Deep Learning Speech Recognition

Enes Witwit
University of Heidelberg

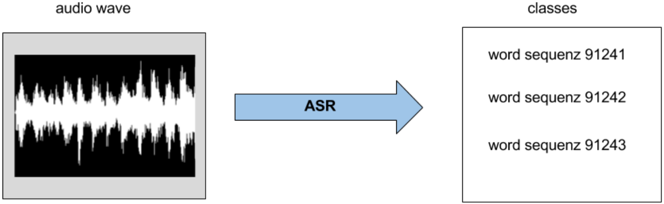
May 10, 2017

Contents

Automatic Speech Recognition (ASR)

Definition Automatic transformation of spoken language by humans into the corresponding word sequence.

Speech recognition as classification problem



ASR Applications

What are the applications for ASR and what do they imply?

- Dictation (Lawyer, Doctor, ...)
- Control devices/systems (Mobile, car, ...)
- Language translation
- Education (Teach reading)

ASR Applications

What are the applications for ASR and what do they imply?

- Dictation (Lawyer, Doctor, ...)
- Control devices/systems (Mobile, car, ...)
- Language translation
- Education (Teach reading)

Depending on the application we face different problems and challenges

1. Does training data fit our purpose?
2. What are the enviromental acoustical settings for our application?
3. . . .

ASR Advantages

- Speed
  - Keyboard 200-1000 characters per minute
  - Speech 1000-4000 characters per minute
- No need of using hands or eyes
- Communication with systems/devices naturally
- Portable

ASR Disadvantages

- Locational requirements
  - Not usable in locations where silence is required
  - Not usable in loudy enviroments
- Error rate still to high

Difficulties

Difficulties

Variability

Difficulties

Size Number of word types in vocabulary

Speaker speaker-independency, adaptation to speaker characteristics and accent

Acoustic environment Noise, competing speakers, channel conditions (microphone, phone line, room acoustics)

Style Planned monologue or spontaneous conversation.Continuous or isolated speech.

Difficulties

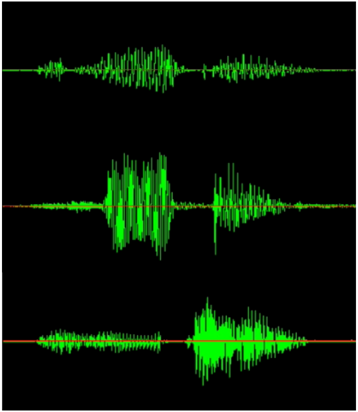
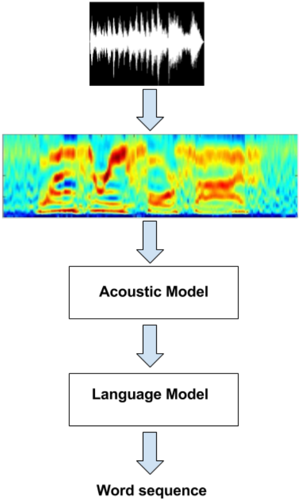


Figure: The word "Sieben" recorded three times

History

- 1952 Bell Labs single speaker digit recognition
- 1968 Dynamic Time Warping (DTW) for Speech Recognition by Vintsyuk
- 1969 Hidden Markov Models (HMM) by Leonard Baum
- 1997 Long short-term memory (LSTM) by Hochreiter and Schmidhuber
- 2006 Connectionist Temporal Classification (CTC) by Graves et al.
- 2007 LSTM Models trained by Connectionist Temporal Classification (CTC) outperforms traditional systems in certain applications

Standard ASR Pipeline



Why do we need signal processing?

- Need a form of signal we can work with easily
- Extract relevant information
- Filter unnecessary information
  - Speaker-dependent information
  - Acoustical enviroment
  - Microfon
- Reduction of data size

Spectogram

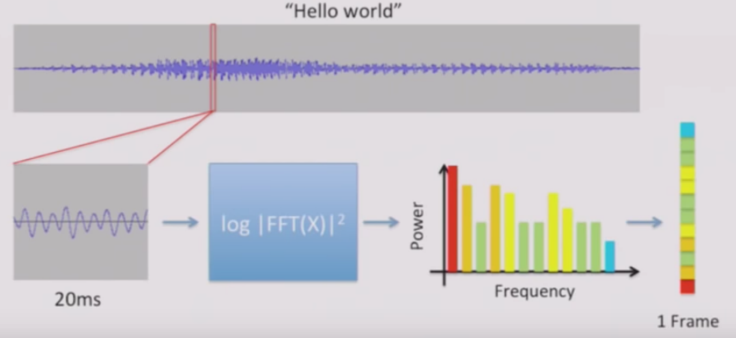


Figure: Deep Learning School 2016 (Talk: Adam Cotes, Baidu)

Spectogram

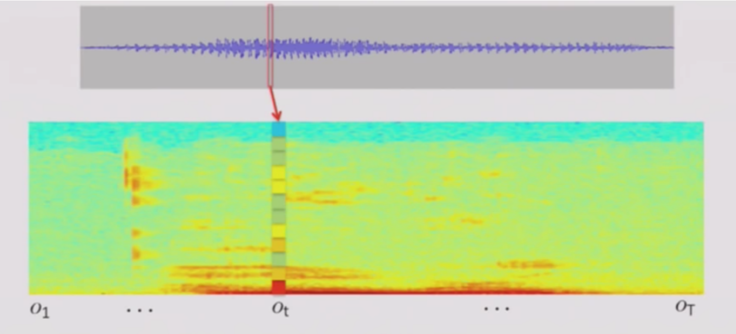
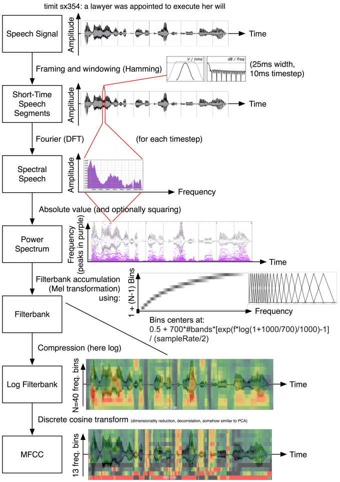


Figure: Deep Learning School 2016 (Talk: Adam Cotes, Baidu)

Mel Frequency Cepstral Coefficients(MFCC)



Make signal processing intelligent again

Using audio wave as raw input for model training

- Sainath et al., Interspeech 2015

Fundamental Equation of Statistical Speech Recognition

- Let X be a sequence of acoustic feature vectors
- Let W denote a word sequence
- Let W^* denotes the most likely word sequence

$$W^* = argmax_W P(W|X)$$

Fundamental Equation of Statistical Speech Recognition

- Let X be a sequence of acoustic feature vectors
- Let W denote a word sequence
- Let W^* denotes the most likely word sequence

$$\begin{aligned} W^* &= argmax_W P(W|X) \\ &= arg \max_W \frac{P(X|W)P(W)}{P(X)} \text{ (Bayes Theorem)} \end{aligned}$$

Fundamental Equation of Statistical Speech Recognition

- Let X be a sequence of acoustic feature vectors
- Let W denote a word sequence
- Let W^* denotes the most likely word sequence

$$
\begin{aligned}
W^* &= argmax_W P(W|X) \\
&= arg\max_W \frac{P(X|W)P(W)}{P(X)} \text{ (Bayes Theorem)} \\
&= arg\max_W \underbrace{P(X|W)}_{\text{Acoustic model}} \underbrace{P(W)}_{\text{Language Model}}
\end{aligned}
$$

Approach

There are two approaches for developing an acoustic model

1. Hidden Markov Model
2. Neural Networks

Stochastic process

Definition (Markov chain of order n)

$$P(X_{t+1} = s_{t+1} | X_t = s_t, \dots, X_0 = s_0)$$
$$= P(X_{t+1} = s_{t+1} | X_t = s_t, \dots, X_{t-n+1} = s_{t-n+1})$$

Hidden Markov Model (HMM)

Suppose you cannot observe the states .

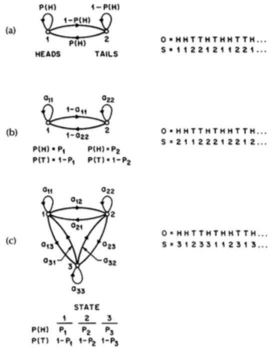


Fig. 2. Three possible Markov models which can account for the results of hidden coin tossing experiments. (a) 1-coin model. (b) 2-coins model. (c) 3-coins model.

Figure: A Tutorial on Hidden Markov Models by Rabiner

HMM Definition $\lambda = (A; B; \pi)$

- N is number of states in the model. S is the set of states $S = (S_1, \ldots, S_N)$ and the state at time t as q_t
- M is number of disctinct observations per state. Observations are denoted by $V = v_1, \ldots, v_M$
- State transition probability distribution $A_{ij} = \{a_{ij}\}$ where

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], 1 \leq i, j \leq N$$

- Observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where

$$b_j(k) = P[v_k \text{ at } t | q_t = S_j], 1 \leq j \leq N, 1 \leq k \leq M$$

- Initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P[q_1 = S_i]$ for $1 \leq i \leq N$

HMM Assumptions

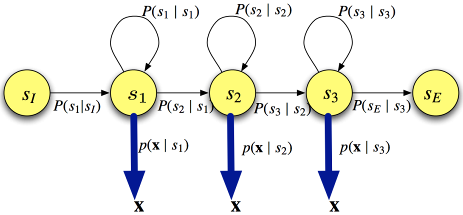


Figure: Probabilistic finite state automaton (Renals and Bell, ASR Lecture, Edinburgh)

1. **Observation independence** An acoustic observation x is conditionally independent of all other observations given the state that generated it
2. **Markov process** A state is conditionally independent of all other states given the previous state

Output Distribution

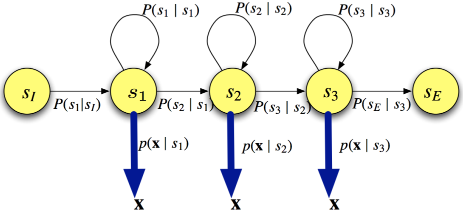


Figure: Probabilistic finite state automaton (Renals and Bell, ASR Lecture, Edinburgh)

- $b_j(x) = p(x|s_j) = \mathcal{N}(x; \mu^j, \Sigma^j)$ (Single multivariate Gaussian)
- $b_j(x) = p(x|s_j) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(x; \mu^{jm}, \Sigma^{jm})$ (M-component Gaussian Mixture Model)

The three HMM Challenges

The three HMM Challenges

1. **Evaluation** Given a HMM λ, an Output O $\rightarrow$ What is the probabilty that O is an Output of the HMM λ: $P(O|\lambda)$?
   Forward or Backward Algorithm

The three HMM Challenges

1. **Evaluation** Given a HMM λ, an Output O $\rightarrow$ What is the probabilty that O is an Output of the HMM λ: $P(O|\lambda)$?
   Forward or Backward Algorithm
2. **Decoding** Given a HMM λ, an Output O. Find a sequence of States $\hat{S} = s_{j1}, \ldots, s_{jT}$ for which holds $\hat{S} = argmax_S P(S, O|\lambda)$
   Viterbi Algorithm

The three HMM Challenges

1. **Evaluation** Given a HMM λ, an Output O $\rightarrow$ What is the probabilty that O is an Output of the HMM λ: $P(O|\lambda)$?
   Forward or Backward Algorithm
2. **Decoding** Given a HMM λ, an Output O. Find a sequence of States $\hat{S} = s_{j1}, \ldots, s_{jT}$ for which holds $\hat{S} = argmax_S P(S, O|\lambda)$
   Viterbi Algorithm
3. **Training** Given a HMM λ and a set of Training Data O. Find better Parameters λ' such that $P(O|\lambda) < P(O|\lambda')$
   Baum Welch Algorithm

1. The Forward Algorithm

Goal Estimate $P(O|\lambda)$

- We need to sum over all possible state sequences $s_1, s_2, \ldots, s_T$ that could result in the observation sequence O
- Rather than enumerating each sequence, compute the probabilities recursively (exploit the Markov Assumption)
- Forward Probability $\alpha_t(s_j)$:the probability of observing the observation sequence $o_1, \ldots, o_t$ and being in state s_j at time t:

$$a_t(s_j) = p(x_1, \ldots, x_t, S(t) = s_j|\lambda)$$

1. The Forward Algorithm

1. Initialization

$$\alpha_0(s_I) = 1$$
$$\alpha_0(s_j) = 0 \text{ if } s_j \neq s_I$$

2. Recursion

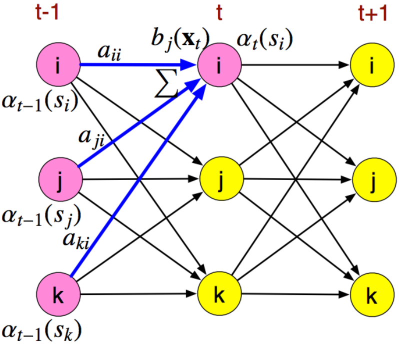
$$\alpha_t(s_j) = \sum_{i=1}^{N} \alpha_{t-1}(s_i) a_{ij} b_j(o_t)$$

3. Termination

$$p(O|\lambda) = \alpha_T(s_E) = \sum_{i=1}^{N} \alpha_T(s_i) a_{iE}$$

Forward Recursion

$$\alpha_t(s_j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda})$$



1. The Backward Algorithm

1. Initialization

$$\beta_T(i) = 1 \ , \ 1 \leq i \leq |S|$$

2. Recursion

$$\beta_t(i) = \sum_{j=1}^{|S|} \beta_j(o_{t+1}) a_{ij} \beta_{t+1}(j) \ , \ 1 \leq i \leq |S| \ , \ 1 \leq t < T$$

3. Termination

$$p(O|\lambda) = \sum_{j=1}^{|S|} \pi_j b_j(o_1) \beta_1(j)$$

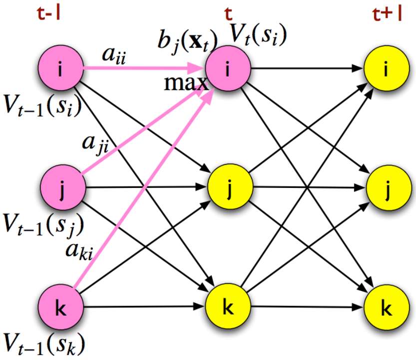
Viterbi approximation

- Instead of summing over all possible state sequences we change the summation to a maximation in the recursion

$$V_t(s_j) = max_i V_{t-1}(s_i) a_{ij} b_j(x_t)$$

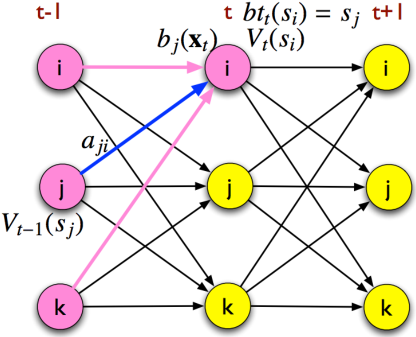
- This change in the recursion gives us now the most probable path
- We need to keep track of the states that make up this path by keeping a sequence of backpointers to enable a Viterbi backtrace: the backpointer for each state at each time indicates the previous state on the most probable path

Viterbi approximation



Viterbi approximation

Backpointers to the previous state on the most probable path



2.Decoding: The Viterbi Algorithm

1. Initialization

$$V_0(s_I) = 1$$
$$V_0(s_j) = 0 \text{ if } s_j \neq s_I$$
$$bt_0(s_j) = 0$$

2. Recursion

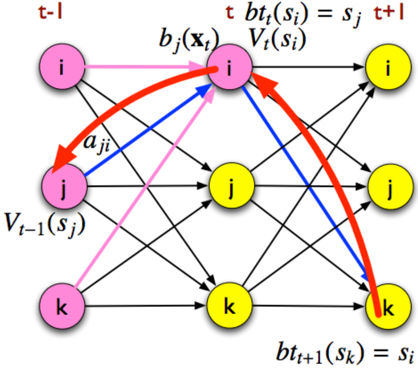
$$V_t(s_j) = \max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(o_t)$$
$$bt_t(s_j) = arg \max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(o_t)$$

3. Termination

$$P^* = V_T(s_E) = \max_{i=1}^{N} V_T(s_i) a_{iE}$$
$$s_T^* = bt_T(q_E) = arg \max_{i=1}^{N} V_T(s_i) a_{iE}$$

Viterbi Backtrace

Backtrace to find the state sequence of the most probable path



3.Training:Baum-Welch Algorithm

1. **Forwad procedure**
   Let $\alpha_i(t) = P(Y_1 = y_1, ..., Y_t = y_t, X_t = i|\theta)$, the probability of seeing the $y_1, y_2, ..., y_t$ and being in state i at time t.
2. **Backward procedure**
   Let $\beta_i(t) = P(Y_{t+1} = y_{t+1}, ..., Y_T = y_T|X_t = i, \theta)$ that is the probability of the ending partial sequence $y_{t+1}, ..., y_T$ given starting state i at time t.
3. **Update**

$$\gamma_i(t) = P(X_t = i|Y, \theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j|Y, \theta) = \frac{\alpha_i(t)a_{ij}\beta_j(t+1)b_j(y_{t+1})}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i(t)a_{ij}\beta_j(t+1)b_j(y}$$

3.Training:Baum-Welch Algorithm

Update parameters

$$\pi_i^* = \gamma_i(1)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t=v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

Neural networks for acoustic models

Goal create a neural network (DNN/RNN) from which we can extract transcription y. Train with labeled pairs (x, y^*).

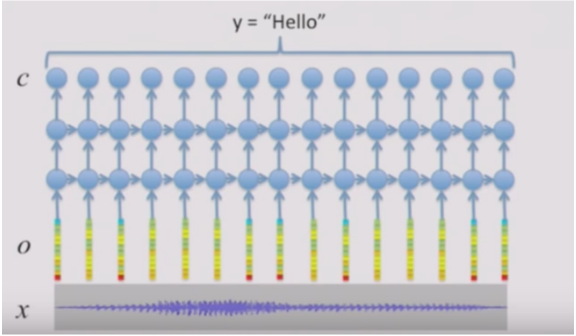


Figure: Deep Learning School 2016 (Adam Cotes, Baidu)

Recurrent Neural Network (RNN)

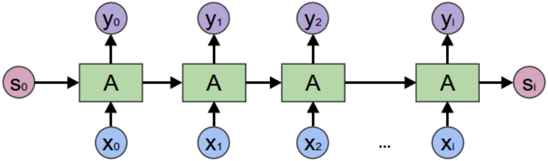


Figure: http://colah.github.io/posts/2015-09-NN-Types-FP/

Recurrent Neural Network (RNN)

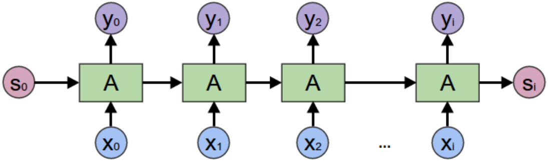


Figure: http://colah.github.io/posts/2015-09-NN-Types-FP/

Forward propagation

$$h_i = \sigma(W_{hh}h_{i-1} + W_{hx}x_i + b_h)$$
$$\hat{y}_i = W_{yh}h_i$$

Long Short-Term Memory (LSTM)

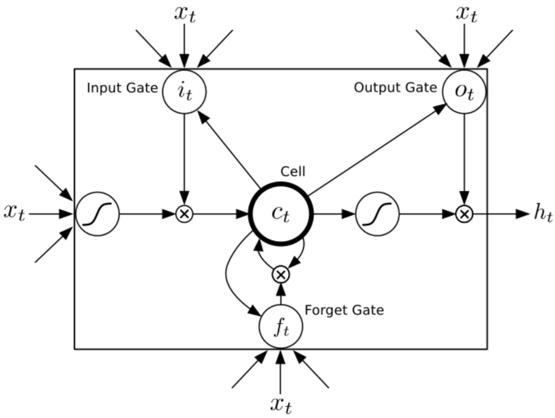
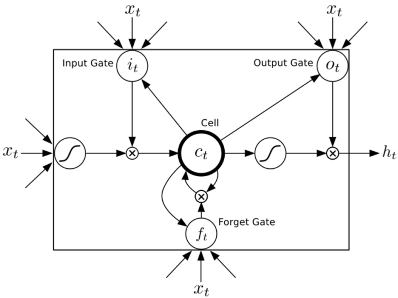


Figure: Long Short-term Memory Cell from Graves et al. 2013

Long Short-Term Memory (LSTM)



$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
h_t &= o_t tanh(c_t)
\end{aligned}
$$

Bidirectional RNN (BRNN)

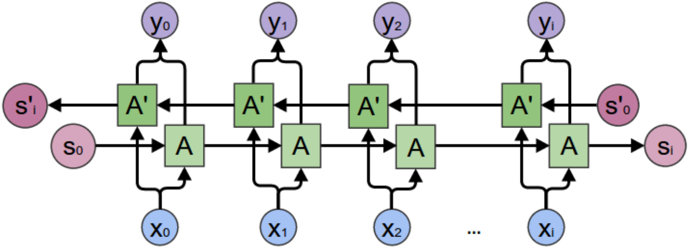


Figure: http://colah.github.io/posts/2015-09-NN-Types-FP/

Train acoustic model

Main issue $length(x) \neq length(y)$

- Solution
  - Connectionist Temporal Classification [Graves et al., 2006]
  - Attention, Sequence to Sequence

Connectionist Temporal Classification (CTC)

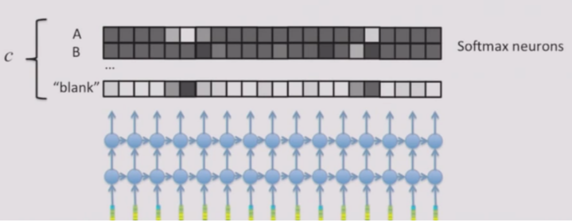
Basic idea

1. RNN output neurons c encode distributions over symbols. (length(c)=length(x))
   For phoneme based models
   $c \in \{AA, AE, AX, \ldots, ER1, blank\}$
   For grapheme based models $c \in \{A, B, C, \ldots, blank\}$
2. Define mapping $\beta(c) \to y$
3. Maximize likelihood of y^* under this model

Connectionist Temporal Classification (CTC)

Basic idea

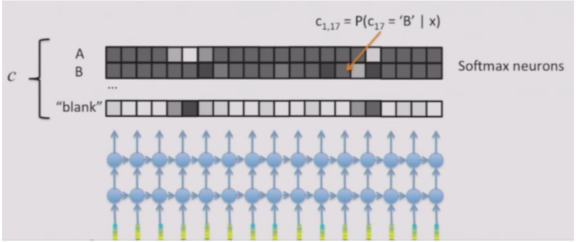
1. RNN output neurons c encode distributions over symbols.
(length(c)=length(x))
For grapheme based models $c \in \{A, B, C, \ldots, blank\}$



Connectionist Temporal Classification (CTC)

Basic idea

1. RNN output neurons c encode distributions over symbols. (length(c)=length(x))
   For grapheme based models $c \in \{A, B, C, \ldots, blank\}$



Connectionist Temporal Classification (CTC)

Basic idea

1. RNN output neurons c encode distributions over symbols. (length(c)=length(x))
   For grapheme based models $c \in \{A, B, C, \ldots, blank\}$
2. Output softmax neurons defines distribution over whole character sequences c assuming independency:

$$P(c|x) = \prod_{i=1}^{N} P(c_i|x)$$

$$P(c = HH\_E\_\_L\_LO\_|x) = P(c_1 = H|x)P(c_2 = H|x) \ldots P(c_{15} = blank$$

How do we get our independency?

→ Forbid connections from the output layer to other output layers or to other hidden layers

Connectionist Temporal Classification (CTC)

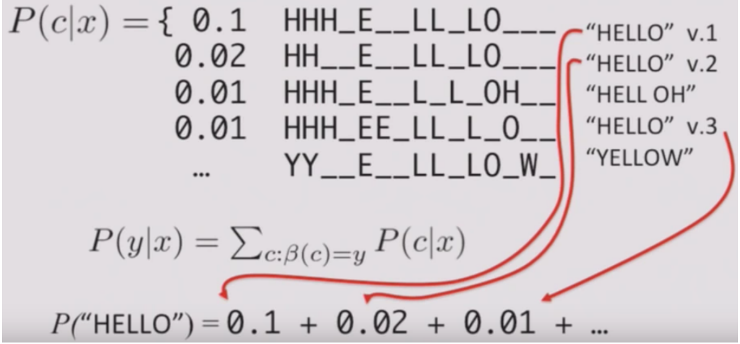
Basic idea

2. Define function $\beta(c) = y$
   What it does:
   - squeeze out duplicates
   - removes blanks

$$y = \beta(c) = \beta(HH\_E\_\_L\_LO\_) = "HELLO"$$

Connectionist Temporal Classification (CTC)

- Our function gives us a distribution for all possible transcriptions y



Connectionist Temporal Classification (CTC)

3. Update network parameters θ to maximize likelihood of correct label y^*:

$$\theta^* = arg \max_{\theta} \sum_i logP(y^{*(i)} | x^{(i)})$$

Connectionist Temporal Classification (CTC)

3. Update network parameters θ to maximize likelihood of correct label y^*:

$$\begin{aligned}\theta^* &= arg\max_{\theta}\sum_i logP(y^{*(i)}|x^{(i)}) \\ &= arg\max_{\theta}\sum_i log\sum_{c:\beta(c)=y^{*(i)}} P(c|x^{(i)}) \text{ (Thanks CTC)}\end{aligned}$$

Decoding

- How do we find most likely transcription

$$y_{max} = \max_{y} P(y|x)$$

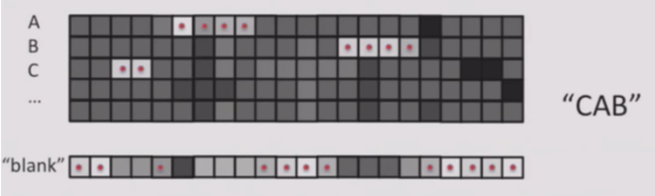
Decoding

- How do we find most likely transcription

$$y_{max} = \max_{y} P(y|x)$$

- Best Path Decoding (not the most likely)

$$\beta(arg\max_{c} P(c|x))$$



Language Model

RNN output	Decoded Transcription
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game

Figure: Examples of transcriptions directly from the RNN (left) with errors that are fixed by addition of a language model (right). (Hannun et al. 2014)

Standard approach: N-gram Model

Goal Apply grammar and spelling rules

- Word sequence $w_1^n = w_1 \dots w_n$
- N-gram approximation

$$P(w_1^n) = \prod_{k-1}^{n} P(w_k | w_{k-N-1}^{k-1})$$

Decoding with LM

- Given a LM Hannun et. al optimizes:

$$arg\max_{w} P(w|x)P(w)^{\alpha}[length(w)]^{\beta}$$

Decoding with LM

- Given a LM Hannun et. al optimizes:

$$\arg\max_{w} P(w|x)P(w)^{\alpha}[length(w)]^{\beta}$$

- α is tunable parameter to govern weight of LM

Decoding with LM

- Given a LM Hannun et. al optimizes:

$$arg \max_w P(w|x)P(w)^\alpha[length(w)]^\beta$$

- α is tunable parameter to govern weight of LM
- β penalty term for long words

Decoding with LM's

Basic strategy Beam search to maximize

$$arg \max_{w} P(w|x)P(w)^{\alpha}[length(w)]^{\beta}$$

Start with set of candidate transcript prefixes $A = \{\}$.
For $t = 1, \ldots, T$
For each candidate in A consider

1. Add blank; dont change prefix; update probability using AM;
2. Add space to prefix; update probability using LM
3. Add a character to prefix; update probability using AM; Add new candidates with updated probabilities A_{new}

A:=K most probable prefixes in A_{new}

Neural Network Language Model

Idea: Rescore list of candidate transcriptions on basis of neural network

N-gram model just gave us grammar and spelling rules but sometimes we need also "semantic understanding"

neural network models to simulate the semantic correctness of candidate transcriptions

- RNN
- LSTM
- ...

End to end Speech Recognition with neon

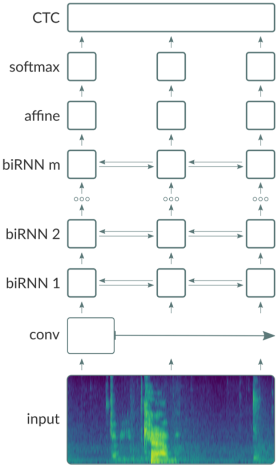


Figure: https://www.nervanasys.com/end-end-speech-recognition-neon/

State of the art (IBM, March 2017)

- **Acoustic model** score fusion of three models: one LSTM with multiple feature inputs, a second LSTM trained with speaker-adversarial multi-task learning and a third residual net (ResNet) with 25 convolutional layers and time-dilated convolutions
- **Language model** word and character LSTMs and convolutional WaveNet-style language models.

Summary

- Historically used approach for ASR: Dynamic Time Warping later statistical models
- Standard ASR Pipeline: 1.Signal Processing 2. Acoustic Model 3.Language Model
- Signal processing: MFCC
- Acoustic model two approaches: HMM and Neural Networks
  - GMM for HMM Distribution
  - Three problems of HMM: Evaluation(Forward/Backward Algorithm), Decoding(Viterbi), Training (Baum-Welch Algorithm)
  - Neural networks approach: RNN, LSTM, BRNN
  - Neural networks training: CTC
- Language Model: N-gram model

Future

- End-to-end systems: Go deeper in the whole pipeline
- Image Processing: Lip reading?
- Train better: Batch normalization (Ioffe and Szegedy, 2015) and more
- Scale: More data, better data, more computational power, ...

Bibliography

- Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks (Graves et al,. 2006)
- Deep Speech: Scaling up end-to-end speech recognition (Hannun et al., 2014)
- Towards End-to-End Speech Recognition with Recurrent Neural Networks (Graves and Jaitly, 2014)
- Generating Sequences With Recurrent Neural Networks (Graves, 2014)
- Speech Recognition with Deep Recurrent Neural Networks (Graves, Mohamed and Hinton, 2013)
- Bidirectional Recurrent Neural Networks (Schuster and Paliwal, 1997)
- A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition (Rabiner et al., 1989)

Bibliography

- English Conversational Telephone Speech Recognition by Humans and Machines (Saon et al.,2017)
- Deep Neural Networks for Acoustic Modeling in Speech Recognition (Hinton et al.,2012)