CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

DNA Sequence Classification Using Variable Length Markov Models

Master's thesis in Computer science and engineering

Sebastian Norlin

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

DNA Sequence Classification Using Variable Length Markov Models

Sebastian Norlin

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2019

DNA Sequence Classification Using Variable Length Markov Models

Sebastian Norlin

Supervisor:	Alexander Schliep Computer Science and Engineering
Advisors:	Joel Gustafsson & Peter Norberg Institute of Biomedicine, Gothenburg University
Examiner:	Devdatt Dubhashi Computer Science and Engineering

Master's Thesis 2019
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Gothenburg, Sweden 2019

DNA Sequence Classification Using Variable Length Markov Models

Sebastian Norlin
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Pathogens such as viruses and bacteria are a major health concern today. To effectively treat these it is important to identify known pathogens and potential new ones from DNA samples. Modern methods are however not good enough at classifying rare, previously undocumented pathogens.
This thesis explores nearest neighbor classification using variable length Markov chains (VLMC) as a possible solution. A vantage point tree is used to store the database of VLMC being queried against. This gives promising results when classifying VLMC from complete genomes or chromosomes. Multiple techniques, both greedy approximations and new lower bounds are explored. This results in order of magnitude faster classification than previous research. However the technique ultimately fails at classifying shorter DNA sequences of lengths typically found when sequencing DNA. Multiple reasons for this are given with a possible way forward if further research is deemed relevant.

Keywords: Computer science, Bioinformatics, Master's thesis, vantage point tree, metric space, Variable length Markov chains, Markov Models, DNA, Classification

Acknowledgements

I would like to thank Joel for the great feedback and help all throughout the thesis. Especially for helping me understand VLMCs and for letting me use his code. I would also like to thank my supervisor, Alexander, for sharing some of his great knowledge of the area and for guiding me when all felt lost. Peter also deserves great thanks for explaining the relevant biology and the bigger picture. Lastly I would like to thank Devdatt, my examiner.

Sebastian Norlin, Gothenburg, September 2019

Contents

1
Introduction

Various pathogens such as viruses, bacteria and fungi are a major health problem for humans today. This is especially true for immunocompromised patients where otherwise fairly harmless pathogens can cause major health problems [16]. To effectively treat these pathogens an exact and quick identification of the pathogen is needed. However existing methods fail to classify many rare pathogens because they require observed pathogens to search for [16, 23]. Because of this approximately 30% of all samples from immunocompromised patients at Sahlgrenska University hospital (SU) remain unclassified [16]. This makes treating these patients very hard because correct antibiotics, antivirals or fungicides cannot be administered [16]. A fast and accurate method that can also classify rare pathogens is therefore needed.
One promising method of classifying DNA sequences proposed by Dalevi et al. relies on genomic signatures as variable length markov chains(VLMC, section 2.2.2) [7]. To classify DNA sequences using this method the sequence must first be turned into a VLMC. This can be done efficiently by the algorithm created by Schultz et al. [20]. The genomic signature must then be compared to a large database of genomic signatures of known pathogens to find matches. Dalevi et al. perform an exhaustive search minimizing KL-divergence [7] (section 2.3.2.3) to find matches but this is too slow even for moderate collections of pathogens. For the classification algorithm to be feasible, an efficient data structure is needed. This data structure should allow for rapidly finding the nearest neighbours of any genomic signature and the goal of this thesis is to create such a data structure.

1.1 Delimitations

The thesis will focus on adapting existing data structures and methods to efficiently store VLMC rather than creating new ones. Another delimitation is that only VLMC are considered. There are other ways to represent genomic signatures but they are not considered in this thesis. String based or deep learning based methods will also not be considered.
The thesis will use existing datasets and is not focused on expanding these.

2
Background

2.1 Biological background

This section will give a brief introduction to the relevant biological background.

2.1.1 DNA

Deoxyribonucleic acid (DNA) is a long sequence of nucleobases (guanine [G], cytosine [C], adenine [A] or thymine [T]) used to store the genetic information needed for any cell to function, grow or reproduce. It is found in every organism except for some viruses that use RNA instead. DNA is generally stored in two strands running anti-parallel in a double helix, see figure 2.1. These two strands are connected such that A binds to T and G binds to C. Because of the pair-wise bindings the length of a DNA sequence is often described by the number of base pairs (bp) it contains. A base pair is either a pair of G and C or A and T. If a length of a DNA sequence is given without a unit, it can be assumed to be in bp.
The DNA is extracted from an organism through a process called sequencing. Current sequencing techniques split the DNA and extract many short sequences between 100-1000bp long, called reads. These reads can then be combined, by finding overlapping regions, to form a complete genome or chromosome.
DNA contains both coding and non-coding parts. The coding parts are used to code for amino acids that are used to construct proteins. The non coding parts are less understood, yet shown to be important for example for regulatory purposes or in embryotic development [14]. The non-coding parts also contain viral DNA from retroviruses and, especially in bacteria, DNA from horizontal gene transfers. These foreign DNA fragments can sometimes be problematic during classification.

2.1.2 Taxonomy

Taxonomy is the science of naming and arranging organisms into specific groups based on evolutionary relationships. These groups form a hierarchical system where the most commonly used ranks, or levels, in this system are: Domain, Kingdom, Phylum, class, order, family, genus and species. For example the swedish elk (or moose) is in the domain Eukaryote (such as animals and plats), phylum Chordata (mostly vertebrates), class Mammalia (mammals), order Artiodactyla (even-toed ungulates), family Cervidae (Deer), genus Alces (only species in genus) and species Alces.

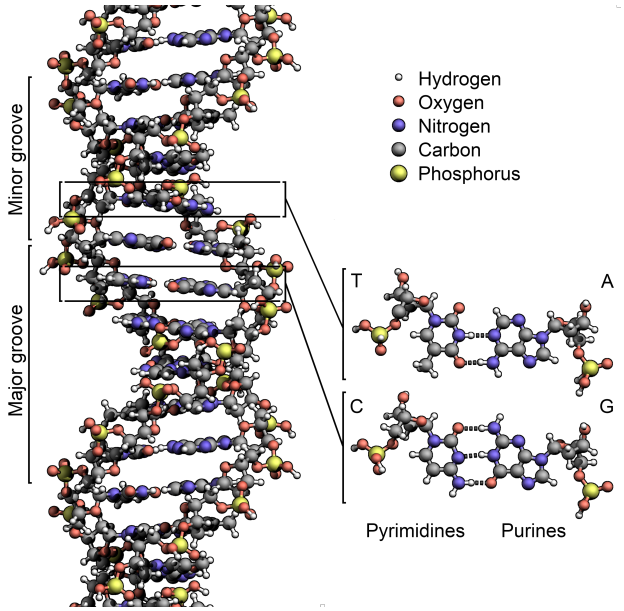

Figure 2.1: Image showing the double helix structure of the DNA sequence. Note how A binds to T and G binds to C. By Zephyris - Own work, CC BY-SA 3.0, unmodified, https://commons.wikimedia.org/w/index.php?curid=15027555

Organism is a term often used in taxonomy and biology to refer to anything exhibiting signs of life. For the purpose of this thesis however, the term organism will include viruses and other forms of "life" that may lack metabolism or the ability to replicate or evolve without exterior mechanisms. Instead anything that contains DNA or RNA will be referred to as an organism. As such organisms are anything that can be taxonomically classified using its DNA or RNA.

2.1.3 Genomic signatures

A genomic signature is a representation of an organism's genome by finding frequently repeating patterns. This was first proposed by Kariin et al. by recording the frequency of length 2 substrings called dinucleotides or 2-mers [13]. This has evolved over the years as more data and more powerful computers become available. Today the practise of recording the frequency of all substrings of length k is referred to as k-mer counting and the k long substrings are called k-mers.

The goal of calculating genomic signatures is to speed up algorithms, such as classification, that may be performed on the genome. This can be performed with methods such as k-mer counting, k-mer lookup tables [23] or statistical models such as Markov models [7]. The simplest form of genomic signature is calculating the GC content. The GC content is the proportion of all neucleobases being either guanine or cytosine.

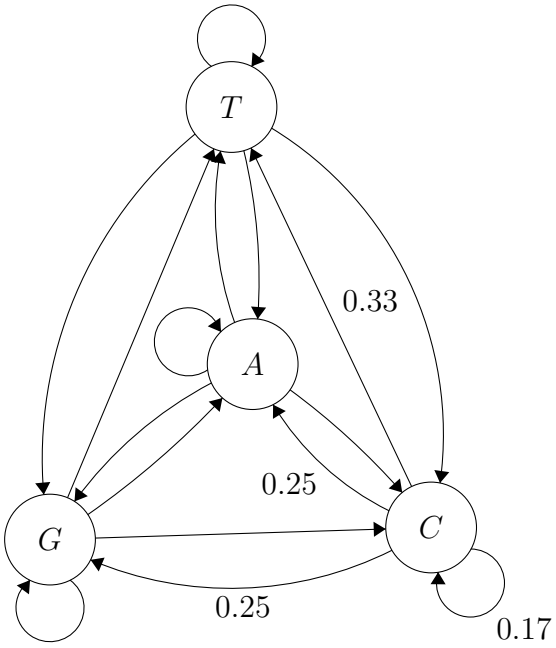

Figure 2.2: A markov chain with states A,T,C, and G. The arrows are the transitions and the numbers the transition probabilities. Only the transition probabilities for when the markov chain is in state C is written out for clarity. If the markov chain is in state C then it has transition probabilities $\mathrm{p}(T|C) = 33\%, \mathrm{p}(A|C) = 25\%, \mathrm{p}(G|C) = 25\%, \mathrm{p}(C|C) = 17\%$.

2.2 Markov models

2.2.1 Fixed order Markov models

A Markov model is a statistical model with a set of states Σ and knowledge of some finite part of the history. Given this history the Markov model returns a distribution over the probabilities of transitioning to the different states. A Markov model with history of just the current state, also called a Markov chain, is given in Figure 2.2. That Markov model has the states $\Sigma = \{A, T, G, C\}$ corresponding to the four nucleobases of DNA.

The length of the history is called the order of that Markov model. If instead an order two Markov model was used to model the DNA sequence it would have $16 \cdot 4$ transition probabilities $\mathrm{p}(A|AA), \mathrm{p}(T|AA), \mathrm{p}(G|AA), \mathrm{p}(C|AA)$ and so on for each possible history of length 2. If T is generated then it is said to be emitted by the model. This probability is either called the transition, or the emission probability. Generating a T in state AA will also transition the model to state AT.

One way to store Markov models is by storing their emission matrix. This is a $|\Sigma| \cdot N$ matrix for a Markov model of N states that stores the emission probabilities of each character in Σ for every state.

Sometimes the shorthand $\mathrm{p}(\Sigma|AA)$ will be used to denote the vector of emission probabilities from state (AA). In general an order α Markov model will have $|\Sigma|^{\alpha}$ states and $|\Sigma|^{\alpha+1}$ transition (or emission) probabilities.

2.2.2 Variable length Markov chains

A variable length Markov chain (VLMC) [4] is a Markov model that allows the history, called the context, to vary between different states. They are sometimes also referred to as variable order Markov models, because the order of the model is different for different states. Consider for example a VLMC with $\Sigma = \{A, T, G, C\}$. A VLMC could store transition probabilities p($\Sigma|ATTG$), p($\Sigma|ATTGG$) and p($\Sigma|GCTG$) but no other transition probabilities for states with context of length 4 or 5. The reason for not storing the other states could be that they do not contain enough new information. An example of a small VLMC with $\Sigma = \{A, B\}$ is shown in figure 2.3. The benefit of VLMC becomes apparent when modeling an organism that has some long, important sequences. Consider an organism with a sequence of length 15 that occurs fairly frequently and this sequence might be needed to differentiate between two different species. It would then be important that the whole sequence is captured by the model. However to capture this information in a fixed order Markov model 4^{15} states and a total of $4^{16} = 2^{32}$ transition probabilities have to be stored. A VLMC on the other hand can store this information in fewer than 1000 states by pruning noise or less important states. Note however that the VLMC always stores an emission probability for every character in Σ for every state. As such a VLMC has N states and $N * |\Sigma|$ transition emission probabilities.
The main benefit of VLMC is therefore a more compact, but comparatively expressive, model compared to fixed order Markov models. VLMC have been shown to extract relevant information from whole genomes and can be used to perform taxonomic classification [7].
When constructing VLMCs the algorithm by Schulz et al. [20] can be used. Describing this algorithm is out of scope for this thesis but it takes three parameters. These are: mincount, maximum length and a pruning value. The mincount specifies the minimum number of times a specific sequence has to be found in the original string to be considered. Maximum length is the maximum length the context, or history, a state may have. The pruning value is either a floating point value used to prune the number of states or a fixed maximum number of states.
As an example, we can construct a VLMC from the string ATGATGACC with mincount 2, maximum length 2 and no pruning value. The sub strings {A, T, G, C, AT, TG, GA, ATG} all occur at least twice. However, because ATG is longer than the maximum length it will be removed. A root node is also added resulting in a VLMC with 8 states and the emission probabilities of state A is: p($A|A$) = 0, p($T|A$) = 0.66, p($G|A$) = 0, p($C|A$) = 0.33.

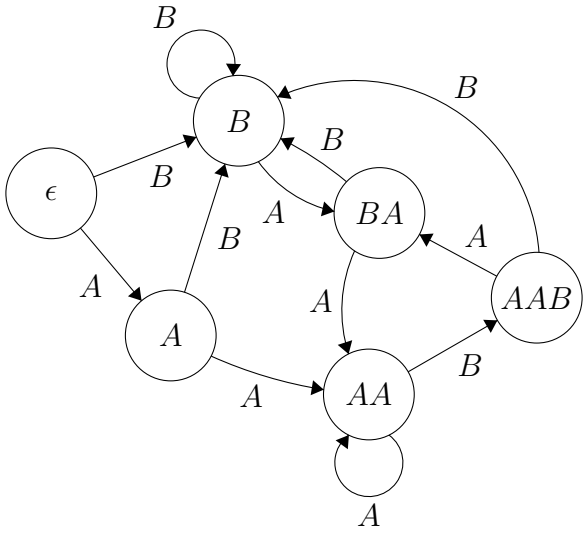

Figure 2.3: Example VLMC. Compared to a fixed order Markov model of order three it has five states instead of eight. The fixed order model would not have states {A,B,BA,AA} and instead all possible order three states. The benefit of VLMCs is mostly apparent for large models however.

2.3 Distances

2.3.1 Metric distances

Most functions that one thinks of as distance functions are metric distances. Because of this there exists many data structures and algorithms that depend on metric distances. This is problematic when working with VLMC as there are few metric distances for VLMCs. For a distance function d to be metric it must fulfill the following four properties:

$$d(x, y) \geq 0 \qquad \text{Non negative} \tag{2.1}$$

$$d(x, y) = 0 \Rightarrow x = y \qquad \text{Identity of indiscernibles} \tag{2.2}$$

$$d(x, y) = d(y, x) \qquad \text{Symmetry} \tag{2.3}$$

$$d(x, y) \leq d(x, z) + d(z, y) \qquad \text{Triangle inequality} \tag{2.4}$$

Equation (2.2) means that if the distance between two points is 0 then these two points must be equal. Equation (2.4) means that the distance between two points x, y is equal to, or less than the distance between x, y via some third point z.

2.3.2 Distance functions on VLMCs

To be able to perform a nearest neighbor search a distance function is required. There is no single optimal distance function for VLMCs instead they have various pros and cons. In general the more accurate distance functions are slower and the fast distance functions are less accurate. The Kullback–Leibler divergence will give the true difference between two VLMCs and can be seen as a ground truth

in distance between two VLMC. The following section describe the best distance functions found by [10].

2.3.2.1 GC

The simplest, and fastest distance function one can use for DNA sequences, or genomic signatures, is the so called GC distance. This distance calculates the difference in GC content between two sequences. The GC content can easily be stored as a constant and therefore retrieved with a single look-up.
The trade off is that GC distance discards a lot of the information in the genome. This means two very taxonomically different genomes can have low GC distance. GC distance is therefore not enough to make correct classifications [7].
The GC distance obeys non-negativity, symmetry and triangle inequality but not Identity of indiscernibles.

2.3.2.2 Frobenius norm

Frobenius norm is a matrix norm on the emission matrix of both models. This was applied to HMM [6] and later adopted to fit VLMC [10]. The Frobenius norm requires that both models have the same set of states. However, due to the nature of VLMCs this is not always the case. Instead either the union or the intersection of the two state spaces has to be used. This new state space is then used to determine what part of the emission matrices to use. Using the intersection is both faster and more accurate compared to union and will therefore be used in this thesis [10]. The Frobenius norm of the intersection of the emission matrices will be denoted Frobenius intersection or FN.
The Frobenius intersection is given by equation (2.5) where x, y are VLMCs and $S = E_x \cap E_y$.

$$d_{fn}(x, y) = \sqrt{\frac{1}{|S|} \sum_{s \in S} \sum_{a \in \Sigma} (P_x(a|s) - P_y(a|s))^2} \tag{2.5}$$

The Frobenius intersection obeys non-negativity and symmetry but not Identity of indiscernibles nor triangle inequality.

2.3.2.3 Kullback–Leibler divergence

The Kullback-Leibler (KL) divergence is a measurement of how different two probability distributions are. It is defined for any two probability distributions over the same probability space. Equation (2.6) gives the definition of KL-divergence on two probability distributions P and Q defined on some discrete probability space X. The continuous version is similar but it integrates over all of X.

$$D_{KL}(P, Q) = -\sum_{x \in X} P(x) \log\left(\frac{Q(x)}{P(x)}\right) \tag{2.6}$$

Note that the KL-divergence is only defined if $P(x) = 0 \rightarrow Q(x) = 0$.
When comparing two VLMCs it is possible that a state exists in one model but not the other. Because of this it is not possible to calculate the KL divergence using just the transition probabilities of both models. Instead one has to create a probability space where both VLMCs are defined. This can be done in the say way that [12] defined KL divergence for hidden Markov models. Let X be a set of one or more generated sequences of length s. Equation (2.6) is then defined on this set X and will approach the true KL divergence between the VLMCs as the sequence length s approaches infinity.
KL divergence obeys non-negativity, symmetry if $D_{KL-sym} = \frac{1}{2}(D_{KL}(P,Q)+D_{KL}(Q,P))$, Identity of indiscernibles, but not triangle inequality.

2.4 K nearest neighbor classification

K nearest neighbor (knn) classification is a classical machine learning, or data mining technique that consists of two parts, a database and set of points to classify. The data in the database consists both of some value, or point, and a label, such as a specific virus or bacteria. To classify a point using Knn is a two stage process. First, the point is queried against the database which returns the k closest, or nearest, points from the database. These are referred to as the k nearest neighbors, hence the name. The second stage is to use these k points to classify — determine the label of — the queried point by for example majority vote, see below.
The benefit of knn classification is that it allows for classifying a point without comparing it to the whole database. Techniques for increasing the accuracy or speed of a knn classifier comes in three forms. Either they modify the data set, change the database structure or change the classification step at the end.

2.4.1 Classification algorithm

The simplest of all classification algorithms is to just classify the point as the most frequent label of the k neighbors. This is however prone to errors, especially if the queried point is in a rare class or close to the decision boundary. See figure 2.4. If three neighbors are used the green point is classified as a red triangle. If five neighbors are used it is classified as a blue square.
To avoid this problem weighted nearest neighbor classification can be used. This adds some weights w_i to each neighbor's label l_i for $i = 1..k$. The weight can be as simple as $k-i$ or $\frac{1}{d_i}$ where d_i is the distance to the query from the ith neighbor. The weights are then summed and the query is classified as the most heavily weighted label. Choosing $w_i = 1$ for all neighbors is equivalent to the naive classifier above. More complex weighting schemes can be chosen that are more or less task specific, see for example [8, 5].

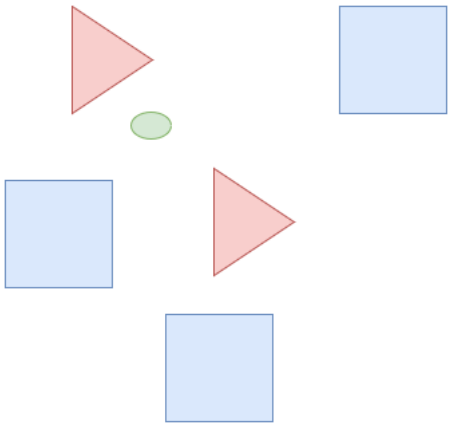

Figure 2.4: Nearest neighbor classification example with the query in green and two different classes represented by the red triangles and blue squares. If one neighbor is used it is classified as a red triangle, same goes for three. However, for five neighbors it is classified as a blue square if unweighted NN classification is use.

2.4.2 Data structures

Trees are commonly used to structure the data before performing a nearest neighbor search. The goal of storing the data in a tree is to find some way of pruning branches. That is, determine that the nearest neighbor cannot be in a specific sub tree and avoid traversing it. Another benefit with trees is that algorithms are often fairly simple.

2.4.2.1 Kd-tree

A kd-tree, or k dimensional tree is a binary tree data structure used to store multi-dimensional data. For each level of the tree only one dimension is used to determine whether a node is greater than, or less than its parent. Because of this a Kd-tree requires that the data has the same, fixed, dimensions and is therefore hard to use for vlmcs.

2.4.2.2 Vantage point tree

A vantage point tree (VP-tree) proposed by Uhlman in [22] is a tree structure for storing metric data. It partitions the data by selecting elements to be used as vantage points. These vantage points are stored in each node and the children are partitioned by putting all sufficiently close points in the left sub tree and the rest in the right sub tree.
A vantage point tree is constructed using algorithm 1 where $d(p, vp)$ is the distance from p to vp. The function *select_vp* selects the vantage point to use from all of the data. It can have a large impact on the nearest neighbor search speed [2].
The algorithm for finding the nearest neighbor in a VP tree given by algorithm 2. It will recursively traverse the tree and takes as input a node n, a query q and the so far nearest neighbor *best*. To understand how this algorithm works an example is given where the point X figure 2.5 is classified. The given line numbers refer to

```
   input  : data: D
   output: root node: n
 1 Function construct(D)
 2     if D ≠ ∅ then
 3         vp = select_vp(D)
 4         n.data = vp
 5         n.cutoff = cutoff
 6         n.left = construct( {p ∈ D |d(p, vp) ≤ cutoff} )
 7         n.right = construct( {p ∈ D |d(p, vp) > cutoff} )
 8         return n
 9     else
10         return ∅
11     end
12 end
```

Algorithm 1: algorithm for constructing a vp tree. Generally the cutoff is chosen to be the median distance from all points in D to *vp*.

line numbers in algorithm 2. First the distance to 1 is calculated and because this is lower than the cutoff for 1 the left tree is explored first (5-6). The distance to 2 is calculated and the best distance updated (2-3). Again it is closer than the cutoff and the left tree will be explored (5-6) and the distance to three is calculated. The best distance is updated because 3 is the closest node so far (2-3). 3 is a leaf and the algorithm cannot progress further down the tree. The closest distance so far (black line) is compared to the distance to the cutoff boundary of 2 (dotted line) (7). The black line is shorter and node 4 can be pruned. The black line is then compared to the dotted line of 1 (7). Again it is shorter and the whole right subtree can be pruned and the nearest neighbor of X is 3.
Note that the pruning done on line 7 and 12 uses triangle inequality. Because of this it will not guarantee optimal results for distance functions that do not obey triangle inequality. Also note that the algorithm visits at least one leaf and that it can never terminate unless it has just visited node with 1, or 0 children. Therefore to classify Y it will visit 1,5,8,6,7,2,4 in that order.

2.4.2.3 Ball-trees

Ball trees work very similar to VP trees in that it is generally a binary tree that partitions space. Ball trees however store all data in spheres as opposed to a VP tree that stores the closer points in a sphere and the other points in a, potentially thick, shell. This has the benefit of ensuring that objects are at most some radius r away from the sphere center.
There are multiple construction algorithms for ball trees. A general goal however is to minimize the total volume of all balls in the tree [17]. The construction algorithm can have a large impact on the final tree however.

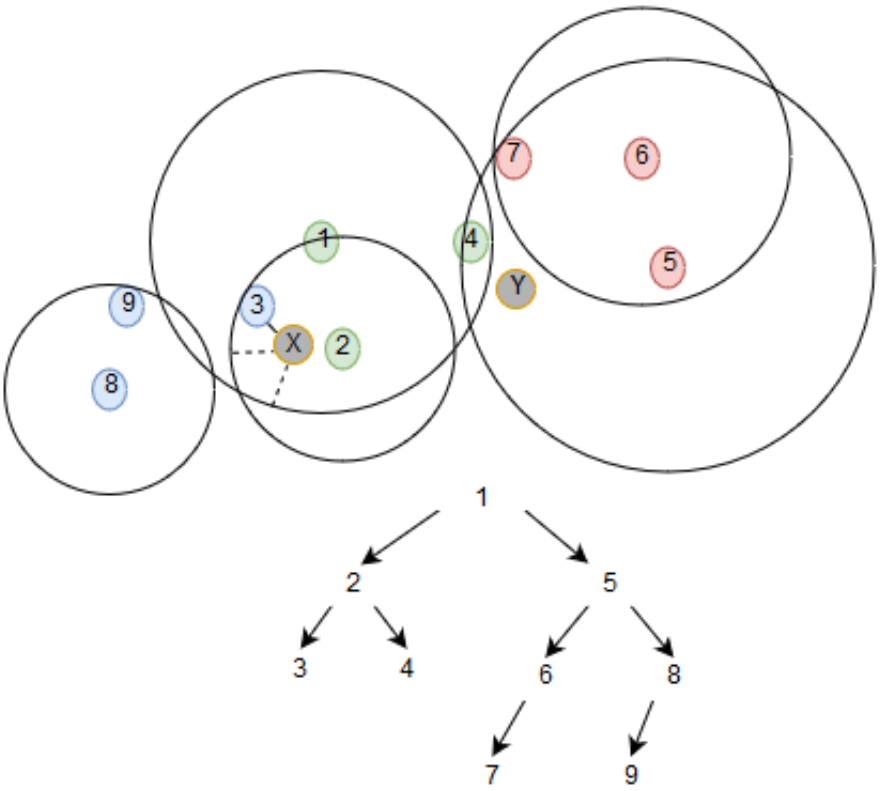

Figure 2.5: Example of a VP tree storing data from three different classes (blue, green red) and two queries (X,Y). A representation of the 2D space is given on top where the circles represent the cutoff distance and underneath a VP tree of this data is shown.

```
   input  : n, q, best
   output: nearest neighbor
 1 Function nn()
 2 |  if d(n.data, q) < d(best,q) then
 3 |  |  best = n.data
 4 |  end
 5 |  if d(n.data,q)<n.cutoff then
 6 |  |  best = nn(n.left,q,best)
 7 |  |  if d(n.data,q)+d(best,q)>n.cutoff then
 8 |  |  |  best = nn(n.right,q,best)
 9 |  |  end
10 |  else
11 |  |  best = nn(n.right,q,best)
12 |  |  if d(n.data,q)-d(best,q)<n.cutoff then
13 |  |  |  best = nn(n.left,q,best)
14 |  |  end
15 |  end
16 |  return best
17 end
```

Algorithm 2: algorithm to find nearest neighbor in vp tree

2.5 Other classification methods

2.5.1 String based methods

Traditionally string based methods are used to classify DNA sequences. Examples of such methods are BLAST[1], kraken[23], Megan[11], MetaPhlAn[21]. MetaPhlAn work by finding pre-determined specific DNA substrings that are known to be important for a specific species or genus. The sequence that is being classified is then searched for these specific substrings. This means that any species or genus not in that pre-determined dataset cannot be classified. The method is however fast and can be used to give an overview of what organisms are present.
Kraken [23] works by constructing a lowest common ancestor database based on k-mers of the DNA sequence. These k-mers can be obtained by using a sliding window. The database is queried for every k-mer in the DNA sequence being classified. The total number of times that k-mer is found is stored in a tree. When all k-mers have been queried into the database this tree is traversed following the highest scoring path until a leaf is reached. The DNA sequence is classified as that leaf's class. See fig 2.6 for an illustration from the Kraken paper.
When analyzing DNA sequences using k-mer based methods it is important to choose the correct size of k-mers. Kraken uses a $k = 31$ as a standard length of their k-mers. This gave them a 70GB large database [23]. They also released a version with rare k-mers cut away. This is significantly smaller at only 4GB and is called mini-kraken.

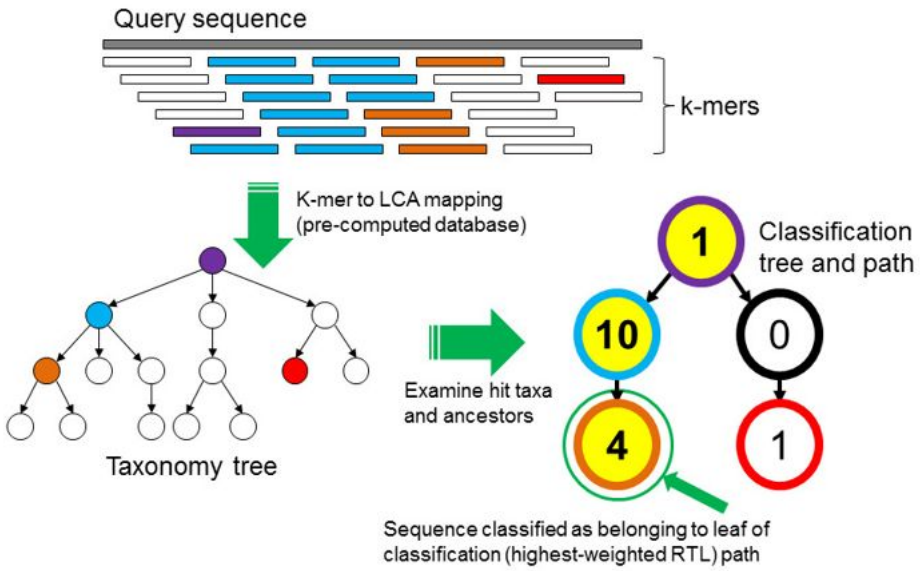

Figure 2.6: Figure showing the Kraken classification algorithm. The top shows all k-mers extracted from the DNA sequence. They are matched with the LCA, or Taxonomy tree. The total counts of each node found is then stored in a separate tree. The branch with the highest counts is chosen as the tree is traversed. When a leaf node is reached that is chosen as the class for the DNA sequence. Figure from [23].

3
Methods

3.1 Code

The code is mostly written in python and can be found on Github under the BSD 3 Clause license at:

`https://github.com/Gertdor/genomic-signatures-data-structure`

The generation of VLMC is done using code by Norlander and Gustafsson [10].

3.2 Data

The database used contains 8901 organisms from different superkingdoms and the distribution can be observed in table 3.1. It contains only complete genomes or whole chromosomes taken from the NCBI database [9]. All the organisms used can be found in the file *settings/organisms_in_db* on Github.
This database was used because it already existed with pre-computed signatures and relevant metadata. It was also sufficiently large and varied to perform the experiments. Most of the database was constructed by adding various viruses and their hosts.

3.2.1 GC content

Figure 3.1 gives an XY plot between the GC distances and FN distances in the dataset. Interestingly the GC content is a lower bound for the FN distance. This allows for pruning the left tree if the queried point's GC content is sufficiently different from the vantage point's GC content. Because GC distance is orders of magnitude faster to compute than FN this can speed up computation.

Number of Genomes	Kingdom
867	Bacteria
712	Plasmid
332	Eukaryote mitochondrion
1788	Eukaryote chloroplast
2761	Eukaryote
256	Archaea
2185	Viruses

Table 3.1: Distribution of kingdoms present in database.

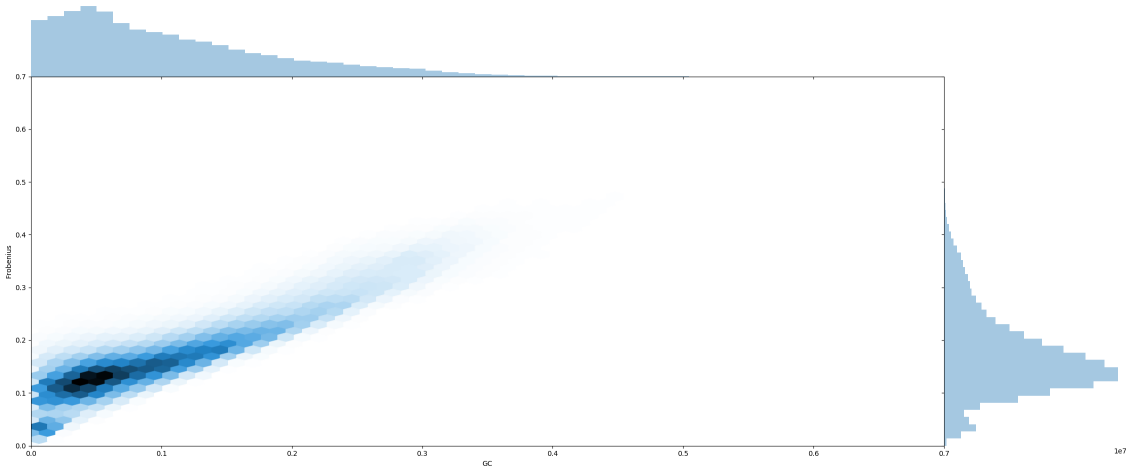

Figure 3.1: distribution between GC and FN distance. GC is a lower bound on the FN distance. The hump in very low FN distance are plasmids. X axis is GC distance and Y axis FN distance

3.2.2 Data division for experiments

For each experiment, the data was divided into two parts. 80% was used to construct the tree and the remaining 20% was used to query into the tree. These two parts were generated using RepeatedKFold function from sklearn [18]. This function takes all of the data and randomizes it, then divides it into five equal parts $\{p_1, p_2, p_3, p_4, p_5\}$. p_1 will first be used to query into a tree constructed from the remaining four parts. Then p_2 is used to query into a tree constructed from the other four parts and so on until all five segments have been used to query different trees. Importantly all the data is used as queries once and the queried points are never in the tree. This whole process of randomizing and splitting the data is repeated multiple times.

To be able to compare the accuracy between VLMCs from full genomes and VLMCs from reads, the reads were generated from the full genomes. This was done by two different methods. The first was taking sliding windows of different sizes and extracting substrings which were then saved. The sizes 160, 480, 720 and 1200 were used because they are divisible by 80 and as such contains a full row from the standard fna file format for DNA sequences. The four categories also cover most of the common read sizes. At most 10000 reads were generated from one genome. This is to limit the proportion of reads from longer genomes. The number of reads per superkingdom for reads of length 480 can be found in table 3.2

The second method was using wgsim from samtools [19] to create sequences with similar defects found by modern high-throughput sequencing machines. wgsim was run on the same original genomes but 1000 reads were generated from each original genome. Because of this there is a smaller proportion of Eukaryote reads in this dataset. The standard error values from samtools1.9 were used. wgsim was also run without errors to get a fair comparison on exactly how much the errors affected the classification accuracy.

Superkingdom	number of reads
Viruses	51159
Eukaryote chloroplast	149359
Eukaryote	4782712
Eukaryote mitochondrion	2684
Bacteria	1402675
Archaea	341102
unclassified	1023

Table 3.2: Table showing the number of reads per superkingdom. Even if the number of reads is limited to 10000 per organism there are still a clear majority of Eukaryote and Bacteria. The reads are 480bp long.

3.3 Classification methods

There were three different types of experiments. Either brute force or Classification with VLMC generated from either complete genomes or from reads. These three methods are explained further in this section.

3.3.1 Brute force method

The brute force method calculates all the pairwise distances between all the organisms in the database. These are then stored in a matrix and can be used to retrieve the true nearest neighbor using that distance function. This is used as a baseline to determine how much the different algorithms and pruning techniques alter the classification accuracy.

3.3.2 VLMC from complete genomes

The database contains VLMC generated from complete genomes. This data can, therefore, be used directly after splitting it as described in section 3.2.2. This kind of experiment was used to understand how different pruning techniques affect the classification accuracy and speed. They can also be seen as the likely best possible performance in terms of accuracy. They are however probably less useful than previously thought. Because these models are generated from much longer sequences many of the insights are hard to carry over to classification from reads, which is the end goal. However, unless specifically specified otherwise, all experiments are performed using this technique in the results section.

3.3.3 VLMC from reads

When performing nearest neighbor classification using VLMC the sequences first have to be turned into VLMCs. It is important to choose correct hyperparameters however it is not clear exactly how they should be chosen for optimal results. Because the reads are much shorter than the genomes the VLMC tend to be much smaller

than those in the tree. However, because the distance functions work for sequences of different sizes this is not a problem.
The generated VLMC is then used to query the tree. Some number of nearest neighbors were then saved and were used for classification. The classification algorithm used is weighted nearest neighbor classification.

3.4 VLMC

VLMC were used because Norlander and Gustafsson [10] found these to be better than fixed order Markov models. However, correctly choosing parameters when creating the VLMC can have a large impact on classification accuracy. Below are two tables 3.3 and 3.4 showing the effect of changing the min count when creating VLMCs for viruses.
As can be observed these parameters can have a large impact and should be chosen differently depending on what distance function is used and how long the DNA sequence is. The mincount values used for the experiments were 10 for viruses and 100 for everything else. The maximum number of parameters were 193 unless otherwise stated.

Min count	rank	classification accuracy
100	genus	**73.1%**
10	genus	55.2 %
100	family	**85.9%**
10	family	64.2 %

Table 3.3: Effect of min count on KL-divergence classification accuracy for viruses.

Min count	rank	classification accuracy
100	genus	48.7%
10	genus	**59.1%**
100	family	53.9%
10	family	**71.2%**

Table 3.4: Effect of min count on FN classification accuracy for viruses.

Distance function	average time (s)	genus accuracy
KL-div	18.5	39%
FN	0.0485	74%

Table 3.5: Difference between KL divergence and FN when used to classify herpes virus in a VP tree.

distance function	rank	max parameters	max accuracy
FN	genus	193	84.0%
FN	genus	255	83.0%
FN	family	193	91.6%
FN	family	255	91.7%
KL-div	genus	193	82.6%
KL-div	genus	255	77.2%
KL-div	family	193	90.8%
KL-div	family	255	86.6%

Table 3.6: Difference in classification accuracy between KL divergence and FN for bacteria for different number of parameters

3.5 Distances

The two main distance functions to consider are KL divergence and FN. KL divergence will give the true statistical difference between two VLMCs if the generated strings are long enough and FN is faster than KL divergence while still gives similar results. FN also obeys the triangle inequality much better than KL-divergence and has better performance if triangle inequality is used to prune results as can be seen in table 3.5.
It is also not true that KL-divergence will always have better classification accuracy. If the models are noisy, very long sequences can be needed before KL-divergence converges to the true difference. This can be observed in table 3.6. The KL-divergence was calculated using a sequence of length 10000 yet it is still not enough to overcome the noise of the mincount 10 models. Sequence length 10000 was used as it gives reasonable accuracy without being unreasonably slow.

3.6 Speedup techniques

3.6.1 Greedy factor

One simple way to speed up the nearest neighbor queries is to more greedily prune the branches of the VP tree. This can be done fairly easily by simply adding a factor $\alpha \in [0, 1]$, referred to as a *greedy factor*, to line 6 and 11 of algorithm 2. This means that instead of pruning a branch if $d(n.data, q) - d(best, q) < n.cutoff$ it is now done if $d(n.data, q) - d(best, q) < n.cutoff * \alpha$. For metric distances this also gives a $\frac{1}{\alpha}$ approximation.

3.6.2 Parallel searching

Because each query into the tree is completely independent it also supports parallel searching of nearest neighbors. Because each query is fairly fast this currently only scale well for 1000s of queries. It might be possible to improve this but it was not explored further in this thesis.

3.6.3 Forests

Another possible speedup technique is to split the data and create multiple trees which each contain a subset of the data. One possible way to divide this is by GC content because if two organisms have sufficiently different GC content they also cannot be near neighbors with regards to FN distance. This is a direct result of GC being a lower bound for FN distance. It could, therefore, be possible to split the dataset into multiple, slightly overlapping, buckets such that only one of these buckets have to be searched for the nearest neighbor. This would work best if the GC content is distributed uniformly, however, this is not the case. Figure 3.2 gives the GC distribution in the database. There is a large spike around 40% GC content and the majority of all organisms are in the 30-70% range.
Almost all of the nearest neighbors with respect to FN distance are within 11% GC. This gives a minimum bucket size of 22 plus the width of the actual bucket. Because of this most buckets would still contain almost the whole original database. It may be possible to split the data in some other way but this was not explored further.

3.7 Comparison to other methods

When comparing VLMC classification to Kraken the seed 15567 was used for RepeatedKFold and the first 20% was used as queries. The aid of each organism can also be found on Github under *settings/query_aids.txt* and *settings/db_aids.txt*.

3.7.1 Definition of classification accuracy

Classification accuracy refers to the number of correct classifications divided by the total number of classifications made at that rank. For example, for Kraken, the genus accuracy will calculate the total number of correct classifications at genus level divided by the total number of classifications made at genus level. That is, a superkingdom classification will not count towards the total number of classifications.

3.7.2 Kraken database and classification

Two Kraken databases were constructed. One containing all the 8901 organisms and one containing 80%. This was done with:

```
kraken2-build --add-to-library /path/to/data --db /path/to/db --no-masking
kraken2-build --download-taxonomy --db /path/to/db
kraken2-build --build --db /path/to/db --threads 16
```

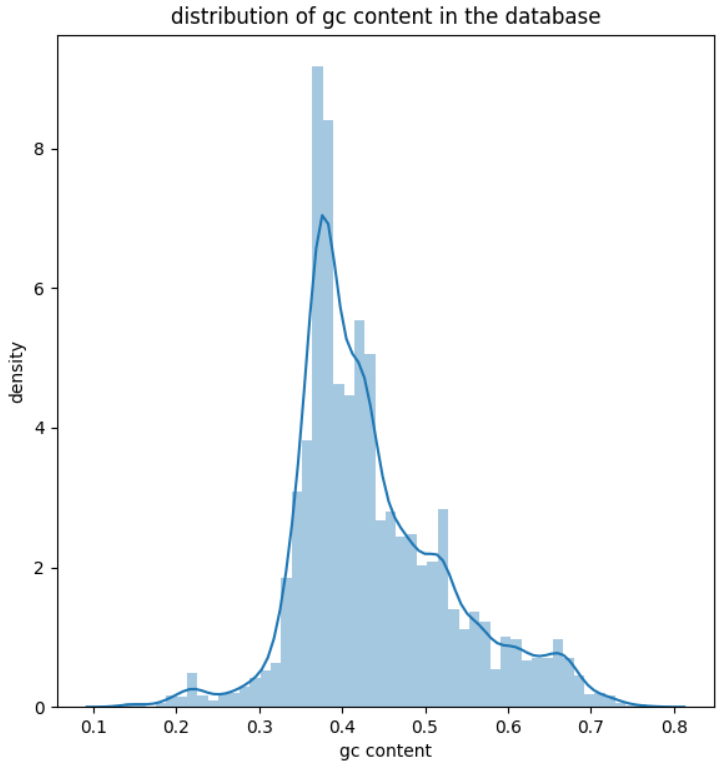

Figure 3.2: Distribution of GC content of all organisms in the database.

This resulted in either a 104Gb or 129Gb large database with a 29Gb large taxonomy database. Masking was not used because the necessary 3rd party dependencies were not installed on the server. The actual classification was done on one core with:

```
kraken2 --db /path/to/db /path/to/fna/folder/* | python reduceOutput.py
```

where reduceOutput.py takes the output of Kraken and saves only the id of the organism in the fna file as well as the taxonomic classification it was given.

4
Results

4.1 VP tree

Because the VP tree prunes using triangle inequality it no longer guarantees optimal results. Experiments show that around 2% of the nearest neighbors found are not the true nearest neighbor but rather a fairly close one.

4.1.1 GC pruning

Another technique for pruning the VP tree is to use the GC content of the signatures. As can be seen in figure 3.1 the GC distance can be used as a lower bound for the FN distance. It also tends to find the exact same neighbor. In the cases where it doesn't, it seems to be random whether the new neighbor is better, or worse than previously. The reason why it is possible to find a better neighbor is probably that FN does not obey triangle inequality. See figure 4.1 for a box plot over the result.

4.1.2 Forests

One way to construct a forest would be to split the data by GC content. This is however not very useful as most organisms have a GC content in the 0.4-0.5 range and the VP-tree does a good job of pruning distant points anyway. See fig 3.2 for the distribution of GC content in the database. The effect of using a forest can be seen in figure 4.1. Note how it has little to no effect on the number of distance calculations but may slightly improve accuracy.

4.1.3 Greedy factor

The effect of the greedy factor can be seen in 4.2 and 4.3. As can be observed the number of distance calculations needed decreases rapidly but the quality of neighbors is effected very little.

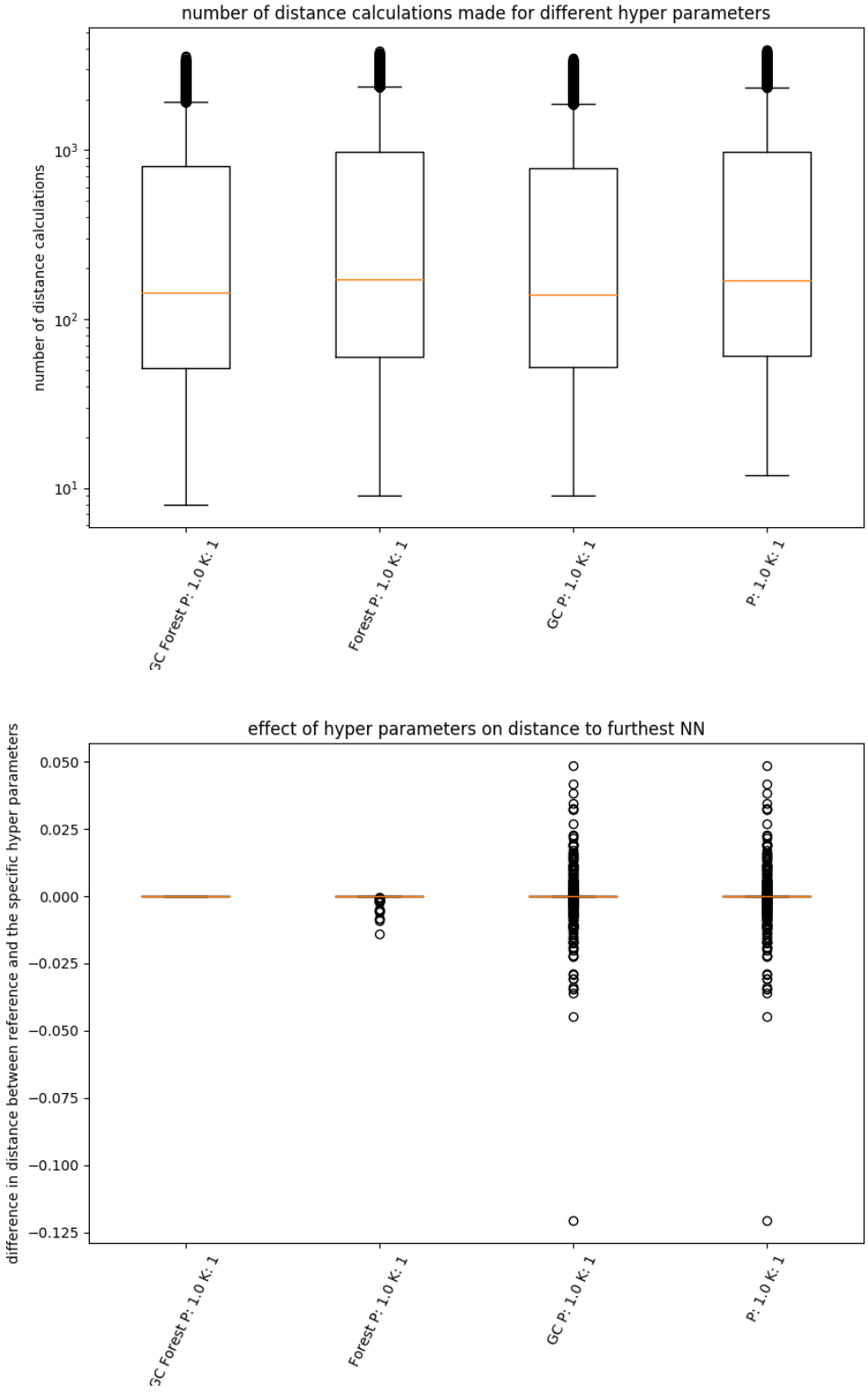

Figure 4.1: The effect of GC pruning and forest on the number of distance calculations made and effect on distance to the nearest neighbor.

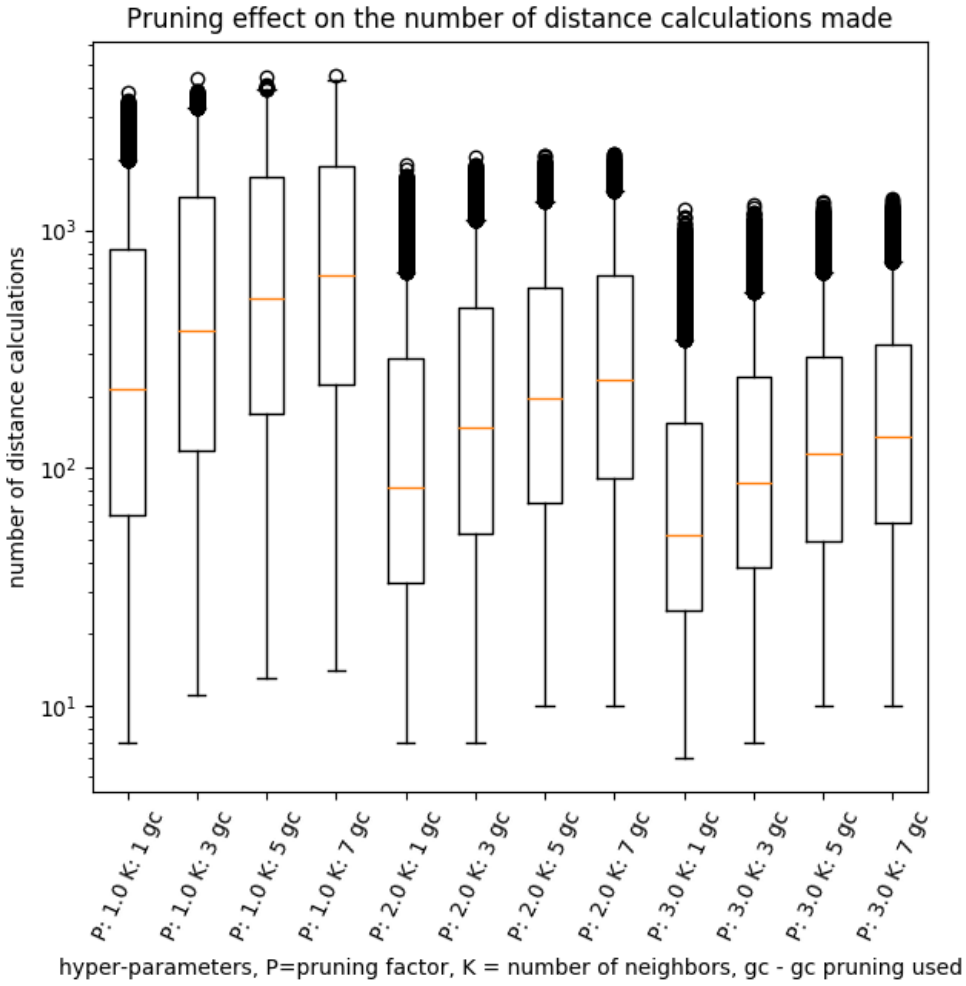

Figure 4.2: Number of distance calculations needed to find the nearest neighbor for different hyper parameters. Each block of slowly increasing 4 is for different number of neighbors: 1,3,5 or 7. Each block uses a different value of greedy factor: 1, 2 or 3. GC pruning is used everywhere

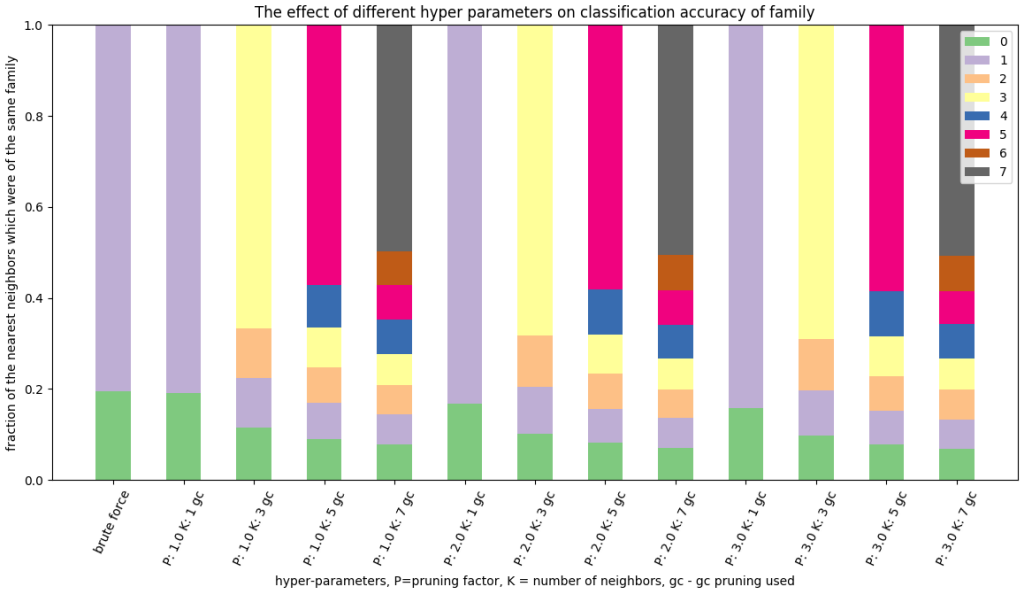

Figure 4.3: Effect of different hyperparameters family classification accuracy. The different colors represent the number of nearest neighbors that are of the same family. As the number of neighbors increases the probability of including the true family also increases. Higher greedy factor has little effect on the result. Note that it is possible that the family is not present at all in the database.

4.1.4 Multiple neighbors

Including more neighbors means that more information is passed to the classification portion of the algorithm. In general, this can aid classification but at the expense of more calculation. However, if the classification algorithm is not good enough it can decrease accuracy. Currently, even if using weighted nearest neighbor classification the accuracy decreases as more neighbors are added. Figure 4.3 shows the the number of times the correct family is present within the nearest neighbors found. If it exists atleast once the classification algorithm has some chance to classify correctly. If it is not (the green bar) then the classification algorithm cannot ever classify correctly. Similar results are obtained for genus also.

4.2 VLMC parameters

4.2.1 Maximum number of parameters

Changing the maximum number of parameters has a potentially large effect on both the classification accuracy and speed as can be seen in figure 4.4. The average distance to the nearest neighbor in the tree increases with model size which might be a result of the distance function used. Because the Frobenius norm on VLMC takes the intersection of states it will compare a different number of parameters depending on the size of the model. Because there are more states in the larger

model, there will in general also be more intersecting states. It is possible that this in turn results in larger distances.
The increase in distance calculations can be the result of two things. Either the distances to the query increases more than the distances within the tree or the dimensionality of the problem increases. Let us consider these in order: if the distances between elements in the tree remain constant but the distance to the query increases it will take longer to classify it. This is because the tree pruning is done by checking if a sphere around the queried point intersects with any possible nearer neighbors. If the distance to the nearest neighbor is larger then the volume of this sphere is also larger. This, in turn, increases the number of possible candidates and the search space can be pruned less aggressively resulting in more distance calculations. If the dimensionality increases the total amount of space increases exponentially and points tend to be more equidistant. This also results in fewer branches being pruned, and therefore an increase in distance calculations.
The effect of the maximum number of parameters also depends on the length of the DNA sequence which in turn depend on the organisms themselves. To study the effect of the maximum number of parameters on classification accuracy three representative groups were selected: Viruses, representing short genomes, Bacteria representing medium-sized genomes and Eukaryote representing the large genomes. Figure 4.5 shows the classification accuracy at family level for these three superkingdoms using models of different sizes. The classification accuracy was measured using FN and the brute force method. Note how the impact on viruses was much greater than the impact on both Bacteria and Eukaryotes. Eukaryotes and especially Bacteria see a small decrease in classification accuracy for 146 maximum parameters but otherwise stay fairly constant. Viruses, however, seem to have a clear spike near the 193 mark. The accuracy levels out at around 1875 because few viruses contain enough information to generate models this large even with a mincount of 10. The data for genus classification looks similar and is found in Appendix A.

4.2.2 Mincount

The original genomes that the models are generated from have vastly different lengths. Because of this they also contain vastly different amounts of information. This leads to the models behaving poorly if constants, such as mincount, do not depend on the genome length. Because of this, the mincount for the much shorter viruses was changed to 10, the performance increase can be found in table 3.4.

4.3 VLMC sequence classification results

4.3.1 Time

The time it takes to classify a sequence depends on two things. The time it takes to generate a VLMC from a given sequence and the time it takes to find the nearest neighbors of this VLMC. The time it takes to generate this VLMC depends on three major things: The length of the sequence, the maximum length, and the mincount. These all depend on each other and it is not trivial to predict the time it will take

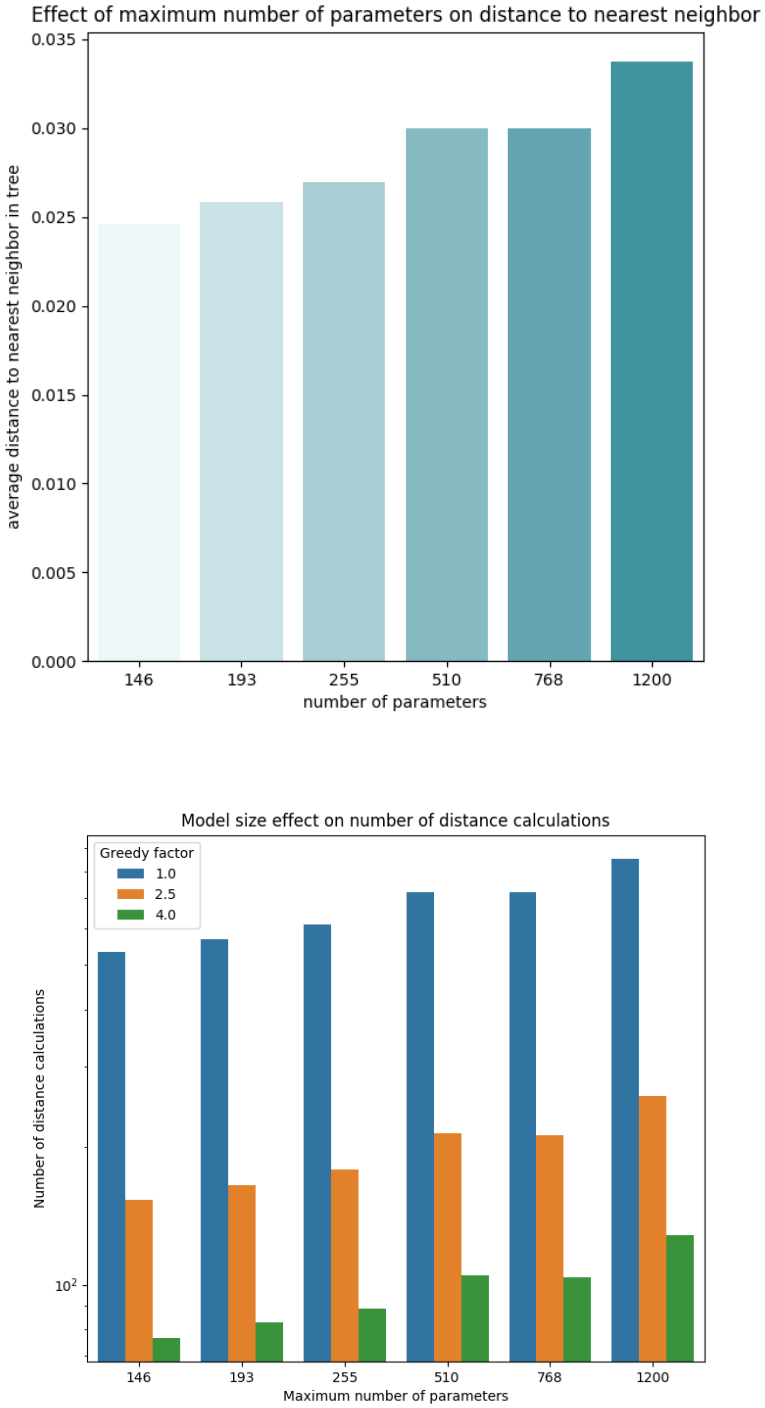

Figure 4.4: Barplots showing the effect of maximum number of parameters on the average distance to nearest neighbor found in tree and the number of distance calculations needed for different greedy factors. Note how the number of calculations made follows the average distance very well. Other parameters were Mincout 10 for Viruses and 100 for everything else, maximum branch length was 15.

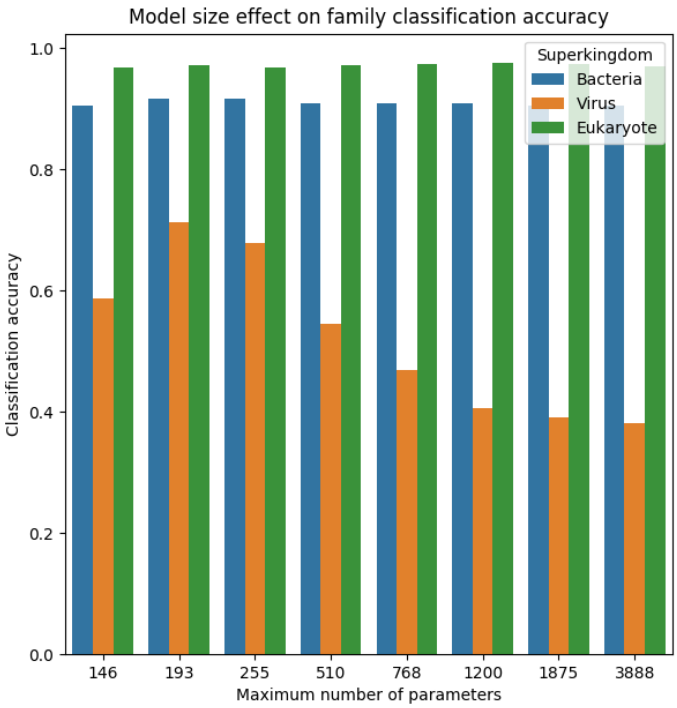

Figure 4.5: Family classification accuracy for Viruses, Bacteria and Eukaryotes using differently sized models. FN and brute force was used with only Viruses, Bacteria or Eukaryotes in the database.

max parameters	mincount	read length	time (minutes)
50	4	480	769
100	8	480	845
50	6	480	691
100	4	480	831
50	4	1200	1244
100	8	1200	848
50	6	1200	845
100	4	1200	1300

Table 4.1: Classification time for different VLMC parameters and reads of length 480 and 1200. Note how generally reducing mincount and having longer reads increases classification time.

to generate a sequence even given the value of these three variables. This is because the time also depends on the distribution of $\{A, T, G, C\}$ in the sequence.

The time it takes to classify a VLMC depends mostly on how close its nearest neighbor is and whether or not there is a large cluster of VLMCs in that area. If the nearest neighbor is close then more VLMCs can be pruned. However, if there are many other VLMCs close, many distance calculations may be needed. As seen before, having larger VLMCs tend to increase the time it takes to find the nearest neighbor.

When using GC pruning and a greedy factor of 2.5 it takes about equal time to generate the VLMC as it does to classify it for sequences of length 480 (mincount 4, maximum number of parameters 100, maxlength 5). For shorter sequences, the time it takes is slightly dominated by the classification time. For longer sequences, the classification time increases slightly while the VLMC generation takes much longer. The exact amount of time it takes can be found in table 4.1. As can be seen, the difference between 480bp and 1200bp is massive for mincount equals 4 and it is due to the slower VLMC generation. Why it was so slow to classify sequences with maximum number of parameters 100, mincount 8 and 480bp is unknown.

4.3.2 Accuracy

The classification accuracy is very poor when classifying reads using VLMCs as can be observed in table 4.3. The accuracy is low enough to be useless. The accuracy is, however, better for viruses than for any other organism. This is mostly because the VLMCs tends to classify everything as a virus. If one compares the distribution between the superkingdoms in table 4.2 and table 3.2 it is vastly different. Most reads are from Eukaryotes but most reads are classified as viruses.

If one instead looks at the classification accuracy of viruses in table 4.3 they are better. Importantly viruses tend to at least be classified into the correct superkingdom as can be seen in table 4.4. Even for only viruses, and superkingdom accuracy it is much too low to be of use.

Superkingdom	Classifications
Viruses	5057436
Archaea	71804
Eukaryote	712277
Bacteria	320126
Plasmids	569071

Table 4.2: The table shows which superkingdom each read was classified into. The reads are mostly eukaryote but classified as viruses

Queried Organisms	Sequence Length	Taxonomic Rank	Accuracy
all	480	genus	1.61%
all	480	family	2.28%
all	1200	genus	1.64%
all	1200	family	2.36%
Virus	1200	genus	14.5%
Virus	1200	family	18.3%
Virus	480	genus	9.92%
Virus	480	family	13.5%

Table 4.3: Classification accuracy for VLMC generated from different sized sequences. The accuracy is calculated when only virus reads are used and when all reads are used to query into the database. The hyper parameters were maxlength 9, mincount 4, maximum number of parameters 100

Superkingdom	Count
Viruses	45598
Archaea	559
Eukaryote	935
Bacteria	1365
Plasmids	2702

Table 4.4: Distribution of superkingdoms from querying 51159 virus reads of length 480 into the database. The hyper parameters were maxlength 9, mincount 4, maximum number of parameters 100

4.3.3 Effect of Noise on Classification Accuracy

Table 4.5 shows the effect of read noise on classification accuracy for VLMC. As can be seen, the classification accuracy is overall low and the effect of noise is similar for all experiments. Removing the noise also gives a decent accuracy improvement.

Rank	Read length	Noisy Read	Flawless Read	Improvement
genus	160	2.46%	2.65%	7.7%
genus	420	3.04%	3.30%	8.6%
family	160	4.24%	4.54%	7.0%
family	420	4.99%	5.31%	6.4%

Table 4.5: Effect of noise in sequence reads on classification accuracy. The accuracy is improved when using flawless reads. The classification was run on 1000 sequences from each genome.

4.3.4 Memory Usage

The VP tree containing the database of VLMC as well as the docker container and other overhead takes up around 3Gb of space. The memory usage of VLMC classification spikes to around 5.5 GB when training length 1200 sequences. This makes the VP tree very portable and easy to run on, for example, a laptop.

4.4 Kraken2 results

4.4.1 Accuracy

When kraken2 does it classification it will classify at different taxonomic ranks depending on the counts in the LCA database. Kraken tries to maintain accuracy while being as specific as possible. This type of dynamic classification is out of scope for this thesis and as such the Kraken results have been converted to three levels of specificity: genus, family or too general/unclassified. The distribution can be seen in the table below. For an exact distribution see Appendix A.
As can be seen in table 4.6 the longer the sequence the more specific the classification can become. Interestingly, the accuracy tends to get worse as the sequences, and the specificity, increases. It is also worth noting that for all sequence length if Kraken is forced to make a more general prediction at family level it is often incorrect. This can be observed in table 4.7 as a noticeable drop in accuracy even if family level predictions stand for only 3.6%-0.6% of the total predictions.

4.4.2 Time

Kraken is very fast and the time it takes to classify a sequence scales linearly with the sequence length.

Sequence Length	Proportion Genus	Proportion Family	Unclassified
160	82.3%	3.6%	14.0%
480	89.7%	1.3%	9.0%
720	91.8%	0.9%	7.4%
1200	93.7%	0.6%	5.7%

Table 4.6: Distribution of the taxonomic rank of classifications made by kraken. The genus column contains all classifications made at genus level or lower and family all classifications between genus and up to family. Everything more general is labeled unclassified.

Sequence Length	Genus Accuracy	Family Accuracy
160	92.8%	91.0%
480	92.0%	90.4%
720	91.5%	91.0%
1200	91.2%	90.8 %

Table 4.7: Total accuracy for predictions by kraken at genus level or lower and family level or lower. Note therefore that the family column also contains all genus level or lower predictions.

Number of Sequences (10^6)	Sequence Length	Time(s)
8.10	160	92
6.78	480	263
6.12	720	359
5.27	1200	660

Table 4.8: Classification time for different length of sequences in the kraken database containing 80% of the whole database.

4.4.3 Effect of noise on classification accuracy

Table 4.9 shows the effect of noise on Kraken's classification accuracy. There are two major things to note. Kraken seems to be better at classifying Eukaryote compared to other superkingdoms. This can be seen when comparing these results to the results of table 4.7. Second, the total effect of read noise on the classification accuracy is very small. Surprisingly, adding noise increases the accuracy of family level classifications for Kraken. This is because Kraken performs fewer species or genus classifications (just over 1.5% fewer) on the noisy reads. Instead these are mostly classified at a sub-family or family level. Kraken is therefore "over confident" when classifying flawless reads. This also means that the accuracy difference between genus and family classifications is much larger for noisy reads compared to flawless reads.

Rank	Read length	Noisy Read	Flawless Read	Improvement
genus	160	76.0%	77.6%	2.1%
genus	420	72.5%	73.6%	1.5%
family	160	82.6%	81.7%	-1.1%
family	420	89.1%	88.2%	-1.0%

Table 4.9: Effect of noise in sequence reads on classification accuracy. Kraken is affected surprisingly little by the noisy reads. Indeed the performance is better on the noisy reads for family classification. The classification was run on 1000 sequences from each genome.

4.4.4 Memory usage

Kraken2, on the other hand, used approximately 110-120Gb of memory when classifying the same length sequences in a database with the same information. Kraken2 does, however, have a special — 8GB — database that it is released with which is much more portable and can be run on a laptop. This small database was not used because it could not be used to train VLMCs. This meant that the algorithms would have vastly different possibilities of classifying the queries.

5
Discussion and Conclusion

5.1 Tree settings

5.1.1 Forest

A forest based on GC content does not work very well. It may be possible to divide the data by other factors. One possible solution is to use A, T, G and C content. This could, however, run into the same trouble. It is also important not to create too many buckets or the memory usage may explode.

5.1.2 GC pruning

GC pruning works well and there is almost no reason not to use it. It can however only do so much. Because it is a lower bound it is not possible to prune the right tree for example. If one were to find a fast heuristic that is an upper bound to the FN distance, or whichever distance function is used, that can be used to prune the right tree.

5.1.3 Greedy factor

The greedy factor is one of the main components that improve classification speed. It also has surprisingly little effect on the end classification if FN is used. The improvement from brute force to VP tree is about equal to the improvement from VP tree to VP tree with greedy factor of 2.5. The accuracy is also reduced more by going from brute force to tree than from tree without greedy to tree with. It is not known exactly why the tree is so good at predicting which branch to traverse first. It does affect the overall distance to the nearest neighbor, but if a further neighbor is chosen it tends to be of the same genus. It may be because the genus clusters well enough.

5.2 VLMC hyper parameters

The importance of the hyperparameters when generating VLMC is just starting to be understood. Going forward it may be reasonable to use genetic algorithms or other optimization strategies to find optimal hyperparameters for the different genomes. This can also shed some light on which parameters, if any, should depend on for example genome length or the superkingdom of the organism.

Currently, larger models do not seem to increase classification accuracy. This can, however, be because FN is ill-suited for large models or it may be because the smaller models capture all the information.

5.3 Distance functions

5.3.1 Between VLMCs from complete genomes

KL divergence was previously seen as the ground truth measurement of the distance between two VLMC. However, it seems to be much more susceptible to noise as indicated by table 3.3. It may, therefore, be infeasible to calculate these ground truth distances even for small databases. As such it is a idea good to consider other metrics as well when evaluating the performance of different distance functions. Examples could be classification accuracy, proportion of genus or family in x near neighbors, or the average number of neighbors needed to cover all members of a genus or family.
It may also be possible that the tree structure of the VLMC contains additional information not captured by the KL divergence. Because of this, it is possible that other distance functions can perform better than KL divergence even if KL divergence reaches the true statistical difference in the limit. Frobenius Norm is sadly especially ineffective at utilizing this tree structure. Due to taking the intersection of the trees long branches often get excluded and are therefore not counted. This also means that one of the main benefits of VLMCs are lost, namely the ability to store these long important branches. It may, therefore, be beneficial to combine FN with a distance function on the tree structure. This tree structure function could be as simple as the overlap of nodes, potentially weighted such that deeper branches contribute more. FN is still surprisingly effective at comparing VLMCs however as can be seen in both the results and [10].
Other possible information to consider is the exact number of times each node has been observed. Currently only the probability of observing each nucleobase is stored. If the number of times the current state was observed in the original genome was also stored that could be used to weigh these probabilities. Currently two probabilities weigh equally even if one has been observed 10 times and the other 1000 times.
Additionally it was found in this thesis that smaller models tend to result in smaller distances. Going forward it may be interesting to see if the average distance between models depends on the model size. For example, if all models tend to be closer to 190 parameter models than 500 parameter models that can be problematic. One would then have to take care to generate models all of the same size or classification results would be skewed.
In conclusion, one should make sure to use as much information as possible from the models when they are generated from a complete genome. This is because the models are very information-rich as they compress multiple megabytes of data into just kilobytes. KL divergence is more susceptible to noise than previously thought and going forward, new distance functions may be needed to increase performance. It is also important to note that all of this only applies to models generated from complete genomes, or at least very long sequences.

5.3.2 For classifying reads

When generating VLMC from reads the VLMC will either be very small, noisy, or both. Because of this, it is important that the distance function used is resistant to noise. It is also important that this distance function does not rely too much on the tree structure. The models one can feasibly generate from reads are much smaller than those stored in the database. If one were to compare the tree structure of the models it would favor small models unless all models are equal in size. It would then also be important to study the general structure of models generated from shorter sequences.
Instead, it may be better to find distance functions that work well on reads directly. If we compare the results from [7] and this thesis the classification accuracy of the VP tree is much worse than the brute force method of [7]. This is despite using similar sized reads and relying on VLMC to perform the classifications. One reasonable explanation for this difference is that reads contain too little information to generate useful VLMC. In [7] the classification was done by finding the model most likely to have randomly generated the read. This meant that none of the information in the read was lost, something that might happen when a VLMC is created.
It is also worth noting that the reason KL divergence is so slow is that it has to generate long sequences to overcome the randomness of the model. If these sequences are already generated, and quite short, the KL divergence is much faster to compute. This may make it, or other similar distance functions, viable for classifying reads.

5.4 Classification method

If one can skip the generation of VLMC from reads one can cut down on computation cost quite a bit, especially for slightly longer sequences. The main problem with KL is the generation of sequences and how long they need to be to accurately compare the two models. This is not the case for reads. They are short and if KL is used, reads can be directly queried into the tree. KL does, however, perform very poorly with triangle inequality pruning. As such a different pruning technique needs to be applied, for example, Bregman VP trees proposed by Nielsen et al. [15]. Even this is not enough, however, because the KL divergence is not symmetrical. This is also hard to overcome as it is not possible to compute the divergence from the read to the model. If one is able to overcome this problem it could improve performance drastically. If the Bregman VP tree is used then the intersection is no longer trivial to compute. It will, therefore, be important to limit the number of branches traversed. This can be done by having multiple children in the leaves or by having multiple vantage points as proposed by [2, 3].
It is also possible to improve the actual classification step of the nearest neighbor classifier significantly. This can be done by for example allowing classification at different taxonomic ranks. If multiple neighbors are found it is possible to calculate weights for species, genus or family accuracy, much like how Kraken does. This can increase the accuracy of the model and give users an indication of how confident the classification is. It is also possible to apply many distance functions in the

last classification step. Say five near neighbors are found, it is then reasonable to calculate multiple metrics between the read and the five found models. This can give additional information to the classification step and the computation time is constant with regards to the database size. It is, however, unreasonable to spend time on the classification step if the neighbors found are as bad as they are currently. If the true class is not within the found neighbors no classification algorithm can make a correct prediction.

5.5 Comparison to kraken

5.5.1 Classification speed

The classification speed of Kraken is orders of magnitude faster than VLMC classification. This is because it only does a linear amount of hash-lookups depending on the DNA sequence length. The only way for VLMC classification to be faster is if it did not need to generate a VLMC from the sequence and if the distance function was faster.
Kraken does, however, outperform most other classification methods in a similar manner in terms of speed. That is also one of the reasons Kraken was chosen, it is one of, if not the fastest method currently while still having great accuracy. The goal should instead be to get close enough to Kraken's classification speed but improve on the accuracy.

5.5.2 Classification accuracy

The read classification accuracy of VLMC classification is currently very poor. For it to be a useful classification algorithm this needs to be significantly higher.
For Kraken the proportion of sequences classified seems to match their paper quite well. However the classification accuracy is much lower than what they receive [23]. The experiments in the Kraken paper used reads from organisms in the database. For our experiments this was not the case as the reads were generated from chromosomes or genomes not in the database. In reality, where the sample is taken from will have a large impact on whether the species can be expected to be in the database or not. Kraken also had an even lower accuracy if the proportion of Viruses, Bacteria and Archaea was high. This is likely because almost none of these organisms have multiple chromosomes. As such it is impossible to make species level classifications — with the current experiment setup — for most of these.
Surprisingly Kraken had very poor results when it made classifications at family or subfamily level. For some experiments, it was as low as 30%. This is very different from the near 99% accuracy the paper states.

5.5.3 Sensitivity to read noise

One of the proposed reasons to use VLMC for classification is that it would be more noise resistant than string based methods such as k-mer counting [16]. This does however not seem to be the case. The experiments show that Kraken deals very well

with the noise that can be expected from modern sequencing methods. If further research is conducted on VLMC based classification improving the classification quality on poor reads could be interesting. At least it is important to use realistic reads as read quality has noticeable impact on classification accuracy.

5.5.4 Memory usage

This is an area where VLMC classification can be quite far ahead of other methods. The VLMC compress the database size significantly and as such the whole database fits without problem on a modern laptop. This is not true for Kraken. This one factor is not enough to make a classification algorithm useful. It is also worth noting that genome sequencing machines are very expensive. It is therefore not a large extra cost to also invest in a small to medium server that can easily fit Kraken's database in RAM.

5.6 Ethical considerations

When construction an algorithm that seeks to aid doctors, researchers and others to understand pathogens it is important not to make false claims. Exaggerating the accuracy by, for example, selecting only successful results could give misleading results that might lead to human harm or suffering. However, a successful and accurate algorithm has great potential to be instrumental in improving human health.
Because of how specialized the algorithm is, it is hard to use for anything else. It is also hard to do anything malicious with taxonomic classifications of DNA samples.

5.7 Concluding remarks

During this thesis, the complexity of using VLMCs for capturing genomic signatures has been better understood. It is now clear that it is not as straight forward as previously thought. Many of the results of the thesis are mostly relevant when comparing VLMCs from complete genomes. This is valuable information for many applications but probably not for classification.
The current method of generating VLMCs from reads performs poorly and major changes have to be made. This can be both in the form of a complete revision of the algorithm where VLMCs of the reads are no longer generated to changing the distance function and database data structure.
Current methods lack the ability to classify rare pathogens. If unparalleled accuracy can be achieved orders of magnitude slower performance is excusable. Currently, however VLMC classification cannot match current methods even if the queries are generated with the whole genome, except for Eukaryotes.
Going forward a test with a small number of reads should be conducted where the brute force accuracy of FN, KL divergence between VLMC, and KL divergence on the sequence should be tested for different VLMC parameters. This gives a theoretical maximum classification accuracy for VLMC classification. If none of the methods gives decent classification accuracy either the model, or the distance

function has to change. If something works then it is relevant to test the loss of accuracy by using nearest neighbor classification. This would include testing the effect of forest, GC pruning. greedy factor, multiple nodes per leaf and/or VP as well as different pruning techniques.

Bibliography

[1] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[2] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based indexing for high-dimensional metric spaces. In *ACM SIGMOD Record*, volume 26, pages 357–368. ACM, 1997.

[3] Tolga Bozkaya and Meral Ozsoyoglu. Indexing large metric spaces for similarity search queries. *ACM Transactions on Database Systems (TODS)*, 24(3):361–404, 1999.

[4] Peter Bühlmann, Abraham J Wyner, et al. Variable length markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[5] Scott Cost and Steven Salzberg. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine learning*, 10(1):57–78, 1993.

[6] Fabio Cuzzolin and Michael Sapienza. Learning pullback hmm distances. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1483–1489, 2014.

[7] Daniel Dalevi, Devdatt Dubhashi, and Malte Hermansson. Bayesian classifiers for detecting hgt using fixed and variable order markov models of genomic signatures. *Bioinformatics*, 22(5):517–522, 2006.

[8] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327, 1976.

[9] Lewis Y Geer, Aron Marchler-Bauer, Renata C Geer, Lianyi Han, Jane He, Siqian He, Chunlei Liu, Wenyao Shi, and Stephen H Bryant. The ncbi biosystems database. *Nucleic acids research*, 38(suppl_1):D492–D496, 2009.

[10] Joel Gustafsson and Erik Norlander. Clustering genomic signatures. 2018.

[11] Daniel H Huson, Alexander F Auch, Ji Qi, and Stephan C Schuster. Megan analysis of metagenomic data. *Genome research*, 17(3):000–000, 2007.

[12] B-H Juang and Lawrence R Rabiner. A probabilistic distance measure for hidden markov models. *AT&T technical journal*, 64(2):391–408, 1985.

[13] Samuel Kariin and Chris Burge. Dinucleotide relative abundance extremes: a genomic signature. *Trends in genetics*, 11(7):283–290, 1995.

[14] Tim R Mercer, Marcel E Dinger, and John S Mattick. Long non-coding rnas: insights into functions. *Nature reviews genetics*, 10(3):155, 2009.

[15] Frank Nielsen, Paolo Piro, and Michel Barlaud. Bregman vantage point trees for efficient nearest neighbor queries. In *2009 IEEE International Conference on Multimedia and Expo*, pages 878–881. IEEE, 2009.

[16] Peter Norberg. A novel pipeline for fast and accurate diagonistics of atypical pathogens, 2017.

[17] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Samtools. Samtools version 1.9. https://github.com/samtools/samtools, 2019. Accessed: 2019-07-13.

[20] Marcel H Schulz, David Weese, Tobias Rausch, Andreas Döring, Knut Reinert, and Martin Vingron. Fast and adaptive variable order markov chain construction. In *International Workshop on Algorithms in Bioinformatics*, pages 306–317. Springer, 2008.

[21] Nicola Segata, Levi Waldron, Annalisa Ballarini, Vagheesh Narasimhan, Olivier Jousson, and Curtis Huttenhower. Metagenomic microbial community profiling using unique clade-specific marker genes. *Nature methods*, 9(8):811, 2012.

[22] Jeffrey K Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, 40(4):175–179, 1991.

[23] Derrick E Wood and Steven L Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, 15(3):R46, 2014.

A

Appendix 1

Below are the taxonomic ranks of classifications made by kraken for 480 long reads:

'family': 50257, 'no rank': 678408, 'species': 4667413, 'genus': 241211, 'tribe': 9626, 'subtribe': 1869, 'class': 1490, 'subfamily': 21642, 'subclass': 227, 'superorder': 3269, 'superkingdom': 14392, 'subgenus': 10036, 'order': 7488, 'subspecies': 364238, 'species group': 2430, 'infraorder': 1927, 'parvorder': 1329, 'species subgroup': 2891, 'varietas': 71111, 'phylum': 351, 'infraclass': 3067, 'kingdom': 88, 'superfamily': 105, 'cohort': 59, 'forma': 38011, 'subkingdom': 34, 'suborder': 101, 'subphylum': 14, 'section': 51

max parameters	mincount	sequence len	time(min)
50	4	160	674
50	4	480	769
50	4	720	877
50	4	1200	1244
100	8	160	683
100	8	480	845
100	8	720	864
100	8	1200	848
50	6	160	683
50	6	480	691
50	6	720	730
50	6	1200	845
100	4	160	813
100	4	480	831
100	4	720	955
100	4	1200	1300

Table A.1: classification times for more hyper parameters when generating VLMC from reads

Method	Read length	Imperfect Read	Flawless Read	Improvement
Kraken	160	82.6%	81.7%	-1.1%
Kraken	420	89.1%	88.2%	-1%
VLMC	160	4.24%	4.54%	7.0%
VLMC	420	4.99%	5.31%	6.4%

Table A.2: Effect of noise in sequence reads on classification accuracy. This is run on 1000 sequences from each genome. As such there are fewer Eukaryote sequences in this dataset compared to the other. The classification is made at family level or lower.

number of parameters	average distance
146	0.02458
193	0.02585
255	0.02699
510	0.02997
768	0.03000
1200	0.03372

Table A.3: Exact value of average distance to nearest neighbor in tree for different sizes of models

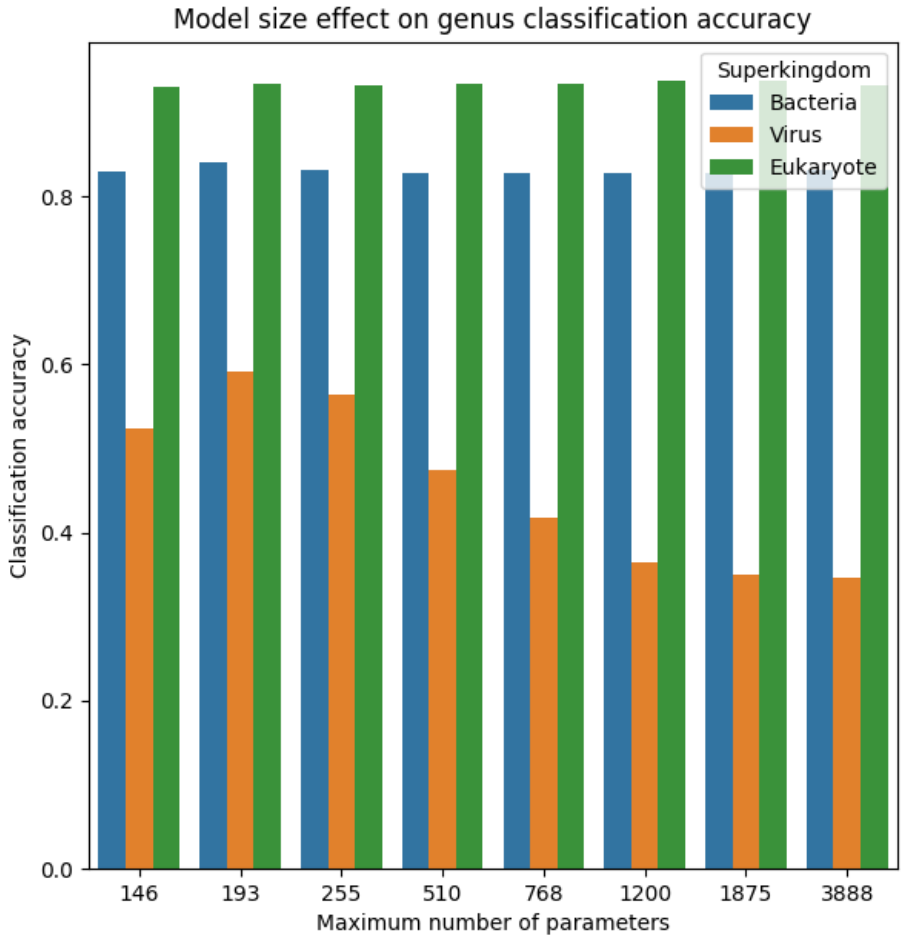

Figure A.1: Genus accuracy for Viruses, Bacteria and Eukaryotes. These were done using FN and brute force with just Viruses, Bacteria or Eukaryotes in the database.