Logistics

• Early submission is due **Nov 3rd, 11:59PM ET**
  • Kaggle submission a with Lev. Dist <= 30
  • Canvas MCQ

• On time submission deadline: **Nov 17th, 11:59PM ET**

• This part may not take time as much as HW2P2 for training but the high cut-off will be significantly harder

• Constrains:
  • No attention
Problem at hand

Input Utterance MFCC

15

MODEL

Sequence of Phonemes

[ 'B', 'IH', 'K', 'SH', 'A' ]
Data and Task

- Features: Same as HW1P2 (15D)
- Labels: Order synchronous but not time synchronous
- Should output sequence of phonemes
  - ['B', 'IH', 'K', 'SH', 'A'] (precisely the indexes)
- Loss: CTCLoss
- Metric: mean Levenshtein distance
  - Can import (given in starter notebook)
  - Sequence of Phonemes -> String and then calculate distance (Use CMUdict and ARPABet)
Batch of Variable Length Inputs: Padding

• HW1, HW2: Equal length inputs
• HW3: Variable Length sequences
• Steps:
  • Padding
  • Packing
Batch of Variable Length Inputs: Padding

- Padding

Need to store unpadded lengths as well. Have the variables `lengths_x, lengths_y` in the starter notebook.

Ref: 11785 Fall 21 Bootcamp
Batch of Variable Length Inputs: Padding

- Padding

Need to store unpadded lengths as well. Have the variables $lengths_x$, $lengths_y$ in the starter notebook.

Padded to equal lengths

$$(B, *, 15) \rightarrow (B, T, 15)$$

Ref: 11785 Fall 21 Bootcamp
Batch of Variable Length Inputs: Padding

- Padding

Need to store unpadded lengths as well. Have the variables `lengths_x, lengths_y` in the starter notebook.

Not for the whole dataset (instead we pack after padding)

Ref: 11785 Fall 21 Bootcamp
Batch of Variable Length Inputs: Packing

List of Tensors to be packed. Each has same number of features but different time steps.

Figure 2: List of tensors we want to pack

Tensors sorted in descending order based on the number of time steps in each sample.

Figure 3: First we sort the list in a descending order based on number of timesteps in each

Ref: 11785 Fall 21 Bootcamp
Batch of Variable Length Inputs: Packing

List of Tensors to be packed. Each has same number of features but different time steps.

Figure 2: List of tensors we want to pack

Tensors sorted in descending order based on the number of time steps in each sample.

Figure 3: First we sort the list in a descending order based on number of timesteps in each

Final 2d Packed Tensor

Figure 4: Final Packed 2d Tensor

Ref: 11785 Fall 21 Bootcamp
Parts of a Sequence Model

Embedding Layer → Sequence Model → Classification Layer
Embedding Layer

- Optional but recommended
- Used to increase/decrease the dimensionality of the input
Embedding Layer

- Optional but recommended
- Used to increase/decrease the dimensionality of the input
- Eg. In NLP, 10k vocabulary represented as 1 hot vectors with 10k dim

<table>
<thead>
<tr>
<th>'deep'</th>
<th>'neural'</th>
<th>'net'</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Shape 10,000 x 1

Real valued vectors

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td>0.4</td>
<td></td>
</tr>
</tbody>
</table>

Shape emb_dim x 1
Embedding Layer

- Optional but recommended
- Used to increase/decrease the dimensionality of the input
- Our task:
  - Input dim = 15
  - Expand to emb_dim > 15 for feature extraction

Ref: HW1P2 Write-up
Embedding Layer: Conv1d Layers

• Consider the below as an input having 3 features at each time instant.
Embedding Layer: Conv1d Layers

- We can use Convolution which increases the channels of the input as we go deeper.
Embedding Layer: Conv1d Layers

- We can use Convolution to which increases the channels of the input as we go deeper.

- No. Filters = 5
- Kernel= 3; Padding= 1; Stride= 1
- Kernel= 5; Padding= 2; Stride= 1
  (Or anything similar)
Embedding Layer: Conv1d Layers

• We can use Convolution to which increases the channels of the input as we go deeper.

• No. Filters = 5
• Kernel= 3; Padding= 1; Stride= 1
• Kernel= 5; Padding= 2; Stride= 1
(Or anything similar)
Embedding Layer: Conv1d Layers

- Our input is of shape (B, T, 15) (after padding). How can we change it to (B, T, 64)?

Assuming `batch_first = True` (You may also have it as (T, B, 13))
Embedding Layer: Conv1d Layers

- Our input is of shape \((B, T, 15)\) (after padding). How can we change it to \((B, T, 64)\) ?

- Transpose/Permute: \((B, T, 15) \rightarrow (B, 15, T)\) which makes \#channels = 15 (Conv1d)

- Apply convolution \((B, 15, T) \rightarrow (B, 64, T)\)

- Transpose/Permute: \((B, 64, T) \rightarrow (B, T, 64)\) (pack and pass to LSTM/GRU)

- Note: This is done in the forward function

Assuming \(\text{batch\_first} = \text{True}\) (You may also have it as \((T, B, 13)\)
Embedding Layer: Conv1d Layers

If stride > 1, we effectively reduce the time steps

- stride = 1
- stride = 2
Embedding Layer: Conv1d Layers

- Stride > 1 reduces computation for LSTM and training is faster.
- However, too much reduction in time steps will lead to loss of information (we don’t recommend downsampling more than 4x)
Embedding Layer: Conv1d Layers

- Stride > 1 reduces computation for LSTM and training is faster.
- However, too much reduction in time steps will lead to loss of information (we don’t recommend downsampling more than 4x)

- **Note:** Stride > 1 alters number of time steps. You need to change lengths_x accordingly
  - Use convolution formula \((X - K + 2*P)/S\) (or)
  - Clamp lengths to length of embedding (torch function)
Embedding Layer: Conv1d Layers

- You can try convolution layers based on residual blocks
- Hint: Remember HW2P2!

Sequence Model

- Can use RNN, GRU, LSTM (recommended) from \textit{torch.nn}

\texttt{http://colah.github.io/posts/2015-08-Understanding-LSTMs/}
Sequence Model

• Important parameters/hyper parameters in `nn.LSTM()`
  - `input_size` (15 or `emb_size`)
  - `hidden_dim`
  - `num_layers`
  - `dropout`
  - `bidirectional`
  - Note: when `bidirectional = True`, LSTM outputs a shape of `hidden_dim` in the forward direction and `hidden_dim` in the backward direction (in total, `2*hidden_dim`)
Classification Layer

- Same as HW1P2
- Output from the sequence model goes to the classification layer
- Variations
  - Deeper
  - Wider
  - Different activations
  - Dropout
Hyperparameters and Regularization

- Cepstral Normalization:
  \[ X \rightarrow \frac{(X - \text{mean})}{\text{std}} \]
- Different weight initialization (for Conv and Linear layers)
- Weight decay with optimizer
Hyperparameters and Regularization

• Scheduler is very important
  • ReduceLRonPlateau (Most of our ablation)
    • Lev distance might start to oscillate at lower values
  • Cosine Annealing
    • Try with higher number of epochs
Hyperparameters and Regularization

• Dropout is key
  • Can use dropout in all the 3 layers: Embedding, Sequence model and classification
  • You can also start with a small dropout rate and increase after the model gets trained
• Locked Dropout for LSTM layer
Hyperparameters and Regularization

• Addition of Noise *(only during training)*
  • Gaussian Noise
  • Gumbel Noise

• Need not add to all samples. Implement your module `AddNoise(nn.module)` in such a way that it adds noise to random inputs

https://en.wikipedia.org/wiki/Gumbel_distribution
https://en.wikipedia.org/wiki/Normal_distribution
Hyperparameters and Regularization

- Torch Audio Transforms [docs]
  - Time Masking
  - Frequency Masking
Hyperparameters and Regularization

• Beam width
  • Higher beam width may give better results (advisable to keep test beam width below 50 for computation purposes)
  • Sometimes bw = 1 (greedy search) also gives good results
  • Tip: Don’t use a high beam width while validating in each epoch (time per epoch will be higher)
Final Tips

• Make sure to split work within your study groups
All the best!