4CSLL5
IBM Translation Models

Martin Emms

October 17, 2018

Parameter learning (efficient)

How to sum alignments efficiently

Efficient EM via $p((j,i) \in a|\mathbf{o},\mathbf{s})$

Avoiding Exponential Cost

Outline

Parameter learning (efficient)

- How to sum alignments efficiently
- Efficient EM via $p((j,i) \in a|\mathbf{o},\mathbf{s})$

but what about Exponential cost?

but what about Exponential cost?

- the learnability of translation probabilites in an unsupervised fashion from just a corpus of pairs is a remarkable thing

but what about Exponential cost?

- the learnability of translation probabilites in an unsupervised fashion from just a corpus of pairs is a remarkable thing
- however, as we have formulated it, each possible alignment has to be considered in turn, and each contributes increments to expected counts

but what about Exponential cost?

- the learnability of translation probabilites in an unsupervised fashion from just a corpus of pairs is a remarkable thing
- however, as we have formulated it, each possible alignment has to be considered in turn, and each contributes increments to expected counts
- it was already noted that the number of possible alignments is $(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}$ – ie. exponential in the length of $\mathbf{o}$. For $\ell_{\mathbf{s}}+1=\ell_{\mathbf{o}}=10$, this is 10^{10}, or 10,000 million

but what about Exponential cost?

- the learnability of translation probabilites in an unsupervised fashion from just a corpus of pairs is a remarkable thing
- however, as we have formulated it, each possible alignment has to be considered in turn, and each contributes increments to expected counts
- it was already noted that the number of possible alignments is $(\ell_{\mathbf{s}}+1)^{\ell_{\mathbf{o}}}$ – ie. exponential in the length of $\mathbf{o}$. For $\ell_{\mathbf{s}}+1=\ell_{\mathbf{o}}=10$, this is 10^{10}, or 10,000 million
- so unless a way can be found to make the EM process on this model much more efficient, its learnability in principle would just be an interesting curiosity

but what about Exponential cost?

- the learnability of translation probabilites in an unsupervised fashion from just a corpus of pairs is a remarkable thing
- however, as we have formulated it, each possible alignment has to be considered in turn, and each contributes increments to expected counts
- it was already noted that the number of possible alignments is $(\ell_{\mathbf{s}} + 1)^{\ell_{\mathbf{o}}}$ – ie. exponential in the length of **o**. For $\ell_{\mathbf{s}} + 1 = \ell_{\mathbf{o}} = 10$, this is 10^{10}, or 10,000 million
- so unless a way can be found to make the EM process on this model much more efficient, its learnability in principle would just be an interesting curiosity
- it turns out that by studying a little more closely the formula where alignments are summed over, and doing some conversions of 'sums-over-products' to 'products-over-sums', it is indeed possible to make the EM process on this model much more efficient.

Summing over alignments

[2] in the formula for $p(\langle \mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s} \rangle)$ everything except the translation probs is going to cancel out when you take ratios . . .

Summing over alignments

Looking at the brute-force EM algorithm, need to calculate $p(a|\mathbf{o}, \mathbf{s})$ – call this $\gamma_d(a)$.

[2] in the formula for $p(\langle \mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s} \rangle)$ everything except the translation probs is going to cancel out when you take ratios ...

Summing over alignments

Looking at the brute-force EM algorithm, need to calculate $p(a|\mathbf{o},\mathbf{s})$ – call this $\gamma_d(a)$. its fairly easy to see that this is[2]

$$\gamma_d(a) = \frac{\prod_j p(o_j|s_{a(j)})}{\sum_{a'} \prod_j p(o_j|s_{a'(j)})} \tag{11}$$

[2] in the formula for $p(\langle \mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s}\rangle)$ everything except the translation probs is going to cancel out when you take ratios . . .

Summing over alignments

Looking at the brute-force EM algorithm, need to calculate $p(a|\mathbf{o}, \mathbf{s})$ – call this $\gamma_d(a)$. its fairly easy to see that this is[2]

$$\gamma_d(a) = \frac{\prod_j p(o_j|s_{a(j)})}{\sum_{a'} \prod_j p(o_j|s_{a'(j)})} \tag{11}$$

- The numerator is a product.

[2] in the formula for $p(\langle \mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s} \rangle)$ everything except the translation probs is going to cancel out when you take ratios ...

Summing over alignments

Looking at the brute-force EM algorithm, need to calculate $p(a|\mathbf{o}, \mathbf{s})$ – call this $\gamma_d(a)$. its fairly easy to see that this is[2]

$$\gamma_d(a) = \frac{\prod_j p(o_j|s_{a(j)})}{\sum_{a'} \prod_j p(o_j|s_{a'(j)})} \tag{11}$$

- The numerator is a product.
- It turns out the denominator can also be turned into *product of sums*

[2] in the formula for $p(\langle \mathbf{o}, a, \ell_{\mathbf{o}}, \mathbf{s} \rangle)$ everything except the translation probs is going to cancel out when you take ratios ...

Summing over alignments contd

each j can be aligned to any i between 0 and I, hence

$$\sum_{a} \prod_{j} t(o_j|s_{a(j)}) = \sum_{a(1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} t(o_j|s_{a(j)})$$

$$=$$

$$=$$

$$=$$

Summing over alignments contd

each j can be aligned to any i between 0 and I, hence

$$\begin{aligned}
\sum_a \prod_j t(o_j|s_{a(j)}) &= \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} t(o_j|s_{a(j)}) \\
&= \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} [t(o_1|s_{a(1)}) \ldots t(o_J|s_{a(J)})] \\
&= \\
&=
\end{aligned}$$

Summing over alignments contd

each j can be aligned to any i between 0 and I, hence

$$\sum_a \prod_j t(o_j|s_{a(j)}) = \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} t(o_j|s_{a(j)})$$

$$= \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} [t(o_1|s_{a(1)}) \ldots t(o_J|s_{a(J)})]$$

each $\sum_{a(j)=0}^{I}()$ effects just one $t(o_j|s_{a(j)})$ term, and this means we can use a sum-of-products to product-of-sums conversion, hence

$$=$$

$$=$$

Summing over alignments contd

each j can be aligned to any i between 0 and I, hence

$$\sum_a \prod_j t(o_j|s_{a(j)}) = \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} t(o_j|s_{a(j)})$$

$$= \sum_{a(1)=0}^{I} \ldots \sum_{a(J)=0}^{I} [t(o_1|s_{a(1)}) \ldots t(o_J|s_{a(J)})]$$

each $\sum_{a(j)=0}^{I}()$ effects just one $t(o_j|s_{a(j)})$ term, and this means we can use a sum-of-products to product-of-sums conversion, hence

$$= \prod_{j=1}^{J} [\sum_{a(j)=0}^{I} t(o_j|s_{a(j)})]$$

$$=$$

Summing over alignments contd

each j can be aligned to any i between 0 and I, hence

$$\sum_a \prod_j t(o_j|s_{a(j)}) = \sum_{a(1)=0}^{I} \dots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} t(o_j|s_{a(j)})$$

$$= \sum_{a(1)=0}^{I} \dots \sum_{a(J)=0}^{I} [t(o_1|s_{a(1)}) \dots t(o_J|s_{a(J)})]$$

each $\sum_{a(j)=0}^{I}()$ effects just one $t(o_j|s_{a(j)})$ term, and this means we can use a sum-of-products to product-of-sums conversion, hence

$$= \prod_{j=1}^{J} [\sum_{a(j)=0}^{I} t(o_j|s_{a(j)})]$$

$$= \prod_{j=1}^{J} [\sum_{i=0}^{I} t(o_j|s_i)]$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.

for the case of **o s** having length 2 can relatively easily verify by brute force

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o s** having length 2 can relatively easily verify by brute force

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) =$$

$$= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) +$$

$$=$$

$$=$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o s** having length 2 can relatively easily verify by brute force

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) =$$

$$= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) +$$

$$=$$

$$=$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o** **s** having length 2 can relatively easily verify by brute force

$$\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) =$$

$$\begin{aligned} &= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + \\ &\quad t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) + \\ &\quad t(o_1|s_2)\ t(o_2|s_0) + t(o_1|s_2)\ t(o_2|s_1) + t(o_1|s_2)\ t(o_2|s_2) = \\ &= \\ &= \end{aligned}$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o s** having length 2 can relatively easily verify by brute force

$$
\begin{aligned}
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) = \\
&= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + \\
&\quad t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) + \\
&\quad t(o_1|s_2)\ t(o_2|s_0) + t(o_1|s_2)\ t(o_2|s_1) + t(o_1|s_2)\ t(o_2|s_2) = \\
&= t(o_1|s_0)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&=
\end{aligned}
$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o s** having length 2 can relatively easily verify by brute force

$$
\begin{aligned}
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) = \\
&= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + \\
&\quad t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) + \\
&\quad t(o_1|s_2)\ t(o_2|s_0) + t(o_1|s_2)\ t(o_2|s_1) + t(o_1|s_2)\ t(o_2|s_2) = \\
&= t(o_1|s_0)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&\quad t(o_1|s_1)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&=
\end{aligned}
$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.
for the case of **o s** having length 2 can relatively easily verify by brute force

$$
\begin{aligned}
&\sum_{a(1)=0}^{2}\sum_{a(2)=0}^{2}\prod_{j=1}^{2} t(o_j|s_{a(j)}) = \\
&\quad = t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + \\
&\quad\quad t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) + \\
&\quad\quad t(o_1|s_2)\ t(o_2|s_0) + t(o_1|s_2)\ t(o_2|s_1) + t(o_1|s_2)\ t(o_2|s_2) = \\
&\quad = t(o_1|s_0)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&\quad\quad t(o_1|s_1)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&\quad\quad t(o_1|s_2)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] = \\
&\quad =
\end{aligned}
$$

Pause: did you believe that?

the key step above was a conversion from a sum-of-products to a product-of-sums.

for the case of **o s** having length 2 can relatively easily verify by brute force

$$
\begin{aligned}
\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \prod_{j=1}^{2} t(o_j|s_{a(j)}) = \\
&= t(o_1|s_0)\ t(o_2|s_0) + t(o_1|s_0)\ t(o_2|s_1) + t(o_1|s_0)\ t(o_2|s_2) + \\
&\quad t(o_1|s_1)\ t(o_2|s_0) + t(o_1|s_1)\ t(o_2|s_1) + t(o_1|s_1)\ t(o_2|s_2) + \\
&\quad t(o_1|s_2)\ t(o_2|s_0) + t(o_1|s_2)\ t(o_2|s_1) + t(o_1|s_2)\ t(o_2|s_2) = \\
&= t(o_1|s_0)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&\quad t(o_1|s_1)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] + \\
&\quad t(o_1|s_2)[t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)] = \\
&= [t(o_1|s_0) + t(o_1|s_1) + t(o_1|s_2)][t(o_2|s_0) + t(o_2|s_1) + t(o_2|s_2)]
\end{aligned}
$$

Making $p(a|\mathbf{o},\mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

Making $p(a|\mathbf{o}, \mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

$$\gamma_d(a) = \frac{\prod_{j=1}^{J}[t(o_j|s_{a(j)})]}{\prod_{j=1}^{J}[\sum_{i=0}^{I} t(o_j|s_i)]}$$

$$=$$

Making $p(a|\mathbf{o},\mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

$$\gamma_d(a) = \frac{\prod_{j=1}^{J}[t(o_j|s_{a(j)})]}{\prod_{j=1}^{J}[\sum_{i=0}^{I} t(o_j|s_i)]}$$

and this is just one *big* product

$$=$$

Making $p(a|\mathbf{o},\mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

$$\gamma_d(a) = \frac{\prod_{j=1}^{J}[t(o_j|s_{a(j)})]}{\prod_{j=1}^{J}[\sum_{i=0}^{I} t(o_j|s_i)]}$$

and this is just one *big* product

$$= \prod_{j=1}^{J}\left[\frac{t(o_j|s_{a(j)})]}{\sum_{i=0}^{I} t(o_j|s_i)]}\right]$$

Making $p(a|\mathbf{o}, \mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

$$\gamma_d(a) = \frac{\prod_{j=1}^{J}[t(o_j|s_{a(j)})]}{\prod_{j=1}^{J}[\sum_{i=0}^{I} t(o_j|s_i)]}$$

and this is just one *big* product

$$= \prod_{j=1}^{J}\left[\frac{t(o_j|s_{a(j)})]}{\sum_{i=0}^{I} t(o_j|s_i)]}\right]$$

each term in this product can be seen as the probability of a particular alignment step (j, i), given $\mathbf{o}, \mathbf{s}$, and it makes sense for the overall alignment probability to be a product of the individual steps. If we use the notation $\gamma_d(j, i)$ for this probability of a single alignment step, we get

Making $p(a|\mathbf{o}, \mathbf{s})$ into a product

Armed with this, we can rewrite (11) the formula for $\gamma_d(a)$ as

$$\gamma_d(a) = \frac{\prod_{j=1}^{J}[t(o_j|s_{a(j)})]}{\prod_{j=1}^{J}[\sum_{i=0}^{I} t(o_j|s_i)]}$$

and this is just one *big* product

$$= \prod_{j=1}^{J}\left[\frac{t(o_j|s_{a(j)})]}{\sum_{i=0}^{I} t(o_j|s_i)]}\right]$$

each term in this product can be seen as the probability of a particular alignment step (j, i), given $\mathbf{o}, \mathbf{s}$, and it makes sense for the overall alignment probability to be a product of the individual steps. If we use the notation $\gamma_d(j, i)$ for this probability of a single alignment step, we get

$$\gamma_d(a) = \prod_{j=1}^{J}[\gamma_d(j, a(j))]$$

We have for $\gamma_d(j, i)$

$$\gamma_d(j, i) = \frac{t(o_j | s_i)}{\sum_{i'=0}^{l} t(o_j | s_{i'})} \tag{12}$$

We have for $\gamma_d(j, i)$

$$\gamma_d(j, i) = \frac{t(o_j|s_i)}{\sum_{i'=0}^{I} t(o_j|s_{i'})} \tag{12}$$

- crucially the cost of calculating $\gamma_d(j, i)$ is trivial – its linear in length of $\mathbf{s}$

We have for $\gamma_d(j, i)$

$$\gamma_d(j, i) = \frac{t(o_j|s_i)}{\sum_{i'=0}^{l} t(o_j|s_{i'})} \tag{12}$$

- crucially the cost of calculating $\gamma_d(j, i)$ is trivial – its linear in length of $\mathbf{s}$
- The efficient version of EM rests on seeing that once $p(j, i|\mathbf{o}, \mathbf{s})$ is worked out for each j, i, the desired *expected* (o, s) counts can be worked out from them

Outline

recall the [E] step of the brute-force algorithm (if $\mathbf{o}, \mathbf{s}$ are the d^{th} pair):

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`
            $\#(o_j, s_{a(j)})$ += $p(a|\mathbf{o}, \mathbf{s})$

[3] The notation $\sum_{a|(j,i)\in a}()$ means 'sum over only those a that have $(j, i) \in a$

recall the [E] step of the brute-force algorithm (if $\mathbf{o},\mathbf{s}$ are the d^{th} pair):

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`
      $\#(o_j, s_{a(j)})$ += $p(a|\mathbf{o},\mathbf{s})$

- consider a particular (j,i). As you go through all possible a for $\mathbf{o},\mathbf{s}$, each time the alignment a contains this pairing you make the increment $\gamma_d(a)$.

[3] The notation $\sum_{a|(j,i)\in a}()$ means 'sum over only those a that have $(j,i) \in a$

recall the [E] step of the brute-force algorithm (if $\mathbf{o},\mathbf{s}$ are the d^{th} pair):

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`
      $\#(o_j, s_{a(j)})$ += $p(a|\mathbf{o},\mathbf{s})$

- consider a particular (j,i). As you go through all possible a for $\mathbf{o},\mathbf{s}$, each time the alignment a contains this pairing you make the increment $\gamma_d(a)$.

  ...aim for an algorithm which works out quickly for each j,i what the **sum** of these increments will be, ie.[3]

$$\sum_{a|(j,i)\in a} \gamma_d(a) \tag{13}$$

[3] The notation $\sum_{a|(j,i)\in a}()$ means 'sum over only those a that have $(j,i) \in a$

Summing over the alignments gives just $\gamma_d(j, i)$

for **o** position j, **s** position i is fixed. For every other **o** position j', j' can be aligned to any i between 0 and I, hence

$$\sum_{a|(i,j)\in a} \gamma_d(a) =$$

$$=$$

$$= \gamma_d(j, i)$$

Summing over the alignments gives just $\gamma_d(j, i)$

for **o** position j, **s** position i is fixed. For every other **o** position j', j' can be aligned to any i between 0 and I, hence

$$\sum_{a|(i,j)\in a} \gamma_d(a) = \sum_{a(1)=0}^{I} \cdots \sum_{a(j-1)=0}^{I} \sum_{a(j+1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \left[\gamma_d(j,i) \prod_{j' \neq j} \gamma_d(j', a(j')) \right]$$

$$=$$

$$= \gamma_d(j,i)$$

Summing over the alignments gives just $\gamma_d(j, i)$

for **o** position j, **s** position i is fixed. For every other **o** position j', j' can be aligned to any i between 0 and I, hence

$$\sum_{a|(i,j)\in a} \gamma_d(a) = \sum_{a(1)=0}^{I} \cdots \sum_{a(j-1)=0}^{I} \sum_{a(j+1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \left[\gamma_d(j,i) \prod_{j' \neq j} \gamma_d(j', a(j')) \right]$$

we can pull out $\gamma_d(j, i)$ and again do a sum-of-products to product-of-sums conversion with the rest, hence

$$=$$

$$= \gamma_d(j, i)$$

Summing over the alignments gives just $\gamma_d(j, i)$

for **o** position j, **s** position i is fixed. For every other **o** position j', j' can be aligned to any i between 0 and I, hence

$$\sum_{a|(i,j)\in a} \gamma_d(a) = \sum_{a(1)=0}^{I} \cdots \sum_{a(j-1)=0}^{I} \sum_{a(j+1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \left[\gamma_d(j, i) \prod_{j' \neq j} \gamma_d(j', a(j')) \right]$$

we can pull out $\gamma_d(j, i)$ and again do a sum-of-products to product-of-sums conversion with the rest, hence

$$= \gamma_d(j, i) \prod_{j' \neq j} \left[\sum_{a(j')=0}^{I} \gamma_d(j', a(j')) \right]$$

$$= \gamma_d(j, i)$$

Summing over the alignments gives just $\gamma_d(j, i)$

for **o** position j, **s** position i is fixed. For every other **o** position j', j' can be aligned to any i between 0 and I, hence

$$\sum_{a|(i,j)\in a} \gamma_d(a) = \sum_{a(1)=0}^{I} \cdots \sum_{a(j-1)=0}^{I} \sum_{a(j+1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \left[\gamma_d(j, i) \prod_{j' \neq j} \gamma_d(j', a(j')) \right]$$

we can pull out $\gamma_d(j, i)$ and again do a sum-of-products to product-of-sums conversion with the rest, hence

$$= \gamma_d(j, i) \prod_{j' \neq j} \left[\sum_{a(j')=0}^{I} \gamma_d(j', a(j')) \right]$$

each sum runs over *every* possible alignment destination for j' and so each one **sums to one**, so you get just

$$= \gamma_d(j, i)$$

Efficient EM algorithm for IBM Model 1 training

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

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

[E]

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

$$
\begin{array}{l}
\textit{for each pair } \mathbf{o}, \mathbf{s} \\
\quad \textit{for each } j \in \; 1 : \ell_{\mathbf{o}} \\
\quad\quad \textit{for each } i \in \; 0 : \ell_{\mathbf{s}} \\
\quad\quad\quad \#(o_j, s_i) \mathrel{+}= p((j, i)|\mathbf{o}, \mathbf{s}) \quad (\textit{using } (12))
\end{array}
$$

[M]

$$
\begin{array}{l}
\textit{for each } s \in \mathcal{V}_s \cup \{NULL\} \\
\quad \textit{for each } o \in \mathcal{V}_o \\
\quad\quad tr(o|s) \; = \; \dfrac{\#(o, s)}{\sum_o \#(o, s)}
\end{array}
$$

Further details

from the above outline to real code is a fairly short distance

Further details

from the above outline to real code is a fairly short distance

1. the formula for $p((j, i)|\mathbf{o}, \mathbf{s})$ is $\frac{t(o_j|s_i)}{\sum_{i'=0}^{l} t(o_j|s_{i'})}$, and the denominator stays the same as i is varied in $p((i, j)|\mathbf{o}, \mathbf{s})$, so this denominator should be calculated once for each j

Further details

from the above outline to real code is a fairly short distance

1. the formula for $p((j, i)|\mathbf{o}, \mathbf{s})$ is $\frac{t(o_j|s_i)}{\sum_{i'=0}^{I} t(o_j|s_{i'})}$, and the denominator stays the same as i is varied in $p((i, j)|\mathbf{o}, \mathbf{s})$, so this denominator should be calculated once for each j
2. likewise in the M step, in $\frac{\#(o,s)}{\sum_{o'} \#(o', s)}$ the denominator stays the same as o is varied, so this denominator should be calculated once for each s

Example One

Assuming a corpus of 2 pairs:

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

initialising all $tr(o|s)$ to uniformly to $\frac{1}{3}$ the evolution of looks like this:

		$o\|s$ 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

Example Two (Koehn p92)

assuming a corpus of 3 pairs

$\mathbf{s}^1$	das Haus	$\mathbf{s}^2$	das Buch	$\mathbf{s}^3$	ein Buch
$\mathbf{o}^1$	the house	$\mathbf{o}^2$	the book	$\mathbf{o}^3$	a book

initialising all $t(o|s)$ uniformly to 0.25, evolution of $t(o|s)$ is

		$o\|s$ at each iteration					
Obs	Src	0	1	2	3	...	final
the	*das*	0.25	0.5	0.6364	0.7479	...	1
book	*das*	0.25	0.25	0.1818	0.1208	...	0
house	*das*	0.25	0.25	0.1818	0.1313	...	0
the	*buch*	0.25	0.25	0.1818	0.1208	...	0
book	*buch*	0.25	0.5	0.6364	0.7479	...	1
a	*buch*	0.25	0.25	0.1818	0.1313	...	0
book	*ein*	0.25	0.5	0.4286	0.3466	...	0
a	*ein*	0.25	0.5	0.5714	0.6534	...	1
the	*haus*	0.25	0.5	0.4286	0.3466	...	0
house	*haus*	0.25	0.5	0.5714	0.6534	...	1