Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Tesi di Laurea Magistrale in Ingegneria Informatica

New Robust Similarity Measures Derived from Entropic Profiles

Laureando:
Morris Antonello

Relatore:
Matteo Comin

July 9, 2014
A.Y. 2013/2014

Abstract

Enhancers are stretches of DNA (100-1000 bp) that play a major role in development gene expression, evolution and disease. They contain short (6-10 bp) DNA motifs that act as binding sites for transcription factors (TFBSs) so that their prediction and classification can be addressed by similarity searches. Alignment-free approaches provide viable alternatives to the common way, i.e. sequence alignment. For example, D_2, D_2^* and N_2 are well-known similarity measures based on word counts. In this work, I contribute showing that entropic profiles pave the way to more robust but still efficient methods. First, I delve deeper into the definition of entropic profile and its statistical properties, which have not been fully explored yet. Then, I describe the experimental validation procedures and compare my results with the state of the art paying particular attention to the role of the statistical model. Experiments on both simulated and real enhancers reveal that the multi-resolution property of the new methods make the similarity scores more robust with the change of the k-mer length. Finally, the main results are summarized and suggestions for future research are given.

Contents

Chapter 1

Introduction

1.1 Biological sequence representation

The genome is the genetic material of an organism. In most of them, it is encoded in DNA and includes both the genes and the non-coding sequences. Most DNA is located in the cell nucleus, where genes are packaged in thread-like structures called chromosomes. DNA is made of two strands and looks like a double helix. Each strand can be considered as a long sequence of nucleotides, namely adenine, cytosine, guanine and thymine. Figure 1.1 is a diagram of the double helix structure that shows that both strands are anti-parallel and complementary. Further information about DNA can be found in [1].

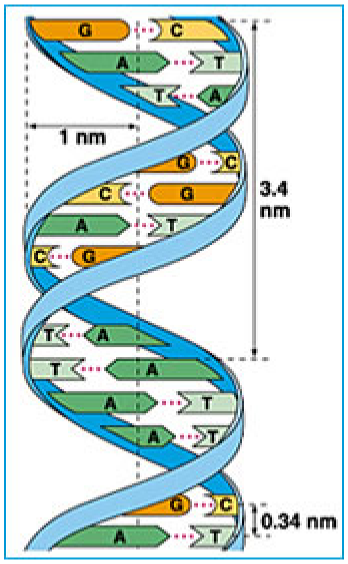

Figure 1.1: diagram of the DNA double helix structure, both strands are anti-parallel and complementary

Even if DNA is a flexible, three-dimensional molecule interacting in a dynamic environment, it can be represented as a string of arbitrary length built on the alphabet

$\Sigma = \{A, C, G, T\}$ [2]. Along with this standard assumption, many biological problems, such as the one presented in the next section, can be addressed computationally.

1.2 Transcriptional enhancers

Many articles [3], in particular [4], discuss recent views on enhancers or cis-regulatory modules (CRMs), one of several types of genomic regulatory elements, and their coordinated action in regulatory networks, as well as methods to predict them on the basis of their currently known properties. They are stretch of DNA (100-1000 bp) that play a major role in development gene expression, evolution and disease. Indeed, they can upregulate, i.e. enhance, the transcription process, that otherwise would be weak, of a target gene into RNA by RNA polymerase II (Pol II), see Figure 1.2. As a result, during animal development, a single cell, the fertilized egg, gives rise to a multitude of different cell types and organs, that acquire different morphologies and functions by expressing different sets of genes.

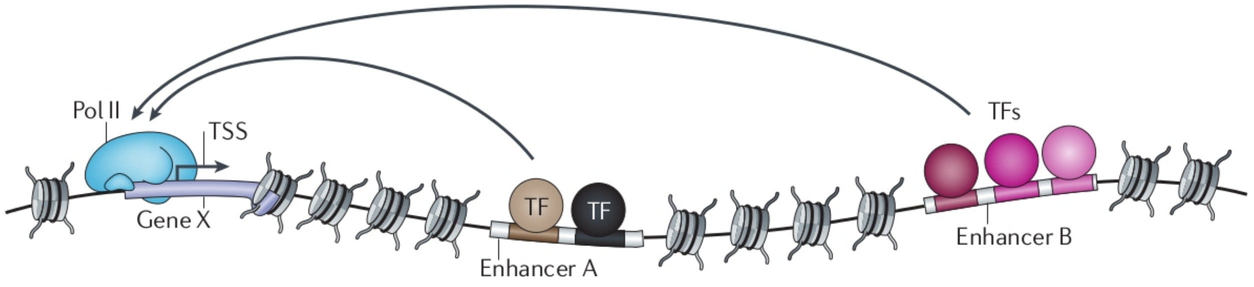

Figure 1.2: transcriptional enhancers

It is worthwhile summing up their main features. First, they contain short (6-10 bp) DNA motifs that act as binding sites for transcription factors (TFBSs) and often allow different nucleotides at some of the binding positions, in other words there may be word mismatches. Second, they act seemingly independently of the distance and orientation to their target genes as a consequence of looping, see Figure 1.3. It follows that the strand to which a CRM under study belongs is unknown so both cases need to be considered. Third, they maintain their functions independently of the sequence context, they are modular and contribute additively and partly redundantly to the overall expression pattern of their target genes. Finally, enhancers with similar transcription factors binding sites have a high probability of bearing the same function. Thus, it is evident that predictions and classifications of enhancers can be addressed by similarity searches.

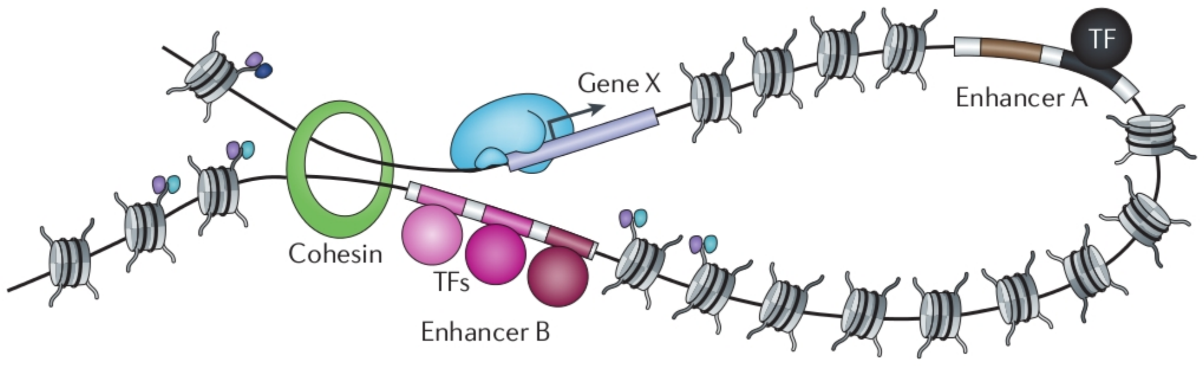

Figure 1.3: enhancer looping

1.3 Computational approaches

The common way to identify homologous sequences is sequence alignment, for which many algorithms have been proposed in literature, e.g. the Smith-Waterman algorithm [5] and BLAST [6]. Nevertheless, as reported in many articles such as [7] [8], they are unsuitable for predicting and classifying enhancers through the matching of transcription factor binding sites for many reasons:

- transcription factor binding sites are short motifs so they frequently match to genomic or even random DNA sequences and enhancer similarity or dissimilarity may be due primarily to their background;
- enhancer location and orientation do not matter so no reliable alignment can be obtained;
- they are time-consuming and inadequate for comparing sequences in realistically large datasets, e.g. ChIP-seq datasets;
- enhancers do not work alone and their coordinated action has not been already fully explored.

On the contrary, alignment-free approaches provide viable alternatives [7] [8]. With the aim of effectively summing up sequence content, they can be based on many ideas, e.g. k-mer counts, chaos theory or common substrings. Concerning the first type, many solutions have been proposed and their common procedure is explained in [8] and summarized in the flowchart in Figure 1.4. The input can be a set of sequences, the similarity is measured for every pair of sequences and a matrix of pairwise scores is filled. The goodness of the results can be assessed by many types of tests on simulated and real data. It is worthwhile mentioning some well-known measures that will be further investigated. Consider two genome sequences A and B and let A_w and B_w be the frequencies of word w in A and B.

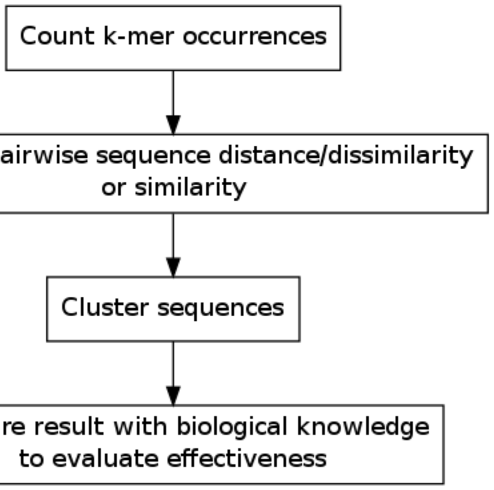
Count k-mer occurrences
Measure pairwise sequence distance/dissimilarity or similarity
Cluster sequences
Compare result with biological knowledge to evaluate effectiveness

Figure 1.4: common procedure of k-mer counts alignment-free methods

Historically, D_2 [9], see formula 1.1, is one of the first proposed similarities and is defined as the inner product of the k-mer frequency vectors. Despite its simplicity and distance properties, D_2 can be dominated by the noise caused by the randomness of the background and has low statistical power to detect potential relationship. As a result, more powerful variants, D_2^S and D_2^* [10], see formulas 1.2 and 1.3, have been developed by standardizing the k-mer counts with their expectations and standard deviations. In practice, the statistical model under which they are calculated is estimated from the concatenation of the two sequences AB. Another variant is D_2^Z [11], which measures the number of standard deviations by which the observed value of D_2 deviates from the mean.

$$D_2 = \sum_w A_w B_w \tag{1.1}$$

$$D_2^S = \sum_w \frac{(A_w - E[AB_w])(B_w - E[AB_w])}{\sqrt{(A_w - E[AB_w])^2 + (B_w - E[AB_w])^2}} \tag{1.2}$$

$$D_2^* = \sum_w \frac{(A_w - E[AB_w])(B_w - E[AB_w])}{\sqrt{Var[AB_w]}\sqrt{Var[AB_w]}} = \sum_w \frac{(A_w - E[AB_w])(B_w - E[AB_w])}{Var[AB_w]} \tag{1.3}$$

$$D_2^z = \frac{D_2 - E[D_2]}{\sqrt{Var(D_2)}} \tag{1.4}$$

An implementation of D_2, D_2^* and D_2^Z is provided by ALF [13], which, by default, uses another similarity measure named N˙2, one of the best available methods.

N_2 aims at overcoming the limitation of exact word counts by taking into account word neighbourhood counts. For every word w, they are termed as $n(w)$ and define a set of words somehow linked to w, e.g. reverse complement and mismatches. Nevertheless, this can lead not only to smoothing but also to blurring, depending on the neighbour weights, and, in any case, taking them into account slows down the application. Consider the generic sequence S, the weighted word neighbourhood counts for every word w are defined as follows:

$$N^S_{n(w)} = \sum_{w' \in n(w)} a_{w'} N^S_{w'} \tag{1.5}$$

they are then standardized for correcting inter-variable dependence:

$$\tilde{N_w}^S = \frac{N^S_{n(w)} - E[N^S_{n(w)}]}{\sqrt{V(N^S_{n(w)})}} \tag{1.6}$$

as it will be shown, the calculation of expected value and variance is not trivial. Finally they are normalized with the Euclidean norm of the vector of neighbourhood counts:

$$\hat{N}^S_w = \frac{\tilde{N}^S_w}{||\tilde{N}_w||} \tag{1.7}$$

so:

$$N_2 = \sum_w \hat{N}^A_w \hat{N}^B_w \tag{1.8}$$

It is worthwhile pointing out one of the major differences between the implementation of D_2* and N_2. The statistical model under which standardization is performed is no longer estimated from the concatenation of the two sequences, on the contrary it is that of the respective sequence. This choice is an important speed-up.

In literature, many other measures have been proposed, the simplest is the euclidean l2, others are Kullback–Leibler and Jensen-Shannon divergences [12]. Nevertheless, alignment-free comparisons are less accurate than sequence alignment as replacing the sequence with the vector of counts results in a loss of information. In addition, another drawback is that they are all based on the choice of the resolution k, i.e. the motif length, which crucially influences performances but cannot be known in advance. The attention on these issues is drawn by UnderII [14] [15] [16], a parameter-free alignment-free method based on variable length words, more precisely on two driving principles: irredundancy and underlying positioning. Going in this direction but developing other ideas, in this work, I contribute showing that entropic profiles pave the way to more robust but still efficient alignment-free methods.

1.4 Entropic profiles

There are two definitions of entropic profiles: the former is global [18], i.e. with respect to the entire sequence, while the latter [19][20] [21] [22] is local, i.e. defined for every sequence position. Both of them are based on chaos-game representation (CGR) [23], the term chaos game refers to a method of generating fractals using a polygon and an initial point selected at random inside it.

The common starting point is the adaptation of the construction rules of the chaos game so as to generate fractals from DNA sequences [24]. To be clear, each of the four corners of a square is labelled A, C, G or T. The first base of the sequence, C for example, is plotted half way between the center of the square and the C corner; if the next base is T, for example, then a point is plotted half way between the previous point and the T corner; and so on. Formally, the CGR mapping $x_i \in \mathbf{R}^2$ of a l-length DNA sequence $S = s_1 s_2 ... s_l$, $s_i \in \Sigma = \{A, C, G, T\}, i = 1, ..., l$ is expressed as:

$$\begin{cases} x_0 = (\frac{1}{2}, \frac{1}{2}) \\ x_i = x_{i-1} + \frac{\gamma_i - x_{i-1}}{2} \end{cases}, \text{ where} \tag{1.9}$$

$$\gamma_i = \begin{cases} (0,0) & \text{if } s_i = A \\ (0,1) & \text{if } s_i = C \\ (1,0) & \text{if } s_i = G \\ (1,1) & \text{if } s_i = T \end{cases}. \tag{1.10}$$

Figure 1.5 compares two images built with the above construction rules, the former is obtained from the human β-globin sequence while the latter, much more uniform, from a random sequence of the same length. CGRs have four main properties:

1. each point can be drawn by knowing the corresponding base and the previous point;

2. the last point reveals the total sequence;

3. if the CGR square is divided into small subsquares 4^{-m} in size, each m-square corresponds one to one with each of the possible m-mer;

4. the density of points (number of points in a given m-square) is the frequency of the corresponding m-mer in the chain.

The global definition of entropic profiles [18] is one of the first fruitful attempt to apply information theory to the interpretation of DNA sequences. The underlying idea is that crude CGR subsquare counts, i.e. m-mer frequencies, should not be taken into account directly. As an alternative, Shannon's entropy should be calculated from the density histogram, i.e. the plot of the number of m-squares with a given density versus the density of points. This method proved to be useful to clearly discriminate between random and natural sequences.

The local definition of entropic profiles [19] [20] comes from the use of the CGR representation to estimate the sequence Rényi entropy on the basis of the Parzen window density estimation method. The selection of a fractal kernel leads to the entropic profile function, the main EP, which is defined for every location i of the entire sequence S:

$$\hat{f}_{L,\varphi}(x_i) = \frac{1 + \frac{1}{l}\sum_{k=1}^{L} 4^k \varphi^k \cdot c\left([i-k+1, i]\right)}{\sum_{k=0}^{L} \varphi^k}, \text{ where} \tag{1.11}$$

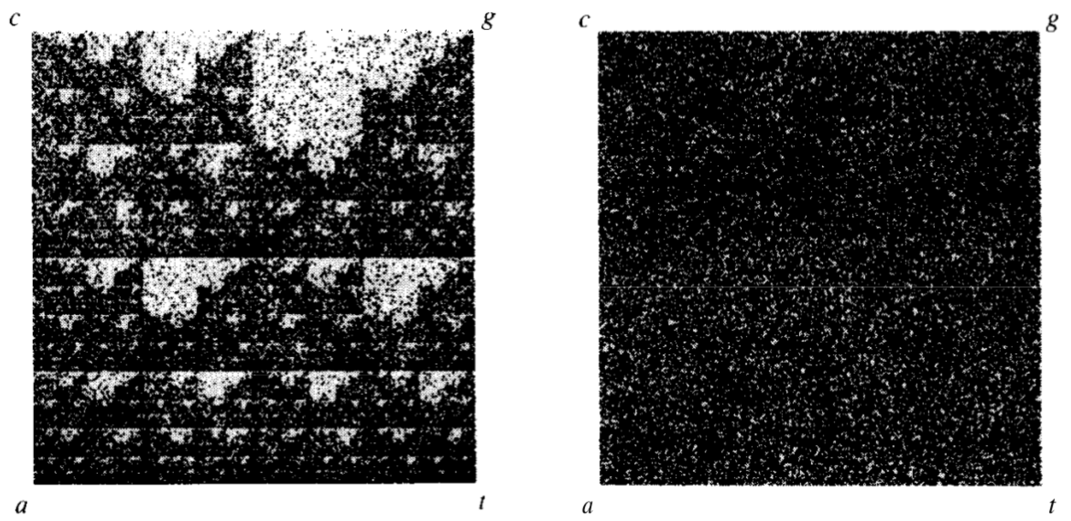

Figure 1.5: CGR obtained for the human β-globin sequence and for a random sequence of the same length

l is the length of the entire sequence, L the resolution, i.e. the k-mer length, φ is a smoothing parameter, and $c([i-k+1,i])$ is the number of occurrences of $(x_{i-k+1}\dots x_i)$, i.e. the suffix of length k that ends at position i. The main EP can also be redefined [21] [22] taking into account the prefixes, i.e. the words starting at position i, instead of the suffixes. This redefinition is equivalent to calculate $\hat{f}_{L,\varphi}(x_{|s|-i})$ for the reverse of s.

The cited works are not only of theoretical nature, two entropic profilers exist and proved to be useful for the discovery of patterns in genome. The former is a quadratic implementation based on the original definition and uses a truncated standard trie while the latter, named FastEP, is based on the redefinition in function of the prefix counts and improves their computation using a compressed suffix trie, i.e. a suffix tree, which leads to a linear time and space complexity. They are different also because they are based on two alternative post-processing, both of which are thought as a way to compare different parameter combinations. In the former, EP values are standardized with their arithmetic mean $m_{L,\varphi}$ and standard deviation $s_{L,\varphi}$, which are calculated taking into account the main EP for each position:

$$EP_{L,\varphi}(xi) = \frac{\hat{f}_{L,\varphi}(x_i) - m_{L,\varphi}}{s_{L,\varphi}}, \text{ where} \tag{1.12}$$

$$m_{L,\varphi} = \frac{1}{l}\sum_{i=1}^{l} \hat{f}_{L,\varphi}(x_i) \tag{1.13}$$

$$s_{L,\varphi} = \sqrt{\frac{1}{l-1}\sum_{i=1}^{l}\left(\hat{f}_{L,\varphi}(x_i) - m_{L,\varphi}\right)^2} \tag{1.14}$$

while, in the latter, they normalized with the maximum value:

$$EP_{L,\varphi}(xi) = \frac{\hat{f}_{L,\varphi}(x_i)}{max[\hat{f}_{L,\varphi}]} \tag{1.15}$$

As a practical example, Figure 1.6 shows the values of FastEP in the position window $[78440, 78540]$ of the Escherichia Coli K12 genome. For each position, several values are reported varying the k-mer length. The greatest peak is at position 78445 and is maximized when the k-mer length is 8. The respective motif is GCTGGTGG, which is a Chi site, a region that has an important biological meaning [25]. It is worthwhile noting that this exceptional pattern is discovered just by looking at the greatest pick in the position histogram without previous knowledge of the motif under study. This fact suggests the focus of this work. The local definition can be extend with the aim of developing new global sequence similarity measures for the prediction and classification of enhancers.

1.5 Organization of the document

First of all, in Chapter 2, the definition of entropic profile is further explored. The statistical properties of entropic profile are studied and particular attention is paid to role of the statistical background model. The adaptation of the well-known methods is discussed and their implementation in the proposed application `ep_sim` is also presented.

Then, in Chapter 3, the experimental setups are described and the main results are presented and compared with the state of the art paying again particular attention to the role of the statistical background model. Experiments on both simulated and real enhancers reveal that the multi-resolution property of the new methods make the similarity scores more robust with the change of the k-mer length.

Further improvements are needed to effectively address the most challenging situations. To this end, in Chapter 4, after recapping the main results, some possible future developments and new applications are suggested.

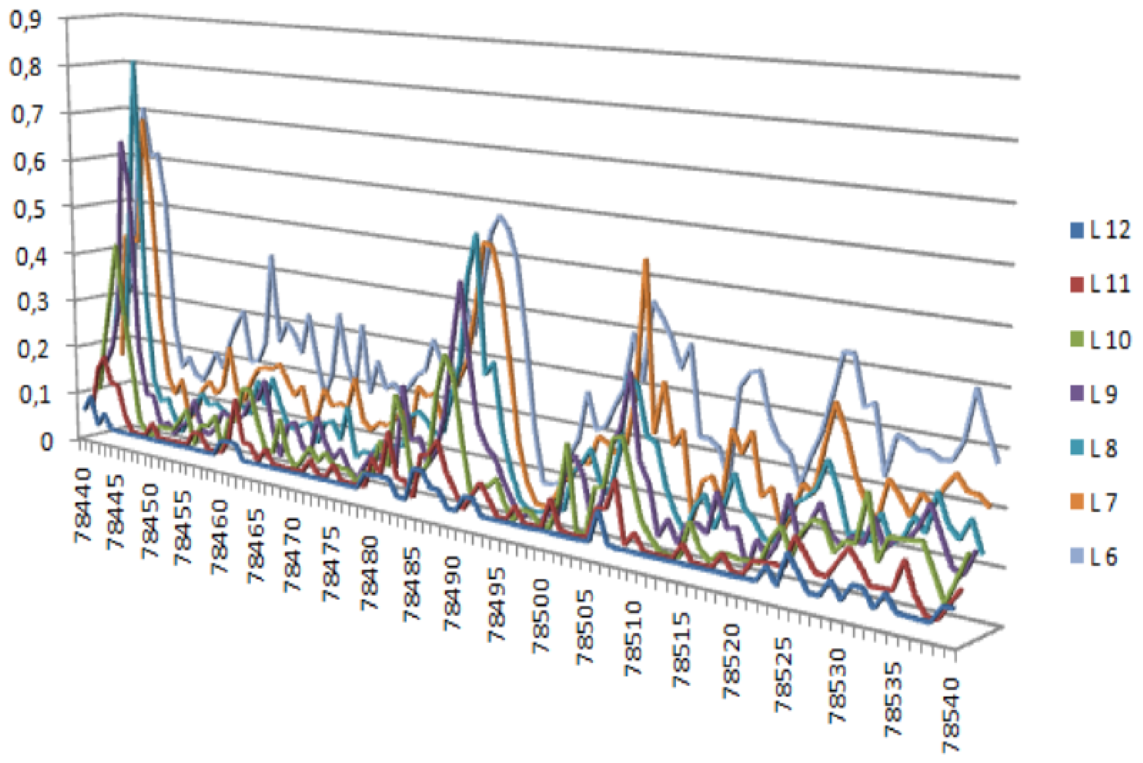

Figure 1.6: example of study of E-Coli genome from position 78440 for various values of the k-mer length

Chapter 2

Methods

2.1 Analysis of entropic profiles

2.1.1 Redefinition

The main EP function, see formula 1.11, consists of four components:

1. the linear combination of the number of occurrences $c([i-k+1,i])$ of the k-mer suffix ending at position i;
2. the exponential kernel $4^k\varphi^k$;
3. the normalization term $\sum_{k=0}^{L}\varphi^k$;
4. some constants, among which the length l of the entire sequence.

Given that what make it effective in pattern discovery are the first two, it can be redefined so as to point them out. In addition, since this study focuses on the global comparison of pair of sequences and not on the local sequence analysis for motif search, the original notation can be simplified. Thus, the simple entropy SE_w^S of a word $w = (w_1, ..., w_L)$ of length L in the sequence S of length l is:

$$SE_w^S = \frac{\sum_{k=1}^{L} a_k c_{w,k}}{\sum_{k=1}^{L} a_k}, \text{ where} \tag{2.1}$$

$c_{w,k}$ is the number of occurrences of the k-mer suffix $s_{w,k}$ and the exponential kernel has been replaced by the weights a_k and thus generalized. Indeed, the starting one will not necessarily work best in practice.

2.1.2 Choice of the kernel

Getting the idea from image smoothing techniques [26], one of the possible choices is a Gaussian function. Instead of setting each pixel new value to a weighted average of that

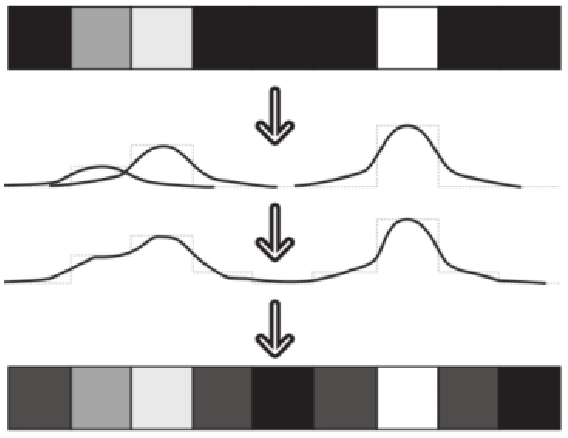
Figure 2.1: example of Gaussian blur on 1D pixel array.

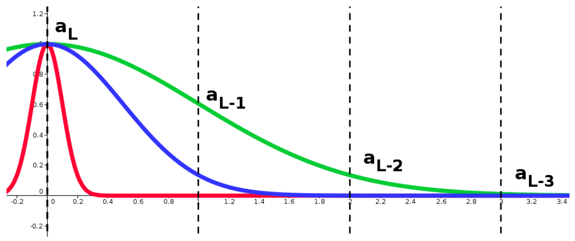

Figure 2.2: example of gaussian kernels. The smaller the standard deviation, the thinner the bell curve.

pixel neighbourhood, see Figure 2.1, each word count is replaced by the word entropy. The equation in one dimension is:

$$G(x) = ae^{-\frac{(x-b)^2}{2\sigma^2}} + d, \text{ where} \tag{2.2}$$

a is the height of the curve peak, b is the position of the center of the peak, σ (the standard deviation) controls the width of the bell, and d is the value that the function asymptotically approaches far from the peak. It is convenient to set a to 1, b and d to 0 obtaining

$$G(x) = e^{\frac{-x^2}{2\sigma^2}}, \text{ where} \tag{2.3}$$

$x = L - 1 - k$, i.e. longer suffixes are weighted by bigger coefficients. Fixing the height makes it easier to understand how the weights assigned to words of different length change as a function of the standard deviation. Three sample kernels are depicted in Figure 2.2: the green curve stands for $\sigma = 1$, the blue for $\sigma = 0.5$ and the red for $\sigma = 0.1$. The smaller the standard deviation is and the more it resembles the starting exponential kernel.

2.2 Statistical properties of entropic profiles

The statistical properties of the main EP have not been carefully studied yet. In the previous works [22], only the expectation of the main EP function has been explored. In addition, in [19] [20], the standardization is done with respect to the arithmetic mean and standard deviation but this approach is not enough because they depend only on the entire sequence and are not function of the single k-mer. The same holds for the normalized version [21] [22]. Thus, in the following, the statistical properties are derived for the proposed redefinition. Of course, since these formulations differ only in some constants, all the results can be easily adapted.

2.2.1 Preliminaries

The entire sequence $S = (X_1, X_2, ..., X_i, ..., X_l)$ can be modelled by a first-order homogeneous stationary Markov chain M1 without loss of generality [17]. In particular, if $x, y \in A, C, G, T$, the transition probability from letter x to y is denoted by $\pi(x, y)$ and the occurrence probabilities (stationary distribution) for the four letters by $\mu(x)$.

Thanks to the stationarity of the Markov chain, the probability that word w occurs does not depend on the position i. It is:

$$\mu(w) = \mu(w_1) \prod_{j=2}^{L} \pi(w_{j-1}, w_j), \text{ where} \tag{2.4}$$

$\mu(w_1)$ is the probability that the first letter occurs and $\pi(w_{j-1}, w_j)$ is the transition probability from letter w_{j-1} to w_j.

It is useful to define the variable $Y_i(w)$, which indicates if w occurs at position i:

$$Y_i(w) = \begin{cases} 1, & \text{if } (X_i,\ X_{i+1},...,X_{i+L-1}) = (w_1,\ w_2,...,w_L), \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

For each i, $Y_i(w)$ is a Bernoulli variable with parameter $\mu(w)$ so its expectation is

$$E[Y_i(w)] = \mu(w), \tag{2.6}$$

and its variance is

$$Var[Y_i(w)] = \mu(w)[1 - \mu(w)]. \tag{2.7}$$

This indicator provides a way to define the number of occurrences c_w of word w:

$$c_w = \sum_{i=1}^{l-L+1} Y_i(w) \tag{2.8}$$

2.2.2 Entropy expectation

The expected entropy $E[SE_w]$ can be directly derived from equations 2.4 and 2.8:

$$E[SE_w^S] = E\left[\frac{\sum_{k=1}^{L} a_k c_{w,k}}{\sum_{k=1}^{L} a_k}\right] = \frac{\sum_{k=1}^{L} a_k E\left[c_{w,k}\right]}{\sum_{k=1}^{L} a_k}, \text{ with} \tag{2.9}$$

$$E[c_{w,k}] = (l - k + 1)\mu(s_{w,k}) \tag{2.10}$$

2.2.3 Entropy variance

The variance $Var[SE_w^S]$ is important to take into account the dependence between entropies of overlapping words and calculate the standard deviation $sd[SE_w^S] = \sqrt{Var[SE_w^S]}$:

$$Var[SE_w^S] = Var\left[\frac{\sum_{k=1}^{L} a_k c_{w,k}}{\sum_{k=1}^{L} a_k}\right] = \frac{\sum_{k'=1}^{L}\sum_{k''=1}^{L} a_{k'}a_{k''}Cov\left[c_{w,k'}, c_{w,k''}\right]}{(\sum_{k=1}^{L} a_k)^2}, \text{ where} \tag{2.11}$$

the derivation of the covariance of the counts is tricky. There two cases which are explored in the next two subsections. In the former, $w' = w''$, in fact there is only one suffix of fixed length $k' = k'' \equiv k$, and $Cov\left[c_{w,k'}, c_{w,k''}\right] = Var(c_{w,k})$. In the latter, $w' = s_{w,k'} \neq w'' = s_{w,k''}$ so one word is the suffix of the other.

Case 1: variance of the count

This formula is derived and reported in [17]. The final expression is made of three terms which respectively take into account:

1. self-overlap of the word with itself;
2. partial self-overlap, the suffix of the word with its prefix or vice-versa;
3. disjoint occurrences.

Formally:

$$\begin{aligned} Cov[c_{w,k'}, c_{w,k''}] = Var[c_{w,k}] &= (l-k+1)\mu(w)(1-\mu(w)) \\ +2\mu(w)\sum_{d=1}^{k-1}(l-k-d+1)&\left[\varepsilon_{k-d}(w)\prod_{j=k-d+1}^{k}\pi(w[j-1], w[j]) - \mu(w)\right] \\ +2\mu^2(w)\sum_{t=1}^{l-2k+1}&(l-2k-t+2)\left[\frac{\pi^t(w[k], w[1])}{\mu(w[1])} - 1\right], \text{ where} \end{aligned} \tag{2.12}$$

$\varepsilon_u(w)$ is the asymmetric overlap indicator

$$\varepsilon_u(w) = \begin{cases} 1 & \text{if w[k-u+1...k] = w[1...u]} \\ 0 & \text{otherwise} \end{cases}, \tag{2.13}$$

and $t = d - k + 1$ and $\pi^t(w[k], w[1])$ is the probability that the last letter of w is separated from an occurrence of $w[1]$ by $t - 1$ letters.

Case 2: covariance of the counts of words of different length

In literature, this expression has not been derived yet. In the following, all the concepts and steps to get it are reported. The starting points are the simpler derivations of the variance and covariance of counts of words of equal length [17], besides grounds of probability theory [27]. As already mentioned, what it is all about is the ability of two words to overlap.

First of all, it can be assumed that $|w''| = k'' < |w'| = k'$ so, in this case, w'' is a suffix of w'. This assumption is without loss of generality because of the symmetry of the covariance, $Cov\left[c_{w,k'}, c_{w,k''}\right] = Cov\left[c_{w,k''}, c_{w,k'}\right]$. For simplicity of notation, let $c_{w,k'} = c_{w'}$ and $c_{w,k''} = c_{w''}$. The covariance can be expressed with respect to the random indicator variables, see formula 2.8, and developed by applying its well-known properties:

$$\begin{aligned} Cov\left[c_{w,k'}, c_{w,k''}\right] &= Cov\left[c_{w'}, c_{w''}\right] \\ &= Cov\left[\sum_{i=1}^{l-k'+1} Y_i(w'), \sum_{j=1}^{l-k''+1} Y_j(w'')\right] \\ &= \sum_{i=1}^{l-k'+1}\sum_{j=1}^{l-k''+1} Cov\left[Y_i(w'), Y_j(w'')\right] \\ &= \sum_{i=1}^{l-k''+1}\sum_{j=1, j\neq i}^{l-k''+1} Cov\left[Y_i(w'), Y_j(w'')\right] + \sum_{h=1}^{l-k''+1} Cov\left[Y_h(w'), Y_h(w'')\right], \end{aligned} \tag{2.14}$$

it is worthwhile noting that the indices vary between 1 and $l - k'' + 1$ so the last $k' - k''$ values of $Y_i(w')$ are all zero since there are not enough letters to make word w'. The representation of the former part of 2.14 in Figure 2.3 exemplifies the concept.

Second, formula 2.14 is interpreted so as to understand how its two parts can be conveniently reformulated:

1. the former stands for all the terms due to two words of different length that do not start at the same position;
2. the latter stands for all the terms due to two words of different length that start at the same position.

There are two convenient ways to reformulate the former and study overlaps, the first consists in fixing w' (the longest) and moving w'' (the shortest, i.e. its suffix), the second in fixing w'' (the shortest, i.e. its suffix) and moving w' (the longest), see Figure 2.4. The chosen one is the first because it is visually simpler. In particular, let d be the shift of the moving word w'' with respect to the fixed word w'. It is now possible to further analyse the two parts.

Part 1 of equation 2.14 To reformulate it,

- i is fixed and d varied in order to consider the positions before i and after i: in fact, $i - 1 + 1 + l - k'' + 1 - i = l - k'' + 1$;

w_1	w_2	·	·	·	·	·	·	·	·
w_1	w_2	w_3	·	·	·	·	·	·	·
·	w_1	w_2	·	·	·	·	·	·	·
·	w_1	w_2	w_3	·	·	·	·	·	·
·	·	w_1	w_2	·	·	·	·	·	·
·	·	w_1	w_2	w_3	·	·	·	·	·
·	·	·	w_1	w_2	·	·	·	·	·
·	·	·	w_1	w_2	w_3	·	·	·	·
·	·	·	·	w_1	w_2	·	·	·	·
·	·	·	·	w_1	w_2	w_3	·	·	·
·	·	·	·	·	w_1	w_2	·	·	·
·	·	·	·	·	w_1	w_2	w_3	·	·
·	·	·	·	·	·	w_1	w_2	·	·
·	·	·	·	·	·	w_1	w_2	w_3	·
·	·	·	·	·	·	·	w_1	w_2	·
·	·	·	·	·	·	·	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$
w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	·

Figure 2.3: the former term of formula 2.14

- the sums are exchanged over i and d according to the pairs of mathematical relations in equations 2.18, 2.19 and 2.16, 2.17.

$$\begin{aligned}\sum_{i=1}^{l-k''+1}\sum_{j=1,j\neq i}^{l-k''+1} Cov\left[Y_i(w'),Y_j(w'')\right] &= \sum_{i=1}^{l-k''+1}\left(\sum_{d=1}^{i-1} Cov\left[Y_i(w'),Y_{i-d}(w'')\right]\right.\\ &\qquad\left.+\sum_{d=1}^{l-k''+1-i} Cov\left[Y_i(w'),Y_{i+d}(w'')\right]\right)\\ &=\sum_{d=1}^{l-k''}\left(\sum_{i=d+1}^{l-k''+1} Cov\left[Y_i(w'),Y_{i-d}(w'')\right]\right.\\ &\qquad\left.+\sum_{i=1}^{l-k''+1-d} Cov\left[Y_i(w'),Y_{i+d}(w'')\right]\right).\end{aligned} \tag{2.15}$$

The first pair of inequalities, which regards the indices of the first term, is:

$$\begin{cases}1\le d\le i-1\\ 1\le i\le l-k''+1\end{cases} \tag{2.16}$$

$$\begin{cases}1\le d\le l-k''\\ d+1\le i\le l-k''+1\end{cases} \tag{2.17}$$

The second pair of inequalities, which regards the indices of the second term, is:

$$\begin{cases}1\le i\le l-k''+1\\ 1\le d\le l-i-k''+1\end{cases} \tag{2.18}$$

$$\begin{cases}1\le d\le l-k''\\ 1\le i\le l-d-k''+1\end{cases} \tag{2.19}$$

1.

·	·	w_1	w_2	w_3	w_4	w_5	w_6	·	·
w_1	w_2	w_3	·	·	·	·	·	·	·
·	w_1	w_2	w_3	·	·	·	·	·	·
·	·	w_1	w_2	w_3	·	·	·	·	·
·	·	·	w_1	w_2	w_3	·	·	·	·
·	·	·	·	w_1	w_2	w_3	·	·	·
·	·	·	·	·	w_1	w_2	w_3	·	·
·	·	·	·	·	·	w_1	w_2	w_3	·
·	·	·	·	·	·	·	w_1	w_2	w_3

2.

·	·	·	·	·	w_1	w_2	w_3	·	·	·	·	·
w_1	w_2	w_3	w_4	w_5	w_6	·	·	·	·	·	·	·
·	w_1	w_2	w_3	w_4	w_5	w_6	·	·	·	·	·	·
·	·	w_1	w_2	w_3	w_4	w_5	w_6	·	·	·	·	·
·	·	·	w_1	w_2	w_3	w_4	w_5	w_6	·	·	·	·
·	·	·	·	w_1	w_2	w_3	w_4	w_5	w_6	·	·	·
·	·	·	·	·	w_1	w_2	w_3	w_4	w_5	w_6	·	·
·	·	·	·	·	·	w_1	w_2	w_3	w_4	w_5	w_6	·
·	·	·	·	·	·	·	w_1	w_2	w_3	w_4	w_5	w_6

Figure 2.4: two ways of studying overlaps

Part 2 of equation 2.14 Part 2 simply represents the case $d = 0$.

Under M1 model (or greater), the indicators $Y_i(w')$ and $Y_j(w'')$ are not independent, not even if the corresponding positions are more than k' letters away from each other [17]. Thus,

$$Cov\,[Y_i(w'), Y_j(w'')] = E[Y_i(w')Y_j(w'')] - E[Y_i(w')]E[Y_j(w'')] \tag{2.20}$$

may be different from zero. Especially, there are three cases:

- left shift, $d \geq 1$;
- right shift, $d \geq 1$;
- zero shift, $d = 0$.

Left shift

$$Cov\,[Y_i(w'), Y_{i-d}(w'')] = E[Y_i(w')Y_{i-d}(w'')] - E[Y_i(w')]E[Y_{i-d}(w'')], \text{ where} \tag{2.21}$$

the first term comprehends two parts that respectively represent:

1. two overlapping words, the latter (red words in Figure 2.4) starts before the beginning and ends before the end of the former: prefix - suffix overlap;

2. two non overlapping words.

In formulas:

$$E[Y_i(w')Y_{i-d}(w'')] = \begin{cases} \varepsilon^{left}_{k''-d}(w'', w')\mu(w'') \prod_{j=k'-k''-d+1}^{k'} \pi(w'_{j-1}, w'_j) & \text{if } 1 \leq d < k'' \\ \mu(w'')\mu(w') \left[\frac{\pi^{d-k''+1}(w''_{k''}, w'_1)}{\mu(w'_1)}\right] & \text{if } d \geq k'' \end{cases}, \text{ where} \tag{2.22}$$

$\varepsilon^{left}_u(w'', w')$ is the asymmetric overlap indicator

$$\varepsilon^{left}_u(w'', w') = \begin{cases} 1 & \text{if } w''[k'' - u + 1...k''] = w'[1...u] \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

$$E[Y_i(w')]E[Y_{i-d}(w'')] = \mu(w')\mu(w''). \tag{2.24}$$

Right shift Analogously (but not symmetrically),

$$Cov\,[Y_i(w'), Y_{i+d}(w'')] = E[Y_i(w')Y_{i+d}(w'')] - E[Y_i(w')]E[Y_{i+d}(w'')], \text{ where} \tag{2.25}$$

the first term comprehends three parts that respectively represent:

1. two overlapping words, the latter (blue words in Figure 2.4) starts after the beginning and ends before the end of the former: substring - string overlap;
2. two overlapping words, the latter (green words in Figure 2.4) starts before the end of the former and ends after it: substring - prefix overlap;
3. two non overlapping words.

$$\begin{aligned} &E[Y_i(w')Y_{i+d}(w'')] \\ &= \begin{cases} \varepsilon^{right}_{k'-d}(w', w'')\mu(w') & \text{if } 1 \leq d \leq k' - k'' \\ \mu(w')\varepsilon^{right}_{k'-d}(w', w'') \prod_{j=k'-d+1}^{k''} \pi(w''_{j-1}, w''_j) & \text{if } k' - k'' < d < k' \\ \mu(w')\mu(w'') \left[\frac{\pi^{d-k'+1}(w'_{k'}, w''_1)}{\mu(w''_1)}\right] & \text{if } d \geq k' \end{cases}, \text{ where} \end{aligned} \tag{2.26}$$

$\varepsilon^{right}_u(w', w'')$ is the asymmetric overlap indicator

$$\varepsilon^{right}_u(w', w'') = \begin{cases} 1 & \text{if } u < k'' \wedge w'[k' - u + 1...k'] = w''[1...u] \\ 1 & \text{if } u \geq k'' \wedge\, w'' \text{ is a substring of } w' \\ 0 & \text{otherwise} \end{cases} \tag{2.27}$$

$$E[Y_i(w')]E[Y_{i+d}(w'')] = \mu(w')\mu(w''). \tag{2.28}$$

Zero shift This case considers two overlapping words starting at the same position (black words in the coloured picture of Figure 2.4), the latter ends before the end of the former: prefix - string overlap.

$$E[Y_h(w')Y_{h+d}(w'')] = E[Y_h(w')Y_{h+0}(w'')] = \mu(w') * 1 + (1 - \mu(w')) * 0 = \mu(w') \tag{2.29}$$

Putting them all together Equations 2.21, 2.25 and 2.29 do not depend on i so substituting them in equation 2.14 leads to

$$\begin{aligned}
&\sum_{h=1}^{l-k''+1} Cov\left[Y_h(w'), Y_h(w'')\right] \\
&+ \sum_{d=1}^{l-k''} \left(\sum_{i=d+1}^{l-k''+1} Cov\left[Y_i(w'), Y_{i-d}(w'')\right] + \sum_{i=1}^{l-k''+1-d} Cov\left[Y_i(w'), Y_{i+d}(w'')\right] \right) \\
&= (l - k'' + 1) Cov\left[Y_h(w'), Y_h(w'')\right] \\
&+ \sum_{d=1}^{l-k''} \left((l - k'' + 1 - d) Cov\left[Y_i(w'), Y_{i-d}(w'')\right] + (l - k'' + 1 - d) Cov\left[Y_i(w'), Y_{i+d}(w'')\right] \right).
\end{aligned} \tag{2.30}$$

Finally, the formula for the covariance of the counts of two words with different length can be derived:

$$\begin{aligned}
Cov\left[c_{w,k'}, c_{w,k''}\right] &= (l - k'' + 1)(\mu(w') - \mu(w')\mu(w'')) \\
&+ \sum_{d=1}^{k'-k''} (l - k'' + 1 - d)\mu(w')(\varepsilon_{k'-d}^{right}(w', w'') - \mu(w'')) \\
&+ \sum_{d=k'-k''+1}^{k'} (l - k'' + 1 - d)\mu(w') \left[\varepsilon_{k'-d}^{right}(w', w'') \prod_{j=k'-d+1}^{k''} \pi(w''_{j-1}, w''_j) - \mu(w'') \right] \\
&+ \sum_{d=1}^{k''} (l - k'' + 1 - d)\mu(w'') \left[\varepsilon_{k''-d}^{left}(w'', w') \prod_{j=k'-k''-d+1}^{k'} \pi(w'_{j-1}, w'_j) - \mu(w') \right] \\
&+ \sum_{d=k''}^{l-k''} (l - k'' + 1 - d)\mu(w'')\mu(w') \left[\frac{\pi^{d-k''+1}(w''_{k''}, w'_1)}{\mu(w'_1)} - 1 \right] \\
&+ \sum_{d=k'}^{l-k''} (l - k'' + 1 - d)\mu(w')\mu(w'') \left[\frac{\pi^{d-k'+1}(w'_{k'}, w''_1)}{\mu(w''_1)} - 1 \right].
\end{aligned} \tag{2.31}$$

For the sake of simplicity, as done in [13], the last two terms, i.e. the non-overlapping terms, will be neglected thereby assuming that the occurrence of non-overlapping words is independent of the sequence in between. The meaning of each of the terms is recapped in the next example.

Example 2.2.1. Take the string AGCCGAGCCGGGA of length 13. Under the M1 model, the stationary probabilities are $\mu(A) = \frac{3}{13}$, $\mu(C) = \frac{4}{13}$, $\mu(G) = \frac{6}{13}$ and $\mu(T) = \frac{0}{13}$ and the transition probabilities are $\mu(AG) = \frac{2}{12} = \frac{1}{6}$, $\mu(GC) = \frac{1}{6}$, $\mu(CC) = \frac{1}{6}$, $\mu(CG) = \frac{1}{6}$, $\mu(GA) = \frac{1}{6}$ and $\mu(GG) = \frac{1}{6}$. To calculate the covariance of the word GCCG and its suffix CCG:

- calculate their probabilities: $\mu(GCCG) = \frac{1}{468}$ and $\mu(CCG) = \frac{1}{117}$;
- study their overlaps and calculate the probabilities of their clumps. The two situations such that the overlap indicator is not zero are shown in Figure 2.5, both shifts d_1, d_2 are equal to 1 and the respective clumps are GCCG and CCGCCG. With regard to their probabilities, $\mu(GCCG) = \frac{1}{468}$ and $\mu(CCGCCG) = \frac{1}{25272}$.

G	C	C	G
·	C	C	G

·	·	G	C	C	G
C	C	G	·	·	·

Figure 2.5: study of the overlaps of GCCG and CCG

Neglecting the non-overlapping terms:

$$\begin{aligned}
Cov(GCCG, CCG) &= (l - k'' + 1)(\mu(GCCG) - \mu(GCCG)\mu(CCG)) \\
&\quad +(l - k'' + 1 - d_1)(\mu(GCCG) - \mu(GCCG)\mu(CCG)) \\
&\quad +(l - k'' + 1 - d_2)(\mu(CCGCCG) - \mu(GCCG)\mu(CCG)) \\
&= (13 - 3 + 1)(\frac{1}{468} - \frac{1}{468}\frac{1}{117}) \\
&\quad +(13 - 3 + 1 - 1)(\frac{1}{468} - \frac{1}{468}\frac{1}{117}) \\
&\quad +(13 - 3 + 1 - 2)(\frac{1}{25272} - \frac{1}{468}\frac{1}{117}) \\
&= 0.045.
\end{aligned} \tag{2.32}$$

2.3 Reverse complement

To address the fact that enhancers act independently of their orientation as a result of looping, recall Figure 1.3, the definition of entropy can be extended to include the reverse complement, indicated as w^{rc} for every word w. The reverse complement of a DNA sequence is formed by reversing the letters, interchanging A and T and interchanging C and G. Thus, the reverse complement of ACCTGAG is CTCAGGT.

$$SE^S_{w+w^{rc}} = \frac{\sum_{k=1}^L a_k c_{w,k} + \sum_{k=1}^L a_k c_{w^{rc},k}}{2\sum_{k=1}^L a_k}. \tag{2.33}$$

The new expectation and variance are studied in the following.

2.3.1 Entropy expectation

The expected entropy $E[SE^S_{w+w^{rc}}]$ is:

$$\begin{aligned} E[SE^S_{w+w^{rc}}] &= E\left[\frac{\sum_{k=1}^{L} a_k c_{w,k} + \sum_{k=1}^{L} a_k c_{w^{rc},k}}{2\sum_{k=1}^{L} a_k}\right] \\ &= \frac{\sum_{k=1}^{L} a_k E\left[c_{w,k}\right] + \sum_{k=1}^{L} a_k E\left[c_{w^{rc},k}\right]}{2\sum_{k=1}^{L} a_k}. \end{aligned} \tag{2.34}$$

$$E[c_{w,k}] = (l-k+1)\mu(s_{w,k}) \tag{2.35}$$

2.3.2 Entropy variance

The variance $V[SE^S_{w+w^{rc}}]$ is:

$$V[SE^S_{w+w^{rc}}] = V\left[\frac{\sum_{k=1}^{L} a_k c_{w,k} + \sum_{k=1}^{L} a_k c_{w^{rc},k}}{2\sum_{k=1}^{L} a_k}\right], \text{ where} \tag{2.36}$$

the numerator consists of the terms due to each possible pair:

$$\begin{aligned} &\sum_{k'=1}^{L}\sum_{k''=1}^{L} a_{k'}a_{k''}Cov[c_{w,k'}, c_{w,k''}] + \sum_{k'=1}^{L}\sum_{k''=1}^{L} a_{k'}a_{k''}Cov[c_{w^{rc},k'}, c_{w^{rc},k''}] \\ &+ \sum_{k'=1}^{L}\sum_{k''=1}^{L} a_{k'}a_{k''}Cov[c_{w,k'}, c_{w^{rc},k''}]. \end{aligned} \tag{2.37}$$

With respect to the properties studied in the previous section, there is a new case: the covariance of counts of different words of equal length, i.e. two suffixes of equal length, one of the string and one of its reverse complement.

Covariance of the counts of different words of equal length

This case is reported in [17].

$$\begin{aligned} Cov[c_{w,k}, c_{w',k}] = &+\mu(w)\sum_{d=1}^{k-1}(l-k-d+1)\left[\varepsilon_{k-d}(w,w')\prod_{j=k-d+1}^{k}\pi(w'[j-1],w'[j]) - \mu(w')\right] \\ &+\mu(w')\sum_{d=1}^{k-1}(l-k-d+1)\left[\varepsilon_{k-d}(w',w)\prod_{j=k-d+1}^{k}\pi(w[j-1],w[j]) - \mu(w)\right] \\ &+\mu(w)\mu(w')\sum_{t=1}^{l-2k+1}(l-2k-t+2)\left[\frac{\pi^t(w[k],w'[1])}{\mu(w'[1])} + \frac{\pi^t(w'[k],w[1])}{\mu(w[1])} - 2\right] \\ &+(l-k+1)\mu(w)\mu(w') \end{aligned} \tag{2.38}$$

$\varepsilon_u(w, w')$ is the asymmetric overlap indicator

$$\varepsilon_u(w, w') = \begin{cases} 1 & \text{if } \; w[k-u+1...k] = w'[1...u] \\ 0 & \text{otherwise} \end{cases}, \tag{2.39}$$

and $t = d - k + 1$ and $\pi^t(w[k], w[1])$ is the probability that the last letter of w is separated from an occurrence of $w'[1]$ by $t - 1$ letters.

2.3.3 Alternative approaches

Weighting the reverse complement as the main word may not be the best choice because this would imply a loss of information due to the fact that the score of every word and the respective reverse complement would be the same. To exemplify the concept,

Example 2.3.1. Take the two pairs of strings AAAA, TTTT and ACAC, TGTG, one the reverse complement of the other. Their entropies are respectively 5, 5 and 1, 9. Taking into account the reverse complement, their entropies become 5, 5 and 5, 5. There is an evident loss of information.

To remedy, a viable approach consists in taking the max of the two entropies. Even though the score of a word and its reverse complement would still be the same, the fact that only the strongest signal is taken makes the effect of exceptional words more incisive. In addition, the formula of the covariance becomes simpler because there is no need to further complicate it and the terms due to the covariance between a word and its reverse complement and between their suffixes are no more needed.

Example 2.3.2. Take the two pairs of strings AAAA, TTTT and ACAC, TGTG, one the reverse complement of the other. Their entropies are respectively 5, 5 and 1, 9. Taking into account the reverse complement with the max approach, their entropies become 5, 5 and 9, 9. The negative smoothing effect is partially avoided.

This solution is only one of the possibilities. In N_2 [13], the k-mer counts from the reverse and forward strand can be combined in many ways. There are four options: bothStrands, to calculate the pairwise score using both strands from the input sequences, mean, min and max.

2.4 Importance of the model

In the previous section, the DNA sequence has been modelled by a first order stationary Markov chain model but neither the motivations nor the implications have been analysed because theoretically there is no difficult to adapt the presented results under a different model. Indeed, they would be simpler under the Bernoulli model and the more general model Mm of order $m \geq 2$ can be considered as a model M1 on a bigger alphabet.

Nonetheless, modelling is a central concept not only with regard to the statical properties of the word counts and entropies but also with regard to its application in an alignment-free schema.

2.4.1 Motivations

As clearly explained in [17], the aim of a model is to provide a satisfying description of the phenomenon under study. The description provides predictions which can be used in two ways:

1. if the model appears to be poor, predictions can be used to define expected behaviours,
2. if the model is more refined and precise, predictions can be trusted to derive biological conclusions.

The biological application of this work deals with motif statistics about enhancers and transcriptional factor binding sites. The aim of these statistics is to detect motifs that have unexpected entropies and derive a similarity measure that points them out effectively. These statistics are based on Markov models that are biologically meaningless since considering a DNA sequence as a succession of random nucleotides, the distribution of one nucleotide depending only on few preceding ones, is very simplistic.

As a consequence, predictions provided by these models are of the first type, i.e. the model will be used as a benchmark. On the one hand, motifs, with entropy, frequency or distribution close to the prediction of the model, will be supposed to be of no biological interest because it would mean that the model has captured enough information on the sequence to correctly mimic the entropy of the word. On the other hand, motifs that have unexpected behaviour, presuming that this behaviour is due to some specific biological function, will be interesting because of their exceptionality compared with the chosen model.

As a result, it is out of the scope considering more sophisticated models so that they can correctly predict what has been observed.

2.4.2 Implications

The influence of the background Markov model order has been quantitatively studied in [13]. They randomly selected sequences from the mouse genome and implanted motifs. Across all methods analysed, the first order Markov model produced the best results, see Table 2.1.

Markov model order variations, m4r2, k=5

	AUC ROC		AUC PR		5%-Precision	
Markov model	M0	M1	M0	M1	M0	M1
$D2$	0.51	0.51	0.50	0.50	0.52	0.52
$D2z$	0.57	**0.67**	0.54	**0.62**	0.57	**0.67**
$D2^*$	0.58	**0.82**	0.55	**0.81**	0.58	**0.91**
$N2^*$	0.61	**0.90**	0.58	**0.89**	0.63	**0.94**

Table 2.1: Influence of background Markov model order on pairwise scores. The data is obtained from randomly selected sequences from the mouse genome.

In fact, a Bernoulli background model may be insufficient to estimate word probabilities unless the sequence is artificial and random generated. For instance CpG dinucleotides are very rare in mammalian genomic sequences and this cannot be captured by a Bernoulli model.

Nevertheless, generally, the optimal order for enhancer sequences is an unknown function of organism complexity and sequence length. This concept will be stressed in the next chapter.

2.5 New alignment-free measures derived from entropic profiles

Entropies and counts are very much alike as illustrated in the next example. This suggests that the adaptation of the state-of-the-art measures can be done replacing the vector of k-mer counts with the vector of entropies.

Example 2.5.1. Take the string AGCCGAGCCGGGA and consider its 4-mer GCCG. Let $\sigma = 0.5$ be the standard deviation of the Gaussian kernel so $G(x) = e^{-x^2}$, the weights are 1 if $L = 4$, 0.37 if $L = 3$, 0.02 if $L = 2$, and 0 if $L = 1$ and the normalization term is $1 + 0.37 + 0.02 + 0 = 1.372$. While the word count is 2, the word entropy is $\frac{1c_{GCCG}+0.37c_{CCG}+0.02c_{CG}+0c_{G}}{1.372} = \frac{1*2+0.37*2+0.02*2+0*6}{1.372} = 2.03$.

Consider two genome sequences A and B and let A_w and B_w be the entropies of word w in A and B. The definitions of D_2, D_2^* and D_2^z do not change.

$$D_2^{EP} = \sum_w A_w B_w \tag{2.40}$$

$$D_2^{*EP} = \sum_w \frac{(A_w - E[AB_w])(B_w - E[AB_w])}{\sqrt{Var[AB_w]}\sqrt{Var[AB_w]}} = \sum_w \frac{(A_w - E[AB_w])(B_w - E[AB_w])}{Var[AB_w]} \tag{2.41}$$

$$D_2^{zEP} = \frac{D_2 - E[D_2]}{\sqrt{Var(D_2)}} \tag{2.42}$$

While the implementation of D_2^{EP} is straightforward, D_2^* and D_2^z are based on the statistical properties of entropies. The theory developed in the previous section is preliminary to the implementation of D_2^* but not to the implementation of D_2^z, which is based on the measurement of the number of standard deviations by which the observed value of D_2 deviates from the mean and, for this reason, its generalization is not developed in this work. Instead, taking the idea from N_2 [13], the statistical properties of the entropies pave the way also to the next measure, which is a variant of D_2^* such that the standardization is no longer performed under the statistical model estimated from the concatenation of the two sequences,

$$EP_2 = \sum_w \frac{(A_w - E[A_w])(B_w - E[B_w])}{\sqrt{Var[A_w]}\sqrt{Var[B_w]}}, \tag{2.43}$$

the background model is estimated separately for every sequence in order to cut down computational costs. Many other variants and measures can be taken into account but to rehash all the existing methods is out of scope of this work.

2.6 Efficient implementation

2.6.1 Starting points

The two already existing entropic profilers have been already described in section 1.4. The programs are respectively in C and Java. The first is implemented on the basis of the original main EP function with words ending at position i while the second, for ease of understanding, redefines it with the words starting at position i.

To implement the proposed similarity measures, the first step is the computation of the entropies for every word of the chosen resolution L, which are 4^L in total. Given that the existing programs are thought as tool for the discovery of patterns in a certain region of the entire sequence, their straightforward adaptation consists in asking them to compute the entropies for a window of length of the entire sequence. The output is a vector of size the length of the entire sequence containing the entropies of the words ending at the respective positions.

Nevertheless, the above solution is not good because the output vector would contain many equal entries associated to different positions but to the same string so there would not be any entry associated to the entropy of words not present in the analysed sequence. The second remark implies two negative consequences:

- space inefficiency;
- the respective entropy should be post-processed or, erroneously, set to 0. To make the concept clear, take a look at the next example.

Example 2.6.1. Take the string CGTCTAGCTTTAGC and consider its 5-mer ATAGC. Let $\sigma = 1$ be the standard deviation of the Gaussian kernel so $G(x) = e^{-\frac{x^2}{2}}$, the weights are 1 if $L = 5$, 0.61 if $L = 4$, 0.14 if $L = 3$, 0.01 if $L = 2$ and 0 if $L = 1$ and the normalization term is $1 + 0.61 + 0.14 + 0.01 + 0 = 1.76$. While the word count is 0, the word entropy is $\frac{1c_{ATAGC}+0.61c_{TAGC}+0.14c_{AGC}+0.01c_{GC}+0c_C}{1.76} = \frac{1*0+0.61*2+0.14*2+0.01*2+0*4}{1.76} = 0.86 \neq 0$.

My application is developed in C++ and designed in order to address the above issues. Like ALF, it is based on SeqAn [28] [29]. This is an open source C++ library of efficient algorithms and data structures for the analysis of sequences with the focus on biological data. It applies a unique generic design that guarantees high performance, generality, extensibility, and integration with other libraries. It is easy to use and simplifies the development of new software tools with a minimal loss of performance.

2.6.2 Computation of entropies

My application does not rely on the previous developed programs and compute the entropies within SeqAn. `String` is one of its data structures, a generic and dynamic array which replaces the vector of the standard library. It can be used to store arbitrary values or large biologically sequences. For my purposes, this structure can be used to store counts and entropies for every word. Every word is associated to one of its entries thanks to an hash, more precisely the lexicographical rank of the word among all the words of that length.

The procedure for the computation of entropies is now presented:

- let L be the resolution, all the counts needed to compute the entropies for every word of length L can be retrieved with a single scan of the sequence and stored in a vector of `String` of size L, each of size equal to the total number of words of length k, $1 \leq k \leq L$;
- the `String` of entropies of size the number of words of length L can be computed from the values in the `String`s of counts.

The counts of the suffixes can be retrieved without using the hash based on the lexicographical rank. Given that the words are ranked by the lexicographical order, the suffixes of the word with index i can be retrieved thanks to the next relation on the indices:

$$j = i \% kMerNumber, \text{ where} \tag{2.44}$$

$kMerNumber$ is the number of strings of length k, i.e. 4^k, k is the length of the suffix and j its index in the `String`s with all the words of length k. In other words, the suffixes can be retrieved iterating the respective `String` in a circular buffer fashion.

Example 2.6.2. Let $L = 2$. Figure 2.6 depicts the relation between the indices in this simple case.

A	**C**	**G**	**T**

AA	AC	AG	AT	C**A**	C**C**	C**G**	C**T**	GA	GC	GG	GT	TA	TC	TG	TT

Figure 2.6: relation between the indices of a word and its suffixes

Significant relations can be proved also for the prefixes and the reverse complement. For example, given a `String` storing the entropies or the counts for every k-mer, the index of the reverse complement of the word of index i is $l - 1 - i$, which implies that the vector of entropies is symmetric if the reverse complement is considered.

2.6.3 Design choices

The proposed application, named `ep_sim`, requires two arguments and five options:

1. the filename of the sequence file;

2. the filename of the output file with the score matrix;
3. L, the L-mer length, i.e. the max resolution. Default is 6;
4. P, the standard deviation of the gaussian kernel. Default is 0.5;
5. B, the background type. Default is M1;
6. D, the measure type. Default is EP_2;
7. RC, the flag to take into account the reverse complement. Default is false.

The main is described by the flowchart in 2.7. Following an object oriented approach, I collected the methods in this way:

1. `EntropicProfiler` class: it manages the computation of the entropies;
2. `KernelGen` class: it manages the creation of kernel;
3. `StatModels` class: it manages the computation of the statistical models;
4. `ScoreFiller` class: it manages the score matrix.

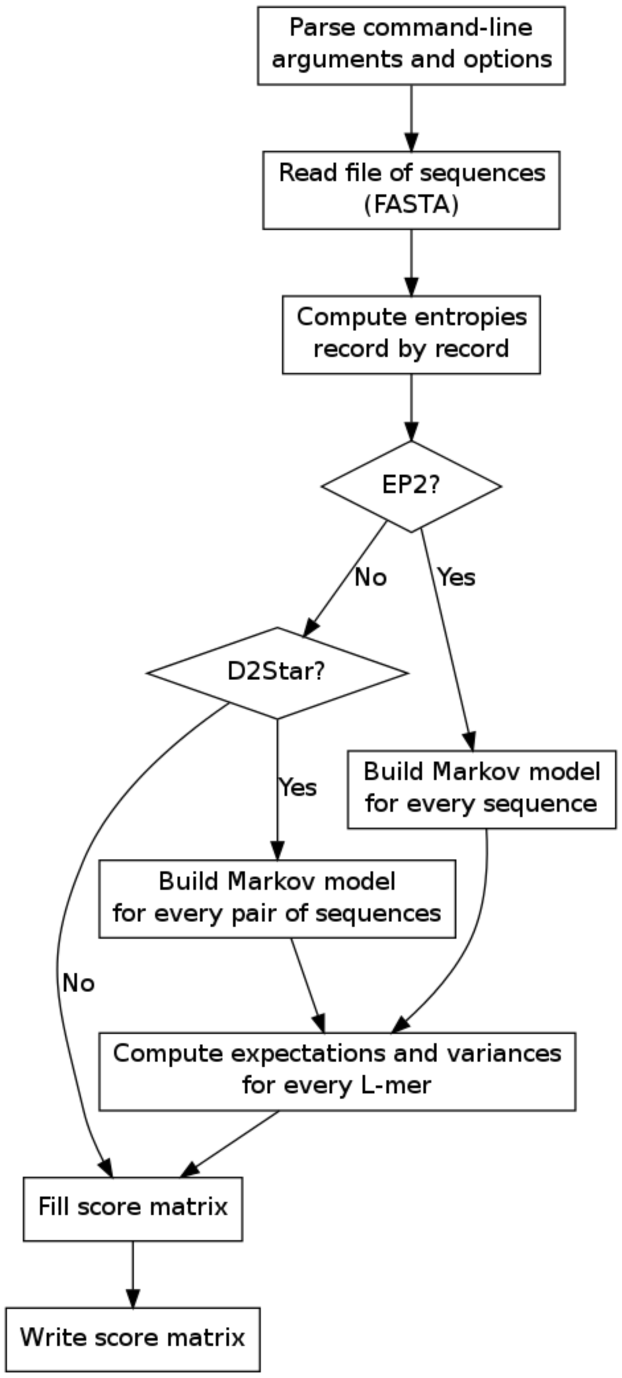
Parse command-line arguments and options
Read file of sequences (FASTA)
Compute entropies record by record
EP2?
No
Yes
D2Star?
Yes
No
Build Markov model for every sequence
Build Markov model for every pair of sequences
Compute expectations and variances for every L-mer
Fill score matrix
Write score matrix

Figure 2.7: main of the application

Chapter 3

Experimental results

This chapter deals with the testing procedures for the experimental verification of the proposed methods. The three main experimental setups are described and the principal experiments are reported for each of them. The experimental conditions, results and remarks are listed for each experiment.

The performance assessment process is the same of [30], [31], [13], [14], [15]. In practice, there are two sets of sequences: the negative and the positive set. The sequences in the former are dissimilar while those in the latter are similar. The entire process is made of two steps:

1. the pairwise scores are computed between all pairs in the negative and in the positive set. This step is performed by the scoring application, e.g. `ALF` or `ep_sim`;

2. the scores are sorted in one combined list and the positive predictive value (PPV) is calculated. This step is performed by a Java program, `compare`, that compares the scores of the matrices of the two sets.

The best PPV is 1 and means a perfect separation between negative and positive sets while a PPV close to 0.5 implies no statistical power.

Moreover, two C++ programs, `ppv_plotter` and `ppv_compare`, have been developed. They furnish the automatic plot of the performance curves in Gnuplot [32], which is a portable command-line driven graphing utility. These utilities plus some bash scripts proved to be very useful to study the effects of the main parameters, such as the length of the entire sequence, the k-mer length and the Gaussian standard deviation.

As it will be shown, the choice of the background model is so crucial that different measures have to be compared without changing it. For this reason, the results are presented for the pair of similarity measures EP_2 and N_2, both of which compute it on the single sequences. The results for the pair D_2^{*EP} and D_2^* have a similar trend but performances are generally higher. In fact, the estimation of the pairwise background for each pair of sequence is more accurate but also computationally more expensive [13].

3.1 Pattern transfer on simulated data

Background sequences are generated with some Perl scripts. Those in the negative set are randomly generated while those in the positive set are obtained by implanting some motifs in the random sequences of the negative set. Patterns are implanted mimicking the exchange of genetic material via the pattern transfer model or the revised one [16]. The former model implants only strings of the same length, e.g 5, while the latter is more realistic and implants strings of different length, e.g. 4, 5 and 6. Implanted motifs can be chosen by the user.

3.1.1 Experiment 1: M0 VS M1 background model

The objective is the study of the influence of the background model. The best one will be chosen in the next experiments of this section.

Experimental conditions

The experiment is performed varying the insertion probability, the entire sequence length and the k-mer length. The parameters for the sequence generator are now summed up:

- **Background**: random with identically distributed probabilities, i.e. $p_A = \frac{1}{4}$, $p_C = \frac{1}{4}$, $p_G = \frac{1}{4}$ and $p_T = \frac{1}{4}$;
- **Implanted motifs**: TACCAG;
- **Insertion probabilities** p_i: 0.009, 0.004;
- **Length of the sequences**: 200 300 400 500 1000 if $p_i = 0.009$, 250 500 1000 2000 3000 4000 5000 10000 if $p_i = 0.004$;
- **Number of sequences per file**: 25.

The parameters for the scoring application `ep_sim` are the following:

- **Background model**: M0, M1;
- **Measure**: EP_2 without reverse complement;
- **Standard deviation**: 0.6;
- **k-mer length**: its range is $[3, 8]$;.

The standard deviation is 0.6 so the two biggest weights are 1 and 0.25.

Results and remarks

Figures 3.1 and 3.2 are two selected plots corresponding to the sequence lengths 400 and 1000. This experiment highlights that the choice of the background is important and not straightforward. M0 (black curve) is slightly better with random generated sequences even if the recommended model is M1 (yellow curve), as explained in section 2.4. This result also suggests that different measures have to be compared without changing the sequence background so as to better understand what really influences the results. Even though only one motif has been implanted, these results are general as proved by other experiments involving the implantation of more than one motif. They are not reported because do not provide additional information.

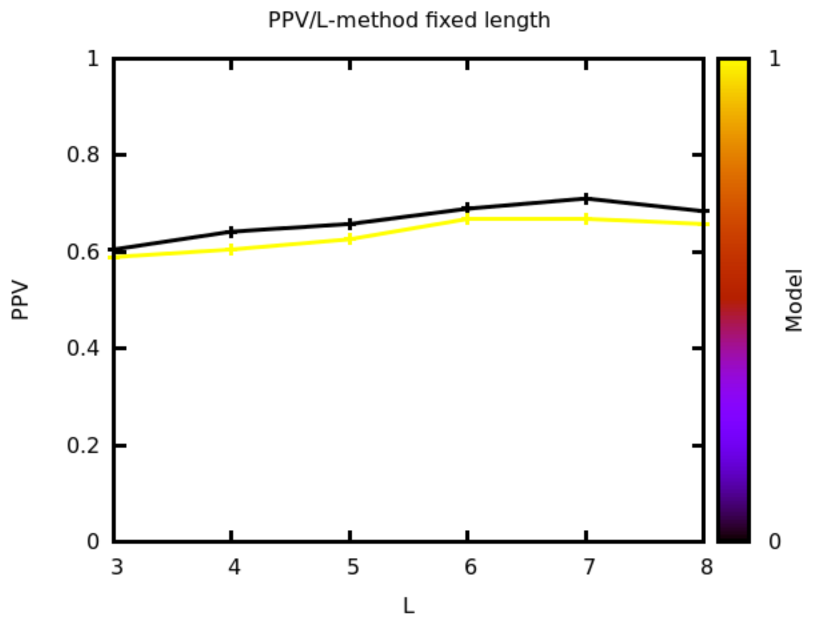

Figure 3.1: experiment 1: sequence length 400, $p_i = 0.009$

3.1.2 Experiment 2: varying the standard deviation of the Gaussian kernel

The objective is the study of the influence of the standard deviation of the Gaussian kernel.

Experimental conditions

This experiment is performed varying not only the standard deviation but also the insertion probability, the entire sequence length and the k-mer length. The parameters for the sequence generator are now summed up:

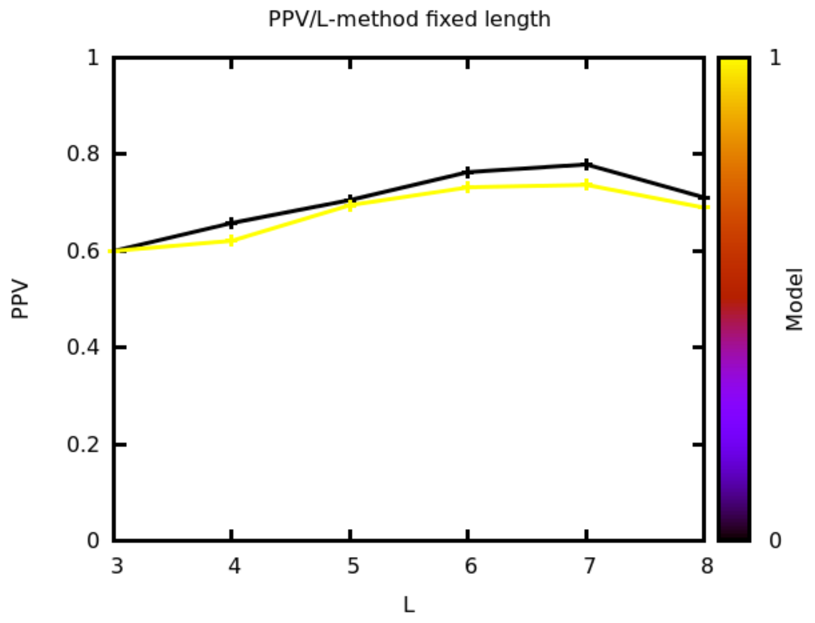

Figure 3.2: experiment 1: sequence length 1000, $p_i = 0.004$

- **Background**: random with identically distributed probabilities, i.e. $p_A = \frac{1}{4}$, $p_C = \frac{1}{4}$, $p_G = \frac{1}{4}$ and $p_T = \frac{1}{4}$;
- **Implanted motifs**: GCATA, ACCT, TAACGT, GATC, TTGAAC;
- **Insertion probabilities** p_i: 0.01, 0.002;
- **Length of the sequences**: 200 300 400 500 1000 if $p_i = 0.01$, 250 500 1000 2000 3000 4000 5000 10000 if $p_i = 0.002$;
- **Number of sequences per file**: 25.

Implanted motifs are very similar and of different length. The parameters for the scoring application `ep_sim` are the following:

- **Background model**: M0;
- **Measure**: EP_2 without reverse complement;
- **Standard deviation**: its range is $[0.1, 1.5]$;
- **k-mer length**: its range is $[3, 8]$.

If the standard deviation is 0.1, the Gaussian bell is so thin that EP_2 is equivalent to N_2. Otherwise, if the standard deviation is 1.5, the four biggest weights are 1, 0.80, 0.41 and 0.13 and their influence is not obvious.

Results and remarks

Figures 3.3 and 3.4 are two selected plots, which correspond to sequence lengths 500 and 10000. The other plots are not reported, being similar.

High values of the standard deviation make short motifs to have bigger weights. This positively impacts performances when the k-mer length L is overestimated, see the yellow curve in both Figures 3.3, 3.4. This result is important since the length of TBFSs is unknown. Anyway, performances slightly worsen when the k-mer length L is underestimated. Other experiments show that this negative effect is considerable when implanted motifs are more dissimilar.

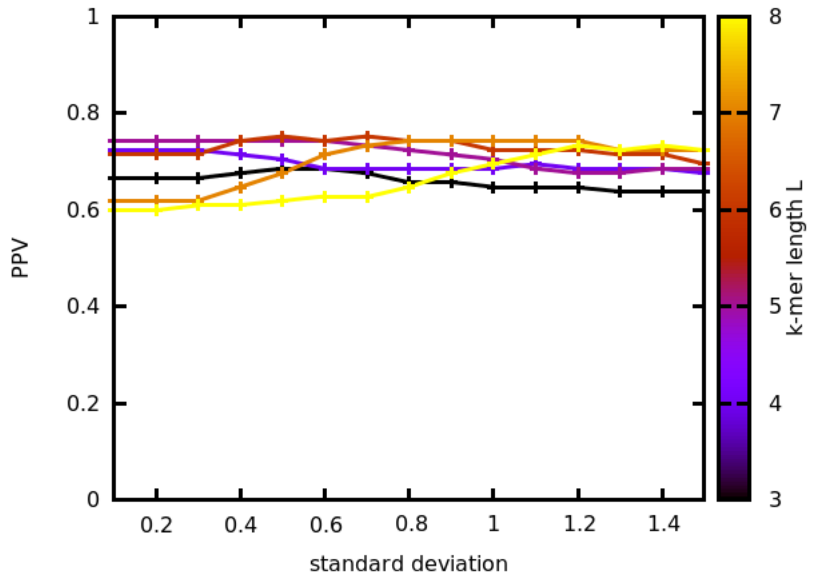

Figure 3.3: experiment 2: sequence length 500, $p_i = 0.01$

3.1.3 Experiment 3: implanting one motif

The objective is the comparison of the measures D_2, D_2^{EP}, N_2 and EP_2 in the simplest case of only one implanted motif.

Experimental conditions

As before, this experiment is performed varying the insertion probability, the entire sequence length and the k-mer length. The standard deviation is fixed because it has been already tuned in the previous experiment. The parameters for the sequence generator are now reported:

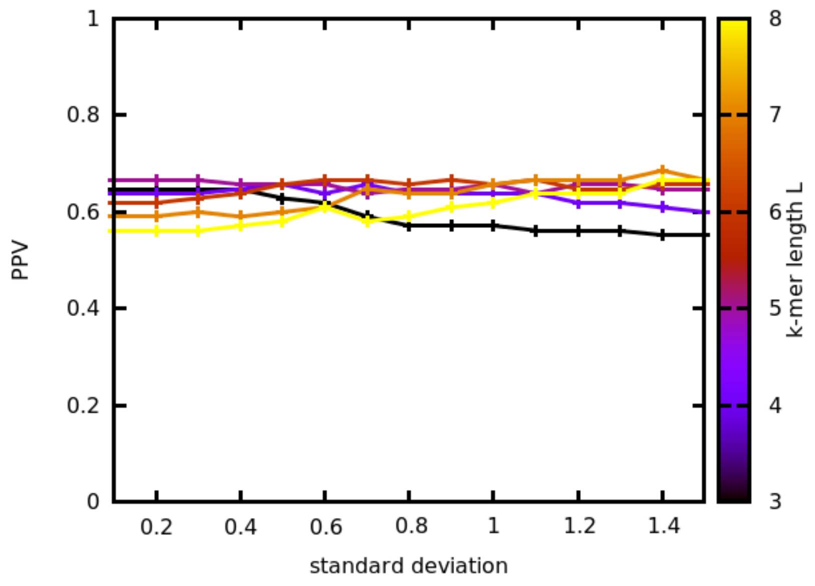

Figure 3.4: experiment 2: sequence length 10000, $p_i = 0.002$

- **Background**: random with identically distributed probabilities, i.e. $p_A = \frac{1}{4}$, $p_C = \frac{1}{4}$, $p_G = \frac{1}{4}$ and $p_T = \frac{1}{4}$;
- **Implanted motifs**: TACCAG;
- **Insertion probabilities** p_i: 0.008, 0.0015;
- **Length of the sequences**: 200 300 400 500 1000 if $p_i = 0.008$, 250 500 1000 2000 3000 4000 5000 10000 if $p_i = 0.0015$;
- **Number of sequences per file**: 25.

The parameters for `ALF` are the following:

- **Background model**: M0;
- **Measure**: D_2, N_2 without reverse complement;
- **k-mer length**: its range is $[3, 8]$;

while those for `ep_sim` are:

- **Background model**: M0;
- **Measure**: D_2^{EP}, EP_2 without reverse complement;
- **Standard deviation**: 0.5;

- **k-mer length**: its range is $[3, 8]$.

the standard deviation is 0.5 so the two biggest weights are 1 and 0.14.

Results and remarks

Figures 3.5 and 3.6 are two selected plots, which correspond to the sequence lengths 500 and 5000. The other plots are analogous a part of the PPV, which is an increasing function of the length of entire sequence.

This experiment confirms that the new measures based on entropic profiles (yellow and purple curves) behave better if the k-mer length is overestimated thanks to the contribute of the k-mer suffixes. Since the selected standard deviation of the Gaussian kernel is relatively small, the Gaussian bell is relatively thin so the weights of the substrings are such that the behaviour of the measures do not change if the k-mer length is underestimated and is enhanced if the k-mer length is overestimated. In fact, the picks correctly correspond to the length of the implanted motif.

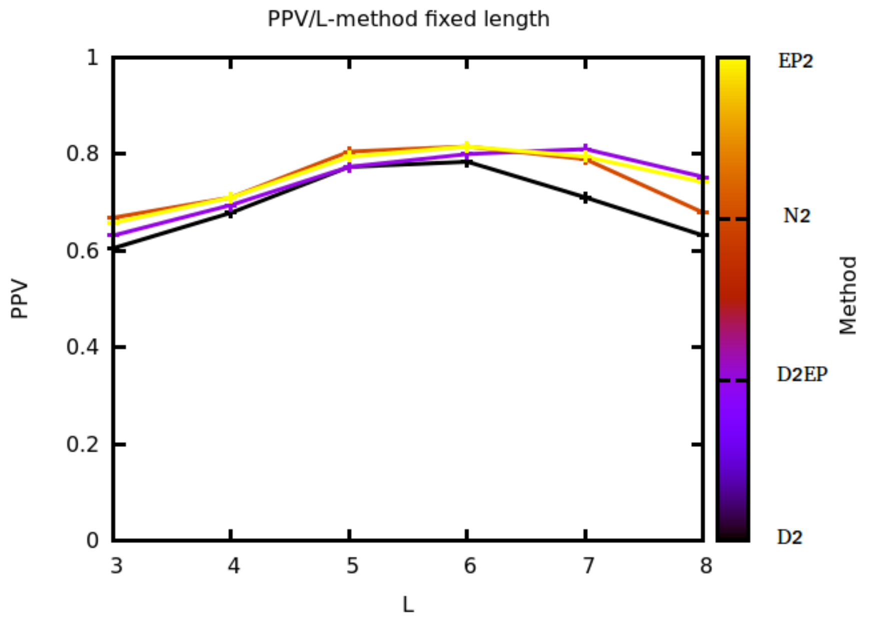

Figure 3.5: experiment 3: sequence length 500, $p_i = 0.008$

3.1.4 Experiment 4: implanting similar motifs

The objective is the comparison of the measures D_2, D_2^{EP}, N_2, EP_2 in the case of many similar implanted motifs, which is more realistic. They are considered similar if they have common substrings, e.g suffixes or prefixes.

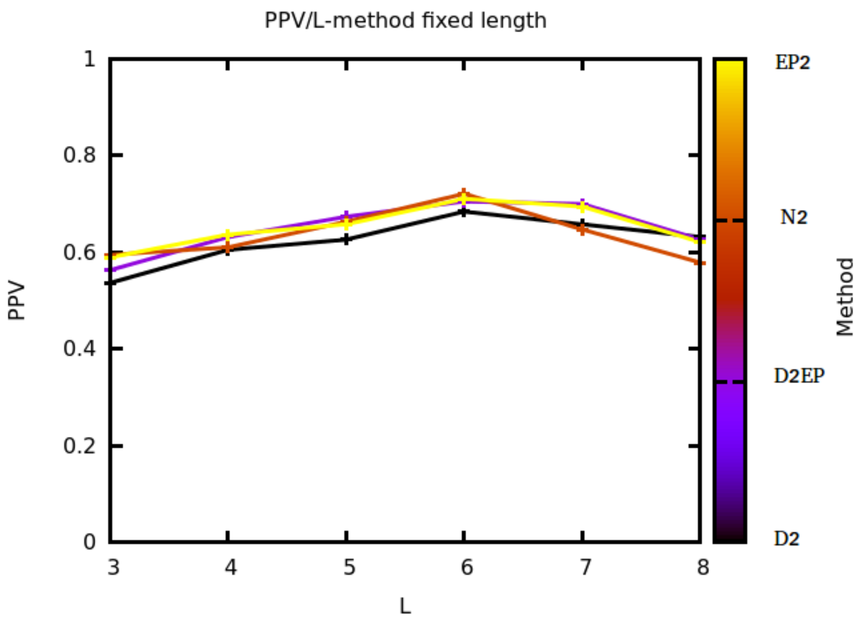

Figure 3.6: experiment 3: sequence length 5000, $p_i = 0.0015$

Experimental conditions

The parameters for the sequence generator are the following:

- **Background**: random with identically distributed probabilities, i.e. $p_A = \frac{1}{4}$, $p_C = \frac{1}{4}$, $p_G = \frac{1}{4}$ and $p_T = \frac{1}{4}$;
- **Implanted motifs**: GCATA, ACCT, TAACGT, GATC, TTGAAC;
- **Insertion probabilities** p_i: 0.01, 0.002;
- **Length of the sequences**: 200 300 400 500 1000 if $p_i = 0.01$, 250 500 1000 2000 3000 4000 5000 10000 if $p_i = 0.002$;
- **Number of sequences per file**: 25;

those for `ALF` are:

- **Background model**: M0;
- **Measure**: D_2, N_2 without reverse complement;
- **k-mer length**: its range is $[3, 8]$;

while those for `ep_sim` are:

- **Background model**: M0;

- **Measure**: D_2^{EP}, EP_2 without reverse complement;
- **Standard deviation**: 0.7;
- **k-mer length**: its range is $[3, 8]$.

The standard deviation is 0.7 so the three biggest weights are 1, 0.36 and 0.02.

Results and remarks

As in the previous experiments, Figures 3.7 and 3.8 are two selected plots, which correspond to sequence lengths 500 and 5000.

This experiment shows that the new measures (red and purple curves) are good in both situations, under or overestimated L, even if the Gaussian bell is wider than the previous experiment. In addition, the performance curves of the proposed methods are flatter than before and with respect to the other methods. This behaviour is due to the fact that the implanted motifs are similar.

It also confirms that, as far as random sequences are concerned, D_2 and D_2^{EP} (black and purple curves) are respectively equivalent to N_2 and EP_2 (yellow and red curves) even though they are not standardized and do not take into account the statistical properties of counts and entropies.

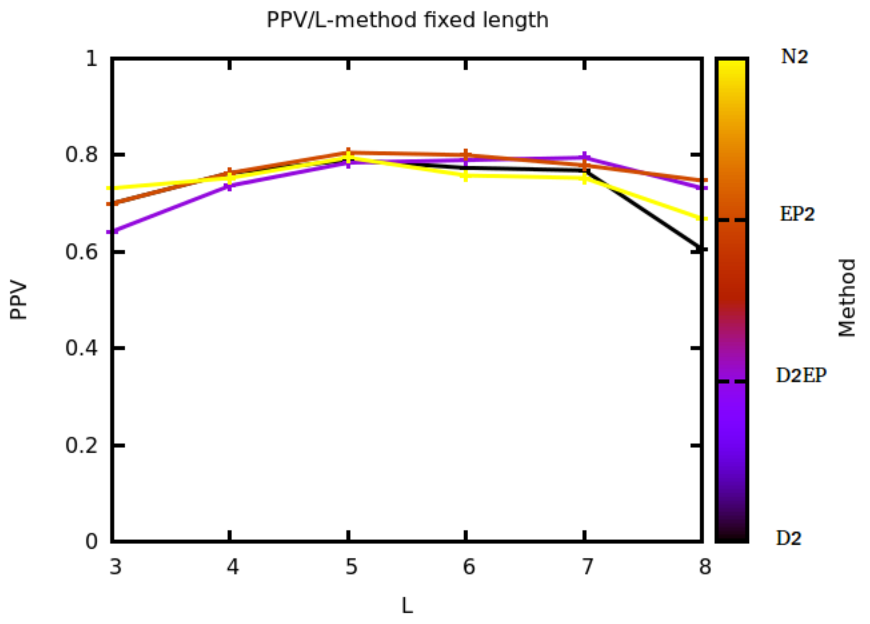

Figure 3.7: experiment 4: sequence length 500, $p_i = 0.01$

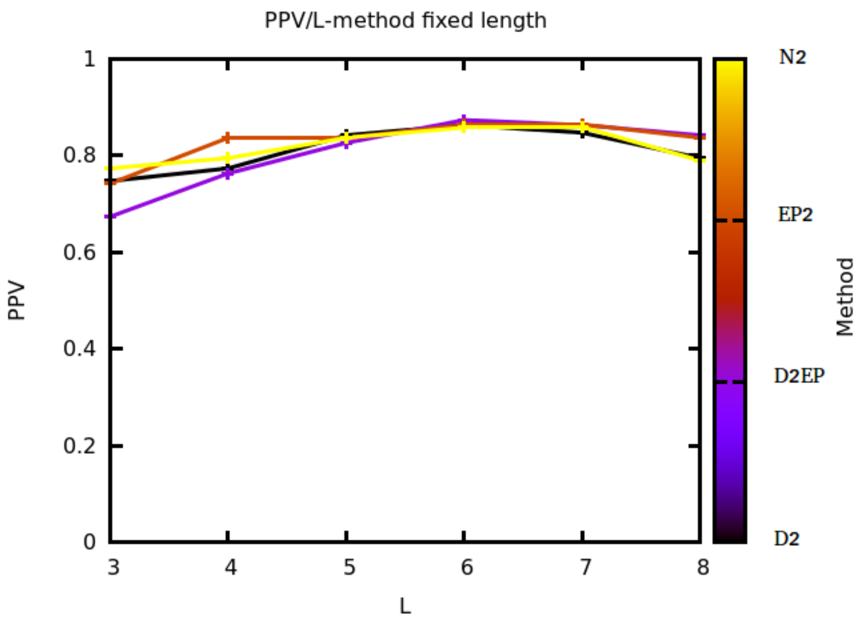

Figure 3.8: experiment 4: sequence length 5000, $p_i = 0.002$

3.1.5 Experiment 5: varying the length of the sequences

The objective is the comparison of the performances of N_2 and EP_2 as a function of the entire sequence length.

Experimental conditions

The parameters for the sequence generator are the following:

- **Background**: random with identically distributed probabilities, i.e. $p_A = \frac{1}{4}$, $p_C = \frac{1}{4}$, $p_G = \frac{1}{4}$ and $p_T = \frac{1}{4}$;
- **Implanted motifs**: ACCTGA, ACCTG, TACCTGA;
- **Insertion probabilities** p_i: 0.015, 0.003;
- **Length of the sequences**: 200 300 400 500 1000 if $p_i = 0.015$, 250 500 1000 2000 3000 4000 5000 10000 if $p_i = 0.003$;
- **Number of sequences per file**: 25;

those for `ALF` are:

- **Background model**: M0;
- **Measure**: N_2 without reverse complement;

- **k-mer length**: its range is $[3, 8]$;

while those for `ep_sim` are:

- **Background model**: M0;
- **Measure**: EP_2 without reverse complement;
- **Standard deviation**: 0.6;
- **k-mer length**: its range is $[3, 8]$.

Results and remarks

The curves in function of the k-mer length have their picks when $L = 6$, which is the selected value for Figure 3.9 and 3.10. This experiment points out two behaviours:

- the performances tends to increase with the length of the sequence, where the number of implanted motifs also increases
- the two methods are almost equivalent for the average length of the implanted motifs, i.e. $L = 6$.

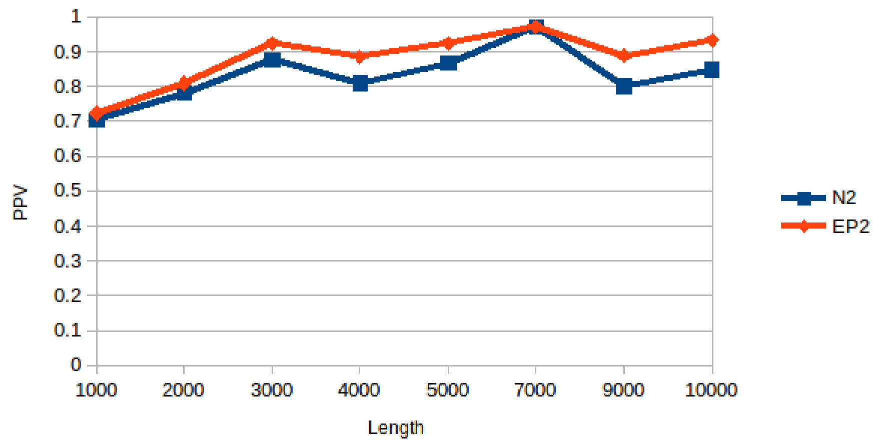

Figure 3.9: experiment 5: performance increment with the length of the sequences, $p_i = 0.008$, $L = 6$

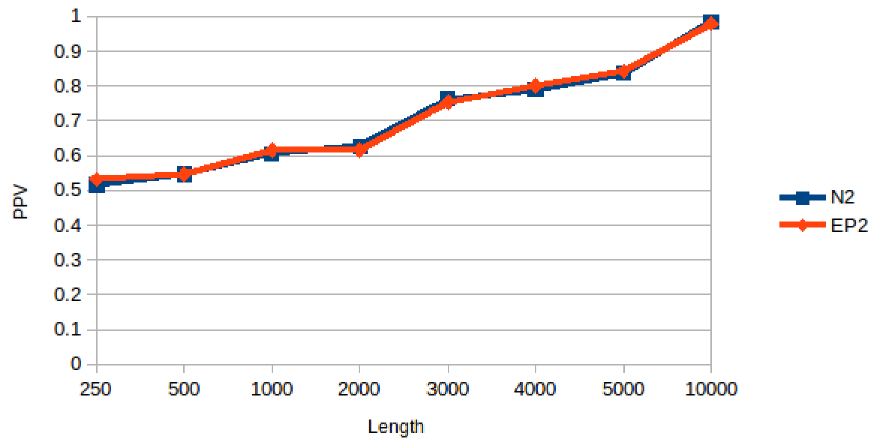

Figure 3.10: experiment 5: performance increment with the length of the sequences, $p_i = 0.0015$, $L = 6$

3.2 Pattern transfer on Drosophila genome

The construction of the negative and positive sets is again performed via bash and Perl scripts.

Even if randomness plays a central role in the theory of genetics, random sequences are an unrealistic background so, this time, the negative set is built by randomly picking sequences from a real genomic sequence. In particular, in the next experiments, the chosen one is a Drosophila intergenic region, `dmel-all-intergenic-r5.49.fasta`, which can be downloaded from FlyBase [33]. An intergenic region is a subset of non-coding DNA located between genes. Despite being called junk DNA, this region contains functionally important elements such as promoters and enhancers. Nevertheless little is known about them.

With regard to the positive set, it is again built artificially by implanting patterns in the sequences of the negative set via the pattern transfer model or the revised one.

3.2.1 Experiment 1: M0 VS M1 background model

The objective is the study of the influence of the background model. The best one, if any, will be chosen in the next experiments of this section.

Experimental conditions

The experiment is performed varying the insertion probability, the entire sequence length and the k-mer length. These are the parameters used for the sequence generator:

- **Background**: real;
- **Implanted motifs**: TGCTAG;

- **Insertion probabilities** p_i: 0.004;
- **Length of the sequences**: 1000 2000 3000 4000 5000 7000 9000 10000;
- **Number of sequences per file**. 15;

The parameters for the scoring application `ALF` are the following:

- **Background**: M0, M1;
- **Measure**: N_2 without reverse complement;
- **k-mer length**: its range is $[3, 8]$;

, those for the scoring application `ep_sim` are:

- **Background model**: M0, M1;
- **Measure**: EP_2 without reverse complement;
- **Standard deviation**: 0.5;
- **k-mer length**: its range is $[3, 8]$;

the standard deviation is 0.5 so the two biggest weights are 1 and 0.14.

Results and remarks

Figures 3.11 and 3.12 are two selected plots which show the gap between M0 (yellow and black curves) and M1 (red and purple curves). Hence, if the background is real, the influence of the model is significant and M1 is clearly the best.

Another remark is about the performances of the two methods. If only one motif is implanted and the background model is fixed, the two methods are almost equivalent.

3.2.2 Experiment 2: implanting similar motifs

The objective is the comparison of the measures D_2, D_2^{EP}, N_2, EP_2 in the case of many similar implanted motifs. They are considered similar if they have common substrings, e.g. suffixes and prefixes.

Experimental conditions

The parameters for the sequence generator are the following:

- **Background**: real;
- **Implanted motifs**: AGCCA, GCCA,TAGCCA,TCCA,AGCCAG;

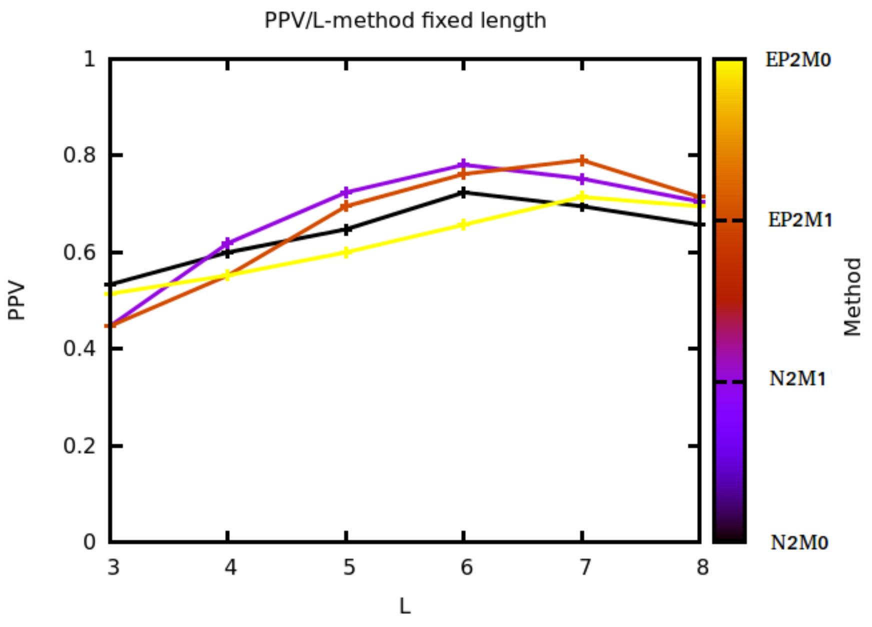

Figure 3.11: experiment 1: m1 (red and purple) outperforms m0 (yellow and black), $p_i = 0.004$, length 2000

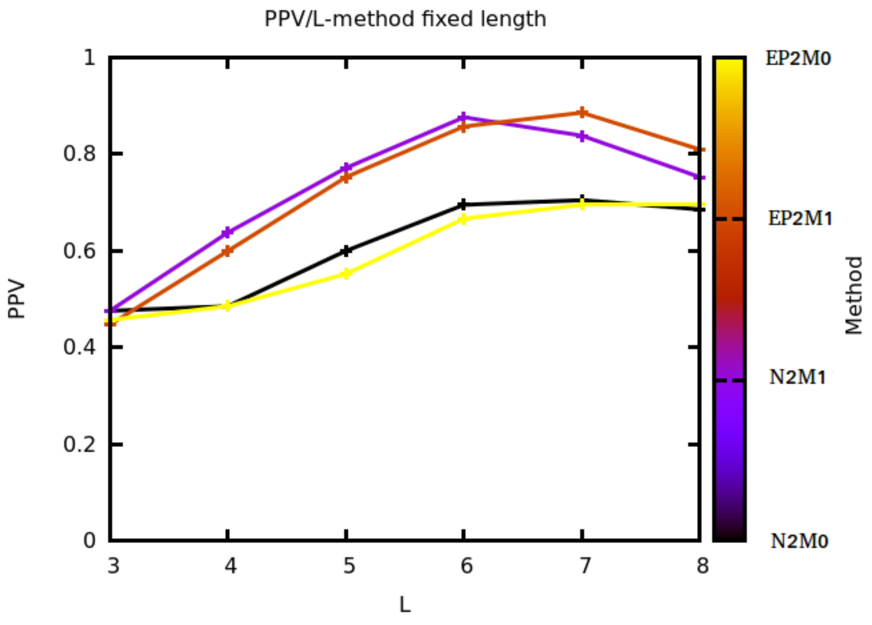

Figure 3.12: experiment 1: m1 (red and purple) outperforms m0 (yellow and black), $p_i = 0.004$, length 4000

- **Insertion probabilities** p_i: 0.008;
- **Length of the sequences**: 1000 2000 3000 4000 5000 7000 9000 10000;
- **Number of sequences per file**: 15, 10 if length is 9000;

those for `ALF` are:

- **Background model**: M1;
- **Measure**: D_2, N_2 without reverse complement;
- **k-mer length**: its range is $[3, 8]$;

while those for `ep_sim` are:

- **Background model**: M1;
- **Measure**: D_2, EP_2 without reverse complement;
- **Standard deviation**: 0.6;
- **k-mer length**: its range is $[3, 8]$;

The standard deviation is 0.6 so the two biggest weights are 1 and 0.25.

Results and remarks

As in the previous experiments, Figures 3.13 and 3.14 are two selected plots, which correspond to sequence lengths 4000 and 10000. It is worthwhile noting that, as expected, both variants of D_2 (in black and purple), which do not take into account the statistical properties of counts and entropies, have no statistical power. Another remark concerns the comparison of the other two methods: EP_2 (in yellow) outperforms N_2 (in red) if the inserted motifs are similar and contain common suffixes.

3.2.3 Experiment 3: varying the length of the sequences

The objective is the comparison of the performances of N_2 and EP_2 as a function of the entire sequence length.

Experimental conditions

The parameters for the sequence generator are the following:

- **Background**: real;
- **Implanted motifs**: AGCCA, GCCA,TAGCCA,TCCA,AGCCAG;

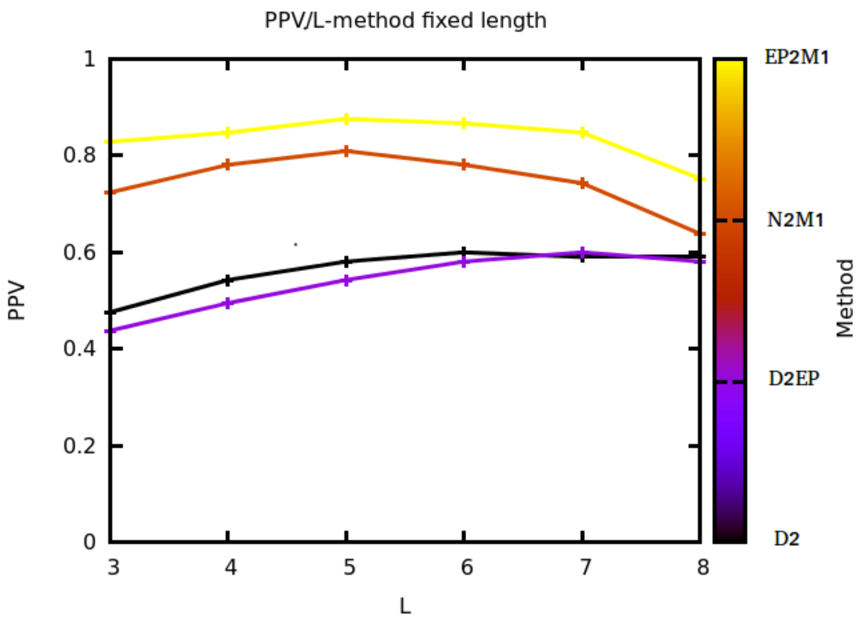

Figure 3.13: experiment 2: length 4000

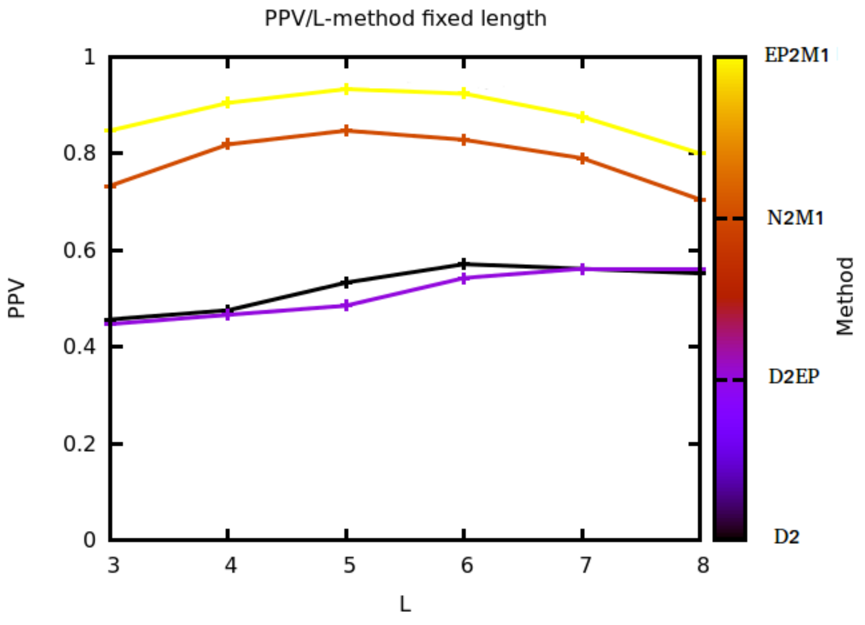

Figure 3.14: experiment 2: length 10000

- **Insertion probabilities** p_i: 0.008;
- **Length of the sequences**: 1000 2000 3000 4000 5000 7000 9000 10000;

- **Number of sequences per file**: 15, 10 if length is 9000;

those for `ALF` are:

- **Background model**: M1;
- **Measure**: N_2 without reverse complement;
- **k-mer length**: its range is $[3, 8]$;

while those for `ep_sim` are:

- **Background model**: M1;
- **Measure**: EP_2 without reverse complement;
- **Standard deviation**: 0.6;
- **k-mer length**: its range is $[3, 8]$;

Results and remarks

The curves in function of the k-mer length have their picks when $L = 5$, which is the selected value for Figure 3.16. The statistics are no longer monotonic with a real background. Indeed, the sequences are taken from different parts of the genome, which might have different statistical properties. The diagram for $L = 7$ is also reported.

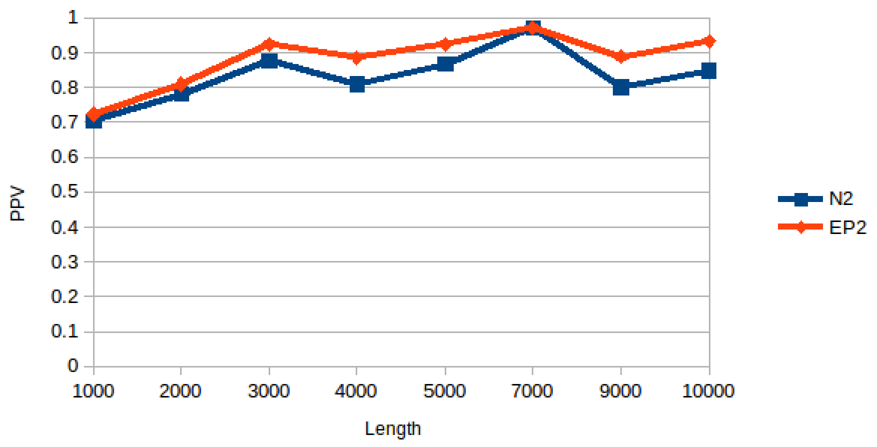

Figure 3.15: experiment 3: no performance increment with the length of the sequences, $p_i = 0.008$, $L = 5$

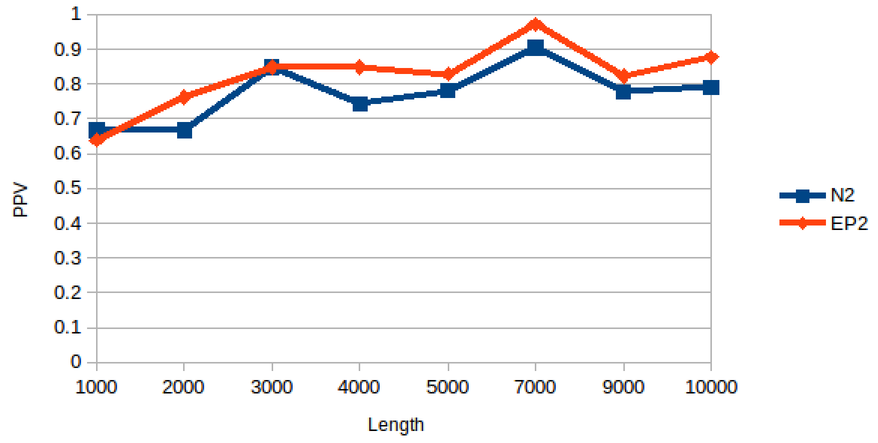

Figure 3.16: experiment 3: no performance increment with the length of the sequences, $p_i = 0.008$, $L = 7$

3.3 Comparison of mouse regulatory sequences

This series of experiments is the most challenging because involves neither artificial enhancers nor implanted transcription factor binding sites. The positive set is build from real enhancers and consists in sequences of various length. More precisely, it contains sequences of length 350, 500, 700 and 1000 taken from tissue-specific enhancers of mouse embryos active in one of the following tissues: forebrain, midbrain, limb or heart. They have been discovered as reported in [35] [34] and are used in many works. On the contrary, the negative set depends on the experiment and will be described time by time.

3.3.1 Experiment 1: ChIP-seq data of mouse tissue-specific enhancers VS random mouse genomic sequences

The negative set contains sequences taken at random from the mouse genome, whose FASTA file `Mus_musculus.GRCm38.75.dna.toplevel.fa` can be downloaded from [36]. To this end, the program `seq_splitter` has been implemented. It requires three arguments: the input FASTA file, whose records are the real sequences, the number of output sequences, which are created by splitting the sequences in the input file, and their length. Finally, the sequences for the negative set can be randomly picked from the output file.

Experimental conditions

The performances of EP_2 and N_2 are evaluated for each of the four tissues, from which sequences of length 500, 700 and 1000 are randomly picked. The number of sequences per file is 20 and the results are averaged over 10 runs. Given that no artificial motif is implanted, the best motif length is unknown and function of the tissue. For this reason, the chosen standard deviation is 0.7. The aim is to take advantage of the multi-resolution property of entropic profiles. Another crucial choice is again that of the background model, whose importance will be pointed out again.

Results and remarks

As a result of the limited size of enhancer sequences, Bernoulli model is better than higher order Markov models as reported in the Tables 3.1, 3.2, 3.3 and 3.4. Markov model M1 already leads to over-fitting, i.e. it exaggerates minor fluctuations in the data and causes poor predictive performance. The tables shows also that EP_2 is better than N_2.

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.76	0.75
Forebrain	0.74	0.71
Midbrain	0.69	0.69
Heart	0.70	0.69

Table 3.1: Average PPV if background model M0, $L = 4$, $\sigma = 0.7$

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.72	0.68
Forebrain	0.66	0.62
Midbrain	0.67	0.64
Heart	0.67	0.62

Table 3.2: Average PPV if background model M0, $L = 7$, $\sigma = 0.7$

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.58	0.57
Forebrain	0.58	0.57
Midbrain	0.59	0.59
Heart	0.55	0.55

Table 3.3: Average PPV if background model M1, $L = 4$, $\sigma = 0.7$

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.57	0.56
Forebrain	0.56	0.54
Midbrain	0.57	0.54
Heart	0.54	0.51

Table 3.4: Average PPV if background model M1, $L = 7$, $\sigma = 0.7$

3.3.2 Experiment 2: ChIP-seq data of mouse tissue-specific enhancers VS other tissue-specific enhancers

The previous test shows that tissue-specific enhancers have similar word content. However, the comparison with random genomic sequences can be biased by the technology [16] [13], e.g when it more likely extracts sequences with high or similar GC-content. To avoid this bias, in this experiment different ChIP-seq sequences are compared with each other. In other words, the positive set contains the enhancers active in one of the tissues while the negative set contains the enhancers active in all the other. This test is more challenging and performances will be inevitably low because also the negative set contains similar sequences.

Experimental conditions

The results are averaged over 10 runs and the number of sequences per file is 35. The length of the enhancers is 350 and because of the results of the previous experiment the background model is M0. As in the previous experiment, the standard deviation of the Gaussian kernel is 0.7.

Results and remarks

The PPV in Tables 3.5 and 3.6 are low as it was expected. EP_2 performs slightly better than N_2.

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.64	0.63
Forebrain	0.60	0.55
Midbrain	0.51	0.49
Heart	0.59	0.59

Table 3.5: Average PPV if background model M0, $L = 4$, $\sigma = 0.7$

Tissue	$\boldsymbol{EP_2}$	$\boldsymbol{N_2}$
Limb	0.55	0.53
Forebrain	0.56	0.53
Midbrain	0.48	0.49
Heart	0.53	0.53

Table 3.6: Average PPV if background model M0, $L = 7$, $\sigma = 0.7$

Chapter 4

Conclusions

4.1 Results

The objective of this work was to explore to which extent the multi-resolution property of the local definition of entropic profiles can enhance alignment-free global similarity measures based on k-mer counts.

The starting points were the local entropic profilers implemented in C and Java and their formulations of entropic profiles respectively based on standardized suffix counts and normalized prefix counts. In a first step, many methods have been explored so as to understand which one was the most suitable and which were the most important variables influencing performances.

The first experiments revealed two aspects. On the one hand, the most promising methods were D_2^* and N_2 but, according to the original exponential kernel, shorter suffixes were weighted too lightly to make a difference. On the other, the original arithmetic standardization and normalization do not suffice to enhance the sequence similarity scores because the arithmetic mean, standard deviation and maximum value are not function of the single k-mer. Thus, on the one hand, the formulation of entropic profile was simplified and its kernel generalized. In practice, a Gaussian kernel was proposed instead of an exponential. On the other, the statistical properties of entropic profile, i.e. entropy expectation, variance and covariance, were studied for the first time starting from the similar derivations of the statistical properties of counts.

The two aspects were merged in a D_2^* and, above all, N_2 schema paying particular attention to the role of the statistical model. The proposed application `ep_sim` is based on the C++ sequence analysis library SeqAn and its implementation is independent from the original ones, which are indeed designed to address different case studies.

Finally, the experimental setups are described and the results compared with the state of the art paying particular attention to the role of the statistical model. Experiments on both simulated and real enhancers reveal that the multi-resolution property of the new methods make the similarity scores more robust with the change of the k-mer length but further improvements are needed to effectively address the most challenging situations.

4.2 Future works

There are many aspects that can be expanded in the future. For example, since entropic profiles proved to enhance the similarity scores, it would be interesting to extend their definition so as to take into account not only suffixes or prefixes but also all the substrings or all the longest down to a certain lower length. With that goal, some useful methods have been already implemented in the developed application.

Furthermore, these measures can also be applied to compare sequences of different length. Rather than develop other similarity measure such in [37], this could be done by grading the words in a chart based on their entropies. If the top words are those with highest entropy, only the top of the chart could be considered to describe the sequence. In this direction, it would be interesting to study how much similar are the words in the top chart and the effect of side effects, i.e. the relationship between the position in the chart of an exceptional word and the positions of words containing its substrings, so as to cluster substrings and build a sort of sequence descriptor based on the entropies of related words belonging to the same cluster.

Other aspects that could be focused on are the application of the underlying idea of this work also to other alignment-free methods and, going back to the local formulation, the application of the developed statistical properties of the entropic profiles to the starting problem, i.e. motif finding. Finally, with the advent of next-generation sequencing (NGS) technologies, one of the most salient problem in bioinformatics is the analysis of the large amount of short reads that they produce [8]. For this purpose, sequence similarity methods based on the frequencies of word patterns, such as those developed in this work, can be potentially useful.

Chapter 5

Bibliography

[1] *http://ghr.nlm.nih.gov/*.

[2] Neil C. Jones and Pavel A. Pevzner, *An introduction to bioinformatics algorithms.* A Bradford Book The MIT Press Cambridge, Massachusetts London, England, 2004.

[3] *http://www.nature.com/nrg/series/regulatoryelements/index.html*.

[4] Daria Shlyueva, Gerald Stampfel, Alexander Stark, *Transcriptional enhancers: from properties to genome-wide predictions.* Nature Reviews Genetics 15, 272–286, 2014.

[5] Smith TF, Waterman MS. *Comparison of biosequences.* Adv Appl Math 1981;2:482-9.

[6] Altschul S.F., Gish W., Miller W., Myers E.W. and Lipman D.J., *Basic local alignment search tool.* J. Mol. Biol. 215: 403-410, 1990.

[7] Susana Vinga, and Jonas Almeida. *Alignment-free sequence comparison—a review.* Bioinformatics (2003) 19 (4): 513-523.

[8] Kai Song, Jie Ren Gesine Reinert, Minghua Deng, Michael S. Waterman, Fengzhu Sun. *New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing.* Brief Bioinform (2014) 15 (3): 343-353.

[9] Blaisdell, B.E. *A measure of the similarity of sets of sequences not requiring sequence alignment.* Proc. Natl Acad. Sci. USA, 83, 5155–5159, 1986.

[10] Gesine Reinert, David Chew, Fengzhu Sun, Michael S Waterman, *Alignment-free sequence comparison (I): statistics and power.* Journal of Computational Biology. December 2009, 16(12): 1615-1634.

[11] Miriam R. Kantorovitz,Gene E. Robinson, Saurabh Sinha, *A statistical method for alignment-free comparison of regulatory sequences.* In Bioinformatics. 2007 July 1.

[12] Alexander Solovyov and W Ian Lipkin, *Centroid based clustering of high throughput sequencing reads based on n-mer counts.* BMC Bioinformatics 2013, 14:268.

[13] Jonathan Göke, Marcel H. Schulz, Julia Lasserre, and Martin Vingron. *Estimation of Pairwise Sequence Similarity of Mammalian Enhancers with Word Neighbourhood Counts.* Bioinformatics, 2012.

[14] Matteo Comin, and Davide Verzotto. *Alignment-Free Phylogeny of Whole Genomes using Underlying Subwords.* BMC Algorithms for Molecular Biology 2012, 7:34.

[15] Matteo Comin, and Davide Verzotto. *Whole-Genome Phylogeny by Virtue of Unic Subwords.* Proceedings of 23rd International Workshop on Database and Expert Systems Applications, BIOKDD 2012.

[16] Matteo Comin, and Davide Verzotto. *Beyond fixed-resolution alignment-free measures for mammalian enhancers sequence comparison.* Computational Biology and Bioinformatics, IEEE/ACM Transactions on (Volume:PP , Issue: 99), 2014.

[17] S. Robin et al, *DNA, Words and Models: Statistics of Exceptional Words.* Cambridge University Press, 2005.

[18] Oliver JL1, Bernaola-Galván P, Guerrero-García J, Román-Roldán R., *Entropic profiles of DNA sequences through chaos-game-derived images.* J. Theor. Biol., 160:457-470, 1993.

[19] Susana Vinga, Jonas S. Almeida, *Local Renyi entropic profiles of DNA sequences.* BMC Bioinformatics 2007, 8:393 .

[20] F Fernandes, AT Freitas, JS Almeida, S Vinga, *Entropic Profiler–detection of conservation in genomes using information theory.* BMC research notes 2 (1), 72.

[21] Matteo Comin, Morris Antonello , *Fast Computation of Entropic Profiles for the Detection of Conservation in Genomes.* Proceedings of Pattern Recognition in Bioinformatics 2013, Lecture Notes in BIoinformatics (LNBI) 2013, 7986, pp. 277-288.

[22] Matteo Comin, Morris Antonello , *Fast Entropic Profiler: An Information Theoretic Approach for the Discovery of Patterns in Genomes.* IEEE/ACM Transactions on Computational Biology and Bioinformatics 2014.

[23] Michael F. Barnsley, *Fractals Everywhere.* 2nd Edition MK, 1993.

[24] Jeffrey HJ , *Chaos game representation of gene structure.* Nucleic Acids Res. 18:2163–2170, 1990.

[25] Sourice S1, Biaudet V, El Karoui M, Ehrlich SD, Gruss A , *Identification of the Chi site of Haemophilus influenzae as several sequences related to the Escherichia coli Chi site.* Mol Microbiol. 1998 Mar;27(5):1021-9.

[26] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly, 2008.

[27] Sheldon M. Ross,, *A First Course in Probability.* Pearson Prentice Hall; 7th Edition, 2006.

[28] Andreas Döring, David Weese, Tobias Rausch and Knut Reinert, *SeqAn an efficient, generic C++ library for sequence analysis.* BMC Bioinformatics, 9:11, 2008.

[29] *https://www.seqan.de/*.

[30] Kantorovitz MR1, Robinson GE, Sinha S, *A statistical method for alignment-free comparison of regulatory sequences.* Bioinformatics. 2007 Jul 1;23(13):i249-55.

[31] X. Liu, L.Wan, G. Reinert, M.S. Waterman, F. Sun, J. Li, *New powerful statistics for alignment-free sequence comparison under a pattern transfer model.* (2011) Journal of Theoretical Biology, 284, 106-116.

[32] *http://www.gnuplot.info/*.

[33] *http://flybase.org/*.

[34] Blow MJ1, McCulley DJ, Li Z, Zhang T, Akiyama JA, Holt A, Plajzer-Frick I, Shoukry M, Wright C, Chen F, Afzal V, Bristow J, Ren B, Black BL, Rubin EM, Visel A, Pennacchio LA., *ChIP-Seq identification of weakly conserved heart enhancers.* Nat Genet. 2010 Sep;42(9):806-10.

[35] Visel A1, Blow MJ, Li Z, Zhang T, Akiyama JA, Holt A, Plajzer-Frick I, Shoukry M, Wright C, Chen F, Afzal V, Ren B, Rubin EM, Pennacchio LA., *ChIP-seq accurately predicts tissue-specific activity of enhancers.* Nature. 2009 Feb 12;457(7231):854-8.

[36] *http://www.ensembl.org/*.

[37] Jie Ren1, Kai Song1, Fengzhu Sun, Minghua Deng1 and Gesine Reinert, *Multiple alignment-free sequence comparison.* Bioinformatics (2013) 29 (21): 2690-2698.