In this memo, we describe the forced alignment algorithm for a CTC acoustic model. In particular, let's assume for a particular utterance, we have

- log-posterior gram $\mathbf{X} = (\boldsymbol{x}_0, ..., \boldsymbol{x}_{T-1})$ where $\boldsymbol{x}_t \in \mathcal{R}^V$, and V is the vocabulary size, including the blank symbol. Without loss of generality, we assume the blank symbol index is 0.
- ground-truth label $\mathbf{Y} = (y_0, ..., y_{W-1})$, where $y_k \in [1, V-1]$ and W is the label length.

We use Viterbi algorithm to find out the optimal path $\boldsymbol{\pi} = (\pi_0, ..., \pi_{T-1})$ which maximize the following log-likelhood:

$$\boldsymbol{\pi}^* = \arg \max_{\boldsymbol{\pi}:\phi(\boldsymbol{\pi})=\boldsymbol{Y}} \mathcal{L}(\boldsymbol{\pi}, \boldsymbol{X}) = \arg \max_{\boldsymbol{\pi}:\phi(\boldsymbol{\pi})=\boldsymbol{Y}} \prod_{t=0}^{T-1} p(\pi_t|\boldsymbol{x}_t) \tag{1}$$

where π_t indicates the alignment of $\boldsymbol{x}_t$ to the ground truth sequence $\boldsymbol{Y}$. In CTC alignment, π_t can be either the blank symbol θ or a ground truth label y_w. To this end, we define a search lattice α of size $(2(W+1), T+1)$, which is shown below.

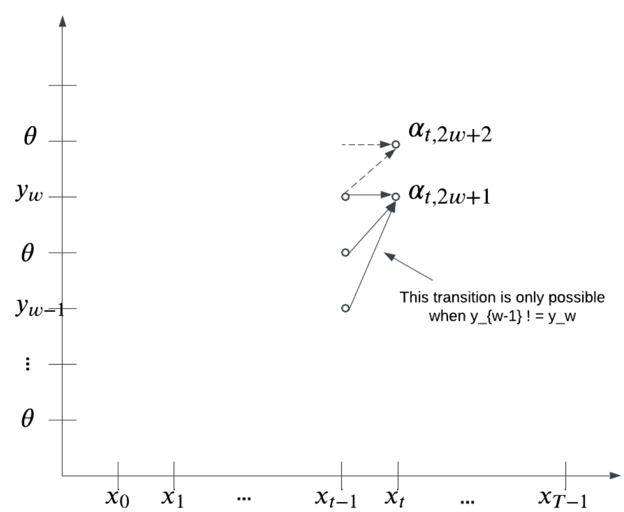

Figure 1: Search Lattice α. Its size is $(T+1, 2(W+1))$. At each time t, depending on w, there are 2 or 3 incoming transitions.

The search algorithm can be described as:

Algorithm 1: Viterbi forced-alignment algorithm (Non-vectorized).

1. Initialize search lattice $\boldsymbol{\alpha}$ *and back-trace table* $\boldsymbol{\beta}$:

$$\begin{aligned}\boldsymbol{\alpha} &= -\inf \\ \alpha_{0,0} &= 0.0 \\ \boldsymbol{\beta} &= \textbf{None}\end{aligned}$$

2. Iterate over time step:

```
for τ = 1, ..., T do
    for u = 1, ..., 2W + 1 do
        if u = 2w − 1 then
            // transit to a blank label
            prev_time_steps = {u, u − 1}
            α(τ, u) = max_{k∈prev_time_steps} α(τ − 1, k) + log p(θ|x_{τ−1})
        end
        if u = 2w then
            // transit to the (w − 1)-th label
            if y_w ≠ y_{w−1} then prev_time_steps = {u, u − 1, u − 2} ;
            else prev_time_steps = {u, u − 1} ;
            α(τ, u) = max_{k∈prev_time_steps} α(τ − 1, k) + log p(y_w|x_{τ−1})
        end
        β(τ, u) = arg max_{k∈prev_time_steps} α(τ − 1, k)
    end
end
```

3. Traceback:

```
if α(T, 2W + 1) = − inf and α(T, 2W) = − inf then
    return None; // No valid alignment can be found.
end
a_T = arg max_{k∈{2W+1,2W}} α(T, k);
for t = T - 1,..., 1 do
    a_t = β(t, a_{t+1})
end
for t = 0,..., T-1 do
    if a_{t+1} is Even then π_t = Y(a_{t+1}//2) ;
    else π_t = θ;
end
```

Algorithm 2: Viterbi forced-alignment algorithm – Vectorized and Jittable.

Input:

- `log_pos`, a $[B, T, V]$-shaped tensor, representing the log-posterior at each time step; `log_pos_padding`, a $[B, T]$-shaped 0/1 tensor.
- `label`, a $[B, W]$-shaped tensor; `label_padding`, a $[B, W]$-shaped 0/1 tensor.
- `blank_id`, a `int32` indicating the index of blank symbol.

Initialize:

- `log_pos = jnp.where(log_pos_paddings, -jnp.inf)`
- `search_lattices = jnp.full((B, T+1, 2(W+1), -jnp.inf)` and `search_lattices[:, 0, 0] = 0`
- Initialize a `[B, 2*(W+1), 3]`-shaped tensor, `prev_labels`, whose `(b, u)`'s slice indicates the indices of extended labels which can transit to the u-th extended label.
  - If u is odd, `prev_labels[:, u, :] = (u-1, u, -1)`
  - If u is even, and $u > 0$
    - If `label[b, u//2]`$\neq$ `label[b, u//2 - 1]`, then `prev_labels[b, u, :] = (u-2, u-1, u)`
    - Else, `prev_labels[b, u, :] = (u, u-1, -1)`
- Initialize a `[B, T+1, 2(W+1)]`-shaped `int32` tensor, `backtrace`, filled with value -1.
- Initialize a `[B, T+1]`-shaped tensor, `align`, filled with value -1; a `[B]`-shaped 0/1 tensor, `is_valid_align`, filled with value 0.

Forward Iterate:

```
for t = 1, ..., T do
    for u = 1, ..., 2W + 1 do
        if u = 2w − 1 then
            // transit to a blank label
            search_lattice[:, t, u] = max(search_lattice[:, t-1,
             prev_labels[:, u]], axis=-1) + log_pos[:, t, blank_id]
        else
            // transit to the (w − 1)-th label
            search_lattice[:, t, u] = max(search_lattice[:, t-1,
             prev_labels[:, u]], axis=-1) + log_pos[:, t, label[:, w - 1]]
        end
        backtrace[:, t, u]=argmax(
            search_lattice[:, t-1, prev_labels[:, u]])
    end
end
```

Traceback:

for $b = 0, \cdots, B-1$ **do**

  if *search_lattice[b, T+1, 2W+1]=-inf* *and* *search_lattice[b, T+1, 2W]=-inf* **then**

    `is_valid_align[b] = 0`

  else

    `align[b, T] = argmax(search_lattice[b, T, k], k = 2W or 2W + 1)`

    for t=*T-1*,$\cdots$,1 **do**

      align[b, t] = traceback[b, align[b, t+1]]

      if *align[b, t] is even* **then**

        align[b, t] = label[b, align[b, t]//2]

      else

        align[b, t] = blank_id

      end

    end

  end

end

Return: `align[:, 1:]` indicating the alignment of each frame, `is_valid_align` indicating whether there is an alignment for each sequence.