Lecture 8: Part of Speech Tagging – Viterbi, Forward, Backward, Forward-Backward, Baum-Welch

Instructor: Jackie CK Cheung & David Adelani

COMP-550

J&M Ch. 7.2-7.3, 8.5 (1st ed); J&M Ch. 5.5, 6.1–6.5 (2nd ed); J&M Ch. 8.4 (3rd ed)

Outline

Hidden Markov models: review of last class

Things to do with HMMs:

- Forward algorithm
- Backward algorithm
- Viterbi algorithm
- Baum-Welch as Expectation Maximization

Parts of Speech in English

Nouns	*restaurant, me, dinner*
Verbs	*find, eat, is*
Adjectives	*good, vegetarian*
Prepositions	*in, of, up, above*
Adverbs	*quickly, well, very*
Determiners	*the, a, an*

Sequence Labelling

Predict labels for *an entire sequence of inputs*:

? ? ? ? ? ? ? ? ? ? ?

Pierre Vinken , 61 years old , will join the board …

↓

NNP NNP , CD NNS JJ , MD VB DT NN

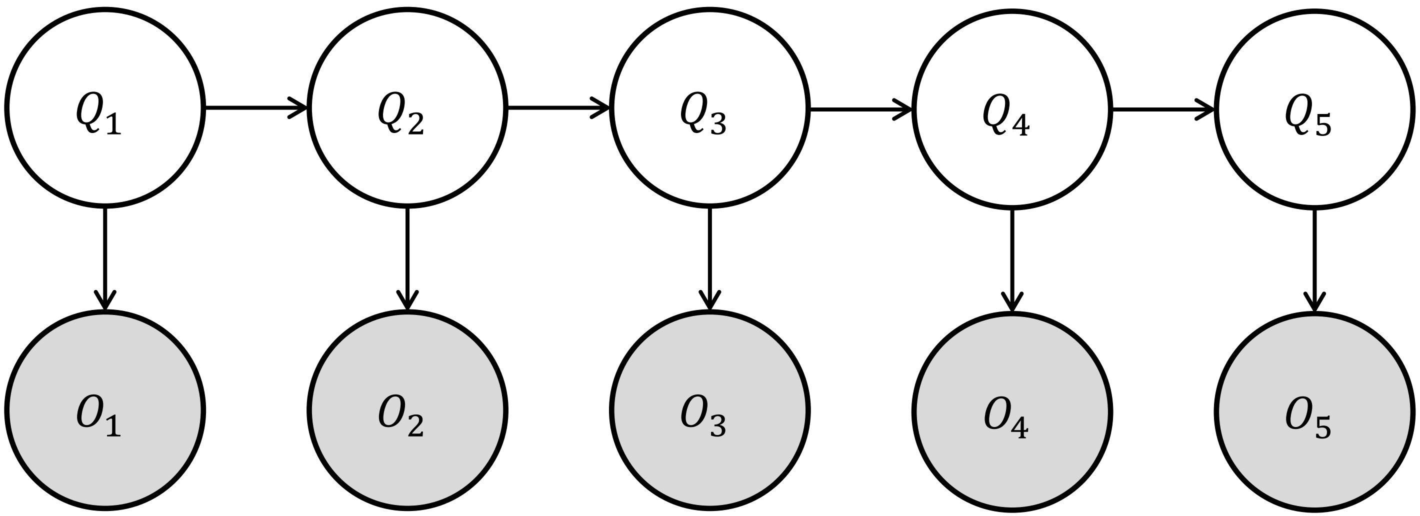
Pierre Vinken , 61 years old , will join the board …

Must consider:

- Current word
- Previous context

Decomposing the Joint Probability

Graph specifies how join probability decomposes



$$P(\boldsymbol{O}, \boldsymbol{Q}) = P(Q_1) \prod_{t=1}^{T-1} P(Q_{t+1}|Q_t) \prod_{t=1}^{T} P(O_t|Q_t)$$

Initial state probability

State transition probabilities

Emission probabilities

Model Parameters

Let there be N possible tags, W possible words

Parameters θ has three components:

1. Initial probabilities for Q_1:

   $\Pi = \{\pi_1, \pi_2, \ldots, \pi_N\}$ (categorical)

2. Transition probabilities for Q_t to Q_{t+1}:

   $A = \{a_{ij}\}\ i, j \in [1, N]$ (categorical)

3. Emission probabilities for Q_t to O_t:

   $B = \{b_i(w_k)\}\ i \in [1, N], k \in [1, W]$ (categorical)

How many distributions and values of each type are there?

Model Parameters' MLE

Recall categorical distributions' MLE:

$$P(\text{outcome i}) = \frac{\#(\text{outcome i})}{\#(\text{all events})}$$

For our parameters:

$$\pi_i = P(Q_1 = i) = \frac{\#(Q_1 = i)}{\#(\text{sentences})}$$

$$a_{ij} = P(Q_{t+1} = j \mid Q_t = i) = \#(i, j) \,/\, \#(i)$$

$$b_{ik} = P(O_t = k \mid Q_t = i) = \#(\text{word } k, \text{tag } i) \,/\, \#(i)$$

Questions for an HMM

1. Compute likelihood of a sequence of observations, $P(\boldsymbol{O}|\theta)$

   Forward algorithm, backward algorithm

2. What state sequence best explains a sequence of observations?

   $$\underset{\boldsymbol{Q}}{\operatorname{argmax}}\, P(\boldsymbol{Q}, \boldsymbol{O}|\theta)$$

   Viterbi algorithm

3. Given an observation sequence (without labels), what is the best model for it?

   Forward-backward algorithm
   Baum-Welch algorithm
   Expectation Maximization

Q1: Compute Likelihood

Marginalize over all possible state sequences

$$P(\boldsymbol{O}|\,\theta) = \sum_{\boldsymbol{Q}} P(\boldsymbol{O}, \boldsymbol{Q}\,|\theta)$$

Problem: Exponentially many paths (N^T) since we have N states, T observations

Solution: **Forward algorithm**

- *Dynamic programming* to avoid unnecessary recalculations
- Create a table of all the possible state sequences, annotated with probabilities

Forward Algorithm

Trellis of possible state sequences

States		o_1	o_2	o_3	o_4	o_5
	VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
	NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
	DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
	JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
	CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

Time

$\alpha_i(t)$ is $P(\boldsymbol{O}_{1:t}, Q_t = i|\theta)$

- Probability of current tag and words up to now

First Column

$\alpha_i(t)$is $P(\boldsymbol{O}_{1:t}, Q_t = i|\theta)$

States	$\boldsymbol{O_1}$	O_2	O_3	O_4	O_5
VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
DT	$\boldsymbol{\alpha_{DT}(1)}$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

$$\boxed{\alpha_j(1) = \pi_j b_j(O_1)}$$

Time

Consider:

- Initial state probability
- First emission

Middle Cells

$\alpha_i(t)$is $P(\boldsymbol{O}_{1:t}, Q_t = i|\theta)$

States	O_1	O_2	$\boldsymbol{O_3}$	O_4	O_5
VB	$\alpha_{VB}(1)$	$\boldsymbol{\alpha_{VB}(2)}$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\alpha_{VB}(5)$
NN	$\alpha_{NN}(1)$	$\boldsymbol{\alpha_{NN}(2)}$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\alpha_{NN}(5)$
DT	$\alpha_{DT}(1)$	$\boldsymbol{\alpha_{DT}(2)}$	$\boldsymbol{\alpha_{DT}(3)}$	$\alpha_{DT}(4)$	$\alpha_{DT}(5)$
JJ	$\alpha_{JJ}(1)$	$\boldsymbol{\alpha_{JJ}(2)}$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\alpha_{JJ}(5)$
CD	$\alpha_{CD}(1)$	$\boldsymbol{\alpha_{CD}(2)}$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\alpha_{CD}(5)$

Time

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_j(O_t)$$

Consider:

- All possible ways to get to current state
- Emission from current state

After Last Column

$\alpha_i(t)$ is $P(\boldsymbol{O}_{1:t}, Q_t = i|\theta)$

States

	O_1	O_2	O_3	O_4	O_5
VB	$\alpha_{VB}(1)$	$\alpha_{VB}(2)$	$\alpha_{VB}(3)$	$\alpha_{VB}(4)$	$\boldsymbol{\alpha_{VB}(5)}$
NN	$\alpha_{NN}(1)$	$\alpha_{NN}(2)$	$\alpha_{NN}(3)$	$\alpha_{NN}(4)$	$\boldsymbol{\alpha_{NN}(5)}$
DT	$\alpha_{DT}(1)$	$\alpha_{DT}(2)$	$\alpha_{DT}(3)$	$\alpha_{DT}(4)$	$\boldsymbol{\alpha_{DT}(5)}$
JJ	$\alpha_{JJ}(1)$	$\alpha_{JJ}(2)$	$\alpha_{JJ}(3)$	$\alpha_{JJ}(4)$	$\boldsymbol{\alpha_{JJ}(5)}$
CD	$\alpha_{CD}(1)$	$\alpha_{CD}(2)$	$\alpha_{CD}(3)$	$\alpha_{CD}(4)$	$\boldsymbol{\alpha_{CD}(5)}$

Time

$$\boxed{P(\boldsymbol{O}|\theta) = \sum_{j=1}^{N} \alpha_j(T)}$$

Sum over last column for overall likelihood

Forward Algorithm Summary

Create trellis $\alpha_i(t)$ for $i = 1 \ldots N, t = 1 \ldots T$

$\alpha_j(1) = \pi_j b_j(O_1)$ for j = 1 … N

for t = 2 … T:

for j = 1 … N:

$$\alpha_j(t) = \sum_{i=1}^{N} \alpha_i(t-1) a_{ij} b_j(O_t)$$

$$P(\boldsymbol{O}|\theta) = \sum_{j=1}^{N} \alpha_j(T)$$

Runtime: O($N^2 T$)

Backward Algorithm

Nothing stops us from going backwards too!

- This is not just for fun, as we'll see later.

Define new trellis with cells $\beta_i(t)$

$$\beta_i(t) = P(\boldsymbol{O}_{t+1:T} | Q_t = i, \theta)$$

- Probability of all subsequent words, given current tag is i.
- Note that unlike $\alpha_i(t)$, it *excludes* the current word

Backward Algorithm Summary

Create trellis $\beta_i(t)$ for $i = 1 \dots N, t = 1 \dots T$

$\beta_i(T) = 1$ for i = 1 … N

for t = T-1 … 1:

for i = 1 … N:

Remember $\beta_j(t+1)$ does not include $b_j(O_{t+1})$, so we need to add this factor in!

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_j(t+1)$$

$$P(\boldsymbol{O}|\theta) = \sum_{i=1}^{N} \pi_i b_i(O_1) \beta_i(1)$$

Runtime: O($N^2 T$)

Forward and Backward

$\alpha_i(t) = P(\boldsymbol{O}_{1:t}, Q_t = i|\theta)$
$\beta_i(t) = P(\boldsymbol{O}_{t+1:T}|Q_t = i, \theta)$

That means

$$\alpha_i(t)\beta_i(t) = P(\boldsymbol{O}, Q_t = i|\theta)$$

Probability of the *entire* sequence of observations, and we are in state i at timestep t.

Thus,

$P(\boldsymbol{O}|\theta) = \sum_{i=1}^{N} \alpha_i(t)\beta_i(t)$ for any $t\ =\ 1\ \dots\ T$

Working in the Log Domain

Practical note: need to avoid underflow—work in log domain

$$\log\left(\prod p_i\right) = \sum \log p_i = \sum a_i$$

$a_i = \log p_i$

Log sum trick

$$\log\left(\sum p_i\right) = \log\left(\sum \exp a_i\right)$$

Let $b = \max a_i$

$$\log\left(\sum \exp a_i\right) = \log\left[\exp(b) \sum \exp(a_i - b)\right]$$
$$= b + \log \sum \exp(a_i - b)$$

Q2: Sequence Labelling

Find most likely state sequence for a sample

$$\boldsymbol{Q}^* = \text{argmax}_{\boldsymbol{Q}}\, P(\boldsymbol{Q}, \boldsymbol{O} | \theta)$$

Intuition: use forward algorithm, but replace summation with max

This is now called the **Viterbi algorithm**.

Trellis cells:

$$\delta_i(t) = \max_{Q_{1:t-1}} P(Q_{1:t-1}, O_{1:t}, Q_t = i | \theta)$$

Viterbi Algorithm Summary

Create trellis $\delta_i(t)$ for $i = 1 \dots N, t = 1 \dots T$

$\delta_j(1) = \pi_j b_j(O_1)$ for j = 1 … N

for t = 2 … T:

    for j = 1 … N:

$$\delta_j(t) = \max_i \delta_i(t-1) a_{ij} b_j(O_t)$$

Take $\max_i \delta_i(T)$

Runtime: O($N^2 T$)

Backpointers

States	o_1	o_2	o_3	o_4	o_5
VB	$\delta_{VB}(1)$	$\boldsymbol{\delta_{VB}(2)}$	$\delta_{VB}(3)$	$\delta_{VB}(4)$	$\delta_{VB}(5)$
NN	$\boldsymbol{\delta_{NN}(1)}$	$\delta_{NN}(2)$	$\delta_{NN}(3)$	$\delta_{NN}(4)$	$\boldsymbol{\delta_{NN}(5)}$
DT	$\delta_{DT}(1)$	$\delta_{DT}(2)$	$\boldsymbol{\delta_{DT}(3)}$	$\delta_{DT}(4)$	$\delta_{DT}(5)$
JJ	$\delta_{JJ}(1)$	$\delta_{JJ}(2)$	$\delta_{JJ}(3)$	$\boldsymbol{\delta_{JJ}(4)}$	$\delta_{JJ}(5)$
CD	$\delta_{CD}(1)$	$\delta_{CD}(2)$	$\delta_{CD}(3)$	$\delta_{CD}(4)$	$\delta_{CD}(5)$

Time

- Keep track of where the max entry to each cell came from
- Work backwards to recover best label sequence

Exercise

3 states (X, Y, Z), 2 possible emissions (!,@)

$$\begin{array}{cccc} & X & Y & Z \\ \pi = & \langle 0.2 & 0.5 & 0.3 \rangle \end{array}$$

$$\text{From}\ \begin{array}{c} X \\ Y \\ Z \end{array} \ A = \begin{bmatrix} 0.5 & 0.4 & 0.1 \\ 0.2 & 0.3 & 0.5 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

$$B = \begin{array}{c} \\ X \\ Y \\ Z \end{array} \begin{array}{cc} ! & @ \\ \end{array}$$
$$\begin{array}{c} X \\ Y \\ Z \end{array} \begin{bmatrix} 0.1 & 0.9 \\ 0.5 & 0.5 \\ 0.7 & 0.3 \end{bmatrix}$$

Run the forward, backward, and Viterbi algorithms on the emission sequence "!@@"

Forward Algorithm

	!	@	@
X	0.2 * 0.1 = 0.02	0.02 * 0.5 * 0.9 + 0.25 * 0.2 * 0.9 + 0.21 * 0.1 * 0.9 = 0.0729	.0729 * 0.5 * 0.9 +.052 * 0.2 * 0.9 +.0885 * 0.1 * 0.9 = 0.05013
Y	0.5 * 0.5 = 0.25	0.02 * 0.4 * 0.5 + 0.25 * 0.3 * 0.5 + 0.21 * 0.1 * 0.5 = 0.052	.0729 * 0.4 * 0.5 + .052 * 0.3 * 0.5 + .0885 * 0.1 * 0.5 = 0.026805
Z	0.3 * 0.7 = 0.21	0.02 * 0.1 * 0.3 + 0.25 * 0.5 * 0.3 + 0.21 * 0.8 * 0.3 = 0.0885	.0729 * 0.1 * 0.3 + .052 * 0.5 * 0.3 + .0885 * 0.8 * 0.3 = 0.031227

Sum: 0.108162

Backward Algorithm

	!	@	@
X	0.68 * 0.5 * 0.9 + 0.48 * 0.4 * 0.5 + 0.38 * 0.1 * 0.3 =0.4134	1 * 0.5 * 0.9 + 1 * 0.4 * 0.5 + 1 * 0.1 * 0.3 = 0.68	1
Y	0.68 * 0.2 * 0.9 + 0.48 * 0.3 * 0.5 + 0.38 * 0.5 * 0.3 = 0.2514	1 * 0.2 * 0.9 + 1 * 0.3 * 0.5 + 1 * 0.5 * 0.3 = 0.48	1
Z	0.68 * 0.1 * 0.9 + 0.48 * 0.1 * 0.5 + 0.38 * 0.8 * 0.3 = 0.1764	1 * 0.1 * 0.9 + 1 * 0.1 * 0.5 + 1 * 0.8 * 0.3 = 0.38	1

P(seq) = 0.2 * 0.4134 * 0.1
+ 0.5 * 0.2514 * 0.5
+ 0.3 * 0.1764 * 0.7 = 0.108162

Viterbi Algorithm

	!	@	@
X	0.2 * 0.1 = 0.02	max(0.02 * 0.5 * 0.9, **0.25 * 0.2 * 0.9**, 0.21 * 0.1 * 0.9) = 0.045	max(**.045 * .5 * .9**, .0375 * .2 * .9, .0504 * .1 * .9) = **0.02025**
Y	0.5 * 0.5 = 0.25	max(0.02 * 0.4 * 0.5, **0.25 * 0.3 * 0.5,** 0.21 * 0.1 * 0.5) = 0.0375	max(**.045 * .4 * .5**, .0375 * .3 * .5, .0504 * .1 * .5) = 0.009
Z	0.3 * 0.7 = 0.21	max(0.02 * 0.1 * 0.3, 0.25 * 0.5 * 0.3, **0.21 * 0.8 * 0.3**) = 0.0504	Max(.045 * .1 * .3, .0375 * .5 * .3, **.0504 * .8 * .3**) = 0.012096

Backtrack to get optimal state sequence: Y X X

Q3: Unsupervised Training

Problem: No state sequences to compute above

Solution: Guess the state sequences

Initialize parameters randomly

Repeat for a while:

1. Predict the current state sequences using the current model
2. Update the current parameters based on current predictions

"Hard EM" or Viterbi EM

Initialize parameters randomly

Repeat for a while:

1. Predict the current state sequences using the current model with the Viterbi algorithm
2. Update the current parameters using the current predictions as in the supervised learning case

Can also use "soft" predictions; i.e., the probabilities of all the possible state sequences

Baum-Welch Algorithm

a.k.a., **Forward-backward** algorithm

EM algorithm applied to HMMs:

Expectation	Get *expected* counts for hidden structures using current θ^k.
Maximization	Find θ^{k+1} to maximize the likelihood of the training data given the expected counts from the E-step.

Responsibilities Needed

Supervised/Hard EM: A single tag

Baum-Welch: *Distribution* over tags

Call such probabilities **responsibilities**.

$\gamma_i(t) = P(Q_t = i | \boldsymbol{O}, \theta^k)$

- Probability of being in state i at time t given the observed sequence under the current model.

$\xi_{ij}(t) = P(Q_t = i, Q_{t+1} = j | \boldsymbol{O}, \theta^k)$

- Probability of transitioning from i at time t to j at time $t + 1$.

E-Step γ

$$\gamma_i(t) = P(Q_t = i|\boldsymbol{O}, \theta^k)$$

$$= \frac{P(Q_t = i, \boldsymbol{O}|\theta^k)}{P(\boldsymbol{O}|\theta^k)}$$

$$= \frac{\alpha_i(t)\beta_i(t)}{P(\boldsymbol{O}|\theta^k)}$$

E-Step ξ

$$\xi_{ij}(t) = P(Q_t = i, Q_{t+1} = j | \boldsymbol{O}, \theta^k)$$

$$= \frac{P(Q_t = i, Q_{t+1} = j, \boldsymbol{O} | \theta^k)}{P(\boldsymbol{O}|\theta^k)}$$

$$= \frac{\alpha_i(t) a_{ij} b_j(O_{t+1}) \beta_j(t+1)}{P(\boldsymbol{O}|\theta^k)}$$

Beginning to state i at time t

Transition from i to j

Emit O_{t+1}

Rest of the sequence

M-Step

"Soft" versions of the MLE updates:

$$\pi_i^{k+1} = \gamma_i(1)$$

$$a_{ij}^{k+1} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^{k+1}(w_k) = \frac{\sum_{t=1}^{T} \gamma_i(t)|_{O_t=w_k}}{\sum_{t=1}^{T} \gamma_i(t)}$$

Compare:

$$\frac{\#(i,j)}{\#(i)}$$

$$\frac{\#(\text{word } k, \text{tag } i)}{\#(i)}$$

- With multiple sentences, sum up expected counts over all sentences

When to Stop EM?

Multiple options

- When training set likelihood stops improving
- When prediction performance on held-out **development** or **validation** set stops improving

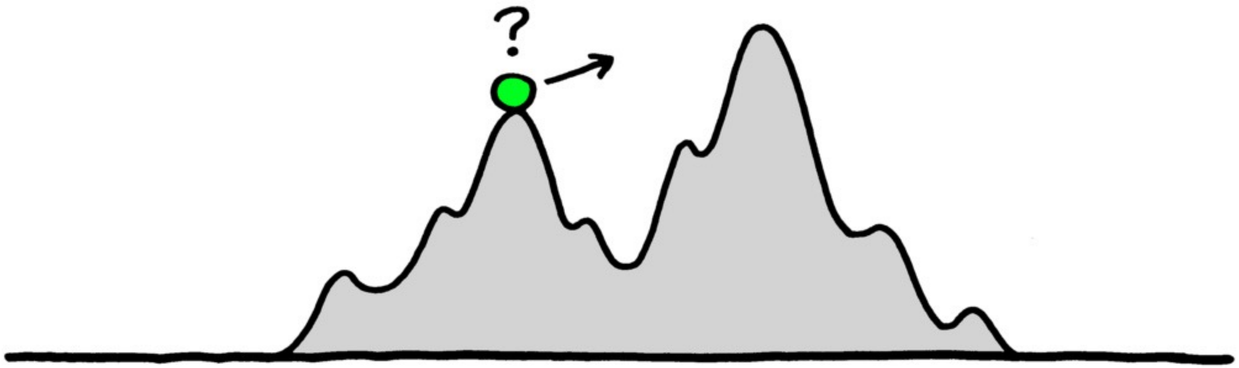
Why Does EM Work?

EM finds a local optimum in $P(\boldsymbol{O}|\theta)$.

You can show that after each step of EM:

$$P\left(\boldsymbol{O}\middle|\theta^{k+1}\right) > P(\boldsymbol{O}|\theta^{k})$$

However, this is not necessarily a global optimum.



Proof of Baum-Welch Correctness

Part 1: Show that in our procedure, one iteration corresponds to this update:

$$\theta^{k+1} = \underset{\theta}{\operatorname{argmax}} \sum_{\boldsymbol{Q}} \log[P(\boldsymbol{O}, \boldsymbol{Q}|\theta)] P(\boldsymbol{Q}|\boldsymbol{O}, \theta^k)$$

https://stephentu.github.io/writeups/hmm-baum-welch-derivation.pdf

Part 2: Show that improving the quantity

$\sum_{\boldsymbol{Q}} \log[P(\boldsymbol{O}, \boldsymbol{Q}|\theta)] P(\boldsymbol{Q}|\boldsymbol{O}, \theta^k)$

corresponds to improving $P(\boldsymbol{O}|\theta)$

https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm#Proof_of_correctness

Dealing with Local Optima

Random restarts

- Train multiple models with different random initializations
- Model selection on development set to pick the best one

Biased initialization

- Bias your initialization using some external source of knowledge (e.g., external corpus counts or clustering procedure, expert knowledge about domain)
- Further training will hopefully improve results

Caveats

Baum-Welch with no labelled data generally gives poor results, at least for linguistic structure (~40% accuracy, according to Johnson, (2007))

Semi-supervised learning: combine small amounts of labelled data with larger corpus of unlabelled data

In Practice

Per-token (i.e., per word) accuracy results on WSJ corpus:

Most frequent tag baseline	~90—94%
HMMs (Brants, 2000)	96.5%
Stanford tagger (Manning, 2011)	97.32%
Current best	97.85%+

- E.g. (Akbik et al., 2018), based on algorithms we will discuss in the next two classes

Other Sequence Modelling Tasks

Chunking (a.k.a., **shallow parsing**)

- Find syntactic chunks in the sentence; not hierarchical

[$_{NP}$The chicken] [$_{V}$crossed] [$_{NP}$the road] [$_{P}$across] [$_{NP}$the lake].

Named-Entity Recognition (**NER**)

- Identify elements in text that correspond to some high level categories (e.g., PERSON, ORGANIZATION, LOCATION)

[$_{ORG}$McGill University] is located in [$_{LOC}$Montreal, Canada].

- Problem: need to detect spans of multiple words

First Attempt

Simply label words by their category

ORG ORG ORG - ORG - - - LOC

McGill, UQAM, UdeM, and Concordia are located in Montreal.

What is the problem with this scheme?

IOB Tagging

Label whether a word is inside, outside, or at the beginning of a span as well.

For n categories, 2n+1 total tags.

B_{ORG} I_{ORG} O O O B_{LOC} I_{LOC}

McGill University is located in Montreal, Canada

B_{ORG} B_{ORG} B_{ORG} O B_{ORG} O O O B_{LOC}

McGill, UQAM, UdeM, and Concordia are located in Montreal.