CTC algorithm: Connectionist Temporal Classification

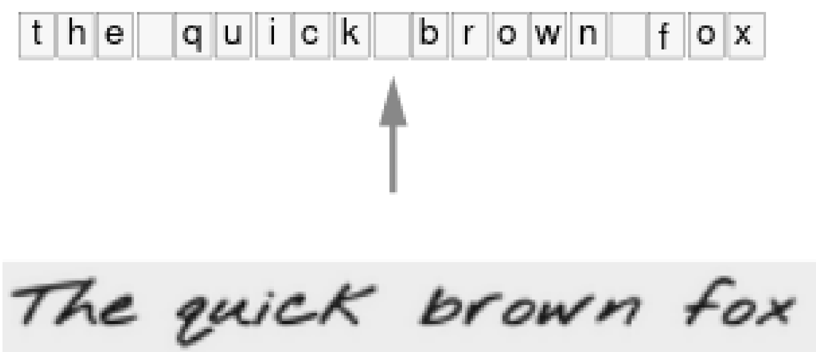
Александра Муравьёва

НИУ ВШЭ

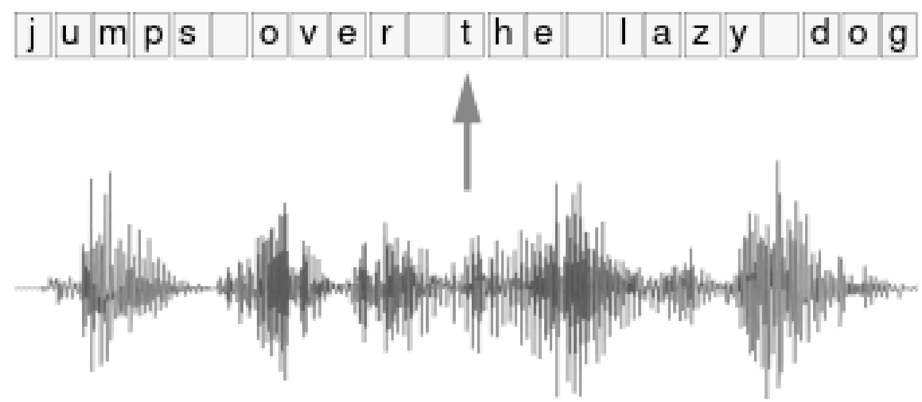
21 мая 2019 г.

Применение

- CTC был предложен в статье Graves et al. (2006)
- Применяется в Speech Recognition
- Может применяться в любой задаче, где несегментированному входу нужно сопоставить последовательность токенов



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



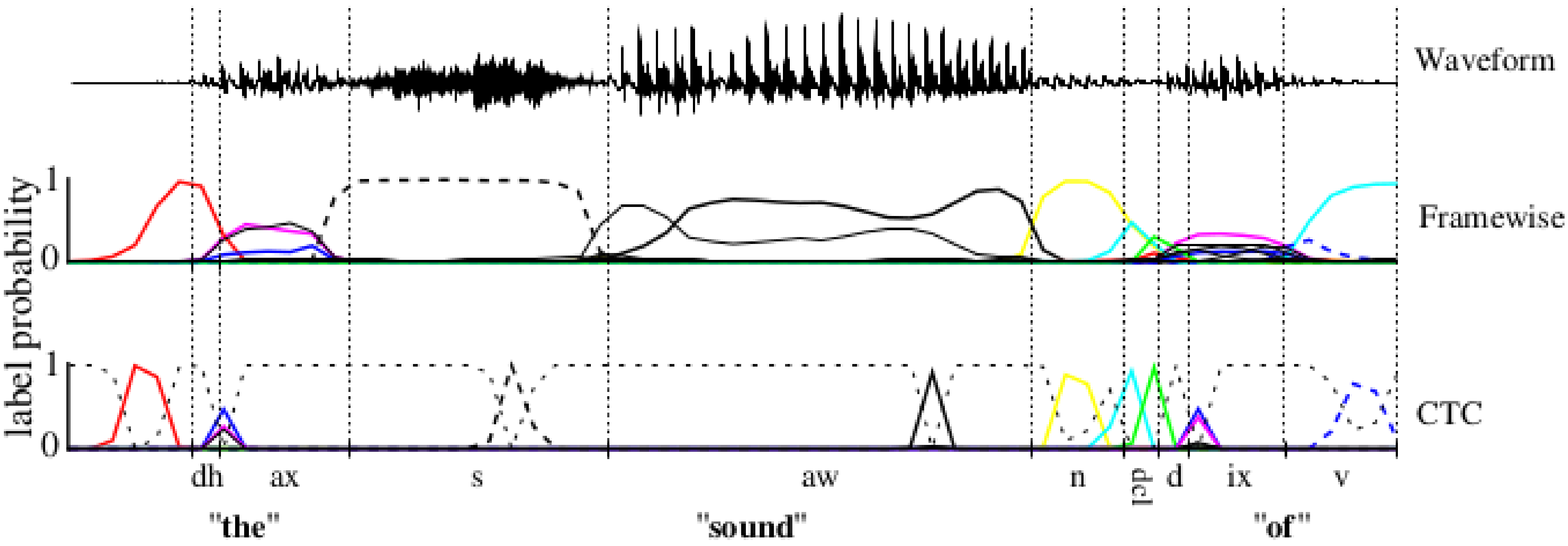
Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

Что было до этого

HMM, CRF и гибриды с RNN:

- Требуют task-specific knowledge (выбрать input features для CRF, смоделировать состояния для HMM)
- Не обходятся без сегментации
- И т.д.

  Самые успешные – гибриды: HMM сегментирует, а также превращает выходы RNN в последовательность лейблов



Основная идея

Выходы сети должны давать распределение вероятностей всех возможных разметок при условии данной входной последовательности

- Имея это распределение вероятностей, можно построить целевую функцию, которая будет впрямую максимизировать вероятность правильной разметки
- Эта целевая функция будет дифференцируема, просто обучим RNN с помощью BPTT

Подробности

- Используем RNN (в статье использовался BLSTM)
- Её выходной слой – softmax с размерностью |L|+1, где L – это множество всех возможных лейблов
- Лишний – лейбл ‘blank’, пустой
- Активации y^t_k – вероятность лейбла k в момент t

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

Вероятность **пути (path)** – последовательности лейблов (включая пустой) π при условии входа x
Пример пути: a_aab__c

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$$

Вероятность **разметки** – последовательности лейблов (без пустого) π при условии входа x
Пример разметки: aabc

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

Выход алгоритма – наиболее вероятная разметка

Хотим уметь предсказывать пустой лейбл:
разметку l меняем на разметку l’, где в начале,
в конце и между двумя лейблами пустой

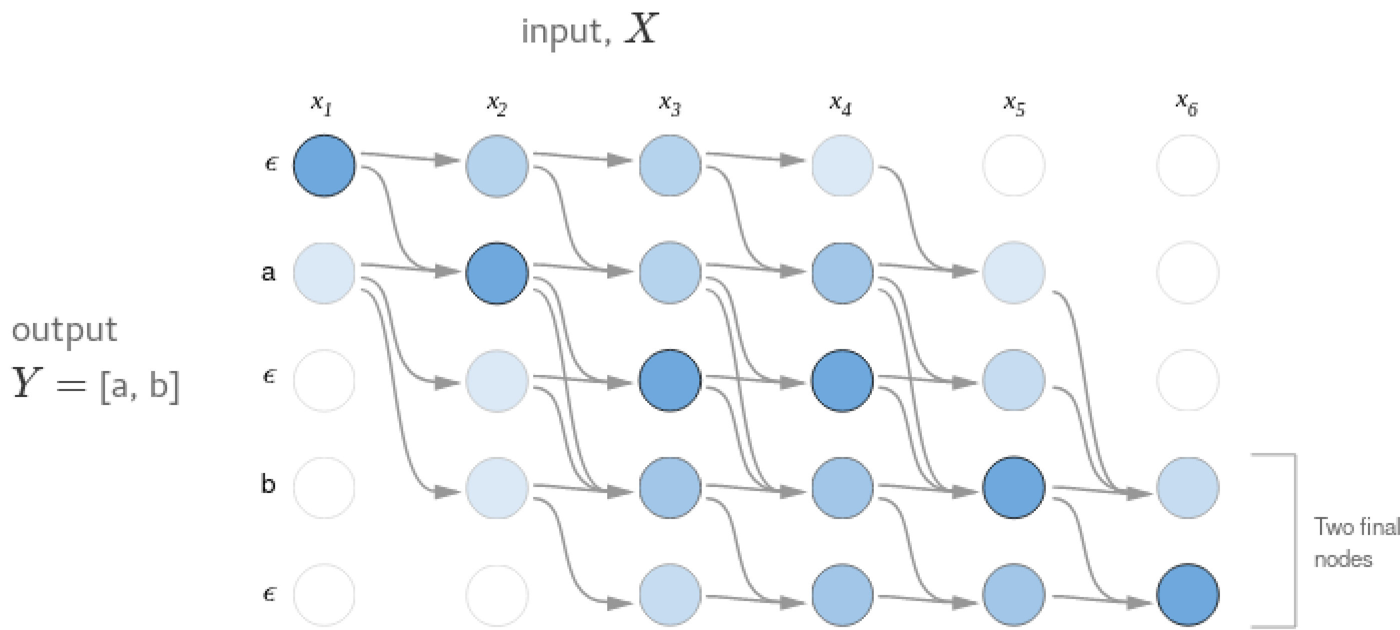
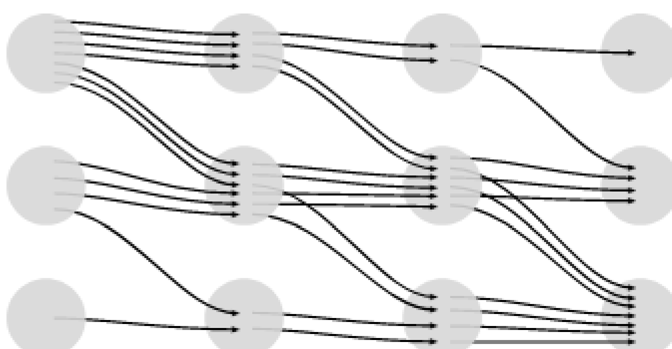


Иллюстрация того, как разные пути приводят к одной
и той же разметке ab

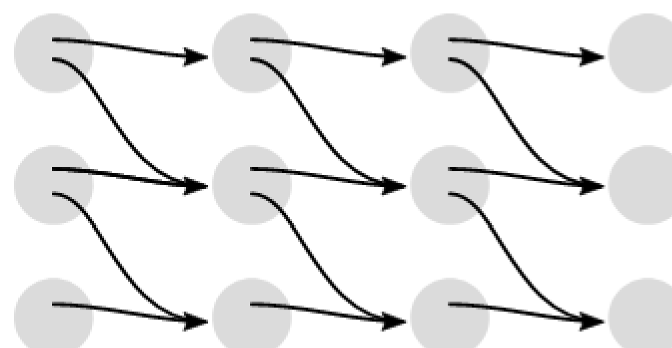
Если будем суммировать все пути, выйдет медленно.

$$p(\mathbf{l}|\mathbf{x}) = \sum_{\pi \in \mathcal{B}^{-1}(\mathbf{l})} p(\pi|\mathbf{x})$$

Можем использовать динамическое программирование



Summing over all alignments can be very expensive.



Dynamic programming merges alignments, so it's much faster.

Два метода

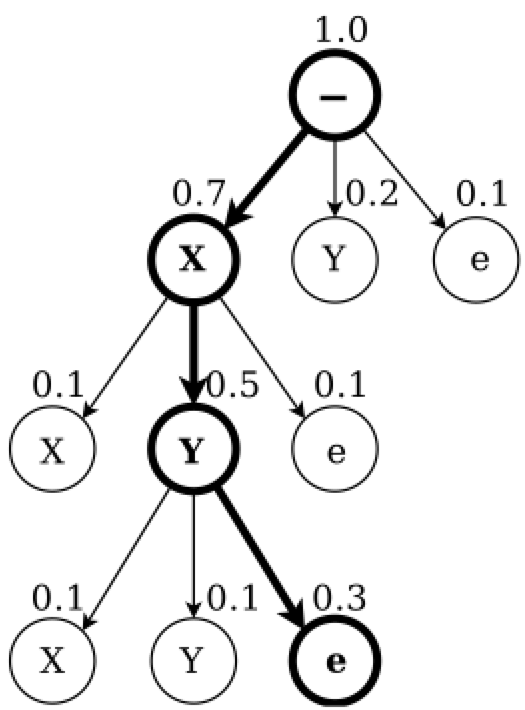
Best path:
наиболее вероятный путь даст
наиболее вероятную разметку

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

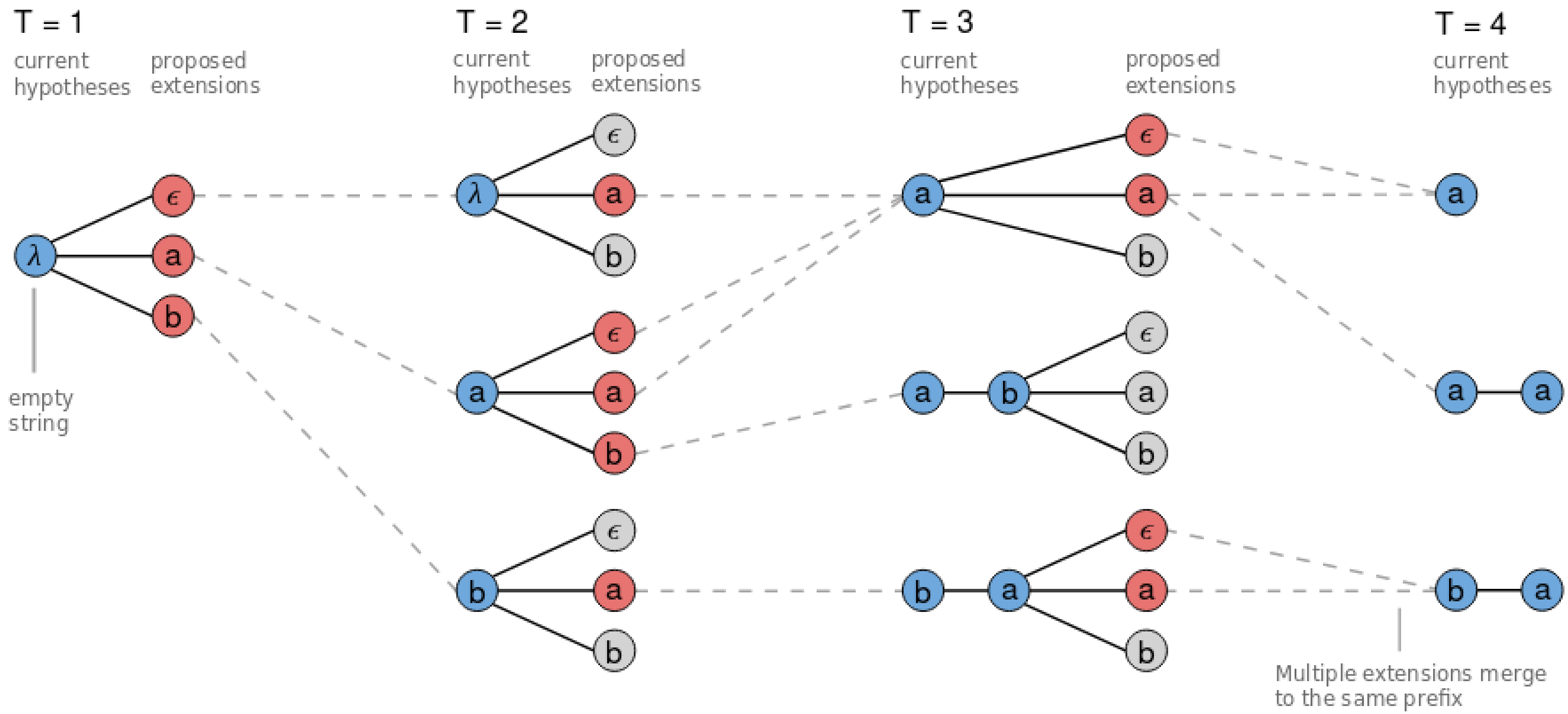
быстр, но нет гарантии
нахождения лучшего решения

Prefix search (beam search)
используя backward-forward algorithm,
считает вероятности расширений уже
посчитанных префиксов

всегда находит лучшее решение,
но растёт экспоненциально – поэтому
используем beam search



Beam search



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

Forward-backward algorithm

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

Forward variable

Инициализация

$$\alpha_1(1) = y^1_b$$

$$\alpha_1(2) = y^1_{\mathbf{l}_1}$$

$$\alpha_1(s) = 0, \ \forall s > 2$$

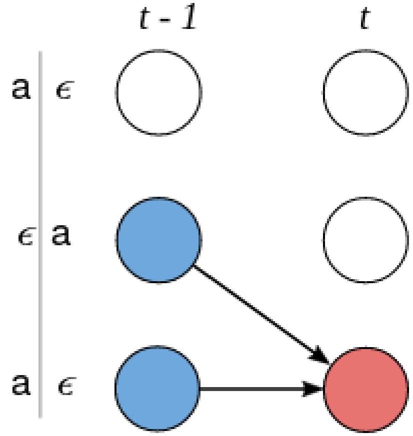
При этом преобразовали l в l', добавив пустой лейбл b в начале, в конце и между соседними лейблами

Рекуррентное вычисление

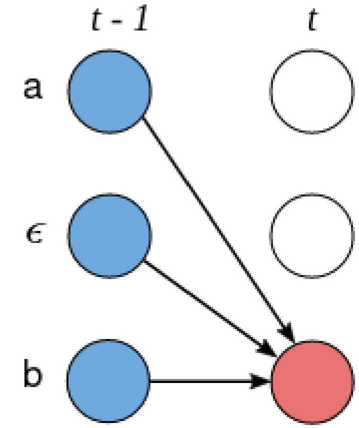
$$\alpha_t(s) = \begin{cases} \bar{\alpha}_t(s) y^t_{\mathbf{l}'_s} & \text{if } \mathbf{l}'_s = b \text{ or } \mathbf{l}'_{s-2} = \mathbf{l}'_s \\ \big(\bar{\alpha}_t(s) + \alpha_{t-1}(s-2)\big) y^t_{\mathbf{l}'_s} & \text{otherwise} \end{cases}$$

where

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



Если пустой или такой же,
как 2 шага назад



Otherwise

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

where

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

Тогда вероятность разметки выражается через forward variable так:

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

Аналогично с backward variable

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

<u>Инициализация</u>

$$\beta_T(|\mathbf{l}'|) = y^T_b$$

$$\beta_T(|\mathbf{l}'| - 1) = y^T_{\mathbf{l}_{|\mathbf{l}|}}$$

$$\beta_T(s) = 0, \ \forall s < |\mathbf{l}'| - 1$$

<u>Рекуррентное вычисление</u>

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

where

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

Эвристика

При такой рекурсии скоро возникнет underflow,
поэтому нормализуем переменные

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

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

и подставим их в предыдущие формулы.

Кстати, в таком случае log-likelihood выглядит очень просто:

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

Обучение

Минимизируем дифференцируемую целевую функцию

$$O^{ML}(S,\mathcal{N}_w) = -\sum_{(\mathbf{x},\mathbf{z})\in S} ln\big(p(\mathbf{z}|\mathbf{x})\big)$$

Используя вышеприведённые формулы для переменных, получаем

$$p(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|\mathbf{l}'|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{\mathbf{l}'_s}} \qquad \frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y^t_k} = \frac{1}{{y^t_k}^2} \sum_{s\in lab(\mathbf{l},k)} \alpha_t(s)\beta_t(s)$$

Когда мы идём через softmax layer, вспоминаем про нашу нормализацию и считаем вот так, где u – unnormalised output

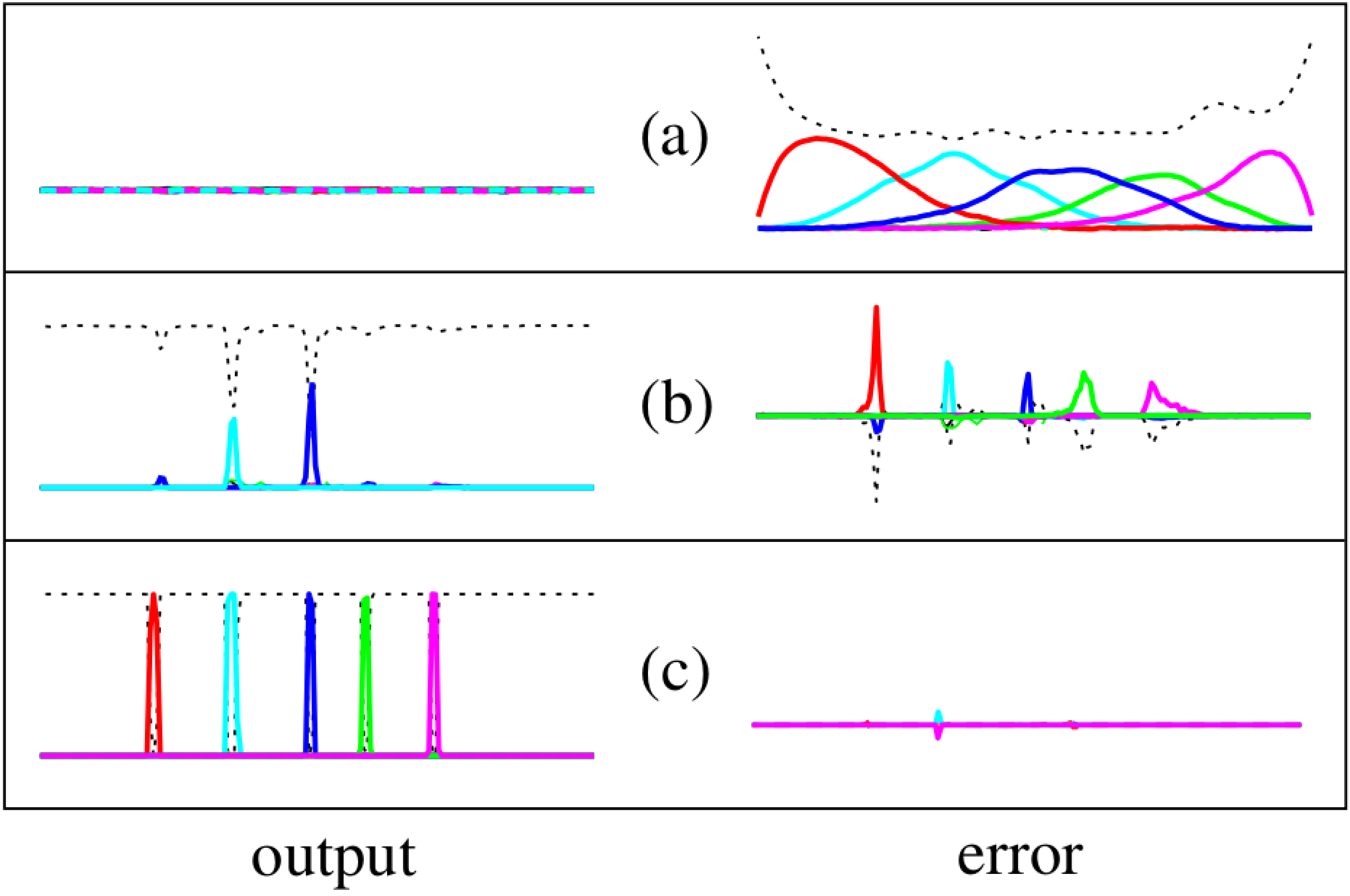
$$\frac{\partial O^{ML}(\{(\mathbf{x},\mathbf{z})\},\mathcal{N}_w)}{\partial u^t_k} = y^t_k - \frac{1}{y^t_k Z_t} \sum_{s\in lab(\mathbf{z},k)} \hat{\alpha}_t(s)\hat{\beta}_t(s)$$

Это наш error signal

where

$$Z_t \stackrel{\text{def}}{=} \sum_{s=1}^{|\mathbf{l}'|} \frac{\hat{\alpha}_t(s)\hat{\beta}_t(s)}{y^t_{\mathbf{l}'_s}}$$

И обучаем с помощью BPTT



Три среза обучающейся сети

Подробный вывод (на всякий случай)

$$\alpha_t(s)\beta_t(s) = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = \mathbf{l}'_s}} y^t_{\mathbf{l}'_s} \prod_{t=1}^{T} y^t_{\pi_t}$$

используя

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

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

$$\frac{\alpha_t(s)\beta_t(s)}{y^t_{\mathbf{l}'_s}} = \sum_{\substack{\pi \in \mathcal{B}^{-1}(\mathbf{l}): \\ \pi_t = \mathbf{l}'_s}} p(\pi|\mathbf{x})$$

используя

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

Используем, что вероятность разметки выражается через сумму вероятностей l’ с последним пустым лейблом и без него

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

$$p(\mathbf{l}|\mathbf{x}) = \sum_{s=1}^{|\mathbf{l}'|} \frac{\alpha_t(s)\beta_t(s)}{y^t_{\mathbf{l}'_s}}$$

$$\frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y^t_k} = \frac{1}{{y^t_k}^2} \sum_{s \in lab(\mathbf{l},k)} \alpha_t(s)\beta_t(s)$$

Используем общий факт

$$\frac{\partial ln(p(\mathbf{l}|\mathbf{x}))}{\partial y^t_k} = \frac{1}{p(\mathbf{l}|\mathbf{x})} \frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y^t_k}$$

При BPTT через softmax layer помним про rescaling; u – unnormalised output

$$\frac{\partial O^{ML}(\{(\mathbf{x},\mathbf{z})\}, \mathcal{N}_w)}{\partial u^t_k} = y^t_k - \frac{1}{y^t_k Z_t} \sum_{s \in lab(\mathbf{z},k)} \hat{\alpha}_t(s)\hat{\beta}_t(s)$$

Эксперимент в статье

Data: Корпус звучащей английской речи TIMIT

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 %

LER (Label error rate): normalised edit distance между правильным и предсказанным лейблом

Итоги

- В сфере Speech Recognition CTC (наряду с Attention) сменила HMMs и позволила использовать нейронные сети
- В отличие от HMMs, не требует пресегментирования и постобработки
- Является alignment-free, в связи с чем отлично подходит для задач, где нужно несегментированному сигналу сопоставить только последовательность лейблов
- Тем не менее, в таких задачах как keyword spotting, где сегментация нужна только приблизительная, CTC тоже работает
- Один из недостатков CTC – предположение о независимости лейблов в последовательности, т.е. языковые зависимости она не учит
- Есть имплементации для Tensorflow, Keras, Theano

Ресурсы

http://lig-membres.imag.fr/blanchon/SitesEns/NLSP/resources/ASR2018.pdf

https://www.cs.toronto.edu/~graves/icml_2006.pdf

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