UCLA

Dept. of Electrical and Computer Engineering

ECE 231A, Spring 2020

Assessment for Part 2

205418959 Shengze Ye

Abstract

In this project, we mainly explore how to use arithmetic coding to compress the data to the entropy and the entropy rate. We used a Markov process, Random walk on a chessboard, to implement arithmetic encoding and decoding. Two versions of the coding schemes are implemented. One targets the entropy of stationary distribution, the other targets the entropy rate of the stochastic process.

Experiments

1. **Generate your personalized long and short sequences of king movements.**

   My UID number is 205418959, after running the Sequence generator function, I got two sequence files of number 595. These are the two files that I am going to work with.

2. **Arithmetic coding that targets the entropy of the stationary distribution.**

a. **What was the stationary distribution that you computed for the nine possible positions of the king in the problem 8 of problem set 2?**

1	2	3
4	5	6
7	8	9

The king must move at each step in the chessboard and that it is equally likely to choose any of the legal chess moves for a king that are available to it. We can use weighted graph to solve the problem.

The stationary distribution is given by $\mu_i = W_i/2W$, where W_i is the number of edges emanating from the node i. Based on the structure of chessboard, $W_1 = W_3 = W_7 = W_9 = 3$, and $W_2 = W_4 = W_6 = W_8 = 5$, and $W_5 = 8$. Therefore, we can get the stationary distribution.

$$\mu = [\frac{3}{40}, \frac{5}{40}, \frac{3}{40}, \frac{5}{40}, \frac{8}{40}, \frac{5}{40}, \frac{3}{40}, \frac{5}{40}, \frac{3}{40}]$$

In addition, as we need to encode symbol EOT at the same time, we need to scale all the stationary probabilities by 14/15.

$$\mu = [\frac{7}{100}, \frac{7}{60}, \frac{7}{100}, \frac{7}{60}, \frac{14}{75}, \frac{7}{60}, \frac{7}{100}, \frac{7}{60}, \frac{7}{100}, \frac{1}{15}]$$

b. Compute the entropy of this stationary distribution.

$$H(X) = \sum_i p(x)\, log_2\big(p(x)\big)$$

$$= -0.075 * \log(0.075) * 4 - 0.125 * \log(0.125) * 4 - 0.2 * \log(0.2)$$

$$= 3.085475297227334\ bits$$

So, it takes approximately 3.09 bits to encode one symbol.

c. Using short sequence, implement an arithmetic coder and the associated decoder that seeks to compress to the entropy of the stationary distribution.

For encoding and decoding, we need an EOT symbol to denote the end of the sequence, as the EPT comes at the end of 15 symbols, we assign a probability of 1/15 to EOT symbol. So, we have to adjust all the stationary probability for the 9 digits to account for the EOR symbol. The detailed implementation and procedures for encoding and decoding are included in the MATLAB code folder. Here, I just report the result of arithmetic coding.

The short sequence 595 is: 3536953269842320
The code is: 0011 0111 0100 0001 1011 0111 1001 1011 0110 1111 1101 0011 01
The value is: 0.215846515129511

The length of the code is 50, with additional "1" at the end to make sure the EOF symbol is achieved. We have 16 symbols in this short sequence, the average code length for a symbol is 3.125 bits, very close to the entropy of 3.09 bits.

d. Now we will encode the long sequence. How close did you get to the entropy in terms of bits? What was the percent of excess redundancy?

The length of the long sequence is 10000. Thus, we adjust our probability of the EOT symbol to be 1/100. Again, we need to scale all the stationary probability account for the EOT symbol. The details are included in the code folder.

The value is: 0.397849670782595
The total length of the code: 31018 bits
The average length for a symbol: 31018/10000 = 3.1018 bits/symbol
Excess redundancy:

$$Redundancy = 100 * \frac{Rate - H(X)}{H(X)} = \frac{3.1018 - 3.0855}{3.0855} = 0.529\%$$

We see that as the length of the sequence increases, the average length for a symbol becomes

closer to the entropy. For n = 10000, the average length is 3.1018 bits and the difference between the rate and the entropy is 0.0163 bits. In addition, the excess redundancy is 0.529%. we can speculate that as n goes to infinity, the average length will be infinitely close to the entropy.

3. Arithmetic coding that targets the entropy rate of the stochastic process.

a. For each of the nine positions, what is the conditional distribution for the next position of the king in problem 8 of problem set 2?

1	2	3
4	5	6
7	8	9

The king must move at each step in the chessboard and that it is equally likely to choose any of the legal chess moves for a king that are available to it. At first, we need to figure out what the next position could be for the nine positions, respectively.

Next position for position 1: [2, 4, 5, EOT]
Next position for position 2: [1, 3, 4, 5, 6, EOT]
Next position for position 3: [2, 5, 6, EOT]
Next position for position 4: [1, 2, 5, 7, 8, EOT]
Next position for position 5: [1, 2, 3, 4, 6, 7, 8, 9, EOT]
Next position for position 6: [2, 3, 5, 8, 9, EOT]
Next position for position 7: [4, 5, 8, EOT]
Next position for position 8: [4, 5, 6, 7, 9, EOT]
Next position for position 9: [5, 6, 8, EOT]

Since the king will equally choose any of the legal chess moves, for each position, the conditional distribution is uniform distribution.

For position 1, 3, 7, 9, the conditional probability is:

$$\mu = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$$

For position 2, 4, 6, 8, the conditional probability is:

$$\mu = [\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]$$

For position 5, the conditional probability is:

$$\mu = [\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}]$$

However, we need to consider the EOT symbol with probability 1/15, and we need to multiply 14/15 to scale all the other conditional probabilities. The nine positions could be divided into

three categories. Position 1, 3, 7, 9 can equally move to 3 different positions. Position 2, 4, 6, 8 can equally move to 5 different positions and position 5 can move to 8 different positions. Below is the conditional probability that we consider EOT.

For position 1, 3, 7, 9, the conditional probability is:

$$\mu = [\frac{14}{45}, \frac{14}{45}, \frac{14}{45}, \frac{1}{15}]$$

For position 2, 4, 6, 8, the conditional probability is:

$$\mu = [\frac{14}{75}, \frac{14}{75}, \frac{14}{75}, \frac{14}{75}, \frac{14}{75}, \frac{1}{15}]$$

For position 5, the conditional probability is:

$$\mu = [\frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{14}{120}, \frac{1}{15}]$$

b. What is the entropy of the PMF or the next position given each of the nine possible starting positions?

If we do not consider the EOT symbol.

$$H(X_2|X_1 = i) = log3\ bits = 1.585\ bits \qquad for\ i = 1,3,7,9$$
$$H(X_2|X_1 = i) = log5\ bits = 2.322\ bits \qquad for\ i = 2,4,6,8$$
$$H(X_2|X_1 = i) = log8\ bits = 3\ bits \qquad for\ i = 5$$

If we consider the EOT symbol.

$$H(X_2|X_1 = i) = -\frac{14}{45}\log\left(\frac{14}{45}\right) * 3 - \frac{1}{15}\log\left(\frac{1}{15}\right) = 1.8327\ bits \qquad for\ i = 1,3,7,9$$
$$H(X_2|X_1 = i) = -\frac{14}{75}\log\left(\frac{14}{75}\right) * 5 - \frac{1}{15}\log\left(\frac{1}{15}\right) = 2.5205\ bits \qquad for\ i = 2,4,6,8$$
$$H(X_2|X_1 = i) = -\frac{14}{120}\log\left(\frac{14}{120}\right) * 8 - \frac{1}{15}\log\left(\frac{1}{15}\right) = 3.1534\ bits \qquad for\ i = 5$$

c. What was the entropy rate for the stochastic process that you computed in problem 8 of problem set 2?

$$H(X) = \sum_{i=1}^{9} \mu_i H(X_2|X_1 = i)$$
$$= 0.3 * log3 + 0.5 * log5 + 0.2 * log8 = 2.24\ bits$$

The entropy rate is 2.24 bits.

d. Using the short sequence, implement an arithmetic coder and the associated decoder that seeks to compress to the entropy rate of the stochastic process.

The encoding and decoding implementation in this part are somewhat different from the

previous question. The main point is that for each step, we may use a different probability distribution. And it depends on the previous symbol that we encode. Therefore, we need to provide the current and the previous symbol to the encode function at the same time. Another thing that we should notice is that we need to create a dictionary for conditional probability distributions, which could help us find the corresponding distribution for next step.

At first, I create a cell to store all the possible next steps for each digit.

```
next_step = cell(1, 9);
next_step{1} = [2, 4, 5, 10];
next_step{2} = [1, 3, 4, 5, 6, 10];
next_step{3} = [2, 5, 6, 10];
next_step{4} = [1, 2, 5, 7, 8, 10];
next_step{5} = [1, 2, 3, 4, 6, 7, 8, 9, 10];
next_step{6} = [2, 3, 5, 8, 9, 10];
next_step{7} = [4, 5, 8, 10];
next_step{8} = [4, 5, 6, 7, 9, 10];
next_step{9} = [5, 6, 8, 10];
```

Secondly, I locked down the conditional probability distributions. Here, the start probability is stationary distribution. I use a function to create the ‘ll’ and ‘ul’ vectors for each digit. Then I store them in two cells.

```
start_prob = [3/40, 5/40, 3/40, 5/40, 8/40, 5/40, 3/40, 5/40, 3/40];
condition_prob_1379 = [1/3, 1/3, 1/3];
condition_prob_2468 = [1/5, 1/5, 1/5, 1/5, 1/5];
condition_prob_5 = [1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8, 1/8];

[ll_start,ul_start] = get_ll_ul(start_prob,1/100);
ll_condition = cell(1, 9);
ul_condition = cell(1, 9);
for i = 1:9
    if (i == 1 || i == 3 || i == 7 || i == 9)
        [ll_condition{i}, ul_condition{i}] = get_ll_ul(condition_prob_1379, 1/100);
    elseif (i == 2 || i == 4 || i == 6 || i == 8)
        [ll_condition{i}, ul_condition{i}] = get_ll_ul(condition_prob_2468,1/100);
    else
        [ll_condition{i}, ul_condition{i}] = get_ll_ul(condition_prob_5,1/100);
    end
end
```

For encoding part, we need the previous symbol and the current symbol. We first use next step dictionary to find out all the possible next steps for the previous symbol, and we also need to know the index of the current symbol. Next, we could find the right ‘ul’ and ‘ll’. Then we just use the regular encode function in question 2 to encode the symbol and return the range.

```
function [new_low,new_high] = q3_encode_symbol(symbol_prev, symbol_cur, low, high, next_step, ll_condition, ul_condition)

    symbol_position = find(next_step{symbol_prev} == symbol_cur);
    ll = ll_condition{symbol_prev};
    ul = ul_condition{symbol_prev};
    [new_low, new_high] = encode_symbol(symbol_position, low, high, ll, ul, 0);
    fprintf(1, 'Encode symbol %d\n', mod(symbol_cur, 10));
    fprintf(1, 'New interval is [%f, %f) \n',new_low, new_high);
end
```

The sending bits part is same with the previous question.

The crucial part for this problem is the decoding. As each step is depends on the previous step, we need to decode the next symbol based on the knowledge of the current symbol. This is the reason why we need the previous symbol in the input parameters.

```
function [symbol, new_value] = q3_decode_symbol(value, symbol_prev, next_step, ll_condition, ul_condition)
    ll = ll_condition{symbol_prev};
    ul = ul_condition{symbol_prev};
    [symbol_position, new_value] = decode_symbol(value, ll, ul, 0);
    symbol_real = next_step{symbol_prev}(symbol_position);
    symbol = symbol_real;
    symbol = mod(symbol, 10);
    fprintf('Symbol is %d. New value is %f. \n', symbol, new_value);

end
```

Again, the first thing we need to do is figuring out the conditional probability distribution. Once we know that, we use the regular function to get the decoded symbol and map it to the real symbol in the end.

More detailed implementations are included in the code folder.

Results:

The short sequence 595 is: 3536953269842320

The code is: 0011 0111 0011 1000 1011 1000 1101 1010 0101

The value is: 0.215709260271979

We can see from the result that the length of code is 36, less than 50, which was the length of code using stationary distribution. We also have 16 symbols for this sequence. Therefore, the average length to code a symbol is 2.25 bit, very close to the entropy rate of this Markov process. The final value is different from what we get in question 2, this is because for each step we use a different conditional distribution. Accordingly, the final value is different.

e. **Now we will encode the long sequence. How close did you get to the entropy rate in terms of bits? What was the percent of excess redundancy?**

The length of the long sequence is 10000. Thus, we adjust our probability of the EOT symbol to be 1/100. Again, we need to scale all the probability. Different from the previous question, we need to scale all the 9 conditional probability distributions since the EOT symbol could appear after any digits. The details are included in the code folder.

The value is: 0.401070300097706

The total length of the code: 22504 bits

The average length for a symbol: 22504/10000 = 2.2504 bits/symbol

Excess redundancy:

$$Redundancy = 100 * \frac{Rate - H(X)}{H(X)} = \frac{2.2504 - 2.24}{2.24} = 0.464\%$$

For n = 10000, the average length for stochastic processing is 2.2504 bits and the difference between the rate and the entropy is 0.0104 bits. In addition, the excess redundancy is 0.464%. we can speculate that when n is very large, the average length will be infinitely close to entropy rate. Another thing that we need to take into consideration is the probability of EOT. If we make the probability of EOT smaller, we will be closer to the entropy rate. But in practice, I think we need to set the probability of EOT according to statistical results.

f. What was the percentage rate reduction that was achieved by compressing to the entropy rate instead of the entropy?

$$Rate\ reduction = 100 * \frac{R1 - R2}{R1} = \frac{3.1018 - 2.2504}{3.1018} = 27.459\%$$

The percentage of rate reduction is 27.459%. we see that using arithmetic coding targeting at entropy rate could reduce about 27.5% than targeting at entropy. This helps us save a lot of memory when we encode very long sequences.