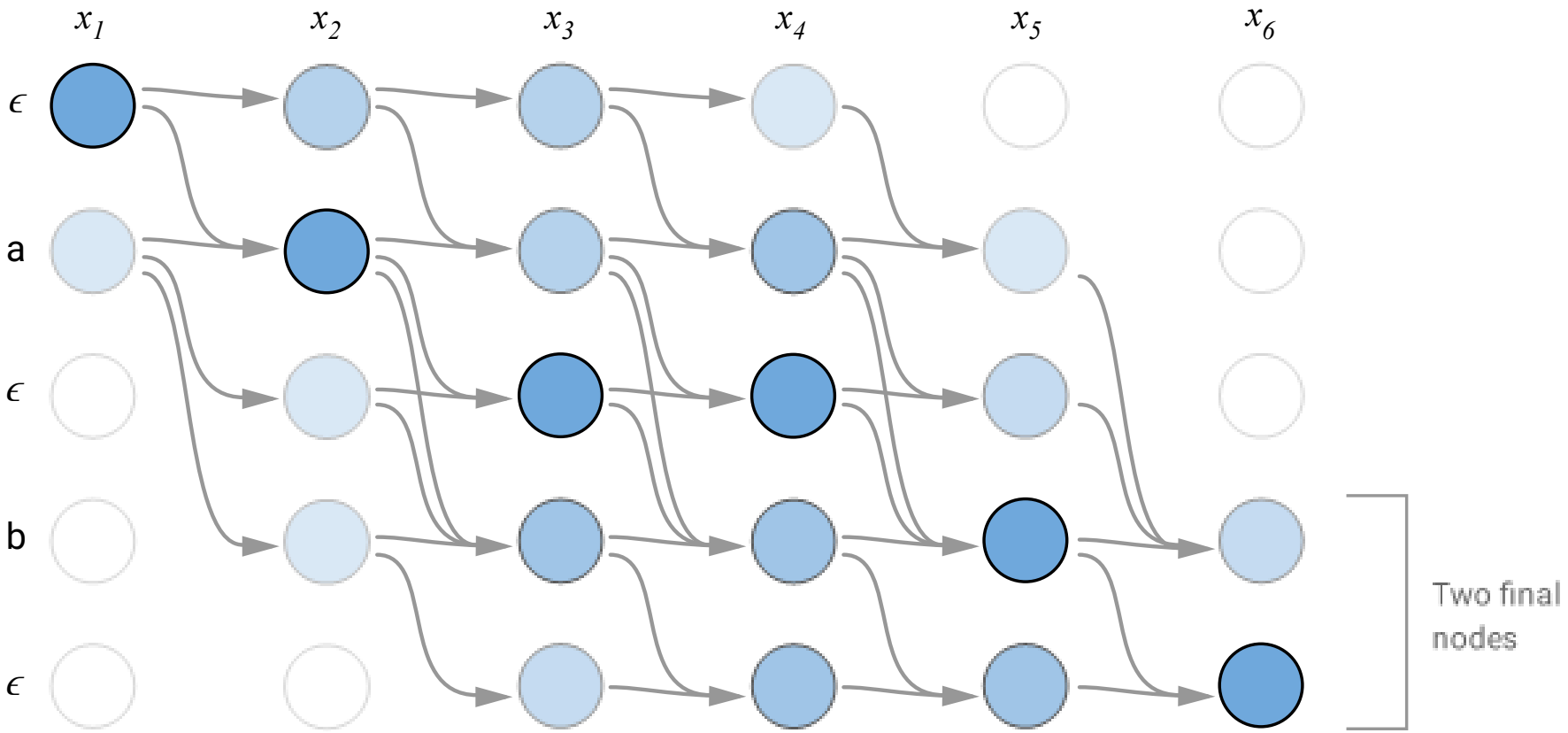


- Case 2 :
  - t 시점의 symbol이 blank가 아닐 경우

$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \quad \cdot \quad p_t(z_s \mid X)$$

image from : https://distill.pub/2017/ctc/

6.3.3 Forward Probability α



$$\alpha_t(s) = y^t_{l'_s} \begin{cases} \sum_{i=s-1}^{s} \alpha_{t-1}(i) & \text{if } l'_s = b \text{ or } l'_{s-2} = l'_s \\ \sum_{i=s-2}^{s} \alpha_{t-1}(i) & \text{otherwise} \end{cases}$$

6.3.3.1 참고 : Baum-Welch Algorithm

- Viterbi training algorithm은 특정 시점에 특정 상태에 있는 확률을 0 또는 1로 hard decision을 했음
- Baum-Welch algorithm에서는 특정 시점에 특정 상태에 $L_j(t)$의 확률로 있을 수 있다고 가정함
  - Viterbi algorithm은 Baum-Welch algorithm의 특수한 경우임
- 파라미터 값들은 가중 평균값으로 추정됨

$$\hat{\mu}_j = \frac{\sum_{r=1}^{R}\sum_{t=1}^{T^r} L^r_j(t)\boldsymbol{o}^r_t}{\sum_{r=1}^{R}\sum_{t=1}^{T^r} L^r_j(t)}$$

- Viterbi training algorithm과 같이, Baum-Welch algorithm도 반복적인 모델 업데이트로 학습 진행

6.3.3.1 참고 : Baum-Welch Algorithm

- $L_j(t)$

$$Pr(s(t) = j, \mathbf{O}|\mathcal{M}) = \alpha_j(t)\beta_j(t)$$

$$L_j(t) = Pr(s(t) = j|\mathbf{O}, \mathcal{M})$$

$$= \frac{1}{Pr(\mathbf{O}|\mathcal{M})}\alpha_j(t)\beta_j(t)$$

6.3.3.1 참고 : Baum-Welch Algorithm

- 비슷한 방법으로

$$Pr(s(t)=i, s(t+1)=j, \mathbf{O}|\mathcal{M}) = \alpha_i(t)a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1)$$

$$Pr(s(t)=i, s(t+1)=j|\mathbf{O}, \mathcal{M}) = \frac{1}{Pr(\mathbf{O}|\mathcal{M})}\alpha_i(t)a_{ij}b_j(\mathbf{o}_{t+1})\beta_j(t+1)$$

- 따라서, a_{ij}에 대한 재추정 식은 아래와 같다

$$\hat{a}_{ij} = \frac{\sum_{r=1}^{R}\frac{1}{Pr(\mathbf{O}^r|\mathcal{M})}\sum_{t=1}^{T^r}\alpha_i^r(t)a_{ij}b_j(\boldsymbol{o}_{t+1}^r)\beta_j^r(t+1)}{\sum_{r=1}^{R}\sum_{t=1}^{T^r}L_i^r(t)}$$

- $1 \le i,\ \ j < N$인 경우, final state transition은 다음과 같다

$$\hat{a}_{iN} = \frac{\sum_{r=1}^{R}L_i^r(T^r)}{\sum_{r=1}^{R}\sum_{t=1}^{T^r}L_i^r(t)}$$

image from : http://www.tbluche.com/ctc_and_blank.html

6.3.4 CTC Training

- CTC forward-backward probabilities at different stage of CTC training.
  - ∅ means blank