Foundations of Natural Language Processing
Lecture 8c
Expectation Maximisation

Alex Lascarides

School of informatics

Recap: Noise model and minimum edit distance

- If you have costs for
  - deletion, insertion, substitution

  you can use dynamic programming to compute the minimum edit distance between two arbitrary strings.
  - no much effert $\mapsto$ not much effort
- Computing MED is tractable and fast.
- But how can you learn the costs for the operations without annotated data?
- The answer should *not* rely on access to vast amounts of character aligned data!

Now: Expectation Maximisation (EM)

General formulation

This sort of problem actually happens a lot in NLP (and ML):

- We have some probabilistic model and want to estimate its **parameters** (here, the character rewrite probabilities: prob of each typed character given each intended character).
- The model also contains variables whose value is unknown (here: the correct character alignments).
- We would be able to estimate the parameters if we knew the values of the variables...
- ...and conversely, we would be able to infer the values of the variables if we knew the values of the parameters.

EM to the rescue

Problems of this type can often be solved using a version of **Expectation-Maximisation** (EM), a general algorithm schema:

1. Initialize parameters to arbitrary values (e.g., set all costs $= 1$).

2. Using these parameters, compute optimal values for variables (run MED to get alignments).

3. Now, using those alignments, **recompute** the parameters (just pretend the alignments are hand annotations; estimate parameters as from annotated corpus).

4. Repeat steps 2 and 3 until parameters stop changing.

EM vs. hard EM

- The algorithm on the previous slide is actually "hard EM" (meaning: no soft/fuzzy decisions)
- Step 2 of true EM does not choose **optimal** values for variables, instead computes **expected** values (we'll see this for HMMs).
- True EM is guaranteed to converge to a local optimum of the **likelihood function**.
- Hard EM also converges but not to anything nicely defined mathematically. However it's usually easier to compute and may work fine in practice.

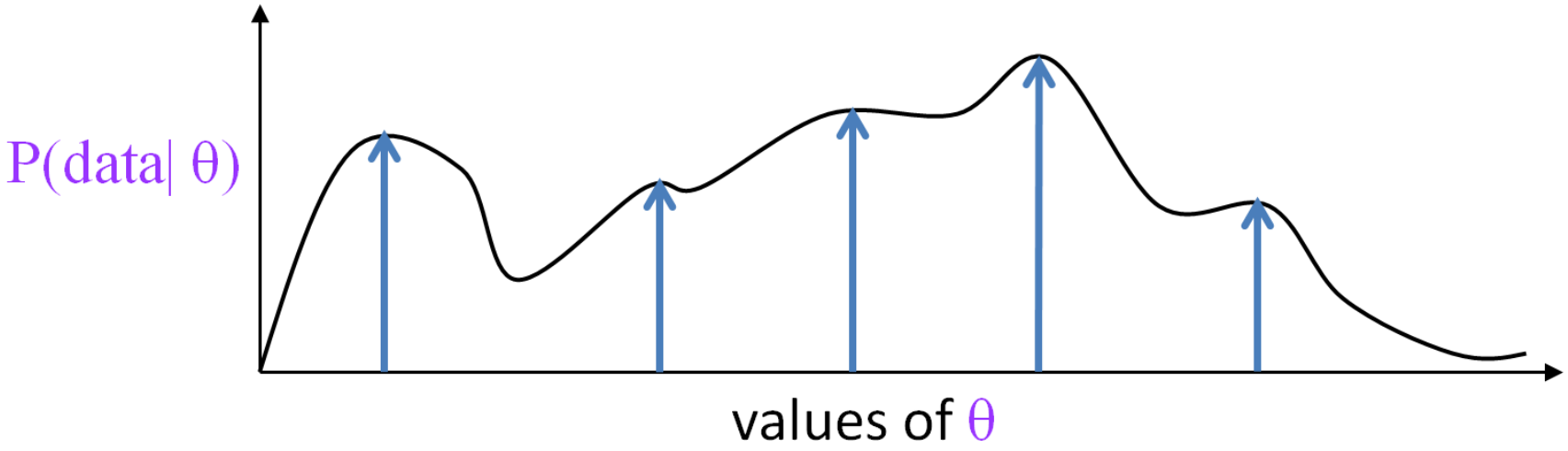
Likelihood function

- Let's call the parameters of our model θ.
  - So for our spelling error model, θ is the set of all character rewrite probabilities $P(x_i|y_i)$.

- For any value of θ, we can compute the probability of our dataset $P(\text{data}|\theta)$. This is the **likelihood**.
  - If our data includes hand-annotated character alignments, then $P(\text{data}|\theta) = \prod_{i=1}^{n} P(x_i|y_i)$
  - If the alignments a are latent, sum over possible alignments: $P(\text{data}|\theta) = \sum_a \prod_{i=1}^{n} P(x_i|y_i, a)$

Likelihood function

- The likelihood $P(\text{data}|\theta)$ is a function of θ, and can have multiple local optima. Schematically (but θ is really multidimensional):



- EM will converge to one of these; hard EM won't necessarily.
- Neither is guarateed to find the global optimum!

Summary

Our simple spelling corrector illustrated several important concepts:

- Example of a noise model in a noisy channel model.
- Difference between model definition and algorithm to perform inference.
- Confusion matrix: used here to estimate parameters of noise model, but can also be used as a form of error analysis.
- Minimum edit distance algorithm as an example of dynamic programming.
- (Hard) EM as a way to "bootstrap" better parameter values when we don't have enough annotated data.