Seed-driven Learning of Position Probability Matrices from Large Sequence Sets*

Jarkko Toivonen[1], Jussi Taipale[2], and Esko Ukkonen[1]

1 **Department of Computer Science, University of Helsinki, Helsinki, Finland**
`jarkko.toivonen@cs.helsinki.fi`
2 **Department of Biosciences and Nutrition, Karolinska Institutet, Stockholm, Sweden**
`jussi.taipale@ki.se`
3 **Department of Computer Science, University of Helsinki, Helsinki, Finland**
`esko.ukkonen@cs.helsinki.fi`

Abstract

We formulate and analyze a novel seed-driven algorithm SeedHam for PPM learning. To learn a PPM of length ℓ, the algorithm uses the most frequent ℓ-mer of the training data as a seed, and then restricts the learning into the ℓ-mers of training data that belong to a Hamming neighbourhood of the seed. The PPM is constructed from background corrected counts of such ℓ-mers using an algorithm that estimates a product of ℓ categorical distributions from a (non-uniform) Hamming sample. The SeedHam method is intended for PPM learning from large sequence sets (up to hundreds of Mbases) containing enriched motif instances. A variant of the method is introduced that decreases contamination from artefact instances of the motif and thereby allows using larger Hamming neighbourhoods. To partially solve the motif orientation problem in two-stranded DNA we propose a novel seed finding rule, based on analysis of the palindromic structure of sequences. Test experiments are reported, that illustrate the relative strengths of different variants of our methods, and show that our algorithm outperforms two popular earlier methods.

Availability and implementation: A C++ implementation of the method is available from `https://github.com/jttoivon/seedham/`

Contact: `jarkko.toivonen@cs.helsinki.fi`

1998 ACM Subject Classification I.2.6 Learning, G.2.1 Combinatorics, I.5.1 Models, J.3 Life and Medical Sciences

Keywords and phrases motif finding, transcription factor binding site, sequence analysis, Hamming distance, seed

Digital Object Identifier 10.4230/LIPIcs.WABI.2017.25

1 Introduction

Position probability matrix (PPM), introduced by Stormo et al [13, 12], is a simple probabilistic model for motifs in biological sequences. PPM represents a product of mutually independent categorical variables, and it is currently the most popular representation, for example, of the DNA motifs for binding sites of transcription factors (collected in motif databases such as Transfac [18] and Jaspar [11]) as well as of motifs in RNA and in protein sequences. Besides PPMs, several other representations of sequence motifs have been

* This work was supported by EU FP7 project SYSCOL (UE7-SYSCOL-258236).

proposed, most of them being various generalizations of the consensus sequence of a motif. Motif representations and their discovery from sequence data is surveyed, e.g., in [9, 14].

In this paper we formulate and analyze a seed-driven algorithm SeedHam for PPM learning. Seed-driven means that a selected seed sequence is used as a starting point of a search that constructs the PPM by analyzing the segments of training data which are similar to the seed. This is in contrast with the well-known alignment method to learn a PPM of length ℓ. This method takes a collection of ℓ-mers, that are supposed to be (somehow verified) instances of the motif, and aligns the ℓ-mers which gives ℓ columns of bases A, C, G, and T. The frequency of each base on each column is counted which gives a position frequency matrix of size $4 \times \ell$. When normalized column-wise, this matrix gives the probability matrix θ for the motif. Assuming that the collection of ℓ-mers is an unbiased sample from the distribution of the motif instances and assuming mutual independence of different positions of the motif, this very simple procedure gives an unbiased estimate of motif distribution.

While verified samples of motif instances are not easily available, there is currently lots of sequence data in which instances of motif(s) are enriched within longer background sequences. Such sequence sets are produced, for example, by high-throughput SELEX [5, 8, 15] or by variants of ChIP-seq [10]. It is possible to learn PPMs from such sequences, by analyzing their over-represented ℓ-mers that are considered instances of the motif. This is what we do in our seed-driven approach to learning PPMs.

We assume that the training data D for learning a PPM is a collection of one or several DNA sequences that contain a relatively high number of instances of the target motif X. Different instance variants should be present in D according to the probability distribution to be learned but the exact locations of the instances within D are not known. The rest of D outside the motif instances is assumed neutral background (although in practice it may contain instances of some other motifs).

Our method locates plausible motif instances in D using the following rule. A most frequent ℓ-mer s of D is taken as the *seed*. Here the motif length ℓ is a user-given constant. Then s as well as the ℓ-mers of D within a short Hamming distance d from s are taken as instances of the target motif X. Using background corrected counts of such ℓ-mers we estimate a PPM that represents X. The aforementioned simple alignment method cannot be used as such because the restriction to a Hamming neighbourhood yields a non-uniform sample of the target distribution.

Our algorithm, called SeedHam, does not contain an iterative search. It is therefore very fast and can learn from large high-throughput sequence sets. An early version of SeedHam method was used for learning PPMs from HT-SELEX sequence sets [5]. Here we present a complete formal definition of a general version of the algorithm as well as extensions for correcting self-overlaps and how to decide the orientation of the motif in two-stranded case. The 'seed-and-wobble' procedure [2] independently developed for the analysis of protein microarray data is analogous to (restricted) SeedHam.

Learning a PPM is complicated by two issues of combinatorial nature. First, if motif X is strongly self-similar, then D may contain lots of artefact instances that overlap the true instances of X [3]. We give an instance elimination technique and associated background correction that decrease contamination from artefacts. Second, in two-stranded DNA the orientation of instances has to be decided. If the Hamming neighbourhoods with radius d of s and its reverse complement $\overline{s}$ do not intersect, we get a heuristic rule for deciding the orientation. We show that if the so-called *palindromic index* of the seed is large enough, then the neighbourhoods become separate for a given d. Algorithm SeedHam+ finds a seed that has high-enough palindromic index at the expense of having sub-maximal count.

The experimental tests illustrate relative strengths of different variants of our methods. We demonstrate the algorithms' capacity to relearn the PPM from simulated data that contains implanted instances of the motif represented by the PPM. The experiments show that the accuracy depends on the Hamming radius d such that the optimal d increases when motif length ℓ increases but decreases when the number of training instances increases.

We compared our methods experimentally with two earlier algorithms, DREME [1] and DECOD [4]. Both are seed-driven, discriminative PPM learning methods that start from a seed and make a heuristic search (beam search in DREME and hill-climbing search in DECOD) to find a PPM that maximizes the discriminative power of the motif to separate between positive and negative training data sets. In a large majority of cases, SeedHam and SeedHam+ relearned the PPM from generated data more accurately than the two other methods.

The paper is organized as follows. After preliminaries in Section 2, Section 3 gives the basic SeedHam algorithm and its artefact eliminating variant. Section 4 introduces palindromic index and its application in selecting orientation, implemented in algorithm SeedHam+. Experimental tests are reported in Section 5.

2 Preliminaries of PPM models

A PPM representing a motif of length ℓ in the DNA alphabet $\Sigma = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ is a $4 \times \ell$ matrix $\theta = (\theta_{aj})_{a\in\Sigma, j=1,\ldots,\ell}$ such that θ_{aj} gives the occurrence probability of base $a \in \Sigma$ in position j of the motif. Hence each column $\theta_j = (\theta_{\cdot j})$ of θ defines a categorical distribution $\mathrm{Cat}(\theta_j)$ of Σ. The entire matrix θ represents a random variable $X = X(\theta)$ which is a product of ℓ mutually independent categorical random variables X_j:

$$X = X_1 \times X_2 \times \ldots \times X_\ell,$$

where $X_j \sim \mathrm{Cat}(\theta_j)$, and the values of X are in Σ^ℓ.

As the component variables are assumed mutually independent, the probability $P(u) = P_\theta(u)$ of $u = u_1 \ldots u_\ell \in \Sigma^\ell$ of X is

$$P(X = u_1 \ldots u_\ell) = P(X_1 = u_1)P(X_2 = u_2)\cdots P(X_\ell = u_\ell) = \theta_{u_1,1}\theta_{u_2,2}\ldots\theta_{u_\ell,\ell}.$$

Learning of PPMs from DNA sequence data is complicated by the two-stranded structure of DNA. The reverse complement $\overline{u}$ of a sequence $u = u_1 \ldots u_\ell \in \Sigma^\ell$ is sequence $\overline{u_\ell} \ldots \overline{u_1}$ where $\overline{u_j}$ denotes the complementary base of base u_j. Sequence u is *palindromic* if $u_j = \overline{u_{\ell-j+1}}$ for $j = 1, \ldots, \lceil \ell/2 \rceil$. Similarly, a PPM θ is palindromic if $\theta_{aj} = \theta_{\overline{a},\ell-j+1}$ for all (a, j). Note that $P(u) = P(\overline{u})$ if θ is palindromic.

3 SeedHam Algorithm

3.1 Finding a seed and locating motif instances from training data

Let D be the training data (a collection of sequences in Σ^*) that contains enriched amounts of instances of a target PPM motif X of length ℓ, and let s be a sequence of length ℓ. Let $H_d(s) = \{u \in \Sigma^\ell | h(s, u) \leq d\}$ be the *Hamming d-neighbourhood* of the sequence s. Here $h(s, u)$ is the Hamming distance of (equal-length) sequences s and u, and the radius d is an integer $\leq \ell$. If the given training data is of two-stranded origin, we always denote by D the original data and its reverse complement combined.

Seed-driven motif discovery then proceeds in the following general steps.

1. $s \leftarrow$ a most frequent ℓ-mer of D. Sequence s is selected as the *seed*.
2. For each ℓ-mer $u \in H_d(s)$, $\text{count}(u) \leftarrow$ number of occurrences of u in D.
3. Estimate PPM θ for motif X from the sequences $u \in H_d(s)$ and their counts $\text{count}(u)$.

Detailed implementation of above steps 1 and 2 is possible using elementary techniques that often are fast enough in practice. For big D or ℓ more elaborate implementation techniques from string algorithmics may be needed. We will describe such methods in Subsection 3.3.

The more interesting step 3 is the topic of the next subsection.

3.2 Learning PPM in Hamming neighbourhoods

As the target motif $X(\theta)$ is a product of mutually independent categorical variables $X_1, \ldots, X_\ell$, it follows that, for any $1 \leq j \leq \ell$,

$$P(X_j) = P(X|X_1 \cdots X_{j-1} X_{j+1} \cdots X_\ell). \tag{1}$$

Consider now the multiset of all ℓ-mers of D. This multiset is a mixture sample of ℓ-mers, some of which are coming from X and the rest come, fully or partially, from the background. Our goal is to learn column θ_j of θ, i.e., we want to estimate the parameters of X_j, for some fixed j. It follows from Equation (1), that by conditioning X on ℓ-mer positions other than j, we get a sample of X_j, possibly contaminated by the background. Let a *j-condition* be any $w = (w_L, w_R)$ such that $w_L \in \Sigma^{j-1}$ and $w_R \in \Sigma^{\ell-j}$. We let $\text{count}_j(a, w)$ denote the number of ℓ-mers $w_L a w_R$ in the data, that is, the number times the symbol $a \in \Sigma$ occurs in context (w_L, w_R) in D.

Then, omitting for a moment the correction for background, we could estimate

$$\theta_{aj} \approx \frac{\text{count}_j(a, w)}{\Sigma_{c \in \Sigma} \text{count}_j(c, w)}. \tag{2}$$

This immediately generalizes to a set of j-conditions, i.e., to several different w combined. We will use conditions taken from a Hamming neighbourhood of the seed s. To minimize contamination from background noise, we restrict the learning to ℓ-mers of D that belong to a small Hamming neighbourhood of s. By the properties of products of categorical variables, such ℓ-mers are likely to have a relatively high count which comes mostly from X and not only from the background as they are small variants of s whose count is the highest.

We restrict the learning of θ to a Hamming neighbourhood $H_d(s)$ of s in D, by using the implied set of j-conditions. For learning θ_j, the set of j-conditions becomes

$$W_{j,d}(s) = \{(w_L, w_R) \in \Sigma^{j-1} \times \Sigma^{\ell-j} \mid h(s, w_L c w_R) \leq d \text{ for all } c \in \Sigma\}.$$

As samples for different j-conditions can be combined, (2) becomes

$$\theta_{aj} \approx \frac{\text{count}_j(a)}{\Sigma_{c \in \Sigma} \text{count}_j(c)}, \tag{3}$$

where we have written $\text{count}_j(c) = \Sigma_{w \in W_{j,d}(s)} \text{count}_j(c, w)$.

We have to evaluate (3) for all $j = 1, \ldots, \ell$. It is convenient to organize this such that the contribution of each $u \in H_d(s)$ to the counts is accumulated in one pass that scans through u. Recall that $\text{count}(u)$ denotes the number of occurrences of $u = u_1 \cdots u_\ell \in H_d(s)$ in D. Then it is not difficult to see that the rule for accumulating variables $\text{count}_j(c)$ becomes as follows: if $h(s, u) < d$, then u contributes $\text{count}(u)$ to $\text{count}_j(u_j)$ for all j; if $h(s, u) = d$, then it contributes $\text{count}(u)$ to $\text{count}_j(u_j)$ only for j such that $u_j \neq s_j$.

As data D is only partially covered by ℓ-mers from signal X, and the rest is background, we have at first to correct the counts $\text{count}(u)$ by subtracting estimated contribution from the background. We use here a simple 0-order background model $q = (q_A, q_C, q_G, q_T)$ where each q_c is the frequency of c in D. The background probability of a ℓ-mer $u = u_1 \cdots u_\ell$ is $P_q(u) = \prod_j q_{u_j}$, and the expected number of occurrences of u in a random dataset of the same size as D is $\text{Ecount}(u) = NP_q(u)$, where N is the number of ℓ-mer positions in D.

So we get the following PPM learning algorithm. We say that this algorithm uses *basic counting* of ℓ-mer occurrences, to separate from the counting used in the algorithm variant to be given in Section 3.4.

Algorithm SeedHam

Input: Set of sequences (with reverse complements) D, length of PPM ℓ, Hamming radius d
Output: PPM θ

1. $s \leftarrow$ a most frequent ℓ-mer of D
2. **for all** $u \in H_d(s)$ **do** $\text{count}(u) \leftarrow \max(0,$ number of occurrences of u in $D - \text{Ecount}(u))$
3. **for** $j \leftarrow 1, \ldots, \ell$ **and** $a \in \Sigma$ **do** $\text{count}_j(a) \leftarrow 0$
4. **for all** $u = u_1 \cdots u_\ell \in H_d(s)$ **do**
   if $h(s, u) < d$ **then**
   for $j \leftarrow 1, \ldots, \ell$ **do** $\text{count}_j(u_j) \leftarrow \text{count}_j(u_j) + \text{count}(u)$
   else
   for $j \leftarrow 1, \ldots, \ell$ **do if** $u_j \neq s_j$ **then** $\text{count}_j(u_j) \leftarrow \text{count}_j(u_j) + \text{count}(u)$
5. **for all** $j \leftarrow 1, \ldots, \ell$ **and** $a \in \Sigma$ **do** $\theta_{aj} \leftarrow \text{count}_j(a) / \sum_{b \in \Sigma} \text{count}_j(b)$

Note that SeedHam algorithm differs from the basic alignment method already mentioned in the introduction that aligns the ℓ-mers in the sample, counts the number of occurrences of each element of Σ on each column, and normalizes these counts to get θ. That this algorithm would not estimate θ correctly in a Hamming neighbourhood can be seen, for example, by considering data D that has no motif embedded, the data being background only. A seed s can still be found, but then the standard algorithm applied on, say, $H_1(s)$ would produce a PPM that gives for s a clearly higher probability than for the other ℓ-mers while the correct model should give uniform distribution. Algorithm SeedHam produces such a uniform model in this case; an illustration is given in Figs 1a and 1b. The standard algorithm does so only if the Hamming neighbourhood does not leave any data out, that is, if $H_\ell(s)$ is used.

3.3 Implementation and complexity

A most frequent ℓ-mer s as well as the counts $\text{count}(u)$ for ℓ-mers $u \in H_d(s)$ can be found in linear time $O(|D|)$ using for example suffix-trees (or suffix arrays): First, construct the suffix-tree of D, and associate with each node x of the tree the number $S(x)$ of leaves in the subtree of x and the length $L(x)$ of the sequence represented by the path from the root to x. This can be done in linear time using well-known algorithms [17, 7, 16]. Second, find from the suffix-tree the most frequent ℓ-mer s of D. This can be done by finding the node x such that $L(x) \geq \ell$ and $S(x)$ is largest possible. Then s is the prefix of length ℓ of the sequence represented by the path from the root to x. This again takes linear time. Third, find the counts of ℓ-mers $u \in H_d(s)$ by a depth-first traversal of the tree. Each branch is followed until the Hamming distance between s and the sequence spelled out by the current

depth-first search node is $> d$, or a ℓ-mer $u \in H_d(s)$ is found. Then $\text{count}(u) = S(x)$ where x is the node corresponding to u.

Obviously, this search finds all members of $H_d(s)$ that occur in D. Then the learning part (Steps 3, 4) of Algorithm SeedHam can be performed. The search and learning takes time proportional to the total length of different ℓ-mers of D. Hence the total time requirement becomes $O(|D| + \ell \cdot \min(|D|, \sum_{h=0}^{d} \binom{\ell}{h}(|\Sigma| - 1)^h))$.

As the overhead of suffix-tree algorithms may be large, a straightforward tabulating algorithm, possibly with hashing techniques, can be used for D of modest size to find counts $\text{count}(u)$ and the seed s, after which the learning part of Algorithm SeedHam can be performed. Again, the running time becomes $O(\ell|D|)$.

3.4 Elimination of artefact instances

Here we make a more accurate analysis of the mixing of instances of X and the background on training data D. The multiset of ℓ-mers of D consists of three types of ℓ-mers: (i) ℓ-mers that are instances of X; (ii) ℓ-mers that are completely outside the instances of X; (iii) ℓ-mers that overlap both an instance of X and background.

When X has strong self-overlaps, D has tendency to have artefact instances of X in category (iii). As an extreme example, consider $X = \text{AAAAAAAA}$, i.e, each position of X has A with probability 1. Then D has lots of 8-mers AAAAAAAA and, by one-symbol shift, lots of 8-mers CAAAAAAA, GAAAAAAA, and TAAAAAAA. If not eliminated, such artefact instances at distance 1 from the seed would leak to the learned X such that C, G, and T will get clearly non-zero probability at position 1.

To avoid counting self-overlapping artefact instances, we introduce a dominance relation of ℓ-mers. Let g be a positive integer giving maximum shift in a self-overlap that is still considered significant (note that if a shift is large and hence the self-overlap is short, then the implied bias in counts rapidly gets very small). We say that the ℓ-mer u in position i of D *dominates*, if $h(s, u) < h(s, v)$ for all ℓ-mers v that are located in $\pm g$ proximity of i in D, that is, in positions $i - g, \ldots, i - 1, i + 1, \ldots, i + g$ of D.

For an example of dominance, consider a single-stranded data in the setting $\ell = 4$, $s = \text{AACG}$, $g = 2$, and $d = 2$. When counting the dominating occurrences of the 4-mer $\text{AGTG} \in H_d(s)$, we need to take the context of the occurrence into account. If we see a string TTAGTGAA in the data, we count this occurrence of AGTG as it dominates here: all the other 4-mers of this string obviously have Hamming distance from the seed greater than 2. But in the context TAAGTGAA the 4-mer AGTG is not counted, because, for instance, $h(\text{TAAG}, s) = 2$ as well.

We then slightly modify the SeedHam algorithm: for each ℓ-mer u, include into its occurrence count only the dominating occurrences of u in D. This way of counting is called *dominance counting*, and the resulting count of an ℓ-mer u is denoted as $\text{count}_{\text{dominant}}(u)$. Intuitively, this rule means that the true instances of X are assumed to locate in D more than g positions apart and to dominate in their $\pm g$ proximity.

Dominance counting needs an accordingly modified background correction. We denote by $\lambda \in [0, 1]$ the relative abundance of the motif instances in D. Hence λN of the N ℓ-mer sites of D are from X. Then the expected artefact count of an ℓ-mer u can be written as the sum of occurrence counts in ℓ-mer categories (ii) and (iii):

$$\text{Ecount}_{\text{dominant},\lambda}(u) = (1 - \lambda(2(\ell + g) - 1))N \cdot P_q(u) + \lambda N \sum_{\substack{j=-(\ell+g-1) \\ j \neq 0}}^{j=\ell+g-1} P_{T_j,\text{dominant}}(u),$$

where $\ell^* = \ell + 2g$ and T_j is a $4 \times \ell^*$ PPM built from θ and q as described below. Note that $j = 0$, which defines the category (i), is excluded. For any ℓ-mer u and a $4 \times \ell^*$ PPM T, $P_{T,\text{dominant}}(u)$ is defined as

$$P_{T,\text{dominant}}(u) = \sum_{v \in \Sigma^{\ell^*}} [v(0) = u] \prod_{\substack{j=-g \\ j \neq 0}}^{g} \prod_{k \in \{-1,1\}} [h(v(j), s^k) > h(u, s)] P_T(v),$$

where $[\cdot]$ are Iverson's brackets, s^k is the seed in direction k, and $v(j)$ is the ℓ-mer that starts from position j of v. The PPMs $T_j, j = -(\ell + g - 1), \ldots, \ell + g - 1$, are models for the alternative ways how a string of length ℓ^* can overlap in D the boundary between motif instance and background. PPM T_0 has θ in the middle, with g columns of q before and after it. PPM T_j for $j > 0$ has θ shifted j positions to the left from the center and for $j < 0$, j positions to the right. Note that our technique here is similar to DECOD [4].

As θ is still unknown when the background correction has to be made, we use uncorrected θ to build models T_j. We solve the parameter λ numerically from the equation

$$\text{count}(u) = \text{Ecount}_\lambda(u) := (1 - \lambda(2\ell - 1))N \cdot P_q(u) + \lambda N \sum_{j=-(\ell-1)}^{j=\ell-1} P_{R_j,\lambda}(u),$$

where $R_{j,\lambda}$ is the background corrected PPM generated from corrected counts

$$\text{count}_{\text{dominant}}(u) - \text{Ecount}_{\text{dominant},\lambda}(u).$$

The running time of background correction is exponential in $\ell^* = \ell + 2g$. Therefore in our implementation we do the correction only for $\ell \leq 10$. Since for longer motifs the effect of background is very small for typical data sizes, this restriction has no big effect on the accuracy of the learned models. We have used $g = 4$ as a default.

Figs 1c and 1d give an example of the effect of dominance counting: for a PPM AAAAAAAA, 10 000 instances were generated to create a total data of length 400 000 bp (hence $\lambda = 0.025$). The PPM relearned by basic SeedHam shows clear contamination from artifact instances while SeedHam with dominance counting removes it.

4 Motif Orientation in Two-Stranded Case: SeedHam+ algorithm

PPM discovery is in practice complicated by the two-strandedness of DNA. Although the motif itself may have direction (for example, a transcription factor binds to DNA in a specific orientation), it is not possible to infer the direction from motif instances that may occur equally in both strands of the DNA. Moreover, an instance in one strand means that we see the reverse complement of it in the other strand. In fact, the counts of a ℓ-mer u and its reverse complement $\overline{u}$ are always equal if the counts are taken along both strands of DNA.

This symmetry should be broken such that we use in the PPM learning only ℓ-mers that have the same direction with respect to the underlying motif. Otherwise we would always get palindromic PPMs.

4.1 Selection of the orientation

The following heuristic could be used for resolving the orientations: To select between u and $\overline{u}$, take the one whose Hamming distance to the seed s is shorter, that is, if $h(s, u) < h(s, \overline{u})$ then take u, and if $h(s, \overline{u}) < h(s, u)$ then take $\overline{u}$.

For this rule to work it is necessary that $h(s, u) \neq h(s, \overline{u})$, which means that s may not be a palindrome. For palindromic seeds we have the following observation.

(a) Alignment algorithm.

(c) `Basic` counting.

(b) SeedHam algorithm.

(d) `Dominance` counting.

Figure 1 (a&b) Learning a PPM from training data that has only uniform background but no motif. Using a seed ACGGTTGG, the alignment algorithm finds a PPM in which the seed dominates while SeedHam correctly finds a PPM that represents the uniform background. **(c&d)** Contamination of the learned model due to artefact occurrences. SeedHam algorithm with basic counting (above) and with dominance counting (below) was used. Both methods used Hamming radius one, and the training data consisted of a single sequence that had the single motif instance AAAAAAAA implanted at every 40th position 10 000 times. Other positions were filled with random uniform background. Dominance counting effectively removes the contamination due to artefact occurrences.

▶ **Theorem 1.** *If the seed s in Algorithm SeedHam is a palindrome, then the resulting PPM θ is palindromic.*

Proof. Let s be a palindrome. Then $H_d(s) = H_d(\overline{s})$, and a sequence $u = u_1 \dots u_\ell$ is in $H_d(s)$ iff $\overline{u}$ is in $H_d(s)$. Then $u = u_1 \dots u_j \dots u_\ell$ contributes 1 to the count of $\theta_{u_j,j}$ iff $\overline{u} = \overline{u_\ell} \dots \overline{u_j} \dots \overline{u_1}$ contributes 1 to the count of $\theta_{\overline{u_j},\ell-j+1}$ which is the element of θ that is palindrome symmetric to $\theta_{u_j,j}$. This is because $h(s_{\neg j}, u_{\neg j}) = h(\overline{s_{\neg j}}, \overline{u_{\neg j}})$, and hence $u_{\neg j}$ is used as a condition iff $\overline{u_{\neg j}}$ is used, where $u_{\neg j}$ denotes the sequence $u_1 \dots u_{j-1} u_{j+1} \dots u_\ell$. Palindrome symmetric elements of θ get equal counts which proves our claim. ◀

Palindromic index $\mathrm{Pi}(u)$ of a sequence $u \in \Sigma^\ell$ is defined as $\mathrm{Pi}(u) = h(u, \overline{u})$. If $|u| = \ell$ is odd then $\mathrm{Pi}(u) \geq 1$ as the symbols in the center position of u and $\overline{u}$ are always different. For even $|u|$, $\mathrm{Pi}(u)$ has an even value $0, 2, \dots, |u|$. For odd $|u|$, $\mathrm{Pi}(u)$ has an odd value $1, 3, \dots, |u|$.

If $\mathrm{Pi}(u) = 0$ (and hence $|u|$ is even), then u is a (DNA) palindrome.

We call a set $Q \subseteq \Sigma^\ell$ *conflict-free* if $Q \cap \overline{Q}$ is empty, i.e., if $u \in Q$ then $\overline{u} \notin Q$. So if $H_d(s)$ is conflict-free then all $u \in H_d(s)$ are such that $h(s, u) < h(s, \overline{u})$. Hence Algorithm SeedHam with such an $H_d(s)$ implicitly applies our rule for choosing between u and $\overline{u}$.

▶ **Theorem 2.**

(a) *If $d < \mathrm{Pi}(s)/2$ then $H_d(s)$ is conflict-free.*

(b) *If $\mathrm{Pi}(s) = 2$ and $d = 1$, then $H_d(s) \cap H_d(\overline{s})$ contains exactly two sequences and these sequences are palindromes.*

Proof. (a) To derive a contradiction, let $d < \mathrm{Pi}(s)/2$ and assume that there is a sequence u in $H_d(s) \cap H_d(\overline{s})$. Then $\mathrm{Pi}(s) = h(s, \overline{s}) \leq h(s, u) + h(u, \overline{s}) \leq 2d$ which contradicts the assumption that $d < \mathrm{Pi}(s)/2$.

(b) $\mathrm{Pi}(s) = 2$ implies that ℓ must be even and that $s_j = \overline{s_{\ell-j+1}}$ except for one value $j \leq \ell/2$. Then $s_j \neq \overline{s_{\ell-j+1}}$ and hence $\overline{s_j} \neq s_{\ell-j+1}$. Then sequences $s(s_j|\overline{s_{\ell-j+1}})$ and $s(s_{\ell-j+1}|\overline{s_j})$ are different palindromes and have distance 1 from both s and $\overline{s}$.

A sequence in $H_1(s) \cap H_1(\overline{s})$ should be the intermediate sequence on the two step path from s to $\overline{s}$ that transforms s to $\overline{s}$ using one symbol changes. There are exactly two such

paths, one changing first s_j and then $s_{\ell-j+1}$, and the other changing first $s_{\ell-j+1}$ and then s_j. This gives the above two sequences. ◀

According to Theorem 2 (a), a conflict-free $H_d(s)$ for Hamming radius $d \geq 3$ requires a seed s such that $\mathrm{Pi}(s) \geq 2d+1$. To achieve this, one could use in Algorithm SeedHam as seed s the ℓ-mer that has the largest count among the ℓ-mers whose palindromic index is $\geq 2d+1$, provided that the count of such a ℓ-mer is sufficiently large. Note that increasing the palindromic index of s may make the count of s smaller, which means that Algorithm SeedHam would utilize a smaller fraction of the data.

For a large data D already Hamming radius $d = 1$ can give an accurate estimate of the PPM. Then, by Theorem 2 (b), we only need a seed s such that $\mathrm{Pi}(s) \geq 2$. However, $H_1(s)$ is not conflict-free as $H_1(s) \cap H_1(\overline{s})$ is not empty but contains palindromes. Fortunately, deciding their orientation is not needed, as palindromes are symmetric and hence there is only one orientation for them. One only has to divide their observed counts by two as each occurrence of a palindrome also appears on the opposite DNA strand and hence is counted twice.

To conclude the above discussion we give Algorithm SeedHam modified such that it uses a seed with high-enough palindromic index to avoid orientation conflicts. Only step 1 needs changes.

Algorithm SeedHam+

Input: Set of sequences (with reverse complements) D, length of PPM ℓ, Hamming radius d, count threshold $m = \max(20, N4^{-\ell} + 2\sqrt{N4^{-\ell}(1-4^{-\ell})})$, i.e., two standard deviations more than expected by uniform background and at least 20.
Output: PPM θ

1. $r \leftarrow$ **if** $d = 1$ **then** 2 **else** $2d+1$.
$s \leftarrow \ell$-mer u such that $\mathrm{Pi}(u) = r$ and the count of u in D is largest possible. However, if this count is $< m$ then $s \leftarrow \ell$-mer u with largest palindromic index among ℓ-mers whose count is $\geq m$.
2.-5. As Algorithm SeedHam.

5 Experimental evaluation

To compare the performance of algorithms SeedHam and SeedHam+ we randomly selected from the article of Jolma *et al* [6] altogether 24 PPMs, eight PPMs of each of lengths $\ell = 8$, 13, and 18. For each PPM, four data sets were randomly generated: number of motif occurrences being either 100 or 10 000 and occurrences oriented either in single direction or in both directions. The occurrences were placed starting at every 40th base in a single sequence. Between motif occurrences uniform random background was used. This long sequence was then used as training data D for SeedHam and SeedHam+ using either basic or dominance counting. The learned PPMs were compared against the originals, and the learning error was measured using the maximum norm (i.e., maximum absolute value of the difference of the corresponding entries of the original and the learned PPM; a shift of -1, 0, +1 columns was allowed and minimum of the max-norm over these shifts was taken as the learning error). Hamming radii $d \in \{1, \ldots, 5\}$ were used in the experiments.

The results from these experiments are reported in Fig. 2. When comparing Figs 2a and 2b, the difficulty of learning the model in the two-directional case is clearly visible.

Table 1 Changes of palindromic index in the SeedHam+ experiments on generated data. For each length, eight factors and two data set sizes and five Hamming radii give 80 experiments in total. The table shows that the required palindromic index is difficult to reach. It is easier for $d = 1$ but rapidly gets difficult with larger d.

	Length 8	Length 13	Length 18
Need for increase	52/80	28/80	24/80
No increase possible	42/52	12/28	12/24
Did not increase enough	4/52	8/28	8/24
Increased enough	6/52	8/28	4/24

Introducing the seed with optimized palindromic index (Fig. 2c) does help a little bit, but with the motif length and dataset sizes we used, it was not always possible to optimize the seed, see Table 1. This is because moving far away (in Hamming distance) from the most common ℓ-mer, decreases the count of seed s (and counts in $H_d(s)$) too much, and hence gives inaccurate estimate of the motif.

Algorithms are fast in practice. On average it took less than a second of CPU time for the basic counting method and 7 minutes of CPU time for the dominance counting method per PPM, when SeedHam was run on a generated data set with 10 000 occurrences (400 000 bp). The longer running time of dominance counting is due to the relatively slow background correction. For motifs of length larger than 10 this correction can be lifted. Performance testing was done using a single thread on Intel Xeon CPU X7350 running at 2.93 GHz.

We also applied SeedHam to real SELEX data. We used data sets from [6] that were used by Jolma *et al* to obtain the previously mentioned 24 PPMs. SeedHam showed in these experiments consistent and robust performance with quite small differences between different variants of the method.

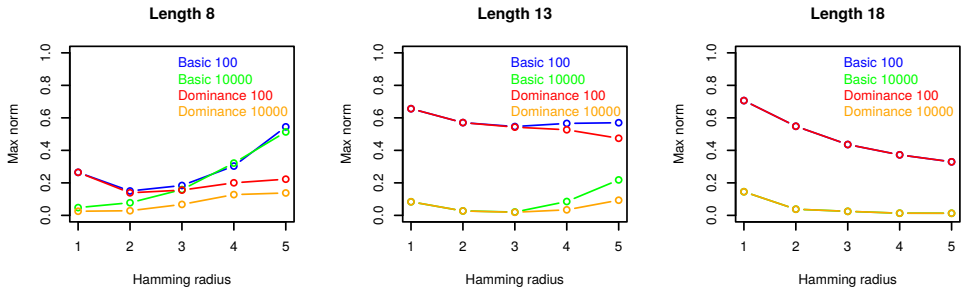
Our experimental evaluation suggests that in practice for large data already Hamming radius $d = 2$ gives accurate results and larger d do not improve too much. For small data this holds true for short motifs but for longer ones using larger d would improve accuracy. For larger d the dominance version of SeedHam clearly improves accuracy.

Results of comparison of SeedHam with earlier algorithms DREME [1] and DECOD [4] are given in Table 2. SeedHam used dominance counting in these experiments. The generated training data contained 1000 instances of the motif, implanted into sequences of length 40. The same motifs were used as in the experiments of Fig. 2. Hamming radius d used in the experiments was 1, 3, and 5 for motif lengths 8, 13, and 18, respectively. SeedHam gives the most accurate result in 21 cases out of 24, often with clear margin. We also tested SeedHam+ with dominance counting. Then the learning accuracy of the LMX1B motif improved to 0.14, making this, too, the best among the compared results.

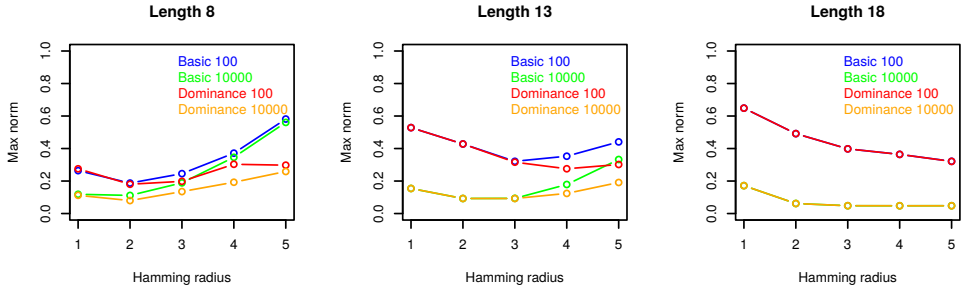
6 Discussion

We gave a general formulation and analyzed the PPM learning algorithm SeedHam that restricts the use of the training data only to a small Hamming neighbourhood $H_d(s)$ of a seed s. We also introduced a dominance counting technique to correct for artifact occurrences of self-similar motifs, as well as a novel seed selection rule, based on the palindromic index, that gives seeds such that the orientation of the motif instances restricted to $H_d(s)$ in two-stranded DNA is unambiguous.

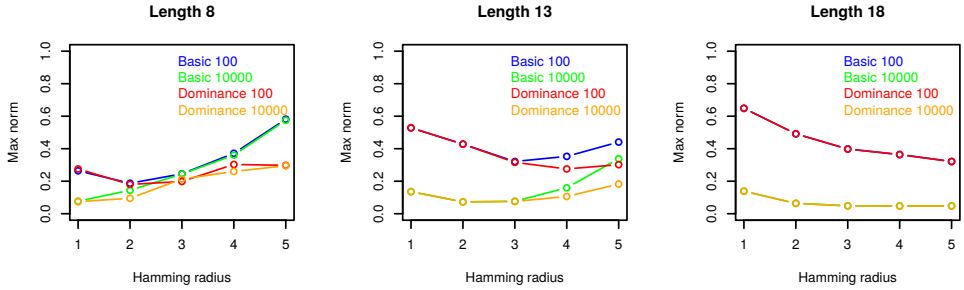
Even if we have chosen a seed with optimal palindromic index, there is still another source of inaccuracy due to the two-strandedness of DNA. When we see a (putative) motif occurrence



(a) Single direction.



(b) Both directions.



(c) Both directions using SeedHam+.

Figure 2 Two data sets were generated for each given PPM: one with 100 random occurrences and another with 10 000 random occurrences of the motif. The occurrences were placed in the same orientation starting at every 40th position. The gaps between the occurrences were filled with uniform random background. The three columns of the figure correspond to experiments with sets of 8 motifs of lengths 8, 13, and 18, respectively. The average maximum error between the original and the relearned models is shown for each Hamming radius, each size of data set, and both methods of counting the number of occurrences of sequences in the Hamming neighbourhood, namely, the basic and dominance counting. Background subtraction was only applied to motif length 8. **(a)** Accuracy of SeedHam algorithm for data with single direction of motif occurrences. For motif lengths 8 and 13 the benefits of the dominance counting over the basic counting becomes visible as the Hamming radius increases. For length 18, the used Hamming radii were not large enough to show any difference between the two counting methods. The accuracy of the learned models is very good. Different motif lengths show different behaviour with respect to the Hamming radius. With every motif length, increasing the Hamming radius initially gives more accurate result, but after some point the increased contamination from the background starts to weaken the result. With factor length 18, however, the dataset would have to be very large for the background to have an effect on the learned motif. **(b)** Accuracy of SeedHam algorithm for data with motif occurrences in random orientation. As expected, the learning error is larger than in the case of single direction; note, however, that for $\ell = 13$, data size 100, the accuracy is here better! **(c)** Accuracy of SeedHam+ algorithm for the same data as in panel (b), i.e., optimization of the palindromic index of the seed enabled. The effect of higher palindromic index is in general minor, except for $\ell = 8, d = 1$ in which case we get here the best accuracy.

Table 2 Comparison of SeedHam to DREME and DECOD. Data sets of 1000 sequences of length 40 were generated using original motifs of TFs indicated below (note that the factor RARA had two distinct motifs). One occurrence of the motif was planted in each sequence. The distances are maximum element-wise distances between the original PPM and the PPM relearned from the generated data. DREME was only run for motifs of length 8 which is its maximum recommended motif length.

Factor	DREME	DECOD	SeedHam dominance
DLX5	0.16	0.26	0.32
GATA3	0.38	0.50	0.10
ISL2	0.22	0.41	0.08
LMX1B	0.44	0.24	0.32
MEIS2	0.16	0.46	0.03
MSX1	0.08	0.35	0.08
NR2F1	0.04	0.55	0.04
OTX1	0.20	0.97	0.06

(a) Motif length 8.

Factor	DECOD	SeedHam dominance
DUXA	0.38	0.07
EOMES	0.55	0.05
LBX2	0.16	0.13
LHX9	0.37	0.36
NFKB1	0.88	0.05
NR2F1	0.50	0.06
PRDM4	1.00	0.01
ZNF238	0.50	0.05

(b) Motif length 13.

Factor	DECOD	SeedHam dominance
CUX1	0.18	0.23
E2F3	0.44	0.03
ETS1	0.22	0.09
KLF13	0.50	0.04
MGA	0.14	0.06
MSX1	0.35	0.06
RARA	0.50	0.02
RARA	0.55	0.01

(c) Motif length 18.

u, that belongs to $H_d(s)$, on a DNA strand, it is not possible to decide, without additional information, whether or not the motif occurrence really is this u or the reverse complement $\overline{u}$ on the opposite strand. In the latter case the occurrence should not be used when building the model. Hence, the occurrence count is necessarily a mixture of counts from u and $\overline{u}$. It is not possible to directly resolve the mixture as the mixing proportions of how much of the count comes from u and how much from $\overline{u}$ are unknown. Mixing proportion depends on the probabilities $P(u)$ and $P(\overline{u})$ of u and $\overline{u}$ to occur as motif instances. A subject for further study is to incorporate into the SeedHam algorithm a maximum likelihood estimator for improved resolution of the mixture.

References

1 Timothy L. Bailey. Dreme: motif discovery in transcription factor chip-seq data. *Bioinformatics*, 27(12):1653, 2011.

2 Michael F. Berger, Anthony A. Philippakis, Aaron M. Qureshi, Fangxue S. He, Preston W. Estep, and Martha L. Bulyk. Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. *Nature Biotech.*, 24(11):1429–1435, 2006.

3 Mathieu Blanchette and Saurabh Sinha. Separating real motifs from their artifacts. *Bioinformatics*, 17(SUPPL. 1), 2001.

4 Peter Huggins, Shan Zhong, Idit Shiff, Rachel Beckerman, Oleg Laptenko, Carol Prives, Marcel H. Schulz, Itamar Simon, and Ziv Bar-Joseph. DECOD: fast and accurate discriminative DNA motif finding. *Bioinformatics*, 27(17):2361, 2011.

5 Arttu Jolma, Teemu Kivioja, Jarkko Toivonen, et al. Multiplexed massively parallel SELEX for characterization of human transcription factor binding specificities. *Genome Res.*, 20(6):861–873, 2010.

6 Arttu Jolma, Jian Yan, Thomas Whitington, Jarkko Toivonen, et al. DNA-binding specificities of human transcription factors. *Cell*, 152(1–2):327–339, 2013.

7 Edward M. McCreight. A space-economical suffix tree construction algorithm. *J. ACM*, 23(2):262–272, 1976.

8 Arnold R. Oliphant, Christopher J. Brandl, and Kevin Struhl. Defining the sequence specificity of DNA-binding proteins by selecting binding sites from random-sequence oligonucleotides: analysis of yeast GCN4 protein. *Mol. Cell. Biol.*, 9(7):2944–2949, 1989.

9 Giulio Pavesi, Giancarlo Mauri, and Graziano Pesole. In silico representation and discovery of transcription factor binding sites. *Brief. Bioinformatics*, 5(3):217–236, 2004.

10 Gordon Robertson, Martin Hirst, Matthew Bainbridge, Misha Bilenky, Yongjun Zhao, Thomas Zeng, Ghia Euskirchen, Bridget Bernier, Richard Varhol, Allen Delaney, Nina Thiessen, Obi L. Griffith, Ann He, Marco Marra, Michael Snyder, and Steven Jones. Genome-wide profiles of STAT1 DNA association using chromatin immunoprecipitation and massively parallel sequencing. *Nat. Methods*, 4(8):651–657, 2007.

11 Albin Sandelin, Wynand Alkema, Par Engstrom, Wyeth W. Wasserman, and Boris Lenhard. JASPAR: an open-access database for eukaryotic transcription factor binding profiles. *Nucleic Acids Res.*, 32(Database issue):D91–94, 2004.

12 Gary D Stormo. DNA binding sites: representation and discovery. *Bioinformatics*, 16(1):16–23, 2000.

13 Gary D. Stormo, Thomas D. Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. *Nucleic Acids Res.*, 10(9):2997–3011, 1982.

14 Martin Tompa, Nan Li, Timothy L. Bailey, George M. Church, Bart De Moor, Eleazar Eskin, Alexander V. Favorov, Martin C. Frith, Yutao Fu, W. James Kent, et al. Assessing computational tools for the discovery of transcription factor binding sites. *Nature biotechnology*, 23(1):137–144, 2005.

15 Craig Tuerk and Larry Gold. Systematic evolution of ligands by exponential enrichment: RNA ligands to bacteriophage T4 DNA polymerase. *Science*, 249(4968):505–510, 1990.

16 Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

17 Peter Weiner. Linear pattern matching algorithms. In *Switching and Automata Theory, 1973. SWAT'08. IEEE Conference Record of 14th Annual Symposium on*, pages 1–11, 1973.

18 Edgar Wingender. The TRANSFAC project as an example of framework technology that supports the analysis of genomic regulation. *Brief. Bioinformatics*, 9(4):326–332, 2008.