Unsupervised Language Learning

Tutorial I

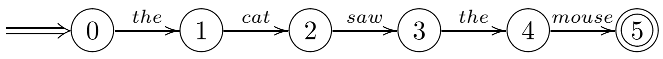
3-2-2011

1 Inducing Finite-state Automata

Consider the following (slightly odd but grammatical) example sentences (from Langley & Stromsten, 2000):

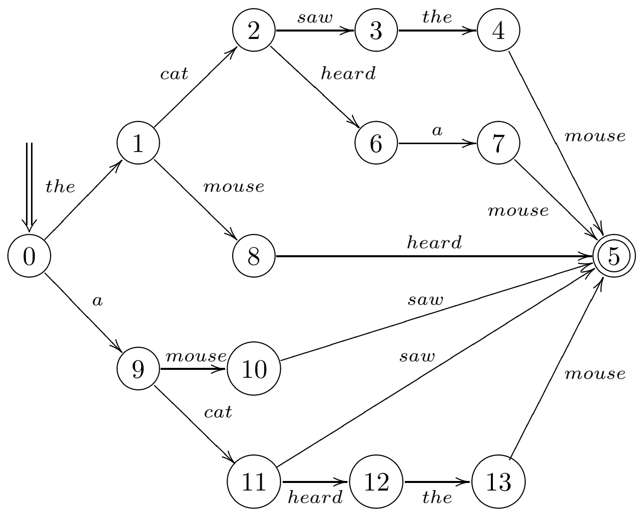
(1) a. The cat saw the mouse.
b. The cat heard a mouse.
c. The mouse heard.
d. A mouse saw.
e. A cat saw.
f. A cat heard the mouse.

There are different ways of representing these sentences in a finite-state automaton, but one way is as follows. We assume a finite set of states (the nodes labeled with numbers) and transitions between these states (the arrows labeled with words). There are two special states: a start state (represented with incoming double arrow) and an end state (represented with a double circle). We can then represent each of the sentences above with an automaton like the following:



(2)

Some of the sentences start with the same words. We can represent them with the same transitions, as in the following automaton that models all 6 sentences:



(3)

Question 1 *Which states does the automaton visit when generating sentence 1b?*

An simple approach to learning finite state automata from example sentences, is based on *incorporating* the sentences in a automaton as above, and then proceeds by *merging* states. In merging state A and B, a new state A' is created with all incoming and outgoing arrows of A and B.

Question 2 *Which states in the automaton above can be merged without changing the sentences it can generate? What is an example of two states that cannot be merged without that effect?*

Two possible heuristics for applying the merge operation, without the guarantee that the set of generatable sentences remains the same, are:

1. If A, B and C are states, and there are arrows $A \rightarrow^x B$ and $C \rightarrow^x B$, both labeled with word x (possibly in addition to other arrows from A or C) then merge A and C.
2. If A, B, C and D are states, and there are paths from $A \rightarrow B \rightarrow D$, and $A \rightarrow C \rightarrow D$, and B and C have one or more equal labels on the incoming or outgoing arrows, then merge B and C.

Question 3 *Which further merges can you perform using these heuristics? Which new sentences can the automaton now generate?*

The finite-state automaton that generates all grammatical sentences with the words in example 1, is the following:

$$\begin{tikzcd}
{} \arrow[r, Rightarrow] & 0 \arrow[r, bend left, "the"] \arrow[r, bend right, "a"'] & 1 \arrow[r, bend left, "cat"] \arrow[r, bend right, "mouse"'] & 2 \arrow[r, bend left, "saw"] \arrow[r, bend right, "heard"'] \arrow[rrr, bend left=50, "saw"] \arrow[rrr, bend right=50, "heard"'] & 3 \arrow[r, bend left, "the"] \arrow[r, bend right, "a"'] & 4 \arrow[r, bend left, "mouse"] \arrow[r, bend right, "cat"'] & \boxed{5}
\end{tikzcd} \tag{4}$$

We can probabilise the automaton (and make it into a *Markov model*) by, for instance, assuming that at every state, one of the possible arrows is chosen with equal probability.

Question 4 *With this assumption, what is, for each of the sentences in 1, the probability that it is generated by the automaton above?*

The product of probabilities of the sentences in 1 is called the *likelihood* of the data: $P(D|G)$, where D is the observed data and G is the induced grammar (automaton) that is assumed to have generated the data.

Question 5 *Does automaton (3) or (4) assign higher likelihood to the data in (1)?*

In *Bayesian inference*, the *a-priori probability* $P(G)$ of the grammar – the probability you assign to a grammar before having seen any data – plays an important role as well. Often, the prior probability of a grammar G is assumed to be exponentially decreasing with the size of G: $P(G) \propto \mathrm{e}^{-\mathrm{size}(G)}$ (where the size is counted as, for instance, the total number of (terminal or nonterminal) symbols in rewrite rules that characterise all transitions, i.e. $0 \rightarrow^{the} 1$ adds 3 to the total).

Question 6 *Which automaton, (3) or (4), receives a higher prior probability according to such a* description length prior*?*

The goal of Bayesian inference is to find the grammar that maximizes the posterior probability $P(G|D)$. Using Bayes' rule, we find:

$$\underset{G}{\operatorname{argmax}} P(G|D) = \underset{G}{\operatorname{argmax}} P(D|G)P(G). \tag{5}$$

Stolcke (1994) presents an approach to inducing (hidden) Markov Models from example sentences, using the incorporation and merge operation we looked at above, and merging those states that maximize the posterior probability. Details of this work go beyond the scope of this exercise.

2 Inducing Context-free Grammars

We can generalize the state merging procedure for learning finite-state machines, to context-free grammars. Cook *et al.* (1976) proposes the following operations:

incorporate($\alpha\beta \dots \omega$)**:** add the following rules to the grammar:

- $A \rightarrow \alpha$
- $B \rightarrow \beta$
- ...
- $W \rightarrow \omega$; and a rule
- $S \rightarrow A\ B\ \dots\ W$.

merge(X, Y)**:** replace all occurrences of Y in the grammar with X.

chunk(X, Y)**:** replace all occurrences of $X\ Y$ in the grammar with Z, and add a rule:

- $Z \rightarrow X\ Y$.

It can be shown that these operations are sufficient to arrive at any target context-free grammar, as long as every rule has been used in generating the training data. A major problem is, however, to decide on which non-terminal symbols to merge and chunk.

A heuristic approach to context-free grammar induction (Cook *et al.*, 1976; Langley & Stromsten, 2000) starts with the merge operation that reduces the size of the grammar the most (after deleting redundant rules). For instance, when learning from the sentences in example 1, making "the" and "a" of the same nonterminal category (by merging their non-terminals) will allow one to delete one rule (and get rid of 1 left-hand side and 5 right-hand side symbols). The merge operation is then repeated for the best nonterminal pair until no more gains in description length can be made. Then the best chunk operation is applied, and the merge cycle is repeated. Hence, a sketch of the algorithm is:

1. *incorporate* all sentences (assigning a single non-terminal to every occurrence of the same word);
2. find the pair of nonterminals (X, Y) that reduces the grammar most if merged;
3. *merge*(X, Y) and continue with step 2 until no more gains in grammar size are made;
4. find the pair of nonterminals (X, Y) that reduces the grammar most if chunked;
5. *chunk*(X, Y) and return to step 2; stop if neither *chunk* nor *merge* leads to any more gains.

Question 7 *Apply this algorithm to the sentences in 1.*

Question 8 *Assigning equal probability to every rule with the same left-hand side what is the likelihood of the data given this induced grammar?*

Again, we can also define a prior probability based on the size of the grammar (where the size is again counted as the total number of characters – counting a nonterminal always as just 1 – in the rewrite rules, which should be 22).

Question 9 *Assuming a prior probability that decreases with the size of the grammar, which model – the Markov model from section 1 or the PCFG model of this section – maximizes the posterior probability? What is the main difference between the two models? (You can answer this question without doing any calculations).*

References

Chomsky, N. (1957). *Syntactic Structures*. The Hague: Mouton.

Cook, C., Rosenfeld, A. & Aronson, A. (1976). Grammatical inference by hill climbing. *Informational Sciences (now: Information Sciences)* **10**, 59–80.

Fitch, W. T. & Hauser, M. D. (2004). Computational constraints on syntactic processing in a nonhuman primate. *Science* **303**, 377–380.

Gentner, T. Q., Fenn, K. M., Margoliash, D. & Nusbaum, H. C. (2006). Recursive syntactic pattern learning by songbirds. *Nature* **440**, 1204–1207.

Langley, P. & Stromsten, S. (2000). Learning context-free grammars with a simplicity bias. In: *Proceedings of the Eleventh European Conference on Machine Learning*, pp. 220–228. Barcelona: Springer-Verlag.

Stolcke, A. (1994). *Bayesian Learning of Probabilistic Language Models*. Ph.D. thesis, Dept. of Electrical Engineering and Computer Science, University of California at Berkeley.