4CSLL5
IBM Translation Models

Martin Emms

October 4, 2018

Brute force EM learning

Outline

Parameter learning (brute force)

- Introduction
- The brute force EM algorithm defined
- A formula for $p(a|\mathbf{o}, \mathbf{s})$
- Examples brute force EM in action

Learning Lexical Translation Models

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie.
  $(\mathbf{o}^1, a^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* . . .)

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie. $(\mathbf{o}^1, a^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* ...)
- but we don't ...

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie. $(\mathbf{o}^1, a^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* ...)
- but we don't ...
- **if we knew the parameters**, it would be (relatively) easy to calculate the 'odds' on alignments ie. $P(a^1|\mathbf{o}^1, \mathbf{s}^1) \ldots P(a^D|\mathbf{o}^D, \mathbf{s}^D)$

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \dots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie. $(\mathbf{o}^1, a^1, \mathbf{s}^1) \dots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* ...)
- but we don't ...
- **if we knew the parameters**, it would be (relatively) easy to calculate the 'odds' on alignments ie. $P(a^1|\mathbf{o}^1, \mathbf{s}^1) \dots P(a^D|\mathbf{o}^D, \mathbf{s}^D)$
- but we don't ...

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie. $(\mathbf{o}^1, a^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* . . .)
- but we don't . . .
- **if we knew the parameters**, it would be (relatively) easy to calculate the 'odds' on alignments ie. $P(a^1|\mathbf{o}^1, \mathbf{s}^1) \ldots P(a^D|\mathbf{o}^D, \mathbf{s}^D)$
- but we don't . . .
- something of a 'Chicken and Egg' situation

Learning Lexical Translation Models

- We would like to estimate the lexical translation probabilities $t(o|s)$ from a parallel corpus $(\mathbf{o}^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, \mathbf{s}^D)$
- this would be easy **if we had the alignments** ie. $(\mathbf{o}^1, a^1, \mathbf{s}^1) \ldots (\mathbf{o}^D, a^D, \mathbf{s}^D)$ (or just how *frequent* . . .)
- but we don't . . .
- **if we knew the parameters**, it would be (relatively) easy to calculate the 'odds' on alignments ie. $P(a^1|\mathbf{o}^1, \mathbf{s}^1) \ldots P(a^D|\mathbf{o}^D, \mathbf{s}^D)$
- but we don't . . .
- something of a 'Chicken and Egg' situation
- but the EM algorithm embraces this exactly

EM Algorithm roughly

Expectation Maximization (EM) in a nutshell

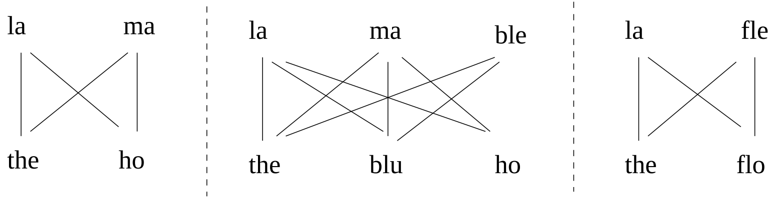
1. initialize model parameters (e.g. uniform)
2. assign probabilities to the missing data
3. treat probabilities like counts in complete data and estimate model parameters from the pseudo-completed data
4. iterate steps 2–3 until convergence

The EM algorithm keeps *re*-estimating the parameters. The following slides show in a graphical fashion the evolution of the parameters when the process is applied to the corpus

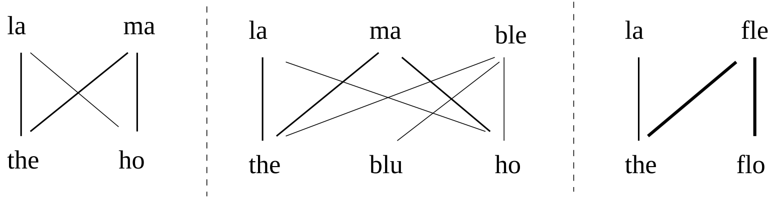
$\mathbf{s}^1$	la maison	$\mathbf{s}^2$	la maison bleu	$\mathbf{s}^3$	la fleur
$\mathbf{o}^1$	the house	$\mathbf{o}^2$	the blue house	$\mathbf{o}^3$	the flower

and with all $tr(o|s)$ values initially equal

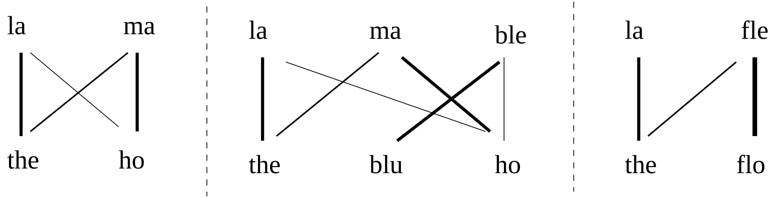
initial



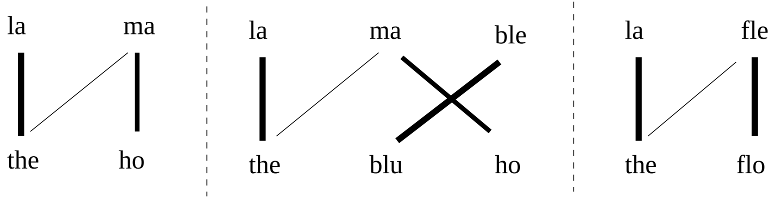
after one



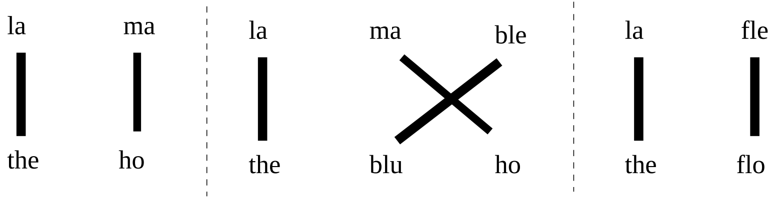
after two



after four



after ten



Outline

- to arrive at the EM algorithm for this case its a good idea to first spell out explicitly what the counting and parameter-estimation would look like if you *had* the alignments

- to arrive at the EM algorithm for this case its a good idea to first spell out explicitly what the counting and parameter-estimation would look like if you *had* the alignments
- then *migrate* that into the EM version replacing anything which assume a definite alignment with lines which consider all possible alignments, treating each has having a 'count' of $p(a|\mathbf{o}, \mathbf{s})$

- to arrive at the EM algorithm for this case its a good idea to first spell out explicitly what the counting and parameter-estimation would look like if you *had* the alignments
- then *migrate* that into the EM version replacing anything which assume a definite alignment with lines which consider all possible alignments, treating each has having a 'count' of $p(a|\mathbf{o}, \mathbf{s})$
- next 2 slides do exactly this

Estimating translation probs $tr(o|s)$ from complete data

Suppose you have a corpus of D pairs of sentence, and each has an alignment a. From this we can estimate the values of $tr(o|s)$ for the model in a straightforward way[1]

COUNT

[1] If we wanted to be really thorough we could set up the differential equations which define the parameters which will maximise the likelihood of the data under the model and show that solving them for $tr(o|s)$ parameters amounts to the counting procedure shown

Estimating translation probs $tr(o|s)$ from complete data

Suppose you have a corpus of D pairs of sentence, and each has an alignment a. From this we can estimate the values of $tr(o|s)$ for the model in a straightforward way[1]

$$
\begin{array}{l}
COUNT \\
for\ \ each\ \ o \in \mathcal{V}_o \\
\quad for\ \ each\ \ s \in \mathcal{V}_s \cup \{NULL\} \\
\quad\quad set\ \ \#(o,s)\ \ =\ \ 0
\end{array}
$$

[1] If we wanted to be really thorough we could set up the differential equations which define the parameters which will maximise the likelihood of the data under the model and show that solving them for $tr(o|s)$ parameters amounts to the counting procedure shown

Estimating translation probs $tr(o|s)$ from complete data

Suppose you have a corpus of D pairs of sentence, and each has an alignment a. From this we can estimate the values of $tr(o|s)$ for the model in a straightforward way[1]

$$
\begin{array}{l}
\textit{COUNT} \\
\textit{for each } o \in \mathcal{V}_o \\
\quad \textit{for each } s \in \mathcal{V}_s \cup \{NULL\} \\
\qquad \textit{set } \#(o,s) = 0
\end{array}
$$

$$
\begin{array}{ll}
\textit{for each aligned pair } (\mathbf{o}, a, \mathbf{s}) & \texttt{// just counting freqs of (o,s)} \\
\quad \textit{for each } j \in 1 : \ell_{\mathbf{o}} & \texttt{// word-pairs in the data} \\
\qquad \#(o_j, s_{a(j)}) \texttt{ += } 1 &
\end{array}
$$

[1] If we wanted to be really thorough we could set up the differential equations which define the parameters which will maximise the likelihood of the data under the model and show that solving them for $tr(o|s)$ parameters amounts to the counting procedure shown

Estimating translation probs $tr(o|s)$ from complete data

Suppose you have a corpus of D pairs of sentence, and each has an alignment a. From this we can estimate the values of $tr(o|s)$ for the model in a straightforward way[1]

COUNT
for each $o \in \mathcal{V}_o$
  for each $s \in \mathcal{V}_s \cup \{NULL\}$
    set $\#(o, s) \;=\; 0$

for each aligned pair $(\mathbf{o}, a, \mathbf{s})$ `// just counting freqs of (o,s)`
  for each $j \in \; 1 : \ell_{\mathbf{o}}$ `// word-pairs in the data`
    $\#(o_j, s_{a(j)})$ `+= 1`

TAKE RATIOS

[1] If we wanted to be really thorough we could set up the differential equations which define the parameters which will maximise the likelihood of the data under the model and show that solving them for $tr(o|s)$ parameters amounts to the counting procedure shown

Estimating translation probs $tr(o|s)$ from complete data

Suppose you have a corpus of D pairs of sentence, and each has an alignment a. From this we can estimate the values of $tr(o|s)$ for the model in a straightforward way[1]

COUNT
for each $o \in \mathcal{V}_o$
  for each $s \in \mathcal{V}_s \cup \{NULL\}$
    set $\#(o, s) = 0$

for each aligned pair $(\mathbf{o}, a, \mathbf{s})$ `// just counting freqs of (o,s)`
  for each $j \in 1 : \ell_{\mathbf{o}}$ `// word-pairs in the data`
    $\#(o_j, s_{a(j)})$ `+= 1`

TAKE RATIOS
for each $s \in \mathcal{V}_s \cup \{NULL\}$
  for each $o \in \mathcal{V}_o$

$$tr(o|s) = \frac{\#(o, s)}{\sum_o \#(o, s)}$$

[1] If we wanted to be really thorough we could set up the differential equations which define the parameters which will maximise the likelihood of the data under the model and show that solving them for $tr(o|s)$ parameters amounts to the counting procedure shown

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat $[E]$ *followed by* $[M]$ *till convergence*

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat $[E]$ *followed by* $[M]$ *till convergence*

`[E]`

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat [E] *followed by* [M] *till convergence*

`[E]`

$for \;\; each \;\; o \in \mathcal{V}_o$
$\quad for \;\; each \;\; s \in \mathcal{V}_s \cup \{NULL\}$
$\quad\quad set \;\; \#(o, s) \;\; = \;\; 0$

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat $[E]$ *followed by* $[M]$ *till convergence*

```
[E]
```

for each $o \in \mathcal{V}_o$
    for each $s \in \mathcal{V}_s \cup \{NULL\}$
        set $\#(o, s) \;=\; 0$

for each pair $(\mathbf{o}, \mathbf{s})$
        for each a calculate $p(a|\mathbf{o}, \mathbf{s})$ `// pseudo counts of (o,s) word pairs`
            for each $j \in \; 1 : \ell_{\mathbf{o}}$ `// in virtual data`
                $\#(o_j, s_{a(j)})$ `+=` $p(a|\mathbf{o}, \mathbf{s})$

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat $[E]$ *followed by* $[M]$ *till convergence*

```
[E]
```

$for\ \ each\ \ o \in \mathcal{V}_o$
$\quad for\ \ each\ \ s \in \mathcal{V}_s \cup \{NULL\}$
$\quad\quad set\ \ \#(o,s) \ = \ 0$

$for\ \ each\ \ pair\ \ (\mathbf{o}, \mathbf{s})$
$\quad\quad for\ \ each\ \ a\ \ calculate\ \ p(a|\mathbf{o}, \mathbf{s})$ `// pseudo counts of (o,s) word pairs`
$\quad\quad\quad for\ \ each\ \ j \in\ 1 : \ell_{\mathbf{o}}$ `// in virtual data`
$\quad\quad\quad\quad \#(o_j, s_{a(j)})$ `+=` $p(a|\mathbf{o}, \mathbf{s})$

```
[M]
```

outline of brute-force EM training for IBM model 1

initialise $tr(o|s)$ *uniformly*

repeat [*E*] *followed by* [*M*] *till convergence*

```
[E]
```

for each $o \in \mathcal{V}_o$
  for each $s \in \mathcal{V}_s \cup \{NULL\}$
    set $\#(o,s) = 0$

for each pair $(\mathbf{o}, \mathbf{s})$
    for each a *calculate* $p(a|\mathbf{o},\mathbf{s})$ `// pseudo counts of (o,s) word pairs`
      for each $j \in 1 : \ell_{\mathbf{o}}$ `// in virtual data`
        $\#(o_j, s_{a(j)})$ += $p(a|\mathbf{o},\mathbf{s})$

```
[M]
```

for each $s \in \mathcal{V}_s \cup \{NULL\}$
  for each $o \in \mathcal{V}_o$

$$tr(o|s) = \frac{\#(o,s)}{\sum_o \#(o,s)}$$

Outline

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o},\mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o},\mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o}, \mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o}, \mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.
By definition this is

$$\frac{p(\mathbf{o}, a, \mathbf{s})}{\sum_{a'} p(\mathbf{o}, a', \mathbf{s})} \tag{8}$$

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o}, \mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o}, \mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.
By definition this is

$$\frac{p(\mathbf{o}, a, \mathbf{s})}{\sum_{a'} p(\mathbf{o}, a', \mathbf{s})} \tag{8}$$

We have a formula for the combinations of $\langle \mathbf{o}, a, \mathbf{s} \rangle$, ie.

$$P(\mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s}) = p(\mathbf{s}) \times \frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}} + 1)^{\ell_{\mathbf{o}}}} \times \prod_j [p(o_j|s_{a(j)})]$$

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o}, \mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o}, \mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.
By definition this is

$$\frac{p(\mathbf{o}, a, \mathbf{s})}{\sum_{a'} p(\mathbf{o}, a', \mathbf{s})} \tag{8}$$

We have a formula for the combinations of $\langle \mathbf{o}, a, \mathbf{s} \rangle$, ie.

$$P(\mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s}) = p(\mathbf{s}) \times \frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}} + 1)^{\ell_{\mathbf{o}}}} \times \prod_j [p(o_j|s_{a(j)})]$$

and when plugged into the numerator and denominator in (8), the $p(\mathbf{s})$ and $\frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}}$ terms cancel,

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o},\mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o},\mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.
By definition this is

$$\frac{p(\mathbf{o},a,\mathbf{s})}{\sum_{a'} p(\mathbf{o},a',\mathbf{s})} \tag{8}$$

We have a formula for the combinations of $\langle \mathbf{o},a,\mathbf{s}\rangle$, ie.

$$P(\mathbf{o},a,\ell_{\mathbf{o}},\mathbf{s}) = p(\mathbf{s}) \times \frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}} \times \prod_j [p(o_j|s_{a(j)})]$$

and when plugged into the numerator and denominator in (8), the $p(\mathbf{s})$ and $\frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}}$ terms cancel, giving

$$p(a|\mathbf{o},\mathbf{s}) = \frac{\prod_j [p(o_j|s_{a(j)})]}{\sum_{a'} \prod_j [p(o_j|s_{a'(j)})]} \tag{9}$$

Brute force EM for IBM Model 1 contd: formula for $p(a|\mathbf{o},\mathbf{s})$

to implement this need to be able to calculate $p(a|\mathbf{o},\mathbf{s})$ for each possible alignment a between $\mathbf{o}$ and $\mathbf{s}$.
By definition this is

$$\frac{p(\mathbf{o},a,\mathbf{s})}{\sum_{a'} p(\mathbf{o},a',\mathbf{s})} \tag{8}$$

We have a formula for the combinations of $\langle \mathbf{o},a,\mathbf{s}\rangle$, ie.

$$P(\mathbf{o},a,\ell_{\mathbf{o}},\mathbf{s}) = p(\mathbf{s}) \times \frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}} \times \prod_j [p(o_j|s_{a(j)})]$$

and when plugged into the numerator and denominator in (8), the $p(\mathbf{s})$ and $\frac{p(\ell_{\mathbf{o}}|\ell_{\mathbf{s}})}{(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}}$ terms cancel, giving

$$p(a|\mathbf{o},\mathbf{s}) = \frac{\prod_j [p(o_j|s_{a(j)})]}{\sum_{a'} \prod_j [p(o_j|s_{a'(j)})]} \tag{9}$$

so we can deploy (9) for $p(a|\mathbf{o},\mathbf{s})$ in the brute-force EM algorithm, and thereby iteratively (re)-estimate the translation probabilities.

Outline

A brute force example

see *Labs/brute_force_ibm_model1_worked_eg.pdf* for detailed worked through of this assuming a corpus of 2 pairs

$\mathbf{s}^1$	la maison		$\mathbf{s}^2$	la fleur
$\mathbf{o}^1$	the house		$\mathbf{o}^2$	the flower

initialising all $tr(o|s)$ uniformly to $\frac{1}{3}$

note: to keep calcs. to manageable size makes slight simplification of not allowing any alignments from **o** to a NULL added to **s**: this does not affect the validity of the formula (9)

Evolution of the translation probabililities $tr(o|s)$

		$o|s$ at each iteration							
Obs	Src	0	1	2	3	4	5	...	final
the	la	0.33	0.5	0.6	0.69	0.77	0.84		1.00
house	la	0.33	0.25	0.2	0.15	0.11	0.081		0.00
flower	la	0.33	0.25	0.2	0.15	0.11	0.081		0.00
the	maison	0.33	0.5	0.43	0.36	0.3	0.24		0.00
house	maison	0.33	0.5	0.57	0.64	0.7	0.76		1.00
flower	maison	0.33	0.00	0.00	0.00	0.00	0.00		0.00
the	fleur	0.33	0.5	0.43	0.36	0.3	0.24		0.00
house	fleur	0.33	0.00	0.00	0.00	0.00	0.00		0.00
flower	fleur	0.33	0.5	0.57	0.64	0.7	0.76		1.00

Evolution of corpus-related statistics

- EM is guaranteed to increase the data probability – the probability with hidden variables summed out, which in full is

$$\prod_d \left[p(\mathbf{o}^d, \mathbf{s}^d)\right] = \prod_d \left[\sum_a \left[p(\mathbf{o}^d, a|\ell^d_{\mathbf{o}}, \mathbf{s})\right] \times p(\ell^d_{\mathbf{o}}|\ell^d_{\mathbf{s}}) \times p(\mathbf{s}^d)\right]$$

Evolution of corpus-related statistics

- EM is guaranteed to increase the data probability – the probability with hidden variables summed out, which in full is

$$\prod_d \left[p(\mathbf{o}^d, \mathbf{s}^d)\right] = \prod_d \left[\sum_a \left[p(\mathbf{o}^d, a|\ell_{\mathbf{o}}^d, \mathbf{s})\right] \times p(\ell_{\mathbf{o}}^d|\ell_{\mathbf{s}}^d) \times p(\mathbf{s}^d)\right]$$

- the length probability $p(\ell_{\mathbf{o}}^d|\ell_{\mathbf{s}}^d)$ and the source probability $p(\mathbf{s}^d)$ are not being updated in the algorithm, so its sufficient to track the product of the $\sum_a \left[p(\mathbf{o}^d, a|\ell_{\mathbf{o}}^d, \mathbf{s})\right]$ terms, which is

$$\prod_d \left[\sum_a \frac{1}{(\ell_{\mathbf{s}}^d + 1)^{\ell_{\mathbf{o}}^d}} \times \prod_j tr(o_j|s_{a(j)})\right] \tag{10}$$

Evolution of corpus-related statistics

- EM is guaranteed to increase the data probability – the probability with hidden variables summed out, which in full is

$$\prod_d \left[p(\mathbf{o}^d, \mathbf{s}^d)\right] = \prod_d \left[\sum_a \left[p(\mathbf{o}^d, a | \ell^d_{\mathbf{o}}, \mathbf{s})\right] \times p(\ell^d_{\mathbf{o}} | \ell^d_{\mathbf{s}}) \times p(\mathbf{s}^d)\right]$$

- the length probability $p(\ell^d_{\mathbf{o}} | \ell^d_{\mathbf{s}})$ and the source probability $p(\mathbf{s}^d)$ are not being updated in the algorithm, so its sufficient to track the product of the $\sum_a \left[p(\mathbf{o}^d, a | \ell^d_{\mathbf{o}}, \mathbf{s})\right]$ terms, which is

$$\prod_d \left[\sum_a \frac{1}{(\ell^d_{\mathbf{s}} + 1)^{\ell^d_{\mathbf{o}}}} \times \prod_j tr(o_j | s_{a(j)})\right] \quad (10)$$

- This quantity should monotonically increase over iterations.

Evolution of corpus-related statistics

- EM is guaranteed to increase the data probability – the probability with hidden variables summed out, which in full is

$$\prod_d \left[p(\mathbf{o}^d, \mathbf{s}^d)\right] = \prod_d \left[\sum_a \left[p(\mathbf{o}^d, a|\ell^d_{\mathbf{o}}, \mathbf{s})\right] \times p(\ell^d_{\mathbf{o}}|\ell^d_{\mathbf{s}}) \times p(\mathbf{s}^d)\right]$$

- the length probability $p(\ell^d_{\mathbf{o}}|\ell^d_{\mathbf{s}})$ and the source probability $p(\mathbf{s}^d)$ are not being updated in the algorithm, so its sufficient to track the product of the $\sum_a \left[p(\mathbf{o}^d, a|\ell^d_{\mathbf{o}}, \mathbf{s})\right]$ terms, which is

$$\prod_d \left[\sum_a \frac{1}{(\ell^d_{\mathbf{s}} + 1)^{\ell^d_{\mathbf{o}}}} \times \prod_j tr(o_j|s_{a(j)})\right] \tag{10}$$

- This quantity should monotonically increase over iterations.
- Practically speaking, the quantity in (10) though increasing will be minutely small, so that some alternatives are often used. If p is just the probability, alternatives often used are $log(p)$ – 'the log prob', $1/p$ – the 'perplexity', and $log(1/p)$ – 'the log perplexity'

Evolution of corpus-related statistics (contd)

	0	1	2	3	4	...	final
	$p(\mathbf{o}^d \vert \mathbf{s}^d)$ at each iteration for each d						
$p(\textit{the house} \vert \textit{la maison})$	0.11	0.19	0.2	0.21	0.22	...	0.25
$p(\textit{the flower} \vert \textit{la fleur})$	0.11	0.19	0.2	0.21	0.22	...	0.25
	corpus level stats at each iteration						
prob	0.012	0.035	0.039	0.044	0.048	...	0.0625
log prob	-6.3	-4.8	-4.7	-4.5	-4.4	...	-4
perp	81	28	25	23	21	...	16
log perp	6.3	4.8	4.7	4.5	4.4	...	4

- the values shown for $p(\mathbf{o}|\mathbf{s})$ are really values for $p(\mathbf{o}|\mathbf{s}, \ell_\mathbf{o})$. If ϵ were the value of $p(\ell_\mathbf{o}|\ell_\mathbf{s})$, then the true values of $p(\mathbf{o}|\mathbf{s})$ would be these multiplied by ϵ
- The values in the 'prob' row increase, as do the values in the 'log prob' row – they are always negative because the probabilities are always < 1.
- Correspondingly, the values in the 'perp' row always fall, as they are just the inverses of the probabilities. The values in the 'log perp' row also fall