

# CpG-Discover: A Machine Learning Approach for CpG Islands Identification from Human DNA Sequence

Man LAN, Yu XU, Lin LI, Fei WANG, Ying ZUO, Yuan CHEN, Chew Lim TAN and Jian SU

**Abstract**— CpG islands (CGIs) play a fundamental role in genome analysis as genomic markers and tumor markers. Identification of potential CGIs has contributed not only to the prediction of promoters of most house-keeping genes and many tissue-specific genes but also to the understanding of the epigenetic causes of cancer. The most current methods for identifying CGIs suffered from various limitations and involved a lot of human intervention for search purpose. In this paper, we implement a HMM-based CGIs identification system, namely *CpG-Discover*. Experiments have been conducted on the EMBL human DNA database and in comparison with other widely-used tools. The controlled experimental results indicate that our system is a promising tool and has the capability of locating CGIs accurately. In addition, our system has significant differences from other tools in that it avoids the disadvantages of using sliding windows and it reduces the large amount of human intervention needed to search for or to combine potential CGIs (such as, the thresholds of initial density or distance seed). Therefore, given annotated training data set, our system has the adaptability to find other specific nucleotides sequences in DNA.

## I. INTRODUCTION

Typically, CpG denotes a pair (or subsequence) of nucleotides, where G (guanine) appears immediately after C (cytosine) along a DNA strand and “p” simply indicates that “C” and “G” are connected by a phosphodiester bond. The regions of genes containing a relatively high concentration of CpG pairs, are collectively referred to along a chromosome as CpG islands, which typically vary in length from a few hundred to a few thousand nucleotides long [1].

CGIs are very useful in genome mapping projects. They are often associated with the promoters of most house-keeping genes and many tissue-specific genes [2]. Due to their involvement in the regulation of gene expression, they can be used as gene markers. Moreover, methylation of promoter-related CGIs is very common in all types of cancer cells, thus the hypermethylated CGIs in promoter regions can be used as molecular tumor markers, which makes the early detection of cancer possible [3] [4]. Consequently, identification of potential CGIs helps not only to find the promoters

of many specific genes but also to find candidate regions for aberrant DNA methylation and therefore contributes to the understanding of the epigenetic causes of cancer.

CGIs can be identified experimentally [5] [6] or computationally [7] [8] [9] [10] [11] [12] [13] [14]. Note that although [12] has integrated additional information, such as DNA sequence properties, DNA structure, and epigenetic states, into computational methods to predict the strength of each CGI quantitatively, the sequence-based approach is still crucial to generate an initial genome-wide map of CGIs and has gained a lot of attention all along.

So far there are quite a number of programs developed for identifying CGIs from DNA sequence. We group them into two categories according to the main ideas these methods adopted. One approach is to employ the conventional sliding window technique for locating CGIs in DNA sequences [7]-[11], etc. Another approach is to iteratively find and cluster short CGIs based on the distance or density measures of CGIs [13] [14].

Unlike the above two approaches, in this paper, we present and implement a machine learning approach for the identification of CGIs. Our system, called *CpG-Discover*, is based on a Hidden Markov Model (HMM). When viewed as a time-series, the DNA sequence has the Markovian property of depending on just the previous state (nucleotide here), thus making it possible to model with a Hidden Markov Model [15]. The idea of our work is motivated by the fact that HMMs perform better than other methods in previous applications of sequence analysis and temporal pattern recognition, such as speech recognition, handwriting recognition, gesture recognition, part-of-speech tagging and biomedical named entity recognition, etc.

Therefore, the purpose of this paper is twofold. First is to present and implement a HMM-based system for the identification of CGIs (a more detailed description of the system framework is in Section III). Second is to explore the capability of this system by experimenting on EMBL human DNA database and in comparison with other tools.

The rest of this paper is structured as follows. Section II reviews the state-of-the-art methods for the identification of CGIs in DNA sequences. Section III presents the detailed information about this two-stage system framework. Section IV reports experiments including data corpus, performance evaluation, experimental results and discussion. Finally, Section V concludes this paper with future work.

Dr. Man LAN is with the Department of Computer Science and Technology, East China Normal University, No.500 Dong Chuan Road, Shanghai 200241, China and the Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (phone: 65-64082179; email: mlan@cs.ecnu.edu.cn or mlan@i2r.a-star.edu.sg). Yu XU, Fei WANG, Lin LI, Ying ZUO, and Yuan CHEN are with the Department of Computer Science and Technology, East China Normal University (emails: 10062130229, 10062130230, 10062130231, 10062130233, 10062130257@ecnu.cn). Prof. Chew Lim TAN is with the School of Computing, National University of Singapore, Singapore 117543 (phone: 65-65162900; email: tancl@comp.nus.edu.sg). Dr. Jian SU is with the Institute for Infocomm Research, 1 Fusionopolis Way, #21-01 Connexis, South Tower, Singapore 138632 (email: sujian@i2r.a-star.edu.sg).

## II. RELATED WORK

In recent years, there has been considerable work on the identification of CGIs from DNA sequences. In this paper, we group them into two categories according to the main ideas of these methods adopted.

The most widely used approach is to adopt the traditional sliding window technique for locating CGIs in DNA sequences, such as CpGPlot/CpGReport [7], CpGProD [8], CpGIS [9], CpGIE [10], CpGED [11], etc. Typically, for each window, a measure (for example, log-odds ratio) is computed. The windows that receive positive scores are potential CG-islands. Then the CGIs within intersecting windows could be merged to determine CGIs within the long sequence. This approach has the high capability of combining small CGIs and it is easy to implement. However, this approach has some built-in disadvantages: (1) it is not clear what window size to use in advance and thus the resulting number and length of CGIs identified depend on the window size and step size. If the window size is big, several short and loosely distributed CGIs might be clustered together to form a big one; (2) since the window is moved in only one direction, it may not correct the previous mistakes and not be able to locate CGIs accurately; (3) CGIs identified by those methods generally do not start and end with a CpG dinucleotide; (4) longer running time. Due to these built-in limitations, current algorithms using sliding windows rely on the ad hoc thresholds of length, CpG O/E ratio and G+C content early defined by Gardiner-Garden and Frommer [17].

Another approach is to iteratively combine neighboring CpG clusters whose physical distance or density measure of CGIs is below or above a pre-defined threshold according to CpGs' genomic property. Examples of this approach include CpGCluster [13] and CpGIE [14]. These methods are presented with the purpose of avoiding the drawbacks of sliding windows approach. Specifically, CpGCluster [13] is presented based on the physical distance between neighboring CpGs on the chromosome. It consists of two main steps: (i) a distance-based algorithm searches for clusters of CpGs in the chromosome sequence. (ii) a p-value is associated with each of these clusters, then only those clusters with a large enough statistical significance will be predicted as CGIs. The search and extension steps are iterated until all the CpG clusters in the sequence are identified. CpGCluster is much faster and more computationally efficient than the sliding windows approach. But several problems still exist in CpGCluster: (1) search results are dependent on the composition of the sequence scanned, i.e. a CGI identified in one sequence may be discarded when planted in another sequence with different composition. (2) Low prediction sensitivity. The CGIs detected by CpGCluster are usually short fragments with high GC content and CpG observed/expected (*o/e*) ratio. This is the reason why the CGIs predicted by CpGCluster exhibit lower degree of overlap with Alu and higher degree of overlap with PhastCons. Furthermore, CpGIF [14] is presented based on the similar idea of using CGIs' genomic property in an iterative way. It first searches

regions with high CpG density, named as *seeds*. The seeds are then extended and clustered into the final CGIs. In the search and extension steps, the GC content and CpG *o/e* ratio are calculated. Therefore, CpGIF also has the tendency to find CGIs with high CpG *o/e* ratio and high CpG density. On the other hand, this built-in limitation decreases its capability of finding CGIs with length less than 210bp. That is the reason why CpGIF excludes CGIs with a short length [14].

## III. SYSTEM FRAMEWORK

Our system, called *CpG-Discover*, is based on Hidden Markov Model and implemented in C++ under Visual Studio 2005, including a Windows command-line application and a Common Gateway Interface (CGI) program. The program is available on request to the authors. The system consists of two stages: (1) HMM-based identification; (2) post-processing including identification refinement and error correcting. We first use HMM for identification. Then we use three post-processing modules to refine and correct identification.

### A. HMM: Description

A Hidden Markov Model (HMM) is a model where a sequence of observations is generated in addition to the Markov state sequence. It is a latent variable model in the sense that only the observation sequence is known while the state sequence remains "hidden". Generally, a HMM is characterized by the following:

- $Q$  is a set of states, and  $N$  is the number of states in  $Q$  (we either know or guess this number);
- $M$  is the number of distinct observation symbols per state (obtained from the training set), i.e. the discrete alphabet size.
- The state transition probability distribution  $A = \{a_{ij}\}$ , where transition probability  $a_{ij}$  is the probability of transitioning from state  $i$  to state  $j$  for  $i, j \in Q$ ;
- The emission probability distribution  $B = \{b_j(k)\}$ , for each state,  $j$ , and each symbol,  $k$ , where  $b_j(k)$  is the probability of seeing symbol  $k$  in state  $j$ ,  $1 \leq j \leq N$ ,  $1 \leq k \leq M$ . The sum of all emission probabilities at a given state must equal 1, that is,  $\sum_k b_j = 1$  for each state  $j$ .
- The initial state probability distribution  $\pi = \{\pi_i\}$ , where  $\pi_i$  is the initial probability of state  $i$ .

There are three canonical problems associated with HMM:

- *Evaluation*: Given the parameters of the model, compute the probability of a particular output sequence, and the probabilities of the hidden state values given that output sequence. This problem is solved by the *forward-backward* algorithm.
- *Decoding*: Given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This problem is solved by the *Viterbi* algorithm.
- *Learning*: Given an output sequence or a set of such sequences, find the most likely set of state transition

and output probabilities. In other words, discover the parameters of the HMM given a data set of sequences. This problem is solved by the *Baum-Welch* algorithm.

Typically, in the case of CGIs, there are two important questions ([16]):

- given a short sequence, is it from a CpG island or not?
- given a long sequence, can we find all of the CGIs within it?

The second question is crucial since once we solve the second question, the model learned can be used to address the first question.

## B. Building HMM

The application of HMM in our system is quite straightforward. In the case of CGIs identification, the HMM model contain eight states, i.e.  $A^+$ ,  $C^+$ ,  $G^+$ ,  $T^+$ ,  $A^-$ ,  $C^-$ ,  $G^-$ ,  $T^-$ . In these labels, “+” and “-” mean the nucleotides being in and out of a CpG island, respectively. The state transition probabilities between the above eight states can be determined using training sequences containing known transitions between CGIs and non-CGIs.

Let  $s = s_1s_2\dots s_L$  be a path of states that generates an observed sequence of symbols  $o = o_1o_2\dots o_L$ . In a Markov chain, the path through the chain for  $o$  is unique. With a hidden Markov model, however, we do not know for sure which states were visited in generating the sequence. For example, the states  $C^+$  and  $C^-$  both emit the symbol  $C$ . The emission probabilities are  $b_{A^+}(A) = 1$ ,  $b_{A^+}(C) = 0$ ,  $b_{A^+}(G) = 0$ ,  $b_{A^+}(T) = 0$ ,  $b_{A^-}(A) = 1$ ,  $b_{A^-}(C) = 0$ ,  $b_{A^-}(G) = 0$ ,  $b_{A^-}(T) = 0$ , and so on.

In this paper, HMMs are learned on the EMBL database. A HMM model is learned in the following steps (we take the sequence with ID *AB016898* for example):

1) *Parameters Initialization*: For each training sample sequence, we start by initializing the values for the parameters based on the sliding windows with length 2, which count the frequency from the first state to the second state, rather than choosing arbitrary values for the parameters. For example, given a training sequence, to estimate the state transition probability  $a_{A^+C^+}$  from state  $A^+$  to state  $C^+$ , we first count the occurrence of neighboring “ $A^+C^+$ ”, i.e.  $N_{A^+C^+}$ , and then count the total occurrence of dinucleotide, i.e.  $N$ . Thus the probability  $a_{A^+C^+}$  is calculated by  $N_{A^+C^+}/N$ . In addition, to make the initialization stage simple, here the probability of emitting a symbol at a state is either 0 or 1, although in general, emission probabilities need not be zero-one. Moreover, for each state  $i$ , its initial probability is calculated by counting the occurrence of each state in the sequence.

After the parameters initialization, we output the state transition probability distribution, the emission probability distribution and the initial state probability for each training sequence. Figure 1 shows the initialized three probabilities for sequence *AB016898*.

A:	A+	C+	T+	G+	A-	C-	T-	G-
A+	0.211765	0.252941	0.258824	0.276471	0	0	0	0
C+	0.136076	0.212025	0.373418	0.275316	0	0	0.003165	0
T+	0.133072	0.238748	0.426614	0.199609	0	0	0	0.001957
G+	0.080808	0.279461	0.434343	0.205387	0	0	0	0
A-	0	0	0.003378	0	0.378378	0.179054	0.175676	0.263514
C-	0	0	0	0	0.344828	0.270936	0.098522	0.285714
T-	0	0	0	0	0.271277	0.191489	0.250000	0.287234
G-	0	0	0	0.003876	0.244186	0.232558	0.263566	0.255814

B:	A	C	T	G
A+	1	0	0	0
C+	0	1	0	0
T+	0	0	1	0
G+	0	0	0	1
A-	1	0	0	0
C-	0	1	0	0
T-	0	0	1	0
G-	0	0	0	1

pi:	A	C	T	G
	0.076271	0.140946	0.132917	0.227921
	0.132025	0.090544	0.11463	0.083854

Fig. 1. Initialized three probabilities for human DNA sequence *AB016898*

2) *Parameters Approximation*: In the second step, we adopt the *Baum-Welch* algorithm to re-estimate the three probabilities of the model based on training sequence. The output of the above initialization step serves as the input of the *Baum-Welch* algorithm. Then the expected frequencies given the model and the observations are computed by weighting the observed transitions by the probabilities specified in the current model. The expected frequencies so obtained are then substituted for the old parameters and we iterate until there is no improvement. On each iteration we improve the probability of being observed from the model until some limiting probability is reached. This iterative procedure is guaranteed to converge on a local maximum performance measure.

After the parameters approximation, we obtain the three modified probabilities for each training sequence. Figure 2 shows the resulting three probabilities for human DNA sequence *AB016898* after the *Baum-Welch* approximation.

A:	A+	C+	T+	G+	A-	C-	T-	G-
A+	0.188552	0.315898	0.492861	0.001904	0.001873	0.002327	0.001742	0.001843
C+	0.020588	0.003854	0.003579	0.961471	0.002774	0.009888	0.001950	0.002897
T+	0.030973	0.001504	0.006008	0.583997	0.001371	0.380282	0.001417	0.001448
G+	0.707376	0.005018	0.007692	0.278452	0.001986	0.003232	0.001429	0.001814
A-	0.001164	0.001179	0.001130	0.001189	0.527872	0.001279	0.413723	0.059464
C-	0.001696	0.950838	0.005025	0.001684	0.001806	0.042887	0.001535	0.001529
T-	0.001119	0.001110	0.001139	0.001166	0.105822	0.001191	0.036547	0.858905
G-	0.001111	0.001153	0.001115	0.001193	0.451884	0.001256	0.502341	0.046947

B:	A	C	T	G
A+	0.221656	0.177676	0.008228	0.595440
C+	0.004683	0.768113	0.228531	0.001673
T+	0.024103	0.007082	0.969909	0.001905
G+	0.120934	0.008867	0.002815	0.870384
A-	0.505645	0.390507	0.069323	0.037525
C-	0.018577	0.701196	0.268166	0.015062
T-	0.190110	0.057188	0.536307	0.219395
G-	0.073088	0.247989	0.257246	0.424677

pi:	A	C	T	G
	0.999851	0.001000	0.001000	0.001061
	0.001000	0.001081	0.001002	0.001004

Fig. 2. Modified probabilities for human DNA sequence *AB016898* after *Baum-Welch* Approximation

3) *Voted Parameters*: So far, for each training sequence, we obtain three modified probabilities. In order to integrate these probabilities from each training sequence, we take a voted averaged technique. That is, for the three probabilities, each is normalized by the length of each training sequence.

The weight of each training sequence can be expressed as:

$$w_i = \frac{n_i}{N} \quad (1)$$

where  $n_i$  is the number of nucleotides in the  $i_{th}$  training sequence and  $N$  is the total number of nucleotides in the training data set. After using such a vote technique, we finally obtain the model parameters for HMM.

Figure 3 shows the final voted model parameters for HMM on a training data set containing 90 DNA sequences.

A:	A+	C+	T+	G+	A-	C-	T-	G-
A+	0.223315	0.201971	0.174550	0.310620	0.022882	0.037661	0.019601	0.016400
C+	0.266680	0.251143	0.313751	0.074593	0.022952	0.046541	0.022647	0.008693
T+	0.093047	0.252023	0.244971	0.345885	0.009289	0.018434	0.025103	0.018248
G+	0.184609	0.270320	0.181478	0.232564	0.039905	0.027194	0.017787	0.053143
A-	0.032332	0.015401	0.015351	0.034236	0.268350	0.167911	0.149636	0.323784
C-	0.024263	0.039224	0.014586	0.008857	0.305190	0.248570	0.301264	0.065047
T-	0.020973	0.027361	0.020936	0.037157	0.096579	0.241488	0.211400	0.351107
G-	0.047992	0.030339	0.018663	0.028742	0.257471	0.206958	0.173654	0.243182

B:	A	C	T	G
A+	0.835810	0.060719	0.050107	0.056365
C+	0.038207	0.900631	0.026556	0.037606
T+	0.062429	0.048578	0.837637	0.054356
G+	0.034371	0.045118	0.033139	0.890372
A-	0.869894	0.043399	0.045242	0.044465
C-	0.049887	0.882633	0.028273	0.042207
T-	0.072931	0.038865	0.855158	0.036046
G-	0.038219	0.030675	0.040569	0.893537

$\pi_i$ :	A	C	T	G
	0.074225	0.100490	0.030921	0.118112
	0.077036	0.249243	0.069228	0.287745

Fig. 3. Final voted model parameters for HMM on a training data set with 90 DNA sequences

### C. Application of HMM

After training HMM, the parameters of the model can be used to find the most likely sequence of hidden states that could have generated a given output sequence. That is, for each test sequence, we can find the most likely sequence of states, then the path cross “+” states will be identified as a CpG island. This decoding problem is solved by using *Viterbi* algorithm.

### D. Post-processing

The tag sequences identified by the *Viterbi* algorithm as above result from a completely mathematical operation and they may not be completely accurate. In order to further improve the performance, we develop three post-processing modules for further refinement as follows. These modules are to refine the identified CGIs according to the genomic properties of CGIs, such as GC content, CpG fraction and length threshold [17] and correct the errors caused by the inconsistency.

1) *Combination of contiguous short CGIs*: It is frequently observed that a small quantity of nucleotides (mostly varying from 1 to 15) within true CGIs in a sequence have been falsely identified as non-CpG islands nucleotides. In order to solve this problem, we set a minimal distance threshold between two neighboring identified CGIs. Practically, if the distance between two neighboring identified CGIs is less than 20, the system would change the states of these non-CpG island nucleotides between them and merge them into one CpG island. After that, this newly extended CpG island will be further examined by the following two modules.

2) *Density Cutoff of CpG Island*: Biologically, CpG island contains high G/C (or C/G) ratio ( $o/e$ ). Typically, this ratio should be higher than 60%. However, due to inappropriate combination in the step mentioned above and the incorrect identification of the system, the identified CGIs may not meet this requirement. Therefore, the system applies a mathematical density cutoff ( $o/e \geq 60\%$ ). If the identified CGIs do not meet this requirement, the system change the states of nucleotides from CGIs to non-CGIs.

3) *Length Cutoff of CpG island*: Besides the above density constraint, our system also applies a length cutoff to filter these recognized CGIs less than 140 bp according to the biological definition of CpG island. Thus the states of these recognized CGIs with length less than 140 will be changed as non-CpG island.

## IV. EXPERIMENTS

Note that in principle our system can be trained on any annotated CpG islands database. However, in this paper we only report the experimental results on one selected database from the web site of the European Bioinformatics Institute (EBI). The data can be downloaded from <ftp://ftp.ebi.ac.uk/pub/databases/cpgisle>.

### A. Data Corpus

The database *embl73hum* contains all predicted human CGIs in the current release of the EMBL database. Only the sequences contained in the files which correspond to the human division were analyzed. There are 233,004 sequences in this database and 61,051 sequences have 142,325 CGIs (mean is 2.33). Among them, the first 1,955 sequences whose ID names start with “AB” are adopted in this paper. After removing null sequences and noisy sequences, the resulting data set contains 1,900 sequences. In order to make the experiments reasonable, we randomly selected out 1,000 training sequences and 100 test sequences under the same distribution guidance. That is, the numbers of CGIs per sequence in training data set and test data set both follow the original distribution in whole data set.

### B. Performance Evaluation

The performance of the system is measured by comparing the predicted nucleotide value (island or non-island) with the true island value for each nucleotide along the test sequence, including *Accuracy* (a single value that summarizes the proportion of nucleotides that have been correctly identified either as islands or as non-islands), *Sensitivity* ( $S_n$  for short, the proportion of island nucleotides that have been correctly predicted as islands), *Specificity* ( $S_p$  for short, the proportion of non-island nucleotides that have been correctly predicted as non-islands), *Positive Predictive Value* ( $PPV$  for short, the proportion of predicted island nucleotides that are actually islands), *Performance Coefficient* ( $PC$  for short, a single scalar value that summarizes  $S_n$  and  $PPV$  as a measure of global accuracy) and the *Correlation Coefficient* ( $CC$  for short, a single scalar value that summarizes  $S_n$  and  $S_p$  as a measure of global accuracy).

### C. Experimental Results

We conduct a series of experiments under various subsets of training data set. Specifically, different HMMs have been trained on different training sets which vary by the number of training examples. The percentage of training subset ranges from 10% to 100% (divided in 10% intervals). The main purpose of using various training sets with different training examples is to examine whether they have effects on the performance of HMMs. Figure 4 depicts the different performance measures in terms of different percentages of training data set. Each curve in the figure represents a specific performance measure as shown in the figure, where PPV, PC and CC denote *Positive Predictive Value*, *Performance Coefficient* and *Correlation Coefficient* respectively (for each curve, see the legend at the top of Figure 4).

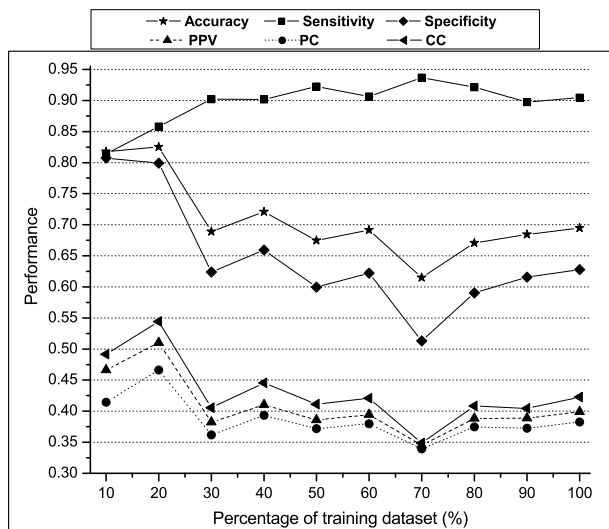


Fig. 4. Different performance measures on different subsets of training data set

Except *sensitivity*, although the trends of the other five measures with regard to various size of training data set cannot be summarized in one sentence but it is clearly observed that they have shown consistent performance with respect to each other as the size of training data set increases. That is, they all reach a peak on the same training subset. On the other hand, the trend of *sensitivity* measure is distinctive in that it generally slowly increases as the size of training data set grows and reaches a peak at a specific training subset where the other five measures arrive at the lowest value. The performance of each HMM is closely related to the annotated training data set on which the model has been learned. Therefore, based on the above observations, we cannot draw a conclusion that there is relationship between different performance measures and the size of training data set.

However, when we carefully take a look at these measures, the experimental results indicate that our system based on

HMM is quite promising. First, the *sensitivity* measure of our system reaches a maximal value of 94% and its averaged value is about 90%. Even though the trend of *specificity* measure is not consistent with that of *sensitivity* measure, they both can reach a comparable performance above 80% together. As a single value that summarizes the proportion of nucleotides that have been correctly identified by system either as islands or as non-islands, the *accuracy* of our system is also very encouraging and it can be above 80%.

### D. Comparison with Other Tools

In order to further assess the performance of our system, we also randomly select several sequences from the test examples and report the results in comparison with the other widely-used tools mentioned before. We choose two widely-used tools, i.e. CpGIE and CpGIF, for two reasons. Firstly, the two perform better than other three tools, namely, CpGReport, CpGProD and CpGIS based on a previous comparative experiment in [14]. Secondly, they provide online or downloadable software, which makes the comparison possible and reasonable. Table I lists the results of several test sequences by using our system in comparison with CpGIE and CpGIF. The first three rows list the sequence ID, the length of sequence and the position of CGIs (the start and end positions per CpG island are separated by symbol “..”).

Although a lot of test sequences have been compared, only several of them are listed in Table I due to their typical representation in this comparison. Generally, several interesting findings can be observed as follows.

- It is difficult for the three tools to accurately identify the boundaries of two neighboring CGIs if the number of non-CGIs nucleotides between them is quite small, for example, not more than 100. It is the case in examples such as sequence *AB014544*, *AB017019* and *AB017365*.
- The average length of the CGIs returned by CpGIF and CpGIE is much longer than those of CGIs detected by our system. In the case of *AB000275*, our system identified the two CGIs while CpGIE and CpGIF both combine them into one long CGI sequence. This is due to the higher capability of them in combining short CGIs.
- Our system locates CGIs more accurately. In the case of *AB000276*, *AB001915* and *AB020631*, the three tools identify all CGIs. However, the outputs of our system are close to the actual CGIs.

Note that for CpGIE and CpGIF, there are several parameters to tune for identification and each set of parameter will result in different performance. It is not possible to design an optimal set of parameters for every test sequence. Generally, according to [10] and [14], the search criteria of CpGIE and CpGIF are: length  $\geq 200$ bp, CpG density  $\geq 60\%$ , and CpG o/e ratio  $\geq 0.60$ .

### V. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we present and implement an HMM-based CGIs identification system, namely *CpG-Discover*. The ex-

Sequence ID	AB000275	AB000276	AB000277	AB001915	AB111100	AB014544	AB017019	AB017365	AB020631
Length(BP)	1740	2393	3347	1633	4186	4933	2180	3851	5834
true positive CGIs	274..524 1472..1684	436..713 1836..2301	308..975 1390..1667 2790..3255	185..642	51..285	48..537 552..1094 1171..1642 1688..2696	47..381 39..930	47..289 293..681 691..1066 1155..1730	49..484
CpG-Discover	471..598 1483..1740	482..667 1843..2294	1445..1531 1569..1621 2823..3248	195..616		1..2834 4277..4797	1..896	1..1821 2096..2406	1..495
CpGIE	210..1739	341..774 1767..2392	1..1083 1295..1728 2721..3346	89..694		1..2834	1..984	1..1829 2020..2235	1..505
CpGIF	3..1734	338..784 1890..2409	10..2567 2826..3345	178..768		23..2949	6..1044 1757..1959	11..2387	6..519

TABLE I

COMPARISON OF OUR SYSTEM WITH CpGIE AND CpGIF FOR IDENTIFYING CGIs. ACCORDING TO [10] AND [14], THE SEARCH CRITERIA OF CpGIE AND CpGIF ARE: LENGTH $\geq$  200BP, CpG DENSITY $\geq$  60%, AND CpG O/E RATIO  $\geq$  0.60.

periments on the EMBL human DNA database showed that our system is promising for discovering potential CGIs. Moreover, in comparison with other tools, our system has the capability of locating CGIs accurately. In addition, our system has significant differences from other tools in that it avoids the disadvantages of using sliding windows and it reduces the large amount of human intervention needed to search for or combine potential CGIs (such as, the thresholds of initial density or distance seed). Therefore, given an annotated training data set, our system has the adaptability to automatically find other specific nucleotides sequences in DNA.

An interesting future work is to experiment by using hybrid approaches. Current systems often rely only on a single technique and use processing pipelines with multiple stages. Whenever more than one system is used in parallel, rule-based or constraint-based post-processing stages deal with refinement of predicted candidates and error correcting, post-processing must include a form of decisions making in case of conflicting results, which can be implemented by using meta-learning. The simplest form is majority voting, a widely-used scheme in machine learning. Therefore, if the system uses an ensemble of several different models trained on different sequence composition or different approaches with majority voting, it seems to enable the system to properly adapt to different sequence composition properties.

## REFERENCES

- [1] Antequera F, Bird A. "CpG islands as genomic footprints of promoters that are associated with replication origins". *Curr Biol.*, 9(17): 661-667, 1999.
- [2] Larsen F, Gundersen G, Lopez R, Prydz H. "CpG islands as gene markers in the human genome". *Genomics*, 13(4):1095-1107, 1992.
- [3] Bird, A. "DNA methylation patterns and epigenetic memory". *Genes Dev.*, 16(1):6-21, 2002.
- [4] Herman, JG; Baylin, SB. "Gene silencing in cancer in association with promoter hypermethylation". *N Engl J Med*, 349(21):2042-2054, 2003.
- [5] Heisler, LE, et al. "CpG Island microarray probe sequences derived from a physical library are representative of CpG Islands annotated on the human genome". *Nucleic Acids Res.*, 33(9):2952-2961, 2005.
- [6] Illingworth, R, et al. "A novel CpG island set identifies tissue-specific methylation at developmental gene loci". *PLoS Biol.*, 6(1):e22, 2008.
- [7] Rice, P, et al. "EMBOSS: the European Molecular Biology Open Software Suite". *Trends Genet.*, 16(6):276-277, 2000.

- [8] Ponger, L; Mouchiroud, D. "CpGProD: identifying CpG islands associated with transcription start sites in large genomic mammalian sequences". *Bioinformatics*, 18(4):631-633, 2002.
- [9] Takai, D; Jones, PA. "Comprehensive analysis of CpG islands in human chromosomes 21 and 22". *Proc Natl Acad Sci*, 99(6):3740-3745, 2002.
- [10] Wang, Y; Leung, F. "An evaluation of new criteria for CpG islands in the human genome as gene markers". *Bioinformatics*, 20(7):1170-1177, 2004.
- [11] Luque-Escamilla PL, Martinez-Aroza J, Oliver JL, Gomez-Lopera JF, Roman-Roldan R. "Compositional searching of CpG islands in the human genome". *Phys Rev E Stat Nonlin Soft Matter Phys*, 71:061925, 2005.
- [12] Bock, C, et al. "CpG island mapping by epigenome prediction". *PLoS Comput Biol*, 3:e110, 2007.
- [13] Hackenberg, M, et al. "CpGcluster: a distance-based algorithm for CpG-island detection". *BMC Bioinformatics*, 7:446, 2006.
- [14] Ye Sujuan, Asai Asaithambi, and Yunkai Liu. "CpGIF: an algorithm for the identification of CpG islands". *Bioinformatics*, 2(8): 335C338, 2008.
- [15] Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, vol.7, no.2, February 1989.
- [16] Jiawei Han and Micheline Kamber. "Data Mining : Concepts and Techniques". 2006.
- [17] Gardiner-Garden M, Frommer M. "CpG islands in vertebrate genomes". *J Mol Biol*, 196(2):261-282, 1987.