Learning Constrained Edit State Machines

Laurent Boyer, Olivier Gandrillon, Amaury Habrard, Mathilde Pellerin, Marc Sebban

Learning Constrained Edit State Machines*

Laurent Boyer[1], Olivier Gandrillon[2], Amaury Habrard[3], Mathilde Pellerin[2], and Marc Sebban[1]
[1] *Laboratoire Hubert Curien,Université Jean Monnet, Université de Lyon*
{laurent.boyer,marc.sebban }@univ-st-etienne.fr
[2] *Centre de Génétique Moléculaire et Cellulaire,Université de Lyon*
olivier.gandrillon@cgmc.univ-lyon1.fr,mathilde.pellerin@gmail.com
[3] *Laboratoire d'Informatique Fondamentale de Marseille, Aix-Marseille Université*
amaury.habrard@lif.univ-mrs.fr

Abstract

Learning the parameters of the edit distance has been increasingly studied during the past few years to improve the assessment of similarities between structured data, such as strings, trees or graphs. Often based on the optimization of the likelihood of pairs of data, the learned models usually take the form of probabilistic state machines, such as pair-Hidden Markov Models (pair-HMM), stochastic transducers, or probabilistic deterministic automata. Although the use of such models has lead to significant improvements of edit distance-based classification tasks, a new challenge has appeared on the horizon: How integrating background knowledge during the learning process? This is the subject matter of this paper in the case of (input,output) pairs of strings. We present a generalization of the pair-HMM in the form of a constrained state machine, where a transition between two states is driven by constraints fulfilled on the input string. Experimental results are provided on a task in molecular biology, aiming to detect transcription factor binding sites.

1 Introduction

The computation of similarities between data structured in the form of strings (*e.g.* words of a language), trees (*e.g.* XML documents) or graphs (*e.g.* biological molecules) requires a particular attention. Indeed, these data can not be directly represented by numerical feature vectors allowing the use of standard metrics, that implies the definition of specific similarity measures. In the string case, for example, among the existing similarity measures (see [3] for an experimental comparison of string distances), the edit distance [18] is probably the most used. The edit distance between two instances is the cost of the best sequence of edit operations that changes a data into another one. Typical edit operations are symbol deletion, insertion and substitution, and to each of them is assigned a cost. The success of this approach has lead to the design of edit distance-based kernels for dealing more efficiently with structured data such as convolution kernels or mapping kernels [8, 17, 19].

However, tuning the edit costs can constitute a difficult task in many applications. Supervised learning methods have then been used during the last decade for learning the parameters of string edit distances. For example, some edit kernel-based methods suggest to directly learn the edit costs calling on an optimization procedure (*e.g.* a gradient descent) that needs both positive and negative instances [16]. Other approaches consider a probabilistic framework (see [14] or [15]) and learn a probability matrix associating a probability to each possible edit operation over the alphabet. These methods are based on the maximum likelihood principle and allow us to deal with applications that suffer from the lack of available negative examples (*e.g.* in language processing).

When there is no reason that the cost of a given edit operation changes according to the context where the operation occurs, the previous models are sufficient and very efficient from an accuracy and algorithmic point of view. However, learning a single matrix of edit costs can be viewed as insufficient in some applications, particularly when the operation plays a part more or less important in the string transformation according to its location. For instance, in molecular biology, it is com-

*This work was funded by the french ANR projects Marmota ANR-05-MMSA-0016 and Bingo2 ANR-MDCO-14 and IST Programme of the European Community, under the PASCAL2 Network of Excellence, IST-2007-216886. This publication only reflects the authors' views.

mon knowledge that the probability to change a symbol $s \in \{a, c, g, t\}$ into another one depends on its membership to a transcription factor binding site. Another example, given in [13], states that the characters in an author's first name after the first character are more likely to be deleted than the first character. To deal with such situations, *non-memoryless* approaches have been proposed in the literature in the form of probabilistic state machines that are able to take into account the string context. They are mainly based on pair-Hidden Markov Models (pair-HMM) [2, 6], probabilistic deterministic automata [1], or stochastic transducers [7]. The string context is described in each state by a statistical distribution over the edit operations. Despite their great interest, these approaches are generative models, meaning that they do not enable the use of constraints or features of the input strings. Thus, they do not allow the incorporation of background knowledge during the learning process. For instance, we would like to be able to integrate prior knowledge such as: "*it is (almost) impossible to have two identical successive vowels in a french word*". To overcome this drawback, one can learn conditional (or discriminative) models, for instance in the form of conditional random fields [11], that allow us to incorporate arbitrary features on the input strings. An adaptation has even been proposed in [13] in the context of the edit distance. However, this approach has two main limitations: First, it requires the use of positive *and* negative pairs of examples of matches of strings. Moreover, this model does not provide for each state a matrix of edit parameters that reduces the possibility of knowledge extraction describing string contexts.

To recap, if each of the mentioned approaches has its own advantages, no one possesses all the skills required to have an efficient learned edit model: (i) adaptability to the context, (ii) ability to take into account background knowledge (iii) understandability of the induced model. The objective of this paper is to fill this gap. We consider a new type of state machine, where the choice of a transition between two states is driven by constraints fulfilled by the input string. This model leads to the improvement of the expressiveness of standard markovian approaches. We assessed the relevance of our method with an new experimental evaluation in molecular biology showing that the set of matrices of edit parameters learned by our model can constitute an invaluable knowledge for this public health problem.

The rest of the paper is organized as follows: In Section 2, we introduce some definitions and notations about edit distance and edit similarities. Section 3 is devoted to the presentation of our constrained state machine, and the way for learning the hidden parameters of that model. In Section 4, we study its expressiveness in comparison with a pair-Hidden Markov Model. Finally, Section 5 presents our experimental study.

2 Definitions and Notations

In this section, we give some definitions and notations that will be useful to present our constrained edit state machine. First, we suppose that strings are defined over a finite alphabet Σ. $\lambda \notin \Sigma$ is the empty symbol. In the following, letters in lowercase denote symbols of $\Sigma \cup \{\lambda\}$ while those in uppercase represent strings built from Σ. As previously mentioned, the core of our approach is based on the learning of the parameters of the string edit distance. In its classical formulation, the string edit distance is evaluated by computing the minimal set of operations (insertions, deletions, substitutions) allowing us to transform a string X into a string Y. Learning the parameters of an edit distance requires the use of an inductive principle. In the context of probabilistic machines, the maximization of the likelihood is often used. In this paper, we follow the same idea that explains why we are interested in learning string edit similarities in a probabilistic context rather than learning a *true* edit metric[1].

In our approach, we aim to learn a conditional (or discriminative) model that takes into account information about the input string X. Therefore, the similarity between two strings X and Y will be assessed from the estimation of $p(Y|X)$ corresponding to the probability of generating Y given an input string X. To model this probability in an edit context, one has to define a statistical distribution δ over the edit operations used to transform the strings. For instance, $\delta(b|a)$ corresponds to the probability of emitting an output symbol b of Y given an input character a of X. If $a = \lambda$, the operation denotes an insertion; If $b = \lambda$ the operation is a deletion; The operation $\delta(\lambda|\lambda)$ is not allowed. To learn a conditional distribution over the edit operations, the δ function must fulfill the following condition [14]:

$$\forall a \in \Sigma, \sum_{b \in \Sigma \cup \{\lambda\}} \delta(b|a) + \sum_{b \in \Sigma} \delta(b|\lambda) = 1. \qquad (1)$$

We now focus on the computation of the probability $p(Y|X)$ from the edit probabilities $\delta(b|a)$. Let us first introduce the notion of probabilistic edit script which corresponds to the set of edit operations allowing us to transform X into Y.

[1] When probabilities are used instead of edit costs, the edit similarity obtained does not satisfy some standard properties like symmetry or the triangular inequality.

Definition 1 *An edit script $e = e_1 \cdots e_n$ is a sequence of edit operations $e_i = (b_i|a_i)$ allowing the transformation of a string X into a string Y. A probabilistic edit script is an edit script s.t. its probability $\pi_s(e)$ is the product of the probabilities of the operations involved in the script: $\pi_s(e) = \Pi_{i=1}^{n} \delta(e_i)$. We define by $S(Y|X)$ the set of scripts allowing the emission of Y given X.*

Definition 2 *The probability $p(Y|X)$ of generating a string Y given an input X is the sum of the probabilities of all the edit scripts transforming X into Y.*
$$p(Y|X) = \sum_{e \in S(Y|X)} \pi_s(e).$$

Using standard dynamic programming techniques, $p(Y|X)$ can be computed in $O(|X| \times |Y|)$. Given $p(Y|X)$, one can then compute an edit similarity between X and Y.

Definition 3 *The edit similarity between two strings X and Y, conditionally to X, is defined as: $sim(X, Y|X) = -\log p(Y|X)$.*

In the following section, we present our constrained state machines (CSM) whose parameters are the edit probability matrices. We explain how to learn them and how to compute $p(Y|X)$ from a CSM using dynamic programming procedures.

3 Constrained State Machines

3.1 General Framework

Our constrained state machine (CSM) is close to a pair-HMM (in fact, we will show in Section 4 that it is a strict generalization). Indeed, as for pair-HMMs, a CSM is composed of a set of states linked by transitions, and for each state a probability distribution over the edit operations is defined. The main difference between a CSM and a pair-HMM is the following: The use of a transition between two states in a pair-HMM is not conditioned by any constraint, while in our model it depends on constraints satisfied on the input string X. Each constraint can express conditions that can be global (*e.g.* `there is an even number of letters in the string`) or local around the letter currently studied in the edit process (*e.g.* `is the current symbol followed by two letters` *a*?).

In our framework, we assume that each constraint takes the form of boolean functions; However, in order to have only one possible active transition between two states, we will impose that exactly one boolean function is true at any time.

Definition 4 *A constraint c is a finite set of boolean functions $c_k : \Sigma^* \times \mathbb{N} \rightarrow \{true, false\}$, $1 \leq k \leq m$. Each of these functions is defined over an input string X and a position t in X. Therefore, $c_k(X, t)$ denotes the boolean value associated by the k^{th} function of c to the input string X at a position t ($0 \leq t \leq |X|$). A constraint must respect the following property: for any $X \in \Sigma^*$, $t \in \mathbb{N}$, there exists exactly one c_k s.t. $c_k(X, t)$ is true. In other words, at any moment t, only one boolean function of c is satisfied.*

In pair-HMMs, the probabilities of outgoing transitions from a given state are learned without any prior knowledge. Whatever the pair of characters treated in the state, each outgoing transition has a priori a non-null probability to be used. In order to improve the expressiveness of pair-HMMs, we aim to incorporate knowledge during the learning process in the form of constraints. In our model, an outgoing transition from a state will be used if and only if a boolean function $c_k(X, t)$ of a given constraint c is satisfied at the instant t over the string X. To achieve this task, we assign to each state a constraint, or in other words, the outgoing transitions from a state will be the different boolean functions c_k of a constraint c. In the following, c_{q_i} will denote the constraint assigned to state q_i.

Definition 5 *A Constrained State Machine (CSM) is a tuple $\langle \Sigma, Q, \mathcal{C}, T, \delta, \pi \rangle$ where Σ is a finite alphabet, Q is a set of states, $\mathcal{C}$ is a set of constraints and:*

- *$T : Q \times \mathcal{C} \times Q \rightarrow [0, 1]$ defines the probability of a transition. $T(q_j|q_i, c_{q_i,k}(X, t))$ will denote the probability of going to state q_j given that we are currently in state q_i and that the constraint $c_{q_i,k}(X, t)$ is satisfied. For a given state q_i and a constraint $c_{q_i,k}$, the outgoing transitions must fulfill the following condition:*
$$\sum_{q_j \in Q} T(q_j|q_i, c_{q_i,k}) = 1. \quad (2)$$
- *δ is a family of $|Q|$ matrices. Each element δ_{q_i} is the matrix of edit probabilities of state q_i. For each state q_i, δ_{q_i} must respect the condition (1).*
- *$\pi : Q \rightarrow [0, 1]$ is the initial probability function which must satisfy the following statistical condition:*
$$\sum_{q \in Q} \pi(q) = 1. \quad (3)$$

Since we assign to each state one constraint, the fact that exactly one boolean function is true at any time allows us to ensure to define conditional distributions over the input strings.

3.2 Computation of $p(Y|X)$ from a CSM

The computation of the probability $p(Y|X)$ can be achieved by the *forward* function described below. Let $X = x_1 \cdots x_T$ and $Y = y_1 \cdots y_V$ (x_0 and y_0 denote the empty string). The forward function is defined recursively to compute the probability of a prefix of an output string given the prefix of an input string. This calculation can be done from any state $q \in Q$. To simplify the notations, $c_{q,k}(X,t)$ will be noted in the following $c_q(x_t)$ without mentioning the k^{th} boolean function that is satisfied at the t^{th} position of X.

$$\begin{array}{l}\alpha_q(y_0|x_0) = \pi(q),\\ \alpha_q(y_v|x_t) = \\ (\sum_{q' \in Q} \alpha_{q'}(y_{v-1}|x_{t-1}) \cdot \delta_{q'}(y_v|x_t) \cdot T(q|q', c_{q'}(x_t)))_{v \geq 1, t \geq 1}\\ +(\sum_{q' \in Q} \alpha_{q'}(y_v|x_{t-1}) \cdot \delta_{q'}(\lambda|x_t) \cdot T(q|q', c_{q' \in Q}(x_t)))_{t \geq 1}\\ +(\sum_{q' \in Q} \alpha_{q'}(y_{v-1}|x_t) \cdot \delta_{q'}(y_v|\lambda) \cdot T(q|q', c_{q'}(x_t)))_{v \geq 1}\end{array}$$

Note that the *forward* function takes into account the three possible edit operations (substitution, deletion, insertion). We can also use the so-called *backward* function, calculated from the suffixes of X and Y.

$$\begin{array}{l}\beta_q(y_{V+1}|x_{T+1}) = 1,\\ \beta_q(y_v|x_t) = \\ (\delta_q(y_v|x_t) \cdot \sum_{q'} T(q'|q, c_q(x_t)) \cdot \beta_{q'}(y_{v+1}|x_{t+1}))_{v \leq V, t \leq T}\\ +(\delta_q(\lambda|x_t) \cdot \sum_{q'} T(q'|q, c_q(x_t)) \cdot \beta_{q'}(y_v|x_{t+1}))_{t \leq T}\\ +(\delta_q(y_v|\lambda) \cdot \sum_{q'} T(q'|q, c_q(x_t)) \cdot \beta_{q'}(y_{v+1}|x_t))_{v \leq V}\end{array}$$

In the computation of the *forward* and *backward* functions, we have to take into account all the possible states. This implies that the dynamic programming version of these functions can be computed in $O(|Y| \times |X| \times |Q| \times f(X))$, where $f(X)$ is an upper bound of the complexity needed for computing the boolean functions of the constraints. This complexity is supposed to be polynomial in $|X|$ and generally linear. Finally, $p(Y|X)$ can be easily computed by using one of the previous two functions:
$p(Y|X) = \sum_{q \in Q} \alpha(y_V|x_T) = \sum_{q \in Q} \pi(q)\beta(y_1|x_1)$.

3.3 Learning the Hidden Parameters

To learn the parameters of our model, we need to learn one matrix δ_{q_i} for each state q_i of the CSM. Moreover, we also need to estimate the probabilities of the transitions and the probabilities $\pi(q_i)$ of entry in the CSM. To achieve these tasks, we use the well known *Expectation-Maximization* (EM) algorithm [5] that aims to iteratively learn the hidden parameters of a probabilistic model by maximizing the likelihood of a learning sample. In our approach, we will use pairs of *(input,output)* strings as learning examples. The algorithm runs in two steps. First, an *expectation* step estimates the expectations of the number of times each edit operation or transition has been used over the learning set of pairs. This is done by using the α and β procedures previously defined. Then, a *maximization* step is achieved by normalizing the previous expectations under the statistical conditions (1), (2) and (3). We describe each of these two steps below.

Expectation step

The expectation step consists of estimating the expectations of the number of times each edit operation or transition has been used on the learning set of pairs.

The expectations of the number of times a state is initial are stored in variables γ_{π_q}:

$$\gamma_{\pi_q} = \frac{\beta_q(y_1|x_1).\pi(q)}{p(Y|X)}.$$

The expectation of an edit operation $(b|a)$ occurring in a state q is accumulated in the variable $\gamma_{\delta_q(b|a)}$:

$$\gamma_{\delta_q(y_v|x_t)} = \frac{\alpha_q(y_{v-1},x_{t-1}) \cdot \delta_q(y_v|x_t) \cdot \sum_{q'} T(q'|q,c_q(x_t)) \cdot \beta_{q'}(y_{v+1},x_{t+1})}{p(X,Y)}.$$
$$\gamma_{\delta_q(\lambda|x_t)} = \frac{\alpha_q(y_v,x_{t-1}) \cdot \delta_q(\lambda|x_t) \cdot \sum_{q'} T(q'|q,c_q(x_t)) \cdot \beta_{q'}(y_{v+1},x_{t+1})}{p(X,Y)}.$$
$$\gamma_{\delta_q(y_v|\lambda)} = \frac{\alpha_q(y_{v-1},x_t) \cdot \delta_q(y_v|\lambda) \cdot \sum_{q'} T(q'|q,c_q(x_t)) \cdot \beta_{q'}(y_{v+1},x_{t+1})}{p(X,Y)}.$$

The expectation of the transitions are stored in variables $\gamma_{T(q'|q,c_{q,k})}$:

$$\begin{array}{l}\gamma_{T(q'|q,c_{q,k})} = \\ \quad \frac{\alpha_q(y_{v-1}|x_{t-1}) \cdot \delta_q(y_v|x_t) \cdot \sum_{q'} T(q'|q,c_{q,k}(x_t)) \cdot \beta_{q'}(y_{v+1}|x_{t+1})}{p(Y|X)}\\ \quad + \frac{\alpha_q(y_{v-1}|x_t) \cdot \delta_q(y_v|\lambda) \cdot \sum_{q'} T(q'|q,c_{q,k}(x_t)) \cdot \beta_{q'}(y_{v+1}|x_{t+1})}{p(Y|X)}\\ \quad + \frac{\alpha_q(y_v|x_{t-1}) \cdot \delta_q(\lambda|x_t) \cdot \sum_{q'} T(q'|q,c_{q,k}(x_t)) \cdot \beta_{q'}(y_{v+1}|x_{t+1})}{p(Y|X)}.\end{array}$$

Note that, for each transition from q to q', there is as many variables as there are boolean functions in the constraint assigned to q.

Maximization step

The maximization step aims to find an optimal normalization under the statistical condition (1). We do not prove the optimality of this normalization in this paper, but the principle of the proof relies on a direct adaptation of the results obtained for the HMM in [5]. We describe below how to normalize the auxiliary variables presented in the previous section to obtain a conditional distribution:

- For each state q, the initial probability is evaluated by: $\pi(q) = \frac{\gamma_{\pi_q}}{\sum_q \gamma_{\pi_q}}$.
- The transition probabilities are computed by the following quantity for each starting state q, for each constraint $c_{q,k}$ and for each arrival state q':
$$T(q'|q, c_{q,k}) = \frac{\gamma_{T(q'|q,c_{q,k})}}{\sum_{q'} \gamma_{T(q'|q,c_{q,k})}}.$$
- We now consider the probabilities of the edit operations in each state q. Let

$N = \sum_{a\in\Sigma\cup\{\lambda\}} \sum_{b\in\Sigma\cup\{\lambda\}} \delta_q(b|a)$ and $N(a) = \sum_{b\in\Sigma\cup\{\lambda\}} \delta_q(b|a)$.

For each $a \in \Sigma$ and each $b \in \Sigma \cup \{\lambda\}$:

$$\delta_q(b|a) = \frac{\gamma_{\delta_q(b|a)}}{N(a)} \times \frac{N - N(\lambda)}{N}.$$

For each $b \in \Sigma$, we have also:

$$\delta_q(b|\lambda) = \frac{\gamma_{\delta_q(b|\lambda)}}{N(\lambda)}.$$

4 Expressiveness of CSMs vs pair-HMMs

A pair-HMM is a tuple $\langle \Sigma, Q, T, \delta, \pi \rangle$ with: Σ a finite alphabet, Q a finite set of states, $T : Q \times Q \rightarrow [0, 1]$ the probability of each transition, $\delta : Q \times \Sigma \cup \{\lambda\} \times \Sigma \cup \{\lambda\} \rightarrow [0, 1]$ the matrices of edit probabilities for each state and $\pi : Q \rightarrow [0, 1]$ the initial probability function. T, δ and π must fulfill these statistical conditions:

$$\forall q_i \in Q, \sum_{q_j \in Q} T(q_j|q_i) = 1 \tag{4}$$

$$\sum_{(a,b)\in(\Sigma\cup\{\lambda\})\times(\Sigma\cup\{\lambda\})\setminus\{\lambda,\lambda\}} \delta(b|a) = 1 \tag{5}$$

$$\sum_{q_i \in Q} \pi(q_i) = 1 \tag{6}$$

The probability of a pair of strings can be evaluated by using the classical forward and backward functions.

Note that from Equation 5, in its classical formulation, the edit matrices of pair-HMMs define joint distributions. However, so far, the δ_{q_i} ($\forall q_i \in Q$) we used in our CSM described *conditional* distributions over the edit operations. This property is ensured by the M-step of the EM algorithm that fulfills the statistical condition (1) of page 2. Following the same strategy as that of [15] and [14] in the context of stochastic transducers, we could also define a pair-HMM with conditional distributions in each state, and then compare it with our CSM. Then, the statistical condition (5) is replaced by condition (1). In a symmetric way, we would only have to slightly modify the M-step to infer joint matrices δ_{q_i} in our CSM to be able to compare it with a standard joint pair-HMM.

Let us now study the expressiveness of CSM. Before the presentation of our first result, we introduce the notion of regular constraint.

Definition 6 *A regular constraint is a constraint such that the result of each of its boolean functions can be represented by the membership to a regular language. In other words, each of these functions can be defined by a finite state machine.*

Theorem 1 *A CSM with conditional (resp. joint) matrices δ_{q_i} is a strict generalization of a pair-HMM with conditional (resp. joint) matrices δ_{q_i}.*

Proof 1 *First, we show that any pair-HMM can be converted in an equivalent CSM. Let $A = \langle \Sigma, Q, T, \delta, \pi \rangle$ a pair-HMM, we build a CSM $C = \langle \Sigma, Q, \{c\}, T_c, \delta, \pi \rangle$ based on the same structure. The unique constraint $c = \{c_1\}$ has one unique boolean function $c_1 : \Sigma^* \times \mathbb{N} \rightarrow$ true and is assigned to every state. For any states q_i, q_j, we define $T_c(q_j|q_i, c_{q_i,1}) = T(q_j|q_i)$. By construction, A and C have the same structure and the same parameters, thus one can easily check that they define the same distribution.*
Second, consider a CSM calling on constraints that can not be modeled by a regular language (e.g. $c_k(X, t)$ is true if the prefix of size t of X belongs to a non rational language such as the context-free language $\{a^n b^n | n > 0\}$). By definition, these constraints can not be represented by a finite state machine and thus such a CSM can not be equivalently represented by a pair-HMM. □

The previous theorem states that a CSM allows us to deal with non regular constraints, that is not possible with pair-HMMs.

On the other hand, one can wonder if we could represent a regular constraint in a pair-HMM by adding additional states. We are going to show that this transformation is not always possible with Proposition 1. When it is feasible, we claim that the built pair-HMM has a more complex structure in terms of both states and transitions, leading to an increase of the number of parameters we have to assess during the learning process.

Proposition 1 *CSMs define a more general class of distributions than pair-HMMs.*

Proof 2 *By Theorem 1, we already know that any*

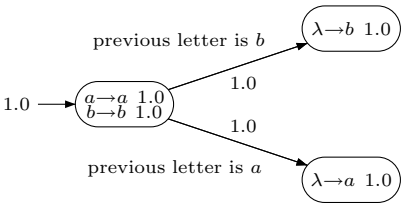


Figure 1. Example of a CSM defining conditional distributions not definable by any pair-HMM. In each state, we only indicate the edit operations with non zero probability.

pair-HMM is a special case of CSM. Let us consider the CSM of Figure 1 with a simple regular constraint on the letter previously read. This model defines two conditional distributions such that $p(aa|a) = p(bb|b) = 1$. *We will show that we can not build an equivalent pair-HMM with conditional matrices* δ_{q_i}[2]. *Each string pair admits a set of 5 possible edit scripts, we have respectively for* $(aa|a)$ *and* $(bb|b)$*:*
$S(aa|a) = \{[(a|a)(a|\lambda)], [(\lambda|a)(a|\lambda)(a|\lambda)], [(a|\lambda)(\lambda|a)(a|\lambda)], [(a|\lambda)(a|\lambda)(\lambda|a)], [(a|\lambda)(a|a)]\}$,
$S(bb|b) = \{[(b|b)(b|\lambda)], [(\lambda|b)(b|\lambda)(b|\lambda)], [(b|\lambda)(\lambda|b)(b|\lambda)], [(b|\lambda)(b|\lambda)(\lambda|b)], [(b|\lambda)(b|b)]\}$.

Suppose that there exists a pair-HMM $\langle \Sigma, Q, T, \delta, \pi \rangle$ *that can combine these scripts to obtain the desired distribution. This would imply that:*

$$\sum_{q \in Q} \pi(q)\beta_q(aa|a) = \sum_{q \in Q} \pi(q)\beta_q(bb|b) = 1. \qquad (7)$$

Now, suppose that there exists $q_i \in Q$ *with* $\pi(q_i) > 0$ *and* $\beta_{q_i}(aa|a) < 1$*, then*

$$\begin{aligned} p(aa|a) &= \textstyle\sum_{q \in Q} \pi(q)\beta_q(aa|a) \\ &= \pi(q_i)\beta_{q_i}(aa|a) + \textstyle\sum_{q \in Q \setminus \{q_i\}} \pi(q)\beta_q(aa|a) \\ &\leq \pi(q_i)\beta_{q_i}(aa|a) + \textstyle\sum_{q \in Q \setminus \{q_i\}} \pi(q) \quad (0 \leq \beta_q(aa|a) \leq 1) \\ &< \textstyle\sum_{q \in Q} \pi(q) = 1. \quad (\beta_{q_i}(aa|a) < 1 \quad \texttt{and condition (6)}) \end{aligned}$$

This implies that $p(aa|a) < 1$ *which contradicts the target distribution. So, for every state* q_i *with* $\pi(q_i) > 0$ *we have* $\beta_{q_i}(aa|a) = 1$*. Moreover, from Equation 7:*

$$\textstyle\sum_{q \in Q} \pi(q)(\beta_q(aa|a) - \beta_q(bb|b)) = 0.$$

Then for any q_i *such that* $\pi(q_i) > 0$, $\beta_{q_i}(aa|a) = \beta_{q_i}(bb|b) = 1$*. Thus,* δ_{q_i} *must have one edit operation among the first operations of the edit scripts for each of the two string pairs. Each of these two operations must have a probability equal to one to ensure to have the* β_{q_i} *equal to one. No insertion operation is possible otherwise condition (1) would be violated by summing up to two. The only possibility is then to have exactly one operation conditioned by a (i.e.* $(a|a)$ *or* $(\lambda|a)$*) and exactly one conditioned by b (i.e.* $(b|b)$ *or* $(\lambda|b)$*). Then, whatever the two edit operations chosen for* q_i*, it follows from the edit scripts that the next operation is* $(a|\lambda)$ *for* $(aa|a)$ *and* $(b|\lambda)$ *for* $(bb|b)$*. These two insertion operations must be of probability one in order to maintain the* β_{q_i} *equal to one. They can't be in the same state otherwise, as mentioned before, condition (1) is violated. Then, they must be in two different states which implies to add two different transitions from* q_i*. Each of these two transitions must be of probability one to ensure* $p(aa|a) = p(bb|b) = 1$*, which is impossible according to condition (4).* □

[2]We recall that pair-HMMs with conditional edit matrices must respect condition (1) of page 2 for the δ_{q_i} matrices, and conditions (4) and (6) of page 5 for the transitions and the initial states.

5 Experiments

5.1 Scientific Context

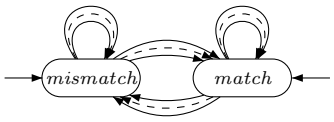
In this section, we propose to assess the relevance of our CSM on a task in molecular biology. We propose to consider the problem of detecting Transcription Factor Binding Sites (TFBS) in sequences of promoters of orthologous genes, *i.e.* genes having the same function and related by descent from a common ancestor. Here, our objective is to show that using our CSM, we are able to (i) take into account contextual information, (ii) integrate some background knowledge and (iii) infer a stochastic model *a posteriori* understandable by the experts of the domain. It has been demonstrated [4] that TFBS are under evolutionary selection, that means that they should have evolved much more slowly than other non-coding sequences. This difference in evolution speed can be observed by comparing sequences of orthologous genes between sufficiently distant species. In terms of edit distance, this means that such TFBS should be at a closer edit distance than other regions. Moreover, one can assume that the edit probabilities involved in a given TFBS do not differ a lot from another binding site. This first background knowledge provides us a very useful information on the structure of the CSM we will have to build (see next section). TFBS are located within the promoters regions of eukaryotic genes [20]. In the present work, we decided to consider as the promoter region the 500 base pairs (bp) located 5' of the Transcription Start Site (TSS) [10]. We focused on a list of orthologous genes from human and mouse extracted by the `BioMart` mining tool[3] from the `Ensembl` database. We obtained the promoter sequences of those genes by querying the newly released database `SQUAT` [12] and using additional information (including the promoter regions of the corresponding genes) through `DBTSS`[4]. This gave us a list of 13520 pairs of 500 bp of orthologous promoter sequences.

5.2 Experimental Setup

Using the previous background knowledge on TFBS, we decided to build a CSM (see Figure 2(a)) with two states: One for describing the TFBS, called *match*, and one for the other regions of the sequence, called *mismatch*. Note that the number of TFBS in a given promoter sequence is relatively small in general. Thus, a

[3]`http://www.ensembl.org/biomart/martview/b0b02309a47c99f8ad5602a6ef398ce5`

[4]`http://dbtss.hgc.jp`, release hg18 and mm8 for human and mouse sequences respectively



(a) CSM that models the search for TFBS.

	λ	a	c	t	g
λ		0.0	0.2	0.4	0.2
a	1.2	17.6	29.7	24.5	26.2
c	0.8	24.9	20.7	22.0	30.8
t	1.2	26.1	27.4	16.5	28.0
g	0.8	23.2	32.3	22.9	20.0

(b) Values learned for the *mismatch* state.

	λ	a	c	t	g
λ		0.0	0.9	1.3	0.9
a	5.3	80.9	3.1	2.4	5.2
c	2.6	2.2	86.0	3.3	2.8
t	5.8	2.8	5.5	79.9	2.9
g	2.9	3.9	3.2	2.2	84.7

(c) Values learned for the *match* state.

Figure 2. CSM and learned edit probabilities matrices, the probabilities were multiplied by 100 for sake of clarity.

non constrained machine, like a two state pair-HMM, would not naturally focus on the TFBS regions since their low frequency will not be helpful to maximize the likelihood of a learning sample. This leads us to define relevant constraints in order to force the model to concentrate on potential TFBS regions. That explains why we will not compare our CSM with other standard methods. In order to estimate reasonable constraints between those states, we explored some characteristics of known TFBS, as found in the `TRANSFAC Suite Databases`[5] and `MATCH` [9]. We observed that the core region of TFBS is often made from a small number of symbols. This new information drives us to use a constraint that takes into account the time we stay in a state of the CSM. More precisely, we decided to use a set of 30 boolean constraints c_k ($k = 1..30$), where c_k means "*Are we in the current state for k times?*" (except for $k = 30$, for which the constraint is "*Are we in the current state for more than 30 times?*"). We learned this CSM over the alphabet $\Sigma = \{a, c, g, t\}$, with the EM-based algorithm of Section 3, from a learning set of 10,000 pairs.

5.3 Result Analysis

The objective of this section is to interpret the results of our CSM in this molecular biology context. Figure 3 shows the evolution of the probability to leave a state according to the time spent in this current state. The dotted curve describes the probability to go from state *match* to state *mismatch*, the solid one represents the opposite. We can make the following remarks:

1. First, the convexity of the dotted curve shows that small (*i.e.* < 4) and large (*i.e.* > 27) TFBS are not highly probable. Indeed, in this case, the probability to leave the *match* state is greater than 0.2. This confirms an hypothesis in molecular biology stating that the core region of a TFBS is at least composed of 4 or 5 symbols while remaining a relatively small sequence of symbols.
2. Between the previous two bounds, the probability to stay in the *match* state is high (> 0.9). We can even note that the average size of a TFBS seems to be about 15 symbols that constitutes a new interesting knowledge in this area.
3. The solid curve is a monotonic increasing function. This means that once we are in state *mismatch*, the probability to leave this state continually increases. However, contrarily to the previous curve, the waiting time before leaving does not follow any regularity. This confirms the biological hypothesis that regions outside of TFBS have evolved much more quickly leading to an almost random distribution over the alphabet Σ.

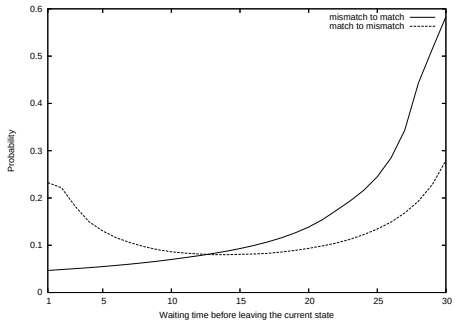


Figure 3. Probabilities to leave *match* and *mismatch* states.

This last remark is confirmed by Table 2(b) that describes the edit probabilities in the *mismatch* state learned after 15 iterations of EM. Indeed, we can note that the mutations are almost randomly distributed. Inversely, Table 2(c) shows that more than 80 percents of the symbols in a potential TFBS are preserved. More interestingly, we can note that symbols a and t are more subject to mutations than c and g. Finally, beyond all these biological considerations, we have verified the quality of our learned CSM from a machine learning point of view. Note that in the domain of TFBS detection, very few labeled data are available. Indeed, even

[5] `http://www.biobase-international.com/pages/` release 2008.3

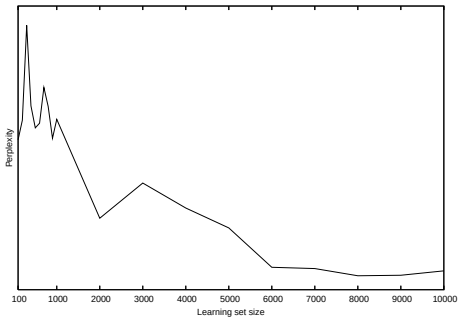


Figure 4. Evolution of the perplexity on a test set.

the experts in biology cannot provide TFBS locations. Therefore, estimations of true/false rates wouldn't be significant. Then, in order to evaluate our model, we calculated the perplexity on a test set composed of 3,520 pairs of sequences according to the learning set size (from 100 to 10,000). Figure 4 clearly shows that the use of an increasing number of learning pairs leads to the decrease of the perplexity. Moreover, it shows that our algorithm converges.

6 Conclusion

In this paper, we have introduced *constrained state machines* which are able to integrate background knowledge to compute an edit similarity between pairs of *(input,output)* strings. This is as far as we know the first approach proposing to take into account both the string context with states and background knowledge on the input strings by learning a conditional probabilistic model. We think that our model offers new challenges in many applications where the need of similarity measures over strings is needed. Indeed, we have shown that using constraints, we are able to characterize new properties that can not be modeled by classical pair-HMMs or memoryless transducers. The experiments carried out on a molecular biology task has shown the good behavior of our approach. To improve the interest of our model, we plan to use machine learning techniques to automatically learn the set $\mathcal{C}$ of constraints. Another perspective is to study the adaptation of our model to tree-structured data.

References

[1] M. Bernard, J.-C. Janodet, and M. Sebban. A discriminative model of stochastic edit distance in the form of a conditional transducer. In *Proceedings of ICGI'06*, pages 240–252, 2006.

[2] M. Bilenko and R. J. Mooney. Adaptive duplicate detection using learnable string similarity measures. In *ACM SIGKDD'03*, pages 39–48. ACM, 2003.

[3] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IJCAI-03 Workshop on Information Integration on the Web*, pages 73–78, 2003.

[4] M. Das and H. Dai. A survey of dna motif finding algorithms. *BMC Bioinformatics*, 8 Suppl 7(S21), 2007.

[5] A. Dempster, M. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc*, B(39):1–38, 1977.

[6] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis*. Cambridge University Press, 1998. Probabilistic models of protein and nucleic acids.

[7] J. Eisner. Parameter estimation for probabilistic finite-state transducers. In *Proceedings of ACL'02*, pages 1–8, Philadelphia, USA, July 2002.

[8] D. Haussler. Convolution kernels on discrete structures. Technical report, Univ. of Santa Cruz, 1999.

[9] A. Kel, E. Gößling, I. Reuter, E. Cheremushkin, O. Kel-Margoulis, and E. Wingender. Matchtm : A tool for searching transcription factor binding sites in dna sequences. *Nucleic Acids Res*, 36:3576–3579, 2003.

[10] M. Koudritsky and E. Domany. Positional distribution of human transcription factor binding sites. *Nucleic Acids Res*, 36:6795–6805, 2008.

[11] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML'01*, pages 282–289, 2001.

[12] L. Leyritz, S. Schicklin, S. Blachon, C. Keime, C. Robardet, J. Boulicaut, J. Besson, R. Pensa, and O. Gandrillon. Squat: a web tool to mine human, murine and avian sage data. *BMC Bioinformatics*, 2008.

[13] A. McCallum, K. Bellare, and P. Pereira. A conditional random field for discriminatively-trained finite-state string edit distance. In *Proceedings of UAI'2005*, pages 388–400, 2005.

[14] J. Oncina and M. Sebban. Learning stochastic edit distance: application in handwritten character recognition. *Pattern Recognition*, 39(9):1575–1587, 2006.

[15] S. Ristad and P. Yianilos. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[16] H. Saido, J. Vert, and T. Akutsu. Optimizing amino acid substitution matrices with a local alignment kernel. *BMC Bioinformatics*, 7:246, 2006.

[17] K. Shin and T. Kuboyama. A generalization of haussler's convolution kernel - mapping kernel. In *Proceedings of ICML'08*, pages 944–951, 2008.

[18] R. Wagner and M. Fischer. The string-to-string correction problem. *J. of the ACM*, 21:168–173, 1974.

[19] C. Watkins. *Advances in Large Margin Classifiers*, chapter Dynamic alignment kernels, pages 39–50. MIT Press, 2000.

[20] G. Wray, M. Hahn, E. Abouheif, J. Balhoff, M. Pizer, M. Rockman, and L. Romano. The evolution of transcriptional regulation in eukaryotes. *Mol Biol Evol*, 20:1377–1419, 2003.