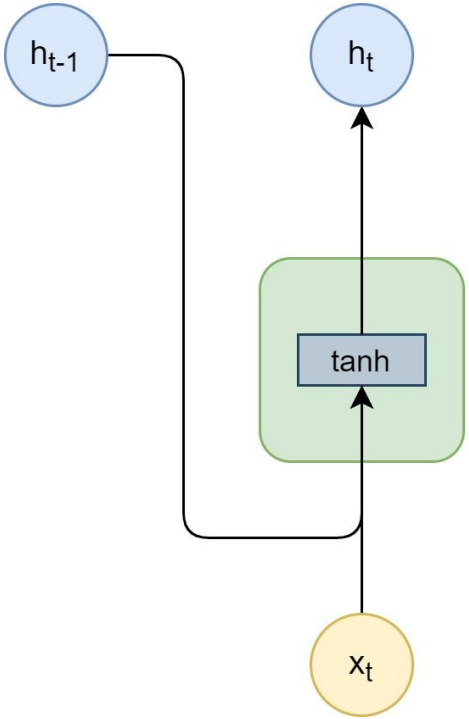
HW3P1 Bootcamp

RNN, GRU, CTC, and Greedy/Beam Search
(Fall 2022)

Aparajith Srinivasan

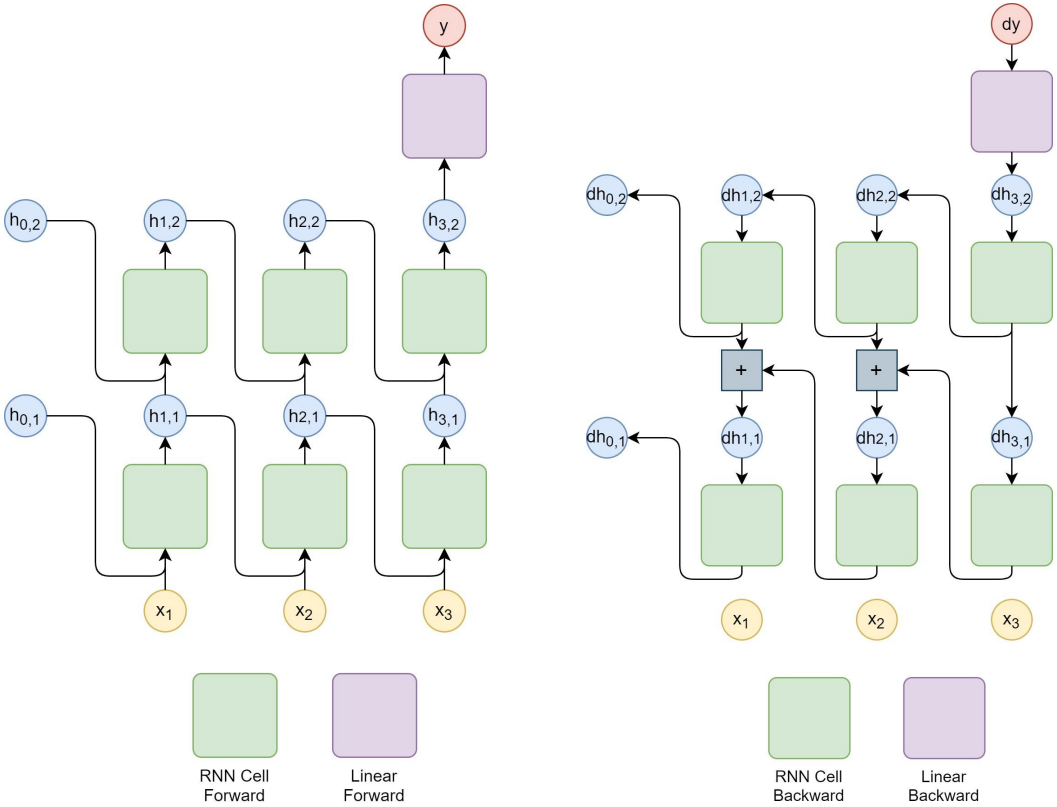
RNN Cell Forward / Backward



$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh})$$

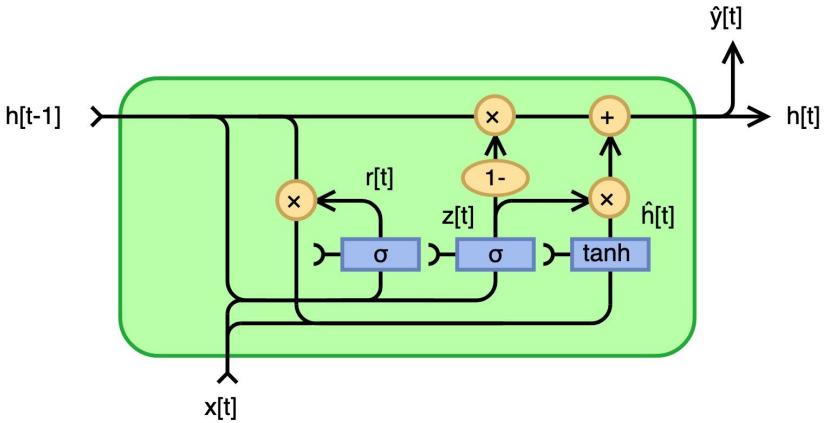
Tip: Very similar to how you did ***linear.py*** in hw1p1.

RNN Phoneme Classifier



- Forward is straight forward (XD)
- Backward is tricky
  - 2 diagrams in the write up for understanding the data flow
  - Then follow the pseudocode exactly

GRU Cell Forward/Backward



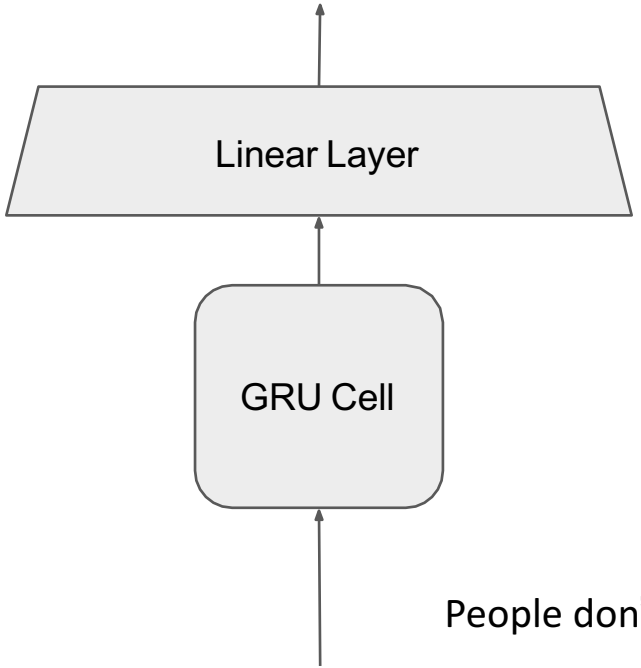
$$\mathbf{r}_t = \sigma(\mathbf{W}_{ir}\mathbf{x}_t + \mathbf{b}_{ir} + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{b}_{hr})$$
$$\mathbf{z}_t = \sigma(\mathbf{W}_{iz}\mathbf{x}_t + \mathbf{b}_{iz} + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{b}_{hz})$$
$$\mathbf{n}_t = \tanh(\mathbf{W}_{in}\mathbf{x}_t + \mathbf{b}_{in} + \mathbf{r}_t \otimes (\mathbf{W}_{hn}\mathbf{h}_{t-1} + \mathbf{b}_{hn}))$$
$$\mathbf{h}_t = (1 - \mathbf{z}_t) \otimes \mathbf{n}_t + \mathbf{z}_t \otimes \mathbf{h}_{t-1}$$

https://colah.github.io/posts/2015-08-Backprop

GRU Cell Forward / Backward

- GRU backward be the longest question in HW3P1
- Tips:
  - Modify the ***test_gru.py*** code accordingly – all **dWs** and **dbs** should correct to make sure that your **dx** and **dh** are correct
  - Can try to decompose eqns in forward (That's how I did :'))
    - A = Tanh(Wx*x + bx + Wh*h + bh)
      - Z1 = Wx*x + bx
      - Z2 = Wh*h + bh
      - Z = Z1 + Z2
      - A = Tanh(Z)
    - Backward is easy now. Need to compute the gradients in this order. Given dA (actually dLdA – ignoring for simplicity)
      - dZ -> dZ1, dZ2 -> dWh, dh, dbh -> …

GRU Inference



People don’t usually face issues in this ☺

CTC based questions

- Lecture slides have everything needed to complete all the CTC sections and also decoding

CTC based questions

- We have given example questions for you to understand the math behind it

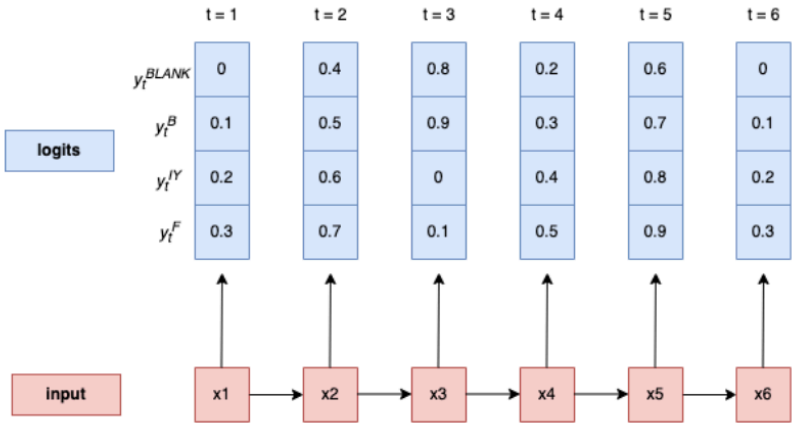


Figure 12: An overall CTC setup example

CTC based questions

- We have given example questions for you to understand the math behind it

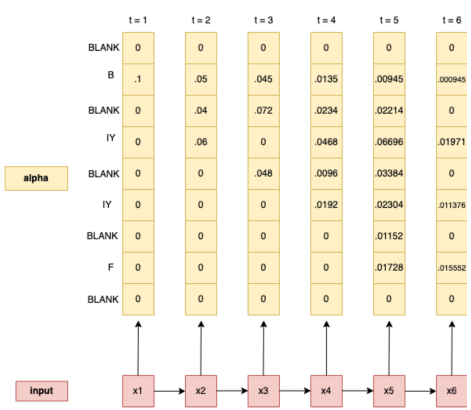


Figure 15: Forward Algorithm

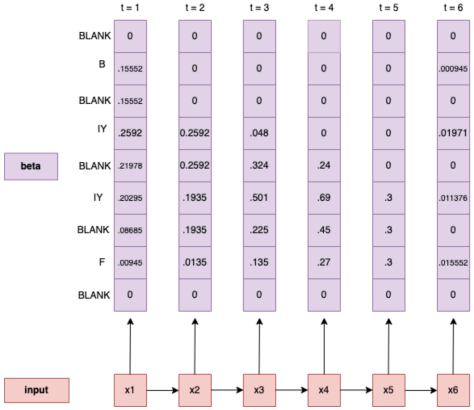


Figure 16: Backward Algorithm

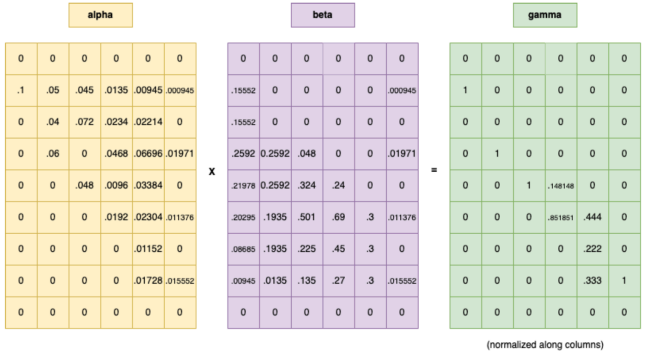


Figure 17: Posterior Probability

Greedy Search

- Taking the most probably output at each time step

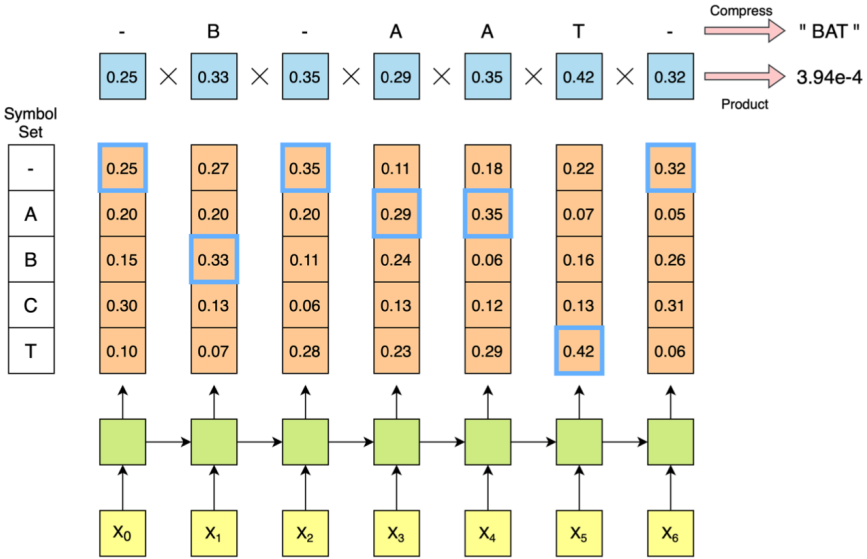


Figure 18: Greedy Search

Greedy Search

- Taking the most probably output at each time step

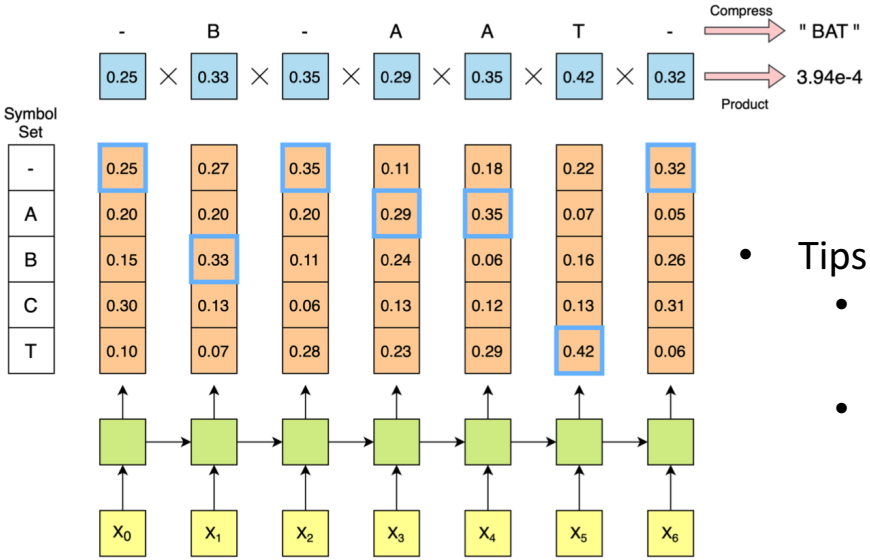


Figure 18: Greedy Search

- Tips
  - Write your compress function separately
  - Can complete without a for loop but a for loop wont cause autolab to time out ☺

Beam Search

- Another hard question in this part
- Tips to complete this question fast
  - Understand beam search from the lecture videos and slides
  - Beware of the definition of ***set()*** *(python { })* and ***list()*** *(python [ ])* from the code given in lecture slides. There is a difference in the python implementation
  - Complete each function ***InitializePaths, Prune, ExtendWithBlank, ExtendWithSymbol, MergeIdenticalPaths*** individually and then check your outputs with the flow chart given in the write up

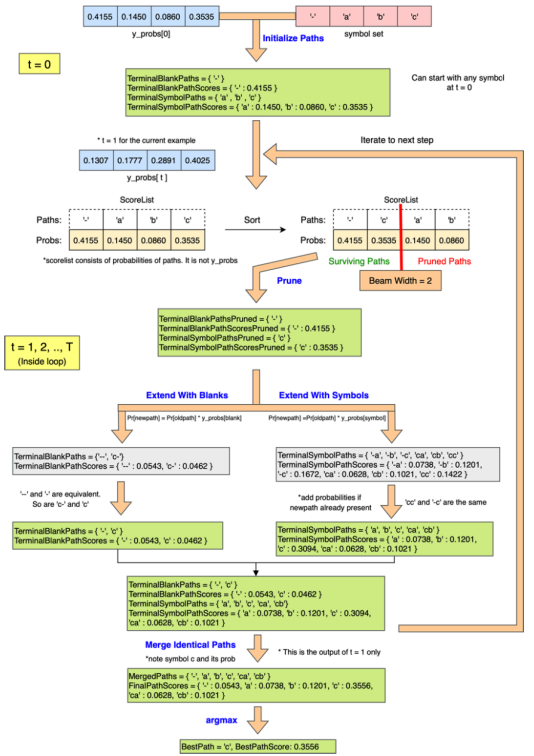


Figure 19: Beam Search

Beam Search

```
BEAM SEARCH

Global PathScore = [], BlankPathScore = []

# First time instant: Initialize paths with each of the symbols,
# including blank, using score at time t=1
NewPathsWithTerminalBlank, NewPathsWithTerminalSymbol, NewBlankPathScore, NewPathScore =
                        InitializePaths(SymbolSet, y[:,0])

# Subsequent time steps
for t = 1:T
    # Prune the collection down to the BeamWidth
    PathsWithTerminalBlank, PathsWithTerminalSymbol, BlankPathScore, PathScore =
                Prune(NewPathsWithTerminalBlank, NewPathsWithTerminalSymbol,
                                        NewBlankPathScore, NewPathScore, BeamWidth)
    # First extend paths by a blank
    NewPathsWithTerminalBlank, NewBlankPathScore = ExtendWithBlank(PathsWithTerminalBlank,
                                                    PathsWithTerminalSymbol, y[:,t])

    # Next extend paths by a symbol
    NewPathsWithTerminalSymbol, NewPathScore = ExtendWithSymbol(PathsWithTerminalBlank,
                                        PathsWithTerminalSymbol, SymbolSet, y[:,t])

end

# Merge identical paths differing only by the final blank
MergedPaths, FinalPathScore = MergeIdenticalPaths(NewPathsWithTerminalBlank, NewBlankPathScore
                                        NewPathsWithTerminalSymbol, NewPathScore)

# Pick best path
BestPath = argmax(FinalPathScore) # Find the path with the best score
```

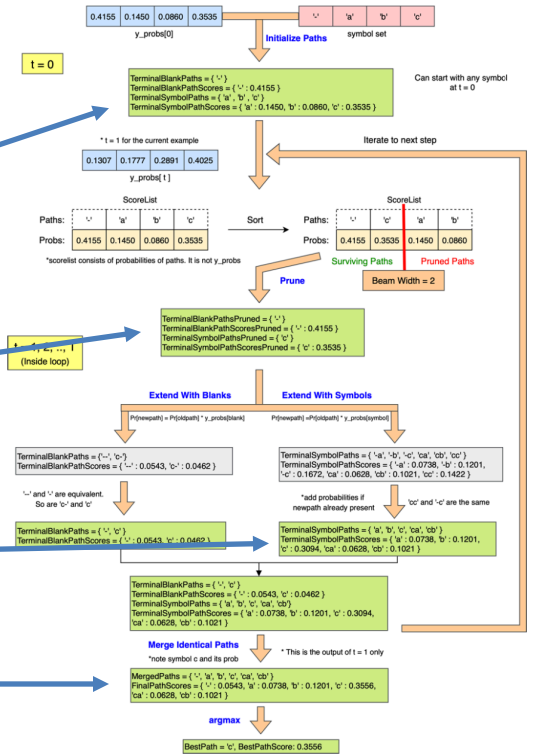


Figure 19: Beam Search

- Green boxes show the output for the 1st test case in the local autograder for just 1 time step
- You can break the flow in-between and check your answers

Thank you!
Q & A