International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 9, November - 2012

Grammatical inference for off-line Thai handwritten character recognition

Anuchit Jitpattanakul
Department of Mathematics, Faculty of Applied Science, King Mongkut's University of Technology North Bangkok,
1518 Pibulsongkram Road, Bangsue, Bangkok 10800, Thailand

Abstract

Grammatical inference (GI) is a newly alternative approach in machine learning. A grammatical representation will be inductively inferred from a set of input examples that are transformed into languages domain. In last decade, many recognition problems have been tested and shown that GI is an effective and efficient alternative to solve these problems. In this paper, the problem of off-line Thai handwritten character recognition is selected to be solved by GI. The experimental result is shown the performance of this approach in term of recognition rate.

1. Introduction

Grammatical inference is well known as one of the most attractive research topics in scientific learning. Both theoretical and experimental results have been obtained in fields like machine learning, data mining, and many related others in last three decades. The usefulness of GI is that the learning result plays as an algebraic system called automata. Moreover, GI is supported with mathematical fundamentals and well-proved theorems. Recently, many recognition problems (i.e., music recognition, speech recognition, DNA sequence recognition) have been tested and shown that GI is an efficient and effective approach to solve these problems [1].

Typically, GI refers to process of learning grammatical representations, which are in form of automata or grammars, of languages from sequential and structured data in form of strings. The learning consists of identifying a correct representation for the target language given a finite set of strings of the language. The representation is used to describe the target language and can be in different forms that depend on the class of languages (e.g., deterministic finite automata (DFA) for regular languages and context-free grammars (CFG) for context-free languages).

Many classes of formal languages have been theoretically studied their learnability and complexity on different learning scenario. Both negative and positive results have been academically reported to help us get better understanding to learning properties of each class of languages. In 1967, a concept of identification in the limit has been firstly proposed by Gold [2] to abstractly measure learning achievement. Later in 1978, Gold [3] proved that the class of regular languages, represented by DFA, is identifiable in the limit from polynomial time and data by using both positive and negative examples. In 1992, Oncina [4] supported Gold's result by proposing a classical state-merging algorithm called RPNI for learning DFA from an informant in polynomial time. The RPNI is the first learning algorithms achieving in the field of GI. In 1994, the Abbadingo competition has been organized to find the best learning algorithm in GI. A heuristic algorithm winning in the competition has been developed to be EDSM (evidence driven state merging) algorithm.

Handwritten character recognition (shortly HCR) is a challenging recognition problem in machine learning. The goal is to make a computer understand and determine which character a human wrote. The handwritten character recognition can be categorized to off-line and on-line recognition. The main difference is that off-line HCR has no temporal information about trajectory of writing, but on-line HCR has. A number of applications have been reported to show the interest of HCR such as bank check recognition, signature verification and handwritten address recognition [5]. In context of Thai languages, scientific researchers have interested Thai handwritten character recognition since last decades. Many learning algorithms has been proposed to solve this problem such as fuzzy logics and neural networks [6], Ant-Miner algorithms [7]. To the best of our knowledge, there is no any GI learning algorithms that are tested to the Thai handwritten character recognition.

In this work, the problem of off-line Thai handwritten character recognition is select to solve with learning algorithm in GI. The RPNI and EDSM algorithm are selected to test with the problem. The experimental result will be shown the performance of this approach in term of recognition rate.

The rest of this paper is organized as follows. Section 2 gives preliminaries used in this work. Section 3 contains implementation of Thai HCR. Section 4 shows experimental results and comparison results. Conclusion and future works are given in the last section.

2. Preliminaries

This section provides the background knowledge used in this work. Firstly, we begin with introducing basic definitions and notations in grammatical inference. Two learning algorithms, which are RPNI and EDSM, will also be provides this sections. Then, we will explain Thai handwritten character recognition.

2.1 Grammatical inference and learning algorithms

Grammatical inference is about find grammars or automata from a set of given data in from of strings. By solving with GI, problems must be initially transformed into language domain. In this section we give the basic definitions used in GI and show two state-merging algorithms RPNI and EDSM that are used to solve the HCR problem in case of Thai characters. The basic definition and notations used in this works are shown as follows.

Let Σ be an alphabet that is a finite and nonempty set of symbols called letters. The size of Σ is a number of letters, denoted by $|\Sigma|$. A finite sequence of letters from Σ is called a string. Given a string w, the length of strings is the total number of letters appearing in w and it is denoted by $|w|$. The string with length zero is called the null string denoted by λ. The infinite set of all possible strings defined over Σ, denoted by Σ^*, is the set of all finite-length strings generated by concatenating zero or more letters of Σ. Given a string $w = uv$, a string u is a prefix of w if and only if there exists a string v in Σ^*. A language over Σ denoted by L is any subset of Σ^*. The complement of a language L is defined by $L' = \Sigma^* - L$.

A finite automaton is a grammatical representation that is typically defined as a 5-tuple $M = (\Sigma, Q, q_0, F, \delta)$, where Σ is a finite alphabet, Q is a finite non-empty set of states, $q_0 \in Q$ is an initial state, $F \subseteq Q$ is a set of final states, and $\delta : Q \times \Sigma \rightarrow Q$ is a state transition function. The state transition function δ can be extended to a mapping $\delta^* : Q \times \Sigma^* \rightarrow Q$ in the following inductive way: (*i*) $\delta^*(q, \lambda) = q$, for each state $q \in Q$, where λ is the null string, and (*ii*) $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$, for each state $q \in Q$, each letter $a \in \Sigma$, and each string $w \in \Sigma^*$. The finite automaton $M = (\Sigma, Q, q_0, F, \delta)$ is deterministic if $|\delta(q, a)| \leq 1$ for each $q \in Q$ and for each $a \in \Sigma$. Then M is called deterministic finite automata (shortly denoted by *DFA*).

In this work, Two learning algorithms are selected for solving Thai handwritten character recognition in this work. The first algorithm named RPNI (regular positive negative inference) was proposed to identify a DFA from a sample set of positive and negative examples. The main idea is to greedily merge states in order to find a solution that is always consistent with a given sample. The incompatibility constraint is used to prevent merging incompatible states that may lead to build an inconsistent DFA. A prefix tree acceptor consistent with a given sample is initially built to hold all information from the given sample. Pairs of states for merging are chosen in lexicographic order. The merging process may return temporary solution, which is nondeterministic. To reduce the non-determinism of temporary solution in learning process, a tree invariant property is maintained by considering two states to be merged; at least one of them is the root of a tree. The property of tree invariant is a sufficient condition for the determinization process to be finite and also helps implementing simple and fast [8]. In the best of cases, if a characteristic set of the language includes in the random sample then the proposed algorithm returns the correct target DFA. The RPNI algorithm is shown in Fig.1.

```
    ALGORITHM : RPNI
    Input : $S = (S+, S-)$
    Output : $M = (\Sigma, Q, q_0, F_A, F_R, \delta)$
1: $M \leftarrow$ BUILD_PTA($S+$)
2: $Red \leftarrow \{q_ÿ\}$
3: $Blue \leftarrow \{q_a : a \in \Sigma$ ō $Prefix(S+)\}$
4: While $Blue \neq \emptyset$ DO
5:         $q_b \leftarrow$ CHOOSE($Blue$)
6:         $Blue \leftarrow Blue - \{q_b\}$
7:         $M^* \leftarrow$ RECMERGE($M, q_r, q_b$) such that $q_r \in Red$
8:         IF COMPATIBLE($M^*, S$) THEN
9:             $M \leftarrow M^*$
10:            $Blue \leftarrow Blue \cup \{\delta(q, a) : q \in Red \wedge a \in \Sigma\}$
11:        ELSE
12:            $M \leftarrow$ PROMOTE($M, q_b$)
13:        ENDIF
```

```
14: ENDWHILE
15: RETURN  $M$
```

Figures 1. RPNI algorithm

The RPNI algorithm starts from constructing a prefix tree acceptor (function BUILD_PTA) consistent with a given sample *S*+ and return a DFA as an output. The set *Red and Blue* in this algorithm are a set of states which are selected for merging. The main state-merging loop begins with choosing a state q_b in set *Blue* by using the function CHOOSE. This function returns a smallest state in the set *Blue* in lexicographic-length order. The merge process is recursively performed to merge the states q_b into their previous states q_r in the *Red* by using the function RECMERGE. The compatibility constraint is considered to prevent leading to obtain inconsistent DFA. In RPNI algorithm, the function COMPATIBLE is used to check the compatibility constraint. If it returns TRUE then the merging q_b into q_r is allowed. The algorithm terminates when all state in the set *Blue* becomes empty.

The second learning algorithm we use in this work is EDSM algorithm. This algorithm is a state-merging algorithm with heuristic function. This is the main difference from RPNI algorithm that performs as a greedy algorithm. Merging every pair of states is considered the score of that merge as the number of strings that end in a same state if that merge is done. To do that the strings from *S*+ and *S*- have to be parsed. If by doing that merge a conflict arises (a negative string is accepted or a positive string is rejected) the score is –i. The merge with the highest score is chosen. The EDSM algorithm is shown in Fig. 2.

```
ALGORITHM : EDSM
Input : $S = (S+, S-)$
Output : $M = (\Sigma, Q, q_0, F_A, F_R, \delta)$
1:   $M$ ← BUILD_PTA($S+$)
2:   $Red$ ← $\{q_ÿ\}$
3:   $Blue$ ← $\{q_a : a \in \Sigma$ ō $Prefix(S+)\}$
4:   While $Blue \neq \emptyset$ DO
5:       promotion ← $FALSE$
6:       FOR $q_b$ Ŗ $Blue$ DO
7:           IF not promotion THEN
8:               $best$ ← -i
9:               $atleastonemerge$ ← $FALSE$
10:              FOR $q_r$ Ŗ $Red$ DO
11:                  $score$←EDSM_COUNT(RECMERGE($M,q_r,q_b$),$S$)
12:                  IF score >-i THEN
13:                      $atleastonemerge$ ←$TRUE$
14:                  ENDIF
15:                  IF score >$best$ THEN
16:                      $best$ ← $score$
17:                      $q_r^*$ ← $q_r$
18:                      $q_b^*$ ← $q_b$
19:                  ENDIF
20:                  IF not $atleastonemerge$ THEN
21:                      $M$ ← PROMOTE($M$, $q_b$)
22:                      $promotion$ ←$TRUE$
23:                  ENDIF
24:              ENDFOR
25:          ENDIF
26:      ENDFOR
27:      IF not promotion THEN
28:          $Blue$ ← $Blue$ – $\{q_b\}$
29:          $M$ ←RECMERGE($M$, $q_r^*$, $q_b^*$)
30:      ENDIF
31:  ENDWHILE
32:  RETURN  $M$
```

Figures 2. EDSM algorithm

2.2 Thai alphabet and handwritten character recognition

Thai alphabet has 44 consonant characters, 15 vowel symbols that combine into at least 20 vowel forms, and four tone marks. Thai characters are composed of circles, lines, curves, and zigzags. Examples of handwritten Thai characters, which are selected only 44 consonant characters for our experimentation in this work are shown in Fig. 3.

ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ	ช
ซ	ฌ	ญ	ฎ	ฏ	ฐ	ฑ	ฒ	ณ	ด
ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ
ฟ	ภ	ม	ย	ร	ล	ว	ศ	ษ	ส
ห	ฬ	อ	ฮ						

Figure 3. Thai characters used for experimentation

In process of handwriting recognition, an important one after the process of pre-processing is feature extraction. The feature extraction step is essential for efficient data representation and extracting meaningful features for later processing. In this work two types of features have been focused for the task, which are zoning density features and crossing count features. In case of zoning density features, this method can be considered as a partition of a binary image into M sub-images, named zones z_1, z_2, ..., z_M, each one providing information by considering its density. Crossing count feature is a number of transitions from background to foreground along hypothesis horizontal line h_1, h_2, , … h_n and vertical line v_1, v_2, …, v_m over the character image.

3. Implementation

Overall model of our implementation includes three stages: pre-processing, feature extraction, and classification. The pre-processing is primarily used to reduce variations of handwritten characters by resizing and thinning character image. The feature extraction step is used to extract meaningful features. These features can be viewed string data used in process of grammatical inference. The later step, a learning algorithm of GI is performed to output a grammatical representation in form of a deterministic finite automata. Our implementation is shown in Fig. 4.

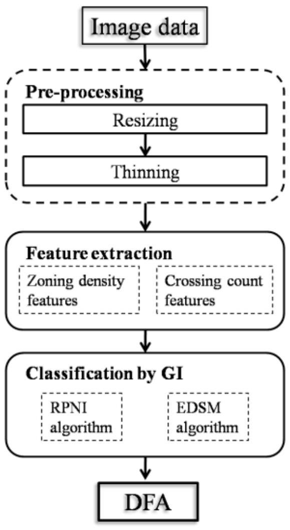

Figure 4. Our implementation

3.1 Data

There are Thai handwritten characters from 30 persons. Each person free-style wrote 44 different Thai characters with a digital pen to create an image data with 1024x1024 pixels. The total character images are 1320. An example of Thai handwritten character collected for this work is shown in Fig. 5.

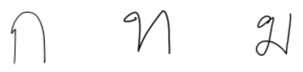

Figure 5. Some Thai handwritten characters.

3.2 Pre-processing

The pre-processing is used to reduce variations of handwritten characters images. The procedure of resizing is firstly performed to normalize the character image into 64x64 pixels. Then, the character image is converted into skeleton image (by the algorithm in [9]). That is a character image with 1 pixel in width.

3.3 Feature extraction

Two different feature extractions are used in this step. The zoning density features are extracted by segmenting the character image into 16 zones (z_1, z_2, …, z_{16}) with the same width and height. The number of foreground pixels is counted to represent for the density feature of each zones. The string data of this image data are represented as $z_1z_2…z_{16}$. An example of zoning density features of the first Thai character is shown in Fig. 6.

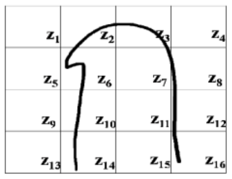

Figure 6. Zoning density features

The crossing count features are computed from 4 horizontal lines (h_1, h_2, h_3, h_4) and 4 vertical line (v_1, v_2, v_3, v_4) in our work. The string data of the image data is represented as $h_1h_2h_3h_4v_1v_2v_3v_4$. An example of crossing count features of the first Thai character is shown in Fig. 7.

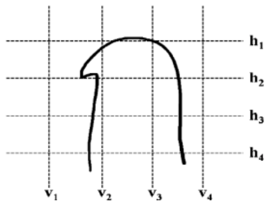

Figure 7. Crossing count features

3.4 Classification by GI

In this work, the RPNI algorithm is used for learning the recognizer in form of deterministic finite automata from given learning examples. The RPNI implementation is available in [10]. The correct rate is obtained by using 5-cross-validation.

4. Experimental results

The aim of this experimentation is to show that GI approach can be used for recognizing Thai handwritten characters. We have done some experiments in order to evaluate the learning algorithm. The description of the experiments is follows:

- We collect image data of each Thai character by using a digital pen with size 1024x1024 pixels from 30 different persons.These images are normalized by resizing into 64x64 pixels and use the thinning method to obtain skeleton images.
- We generate learning samples for each Thai character with 90, 180, 270, 360, 450, 540, 630, 720, 810, and 900 examples. For each sample of a specific Thai character, it consists of 30 examples from same Thai character and remainders from other Thai character by random.
- The obtained DFA from the learning is evaluated in term of correct rates by 5-cross-validation method. The implementation of learning DFA is performed by MATLAB GI Toolbox [11].

Table 1 shows the average of correct rate of the Thai handwritten character recognition with different size of samples by using RPNI algorithm. We found that the average correct rate of crossing count features is greater than the average correct rate derived from zoning density features. Moreover, we have show experimental results to compare the effective of RPNI and EDSM algorithm for solving Thai HCR. The results are shown in Table 2, which indicate that RPNI algorithm is more effective than EDSM algorithm for the recognition problem.

Table 1 Average correct rate of Thai HCR.

#examples	Average correct rate (%)	
	Zoning density feature	Crossing count feature
90	63.462526	65.155033
180	77.702844	78.633081
270	83.669707	88.024700
360	86.984023	87.993141
450	89.153667	90.195567
540	90.677619	91.832983
630	92.107351	93.859633
720	92.925247	94.028980
810	93.894202	94.314043
900	94.611584	95.609600

Table 2. Recognition results by five-cross-validation

Feature extraction	RPNI		EDSM	
	Average correct rate (%)	Average error rate (%)	Average correct rate (%)	Average error rate (%)
Zoning density features	94.6115	5.3884	94.4423	5.5561
Crossing count features	95.8621	4.1379	93.9286	6.0715

5. Conclusion

In this work, we have presented the off-line Thai handwritten character recognition by using grammatical inference. We have selected two learning algorithms (RPNI and EDSM) to solve the recognition problem. The results shows that both of learning algorithms can be used to effectively solve the problem. The average correct rate will be significantly higher when the numbers of examples increases for all Thai alphabet recognition. Moreover, the results of comparison have also been shown that the RPNI algorithm is more effective than EDSM algorithm for the recognition problem.

6. Acknowledgment

This work was supported by Faculty of Applied Science, King Mong's University of Technology North Bangkok under Grant No. 5542104.

7. References

[1] C. de la Higuera, *Grammatical inference: learning automata and grammars*, Cambridge University Press, 2010.
[2] E. M. Gold, "Language identification in the limit", *Information and Control*, Vol. 10, no. 5, 1967, pp. 447–474.
[3] E. M. Gold, "Complexity of automaton identification from given data", *Information and Control*, Vol. 37, 1978, pp. 302-320.

[4] J. Oncina and P. Garcia, “ Identifying regular languages in polynomial time”, *Advances in Structural and Syntactic Pattern Recognition*, *volume 5 of Series in Machine Perception and Artificial Intelligence*, 1992, pp. 99-102.
[5] R. Plamondon and S. Srihari, “On-line and off-line handwriting recognition: A comprehensive survey”, *IEEE Transactions on PAIN*, Vol. 22(1), 2000, pp. 63-84.
[6] P. Gader, J. Keller, R. Krishnapuram, J. Chiang, and M. Mohamed, "Neural and Fuzzy Methods in Handwriting Recognition", *IEEE Computer*, Vol. 30, No. 2, 1997, pp. 79-86.
[7] P.Phokharatkul, K.Sankhuangaw, S.Phaiboon, S.Somkuarnpanit, and C.Kimpan, “Off-Line Hand Written Thai Character Recognition Using Ant-Miner Algorithm”, *Transactions on ENFORMATIKA on Systems Sciences and Engineering*, Vol. 8, 2005, pp. 276-281.
[8] K. Lang, B. Pearlmutter, and R. Price, “Results of the Abbadingo one DFA learning competition and a new evidence-driven state-merging algorithm”, *Grammatical Inference Proceedings of International Colloquium on Grammatical Inference 1998*, pp. 1-12.
[9] J. S. Kwon, J. W. Gi, and E. K. Kang,”An enhanced thinning algorithm using parallel processing”, *Image Processing Proceedings 2001* Vol. 3, 7-10, 2001, pp. 752-755.
[10] H.I. Akram, C. de la Higuera, H. Xiao, and C. Eckert, “Grammatical Inference Algorithms in MATLAB”, *Grammatical Inference Proceedings of International Colloquium on Grammatical Inference 2010*, 2010, pp. 264-268.
[11] H. I. Akrm, and H. Xiao, “MATLAB GI Toolbox Licensed under the MIT license: http://code.google.com/p/gitoolbox/.