COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

DETECTION OF TANDEM REPEATS IN NANOPORE DATA

MASTER THESIS

2020
BC. EDUARD BATMENDIJN

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

DETECTION OF TANDEM REPEATS IN NANOPORE DATA

MASTER THESIS

Study Programme:	Computer Science
Field of Study:	Computer Science
Department:	Department of Applied Informatics
Supervisor:	doc. Mgr. Tomáš Vinař, PhD.

Bratislava, 2020
Bc. Eduard Batmendijn

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Eduard Batmendijn
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: anglický
Sekundárny jazyk: slovenský

Názov: Detection of Tandem Repeats in Nanopore Data
Detekcia tandemových opakovaní v nanopórových dátach

Anotácia: Sekvenátory Oxford Nanopore sekvenujú DNA zaznamenávaním elektrického signálu, ktorý je neskôr prekladaný do reťazcov nad abecedou {A,C,G,T} reprezentujúcich molekuly DNA. Cieľom tejto práce je detekcia tandemových opakovaní s pomocou surového elektrického signálu. Tento problém by sa obvykle riešil priamo na reťazcoch, no v prípade nanopórového sekvenovania výsledné reťazce obsahujú veľké množstvo chýb a použitie surového signálu by malo viesť ku zlepšeniu presnosti.

Vedúci: doc. Mgr. Tomáš Vinař, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: prof. Ing. Igor Farkaš, Dr.

Dátum zadania: 19.02.2020

Dátum schválenia: 19.02.2020 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Comenius University in Bratislava
Faculty of Mathematics, Physics and Informatics

THESIS ASSIGNMENT

Name and Surname:	Bc. Eduard Batmendijn
Study programme:	Computer Science (Single degree study, master II. deg., full time form)
Field of Study:	Computer Science
Type of Thesis:	Diploma Thesis
Language of Thesis:	English
Secondary language:	Slovak

Title: Detection of Tandem Repeats in Nanopore Data

Annotation: Oxford Nanopore DNA sequencing machines record electrical signals that are later base called to strings over alphabet {A,C,G,T} representing DNA molecules. The goal of this thesis is to explore detection of tandem repeats with the use of the raw signals. This task would be ordinarily done on strings, however, due to high error rate of base calling in nanopore sequencing, the use of the raw signal should improve the accuracy significantly.

Supervisor:	doc. Mgr. Tomáš Vinař, PhD.
Department:	FMFI.KAI - Department of Applied Informatics
Head of department:	prof. Ing. Igor Farkaš, Dr.
Assigned:	19.02.2020

Approved: 19.02.2020 prof. RNDr. Rastislav Kráľovič, PhD.
Guarantor of Study Programme

.....
Student

.....
Supervisor

Abstrakt

V niektorých aplikáciách sekvenovania DNA je žiaduce detegovať repetitívne sekvencie v jednotlivých čítaniach. Štandardné metódy detekcie tandemových opakovaní nie sú vhodné pre čítania z nanopórového sekvenovania, nakoľko tieto čítania obsahujú veľké množstvo chýb. V tejto práci navrhujeme dve metódy detekcie tandemových opakovaní, ktoré pracujú priamo so signálom zo sekvenátora MinION, čím sa vyhýbajú nepresnému prekladu signálu na postupnosť báz. Hoci naše metódy presvedčivo neprekonali jednoduchú metódu založenú na preklade signálu na postupnosť báz a následnom hľadaní opakovaní v tejto postupnosti, zdá sa, že jedna z našich metód dobre funguje na dátach obsahujúcich opakovania s relatívne dlhými periódami.

Kľúčové slová: nanopórové sekvenovanie, tandemové opakovania, MinION

Abstract

In some applications of DNA sequencing it is desirable to detect repetitive sequences in individual reads. Standard methods for tandem repeat detection are not suitable for nanopore reads, due to the high amount of errors contained in the reads. We propose two methods for detection of tandem repeats directly in the signal from the MinION sequencer, thus avoiding the need for the inaccurate basecalling step. While our methods do not convincingly outperform the baseline approach of detecting repeats on the basecalled sequence, one of them seems to perform well on sequences that contain repeats of relatively long periods.

Keywords: nanopore sequencing, tandem repeats, MinION

Contents

Chapter 1

Introduction

Deoxyribonucleic acid (DNA) contains genetic information of organisms. A DNA molecule has the shape of a double helix consisting of two strands. Each strand consists of a sequence of nucleotides, each nucleotide containing one of four nucleobases adenine (A), cytosine (C), guanine (G) of thymine (T). The sequence of nucleobases encodes the information stored in the molecule. Nucleotide sequences of the two strands are complementary to each other: where one strand has C, the other one has G and where one has A, the other has T. Thus, the nucleotide sequence of a single strand contains the full information.

1.1 Nanopore sequencing

DNA sequencing is the process of determining the nucleotide sequence (i.e. a string over the alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$) of a physical DNA sample. Various techniques for DNA sequencing have been developed, the oldest ones reaching back to 1970s [16]. Most sequencing techniques can only process DNA fragments of limited length. These processed DNA fragments are called *reads*.

To sequence a longer portion of DNA, multiple copies of it are broken into smaller, overlapping fragments. These fragments are then sequenced. The resulting reads are then aligned and merged to reconstruct the original nucleotide sequence. This reconstruction process is called *sequence assembly*.

Nanopore sequencing is a relatively new DNA sequencing approach, that can produce very long reads. A pore-forming protein is used to puncture a small hole (the *nanopore*) in an electrically resistant membrane. Voltage is apllied accross the membrane, creating an ionic current through the nanopore. On one side of the membrane, strands of the sequenced DNA are unzipped by a motor proteine. One of the strands enters the nanopore and passes through it (Figure 1.1). As the DNA strand passes through the nanopore, it partially blocks the electric current. During the whole process, the electric current passing through the nanopore (also referred to as *signal*) is measured and recorded.

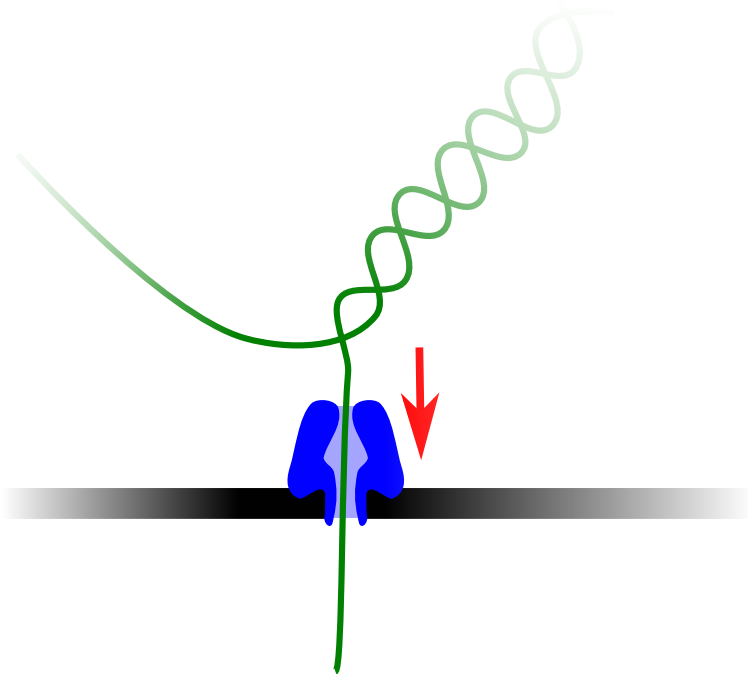

Figure 1.1: A DNA strand passing through the nanopore.

The signal is mostly affected by nucleotides that are currently in the nanopore, or close to it. Nucleotides of different types affect the signal in different ways. Thus, the signal can be used to infer the nucleotide sequence that has passed through the nanopore [4].

In this work, we use data produced by MinION, a commercially available nanopore sequencing device by Oxford Nanopore Technologies. In the version of MinION we are using, the signal is sampled $4,000$ times per second. The speed of the DNA strand passing through the nanopore fluctuates during the process of sequencing. Typically, about 400 nucleotides pass through the nanopore every second, resulting in about 10 signal samples per nucleotide.

The process of translating the measured signal into a sequence of nucleobases is called *basecalling*. Accurate basecalling of the MinION signal has proven to be a challenging problem. There are several reasons for this. Firstly, the electric current is not only affected by a single nucleotide passing through the nanopore, but also by several nucleotides around it. Secondly, the signal is systematically affected by factors other than the DNA sequence passing through the nanopore. Finally, the DNA strand does not move through the nanopore at a constant speed. Therefore, some nucleotides correspond to longer portions of the signal than others.

Several approaches have been tried, with varying results. The most accurate basecallers employ neural networks and their accuracy is about $85\% - 90\%$ for individual reads [21]. A much higher accuracy can be achieved by reading the same portion of DNA several times and combining these reads into a single consensus sequence.

1.2 Tandem repeats

Tandem repeats are a phenomenon that occurs in DNA, where a sequence of nucleotides (a *pattern*) is repeated several times, the repetitions being immediately adjacent to each other. Sometimes, individual copies of the pattern are only approximate. In those cases, we speak of *imperfect* tandem repeats.

Tandem repeats are quite common in DNA and they are interesting from a biological point of view. The expansion of trinucleotide repeats (i.e. repeats whose pattern is three nucleotides long) can cause neurological disorders [14]. Tandem repeats with short patterns (*microsatelites*) are used as markers in applications such as paternity testing, population genetics and cancer diagnosis [11][10][20].

Long repetitive regions of DNA also cause difficulties in sequence assembly [19]. Problems arise when the DNA sequence contains a tandem repeat that is not fully spanned by any read. When aligning two reads whose overlapping part is inside the repeat, there are multiple possible alignments, as shifting a good alignment by one copy of the repeat's pattern yields another good alignment. Because of this, it is virtually impossible to accurately determine how many copies of its pattern the repeat contains.

1.2.1 Notational conventions

We will shortly introduce some terms related to tandem repeats, that will be used throughout this thesis. Before we do that, we need to estabilish some terms and conventions for sequences in general.

Convention 1. Sequences are always denoted by uppercase latin letters, e.g. S, T. Length of a sequence is denoted by its name enclosed in $||$, e.g. length of sequence S is $|S|$.

Individual elements of a sequence are denoted by lowercase version of the letter denoting the whole sequence, with a one-based subscript. For example, $S = s_1 s_2 \dots s_{|S|}$.

Throughout this thesis, we will often need to refer to parts of sequences. In tandem repeats, several occurences of a shorter nucleotide sequence are contained in a longer sequence as substrings. To make clear whether we are refering to the shorter sequence as such, or to its individual occurences, let us formally define the notion of sequence *part*.

Definition 1. A sequence *part* is a 3-tuple (S, l, r), where S is a sequence, $l, r \in \{1, 2, \dots, |S|\}$ and $l \leq r$. We say that (S, l, r) is a part of sequence S. We use notation $S_{[l,r]}$ to denote (S, l, r).

Two parts $S_{[l_1,r_1]}$, $S_{[l_2,r_2]}$ of the same sequence S are *identical* (denote $S_{[l_1,r_1]} = S_{[l_2,r_2]}$) iff $l_1 = l_2$ and $r_1 = r_2$.

Two parts $S_{[l_1,r_1]}$, $S_{[l_2,r_2]}$ of the same sequence S look *alike* (denote $S_{[l_1,r_1]} \approx S_{[l_2,r_2]}$) iff $r_1 - l_1 = r_2 - l_2$ and $s_{l_1+i} = s_{l_2+i}$ for each $i \in \{0, 1, \dots, r_1 - l_1\}$.

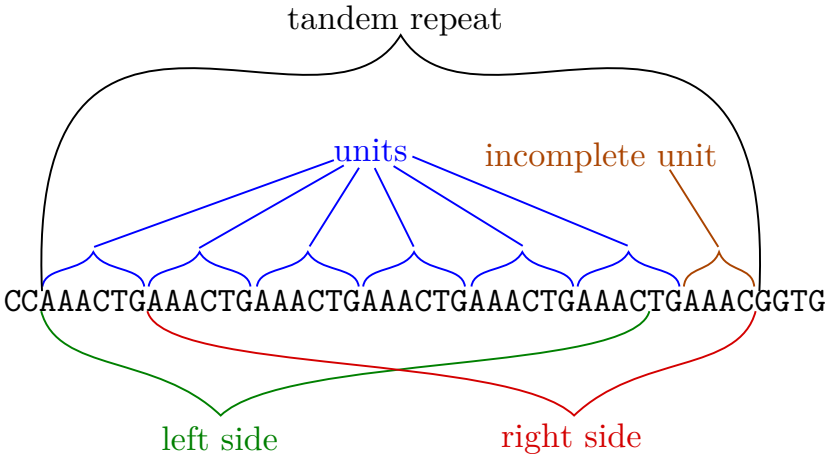

Figure 1.2: Parts of a tandem repeat with period 6, unit count 20/3 and pattern `AAACTG`.

Intuitively, $S_{[l,r]}$ is the part of S from its l-th nucleotide to its r-th nucleotide. For convenience, we will use similar notation for sequence parts as we use for sequences.

Convention 2. Sequence parts are also denoted by uppercase latin letters.

Let $P = S_{[l,r]}$, where S is a nucleotide sequence. Then:

- $|P| \stackrel{\text{def}}{=} r - l + 1$.
- $p_i \stackrel{\text{def}}{=} s_{l+i-1}$ for each $i \in \{1, \ldots, |P|\}$.
- $P_{[l',r']} \stackrel{\text{def}}{=} S_{[l+l'-1,l+r'-1]}$ for each $l', r' \in \{1, \ldots, |P|\}; l' \leq r'$.

1.2.2 Parts of tandem repeat

Definition 2. Let S be a sequence of nucleotides and let $T = S_{[l,r]}$ be its part corresponding to a (perfect) tandem repeat.

- *Period* of the tandem repeat T is the smallest positive integer k such that $T_i = T_{i+k}$ for each $i \in \{1, 2, \ldots, |T| - k\}$.
- *Pattern* of T is the sequence of whose repetitions T consists, i.e. $t_1 t_2 \ldots t_k$.
- *Units* of T are all occurences of T's pattern in T, i.e. $T_{[1,k]}, T_{[k+1,2k]}, T_{[2k+1,3k]}, \ldots$.
- If $|T|/k$ is not an integer, the *incomplete unit* of T is the prefix of T's pattern that occurs at the end of T (i.e. $T_{[x+1,|T|]}$, where $x = \left\lfloor \frac{|T|}{k} \right\rfloor \cdot k$).
- *Unit count* of T is the quantity $|T|/k$.
- The *left side* of T is $T_{[1,|T|-k]}$.
- The *right side* of T is $T_{[k+1,|T|]}$.

Note that unit count of a tandem repeat is always at least two. A unit count from the open interval $(1, 2)$ would mean that a substring of S is repeated, but there are some other nucleotides interposed between the repetitions.

Also note that left side and right side are defined in such a way, that they look alike.

The extension of terms defined in Definition 2 for imperfect tandem repeats should be intuitive.

1.3 Goals

In some applications, it is desirable to detect tandem repeats in a single read. There are several software tools for finding tandem repeats in nucleotide sequences [7] [2] [9]. However, they are not suitable for finding tandem repeats in MinION reads, due to the inaccuracy of basecallers. With approximately 10% of errors, many tandem repeats change to the point when repeat-finding tools cannot detect them.

In this work, we attempt to detect tandem repeats directly in MinION signal. That is, given the signal measured by MinION during a single read, our goal is to determine which parts of the signal were produced, when a tandem repeat was passing through the nanopore.

By detecting tandem repeats directly in the signal, we attempt to avoid inaccuracy introduced during the basecalling step.

In Chapter 3 we present an approach that detects repetitive portions of the signal. In Chapter 4 we present another approach, that first preprocesses the signal using a part of a basecaller. Both approaches are experimantally tested and compared to a baseline approach in Chapter 5.

Chapter 2

Related work

2.1 Repeat detection on nucleotide sequences

An extensive amount of research has been done on detection of tandem repeats in nucleotide sequences and many algorithms were proposed. Some algorithms require prior knowledge of the repeat's pattern [5]. Others can only find repeats of a specified period [3]. There are also algorithms that can detect tandem repeats without prior knowledge of their pattern or period.

Tandem repeat finder proposed by Benson detects perfect and imperfect tandem repeats of any period [2]. It first detects pairs of short substrings that match exactly. Then, if a statistically significant amount of these matching pairs with similar distances between matching fragments is found at one location, they are joined and extended.

Mreps, a tool by Kolpakov et al., uses an exact combinatorial algorithm to collect all maximal runs of tandem repeats with up to k mismatches, and then post-processes them using a heuristic [9].

Tantan, an algorithm proposed by Frith, can detect (perfect and imperfect) tandem repeats with periods up to some threshold. It uses a hidden Markov model to detect nucleotides that belong to right sides of repeats. The approach we propose in Chapter 4 is inspired by this algorithm.

2.2 Repeat detection in nanopore signal

Much less research has been done on detection of repeats in nanopore signal. In his bachelor thesis, Molčan attempts to detect repetitive sequences in signal from MinION [13]. The signal is discretized into several levels and thus transformed into a sequence over some relatively small alphabet. Runs of occurences of the same letter are contracted into a single occurence (e.g. `AAAAABAAACC` gets contracted to `ABAC`). On the resulting sequence, repeats are detected using Tantan.

This approach is reportedly more accurate than basecalling the signal and using a repeat

finder on the resulting sequcence. However, we believe that the accuracy can still be significantly improved.

Chapter 3

Detecting repetitive signal

In this chapter we present an approach to repeat detection based on the idea that repetitive sequences of nucleotides produce repetitive signal. Therefore, finding repetitive portions of the signal could be sufficient for identifying parts of the signal corresponding to tandem repeats.

3.1 Signal normalization

A standard preprocessing step when working with MinION signal is to normalize it. We normalize the signal using *median normalization*. Let S be the signal from MinION (a sequence of real numbers) and let n be its length. We transform it to a sequence of real numbers S' defined by

$$s'_i = \frac{s_i - M}{D} \qquad \text{for } i \in \{1, 2, \ldots, n\},$$

where

$$M = \text{median}\left(s_1, s_2, \ldots, s_n\right)$$

and

$$D = \text{median}\left(|s_1 - M|, |s_2 - M|, \ldots, |s_n - M|\right).$$

3.2 Signal self-alignment

In order to find tandem a repeat, we search for two overlapping parts of the signal that look similar and are as long as possible. Ideally, these two parts will correspond to the left side and the right side of a tandem repeat. To find similar parts of the signal, we use the Smith-Waterman algorithm.

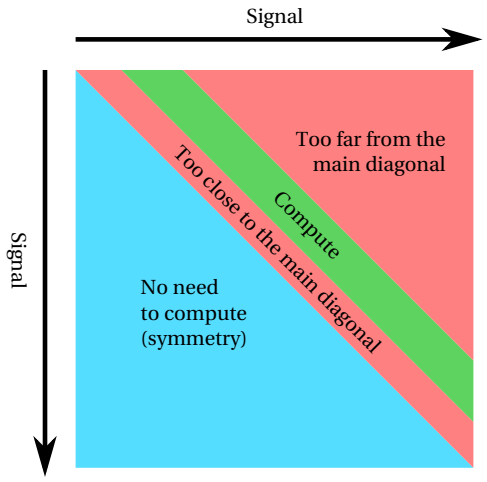

Figure 3.1: The part of the Smith-Waterman matrix that is computed.

3.2.1 Smith-Waterman algorithm

The Smith-Waterman algorithm uses dynamic programming to find the best local alignment of two sequences[17]. The input for the algorithm consists of two sequences A and B and a scoring function f that assigns positive score to pairs of similar elements and negative score to pairs of dissimilar elements. The algorithm computes an alignment matrix D of size $|A| \times |B|$, where $D_{i,j}$ is the score of the best local alignment that ends by aligning a_i to b_j. Any alignment that aligns $a_{i_1} a_{i_2} \dots a_{i_k}$ to $b_{j_1} b_{j_2} \dots b_{j_k}$ can be viewed as a path in the alignment matrix that visits cells $D_{i_1,j_1}, D_{i_2,j_2}, \dots, D_{i_k,j_k}$.

We use Smith-Waterman algorithm to find local alignment of the signal *to itself*. To prevent the algorithm from aligning segments of signal to themselves, or almost to themselves, we constrain it to those cells of its matrix that are sufficiently far away from the main diagonal. As the matrix of the Smith-Waterman algorithm is symmetric in our case, we only need to compute its upper triangle. To reduce memory consumption and running time (and at the cost of losing repeats with long periods), we also constrain the algoirthm to cells that are not too far away from the main diagonal. Therefore, the only part of the matrix we actually compute is a band at some distance from the main diagonal (see Figure 3.1).

As a scoring function, we use

$$f(x, y) = b - (x - y)^2,$$

where b a positive real number that is a paramter of our algorithm. The function awards a constant score bonus b for each aligned point and subtracts score depending on the signal difference between the aligned points.

The DNA strand passes through the nanopore at non-constant speed, therefore the signal corresponding to the left side and the signal corresponding to the right side of a read can be stretched in different ways. Therefore, we allow our version of the Smith-Waterman algorithm

to align a single point in one copy of the signal to multiple points in the other copy. This is a common technique for aligning seqences which vary in speed, known as *dynamic time warping* [15]. To avoid excessive signal stretching in the alignment, we limit the number of points aligned to a single point to be at most five.

When aligning a signal point with value x to multiple points with values $y_1, y_2, \ldots, y_k$, we define the total score for this alignment as

$$f(x, y_1, y_2, \ldots, y_k) = b\frac{k+1}{2} - \sum_{i=1}^{k}(x - y_i)^2.$$

This is to ensure that the total score bonus awarded for aligning a sequence of length m to a sequence of length n is always $b(m+n)/2$, regardless of the exact alignment. If we used a scoring function such as

$$f'(x, y_1, y_2, \ldots, y_k) = kb - \sum_{i=1}^{k}(x - y_i)^2,$$

the algorithm would have an incentive to unnecessarily stretch the aligned signal in order to collect more bonus score.

3.2.2 Multiple repeats

The Smith-Wateman algorithm a single local alignment with maximum score. However, there might be multiple repeats in the read we are processing. To obtain multiple local alignments (an thus, possibly detect multiple repeats), we use the following heuristic.

Algorithm 1. Start with a matrix where the whole band defined in Section 3.2.1 is permitted (see Figure 3.1). Run the following procedure on this band:

FindRepeats(band):

1. Find the best local alignment in the band, using the Smith-Waterman algorithm.
2. If the two aligned parts found in step 1 overlap, report them as a repeat.
3. Block a part of the permitted band near the alignment path from step 1 (see Figure 3.2). This splits the permitted band into two parts. Run this procedure recursively on both of them.

The procedure terminates, if it is called on an empty band, or if no alignment with positive score is found in step 1.

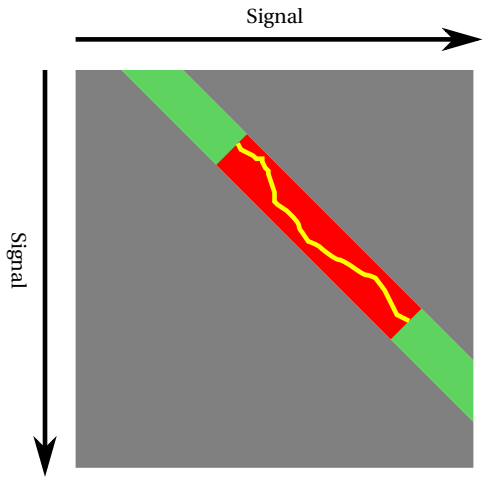

Figure 3.2: A part of the Smith-Waterman matrix that gets blocked when an alignment is found (step 3 of Algorithm 1).

Let n be the length of the signal and let w be the width of the band. The worst case time complexity of our algorithm is $\Theta(n^2 w)$, as we might end up running the Smith-Waterman algorithm $\Theta(n)$ times. However, if the repeats are found in a random order, the expected time complexity becomes $O(n \log nw)$, similar to the Quicksort algorithm.

In later chapters of this thesis, we will refer to the approach presented in this chapter as the *DTW approach*.

Chapter 4

Detecting repeats with help from basecaller

The DTW approach presented in Chapter 3 suffers from the signal's sensitivity to context. The signal measured by MinION is not only affected by the nucleotide currently in the nanopore, but also by several nucleotides around it. Because of this, the signal near the repeat edges can differ significantly from the corresponding signal in other units of the repeat. It also amplifies the effect of imperfections: making a small change in the nucleotide sequence can cause a significantly bigger change in the signal.

The approach presented in this chapter attepmts to overcome this problem by using a part of a basecaller. The basecaller we are using is Chiron, developed by Teng et al. [18]. We process our signal with Chiron, as if we were basecalling it. However, we do not use the output nucleotide sequence. Instead, we use one of Chiron's intermediate results. This intermediate result already represents the information in terms of nucleotides (not in terms of eletric current), but it still contains basecaller's uncertainity about the exact nucleotides that passed through the nanopore.

4.1 Chiron

Chiron uses a deep neural network that consists of a convolutional neural network (CNN) followed by a recurrent neural network (RNN) and a fully connected layer (FC). The FC layer produces a *connectionist temporal classification* output, that is further decoded to compute the final nucleotide sequence. The architecture of the network is illustrated in Figure 4.1.

4.1.1 Connectionist temporal classification

Connectionist temporal classification (CTC) is a technique for labelling sequences using neural networks. It was first introduced by Graves et al. [8]. CTC is used for tasks, where a longer input sequence is to be translated into a shorter output sequence of labels (a *labelling*), and

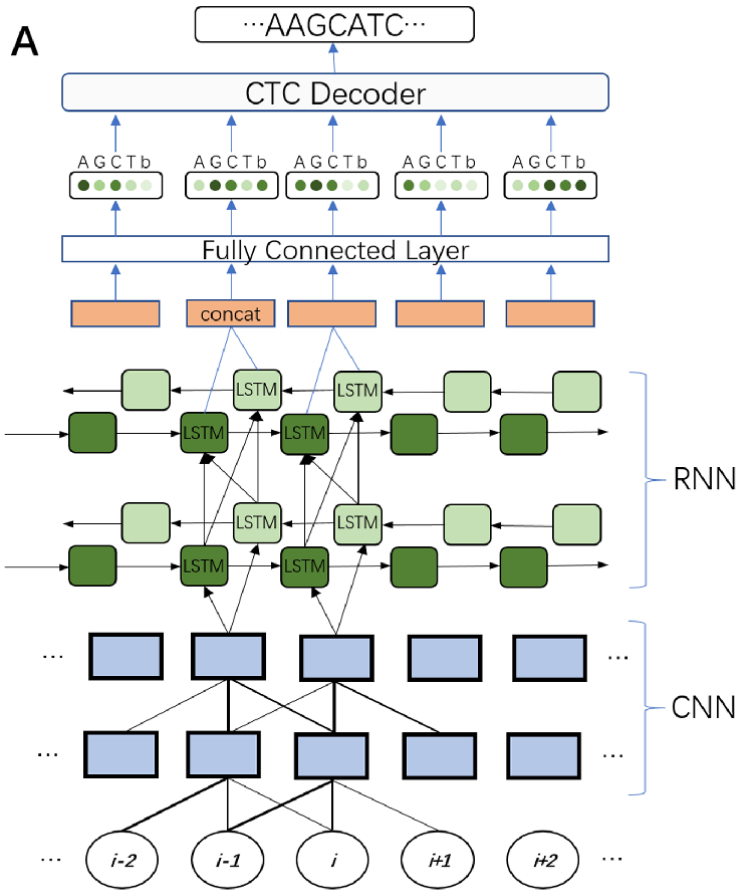

Figure 4.1: Architecture of the neural network used in Chiron. Figure from [18].

the alignment between the input sequence and the output sequence is unknown. Examples of such tasks include speech recognition, gesture recognition and basecalling.

Let I be the input sequence (in our case, the signal), let $n = |I|$ and let $\mathcal{L}$ be the set of labels used in the output labelling (in our case, $\mathcal{L} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$). As the labelling is never longer than the input sequence, the set of possible labellings is $\bigcup_{i=0}^{n} \mathcal{L}^i$ (or $\mathcal{L}^{\leq n}$ for short).

A CTC neural network has a softmax output layer that produces $|\mathcal{L}| + 1$ numbers for each element of the input. Let us denote these outputs as $b_{i,l}$ for $i \in \{1, 2, \ldots, n\}; l \in \bar{\mathcal{L}}$, where $\bar{\mathcal{L}} = \mathcal{L} \cup \{\emptyset\}$ and $\emptyset$ is a special new label, called *blank*. The outputs satisfy $\sum_{l \in \bar{\mathcal{L}}} b_{i,l} = 1$ for each i. Intuitively, these numbers represent the network's prediction of what was happening at individual points of the input.

The output of the CTC neural network can be interpreted as a probability distribution over $\bar{\mathcal{L}}^n$, where

$$\mathrm{P}(S) = \prod_{i=1}^{n} b_{i,s_i} \qquad \text{for each } S \in \bar{\mathcal{L}}^n.$$

A mapping f from $\bar{\mathcal{L}}^n$ to $\mathcal{L}^{\leq n}$ is defined by the following algorithm.

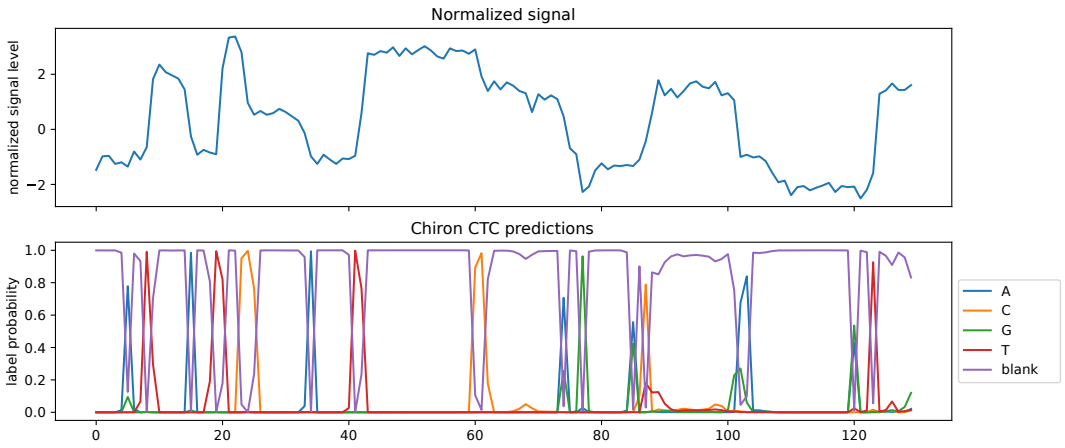

Figure 4.2: Top: level of the (normalized) signal measured by MinION. Bottom: corresponding CTC predictions produced by Chiron.

.

1. Contract each substring that consist of one label repeated multiple times into a single occurence of the label, e.g. `-AACCCAG--GCTTT-A` will be transformed into `-ACAG-GCT-A`.
2. Remove all blanks, e.g. `-ACAG-GCT-A` will be transformed into `ACAGGCTA`.

The probability of a labelling L is then defined as the sum of probabilities of all sequences from $\bar{L}^n$ that are mapped to L by f:

$$\mathrm{P}(L) = \sum_{S \in \bar{L}^n; f(S)=L} \mathrm{P}(S).$$

Labelling of the maximum probability is then chosen as the output. As computing the maximum probability labelling is an algorithmically challenging task, heuristics for its computiation are used in practice.

In the rest of this work we will refer to the output of the neural network (i.e. the values $b_{i,l}$) as *CTC predictions* and to the labels from $\bar{L}$ (including the blank label) as *CTC labels*.

To detect tandem repeats, we use CTC predictions produced by Chiron's neural network. Chiron's implementation does not output its CTC predictions by default, but it is open source and we modified it to log CTC predictions into a file. Figure 4.2 shows an example of CTC predictions produced by Chiron for a portion of signal.

CTC predictions can be seen as something between the raw signal and the final basecalled sequence. Searching for repetitions in CTC predictions avoids main problems with searching for repetitive signal or searching for repeats in basecalled sequences. CTC predictions are not as sensitive to surrounding nucleotides as the raw signal, which makes them more suitable for searching for imperfect or very short repeats. On the other hand, if an ambiguous signal is encountered, CTC predictions preserve the information about basecaller's uncertainity. This allows us to find repeats that we would not be able to find in basecalled sequences due to basecalling errors.

4.2 Our model

Our approach is inspired by the hidden Markov model used by Frith in Tantan [7]. However, Tantan can process nucleotide sequences, not CTC predictions. There are several key differencies between nucleotide sequences and CTC predictions:

1. In addition to the four bases $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, CTC label sequences also contain blank symbols $\varnothing$.
2. We do not have access to exact CTC label sequences, only to CTC predictions.
3. One nucleotide can be represented by its label repeated several times in a CTC label sequence.

In order to adapt Tantan's HMM to work on CTC predictions, we need to overcome all of these differences. In the following, we will address these differences one by one. We will start with a HMM similar to that used in Tantan and gradually work our way to our final model.

4.2.1 A model for nucleotide sequences

Let us start with a Hidden Markov Model for detection of repeats in nucleotide sequences. For now, we are only focusing on perfect repeats whose period is between l and h nucleotides. We will extend our model to detect imperfect repeats in Section 4.2.4.

States. The model has $h - l + 4$ hidden states. We denote them $\rho_l, \rho_{l+1}, \ldots, \rho_h, \beta, \alpha$ and ω. State β generates *background*, i.e. nucleotides that do not belong to the right side of any repeat. Thus, it emits nucleotides at random, from a fixed distribution over $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. For each i, state ρ_i generates right sides of repeats with period i. Therefore, it always emits the same nucleotide as the one emitted i positions before. States α and ω are non-emitting states – they are only used to interconnect other states. These two states are are not strictly necessary in the model, but their usage will simplify the model later. State α serves as an entry point for the repeat-generating part of the HMM. State ω is the exit point for this part of the HMM.

Transitions. When the model is in its background state β at some point in time, it stays in this state for the next step with probability $p_{\text{STAY\_BACKGROUND}}$. With the remaining probability $1 - p_{\text{STAY\_BACKGROUND}}$, it transitions into α. From α, the model transitions into one of the repeat-generating states. The probability of transition from α to any individual ρ_i is zd^{i-l}, where $d \in (0, 1)$ and $z = 1/\sum_{i=l}^{h} d^{i-l}$ is a normalization constant. That is, the probability of transition into a repeat-generating state decays geometrically with increased repeat period. Hovever, the model cannot transition into state ρ_i before generating its i-th nucleotide. When the model is in state ρ_i for some i, it stays in that state with probability $p_{\text{STAY\_REPEAT}}$ and transitions into

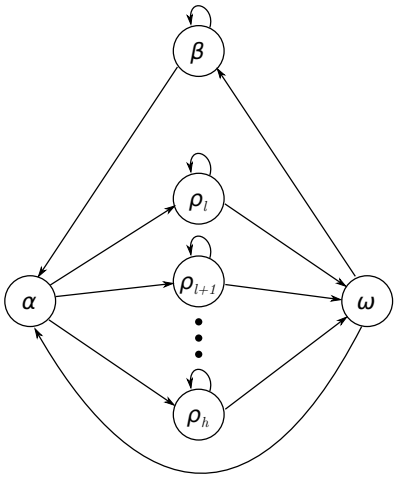

Figure 4.3: A HMM for nucleotide sequences.

ω with probability $1 - p_{\text{STAY\_REPEAT}}$. From ω, the model transitions into β with probability $p_{\text{STAY\_BACKGROUND}}$ and to α with probability $1 - p_{\text{STAY\_BACKGROUND}}$. That is, after the model finishes generating a repeat, it has a nonzero chance to immediately start generating another repeat. The model is illustrated in Figure 4.3.

This HMM can be used with a standard technique, such as Viterbi algorithm or posterior decoding, to determine which nucleotides in the sequence form repeats. More precisely, it can determine which nucleotides belong to right sides of repeats.

Note, however, that the model does not ensure that detected repeats have two or more units. Therefore, it can sometimes detect non-tandem repeats. This effect is partially mitigated by the fact that transitions into repeat-generating states of longer periods are assigned lower probability. This forces repeats with long periods to have longer right sides in order to stand out from the background.

To avoid detecting non-tandem repeats, we have two options. One options is to exclude them during a post-processing step. Another option is to set the decay factor d low enough to ensure that each detected repeat with period i has a right side at least i nucleotides long. We will return to this topic in Section 4.4.

4.2.2 Blanks in CTC label sequences

In this section we will extend our model to work on sequences of nucleotides augmented by blank labels ø. Note that we are still ignoring the fact that a single nucleotide can be encoded by several subsequent labels in CTC label sequences.

The straightforward approach of working with blanks as if they were a fifth type of nucleotide does not work, as blank positions and numbers do not need to be consistent across units of a repeat. For example, the CTC label sequence `AøøøCøTACøøTøøAøCT` encodes a tandem repeat whose units are encoded by `AøøøCøT`, `ACøøTøø` and `AøCT`.

To model blanks in repeats, we change our HMM in the following way. For each $i \in \{l, \ldots, h\}$, we introduce three new states γ_i, γ_i^* and λ_i in addition to ρ_i. States γ_i^* and λ_i are non-emitting states. State γ_i always emits blanks. State ρ_i continues to always emit the same label as has been emitted i positions back.

Transitions between these states are as follows. From ρ_i the model can transition into γ_i^* with probability $p_{\text{STAY\_REPEAT}}$ or into ω with probability $1 - p_{\text{STAY\_REPEAT}}$. From γ_i^* it can transition into γ_i with probability p_{BLANK}, or into λ_i with probability $1 - p_{\text{BLANK}}$. From γ_i the model can only transition into γ_{i+1}^* (with probability 1). Transitions from state λ_i are special. If the label generated i positions back is blank, the model transitions from λ_i into λ_{i-1} with probability 1. If the label generated i positoins back is a nucleotide, the model transitions from λ_i into ρ_i with probability 1. The edge-case transitions that would end in non-existing states (such as λ_{l-1}) point to ω instead.

We also modify how transitions from α work. For each i, we only allow transition from α to ρ_i if a nucleotide label has been generated i positions before. If necessary, probabilities of these allowed transitions are scaled, so that they add up to one. This ensures that states of type ρ_i never generate blanks.

In order to model blanks in the background we simply add blanks to the emission distribution of the background state β. The probability of transition from β to α needs to be decreased to $(1 - p_{\text{BLANK}})(1 - p_{\text{STAY\_BACKGROUND}})$, as there are now more opportunities to leave β, because CTC label sequences are longer than nucleotide sequences. This version of our model is illustrated in Figure 4.4.

The model generates right sides of repeats by emulating the following two pointers algorithm:

Algorithm 2. Start in a situation when the first unit of the repeat has already been generated (i.e. we are about to start generating the right side of the repeat). Let L be a pointer pointing at the first nucleotide of the left side. Let R be a pointer always pointing at the position of the label to be generated.

Until the right side is fully generated, repeat these three steps:

1. Generate a nucleotide label same as the label pointed at by L. Advance both L and R by one position.
2. Generate zero or more blank labels, advancing R by one position for each of them.
3. If L points at a blank label, advance it until it points at a nucleotide label.

The distance between L and R (and thus, the position of L) is encoded by the subscript index of the current state (i.e. when the model is in $\rho_i, \gamma_i^*, \gamma_i$ or λ_i, pointer L points i positions to the left from R). Step 1 is emulated by states ρ_i. Step 2 is emulated by following an alternating chain of states γ_i^* and γ_i, with i increasing. Step 3 is emulated by following a chain of λ_i states, with i decreasing.

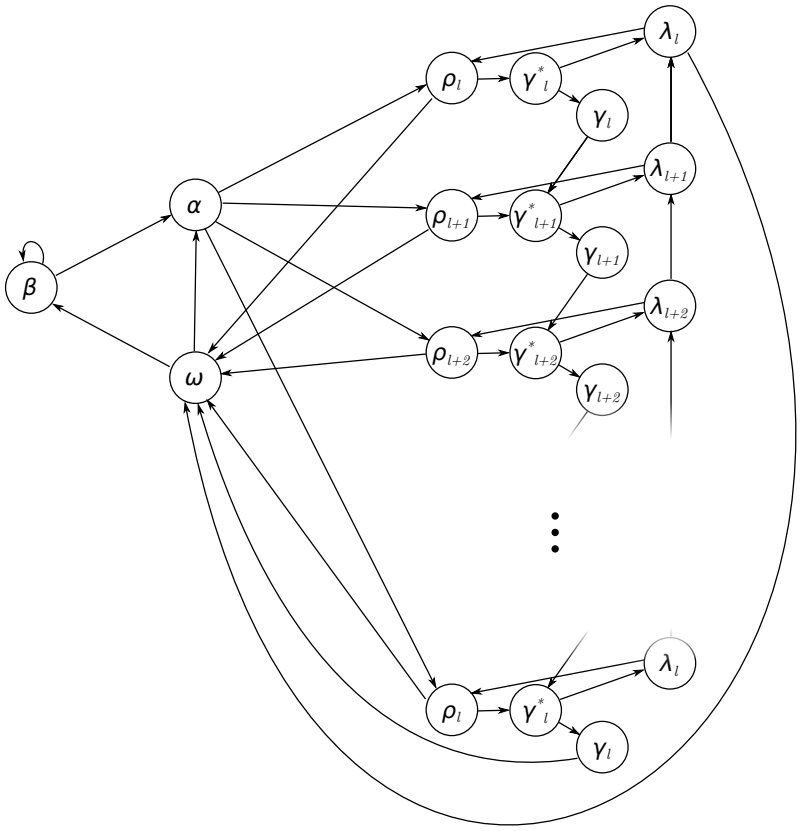

Figure 4.4: A HMM for CTC label sequences containing blanks.

Note that using our new model, we can no longer detect all repeats with period between l and h. A repeat can only be detected, if the distance of pointers L and R stays between l and h throughout the repeat's generation by algorithm 2.

4.2.3 Uncertainity of CTC predictions

A typical usage of Hidden Markov Models for sequence annotation looks as follows. There is a sequence of observable events we know and a sequence of hidden states we want to learn. States and transitions of the HMM determine prior probability of any sequence of hidden states. Emission distributions of the HMM's states determine conditional probability of any sequence of observable events, given a sequence of hidden states. Using this information and Bayes' theorem, we can then compute the conditional probability of any sequence of hidden states given the sequence of events we observed.

In our case, however, we do not know the exact CTC label sequence corresponding to our read. We only know the read's signal and CTC predictions computed from the signal by Chiron. Therefore, there is an additional layer of indirection. The sequence of hidden states affects the CTC label sequence, which in turn affects the observed signal.

To account for this extra layer of indirection, we need to adjust the inference algorithms we use (i.e. Viterbi algorithm and forward-backward algorithm). Let us begin by reviewing

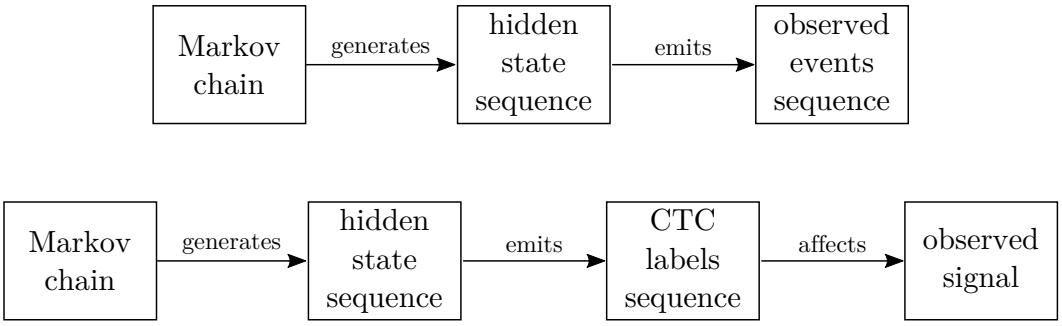

Figure 4.5: Top: a traditional usage of HMM for sequence labeling. Bottom: our situation.

how these inference algorithms work before the adjustment.

Throughout the rest of this chapter, we will encounter multiple models that estimate probabilities of certain events. To indicate probabilities estimated by a HMM, we will use P_{H}, to indicate probabilities estimated by Chiron we will use P_{C} and to indicate probability in general we will use P.

Inference without extra indirection

If we could directly observe the CTC label sequence corresponding to our nanopore read, we could compute the probability of any hidden state sequence, conditional on our observation. Let E be the CTC label sequence that we have observed, let H be a hidden state sequence of the same length and let $n = |E| = |H|$. Probability of H given that we observed E could be computed as

$$\mathrm{P}(H \mid E) = \frac{\mathrm{P}(E, H)}{\mathrm{P}(E)}.$$

As E is fixed, we have

$$\mathrm{P}(H \mid E) \propto \mathrm{P}(E, H).$$

To run our inference algorithms, we do not need to know the exact normalization factor. Thus, it is sufficient to model $\mathrm{P}(E, H)$. The probability $\mathrm{P}(E, H)$ is modelled by the HMM as

$$\mathrm{P}_{\mathrm{H}}(E, H) = \mathrm{P}_{\mathrm{H}}(H)\,\mathrm{P}_{\mathrm{H}}(E \mid H)$$

with

$$\mathrm{P}_{\mathrm{H}}(H) = \prod_{i=1}^{n} \mathrm{P}_{\mathrm{H}}\left(h_i \mid h_{i-1}\right), \tag{4.1}$$

$$\mathrm{P}_{\mathrm{H}}(E \mid H) = \prod_{i=1}^{n} \mathrm{P}_{\mathrm{H}}\left(e_i \mid h_i\right) \tag{4.2}$$

in case of a simple, order one HMM, or

$$\mathrm{P_H}(H) = \prod_{i=1}^{n} \mathrm{P_H}\left(h_i \mid h_{i-1}, e_1, e_2, \ldots, e_{i-1}\right), \tag{4.3}$$

$$\mathrm{P_H}(E \mid H) = \prod_{i=1}^{n} \mathrm{P_H}\left(e_i \mid h_i, e_1, e_2, \ldots, e_{i-1}\right) \tag{4.4}$$

if emission and transition probabilities depend on previously emitted labels (as is our case). Factors on the right side of equation 4.1 or 4.3 are HMM's transition probabilities. The meaning of expression $\mathrm{P_H}\left(h_1|h_0\right)$ (this expression is special, as h_0 is not defined) is the probability of our HMM starting in state h_1. Factors on the right side of equation 4.2 or 4.4 are HMM's emission probabilities.

Therefore, we can compute $\mathrm{P_H}(E, H)$ as:

$$\mathrm{P_H}(E, H) = \prod_{i=1}^{n} \left(\mathrm{P_H}\left(h_i \mid h_{i-1}, e_1, \ldots, e_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i, e_1, \ldots, e_{i-1}\right)\right).$$

Let $\mathcal{H}$ be the set of all states in the HMM. Both Viterbi algorithm and the forward part of the forward-backward algorithm are dynamic programming algorithms that compute

$$\mathrm{DP}[i][p] = \bigsqcup_{\substack{(p_j)_{j=1}^{i-1} \in \mathcal{H}^{i-1} \\ p_i = p}} \prod_{j=1}^{i} \left(\mathrm{P_H}\left(p_j \mid p_{j-1}, e_1, \ldots, e_{j-1}\right) \mathrm{P_H}\left(e_j \mid p_j, e_1, \ldots, e_{j-1}\right)\right) \tag{4.5}$$

for each $i \in \{1, \ldots, n\}, p \in \mathcal{H}$. The $\bigsqcup$ symbol stands for maximum in case of Viterbi algorithm, or sum in case of the forward algorithm. We will not discuss the backward part of the forward-backward algorihm, as it is analogous to its forward counterpart.

Values of DP can be computed efficiently using the following recurrence. For better readability, let $Q(x)$ be a syntactic shorthand for $\mathrm{P_H}\left(p_x \mid p_{x-1}, e_1, \ldots, e_{x-1}\right) \mathrm{P_H}\left(e_x \mid p_x, e_1, \ldots, e_{x-1}\right)$.

$$\begin{aligned}
\mathrm{DP}[i][p] &= \bigsqcup_{\substack{(p_j)_{j=1}^{i-1} \in \mathcal{H}^{i-1} \\ p_i = p}} \prod_{j=1}^{i} Q(j) = \\
&= \bigsqcup_{(p_j)_{j=1}^{i-2} \in \mathcal{H}^{i-2}} \bigsqcup_{p_{i-1} \in \mathcal{H}} \left(\left(\prod_{j=1}^{i-1} Q(j)\right) Q(i)\right) = \\
&= \bigsqcup_{p_{i-1} \in \mathcal{H}} \left(Q(i) \bigsqcup_{(p_j)_{j=1}^{i-2} \in \mathcal{H}^{i-2}} \left(\prod_{j=1}^{i-1} (Q(j))\right)\right) = \\
&= \bigsqcup_{p_{i-1} \in \mathcal{H}} \left(\mathrm{P_H}\left(p \mid p_{i-1}, e_1, \ldots, e_{i-1}\right) \mathrm{P_H}\left(e_i \mid p, e_1, \ldots, e_{i-1}\right) \mathrm{DP}[i-1][p_{i-1}]\right).
\end{aligned}$$

Inference with extra indirection

Let O be the signal we have observed, let H be a hidden state sequence of the same length and let $n = |O| = |H|$. As O is fixed, we have $\mathrm{P}(H \mid O) \propto \mathrm{P}(O, H)$.

We can estimate $\mathrm{P}(O, H)$ by considering all possible CTC label sequences that could correspond to our read. Let $\bar{\mathcal{L}} = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, \varnothing\}$ be the set of all possible CTC labels. We can compute $P(O, H)$ as

$$\mathrm{P}(O, H) = \sum_{E \in \bar{\mathcal{L}}^n} \mathrm{P}(E, O, H) = \sum_{E \in \bar{\mathcal{L}}^n} \mathrm{P}(E, H)\,\mathrm{P}(O \mid E, H) = \sum_{E \in \bar{\mathcal{L}}^n} \mathrm{P}(E, H)\,\mathrm{P}(O \mid E)$$

The last equality holds under the assumption that the observed signal O does not depend on the hidden state sequence H, except for the indirect dependence via E. Values of $\mathrm{P}(E, H)$ for any $E \in \bar{\mathcal{L}}^n$ are modelled by our HMM. It remains to estimate the conditional probabilities $\mathrm{P}(O \mid E)$ for $E \in \bar{\mathcal{L}}^n$. To do that, we use CTC predictions produced by Chiron.

Estimating $P(O \mid E)$ from CTC predictions

Chiron processes the observed signal O and produces CTC predictions $b_{i,e}$ for each $i \in \{1, 2, \ldots, n\}; e \in \bar{\mathcal{L}}$. These predictions can be viewed as the conditional distribution of E given O for any $E \in \bar{\mathcal{L}}^n$:

$$\mathrm{P}_{\mathrm{C}}(E \mid O) = \prod_{i=1}^{n} b_{i,e_i}\ .$$

This is conditional in the opposite direction to what we need. Chiron contains some understanding of how signal and CTC label sequences are related. Even though it is a neural network that we use as a blackbox, we can assume it makes its predictions according to some joint probability distribution of E and O it implicitly contains [1].

If we knew the marginal distribution of E, we could compute the value of

$$\frac{\mathrm{P}_{\mathrm{C}}(O \mid E)}{\mathrm{P}_{\mathrm{C}}(O)} = \frac{\mathrm{P}_{\mathrm{C}}(E \mid O)}{\mathrm{P}_{\mathrm{C}}(E)}.$$

As O is fixed,

$$\mathrm{P}_{\mathrm{C}}(O \mid E) \propto \frac{\mathrm{P}_{\mathrm{C}}(E \mid O)}{\mathrm{P}_{\mathrm{C}}(E)}.$$

Thus, for our purposes, it is sufficient to compute $\mathrm{P}_{\mathrm{C}}(E \mid O)\,/\mathrm{P}_{\mathrm{C}}(E)$.

It remains to estimate the marginal probability distribution of E used by Chiron. Chiron predicts CTC labels for individual positions as if they were independent. Therefore, it seems

[1] There are actually infinitely many joint distributions of E and O that are consistent with Chiron's predictions (one for any choice of marginal distribution of O).

reasonable to use a distribution of E with equally simple structure. That is, we will assume that the CTC label at each position of E is chosen independently from the same distribution over possible labels $\bar{L}$:

$$\mathrm{P_C}(E) = \prod_{i=1}^{n} \mathrm{P_C}\left(e_i\right).$$

Then, our estimate of $\mathrm{P_C}(O \mid E) / \mathrm{P_C}(O)$ is:

$$\frac{\mathrm{P_C}(E \mid O)}{\mathrm{P_C}(E)} = \frac{\prod\limits_{i=1}^{n} b_{i,e_i}}{\prod\limits_{i=1}^{n} \mathrm{P_C}\left(e_i\right)} = \prod_{i=1}^{n} \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)}.$$

Inference algorithms for order one HMMs

We can now model a quantity proportional to $\mathrm{P}(H \mid O)$ as:

$$\mathrm{P}(H \mid O) \propto \sum_{E \in \bar{L}^n} \mathrm{P_H}(E, H)\, \mathrm{P_C}(O \mid E) \propto \sum_{E \in \bar{L}^n} \mathrm{P_H}(E, H)\, \frac{\mathrm{P_C}(E \mid O)}{\mathrm{P_C}(E)}.$$

If transition and emission probabilities in our HMM only depended on the current state (equations 4.1 and 4.2), we could rewrite this quantity as:

$$\sum_{E \in \bar{L}^n} \mathrm{P_H}(E, H)\, \frac{\mathrm{P_C}(E \mid O)}{\mathrm{P_C}(E)} = \tag{4.6}$$

$$= \sum_{E \in \bar{L}^n} \prod_{i=1}^{n} \left(\mathrm{P_H}\left(h_i \mid h_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i\right)\right) \prod_{i=1}^{n} \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} = \tag{4.7}$$

$$= \sum_{E \in \bar{L}^n} \prod_{i=1}^{n} \left(\mathrm{P_H}\left(h_i \mid h_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right) = \tag{4.8}$$

$$= \prod_{i=1}^{n} \sum_{e_i \in \bar{L}} \left(\mathrm{P_H}\left(h_i \mid h_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right) \tag{4.9}$$

Note that we could do the step from expression 4.8 to expression 4.9, as the expression

$$\left(\mathrm{P_H}\left(h_i \mid h_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right)$$

does not depend on e_j for any $j \neq i$.

Our modified versions of inference algorithms will compute

$$\mathrm{DP}[i][p] = \bigsqcup_{\substack{(p_j)_{j=1}^{i-1} \in \mathcal{H}^{i-1} \\ p_i = p}} \prod_{j=1}^{i} \sum_{e_j \in \bar{L}} \left(\mathrm{P_H}\left(p_j \mid p_{j-1}\right) \mathrm{P_H}\left(e_j \mid p_j\right) \frac{b_{j,e_j}}{\mathrm{P_C}\left(e_j\right)} \right) = \tag{4.10}$$

$$= \bigsqcup_{p_{i-1} \in \mathcal{H}} \left(\sum_{e_i \in \bar{L}} \left(\mathrm{P_H}\left(p \mid p_{i-1}\right) \mathrm{P_H}\left(e_i \mid p\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right) \mathrm{DP}[i-1][p_{i-1}] \right). \tag{4.11}$$

Inference algorithms for our HMM

Transition and emission probabilities of our HMM described in Section 4.2.2 depend on previously generated CTC labels (equations 4.3 and 4.1). We could try to estimate quantity 4.6 in a similar manner as in the previous section. That would work up to the analogy of expression 4.8.

$$\sum_{E \in \bar{L}^n} \mathrm{P_H}(E, H) \frac{\mathrm{P_C}(E \mid O)}{\mathrm{P_C}(E)} = \tag{4.12}$$

$$= \sum_{E \in \bar{L}^n} \prod_{i=1}^{n} \left(\mathrm{P_H}\left(h_i \mid h_{i-1}, e_1, \ldots, e_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i, e_1, \ldots, e_{i-1}\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right). \tag{4.13}$$

However, as

$$\left(\mathrm{P_H}\left(h_i \mid h_{i-1}, e_1, \ldots, e_{i-1}\right) \mathrm{P_H}\left(e_i \mid h_i, e_1, \ldots, e_{i-1}\right) \frac{b_{i,e_i}}{\mathrm{P_C}\left(e_i\right)} \right)$$

depends on e_js for $j < i$, we cannot do the next step (i.e. bringing the sum inside the product). We are not aware of any computationally feasible way of circumventing this problem. Therefore, we approximate transition and emission probabilities that depend on previously generated labels, by expressions that do not have this dependence.

Only states in our HMM whose emission distributions depend on previously generated CTC labels are $\rho_l, \ldots, \rho_h$, where for any $i \in \{l, \ldots, h\}; j > i$ and $e \in \bar{L}$ we have

$$\mathrm{P_H}\left(e \mid \rho_i, e_1, \ldots, e_{j-1}\right) = \begin{cases} 1 & \text{if } e = e_{j-i} \\ 0 & \text{otherwise} \end{cases}$$

We approximate this by

$$\mathrm{P_H}\left(e \mid \rho_i, e_1, \ldots, e_{j-1}\right) \approx b_{j-i,e} \,.$$

Transitions probabilities from λ_i and from α depend on previously generated labels. For transitions from λ_i, we have

$$\mathrm{P_H}\left(\rho_i \mid \lambda_i, e_1, \ldots, e_{j-1}\right) = \begin{cases} 0 & \text{if } e_{j-i} \text{ is blank} \\ 1 & \text{otherwise} \end{cases}$$

$$\mathrm{P_H}\left(\lambda_{i-1} \mid \lambda_i, e_1, \ldots, e_{j-1}\right) = \begin{cases} 1 & \text{if } e_{j-i} \text{ is blank} \\ 0 & \text{otherwise} \end{cases}$$

which we approximate by

$$\mathrm{P_H}\left(\rho_i \mid \lambda_i, e_1, \ldots, e_{j-1}\right) \approx 1 - b_{j-i,\emptyset}$$
$$\mathrm{P_H}\left(\lambda_{i-1} \mid \lambda_i, e_1, \ldots, e_{j-1}\right) \approx b_{j-i,\emptyset}$$

For each i, we define the transition probability from α to ρ_i to be

$$\mathrm{P_H}\left(\rho_i \mid \alpha, e_1, \ldots, e_{j-1}\right) = \frac{d^{i-l}(1 - b_{j-i,\emptyset})}{z},$$

where z is a normalization factor ensuring that transition probabilities from α add up to one. With these approximations, we can run the inference algorithms in the same way as described in the previous section.

4.2.4 Imperfect repeats

Our current version of the model can only recognize right sides of perfect tandem repeats. In this section we extend it to also detect repeats where subsequent units do not always look exactly alike. There are three types of imperfections we need to model.

- *Mismatch* – corresponding positions of two subsequent units contain different nucleotides.
- *Insertion* – a unit contains an extra nucleotide that is not present in the previous unit.
- *Deletion* – a nucleotide that is present in one unit is omitted in the next unit.

We model each of these imperfections separately.

Modelling mismatches

To model mismatches, we modify emission distributions of states $\rho_l, \ldots, \rho_h$. For each i, we allow the state ρ_i to emit a CTC label different from the label generated i positions before, with some small probability. For any $e_1, e_2 \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, we define the probability of generating e_2 if the label generated i positions before was e_1 to be $M_{e_1 e_2}$. Our approximation of emission probabilities in ρ_i changes to

$$\mathrm{P_H}\left(e \mid \rho_i, e_1, \ldots, e_{j-1}\right) \approx \sum_{e' \in \{\mathtt{A},\mathtt{C},\mathtt{G},\mathtt{T}\}} \frac{b_{j-i,e'}}{1 - b_{j-i,\emptyset}} M_{e'e} \qquad \text{for each } e \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}.$$

When the model is in the state ρ_i, we already know that the label generated i positions back was not a blank. For that reason, we normalize values $b_{j-i,e'}$ by dividing them by $(1 - b_{j-i,\emptyset})$.

Modelling insertions

Insertions are nucleotides in the right side of the repeat that do not have a pair in its left side. In this regard, they are similar to blanks in the right side. Thus, we can generate them in states $\gamma_l, \ldots, \gamma_h$. First, we need to slightly modify transition probabilities from states $\gamma_l^*, \ldots, \gamma_h^*$. The probability of transition from γ_i^* to λ_i decreases from $1 - p_{\text{BLANK}}$ to $(1 - p_{\text{BLANK}})(1 - p_{\text{INSERTION}})$ and the probability of transition to γ_i increases to $p_{\text{BLANK}} + (1 - p_{\text{BLANK}})p_{\text{INSERTION}}$. Denote the last quantity $p_{\text{BLANK\_OR\_INSERTION}}$. When the model is in state γ_i, it generates a blank with probability $p_{\text{BLANK}}/p_{\text{BLANK\_OR\_INSERTION}}$ and a nucleotide with the remaining probability.

Modelling deletions

Deletions are nucleotides in the left side that do not have a matching nucleotide in the right side. Therefore, we can model them in a similar way as we model blanks in the left side. We modify the transition probabilities from states $\lambda_l, \ldots, \lambda_h$ in the following way. If the model is in state λ_i and the CTC label generated i positions back is a nucleotide, it transitions into λ_{i-1} with probability p_{DELETION} or into ρ_i with probability $1 - p_{\text{DELETION}}$. If the CTC label generated i positions before is blank, the only possible transition from λ_i is to λ_{i-1}. Approximations of these probabilities used by inference algorithms change to:

$$\begin{aligned}\mathrm{P_H}\left(\rho_i \mid \lambda_i, e_1, \ldots, e_{j-1}\right) &= (1 - b_{j-i,\emptyset})(1 - p_{\text{DELETION}}) \\ \mathrm{P_H}\left(\lambda_{i-1} \mid \lambda_i, e_1, \ldots, e_{j-1}\right) &= b_{j-i,\emptyset} + p_{\text{DELETION}}(1 - b_{j-i,\emptyset}).\end{aligned}$$

The effect of modelling imperfections in repeats on the performance of our model is evaluated in Section 5.3.2.

4.2.5 Repeated CTC labels

The last problem we need to address is the fact, that a single nucleotide can be represented by multiple subsequent CTC labels in the CTC label sequence. We will refer to this phenomenon as *label lingering*. Moreover, if two nucleotides of the same type occur next to each other in the nucleotide sequence, they must be separated by at least one blank in the CTC label sequence (otherwise they would be considered a single nucleotide).

In order to model this, we need our HMM to emulate a more complex version of Algorithm 2 (the following version ignores imperfect repeats):

Algorithm 3. Start in the same situation as in Algorithm 2 (i.e. pointers L and R point to the first nucleotide of the left side and the right side of the repeat, respectively).

Until the right side is fully generated, repeat these steps:

1. Generate a nucleotide label same as the label pointed at by L. Advance both L and R by one position.
2. **Generate zero or more copies of the same nucleotide as generated in step 1, advancing R by one position for each of them.**
3. Generate zero or more blank labels, advancing R by one position for each of them. **If no blanks were generated in this step, the next generated nucleotide label must differ from the label generated in step 1.**
4. **If L points at a label identical to the label it pointed at in step 1, advance it until it points at a different label.**
5. If L points at a blank label, advance it until it points at a nucleotide label.

Algorithm 3 does not account for imperfect repeats. To model imperfect repeats, we need to extend it even further:

Algorithm 4. Start in the same situation as in Algorithm 3 Until the right side is fully generated, repeat these steps:

1. Generate a nucleotide label **depending on** the label pointed at by L **and the probability matrix M**. Advance both L and R by one position.
2. Generate zero or more copies of the same nucleotide as generated in step 1, advancing R by one position for each of them.
3. Generate zero or more blank labels, advancing R by one position for each of them. If no blanks were generated in this step, the next generated nucleotide label must differ from the label generated in step 1.
4. **Generate zero or more insertions (an insertion consists of one or more copies of a nucleotide label and zero or more blanks). Advance R for each label generated. If any labels are generated this way and last of them is a nucleotide label, the next generated nucleotide label must be different.**
5. If L points at a label identical to the label it pointed at in step 1, advance it until it points at a different label.
6. If L points at a blank label, advance it until it points at a nucleotide label.
7. **Advance L through zero or more deletions. (a deletion consists of a nucleotide label, followed by its possible copies and by a maximal sequence of blanks).**

Between the steps of Algorithm 4 we need to keep track of the last generated label, to ensure proper separation of subsequent occurences of the same nucleotide by blanks. Between steps 1 and 5 we also need to preserve the information about label pointed at by L during step 1. This information can be stored in the HMM's state. It will, however, require us to considerably increase the number of states our HMM has.

States. For each $i \in \{l, \ldots, h\}, x, y \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, state ρ_i is be replaced by sixteen states $\rho_{i,x,y}$, where the state $\rho_{i,x,y}$ always emits the label y and also stores the information that during step 1, L pointed at an occurence of x. State γ_i is replaced by four states $\gamma_{i,x}$ for $x \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, where x indicates the nucleotide pointed at by L during step 1. The state λ_i is replaced by twenty-five states $\lambda_{i,x,y}$ for $x, y \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}, \emptyset\}$, where y indicates the last generated label (and thus controls which nucleotide labels can be generated next), and x indicates the label pointed at by L before its last advancement (and thus controls when we return to step 1). For simpliciy, we drop states γ_i^*. To model insertions and deletions (steps 4 and 7), we will reuse the existing states.

Transitions. From states of type $\rho_{i,x,y}$ the model transitions as if following these rules:

- With probability p_{LINGER}, transition into $\rho_{i+1,x,y}$. This emulates step 2 of Algorithm 4.
- Otherwise, transition into $\gamma_{i+1,x}$ with probability p_{BLANK}.
- Otherwise, transition into ω with probability $1 - p_{\text{STAY\_REPEAT}}$ (thus ending the current repeat).
- Otherwise, with probability $p_{\text{INSERTION}}$, transition into one of states $\rho_{i+1,x,z}$ for $z \neq y$. This emulates generation of insertions (step 4).
- Otherwise, transition into $\lambda_{i,x,y}$, proceeding to emulate step 5.

The order of these rules matters, as a rule can be only applied if previous rules were not. For example, probability of transitioning into $\lambda_{i,x,y}$ is

$$(1 - p_{\text{LINGER}})(1 - p_{\text{BLANK}})p_{\text{STAY\_REPEAT}}(1 - p_{\text{INSERTION}}).$$

From states of type $\gamma_{i,x}$, there is a similar list of rules.

- With probability p_{BLANK} transition into $\gamma_{i+1,x}$.
- Otherwise, transition into ω with probability $1 - p_{\text{STAY\_REPEAT}}$.
- Otherwise, with probability $p_{\text{INSERTION}}$ transition into one of states $\rho_{i+1,x,y}$ for any y.
- Otherwise, transition into $\lambda_{i,x,\emptyset}$.

Transitions from states of type $\lambda_{i,x,y}$ are as follows. Let z be the label generated i steps before.

- If $z = x$, transition into $\lambda_{i-1,x,y}$. This emulates step 5.

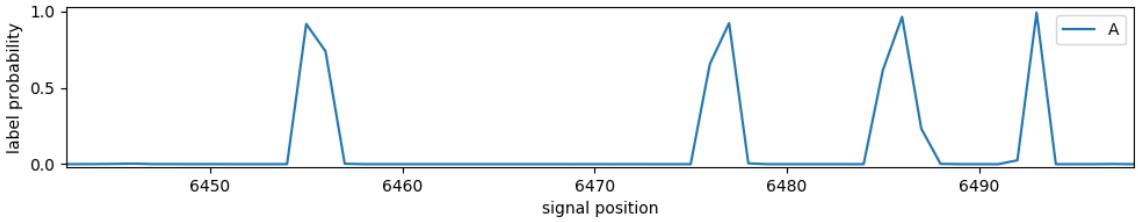

Figure 4.6: Chiron's CTC prediction for label A, on a small portion of signal.

- If $z \neq x$ and $z = \emptyset$, transition into $\lambda_{i-1,\emptyset,y}$. This emulates step 6.
- If $z \neq x$ and $z \neq \emptyset$, transition into $\lambda_{i-1,z,y}$ with probability p_{DELETION} (emulating step 7). With probability $1 - p_{\text{DELETION}}$ transition into one of the states $\rho_{i,z,w}$ for $w \neq y$.

Note that these transition deviate from emulating Algorithm 4 in a small detail: pointer L is not advanced in step 1. Instead, it is advanced after step 4.

An alternative approach

We have considerably increased the size of our HMM, as we moved from $4(h - l + 1) + 3$ states to $45(h - l + 1) + 3$. The number of possible transitions has also increased. This has a sizeable negative impact on speed of inference algorithms that we want to run with our model. Therefore, we devised an alternative, heuristic approach for dealing with repeated CTC labels, that does not require changing our HMM. Instead, we will preprocess the CTC predictions returned from Chiron.

The preprocessing is based on the observation that Chiron's CTC predictions for nucleotide labels tend to be close to zero for most signal positions, with short periods of high probability, that usually only last for several signal positions (see Figure 4.6). If we narrow these periods of high probability to a single signal position, the predicted probability of CTC label sequences containing runs of the same nucleotide label repeated more than once will decrease to negligible.

We transform Chiron's CTC predictions $b_{i,l}$ into $b'_{i,l}$ in the following way:

$$b'_{i,l} = \begin{cases} b_{i,l} & \text{if } b_{i,l} \geq b_{i-1,l} \text{ and } b_{i,l} > b_{i+1,l} \\ b_{i,l}/100 & \text{otherwise} \end{cases}$$

That is, CTC predictions for nucleotide labels that are preserved at local maxima, while they are reduced at all other positions. We chose not to reduce them all the way to zero, in order to avoid pathological cases.

We call this transformation the *peak filter*.

From now on, we will refer to the more complex HMM introduced in Section 4.2.5 as to the *full* model and to the simpler HMM from Section 4.2.4 as to the *simple* model. Perfor-

mance of the two strategies presented in this chapter (the full model and the simple model with peak filter) is compared in Section 5.3.1.

4.3 Parameter estimation

Our HMM has several parameter that need to be estimated before we can use it.

Parameter $p_{\text{STAY\_BACKGROUND}}$ is the probability that a background nucleotide will be followed by another background nucleotide in the nucleotide sequence. Parameter $p_{\text{STAY\_REPEAT}}$ is the probability that a nucleotide belonging to the right side of a repeat will be followed by another nucleotide from the same repeat. Tantan has two parameters with the same meaning as our $p_{\text{STAY\_BACKGROUND}}$ and $p_{\text{STAY\_REPEAT}}$. Thus, we can use Tantan's default values of $p_{\text{STAY\_BACKGROUND}} = 0.995$ and $p_{\text{STAY\_REPEAT}} = 0.95$ (more precisely, Tantan's parameters are negations of our $p_{\text{STAY\_BACKGROUND}}$ and $p_{\text{STAY\_REPEAT}}$, and their default probabilities are 0.005 and 0.05, respectively).

The decay parameter d, used for transitions from α, is also similar to a parameter in Tantan. However, as Tantan runs on nucleotide sequences, it typically needs ten times less states than our HMM. By default, tantan uses a decay value of 0.9. To match this, we will use $0.99 \approx \sqrt[10]{0.9}$.

The CTC label distributions used in the background state can be estimated by averaging CTC predictions over a large number of signal positions. Parameter p_{BLANK} (the probability that a CTC label is blank) can be also estimated this way.

Parameters for modelling repeat imperfections ($p_{\text{INSERTION}}$, p_{DELETION} and $M_{i,j}$ for $i, j \in \{\texttt{A}, \texttt{C}, \texttt{G}, \texttt{T}\}$) can be estimated by observing how often these types of imperfections occur in repeats of some known nucleotide sequence, ideally similar to the sequence our reads come from.

Parameter p_{LINGER} is the probability that a nucleotide CTC label will be followed by the same CTC label. We can estimate this parameter by using reads, where the number of sequenced nucleotides is known (for instance, reads that are aligned to some accurate reference nucleotide sequence). First, we add up CTC predictions for nucleotide labels (not blank) for all positions in the signal, obtaining the expected number of nucleotide CTC labels in the sequence corresponding to the read. Then, we divide this number by the actual number of nucleotides in the read, thus getting the average number of CTC labels per nucleotide. Let this number be x (i.e. an average nucleotide is encoded in the CTC label sequence by a run of x consecutive nucleotide labels). The chance for a nucleotide label not to be the last in its run is then $(x - 1)/x$, thus we put $p_{\text{LINGER}} = x - 1/x$.

4.4 Decoding

We will now discuss two possible ways of using our HMM to determine which signal positions belong to tandem repeats. In the following, *paths* refer to sequences of HMM's states and transitions.

4.4.1 Viterbi algorithm

Viterbi algorithm allows us to find the most likely path (the *Viterbi path*), that could produce the observed signal O [6]. An advantage of the Viterbi algorithm is that it produces a single, coherent path, that carries complete information about individual repeats: where their left sides and right sides start and end, and how they correspond to each other. Therefore, dropping repeats with unit count lower than two (non-tandem repeats) from the output is trivial.

A disadvantage of the Viterbi algorithm is its sensitivity to HMM's level of detail. If we refine a HMM by splitting one of its states σ into multiple, more detailed states $\sigma_1, \ldots, \sigma_k$, any path that originally passed through σ will also get split into multiple, more detailed paths of lower probability. However, probability of paths that avoid σ remains unchanged. Thus, refining a part of a HMM can affect the result of Viterbi algorithm.

In our case, repeat-generating states are much more detailed then the background state. Whenever a path classifies a position in the signal to belong to the right side of a repeat, it specifies whether this position contained a blank, or a nucleotide CTC label. In case of the full model, the path even contains the exact nucleotide. Repeat-predicting paths also have to specify the exact period of the repeat. In contrast, a path that classifies a position to belong to background contains no such information: all options are concentrated in the single background state β. As a result, we fail to detect many repeats, as paths that predict less repeats tend to be favored.

To partially remedy this, we refine our background state to match the level of detail of repeat-generating states. In case of the simple model, we split the background state β into $\beta_\emptyset$ and β_N. State $\beta_\emptyset$ generates blanks, while β_N generates nucleotides. Any transition that originally led to β, will now lead to $\beta_\emptyset$ with probability p_BLANK and to β_N with probability $1 - p_\mathrm{BLANK}$. In case of the full model, we split the background state into five states, one for each CTC label.

4.4.2 Forward-backward algorithm

Using the forward-backward algorithm, we can compute the probability that our HMM was in state σ during generation of i-th CTC label, for any position i in the signal and any state σ. By summing these probabilities for all repeat-generating states, we can compute the probability that position i belongs to the right side of a repeat.

Figure 4.7: An example of a sequence, where the highlighted nucleotide belongs to left sides of two distinct repeats.

Let $X_{i,j}$ be the event that position i is a part of the left side of a repeat and its corresponding position in the right side is $i+j$. If we modify the forward-backward algorithm to also compute probabilities of transitions (not only states), we can compute probability of $X_{i,j}$ for any i, j. The probability that position i belongs to the left side of any repeat is the probability that any of $X_{i,l}, X_{i,l+1}, \ldots, X_{i,h}$ occurs. These events are not exclusive (see Figure 4.7), so it is not sufficient to sum their probabilities. Instead, we estimate the probability of i-th position belonging to a left side as if events $X_{i,l}, X_{i,l+1}, \ldots, X_{i,h}$ were independent, i. e. we compute

$$1 - \prod_{j=l}^{h} \left(1 - \mathrm{P}\left(X_{i,j}\right)\right).$$

To estimate the probability that any given postion i belongs to a repeat, we need to consider the possibilities that it belongs to the left side of a repeat, or to the right side of a repeat. Once again, we combine probabilities of these two events as if they were independent.

An advantage of this approach is that it is not sensitive to HMM's level of detail. Another advantage is that it produces soft predictions. For each position, we get a real number from the interval [0, 1], expressing our confidence that this position belongs to a repeat. By choosing a probability threshold that separates repeats from non-repeats, we can control the ratio of the two types of error (false positives and false negatives) that our detector makes.

A disadvantage of using the forward-backward algorithm is that we may also detect non-tandem repeats (i.e. repeats with unit count below two). This can be partially mitigated by choosing a sufficiently high value of the probability decay parameter d.

From now on, we will refer to this approach as *forward-backward decoding*.

Chapter 5

Evaluation

In this chapter we experimentally evaluate the performance of our approaches from Chapters 3 and 4 and compare them to a baseline approach of basecalling the signal and detecting repeats in the basecalled sequence.

5.1 Experiment design

For our experiments, we use datasets that consist of two parts:

- A known nucleotide sequence, called the *reference*, that contains tandem repeats. The reference has been sequenced by an accurate method, and thus contains no (or very few) sequencing errors.
- A set R of MinION reads, that were produced by sequencing fragments of the reference. Here, by read we mean the recorded signal, not the basecalled nucleotide sequence.

A tandem repeat detection algorithm takes the signal from a read as its input, and labels each position in the signal either as `REPEAT`, or as `NONREPEAT`.

To assess the performance of a repeat detection algorithm, we need to know the correct labelling of the signal for each read.

5.1.1 The gold standard

For each read from R, we determine the part of the reference that was sequenced when this read was produced. We align the read's signal to this part of the reference, i.e. for each nucleotide in the aligned part of the reference, we determine the corresponding part of the signal. This signal-to-reference alignment is done by a tool named *Nadavca* [1]. Usually, not the whole signal is successfully aligned, some parts near the signal's ends remain unaligned. For each point in the aligned part of the signal, we now know the reference nucleotide that was in the nanopore when this signal point was recorded.

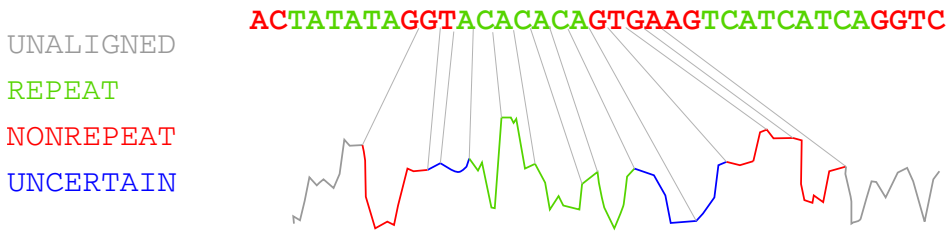

Figure 5.1: Labels used in the gold standard.

Using Tantan, we detect tandem repeats in the reference. Then, we label each position in the signal by one of four labels:

- If the signal position has not been aligned to the reference, we label it as `UNALIGNED`.
- If the signal position is aligned to a reference nucleotide that belongs to a repeat, and the two neighboring nucleotides also belong to a repeat, we label this signal position as `REPEAT`.
- If the signal position is aligned to a reference nucleotide that does not belong to a repeat and neither do the two neighboring nucleotides, we label the signal position as `NONREPEAT`.
- If the signal position corresponds to a reference nucleotide next to an edge of a repeat (either a non-repeat nucleotide next to a repeat, or a repeat nucleotide at one of repeat's ends), we label the signal position as `UNCERTAIN`.

We will refer to this signal labelling as the *gold standard*. Figure 5.1 illustrates how the gold standard is constructed.

The reason for introducing the `UNCERTAIN` label is that there is no good exact definition of when a nucleotide leaves the nanopore and the next nucleotide enters it. Thus, tools may slightly differ in their oppinions about where in the signal a nucleotide ends and another one starts.

5.1.2 Performance assessment

To evaluate the performance of a repeat detection algorithm, we compare its output to the gold standard. We ignore the signal positions that are labelled as `UNALIGNED` or `UNCERTAIN` in the gold standard. Each of the remaining positions falls into one of the following four classes.

Let #(TP), #(FP), #(FN) and #(TN) denote then number of true positive, false positive, false negative and true negative positions, respectively. For every tested repeat detection algorithm, we will compute the following two metrics:

- **True positive rate (TPR):** the fraction of positions labelled as `REPEAT` in the gold

Class	Abbr.	Detection algorithm	Gold standard
True positive	TP	`REPEAT`	`REPEAT`
False positive	FP	`REPEAT`	`NONREPEAT`
False negative	FN	`NONREPEAT`	`REPEAT`
True negative	TN	`NONREPEAT`	`NONREPEAT`

standard, that were also labelled as `REPEAT` by the algorithm, i.e.

$$\mathrm{TPR} = \frac{\#(\mathrm{TP})}{\#(\mathrm{TP}) + \#(\mathrm{FN})}.$$

- **False positive rate (FPR):** the fraction of positions labelled as `NONREPEAT` in the gold standard, that were labelled as `REPEAT` by the algorithm, i.e.

$$\mathrm{FPR} = \frac{\#(\mathrm{FP})}{\#(\mathrm{FP}) + \#(\mathrm{TN})}.$$

We can also define the false negative rate (FNR) to be $1 - \mathrm{TPR}$ and the true negative rate (TNR) to be $1 - \mathrm{FPR}$. An ideal detection algorithm would have TPR of one and FPR of zero.

The ROC curve

In case of using the forward-backward algorithm with our HMM based approach, the output of the detection algorithm is not a discrete labelling of the signal, but a score from the interval $[0, 1]$ for each signal position. We can discretize this output by choosing a threshold t and then labelling all positions with score below t as `NONREPEAT` and all positions with score above t as `REPEAT`. By carefully choosing the value of t, we can trade off between the two types of error. Lowering the threshold t increases the TPR, but it also increases the FPR. Raising the threshold t does the opposite.

Therefore, to assess the performance of the HMM approach with forward-backward algorithm, we compute TPR and FPR for all possible thresholds t. The resulting pairs (FPR, TPR) form a curve in the $[0, 1]^2$ square. This curve is called the *receiver operating characteristic* (ROC) curve [12].

A particularly interesting point on the ROC curve is the one where FPR and FNR are equal. We will call the value of FPR (and FNR) at this point the *equal error rate (EER)*.

5.2 Datasets

We tested our detection algorithms on two datasets, named Saping and Jamang. Their characteristics are summarized in Table 5.1.

Both *repeats ratio* and *average repeat period* are computed from reference positions that align to at least one read. The *repeats ratio* is the fraction of positions that belong to a repeat

Name	Number of reads	Repeats ratio	Average repeat period
Saping	100	7.61%	5.84
Jamang	100	18.07%	63.72

Table 5.1: The datasets used in our experiments.

Model	Peak filtering	Decoding	TPR	FPR	EER
simple	No	forward-backward	-	-	17.39%
simple	No	Viterbi	42.97%	2.26%	-
simple	Yes	forward-backward	-	-	14.89%
simple	Yes	Viterbi	51.46%	1.40%	-
full	No	forward-backward	-	-	14.69%
full	No	Viterbi	22.57%	0.16%	-

Table 5.2: Different strategies for dealing with label lingering.

(according to Tantan). The *average repeat period* is a weighted arithmetic mean of repeat periods, where each repeat is weighted by the number of its positions (that align to at least one read).

Note that the Jamang dataset contains repeats with much longer periods. Therefore, it requires a higher value of h when processed by any of our HMMs, and wider band when processed by our DTW approach.

For parameter estimation in Section 4.3, the Saping dataset was used.

5.3 Results

5.3.1 Strategies for label lingering

In the first experiment, we tested three different strategies for dealing with the fact that a single nucleotide can be encoded by multiple subsequent CTC labels. The first strategy is to use the full HMM. The second strategy is to use the simple HMM on CTC predictions preprocessed by peak filter. The third strategy is to completely ignore the problem and just use the simple HMM.

All three strategies were tested on nine reads from the Saping dataset, with $l = 3$ and $h = 300$. The reason for the small size of the testing sample is the low speed of the full HMM. All strategies were used with both Viterbi and forward-backward decoding. Results are shown in Figure 5.2 and in Table 5.2.

We can see that strategies that account for the problem of lingering perform better then the one that does not. However, the strategy of using the peak filter seems to perform about as well as using the full model. As the simple HMM with peak filter preprocessing is significanlty

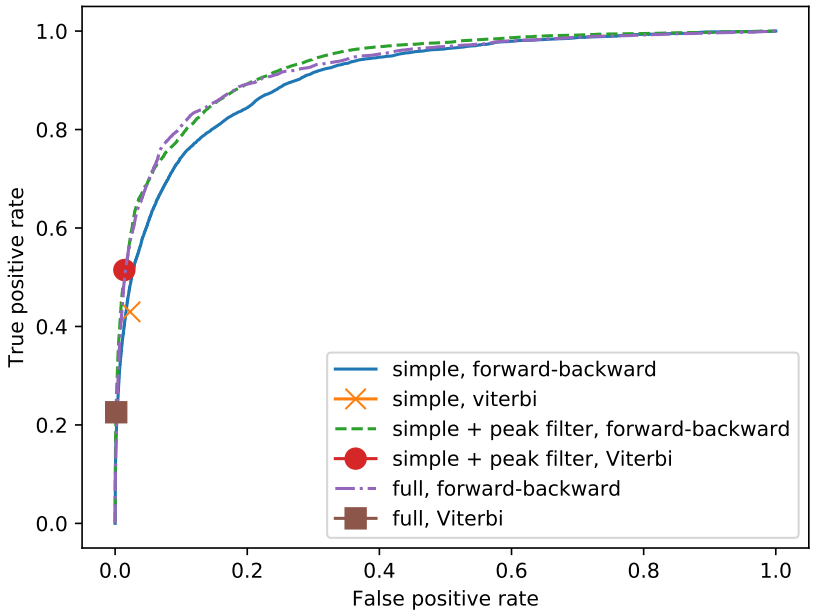

Figure 5.2: Different strategies for dealing with label lingering.

Imperfections modelling	Decoding	TPR	FPR	EER
Yes	forward-backward	-	-	18.27%
No	forward-backward	-	-	17.89%
Yes	Viterbi	43.95%	1.90%	-
No	Viterbi	42.68%	1.59%	-

Table 5.3: The effect of modelling repeat imperfections.

faster than the full HMM, we will use this strategy in the experiments to follow.

5.3.2 Effect of modelling repeat imperfections

In this experiment, we look at how modelling of imperfections in repeats (described in Section 4.2.4) affects the performance of our repeat detector. We used the simple HMM with peak filter preprocessing with $l = 3$ and $h = 300$ on the Saping dataset.

We ran the detection algorithm four times. In the first run, we used standard settings and forward-backward decoding. In the second run, we set the probability of insertions, deletions and mismatches to zero. In the remaining two runs, we did the same, but with Viterbi decoding instead. Results are shown in Figure 5.3 and in Table 5.3.

The effect of modelling imperfections seems to be negligible. Surprisingly enough, the model that ignored imperfections performed marginally better.

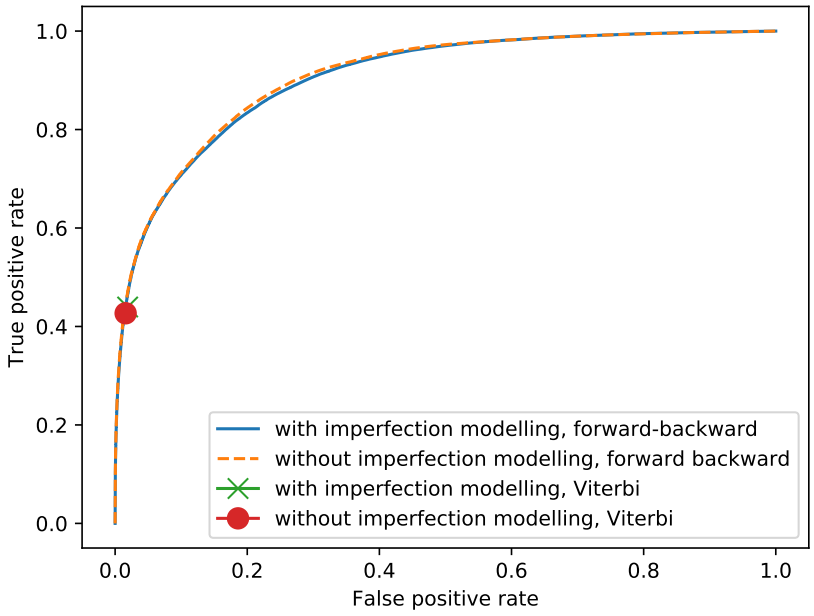

Figure 5.3: The effect of modelling repeat imperfections.

Model	***b***	**Decoding**	**TPR**	**FPR**	**EER**
DTW	0.1	-	42.74%	4.98%	-
DTW	0.2	-	64.36%	14.09%	-
DTW	0.3	-	77.50%	29.37%	-
HMM	-	forward-backward	-	-	18.27%
HMM	-	Viterbi	43.95%	1.90%	-

Table 5.4: Comparison of the DTW and the HMM approach.

5.3.3 Comparison with DTW

In this section we compare the performance of our HMM approach and our DTW approach. We used our DTW approach with the permitted band of 30 to 300 positions from the main diagonal on the Saping dataset. We ran the algorithm three times, with the scoring parameter b set to $0.1, 0.2$ and 0.3, respectively. Results are compared to the simple HMM with peak filter from the previous experiment in Figure 5.4 and in Table 5.4.

In this test, the HMM approach clearly outperformed the DTW approach.

5.3.4 Comparison with the baseline

We now compare our best performing approach (simple HMM with peak filter preprocessing) to the baseline approach of basecalling the signal and finding repeats in the basecalled sequence.

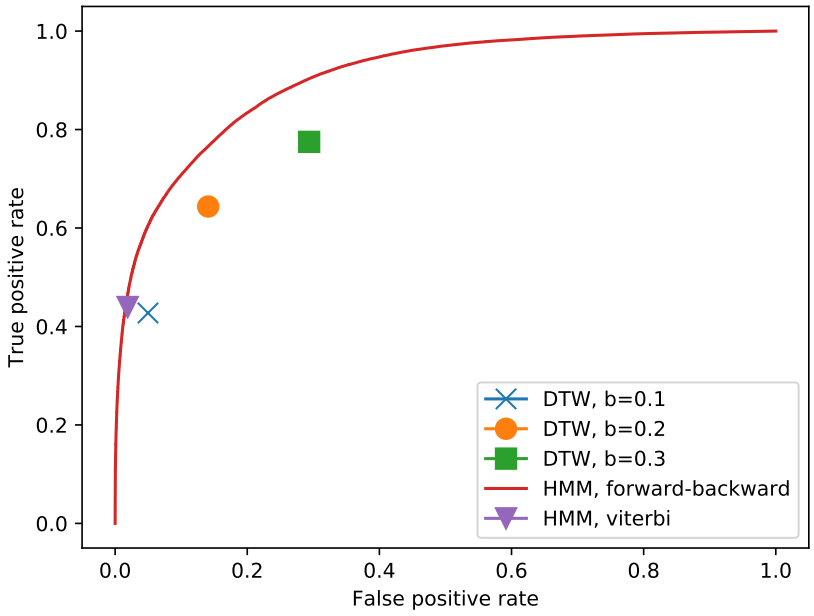

Figure 5.4: Comparison of the DTW and the HMM approach.

The baseline approach

In our implementation of the baseline approach, we basecall the reads with Albacore, a basecaller by Oxford Nanopore Technologies. Besides the output nucleotide sequence, Albacore also provides alignment of the signal to the basecalled sequence. We then detect tandem repeats in the nucleotide sequence from Albacore using Tantan. As a last step, we map the repeats back to the signal, using the alignment provided by Albacore.

Tantan has a configurable parameter that sets the score penalty for mismatches and a parameter that sets the score penalty for indels. The default value of both of these parameters is seven. Besides running Tantan in its default configuration, we also tried running it with decreased penalties, as we expect some imperfections to be introduced into repeats by basecalling errors. We always kept the indel penalty and the mismatch penalty equal.

The experiment

We ran the baseline approach with the penalty parameter p of 2, 3, 4, 5, 6 and 7 on both Saping and Jamang datasets. We also ran simple HMM with peak filter preprocessing, $l = 3$ and $h = 1000$ on the Jamang dataset, with both forward-backward and viterbi decoding. Results of running the simple HMM on Saping were already available from previous experiments. Comparison of these approaches can be found in Figures 5.5 and 5.6 and Tables 5.5 and 5.6. The baseline approach with $p = 2$ predicted all signal positions as non-repeats, thus it is not included.

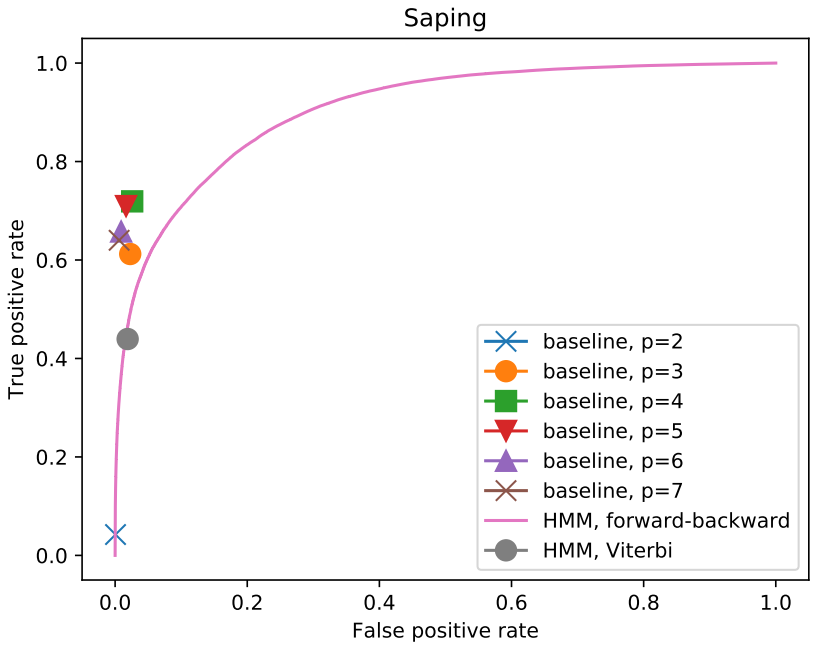

Figure 5.5: Comparison our HMM with the baseline on Saping.

We can see that the Jamang dataset seems to be easier for both our approach and the baseline. On the Saping dataset, our approach is outperformed by the baseline. However, on Jamang our algorithm performs marginally better than the baseline. This could be due to the higher amount of repeats in Jamang, as basecallers tend to have lower accuracy on repetitive sequences.

5.3.5 Speed

In the last experiment, we measured the speed of our approaches on the Saping dataset. Five approaches were evaluated: both HMMs were tested with both types of decoding and the

Model	***p***	**Decoding**	**TPR**	**FPR**	**EER**
baseline	2	-	4.24%	0.06%	-
baseline	3	-	61.22%	2.28%	-
baseline	4	-	71.91%	2.58%	-
baseline	5	-	70.95%	1.66%	-
baseline	6	-	65.78%	0.91%	-
baseline	7	-	64.00%	0.61%	-
HMM	-	forward-backward	-	-	18.27%
HMM	-	Viterbi	43.95%	1.90%	-

Table 5.5: Comparison our HMM with the baseline on Saping.

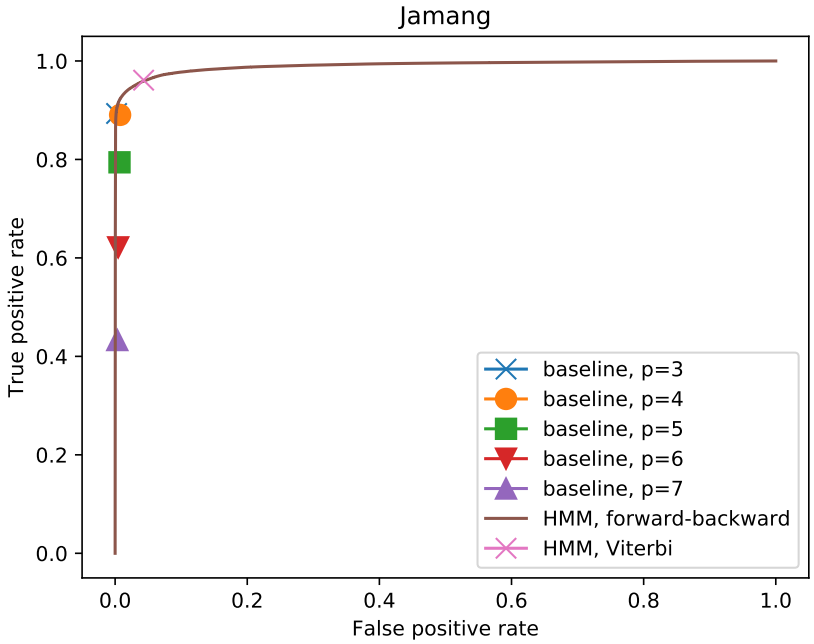

Figure 5.6: Comparison our HMM with the baseline on Jamang.

Model	***p***	**Decoding**	**TPR**	**FPR**	**EER**
baseline	3	-	89.33%	0.22%	-
baseline	4	-	89.05%	0.75%	-
baseline	5	-	79.42%	0.66%	-
baseline	6	-	62.14%	0.46%	-
baseline	7	-	43.32%	0.33%	-
HMM	-	forward-backward	-	-	4.17%
HMM	-	Viterbi	96.10%	4.31%	-

Table 5.6: Comparison our HMM with the baseline on Jamang.

Approach	***l***	***h***	**signal positions per second**
simple model, forward-backward	3	300	479
simple model, Viterbi	3	300	1587
full model, forward-bakward	3	300	7.61
full model, Viterbi	3	300	43.0
DTW	30	300	8321

Table 5.7: Speed comparison of our models.

DTW approach was tested as well. In case of the full HMM, we measured the speed on a single read. Other three approaches were tested on a sample of ten reads.

Time needed to run Chiron in order to produce CTC predictions for the HMM approaches was not included in the measurement. The experiment was done on a machine with Intel(R) Core(TM) i7-4800MQ CPU @ 2.70GHz processor. Results are summarized in Table 5.7.

Conclusion

The goal of this thesis was to devise a method for detection of tandem repeats in a single MinION read, that would be more accurate than detection of repeats in the basecalled sequence.

In our first approach, we try to avoid the accuracy problems associated with basecalling by looking for repetitive sequences directly in the the raw signal produced by the sequencer. To find repetitive sequences in the signal, we align it with itself. With this approach, we never achieved the accuracy of the baseline approach of detecting repeats in the basecalled sequence. A possible reason for this is the fact that small, local changes in the DNA sequence cause less local effects in the signal. This makes imperfect repeats harder to detect on the signal level.

Our second approach uses a basecaller, but it does not undergo the whole process of basecalling the signal into a particular nucleotide sequence. Instead, it stops at the CTC output layer and tries to detect repeats in CTC predictions. By doing that, we try to avoid errors made by the basecaller in situations, when it sees the signal as ambiguous, but it is forced to commit to a single explanation. To detect repeats in CTC predictions, we devised a HMM-based approach inspired by Tantan.

In the experimental testing we conducted, our HMM did not outperform the baseline. On a dataset with repeats of short periods, the baseline approach was better. However, on a dataset with repeats of longer periods, our approach is about as accurate as the baseline. This gives us hope that our HMM-based approach is a step in the right direction and that with further development, it could actually fulfill the original goal of this thesis.

It is also possible, that there already are situations where our HMM would be useful. Further testing is needed to find out if there is an application for our model.

Bibliography

[1] E. Batmendijn and V. Boža. Nadavca. `https://github.com/fmfi-compbio/nadavca`.

[2] G. Benson. Tandem repeats finder: a program to analyze DNA sequences. *Nucleic Acids Research*, 27(2):573–580, 01 1999.

[3] G. Benson and M. S. Waterman. A method for fast database search for all k -nucleotide repeats . *Nucleic Acids Research*, 22(22):4828–4836, 11 1994.

[4] D. Branton et al. The potential and challenges of nanopore sequencing. *Nature biotechnology*, 26(10):1146–1153, Oct 2008. 18846088[pmid].

[5] O. Delgrange and E. Rivals. Star: an algorithm to search for tandem approximate repeats. *Bioinformatics (Oxford, England)*, 20:2812–20, 12 2004.

[6] G. D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[7] M. C. Frith. A new repeat-masking method enables specific detection of homologous sequences. *Nucleic acids research*, 39(4):e23–e23, Mar 2011. 21109538[pmid].

[8] A. Graves et al. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML 2006*, pages 369–376, 2006.

[9] R. Kolpakov et al. mreps: Efficient and flexible detection of tandem repeats in dna. *Nucleic acids research*, 31(13):3672–3678, Jul 2003. 12824391[pmid].

[10] P. Landry et al. Deriving evolutionary relationships among populations using microsatellites and (deltamu)(2): all loci are equal, but some are more equal than others. *Genetics*, 161(3):1339–1347, Jul 2002. 12136035[pmid].

[11] A. Lászik et al. Automated fluorescent detection of a 10 loci multiplex for paternity testing. *Acta Biologica Hungarica*, 51(1):99–105, Mar 2000.

[12] C. E. Metz. Basic principles of roc analysis. In *Seminars in Nuclear Medicine*, volume 8, pages 283–298. Elsevier, 1978.

[13] S. Molčan. Identifikácia a analýza repetitívnych sekvencií v nanopórových dátach. Bachelor thesis, Comenius University in Bratislava, 2019.

[14] H. T. Orr and H. Y. Zoghbi. Trinucleotide repeat disorders. *Annual Review of Neuroscience*, 30(1):575–621, 2007. PMID: 17417937.

[15] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[16] F. Sanger et al. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977.

[17] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, Mar 1981.

[18] H. Teng et al. Chiron: translating nanopore raw signal directly into nucleotide sequence using deep learning. *GigaScience*, 7(5), 04 2018. giy037.

[19] O. K. Tørresen et al. Tandem repeats lead to sequence assembly errors and impose multi-level challenges for genome and protein databases. *Nucleic acids research*, 47(21):10994–11006, Dec 2019. 31584084[pmid].

[20] A. A. G. van Tilborg et al. Selection of microsatellite markers for bladder cancer diagnosis without the need for corresponding blood. *PLOS ONE*, 7(8):1–7, 08 2012.

[21] R. R. Wick et al. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome Biology*, 20(1):129, Jun 2019.

Appendix A: implementation

Implementation of our methods is attached.

The implementation of our DTW-based approach is in directory `signal_dtw`. The implementation of the HMM-based approach is in directory `hornet`.