Hidden Markov Models

and the Forward-Backward Algorithm

Please See the Spreadsheet

- I like to teach this material using an interactive spreadsheet:
  - http://cs.jhu.edu/~jason/papers/#tnlp02
  - Has the spreadsheet and the lesson plan

- I'll also show the following slides at appropriate points.

Marginalization

SALES	Jan	Feb	Mar	Apr	...
Widgets	5	0	3	2	...
Grommets	7	3	10	8	...
Gadgets	0	0	1	0	...
...	...	...	...	...	...

Marginalization

Write the totals in the margins

SALES	Jan	Feb	Mar	Apr	…	TOTAL
Widgets	5	0	3	2	…	30
Grommets	7	3	10	8	…	80
Gadgets	0	0	1	0	…	2
…	…	…	…	…	…	
TOTAL	99	25	126	90		1000

Grand total

Marginalization

Given a random sale, what & when was it?

prob.	Jan	Feb	Mar	Apr	…	TOTAL
Widgets	.005	0	.003	.002	…	.030
Grommets	.007	.003	.010	.008	…	.080
Gadgets	0	0	.001	0	…	.002
…	…	…	…	…	…	
TOTAL	.099	.025	.126	.090		1.000

Grand total

Given a random sale, what & when was it?

Marginalization

joint prob: p(Jan,widget)

prob.	Jan	Feb	Mar	Apr	...	TOTAL
Widgets	.005	0	.003	.002	...	.030
Grommets	.007	.003	.010	.008	...	.080
Gadgets	0	0	.001	0	...	.002
...	...	...	...	...	...	
TOTAL	.099	.025	.126	.090		1.000

marginal prob: p(widget)

marginal prob: p(Jan)

marginal prob: p(anything in table)

Given a random sale in Jan., what was it?

Conditionalization

joint prob: p(Jan,widget)

Divide column through by Z=0.99 so it sums to 1

prob.	Jan	Feb	Mar	Apr	…	TOTAL	p(… | Jan)
Widgets	.005	0	.003	.002	…	.030	.005/.099
Grommets	.007	.003	.010	.008	…	.080	.007/.099
Gadgets	0	0	.001	0	…	.002	0
…	…	…	…	…	…		…
TOTAL	.099	.025	.126	.090		1.000	.099/.099

marginal prob: p(Jan)

conditional prob: p(widget|Jan)=.005/.099

Marginalization & conditionalization in the weather example

- Instead of a 2-dimensional table, now we have a 66-dimensional table:
  - 33 of the dimensions have 2 choices: {C,H}
  - 33 of the dimensions have 3 choices: {1,2,3}
- Cross-section showing just 3 of the dimensions:

$Weather_3=C$

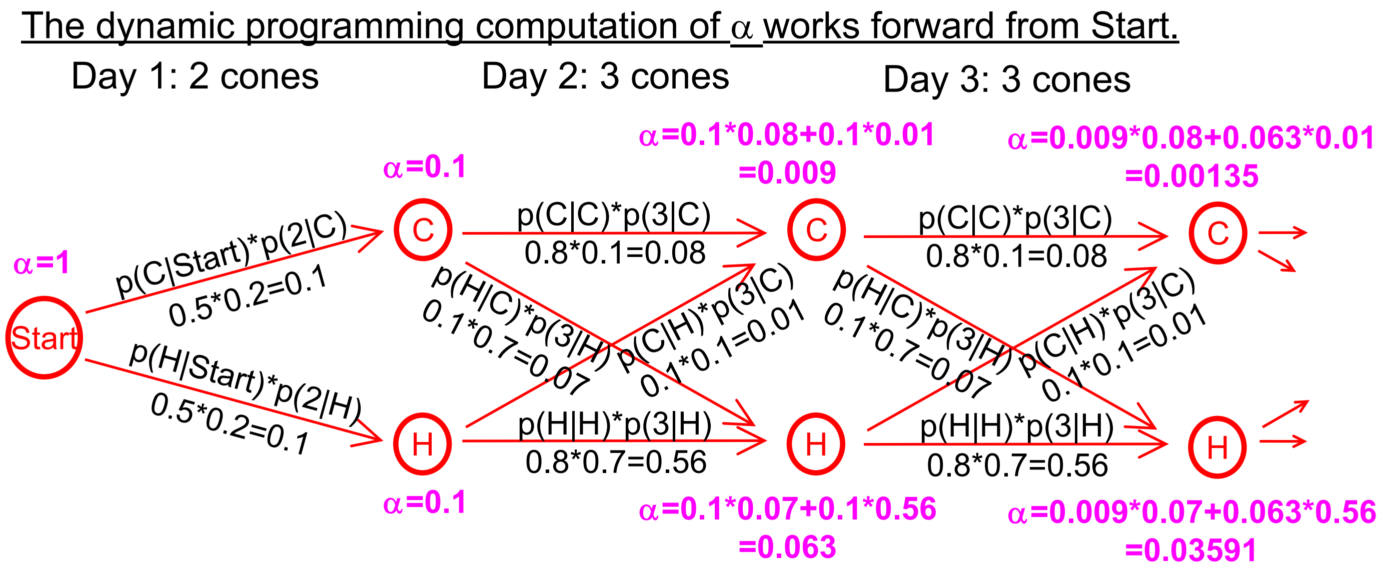
$Weather_3=H$

	$Weather_2=C$	$Weather_2=H$
$IceCream_2=1$	0.000…	0.000…
$IceCream_2=2$	0.000…	0.000…
$IceCream_2=3$	0.000…	0.000…

Interesting probabilities in the weather example

- **Prior** probability of weather:
  p(Weather=CHH…)
- **Posterior** probability of weather (after observing evidence):
  p(Weather=CHH… | IceCream=233…)
- **Posterior marginal** probability that day 3 is hot:
  p(Weather_3=H | IceCream=233…)
  $= \sum_{w \text{ such that } w_3=H}$ p(Weather=w | IceCream=233…)
- **Posterior conditional** probability that day 3 is hot if day 2 is:
  p(Weather_3=H | Weather_2=H, IceCream=233…)

The HMM trellis

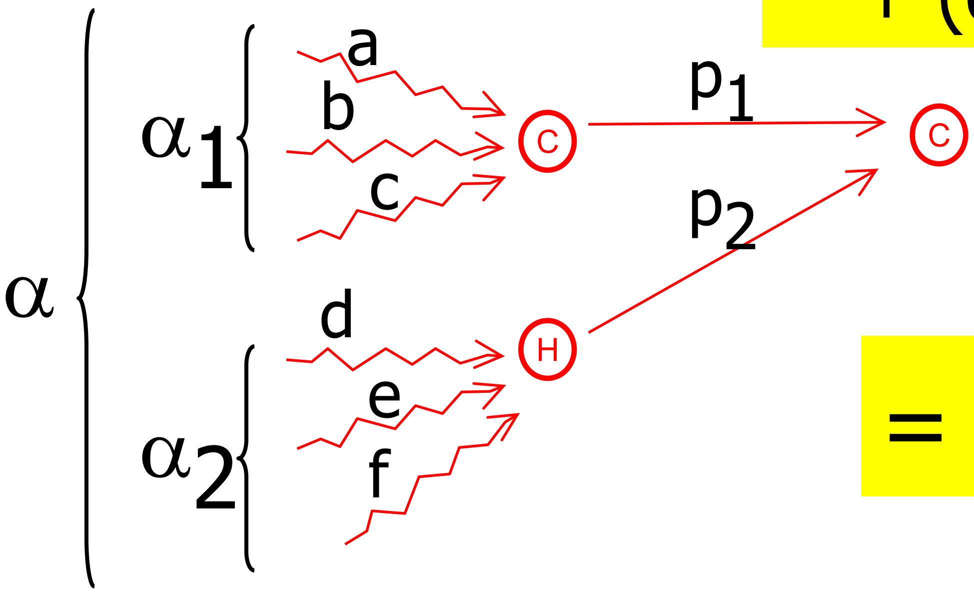


- This "trellis" graph has 2^{33} paths.
  - These represent all possible weather sequences that could explain the observed ice cream sequence 2, 3, 3, …
- What is the product of all the edge weights on one path H, H, H, …?
  - Edge weights are chosen to get p(weather=H,H,H,… & icecream=2,3,3,…)
- What is the α probability at each state?
  - It's the total probability of all paths from Start to that state.
  - How can we compute it fast when there are many paths?

Computing α Values

All paths to state:

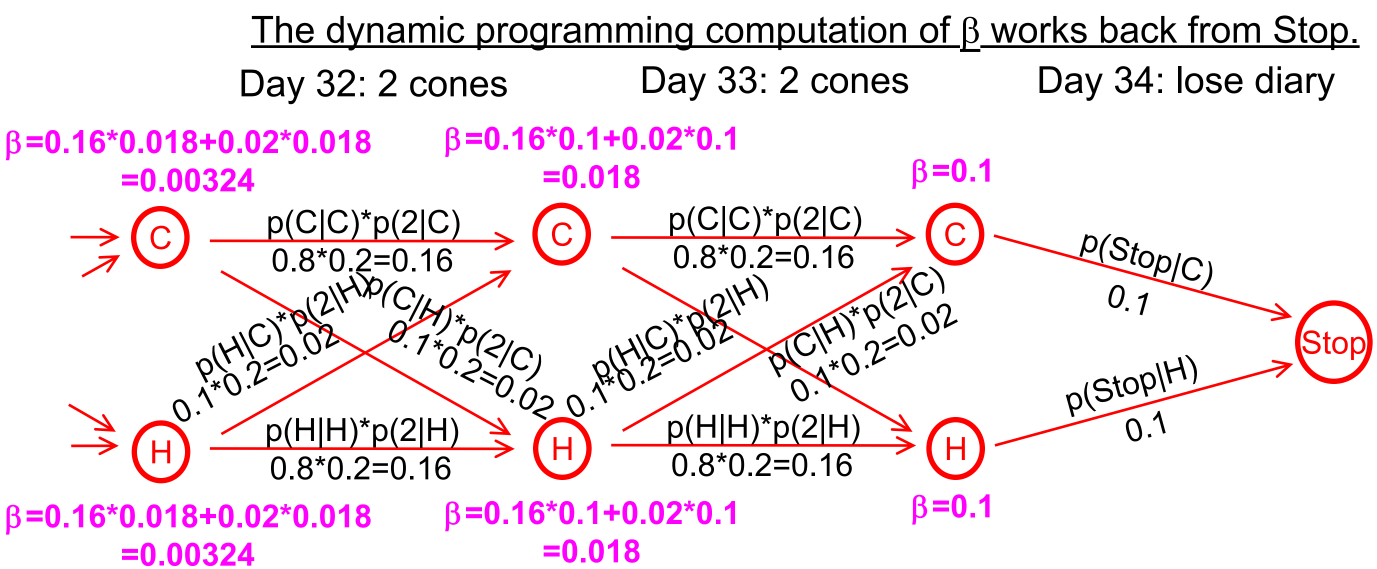
$$\alpha = (ap_1 + bp_1 + cp_1) + (dp_2 + ep_2 + fp_2)$$



$$= \alpha_1 \cdot p_1 + \alpha_2 \cdot p_2$$

Thanks, distributive law!

The HMM trellis

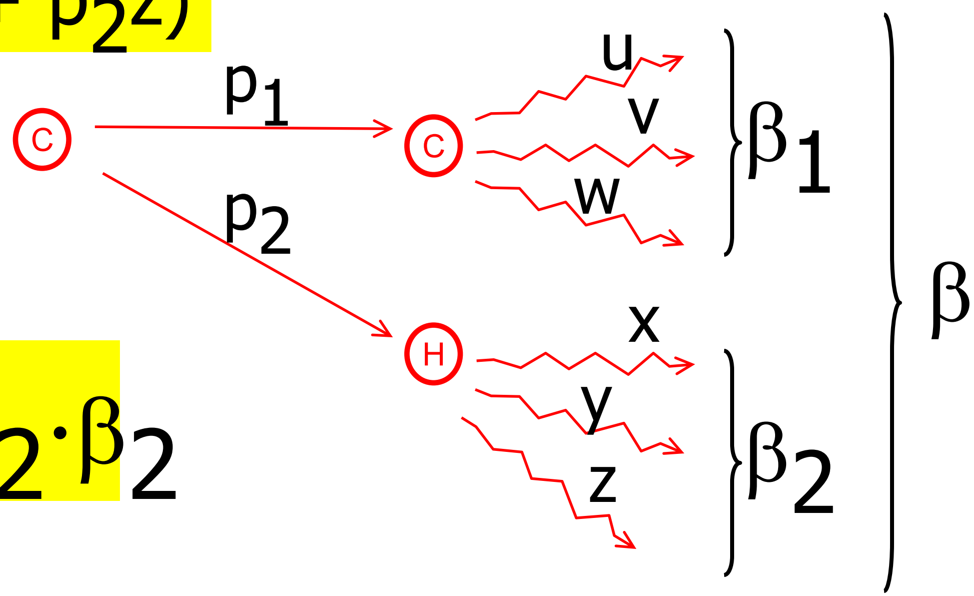


- What is the β probability at each state?
  - It's the total probability of all paths from that state to Stop
  - How can we compute it fast when there are many paths?

Computing β Values

All paths from state:

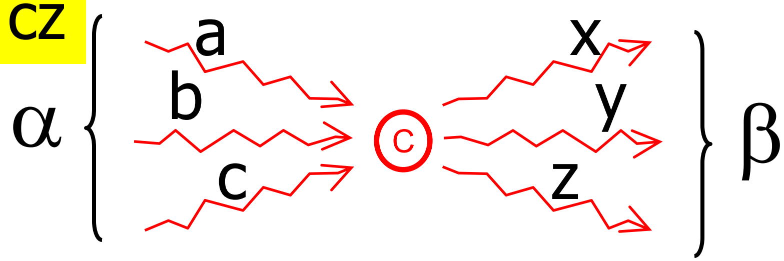
$$\beta = (p_1 u + p_1 v + p_1 w) + (p_2 x + p_2 y + p_2 z)$$



$$= p_1 \cdot \beta_1 + p_2 \cdot \beta_2$$

Computing State Probabilities

All paths through state:
ax + ay + az
+ bx + by + bz
+ cx + cy + cz



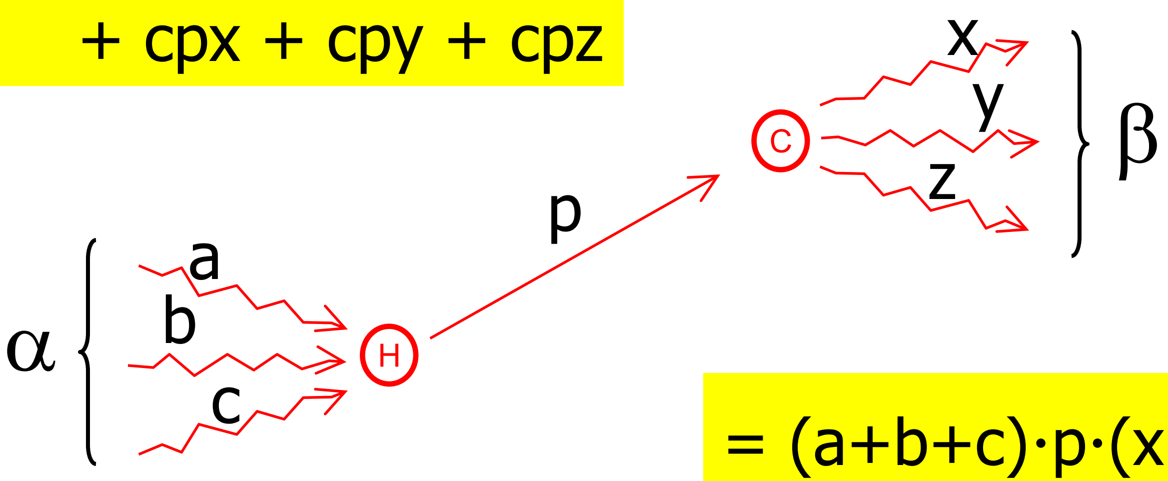
$= (a+b+c)\cdot(x+y+z)$

$= \quad \alpha(C) \cdot \beta(C)$

Thanks, distributive law!

Computing Arc Probabilities

All paths through the p arc:
apx + apy + apz
+ bpx + bpy + bpz
+ cpx + cpy + cpz



$= (a+b+c) \cdot p \cdot (x+y+z)$

$= \quad \alpha(H) \cdot p \cdot \beta(C)$

Thanks, distributive law!

Posterior tagging

- Give each word its highest-prob tag according to forward-backward.
  - Do this independently of other words.
  - **Det Adj** **0.35** ← exp # correct tags = 0.55+0.35 = 0.9
  - **Det N** **0.2** ← exp # correct tags = 0.55+0.2 = 0.75
  - **N V** **0.45** ← exp # correct tags = 0.45+0.45 = 0.9
- Output is
  - **Det V** **0** ← exp # correct tags = 0.55+0.45 = 1.0
- Defensible: maximizes expected # of correct tags.
- But not a coherent sequence. May screw up subsequent processing (e.g., can't find any parse).

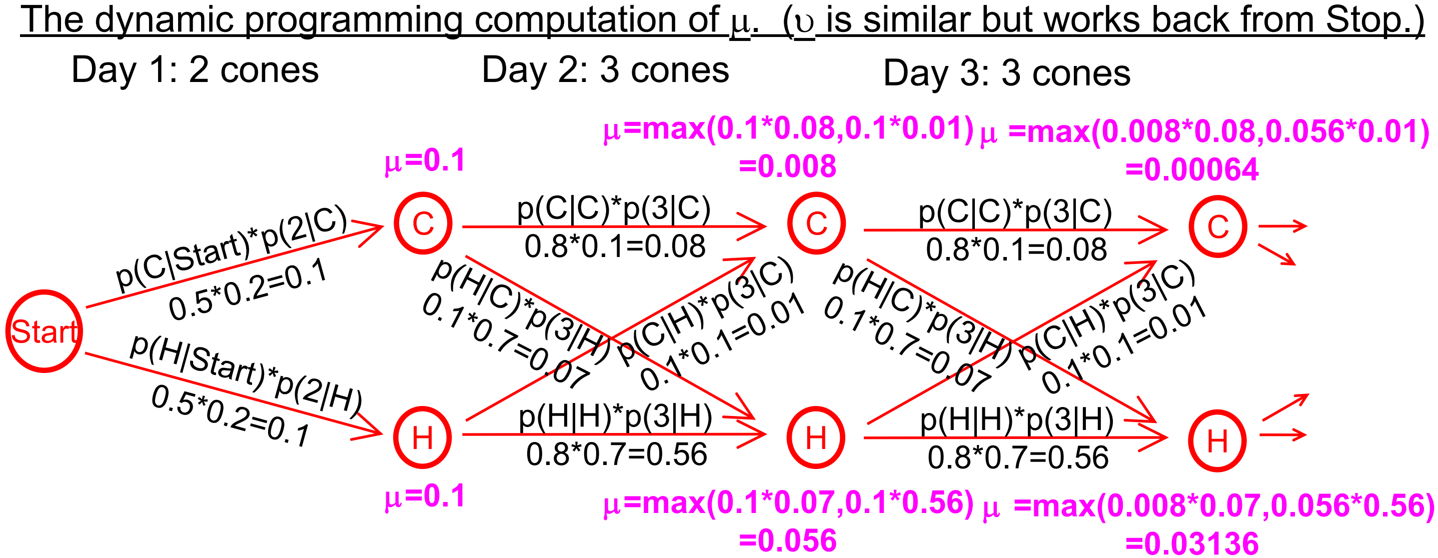
Alternative: Viterbi tagging

- *Posterior tagging:* Give each word its highest-prob tag according to forward-backward.
  - **Det Adj 0.35**
  - **Det N 0.2**
  - **N V 0.45**
- *Viterbi tagging:* Pick the single best tag sequence (best path):
  - **N V 0.45**
- Same algorithm as forward-backward, but uses a semiring that **maximizes** over paths instead of **summing** over paths.

Max-product instead of sum-product

Use a semiring that maximizes over paths instead of summing.

We write μ,ν instead of α,β for these "Viterbi forward" and "Viterbi backward" probabilities."



$\alpha*\beta$ at a state = total prob of all paths through that state

$\mu*\nu$ at a state = max prob of any path through that state

Suppose max prob path has prob p: how to print it?

- Print the state at each time step with highest $\mu*\nu$ (= p); works if no ties
- Or, compute μ values from left to right to find p, then follow backpointers