HW3P1 Bootcamp

RNNs, GRUs, CTC, and Greedy/Beam Search

Fall 2024

Logistics

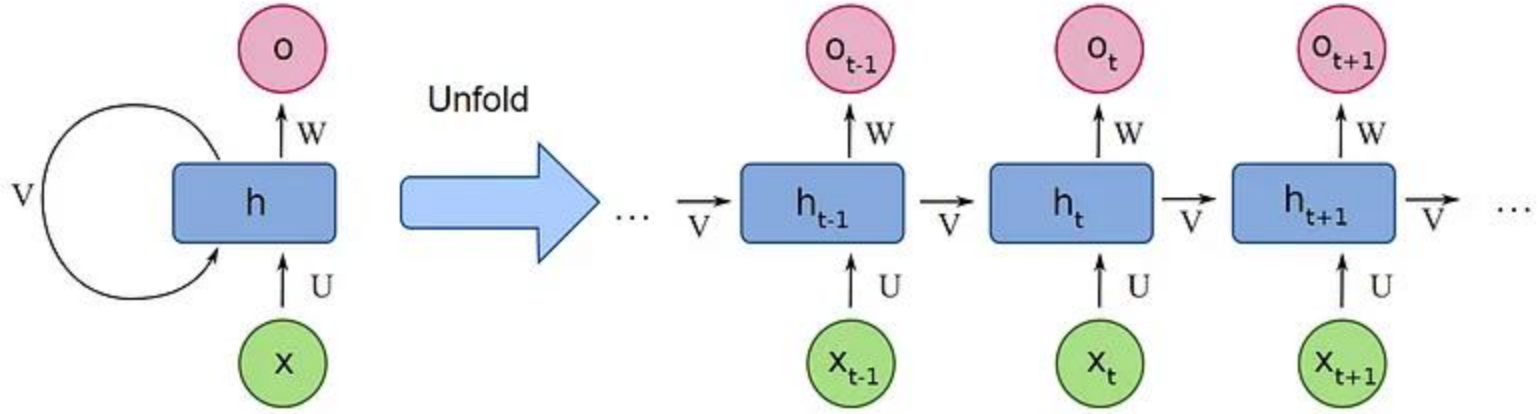
- Early Submission : **November 1st, 11:59pm**
- On-Time Submission: **November 8th, 11:59pm**

Two approaches: **Standard** and **Autograd**

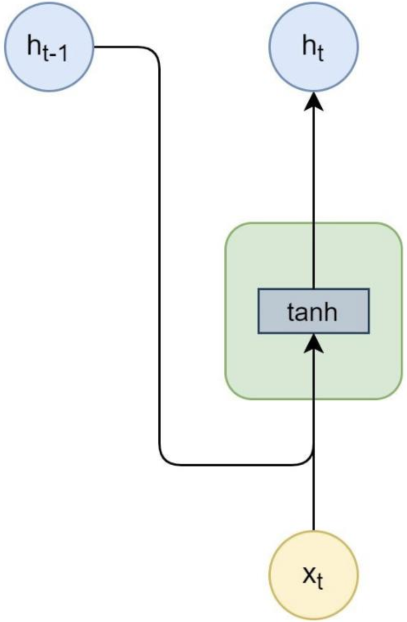
No late days can be used for Homework part 1s, please plan accordingly!

Structure of RNNs

A simple RNN that does seq-to-seq task with one RNN cell



RNN Cell Forward and Backward



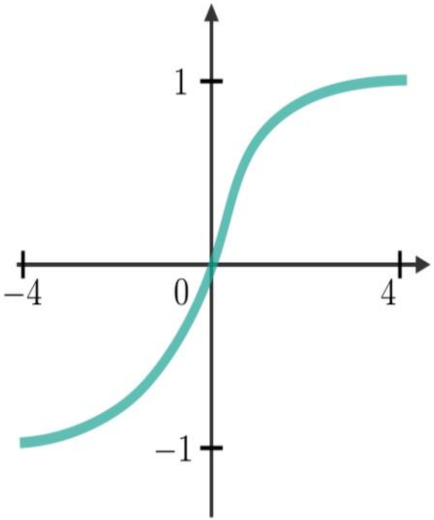
$$h_t = tanh(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh})$$

Tip: very similar to **linear.py** in HW1P1.

We are just applying a tanh function to linear transformations of x_t and h_{t-1}

RNN Cell Forward and Backward
Why tanh?

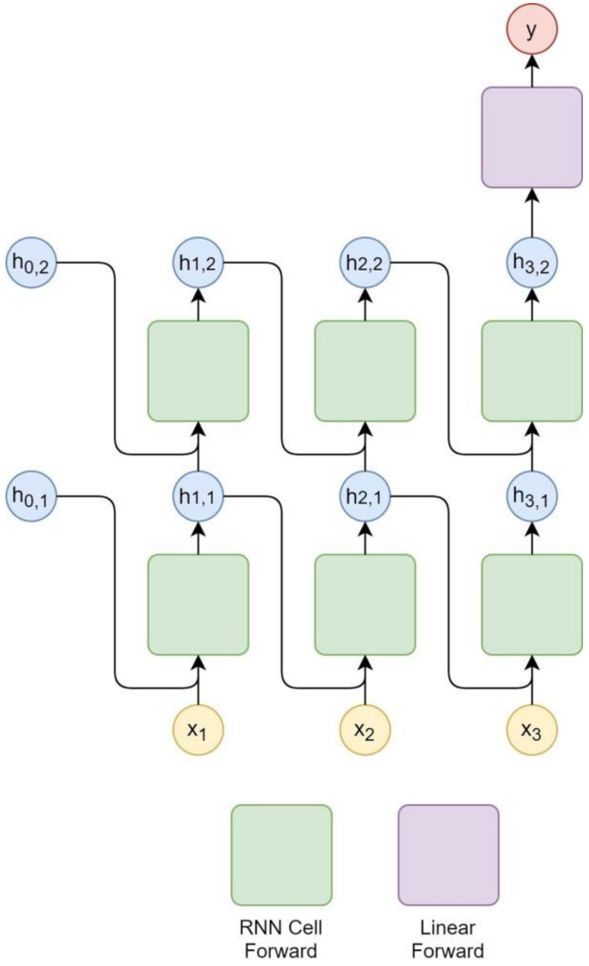
$$h_t = \boxed{tanh}(W_{ih}x_t + b_{ih} + W_{hh}h_{t-1} + b_{hh})$$



1. Non-linearity
2. Tanh is bounded; can mitigate exploding gradient problem

RNN Phoneme Classifier

- **Many-to-one** task
- Input sequence is passed through a few layers of **RNN cells**
- The final hidden state at the final timestamp is passed through a **linear layer** to give us the phoneme class



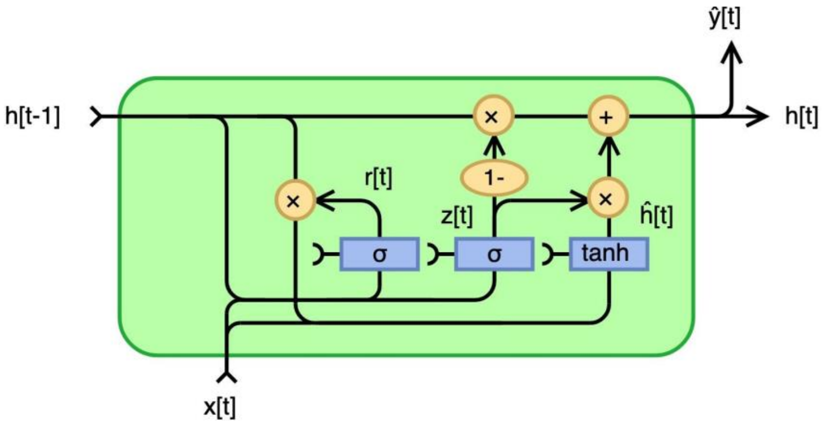
GRU Cell Forward and Backward

Reset Gate: $$\mathbf{r}_t = \sigma(\mathbf{W}_{rx} \cdot \mathbf{x}_t + \mathbf{b}_{rx} + \mathbf{W}_{rh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{rh})$$

Update Gate: $$\mathbf{z}_t = \sigma(\mathbf{W}_{zx} \cdot \mathbf{x}_t + \mathbf{b}_{zx} + \mathbf{W}_{zh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{zh})$$

Memory Content: $$\mathbf{n}_t = tanh(\mathbf{W}_{nx} \cdot \mathbf{x}_t + \mathbf{b}_{nx} + \mathbf{r}_t \odot (\mathbf{W}_{nh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{nh}))$$

Hidden State: $$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{n}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}$$



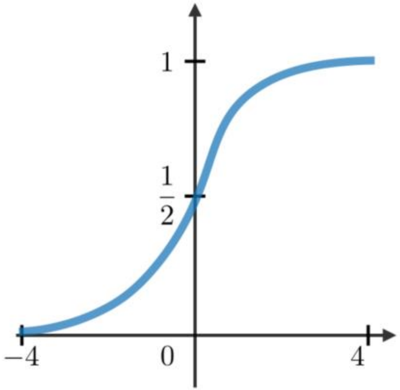
GRU Cell Forward and Backward contin.

Reset Gate: $\mathbf{r}_t = \boxed{\sigma}(\mathbf{W}_{rx} \cdot \mathbf{x}_t + \mathbf{b}_{rx} + \mathbf{W}_{rh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{rh})$

Update Gate: $\mathbf{z}_t = \boxed{\sigma}(\mathbf{W}_{zx} \cdot \mathbf{x}_t + \mathbf{b}_{zx} + \mathbf{W}_{zh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{zh})$

Memory Content: $\mathbf{n}_t = tanh(\mathbf{W}_{nx} \cdot \mathbf{x}_t + \mathbf{b}_{nx} + \mathbf{r}_t \odot (\mathbf{W}_{nh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{nh}))$

Hidden State: $\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{n}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}$



Why sigmoid?

- Sigmoid is limited to the values 0 to 1 → This can describe how much info to pass
  - Close to 0: we want to "forget"
  - Close to 1: we want to "remember"

GRU Cell Forward and Backward contin.

Reset Gate: $$\mathbf{r}_t = \sigma(\mathbf{W}_{rx} \cdot \mathbf{x}_t + \mathbf{b}_{rx} + \mathbf{W}_{rh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{rh})$$

Update Gate: $$\mathbf{z}_t = \sigma(\mathbf{W}_{zx} \cdot \mathbf{x}_t + \mathbf{b}_{zx} + \mathbf{W}_{zh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{zh})$$

Memory Content: $$\mathbf{n}_t = tanh(\mathbf{W}_{nx} \cdot \mathbf{x}_t + \mathbf{b}_{nx} + \boxed{\mathbf{r}_t \odot (\mathbf{W}_{nh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{nh}))}$$

Hidden State: $$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{n}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1}$$

We do an element-wise product with a linear transformation of the previous hidden-state

GRU Cell Forward and Backward contin.

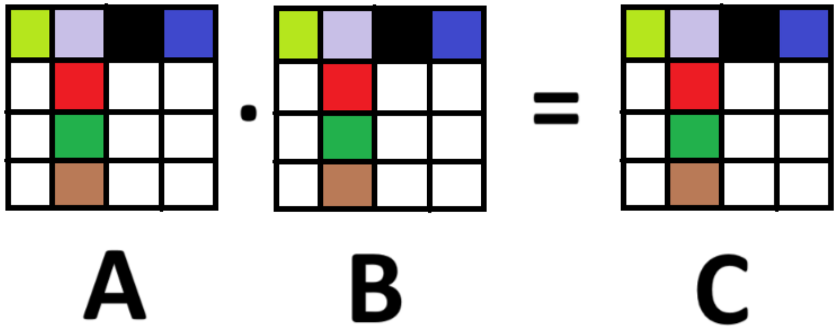
- GRU backward be the longest question in HW3P1
- Tips:
  - All intermediate **dWs** and **dbs** should be correct to make sure that your **dx** and **dh** are correct
  - Useful resource: How to compute a derivative
  - Break down complicated equations into unary/binary operations
  - e.g. $f(x) = \tanh(r \odot (Wx+b))$, we want to decompose it into:
    - $Z1 = Wx + b$
    - $Z2 = r \odot (Wx+b)$
    - $f(x) = \tanh(Z2)$
    - Derivative for each step would be easy and lastly we apply chain rule to get our $f'(x)$

Chain rule through element-wise multiplication

Assume that the shape of derivative wrt a matrix is the same as that of the matrix.

Let C = A ⊙ B (element-wise)

- This means A, B, and C have the same shape
- Only elements of the same position are related to each other → derivatives flow only position-wise.
- Therefore, dLdA = dLdC ⊙ B and dLdB = dLdC ⊙ A



Chain rule through matrix multiplication

Assume that the shape of derivative wrt a matrix is the same as that of the matrix.

Let C = AB (matrix multiplication). The shapes of A, B, C are *a* x *b*, *b* x *c*, and *c* x *a* respectively.

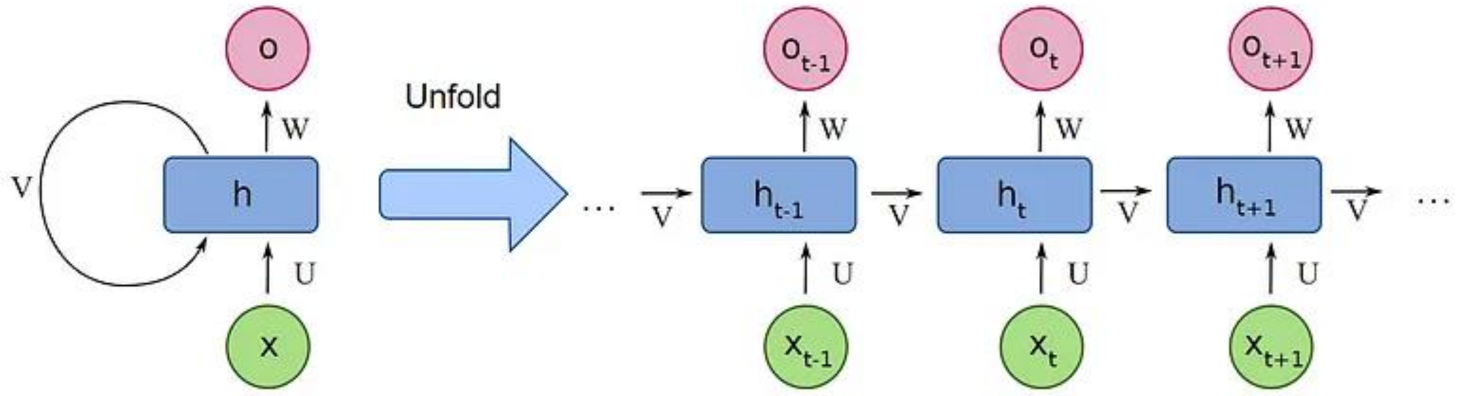
- Think about which all elements of C does A(*i*, *j*) influence.
- It influences all elements of C in row *i* through multiplication with the *j*-th element in every column of B.
- So, dLdA(*i*, *j*) = sum[*k*=1 to *c*] dLdC(*i*, *k*)B(*j*, *k*)
- Doing this for every element gives dLdA = dLdC X B.T (matrix multiplication)

DON'T JUST MATCH SHAPES. UNDERSTAND HOW VALUES MATCH INSTEAD. SHAPES WILL FOLLOW.

GRU Inference - Character Predictor

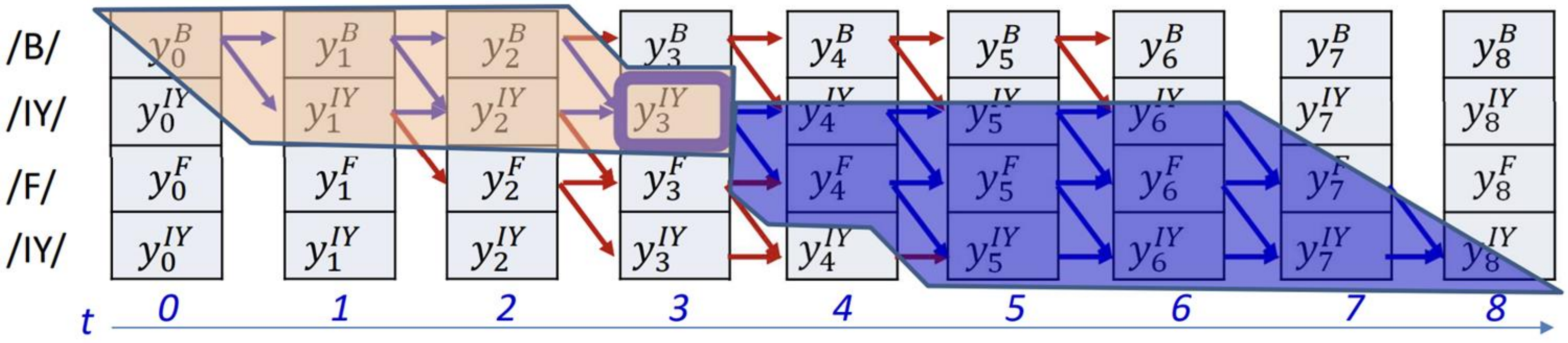
Many-to-many task, the model is supposed to have an output for each timestamp.

Different from RNN Phoneme classifier, here we need to pass the hidden state at **each** timestamp to a linear layer to predict the character at each t, instead of just the previous timestamp's hidden state.



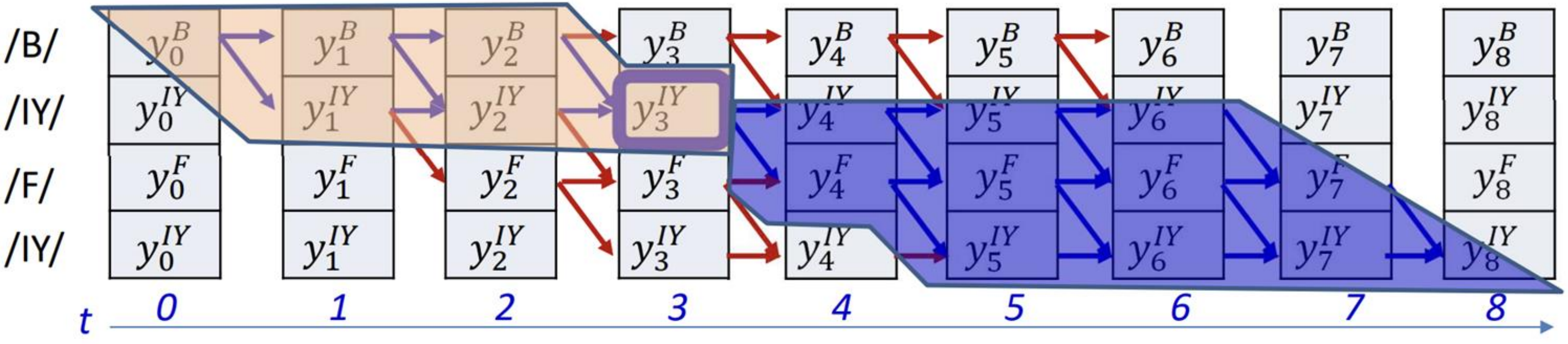
CTC

1. Extend Target With Blank
2. Forward Probabilities
3. Backward Probabilities
4. Posterior Probabilities
5. **CTC Loss**



CTC Introduction

Used to calculate loss when the **length of input sequences and output labels do not match** and there is no **fixed alignment between the input and output**.



CTC

1. Extend target with blank

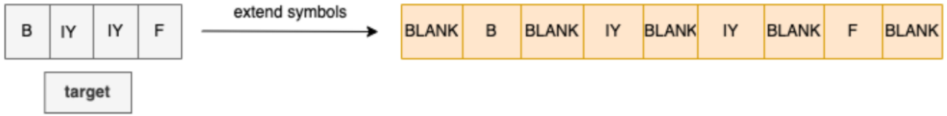


Figure 13: Extend symbols

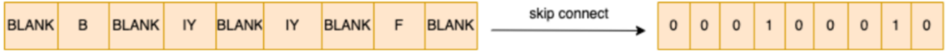


Figure 14: Skip connections

- Skips are permitted across a blank, but only if the symbols on either side are different
  - Because a blank is *mandatory between repetitions of a symbol* but *not required between distinct symbols*

CTC

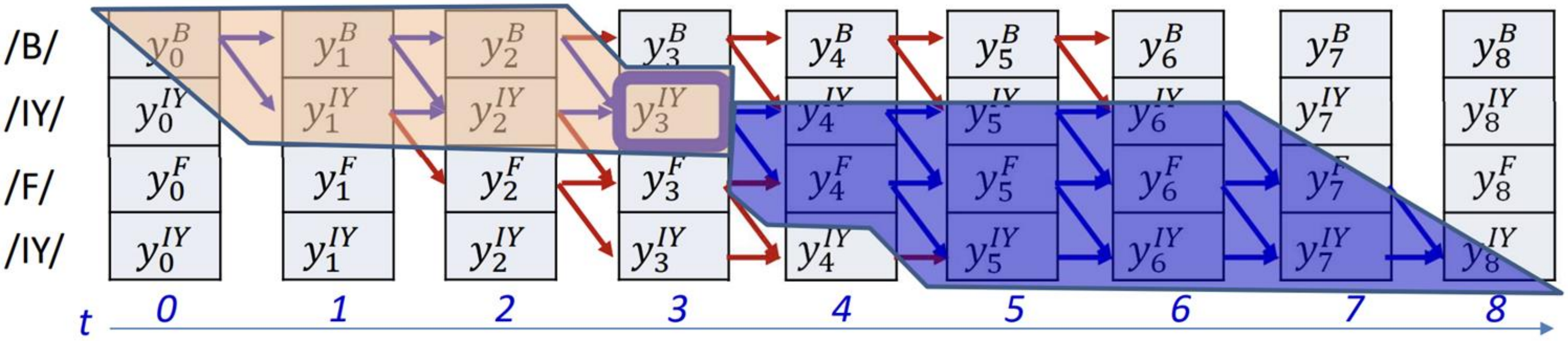
2. Forward Probabilities

$$\alpha(t,r) = P(S_0..S_r, s_t = S_r|\mathbf{X})$$

FORWARD ALGORITHM (with blanks)

```
[Sext] = extendedsequencewithblanks(S)
N = length(Sext) # Length of extended sequence

#The forward recursion
# First, at t = 1
alpha(1,1) = y(1,Sext(1)) #This is the blank
alpha(1,2) = y(1,Sext(2))
alpha(1,3:N) = 0
for t = 2:T
    alpha(t,1) = alpha(t-1,1) y(t,Sext(1))
    for i = 2:N
        alpha(t,i) = alpha(t-1,i) + alpha(t-1,i-1)
        if (i > 2 && Sext(i) != Sext(i-2))
            alpha(t,i) += alpha(t-1,i-2)
        alpha(t,i) *= y(t,Sext(i))
```

Without explicitly composing the output table



CTC

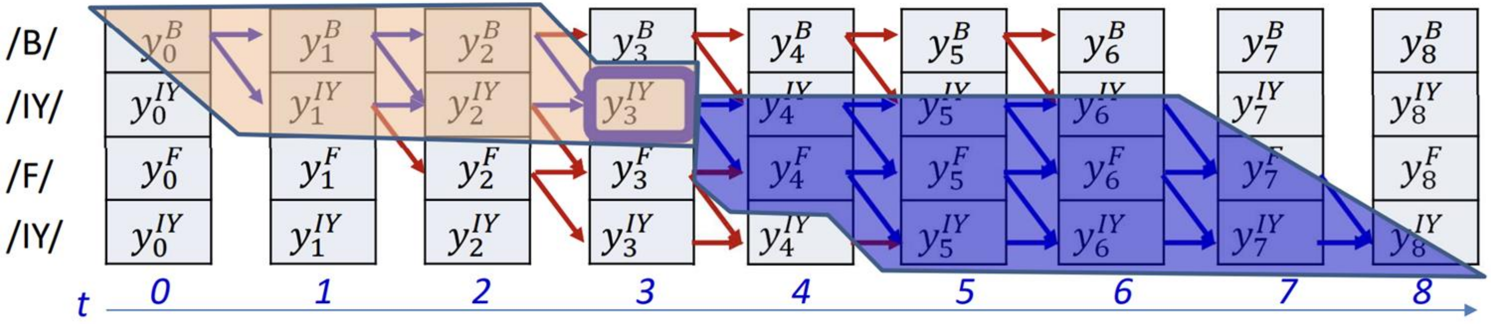
3. Backward Probabilities

$\hat{\beta}(t,r)$ = probability of graph including node at (t,r)

$$\beta(t,r) = \frac{1}{y_t^{S_r}}\hat{\beta}(t,r)$$

```
[Sext] = extendedsequencewithblanks(S)
N = length(Sext) # Length of extended sequence

#The backward recursion
# First, at t = T
betahat(T,N) = y(T,Sext(N))
betahat(T,N-1) = y(T,Sext(N-1))
betahat(T,1:N-2) = 0
for t = T-1 downto 1
    betahat(t,N) = betahat(t+1,N)*y(t,Sext(N))
    for i = N-1 downto 1
        betahat(t,i) = betahat(t+1,i) + betahat(t+1,i+1))
        if (i<=N-2 && Sext(i) != Sext(i+2))
            betahat(t,i) += betahat(t+1,i+2)
        betahat(t,i) *= y(t,Sext(i))

#Compute beta from betahat
for t = T downto 1
    for i = N downto 1
        beta(t,i) = betahat(t,i)/y(t,Sext(i))
```

Without explicitly composing the output table



CTC

4. Posterior Probability

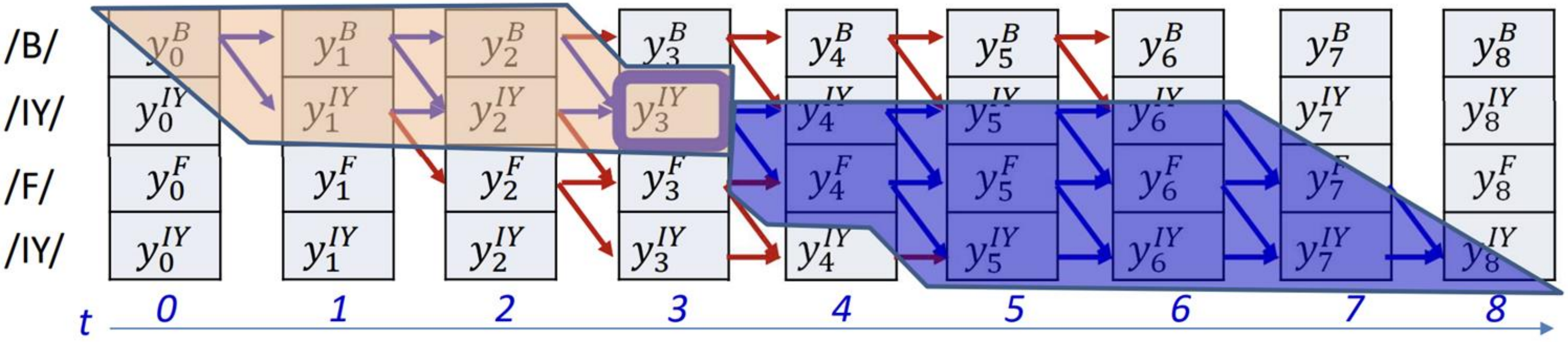
$$P(s_t = S_r, \mathbf{S}|\mathbf{X}) = \alpha(t,r)\beta(t,r)$$

- The *posterior* is given by

$$P(s_t = S_r|\mathbf{S},\mathbf{X}) = \frac{P(s_t = S_r, \mathbf{S}|\mathbf{X})}{\sum_{S'_r} P(s_t = S'_r, \mathbf{S}|\mathbf{X})} = \frac{\alpha(t,r)\beta(t,r)}{\sum_{r'} \alpha(t,r')\beta(t,r')}$$

COMPUTING POSTERIORS

```
#N is the number of symbols in the target output
#S(i) is the ith symbol in target output
#y(t,i) is the output of the network for the ith symbol at time t
#T = length of input

#Assuming the forward are completed first
alpha = forward(y, S)   # forward probabilities computed
beta  = backward(y, S)  # backward probabilities computed

#Now compute the posteriors
for t = 1:T
    sumgamma(t) = 0
    for i = 1:N
        gamma(t,i) = alpha(t,i) * beta(t,i)
        sumgamma(t) += gamma(t,i)
    end
    for i=1:N
        gamma(t,i) = gamma(t,i) / sumgamma(t)
```

CTC

5. Loss

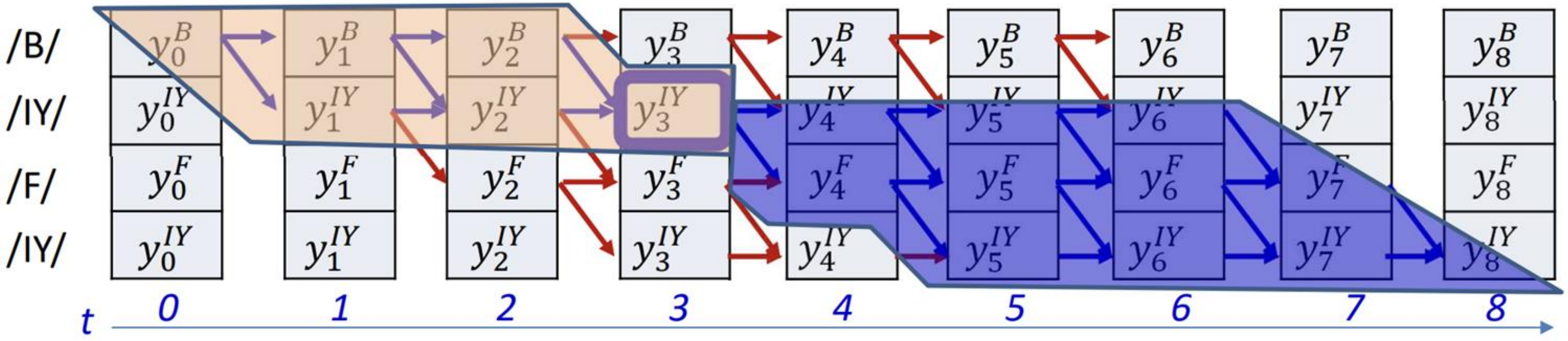
$$DIV = -\sum_t \sum_{s \in S_0 \ldots S_{K-1}} P(s_t = s|\mathbf{S},\mathbf{X}) \log Y(t, s_t = s)$$

$$DIV = -\sum_t \sum_r \gamma(t,r) \log y_t^{S(r)}$$

$$\frac{dDIV}{dy_t^l} = -\frac{1}{y_t^l} \sum_{r\,:S(r)=l} \gamma(t,r)$$

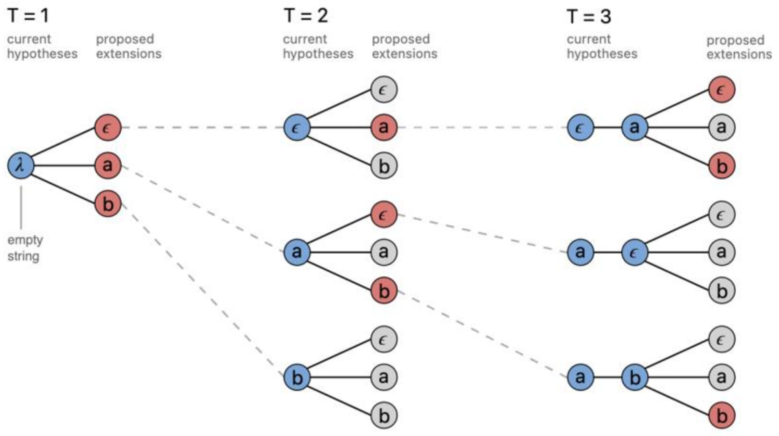
COMPUTING DERIVATIVES

```
#N is the number of symbols in the target output
#S(i) is the ith symbol in target output
#y(t,i) is the output of the network for the ith symbol at time t
#T = length of input

#Assuming the forward are completed first
alpha = forward(y, S)    # forward probabilities computed
beta  = backward(y, S)   # backward probabilities computed

# Compute posteriors from alpha and beta
gamma = computeposteriors(alpha, beta)

#Compute derivatives
for t = 1:T
    dy(t,1:L) = 0   # Initialize all derivatives at time t to 0
    for i = 1:N
        dy(t,S(i)) -= gamma(t,i) / y(t,S(i))
```

CTC Decoding

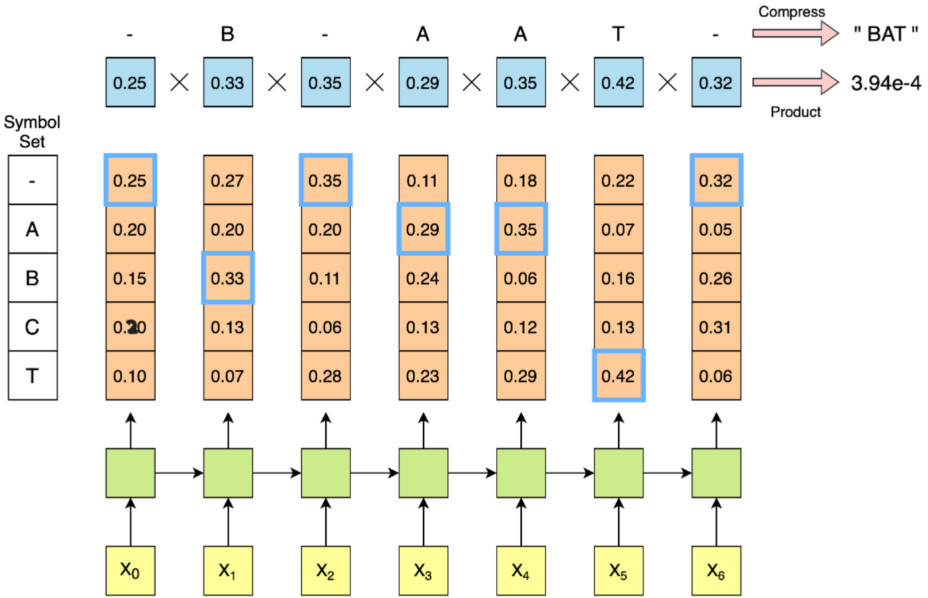
1. Greedy Search
2. Beam Search



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

Greedy Search

- Taking the most probable output at each time step



Beam Search

Efficient Beam Search:

Input: SymbolSets, y_probs, BeamWidth
Output: BestPath, MergedPathScores

0. Initialize:
   1. BestPaths with a blank symbol path with a score of 1.0.
   2. TempBestPaths as an empty dictionary.
1. For each timestep in y_probs:
   1. Extract the current symbol probabilities.
   2. For each path, score in BestPaths limited by BeamWidth:
      1. For each new symbol in the current symbol probabilities:
         1. Based on the last symbol of the path, determine the new path.
         2. Update the score for the new path in TempBestPaths.

   3. Update BestPaths with TempBestPaths.
   4. Clear TempBestPaths.

2. Initialize MergedPathScores as an empty dictionary.

3. For each path, score in BestPaths:
   1. Remove the ending blank symbol from the path.
   2. Update the score for the translated path in MergedPathScores.
   3. Update the BestPath and BestScore if the score is better.

4. Return BestPath and MergedPathScores.

Remember, when extending a path with a new symbol, you'll encounter three scenarios:

1. The new symbol is the same as the last symbol on the path.
2. The last symbol of the path is blank.
3. The last symbol of the path is different from the new symbol and is not blank.

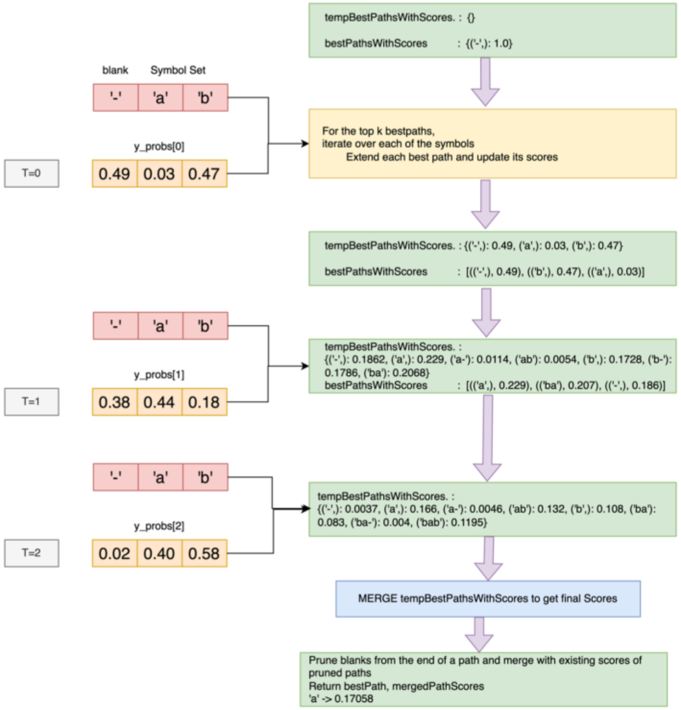
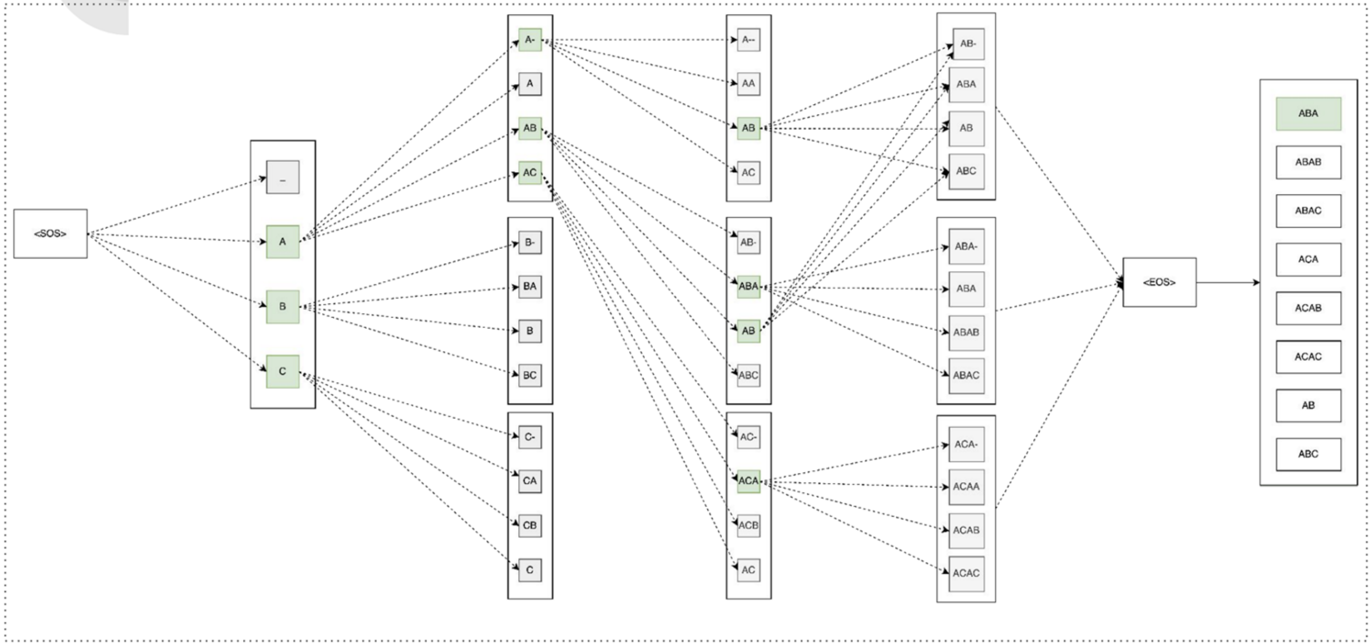


Figure 20: Efficient Beam Search procedure

Beam Search



<SOS>
_
A
B
C
A-
A
AB
AC
B-
BA
B
BC
C-
CA
CB
C
A--
AA
AB
AC
AB-
ABA
AB
ABC
AC-
ACA
ACB
AC
AB-
ABA
AB
ABC
ABA-
ABA
ABAB
ABAC
ACA-
ACAA
ACAB
ACAC
<EOS>
ABA
ABAB
ABAC
ACA
ACAB
ACAC
AB
ABC

Thank you!

Q&A