Solutions to Exercise 3

Synthesis for Interpretable Machine Learning

Reliable and Interpretable Artificial Intelligence 2017
ETH Zürich

October 11, 2017

Problem 1

Task 1 Design domain specific language (called TSeq) that works over sequences instead of trees.

Solution Figure 1 shows a loop-free and branch-free program that accumulates context with values from the input by means of navigating within the input (using `Move` instructions) and writing the observed values (using `Write` instructions). The result of executing a program in this langauge is an accumulated context which is used to condition the prediction.

Move Instructions We define four basic types of `Move` instructions – `LEFT` and `RIGHT` that move to the previous and next character respectively, `PREV_CHAR` that moves to the most recent position in the input with the same value as the current character x_t. Additionally, for each character c in the input vocabulary we generate instruction `PREV_CHAR(c)` that navigates to the most recent position of character c. We note that all `Move` instructions are allowed to navigate only to the left of the character x_t that is to be predicted.

Write Instructions We define three `Write` instructions – `WRITE_CHAR` that writes the value of the character at current position, `WRITE_HASH` that writes a hash of all the values seen between the current position and the position of last `Write` instruction, and `WRITE_DIST` that writes a distance (i.e., number of characters) between current position and the position of last `Write` instruction.

```
SimpleProgram ::= ϵ
                | Move; SimpleProgram
                | Write; SimpleProgram

Move ::= LEFT | RIGHT | PREV_CHAR
Write ::= WRITE_CHAR | WRITE_HASH | WRITE_DIST
```

Figure 1: Syntax of a simple domain specific language for character level language modeling. This is a subset of a more complex language described at `http://www.srl.inf.ethz.ch/papers/charmodel-iclr2017.pdf`.

Example With `Write` and `Move` instructions, we can express various programs that extract useful context for a given position t in text x. For example, we can encode the context used in trigram language model with a program `LEFT WRITE_CHAR LEFT WRITE_CHAR`. We can also express programs such as `LEFT PREV_CHAR RIGHT WRITE_CHAR` that finds the previous occurrence of the character on the left of the current position and records the character following it in the context.

Task 2 Write down programs in TSeq that you believe are useful for the given queries.

Solution

(a) Mg12 He0 Ai31 Fe14 Mg13 Ag22 **?**

`EMPTY`: empty program cooresponds to uncoditioned prediction

`LEFT PREV_POS RIGHT WRITE_CHAR`: first letter of previous word

(b) Mg12 He0 Ai31 Fe14 Mg13 Ag22 Fe**?**

`LEFT PREV_POS WRITE_CHAR RIGHT WRITE_CHAR`: moves to previous usage of letter 'e' and then to the character that was written afterwards

(c) Mg12 He0 Ai31 Fe14 Mg13 Ag22 Fe15**?**

`PREV_CHAR`(␣) `WRITE_DIST`: size of current word

1 Problem 2

Task 1 Write down function f_{bigram} that coressponds to a bigram model.

Solution

$$f_{bigram} ::= \texttt{LEFT WRITE\_CHAR}$$

Task 2 For each word in the dataset the program f_{bigram} specifies the context to be used for its prediction. Write down a table that contains the context (i.e., evaluates program f_{bigram}) for each position of query from task 1b).

Solution The computed context using program f_{bigram} for each position of query 1b is shown in Table 1.

Task 3 Once we computed the context for each position we would like to use it for the actual prediction. Build a model based on maximum likelihood estimation that is used to calculate probability of character at a given position t given it context (i.e., $p(x_t \mid f_{bigram}(x_{<t}))$) as follows:

$$p(w_i \mid f_{bigram}(x_{<t})) = \frac{c(f_{bigram}(x_{<t}) \cdot w_i)}{\sum_w c(f_{bigram}(x_{<t}))}$$

where $c(\boldsymbol{v})$ denotes the number of times the value $\boldsymbol{v}$ has been seen in the training data.

Solution We simply count the entries of the 1 and use them to answer queries. That is, using these count we answer the query 1b as follows:

$$p(0 \mid f_{bigram}(x_{<32})) = p(0 \mid e) = \frac{c(0 \cdot e)}{c(e)} = \frac{1}{2}$$

where $c(e)$ appears at positions 8 and 17 and $c(0 \cdot e)$ at position 8.

$$p(1 \mid f_{bigram}(x_{<32})) = p(1 \mid e) = \frac{c(1 \cdot e)}{c(e)} = \frac{1}{2}$$

where $c(e)$ appears at positions 8 and 17 and $c(1 \cdot e)$ at position 17.

$$p(2 \mid f_{bigram}(x_{<32})) = p(2 \mid e) = \frac{c(2 \cdot e)}{c(e)} = \frac{0}{2}$$

where $c(e)$ appears at positions 8 and 17 and $c(2 \cdot e)$ does not appear.

...

Task 4 Now consider a different program $f' \in TSeq$. In order to compare the programs f_{bigram} and f' we need to define a cost function $cost : TSeq \times \mathcal{D} \to \mathbb{R}$ that assigns cost to each program when trained on a dataset $\mathcal{D}$. Define what a suitable cost function.

Solution The cost function can be defined as follows:

$$cost(f, D) = \frac{1}{n} \sum_{x_t \in D} -\log_2 p(x_t \mid f(x_{<t}))$$

where D is a dataset of training examples (one sample for each character in the input) and f is the program for which we want to compute the cost.

position	x_t	$f_{bigram}(x_{<t})$
1	M	∅
2	g	M
3	1	g
4	2	1
5	␣	2
6	H	␣
7	e	H
8	0	e
9	␣	0
10	A	␣
11	i	A
12	3	i
13	1	3
14	␣	1
15	F	␣
16	e	F
17	1	e
18	4	1
19	␣	4
20	M	␣
21	g	M
22	1	g
23	3	1
24	␣	3
25	A	␣
26	g	A
27	2	g
28	2	2
29	␣	2
30	F	␣
31	e	F
32	**?**	e

Table 1: Computed context using program f_{bigram} for each position of query 1b.