PM520–Homework 4

Chengliang Dong Thursday 26th March, 2015

1 Introduction

In the current homework, we were asked to break the code, encoded using one-to-one substitution of English letters. The substitution is the same for the entire message and a table of pairwise letter frequencies in the original text was given. We propose to use Markov Chain Monte Carlo (MCMC) with Metropolis-Hastings (MH) algorithm to break the code.

In statistics, MCMC methods are a class of algorithms for sampling from a probability distribution based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. The state of the chain after a number of steps is then used as a sample of the desired distribution. The quality of the sample improves as a function of the number of steps.

Examples of random walk MCMC methods includes MH algorithm, which generates a random walk using a proposal density and a method for rejecting some of the proposed moves. This algorithm involves two steps, initialization and iteration. Given data D, parameter(s) θ, transitional kernel Q and probability function $P(D|\theta)$. In initialization step, we initialize θ to some value. In iteration, we update θ using the following steps.

1. calculate
$$h = min(1, \frac{P(\theta'|D)q(\theta' \to \theta)}{P(\theta|D)q(\theta \to \theta')})$$
where,
$$P(\theta'|D) = \frac{P(D|\theta')\pi(\theta')}{P(D)}$$
$$P(\theta|D) = \frac{P(D|\theta)\pi(\theta)}{P(D)}$$
2. move to θ' with probability h

In our current homework settings, we used symmetrical transitional kernels and same prior probability for θ' and θ. Therefore, we have,
$$h = min(1, \frac{P(D|\theta')}{P(D|\theta))})$$

In computer science and statistics, the Jaro–Winkler distance (Winkler, 1990) is a measure of similarity between two strings. It is a variant of the Jaro distance metric (Jaro, 1989, 1995), a type of string edit distance, and was developed in the area of record linkage (duplicate detection) (Winkler, 1990). The higher the Jaro–Winkler distance for two strings is, the more similar the strings are. The Jaro–Winkler distance metric is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match and was used to measure performance of our decoding metrics.

As the performance of MCMC MH can be influenced by several factors, such as the size of data, number

of iterations, initialization and transitional kernels. Here we would like to investigate how some of these factors can impact its performance on decoding the message.

2 Method

2.1 Implementation of MCMC MH algorithm

2.1.1 Frequency table pre-processing

Letter frequency table was process by normalization (normalized such that the sum of all frequencies is equal to 1).

2.1.2 Initialization

Four different initialization processes were used and evaluated. First way is to use row sum of the frequency table of each of the 26 letters to estimate the frequency of each letter occurring in the original text, link the ranked frequency to the actual letter frequency in the encoded text and use such linked letter relation as the initial codebook for code breaking. Second way is to use column sum of the frequency table of each of the 26 letters to estimate the frequency of each letter occurring in the original text, link the ranked frequency to the actual letter frequency in the encoded text and use such linked letter relation as the initial codebook for code breaking. Third way is to use column sum plus row sum of the frequency table of each of the 26 letters to estimate the frequency of each letter occurring in the original text, link the ranked frequency to the actual letter frequency in the encoded text and use such linked letter relation as the initial codebook for code breaking. Fourth way is to use a random permutation of the 26 letters as initial codebook.

2.1.3 Posterior probability

To calculate posterior probability $P(D|\theta)$, we scanned the encoded text using consecutive 2-letters as a unit and calculated the posterior probability for the whole text by multiplying all corresponding consecutive 2-letters frequencies. Note that we discarded all non-letter or uni-letter text content for the sake of simplicity. For example, for decoded text "ab cde f.", given codebook $a \to b, c \to d, d \to e, e \to f, f \to g$ we calculated its posterior probability $P(D|\theta)$ using the following formula,

$$P(D|\theta) = P(bc|a \to b, b \to c)P(de|c \to d, d \to e)P(ef|d \to e, e \to f)$$

2.1.4 Kernel

Permutation of two random letters in the codebook was used as transitional kernel in MCMC. While other kernels could be used, they are not examined in the current homework.

2.2 Factors of interest

We were interested in how size of data, number of iterations and initialization method impact the time and accuracy of decoding. For size of data, we chose an article of 5,724 words, 1,370 words, 693 words and 243 words and marked them as "long message", "medium message", "short message" and "very

short message". As for number of iterations, we chose 100, 500, 1000, 5000, 10000 iteration for running MCMC MH algorithms, each with 20 repeats. As for initialization, we chose aforementioned four different initialization methods to be compared.

2.3 Performance measurement

All four messages were decoded using more than 10000 iteration with "column sum" optimization for 5 times and the results were manually checked. The ones that are 100 percent readable and sensible were used as "gold standard" for performance measure. Normalized Jaro-Winkler distance (1 indicates perfect match, 0 indicates no match) was used to measure the performance of our decoded message and results were shown in the following sections.

3 Result

3.1 Decoded Message

Multiple configurations can be used to decode the messages. Here we used 10000 iterations with column sums as initialization point for all four messages. Note that all letters are transformed to lower cases. As for "very short message", the content is as follows,

"the beginning is simple to mark. we were in sunlight under a turkey oak, partly protected from a strong, gusty wind. i was kneeling on the grass with a corkscrew in my hand, and clarissa was passing me the bottle - a nineteen-eightyseven daumas gassac. this was the moment, this was the pinprick on the time map: i was stretching out my hand, and as the cool neck and the black foil touched my palm, we heard a mans shout. we turned to look across the field and saw the danger. next thing, i was running toward it. the transformation was absolute: i dont recall dropping the corkscrew, or getting to my feet, or making a decision, or hearing the caution clarissa called after me. what idiocy, to be racing into this story and its labyrinths, sprinting away from our happiness among the fresh spring grasses by the oak. there was the shout again, and a childs cry, enfeebled by the wind that roared in the tall trees along the hedgerows. i ran faster. and there, suddenly, from different points around the field, four other men were converging on the scene, running like me. i see us from two hundred feet up, through the eyes of the buzzard we had watched earlier, soaring, circling, and dipping in the tumult of currents: five men running silently toward the center of a hundred-acre field. i approached from the southeast, with the wind at my back."

As for "short message", "medium message" and "long message", the decoded versions are attached.

3.2 Analysis

Since MCMC is computationally costly for relatively long messages, we were only able to finish analysis for "very short message" with 20 repeats with all different configurations. As for longer messages, only 1 repeat was done with initiation configuration of "row sum" approximation and "random permutation" and results were shown below.

3.2.1 Accuracy

From the following figure 1, we can see that at fixed length of messages, when we increase number of iterations to 5000, accuracy of decoding increases and also standard deviation also increases. But when we increase number of iterations from 5000 to 10000, there is no significant increase of accuracy of decoding. Moreover, when we optimize initialization using metrics such as "row sum", "column sum" and "all sum" of entries of frequency table to approximate uni-letter frequency, there is also an increase of frequency, compared to none optimized version. In addition, the performance of these three different optimization metric are similar to each other. In some scenarios, "row sum" approximation performs better, such as in 500 iteration; but in other scenarios, other approximations may perform better.

3.2.2 Time

From the following figure 2, we can see that at fixed length of messages, when we increase number of iterations, time of decoding increases and also standard deviation also increases. Moreover, when we optimize initialization using metrics such as "row sum", "column sum" and "all sum" of entries of frequency table to approximate uni-letter frequency, there is no significant increase of time consumption. This may due to the fact that such optimization only iterates once, which is at the begining of MCMC, so it does not impact speed much. Note that "column sum" has interestingly high time complexity and standard deviation compared to other metrics in the first four scenarios, but I have not figured out why yet.

3.2.3 Short, Medium and Long messages

In general, we have observed increased time and accuracy for longer messages, when we fix all other settings, including number of iterations and optimization metric. Moreover, as for time consumption, the increase seems linear in term of length of the message. In addition, we observed similar trend for time and accuracy as in previous sections, in general, time and accuracy both increase with increase of iterations, but for accuracy, once it can reach plateau.

Time consumption (s) of messages with "row sum" optimization scheme			
Iteration	Short Message	Medium Message	Long Message
100	19.9	37.6	168.1
500	103.8	192.1	762
1000	199.2	377.9	1620.1
5000	952.8	1905.1	6480.1
10000	1896.2	3780.2	16200.5

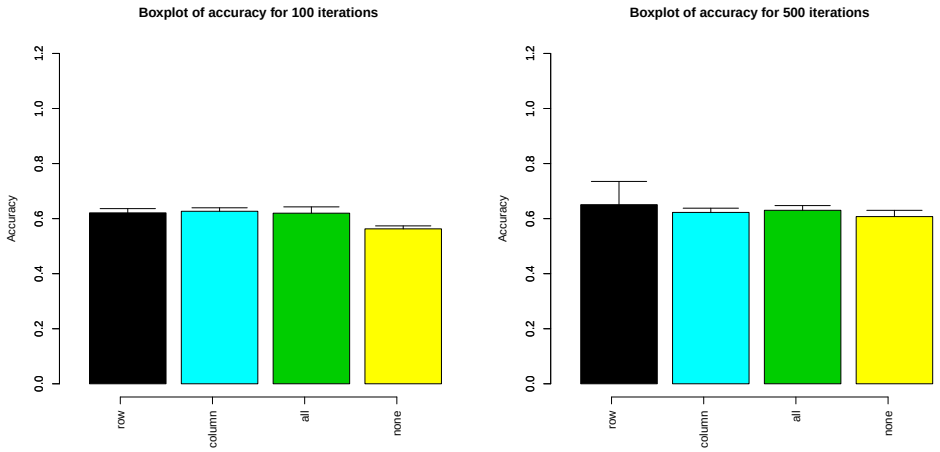
Time consumption (s) of messages with no optimization scheme			
Iteration	Short Message	Medium Message	Long Message
100	17.7	32.6	160.1
500	102.7	190.1	762.1
1000	199.0	375.9	1618.1
5000	950.8	1904.0	6477.0
10000	1895.2	3777.2	16120.1

Jaro-Winkler distance of messages with "row sum" optimization scheme			
Iteration	Short Message	Medium Message	Long Message
100	0.66	0.62	0.66
500	0.94	0.66	0.65
1000	0.67	0.99	0.64
5000	1	1	1
10000	1	1	1

Jaro-Winkler distancer of messages with no optimization scheme			
Iteration	Short Message	Medium Message	Long Message
100	0.56	0.52	0.56
500	0.65	0.62	0.63
1000	0.64	0.64	0.67
5000	1	1	1
10000	0.60	1	1

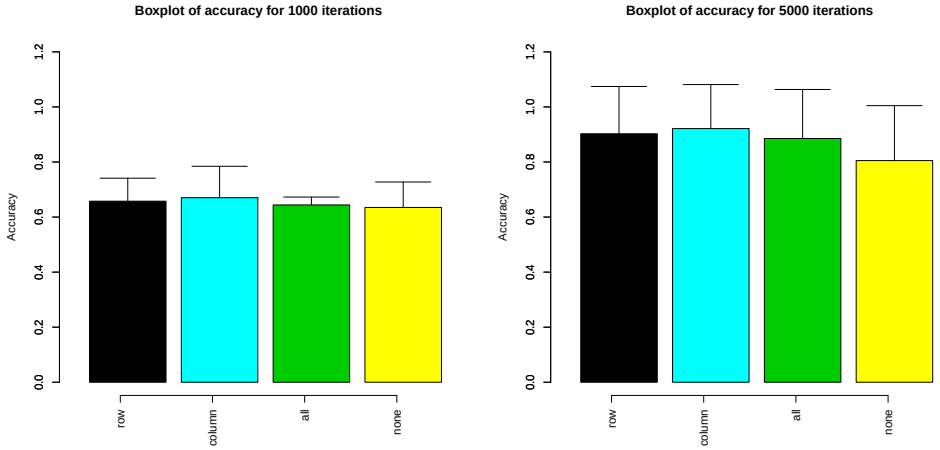
4 Conclusion

In the current homework, we were asked to break the code, encoded using one-to-one substitution of English letters, using Markov Chain Monte Carlo (MCMC) with Metropolis-Hastings (MH) algorithm Jaro-Winkler distance as accuracy measurement. We broke the code found that at fixed length of messages, when we increase number of iterations to 5000, accuracy of decoding increases and also standard deviation also increases. But when we increase number of iterations from 5000 to 10000, there is no significant increase of accuracy of decoding. Moreover, when we increase number of iterations, time of decoding increases and also standard deviation also increases. In addition, we have observed increased time and accuracy for longer messages, when we fix all other settings, including number of iterations and optimization metric. Lastly, optimization methods for initialization can increase accuracy of decoding without adding too much time consumption.



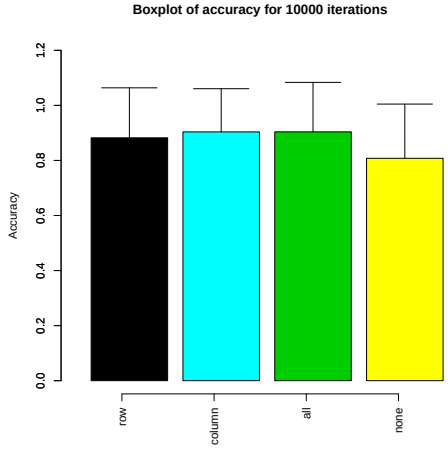
(a) 100 Iterations

(b) 500 Iterations



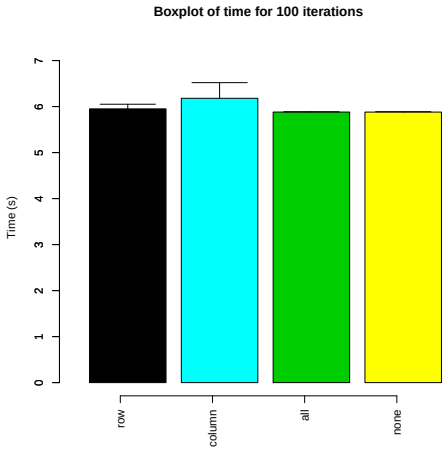
(c) 1000 Iterations

(d) 5000 Iterations

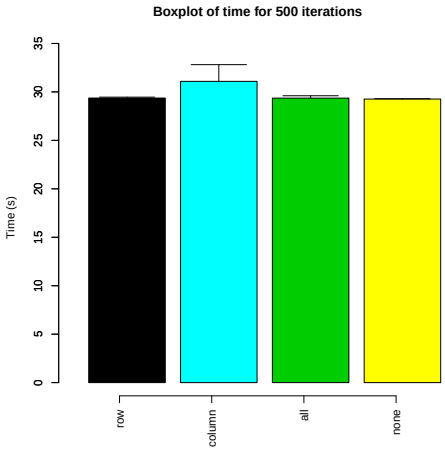


(e) 10000 Iterations

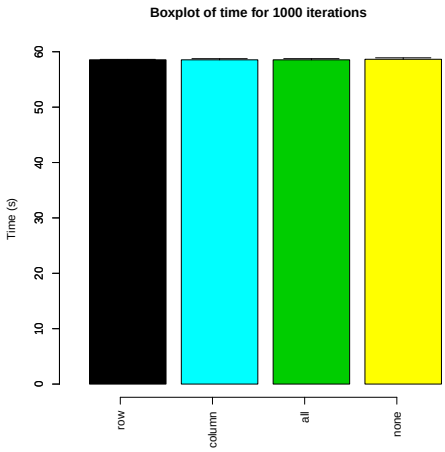
Figure 1: Normalized Jaro–Winkler distance (accuracy measurement) for very short messages at different configurations.



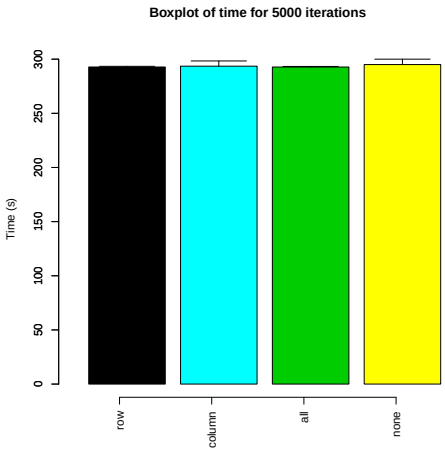
(a) 100 Iterations



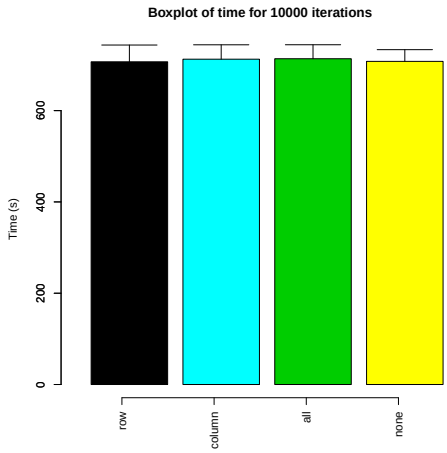
(b) 500 Iterations



(c) 1000 Iterations



(d) 5000 Iterations



(e) 10000 Iterations

Figure 2: Time for very short messages at different configurations.