Speech Recognition and Processing

Lecture 7

Yossi Keshet
Agenda
Agenda

- DNN-HMM Summary
Agenda

• DNN-HMM Summary

• End-to-End Acoustic Models
Agenda

- DNN-HMM Summary
- End-to-End Acoustic Models
- Connectionist Temporal Classification
Agenda

- DNN-HMM Summary
- End-to-End Acoustic Models
  - Connectionist Temporal Classification
- Letter Based Acoustic Models
Recall

Decoder

Most probable word sequence

$P(\bar{w})$  $\cdot$  $P(\bar{p}|\bar{w})$  $\cdot$  $P(\bar{o}|\bar{p})$

Language model

Pronunciation model

Acoustic model

Hypothesized word sequence $\bar{w}$

Words $\bar{p}$

Phones $\bar{p}$

Acoustics $\bar{o}$

Observations $\bar{o}$

Acoustic features
Recall

N-Gram/RNN/CNN/etc.

Decoder

$P(\tilde{w})$ • $P(\tilde{p}|\tilde{w})$ • $P(\tilde{o}|\tilde{p})$

Language model

Pronunciation model

Acoustic model

Most probable word sequence

Hypothesized word sequence

N-Gram/RNN/CNN/etc.

Acoustic features

Observations $\tilde{o}$

Acoustics
Recall

N-Gram/RNN/CNN/etc.

Decoder

$P(\overrightarrow{w})$  
$P(\overrightarrow{p}|\overrightarrow{w})$  
$P(\overrightarrow{\hat{o}}|\overrightarrow{p})$

Language model

Pronunciation model

Acoustic model

Most probable word sequence

hypothesized word sequence

N-Gram/RNN/CNN/etc.

Acoustic features

observations $\overrightarrow{\hat{o}}$

Lexicon / Probabilistic models
Recall

N-Gram/RNN/CNN/etc.

Lexicon / Probabilistic models

DNN + HMM

Most probable word sequence

Language model

P(\tilde{w})

P(\tilde{p}|\tilde{w})

P(\tilde{\alpha}|\tilde{p})

Decoder

hypothesized word sequence

words

phones

Acoustic model

Acoustic features

observations \tilde{\alpha}

N-Gram/RNN/CNN/etc.

Lexicon / Probabilistic models

3
Recall
Recall

• We would like to compute the probability: $P(\bar{o} | \bar{p})$
Recall

• We would like to compute the probability: $P(\bar{o} | \bar{p})$

• Consider the phone sequence [p r aa b l iy]
• We would like to compute the probability: $P(\overline{o} \mid \overline{p})$

• Consider the phone sequence [p r a a b l i y]

• The phone sequence can be written as an expended sequence of phone symbols $\overline{q} = [p, p, r, r, \ldots, r, aa, aa, \ldots, b, b, b, l, \ldots, iy, iy, iy]$
Recall

- We would like to compute the probability: $P(\bar{o} | \bar{p})$

- Consider the phone sequence [p r aa b l iy]

- The phone sequence can be written as an expended sequence of phone symbols $\bar{q} = [p, p, r, r, ..., r, aa, aa, ..., b, b, b, l, ..., iy, iy, iy]$

- The neural network model needs an alignment!
Recall
Recall

- We would like to compute the probability:

- Consider the phone sequence [p r aa b l iy]

- The phone sequence can be written as an expended sequence of phone symbols $\tilde{q} = [p, p, r, r, \ldots, r, aa, aa, \ldots, b, b, b, l, \ldots, iy, iy, iy]$

- The neural network model needs an alignment!

How can we get these alignments?
Recall

• We would like to compute the probability: $P(\tilde{o} | \tilde{p})$

• Consider the phone sequence [p r aa b l iy]

• The phone sequence can be written as an expended sequence of phone symbols $\tilde{q} = [p, p, r, r, ..., r, aa, aa, ..., b, b, b, l, ..., iy, iy, iy]$

• The neural network model needs an alignment!

How can we get these alignments?
DNN-HMM
DNN-HMM

- One way to get these alignments is from manually labeled datasets
DNN-HMM

• One way to get these alignments is from manually labeled datasets

• Example
DNN-HMM

• One way to get these alignments is from manually labeled datasets

  • Example

• Automatic alignments (last week):
DNN-HMM

• One way to get these alignments is from manually labeled datasets

• Example

• Automatic alignments (last week):

  • Forward-Backward:
  \[
  \sum_{a \in A} \prod_{t=1}^{T} P(\bar{\omega}_t | \bar{q}_t^a) \cdot P(\bar{q}_t^a | \bar{q}_{t-1}^a)
  \]
DNN-HMM

- One way to get these alignments is from manually labeled datasets
  - Example

- Automatic alignments (last week):
  - Forward-Backward: \[
    \sum_{a \in A} \prod_{t=1}^{T} P(\tilde{o}_t | \tilde{q}_t^a) \cdot P(\tilde{q}_t^a | \tilde{q}_{t-1}^a)
  \]
  - Viterbi: \[
    \max_{a \in A} \prod_{t=1}^{T} P(\tilde{o}_t | \tilde{q}_t^a) \cdot P(\tilde{q}_t^a | \tilde{q}_{t-1}^a)
  \]
DNN-HMM
This approach has two main problems:
DNN-HMM

- This approach has two main problems:

- Often, it doesn’t make sense to force every input step to align to some output. In speech recognition, for example, the input can have stretches of silence with no corresponding output.
This approach has two main problems:

- Often, it doesn’t make sense to force every input step to align to some output. In speech recognition, for example, the input can have stretches of silence with no corresponding output.

- We have no way to produce outputs with multiple characters in a row. Consider the alignment [h, h, e, l, l, l, o]. Collapsing repeats will produce “helo” instead of “hello”.
Can we train an acoustic model without explicit alignments?
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

- The CTC algorithm is *alignment-free* — it doesn’t require an alignment between the input and the output
Connectionist Temporal Classification (CTC)

- The CTC algorithm is *alignment-free* — it doesn’t require an alignment between the input and the output.

- To get the probability of an output given an input, CTC works by summing over the probability of all possible alignments between the two.
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs

- This new token is sometimes called the \textit{blank} token
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs.
- This new token is sometimes called the `blank` token.
- We’ll refer to it here as $\epsilon'$. 
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs
- This new token is sometimes called the *blank* token
- We’ll refer to it here as $\epsilon$
- It doesn’t correspond to anything and is simply removed from the output
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs.

- This new token is sometimes called the *blank* token.

- We’ll refer to it here as $\epsilon$.

- It doesn’t correspond to anything and is simply removed from the output.

- If the transcription has two of the same character in a row, then a valid alignment must have an $\epsilon$ between them.
Connectionist Temporal Classification (CTC)

- To overcome the conceptual alignment problems the CTC algorithm introduces another new token to the allowed outputs

- This new token is sometimes called the *blank* token

- We’ll refer to it here as $\epsilon$

- It doesn’t correspond to anything and is simply removed from the output

- If the transcription has two of the same character in a row, then a valid alignment must have an $\epsilon$ between them
Connectionist Temporal Classification (CTC)

First, merge repeat characters.

Then, remove any $\epsilon$ tokens.

The remaining characters are the output.
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

A few properties to notice:
Connectionist Temporal Classification (CTC)

A few properties to notice:

1. **The allowed alignments between are monotonic**

   i.e. if we advance to the next input, we can keep the corresponding output the same or advance to the next one
Connectionist Temporal Classification (CTC)

A few properties to notice:

1. **The allowed alignments between are monotonic**
   
i.e. if we advance to the next input, we can keep the corresponding output the same or advance to the next one

2. **The alignment is of many-to-one**
   
i.e. one or more input elements can align to a single output element but not vice-versa
Connectionist Temporal Classification (CTC)

A few properties to notice:

1. **The allowed alignments between are monotonic**
   
   i.e. if we advance to the next input, we can keep the corresponding output the same or advance to the next one

2. **The alignment is of many-to-one**
   
   i.e. one or more input elements can align to a single output element but not vice-versa

3. **The length of the transcription cannot be greater than the length of the audio file**
A few properties to notice:

1. The allowed alignments between are monotonic

   i.e. if we advance to the next input, we can keep the corresponding output the same or advance to the next one

2. The alignment is of many-to-one

   i.e. one or more input elements can align to a single output element but not vice-versa

3. The length of the transcription cannot be greater than the length of the audio file
Connectionist Temporal Classification (CTC)

Valid Alignments

$c c c c a t$
$c c a a t t t$
$c a c c c t$

Invalid Alignments

$c c c c a t$
$c c a a t t$
$c a c c c t$

Corresponds to $Y = [c, c, a, t]$

Has length 5

Missing the 'a'
CTC as a loss function

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

The input is fed into an RNN, for example.

The network gives $p_t(a | X_t)$, a distribution over the outputs \{h, e, l, o, ε\} for each input step.
CTC as a loss function

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

The input is fed into an RNN, for example.

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

With the per time-step output distribution, we compute the probability of different sequences.
CTC as a loss function

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

The input is fed into an RNN, for example.

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

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

By marginalizing over alignments, we get a distribution over outputs.
Connectionist Temporal Classification (CTC)

Formally,

\[
p(Y \mid X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)
\]

The CTC conditional probability marginalizes over the set of valid alignments, computing the probability for a single alignment step-by-step.

The problem is there can be a massive number of alignments. Thankfully, we can compute this probability much faster with a dynamic programming algorithm.
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

- The key idea is that the sum over paths corresponding to a labelling can be broken down into an iterative sum over paths corresponding to prefixes of that labelling
Connectionist Temporal Classification (CTC)

• The key idea is that the sum over paths corresponding to a labelling can be broken down into an iterative sum over paths corresponding to prefixes of that labelling

• First we place $\epsilon$ before or after any token in $Y$
Connectionist Temporal Classification (CTC)

• The key idea is that the sum over paths corresponding to a labelling can be broken down into an iterative sum over paths corresponding to prefixes of that labelling.

• First we place $\epsilon$ before or after any token in $Y$.

• Resulting: $Z = [\epsilon, y_1, \epsilon, y_2, \ldots, y_L, \epsilon]$.
Connectionist Temporal Classification (CTC)

• The key idea is that the sum over paths corresponding to a labelling can be broken down into an iterative sum over paths corresponding to prefixes of that labelling.

• First we place $\epsilon$ before or after any token in $Y$.

• Resulting: $Z = [\epsilon, y_1, \epsilon, y_2, \ldots, y_L, \epsilon]$

• It’s easier to describe the algorithm using a sequence which includes them.
Connectionist Temporal Classification (CTC)

• The key idea is that the sum over paths corresponding to a labelling can be broken down into an iterative sum over paths corresponding to prefixes of that labelling

• First we place $\epsilon$ before or after any token in $Y$

• Resulting: $Z = [\epsilon, y_1, \epsilon, y_2, \ldots, y_L, \epsilon]$

• It’s easier to describe the algorithm using a sequence which includes them
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

- Let’s $\alpha$, be the score of the all alignments ending at a given node.
Connectionist Temporal Classification (CTC)

• Let’s $\alpha$, be the score of the all alignments ending at a given node

• More precisely, $\alpha_{s,t}$ is the CTC score of the subsequence $Z_{1:s}$ after $t$ time steps
Connectionist Temporal Classification (CTC)

• Let’s $\alpha$, be the score of the all alignments ending at a given node

• More precisely, $\alpha_{s,t}$ is the CTC score of the subsequence $Z_{1:s}$ after $t$ time steps

• Then, we can compute the final CTC score $P(Y \mid X)$ from the $\alpha$ ’s at the last time steps
Connectionist Temporal Classification (CTC)

• Let’s $\alpha$, be the score of the all alignments ending at a given node

• More precisely, $\alpha_{s,t}$ is the CTC score of the subsequence $Z_{1:s}$ after $t$ time steps

• Then, we can compute the final CTC score $P(Y \mid X)$ from the $\alpha$’s at the last time steps
Example

Node \((s, t)\) in the diagram represents \(\alpha_{s,t}\) – the CTC score of the subsequence \(Z_{1:s}\) after \(t\) input steps.
Connectionist Temporal Classification (CTC)

• As long as we know the values of $\alpha$ at the previous time-step, we can compute $\alpha_{s,t}$

• There are two cases, let us describe them
Connectionist Temporal Classification (CTC)

Case 1:

In this case, we can’t jump over $z_{s-1}$, the previous token in $Z$.

The first reason is that the previous token can be an element of $Y$, and we can’t skip elements of $Y$. Since every element of $Y$ in $Z$ is followed by an $\epsilon$, we can identify this when $z_s = \epsilon$.

The second reason is that we must have an $\epsilon$ between repeat characters in $Y$. We can identify this when $z_s = z_{s-2}$.

$$\alpha_{s,t} = \left( \alpha_{s-1,t-1} + \alpha_{s,t-1} \right) \cdot \text{CTC probability of the two valid subsequences after } t - 1 \text{ input steps.}$$

$$p_t(z_s \mid X) \quad \text{The probability of the current character at input step } t.$$
Connectionist Temporal Classification (CTC)

Case 2:

In the second case, we’re allowed to skip the previous token in Z. We have this case whenever \( z_{s-1} = \epsilon \) between two unique characters. As a result there are three positions we could have come from at the previous step.

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

The CTC probability of the three valid subsequences after \( t - 1 \) input steps. The probability of the current character at input step \( t \).
Connectionist Temporal Classification (CTC)

What about the start conditions?

We have two optional starting nodes. why?

\[ \alpha_{1,1} = p_1(e \mid X) \]
\[ \alpha_{2,1} = p_1(z_1 \mid X) \]
Connectionist Temporal Classification (CTC)

What about the start conditions?

We have two optional starting nodes. Why?

\[ \alpha_{1,1} = p_1(\epsilon | X) \]
\[ \alpha_{2,1} = p_1(z_1 | X) \]

There are also two valid final nodes since there is an \( \epsilon \) at the end of the sequence. The complete probability is the sum of the two final nodes.
Connectionist Temporal Classification (CTC)
Connectionist Temporal Classification (CTC)

The CTC loss function is differentiable with respect to the per time-step output probabilities since it’s just sums and products of them.
The CTC loss function is differentiable with respect to the per time-step output probabilities since it’s just sums and products of them.

Given that, we can analytically compute the gradient of the loss function with respect to the output probabilities and run back-propagation as usual.
Connectionist Temporal Classification (CTC)

The CTC loss function is differentiable with respect to the per time-step output probabilities since it’s just sums and products of them.

Given that, we can analytically compute the gradient of the loss function with respect to the output probabilities and run back-propagation as usual.

Then, we can optimize the following loss function: 

$$\sum_{(\bar{x}, \bar{p}) \in \mathcal{D}} - \log P(\bar{p} | \bar{x})$$
Inference
Inference

After we’ve trained the model, we’d like to use it to find a likely output for a given input. More precisely, we need to solve:

\[ Y^* = \arg \max_Y p(Y|X) \]

One heuristic is to take the most likely output at each time-step. This gives us the alignment with the highest probability:
Inference

After we’ve trained the model, we’d like to use it to find a likely output for a given input. More precisely, we need to solve:

\[ Y^* = \arg \max_Y p(Y|X) \]

One heuristic is to take the most likely output at each time-step. This gives us the alignment with the highest probability:

\[ A^* = \arg \max_A \prod_{t=1}^{T} p_t(a_t|X) \]
Inference

After we’ve trained the model, we’d like to use it to find a likely output for a given input. More precisely, we need to solve:

$$Y^* = \arg \max_Y p(Y|X)$$

One heuristic is to take the most likely output at each time-step. This gives us the alignment with the highest probability:

$$A^* = \arg \max_A \prod_{t=1}^{T} p_t(a_t|X)$$

We can then collapse repeats and remove $\epsilon$ tokens to get $Y$. 
Inference

After we’ve trained the model, we’d like to use it to find a likely output for a given input. More precisely, we need to solve:

$$Y^* = \arg \max_Y p(Y | X)$$

One heuristic is to take the most likely output at each time-step. This gives us the alignment with the highest probability:

$$A^* = \arg \max_A \prod_{t=1}^{T} p_t(a_t | X)$$

We can then collapse repeats and remove $\epsilon$ tokens to get $Y$. 
The CTC algorithm is \textit{alignment-free}. The objective function marginalizes over all alignments. In practice we see that the CTC ends up allocating most of the probability to a single alignment. However, this isn’t guaranteed.

Can be used to as unsupervised aligner.

Recall, last lesson we saw that we can obtain:

\[ P(\bar{o} | \bar{q})P(\bar{q}) = \prod_{t=1}^{T} P(o_t | q_t)P(q_t | q_{t-1}) \]
Recall, last lesson we saw that we can obtain:

\[ P(\tilde{o} | \tilde{q}) P(\tilde{q}) = \prod_{i=1}^{T} P(o_t | q_t) P(q_t | q_{t-1}) \]

Where the same over all the alignments can be obtained using the forward-backward algorithm
Recall, last lesson we saw that we can obtain:

\[
P(\bar{o} | \bar{q})P(\bar{q}) = \prod_{i=1}^{T} P(o_t | q_t)P(q_t | q_{t-1})
\]

where the same over all the alignments can be obtained using the forward-backward algorithm.
Recall, last lesson we saw that we can obtain:

\[ P(\bar{o} \mid \bar{q})P(\bar{q}) = \Pi_{i=1}^{T} P(o_t \mid q_t)P(q_t \mid q_{t-1}) \]

Probability of observing an acoustic features given a phoneme

Transition probability

Where the same over all the alignments can be obtained using the forward-backward algorithm.
In the CTC model we assume the transition probabilities are uniform

\[ P(\bar{o} | \bar{q})P(\bar{q}) = \prod_{i=1}^{T} P(o_t | q_t)P(q_t | q_{t-1}) \]
In the CTC model we assume the transition probabilities are uniform

\[ P(\tilde{o} | \tilde{q})P(\tilde{q}) = \prod_{i=1}^{T} P(o_t | q_t)P(q_t | q_{t-1}) \]

Then with a few Bayes tricks we get to the same CTC objective
In the CTC model we assume the transition probabilities are uniform.

\[ P(\bar{o} | \bar{q})P(\bar{q}) = \prod_{i=1}^{T} P(o_t | q_t)P(q_t | q_{t-1}) \]

**Probability of observing an acoustic features given a phoneme**

Then with a few Bayes tricks we get to the same CTC objective.
In the CTC model we assume the transition probabilities are uniform.

\[
P(\tilde{o} | \tilde{q})P(\tilde{q}) = \prod_{i=1}^{T} P(o_t | q_t)P(q_t | q_{t-1})
\]

Probability of observing an acoustic features given a phoneme

Transition probability

Then with a few Bayes tricks we get to the same CTC objective.
The CTC algorithm is usually works at the **Character Level**. Hence, no need to a pronunciation model.
The CTC algorithm is usually works at the **Character Level**. Hence, no need to a pronunciation model.
Questions?