Connectionist Temporal Classification

- Labelling unsegmented sequence data with RNN -

발표자 : 김영웅

2020.12.29 (TUE)

Background

Sequence labelling? (손 글씨, 음성, 모션 인식 등) -> HMM

" 과거의 정보는 현재에만 의존한다."

1. 데이터에 대한 사전지식이 요구되고, 모델 설계 난이도가 높다.
2. Dependency를 가정한다.

RNN: 데이터에 대해 사전 지식이 필요 없고, 독립적으로 학습되기에 성능이 우수하다.

RNN의 목적함수는 매 학습 스텝마다 독립적으로 변한다.

rnn의 학습데이터는 이미 클래스별로 구분 지어져 있어야 하고, 출력은 후처리를 해줘야하기 때문에 unsegmented sequence labelling에 바로 적용하기 어렵다.

Background

그래서 해당 논문은 RNN의 클래스 분리 필요성과 후처리 필요성을 없앤 체로
labelling하는 방법을 제안한다.

basic 아이디어는 input sequence에 대해서 가능한 모든 label sequence
조합을 구해 그 확률 분포로 모델의 출력을 이해하는 것이다.

CTC는 학습데이터에 Label만 있고, 해당 label에 맞는 클래스의 위치는 모르는 Unsegmented sequence
데이터의 학습을 위해 사용되는 기법이다.

CTC를 이용해 모든 가능한 순서에서 분류 확률을 최대화할 수 있는 모델을 학습한다.
네트워크의 출력은 라벨 시퀀스에 대한 조건부 확률로 변환한 다음 편집 거리(edit distance)를 최소화 하는 가장 좋은 순서를 선택한다

$$LER(h, S') = \frac{1}{Z} \sum_{(\mathbf{x},\mathbf{z}) \in S'} ED(h(\mathbf{x}))$$

$$\mathbf{z} = (z_1, z_2, ..., z_U)$$
$$\mathbf{x} = (x_1, x_2, ..., x_T),$$

LER: label error rate: target S′에 대한 classifier h의 edit distance
Z : target label의 총 수

네트워크 출력은 label과 input sequenc의 edit distance를 최소화하는 순서를 선택한다.

The aim is to use S to train a temporal classifier $h : \mathcal{X} \mapsto \mathcal{Z}$ to classify previously unseen input sequences in a way that minimises some task specific error measure.

Connectionist Temporal Classification

$L: label\ set$

L': label + $'\varepsilon'$, (ε $= blank$)

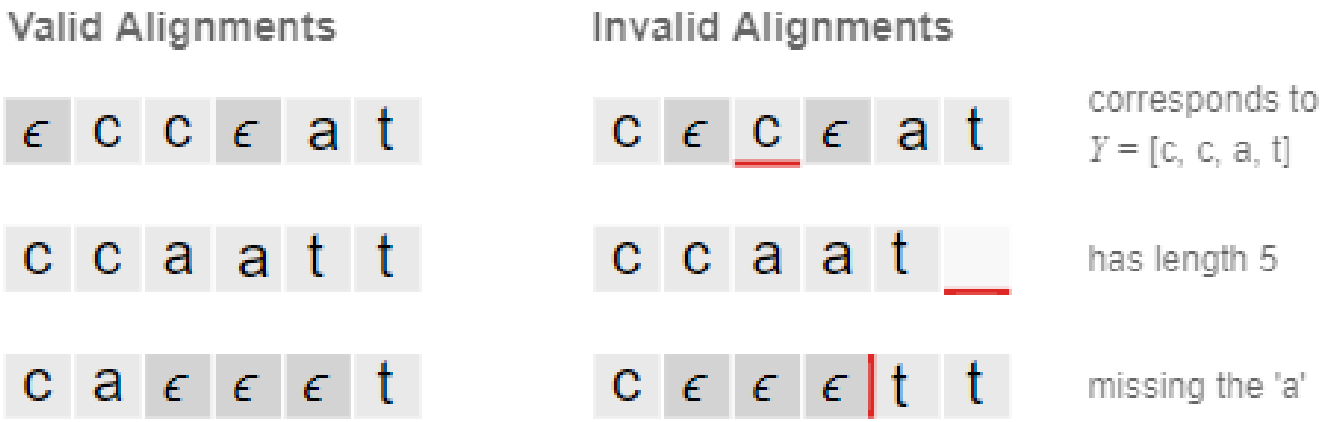
$$p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \ \forall \pi \in L'^T.$$

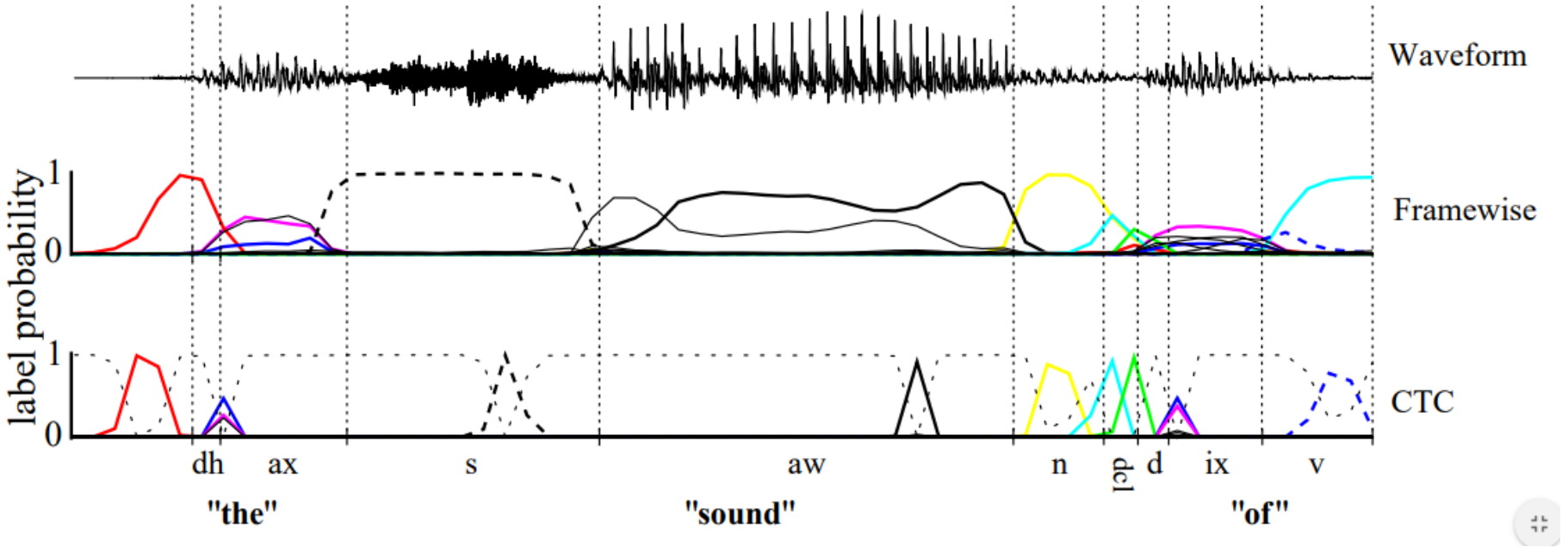
$$(\mathbb{R}^m)^T \mapsto (\mathbb{R}^n)^T$$

T : input sequence length

π = 가능한 label aligning의 한 경우 = path

y_k^t: 't' $time\ step$에 $label$ 'k'에 대한 확률





Framewise :직접 labelling. 같은 음소 안에서도 확률분포가 달라진다, 구간을 잘못 정렬
사용하기 전에 후처리가 필요하다.

CTC: blank가 대부분이고, 삼각형 형태로 특정 부분에서만 음소의 확률분포가 나타난다.
여러 음소가 연음이 되는 경우, 이중 삼각형태로 해당 음소 확률 값을 출력

β 함수 : Many to one mapping

$$\mathcal{B}(a-ab-) = \mathcal{B}(-aa--abb) = aab$$

$$p(l|x) = \sum_{\pi \in \beta^{-1}(l)} p(\pi|x) = \sum_{\pi \in \beta^{-1}(l)} \prod_{t=1}^{T} y_{\pi t}^{t} \qquad p(\pi|\mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^{t}, \ \forall \pi \in L'^{T}.$$

존재 가능한 모든 path들에 대해서, input - output class의 mapping의 확률을 전부 곱한 것들을 더한 값으로 input에 대한 label의 조건부 확률을 정의한다.

Constructing the Classifier

$$h(\mathbf{x}) = \arg \max_{\mathbf{l} \in L^{\leq T}} p(\mathbf{l}|\mathbf{x}).$$

Classifier의 출력 값은 해당 input sequence에 대한 label의 조건부 확률을 최대화 시키는 함수로 정의한다.

어떻게 최대화 시킬 것인가?

1. Best path decoding

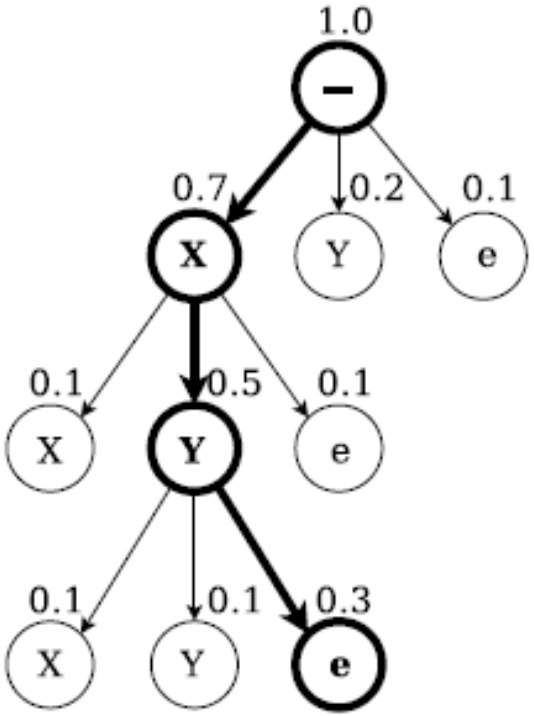
가장 확률이 높은 path의 labelling을 기준으로 concatenation
H(x)를 정의하는 방법
-> 최적의 labelling을 보장하지는 않는다. 그냥 가정 일뿐

$$h(\mathbf{x}) \approx \mathcal{B}(\pi^*)$$
$$\text{where } \pi^* = \arg \max_{\pi \in N^t} p(\pi|\mathbf{x}).$$

Constructing the Classifier

2. Prefix search decoding
충분한 시간이내에 forward – backward algorithm을 이용하여
Optimal한 labelling path를 찾는 방법: 전수조사와 비슷

-> sequence가 길어질 때, 소요 시간이 exponential하게 증가하는 문제



노드 위의 숫자는 하위 노드들의 확률의 합이다.

탐색 시간이 충분히 주어진다면 이 방법은 항상 적절한 라벨링을 찾을 수 있다.
그렇지만 시퀀스가 길어질수록 탐색할 노드의 수는 지수로 증가하는 문제가 있다.
그래서 이문제를 해결하는 적절한 방법을 논문에서 제안한다.

Constructing the Classifier

$$\alpha_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{1:t})=\mathbf{l}_{1:s}}} \prod_{t'=1}^{t} y^{t'}_{\pi_{t'}}.$$

t에서 모든 paths 의 시퀀스 1부터 s 까지의 확률

$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \left(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\right) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

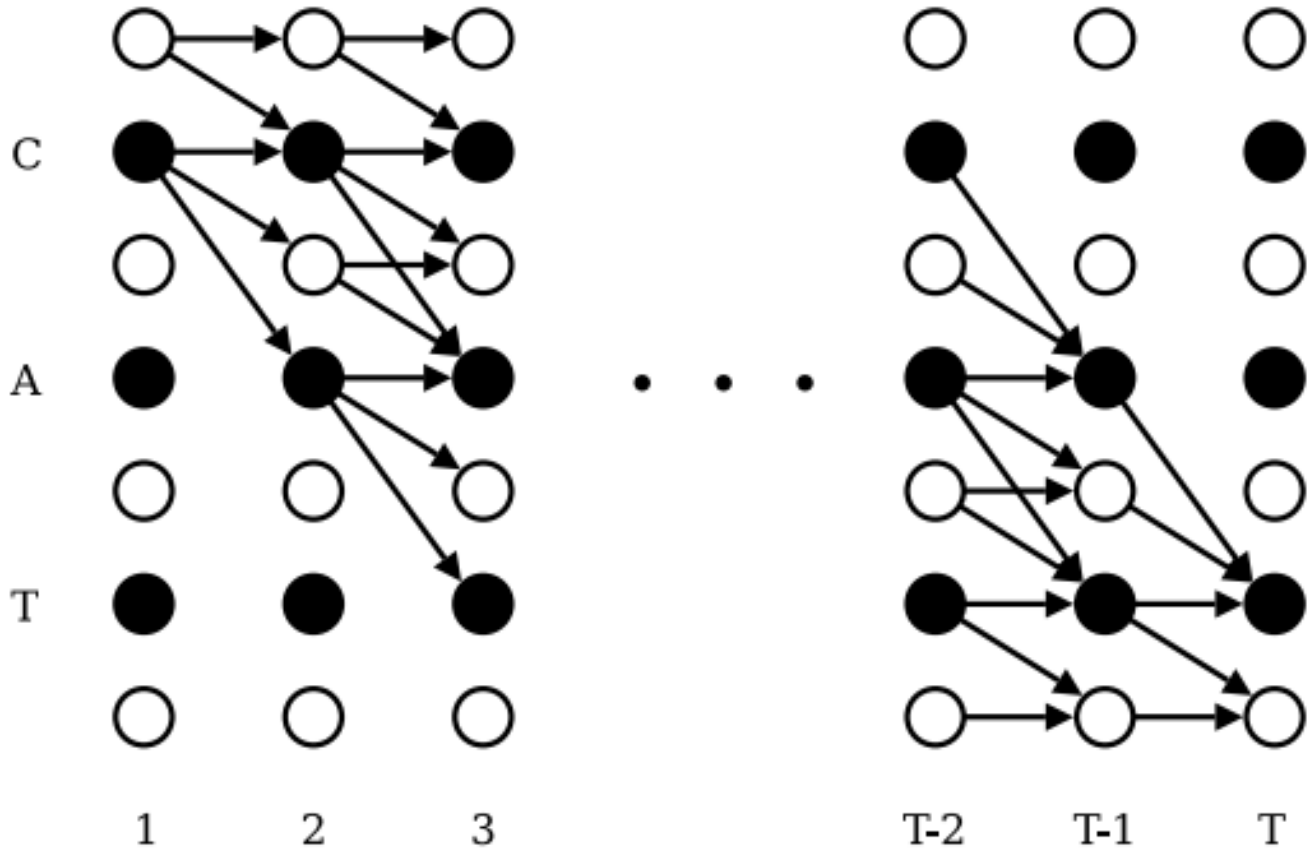
$$\bar{\alpha}_t(s) \stackrel{\text{def}}{=} \alpha_{t-1}(s) + \alpha_{t-1}(s-1).$$

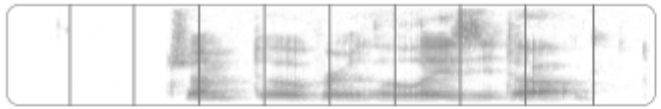
$$p(\mathbf{l}|\mathbf{x}) = \alpha_T(|\mathbf{l}'|) + \alpha_T(|\mathbf{l}'|-1).$$

최종 노드 2개에 더해진 확률의 합이 input sequence 에 대한 label의 조건부 확률

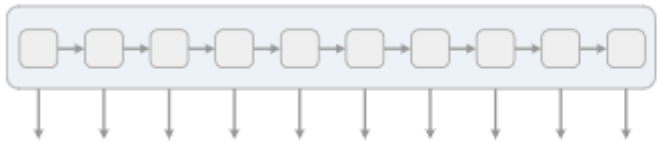
Dynamic programming(forward – backward)
L' = 2L+1

key insight is that if two alignments have reached the same output at the same step, then we can merge them.





We start with an input sequence, like a spectrogram of audio.



The input is fed into an RNN, for example.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
l	l	l	l	l	l	l	l	l	l
o	o	o	o	o	o	o	o	o	o
ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

h	e	ϵ	l	l	ϵ	l	l	o	o
h	h	e	l	l	ϵ	ϵ	l	ϵ	o
ϵ	e	ϵ	l	l	ϵ	ϵ	l	o	o

With the per time-step output distribution, we compute the probability of different sequences

h	e	l	l	o
e	l	l	o	
h	e	l	o	

By marginalizing over alignments, we get a distribution over outputs.

Backward variable Beta Forward variable과 유사

$$\beta_t(s) \stackrel{\text{def}}{=} \sum_{\substack{\pi \in N^T: \\ \mathcal{B}(\pi_{t:T}) = \mathbf{l}_{s:|\mathbf{l}|}}} \prod_{t'=t}^{T} y^{t'}_{\pi_{t'}}$$

$$\beta_t(s) = \begin{cases} \bar{\beta}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s+2} = \mathbf{l}'_s \\ \big(\bar{\beta}_t(s) + \beta_{t+1}(s+2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases} \quad \text{where} \tag{10}$$

$$\bar{\beta}_t(s) \stackrel{\text{def}}{=} \beta_{t+1}(s) + \beta_{t+1}(s+1). \tag{11}$$

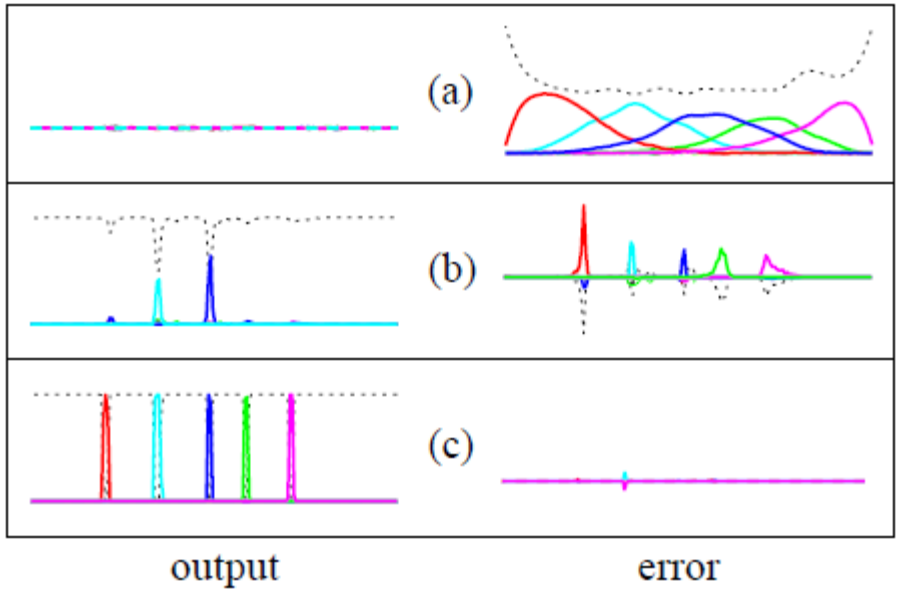
Variable이 너무 작기 때문에 실제 프로그램에서는underflow 발생
-> 비율로 rescaling

$$C_t \stackrel{\text{def}}{=} \sum_s \alpha_t(s), \qquad \hat{\alpha}_t(s) \stackrel{\text{def}}{=} \frac{\alpha_t(s)}{C_t}$$

$$D_t \stackrel{\text{def}}{=} \sum_s \beta_t(s), \qquad \hat{\beta}_t(s) \stackrel{\text{def}}{=} \frac{\beta_t(s)}{D_t}$$

$$ln(p(\mathrm{l}|\mathrm{x})) = \sum_{t=1}^{T} ln(C_t)$$

Evaluating Maximum likelihood error



학습이 진행됨에 따라 Sequence 전반에 있던 에러가 점차 0에 수렴하는 것을 확인 할 수 있다.

System	LER
Context-independent HMM	38.85 %
Context-dependent HMM	35.21 %
BLSTM/HMM	33.84 ± 0.06 %
Weighted error BLSTM/HMM	31.57 ± 0.06 %
CTC (best path)	31.47 ± 0.21 %
CTC (prefix search)	30.51 ± 0.19 %

Prefix search decoding이 가장 성능이 좋다.
같은 CTC 모델 중에서도 Prefix search decoding이 best path보다 약간 더 좋다.

CTC problem – Conditional independent problem

현재 라벨의 추론이 이전 라벨 추론과 독립적이다.

음성의 경우 시계열 데이터의 정보를 받지 않고, 발음에 의존적이다.