1 Recap from previous lecture

- task: how to induce alignments that are not given in parallel data?
- concave (but not strictly concave) log-likelihood for IBM Model 1

$$\log \ell(\mathcal{D};t) = \log \prod_{\mathbf{e},\mathbf{f}\in\mathcal{D}} p(\mathbf{e}|\mathbf{f}) = \sum_{\mathbf{e},\mathbf{f}\in\mathcal{D}} \sum_{j=1}^{l_e} \log \sum_{i=0}^{l_f} t(e_j|f_i) + \text{const}$$

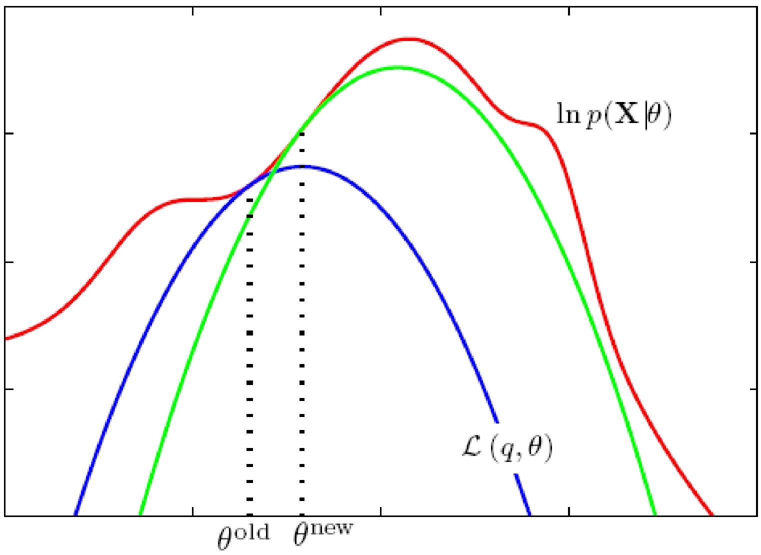
- note the marginalization $\sum_i$ over (unknown) alignments

EM (Expectation-Maximization) in a nutshell:
1. Initialize model parameters, e.g., uniform.
2. Assign probabilities to missing data. **E-step**
3. Estimate model parameters from manufactured/expected data. **M-step**
4. Iterate step 2 - 3 until convergence.

2 Expectation-Maximization

A general-purpose algorithm for unsupervised learning, not only for concave likelihoods or machine translation.

Oscillates between building a lower-bound to a function (E-step) and optimizing this lower-bound (M-step):



2.1 General Formulation

We have data $\mathcal{D} = \{x_i, i = 1 \dots m\}$, with unobserved (latent) variables $z_i \in \mathcal{Z}$ and want to optimize log-likelihood

$$\ell(\mathcal{D};\theta) = \sum_{i=1}^{m} \log \sum_{z \in \mathcal{Z}} p(x_i, z; \theta)$$

Let's assume we have some probability distributions $q_i(z_i)$. Using Jensen inequality and log concavity:

$$\begin{aligned}
\ell(\mathcal{D};\theta) &= \sum_{i=1}^{m} \log \sum_{z \in \mathcal{Z}} p(x_i, z_i; \theta) \\
&= \sum_{i=1}^{m} \log \sum_{z \in \mathcal{Z}} q_i(z_i) \frac{p(x_i, z_i; \theta)}{q_i(z_i)} \\
&\geq \sum_{i=1}^{m} \sum_{z \in \mathcal{Z}} q_i(z_i) \log \frac{p(x_i, z_i; \theta)}{q_i(z_i)}.
\end{aligned}$$

Equality is achieved (see exercise in the previous homework) when $\frac{p(x_i,z_i;\theta)}{q_i(z_i)}$ are constant. To achieve this set $q_i(z_i)$ proportional to $p(x_i, z_i; \theta)$ and normalize to make a valid probability distribution:

$$q_i(z_i) = \frac{p(x_i, z_i; \theta)}{\sum_{z \in \mathcal{Z}} p(x_i, z_i; \theta)} \qquad \textbf{E-step}$$

The above lower bound is true for all θ. To make the tighter, we maximize over θ, considering $q_i(z_i)$ fixed:

$$\theta = \arg\max_{\theta} \sum_{i=1}^{m} \sum_{z \in \mathcal{Z}} q_i(z_i) \log \frac{p(x_i, z; \theta)}{q_i(z_i)} \qquad \textbf{M-step}$$

Now alternate between **E-step** and **M-step**.

Why do we optimize log-likelihood in the process?

$$\begin{aligned}
\ell(\theta_{t+1}) &\geq \sum_{i=1}^{m} \sum_{z \in \mathcal{Z}} q_i^t(z_i) \log \frac{p(x_i, z; \theta_{t+1})}{q_i^t(z_i)} && \text{Jensen's inequality} \\
&\geq \sum_{i=1}^{m} \sum_{z \in \mathcal{Z}} q_i^t(z_i) \log \frac{p(x_i, z; \theta_t)}{q_i^t(z_i)} && \text{because } \theta^{t+1} \text{ is } \arg\max_{\theta} \\
&= \ell(\theta_t) && \text{because we chose } q_i(z_i) \text{ to deliver equality}
\end{aligned}$$

Note:

- in general parameters θ can be anything (e.g., a graph) so difficult to speak about convergence in parameters.
- in practice we stop when log-likelihood stops improving enough between iterations
- random initialization and other heuristic are useful (if your function is not concave)

3 IBM Model 1 and EM

3.1 Mapping general EM to SMT

$$
\begin{aligned}
x_i &\to (\mathbf{e}, \mathbf{f}) \\
z_i &\to a \\
q_i(z_i) &= p(a|\mathbf{e}, \mathbf{f}) \\
p(x_i, z_i; \theta) &= p(a, \mathbf{e}|\mathbf{f}; \theta)
\end{aligned}
$$

$$
\begin{array}{ccc}
\sum_{i=1}^{m} & \sum_{z \in \mathcal{Z}} q_i^t(z_i) & \log \frac{p(x_i, z; \theta_{t+1})}{q_i^t(z_i)} \\
\sum_{(\mathbf{e},\mathbf{f}) \in \mathcal{D}} & \sum_{a} p^t(a|\mathbf{e}, \mathbf{f}) & \log \frac{p(a, \mathbf{e}|\mathbf{f}; \theta)}{p^t(a|\mathbf{e}, \mathbf{f})}
\end{array}
$$

3.2 E(xpectation)-Step

Now we need to compute $p(a|\mathbf{e}, \mathbf{f})$, the probability of an alignment given the English and foreign sentences:

$$
p(a|\mathbf{e}, \mathbf{f}) = \frac{p(a, \mathbf{e}, \mathbf{f})}{p(\mathbf{e}, \mathbf{f})} = \frac{p(\mathbf{e}, a|\mathbf{f}) \cdot p(\mathbf{f})}{p(\mathbf{e}|\mathbf{f}) \cdot p(\mathbf{f})} = \frac{p(\mathbf{e}, a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} \tag{1}
$$

We are able to get $p(\mathbf{e}, a|\mathbf{f})$ from our IBM Model 1 equation, but we still need to compute $p(\mathbf{e}|\mathbf{f})$, the probability of translating the foreign sentence $\mathbf{f}$

into the English sentence $\mathbf{e}$ with any alignment.

$$p(\mathbf{e}|\mathbf{f}) = \sum_a p(\mathbf{e}, a|\mathbf{f}) \tag{2}$$

$$\text{(all alignment positions)} \quad = \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} p(\mathbf{e}, a|\mathbf{f}) \tag{3}$$

$$\text{(IBM Model)} \quad = \sum_{a(1)=0}^{l_f} \dots \sum_{a(l_e)=0}^{l_f} \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)}) \tag{4}$$

$$\text{("The Trick")} \quad = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i) \tag{5}$$

The trick in line (5) removes the need for an exponential number of products, which reduces the computational complexity significantly.

"The Trick":
Instead of summing over all possible alignments, we look at the English word positions and ask which foreign words could have generated them.

Example: $l_e = l_f = 2$

$$\begin{aligned} &\sum_{a(1)=0}^{2} \sum_{a(2)=0}^{2} \frac{\epsilon}{3^2} \prod_{j=1}^{2} t(e_j|f_{a(j)}) \\ =&\frac{\epsilon}{3^2}(t(e_1|f_0)t(e_2|f_0) + t(e_1|f_0)t(e_2|f_1) + t(e_1|f_0)t(e_2|f_2) \\ &+ t(e_1|f_1)t(e_2|f_0) + t(e_1|f_1)t(e_2|f_1) + t(e_1|f_1)t(e_2|f_2) \\ &+ t(e_1|f_2)t(e_2|f_0) + t(e_1|f_2)t(e_2|f_1) + t(e_1|f_2)t(e_2|f_2)) \end{aligned} \tag{6}$$

$$\begin{aligned} =&\frac{\epsilon}{3^2}(t(e_1|f_0)(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) \\ &+ t(e_1|f_1)(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) \\ &+ t(e_1|f_2)(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2))) \\ =&\frac{\epsilon}{3^2}(t(e_1|f_0) + t(e_1|f_1) + t(e_1|f_2))(t(e_2|f_0) + t(e_2|f_1) + t(e_2|f_2)) \\ =&\frac{\epsilon}{3^2} \prod_{j=1}^{2} \sum_{i=0}^{2} t(e_j|f_i) \end{aligned}$$

$$(\text{"The Trick"}) \qquad p(\mathbf{e}|\mathbf{f}) = \frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i) \tag{7}$$

Note: Doing the "trick" can be avoided by remembering that the generative story of the IBM Model 1 assumes independent word generation process:

$$p(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{l_e} p(e_i|\mathbf{f}). \tag{8}$$

Combine what we have:
We know $p(\mathbf{e}, a|\mathbf{f})$ from the IBM Model 1 equation (Lecture 1) and $p(\mathbf{e}|\mathbf{f})$ from the simplified equation in line (7).

$$\begin{aligned} p(a|\mathbf{e},\mathbf{f}) &= \frac{p(\mathbf{e},a|\mathbf{f})}{p(\mathbf{e}|\mathbf{f})} \qquad (9)\\ &= \frac{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} t(e_j|f_{a(j)})}{\frac{\epsilon}{(l_f+1)^{l_e}} \prod_{j=1}^{l_e} \sum_{i=0}^{l_f} t(e_j|f_i)} \\ &= \prod_{j=1}^{l_e} \frac{t(e_j|f_{a(j)})}{\sum_{i=0}^{l_f} t(e_j|f_i)} \end{aligned}$$

3.3 M(aximization)-Step

Now we need to maximize the lower-bound:

$$\arg\max_\theta \sum_{(\mathbf{e},\mathbf{f})\in\mathcal{D}} \sum_a p^t(a|\mathbf{e},\mathbf{f}) \log \frac{p(a,\mathbf{e}|\mathbf{f};\theta)}{p^t(a|\mathbf{e},\mathbf{f})},$$

where $p^t(a|\mathbf{e},\mathbf{f})$ are fixed by the preceding E-step. Substituting the expression for $p(\mathbf{e},a|\mathbf{f};\theta)$ from the IBM Model 1 and dropping constant (with respect to θ) terms we get a maximization problem:

$$\begin{aligned} &\arg\max_\theta \sum_{\mathbf{e},\mathbf{f}\in\mathcal{D}} \sum_{j=1}^{l_e} \big(\sum_a p^t(a|\mathbf{e},\mathbf{f}) \log t(e_j|f_{a(j)}) \big) \\ =&\arg\max_\theta \sum_{\mathbf{e},\mathbf{f}\in\mathcal{D}} \sum_{j=1}^{l_e} \big(\sum_{i=1}^{l_f} p^t(a|\mathbf{e},\mathbf{f}) \log t(e_j|f_i) \big) \qquad (10) \end{aligned}$$

The optimal $t(e|f)$ can be found by exactly the same method of Lagrangian multipliers from Lecture 1 (with probability constraint on $t(e|f)$). In the following we state only the result of the calculation.

Define a count function c, that collects evidence from a sentence pair $\mathbf{e}, \mathbf{f}$ that word e is translation of word f:

$$c(e|f; \mathbf{e}, \mathbf{f}) = \sum_{a} p(a|\mathbf{e}, \mathbf{f}) \sum_{j=1}^{l_e} \delta(e, e_j)\delta(f, f_{a(j)}) \tag{11}$$
$$\text{where } \delta(x, x') = 1 \text{ if } x = x', 0 \text{ else}$$

The Trick:
With the same simplification as before:

$$c(e|f; \mathbf{e}, \mathbf{f}) = \frac{t(e|f)}{\sum_{i=0}^{l_f} t(e|f_i)} \sum_{j=1}^{l_e} \delta(e, e_j) \sum_{i=0}^{l_f} \delta(f, f_i) \tag{12}$$

In result, we obtain the relative frequencies (the Maximum Likelihood Estimates) from the alignments weighted by $p(a|\mathbf{e}, \mathbf{f})$ from the E-Step.

Final estimate:

$$t(e|f) = \frac{\sum_{(\mathbf{e},\mathbf{f})} c(e|f; \mathbf{e}, \mathbf{f})}{\sum_{e} \sum_{(\mathbf{e},\mathbf{f})} c(e|f; \mathbf{e}, \mathbf{f})} \tag{13}$$

3.3.1 Pseudocode

Require: set of sentence pairs $(\mathbf{e}, \mathbf{f})$
Ensure: translation prob. $t(e|f)$
initialize $t(e|f)$ uniformly
while not converged **do**
  {initialize}
  count$(e|f) = 0$ **for all** e, f
  total$(f) = 0$ **for all** f
  for all sentence pairs $(\mathbf{e},\mathbf{f})$ **do**
    {compute normalization}
    for all words e in $\mathbf{e}$ **do**
      s-total$(e) = 0$
      for all words f in $\mathbf{f}$ **do**
        s-total(e) += $t(e|f)$

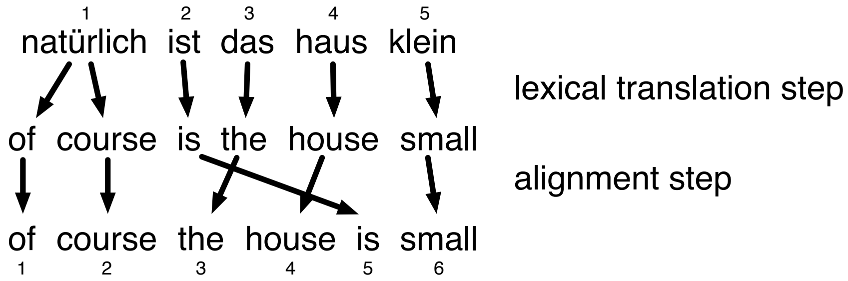
end for
end for
{collect counts}
for all words e in **e do**
for all words f in **f do**
count$(e|f)$ += $\frac{t(e|f)}{\text{s-total}(e)}$
total(f) += $\frac{t(e|f)}{\text{s-total}(e)}$
end for
end for
end for
{estimate probabilities}
for all foreign words f **do**
for all English words e **do**
$t(e|f) = \frac{\text{count}(e|f)}{\text{total}(f)}$
end for
end for
end while

3.4 IBM Model 2

Higher IBM Models

IBM Model 1:	lexical translation, all reorderings / alignments are equally likely
IBM Model 2:	adds explicit reordering / alignment model
IBM Model 3:	adds fertility model
IBM Model 4:	adds improved reordering model
IBM Model 5:	fixes deficiency

IBM Model 2 is a two-step model: The lexical translation model is similar to IBM Model 1, but it is also adding an explicit alignment probability distribution $a(i|j, l_e, l_f)$, which predicts a foreign input at position i given an English output at position j. (N.B. alignment probability distribution is defined in same direction as alignment function.)



IBM Model 2:

$$p(\mathbf{e}, a|\mathbf{f}) = \epsilon \prod_{j=1}^{l_e} t(e_j|f_{a(j)})a(i|j, l_e, l_f) \tag{14}$$

The estimation of probabilities for IBM Model 2 uses formulae similar to IBM Model 1.

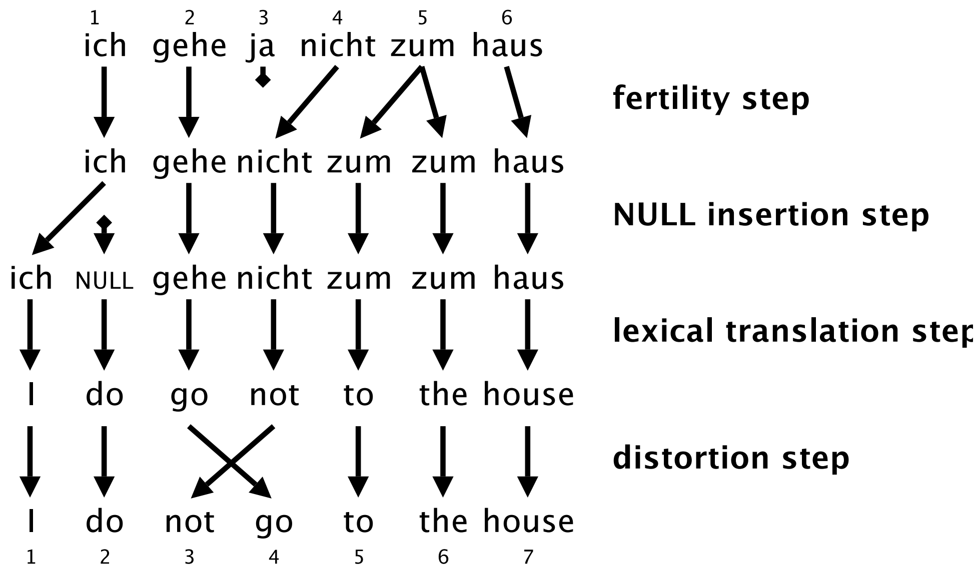
3.5 IBM Model 3

IBM Model 3 adds a **model of fertility**:
Fertility describes the number of English words generated by foreign words, modelled by the probability distribution $n(\phi|f)$ for $\phi = 0, 1, 2, \ldots$
Example:

$$\begin{aligned} n(0|\text{ja}) &\simeq 1 \\ n(2|\text{zum}) &\simeq 1 \\ n(1|\text{Haus}) &\simeq 1 \end{aligned}$$

Instead of the alignment model, we now use a **distortion model:**
$d(j|a(j), l_e, l_f)$ predicts an output position j given an input position $a(j)$.

NULL token insertion:
Instead of modelling the fertility of the NULL token in the same way as for all the other words, we model it as a special step: after the fertility step, we insert a NULL token after each word with probability p_1 or no NULL token with probability $1 - p_1$. (N.B. distortion is set up in translation direction, not alignment direction.)



3.5.1 Estimation of parameters for model 3

"The Trick" does not work in complex models like IBM Model 3.

Solution 1: EM algorithm for exponentially many possibilities.

Solution 2: Sampling most probable solutions by hill-climbing:

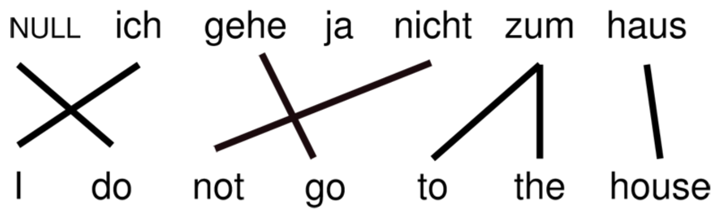
1. Start with initial alignment.
2. Change alignments by moving or swapping one word.
3. Keep change if it has higher probability.
4. Continue until convergence.

3.6 IBM Model 4

IBM Model 4 adds a **relative reordering model**:
The idea is, that words do not move independently, but in groups. We place the English translation of a foreign input word relative to the translation of the previously translated input word.

Relative reordering is defined with respect to **cepts**:

Cepts are formed by foreign words with non-zero fertility. The center $\odot_i$ of a cept π_i is the ceiling of avg(j). $\pi_{i,k}$ is the position of the k^{th} word in the i^{th} cept.



cept π_i	π_1	π_2	π_3	π_4	π_5
foreign position $[i]$	1	2	4	5	6
foreign word $f_{[i]}$	ich	gehe	nicht	zum	haus
English words $\{e_j\}$	I	go	not	to,the	house
English positions $\{j\}$	1	4	3	5,6	7
center of cept $\odot_i$	1	4	3	6	7

j	1	2	3	4	5	6	7
e_j	I	do	not	go	to	the	house
in cept $\pi_{i,k}$	$\pi_{1,0}$	$\pi_{0,0}$	$\pi_{3,0}$	$\pi_{2,0}$	$\pi_{4,0}$	$\pi_{4,1}$	$\pi_{5,0}$
$\odot_{i-1}$	0	-	4	1	3	-	6
$j - \odot_{i-1}$	$+1$	-	-1	$+3$	$+2$	-	$+1$
distortion	$d_1(+1)$	1	$d_1(-1)$	$d_1(+3)$	$d_1(+2)$	$d_{>1}(+1)$	$d_1(+1)$

3 cases of relative distortion:

1. Uniform for all NULL generated words.
2. First word of a cept: $d_1(j - \odot_{i-1})$ is the distortion of an English word position j relative to the center of the preceding cept $\odot_{i-1}$.

   Example:
   not: $d_1 = -1$
   $d_1(j - \odot_{i-1}) = d_1(j - \odot_2) = d_1(3 - 4) = -1$

3. Next words in a cept: $d_{>1}(j - \pi_{i,k-1})$ is the distortion of an English position j relative to the previous word in the cept.

   Example:
   the: $d_{>1}(j - \pi_{i,k-1}) = d_{>1}(6 - \pi_{4,0}) = d_{>1}(6 - 5) = 1$

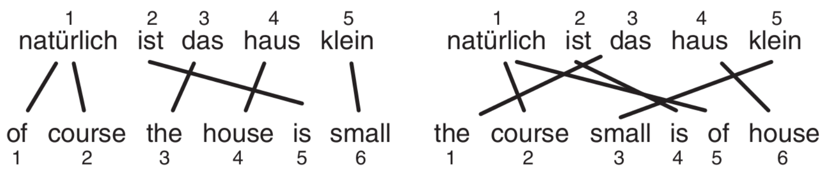
Training for IBM Model 4: some hill-climbing heuristic as for Model 3.

3.7 IBM Model 5

IBM Model 5 fixes the deficiency in IBM Models. In IBM Models 3 and 4, multiple outcome words can be placed in the same position, thus there is loss of probability mass and the probability model is deficient.

3.8 Efficient reparametrization of IBM Model 1 & 2 (`fast_align`)

- IBM Model 1 is too simplistic (all alignment are equally likely)



- IBM Model 2 is over parameterized (a separate alignment probability $a(j|i)$ for every combination of input-output position)
- both, however, support inference in roughly quadratic time in the length of the sentence

Observe $(\mathbf{e}, \mathbf{f})$ if lengths m and n. Introduce a distance function between source and target word positions: $h(i, j) = -\left|\frac{i}{m} - \frac{j}{n}\right|$.

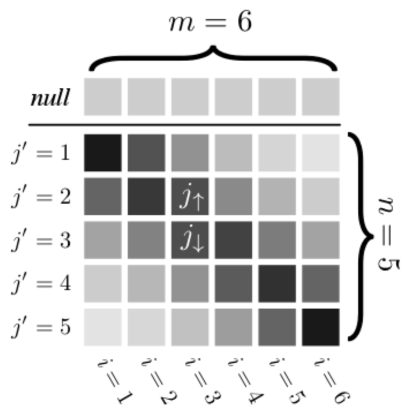
`fast_align` follows a similar generative story as IBM Model 1:

1. for each $i = 1, \ldots, l_e$ decide if it is a NULL word with probability p_0
2. if not, chose a value for a_i from $j = 0, \ldots, n$ according to log-linear distribution $\frac{e^{\lambda h(i,j)}}{Z_\lambda(i)}$
3. for each $j = 1, \ldots, l_e$ choose a output word e_j according to $t(e_j|f_{a(j)})$

Compared to IBM Model 1, `fast_align` has two additional parameters to learn, p_0 and λ, used in the probability of source position j being aligned with target position i:

$$a(j|i) = \begin{cases} p_0, & j = 0 \\ (1 - p_0) \times \frac{e^{\lambda h(i,j)}}{Z_\lambda(i)}, & 0 < j \leq n \\ 0, & \text{otherwise} \end{cases}$$

In the limiting case $\lambda \to 0$ the distribution approaches the uniform distribution of Model 1. For other of values the probability assigns higher probability mass to alignments close to diagonal.



Z_λ **calculation**:

$$Z_\lambda = \sum_{j=1}^{n} \exp(\lambda h(i,j))$$

Denote the closest cell on or above diagonal as $j_\uparrow$, and the next cell down as $j_\downarrow$:

$$j_\uparrow = \lfloor \frac{i \times n}{m} \rfloor \qquad j_\downarrow = j_\uparrow + 1$$

Starting at $j_\uparrow$ and moving up the alignment column, as well as starting at $j_\downarrow$ and moving down, the unnormalized probabilities decrease by a factor of $r = \exp(-\lambda n)$ per step.

Therefore, denoting the sum of geometric progression with multiplier r and starting element g as $\sigma(g, r)$ we get:

$$Z_\lambda = \sigma_{j_\uparrow}(\exp(\lambda h(i, j_\uparrow)), r) + \sigma_{n-j_\downarrow}(\exp(\lambda h(i, j_\downarrow)), r)$$

The probabilities needed for the **E-step**:

$$p(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{m} p(e_i|\mathbf{f}) = \prod_{i=1}^{m} \sum_{j=0}^{n} a(j|i) t(e_i|f_j), p(a|\mathbf{e}, \mathbf{f}) = \frac{p(a, \mathbf{e}, \mathbf{f})}{p(\mathbf{e}, \mathbf{f})}.$$

M-step requires, again, aggregating and counting counts as in (13). During the M-step, the λ parameter must also be updated to make the E-step posterior distribution over alignment points maximally probable under $a(j|i)$. This maximizing value cannot be computed analytically, but a gradient-based optimization can be used. (exercise to derive a gradient).

3.9 Conclusion

The IBM Models are still in use for word alignment in state-of-the-art SMT. Efficient reparametrization is often helpful in practice.

Important concepts:
- alignment
- EM training
- reordering models