Ernst-Moritz-Arndt-Universität Greifswald
Institute for mathematik and computer science

Subfamily Identification and Classification of the Influenza A Virus

Diploma thesis to obtain the degree

Diplom Biomathematikerin

submitted by

Sandra Van der Auwera

Thesis reviewer:
Prof. Dr. Mario Stanke
Dr. Mario Ziller

Greifswald, July 31, 2012

Contents

1 Motivation

"The single biggest threat to man's continued dominance on the planet is the virus. [1]" (Joshua Lederberg[1])

What Lederberg has realized years ago is catching up with all of us now. No matter whether it is bird or swine flu, every outbreak has a great influence on almost any area of our life.

Considering the recent worldwide flu pandemics like bird flu in 2007 or swine flu in 2009. Both of them were discussed in public and not always in an objective way ("'Mutant' fear for bird flu" The Sun June 24, 2006 [3]; "Four people with swine flu in Scotland die in past 24 hours" The Times October 23, 2009 [4]). The fear of this invisible enemy spreads panic ("Flu pandemic beats terrorism and flood in official table of reasons to be fearful" The Times August 8, 2008). Another aspect is the influence on the daily life ("Should we close our schools to slow down swine flu?" The Times July 22, 2009; "Airlines to turn away 'swine flu' passengers" The Times July 20, 2009). No matter if it is about health, public life or economy, a flu pandemic affects the whole world ("WHO declares global swine flu pandemic and says virus is 'unstoppable" ' The Times June 11, 2009). At this point scientist come into play. However, also the research on such a topic is of controversial public discussion ("Flu study sparks terrorist fears" The Sunday Times November 27, 2011). This got obvious when Ron Fouchier from the Erasmus Medical Center in Rotterdam[2] had to retain his research results on HPAIV last year ("Science journal asked to restrict bird flu virus information" The Times December 20, 2011).

It is important to keep the spread of these viruses under surveillance. Therefore, a unified nomenclature of the influenza subtypes is required to facilitate the exchange of information between all concerned scientists in different laboratories around the globe. For the *highly pathogenic avian influenza viruses* (HPAIV) of H5N1, commonly known as bird flu, such a nomenclature already exists. The next step in monitoring the evolution of the virus is the implementation of a classification system. My thesis aims to develop a classification method for HPAIV and to apply it to other influenza A subtypes. I want to describe a consistent method for the identification and classification of influenza A subfamilies. Different models and

[1]American molecular biologist; May 23, 1925 -February 2, 2008 [2]

[2]http://www.erasmusmc.nl/MScMM/faculty/CVs/fouchier_cv?lang=en (July 25, 2012)

methods are required for that task; a clustering to identify the subfamilies and a classification to assign sequences to these subfamilies. As one can imagine, there is a variety of methods that can be taken into account.

My objective is to develop an identification and classification pipeline for Influenza A subfamilies based on the current HPAIV nomenclature. For that reason I present the virus and the gene sequences I am working with (chapter 2). After that, in chapter 3 I introduce the theoretical sequence and distance models that are applied. Chapter 4 has a more practical attitude. I suggest clustering and classification methods and describe my algorithmic approaches. As a tool for classification I choose the `HMMER` [5] tool which is a profile based method. This pipeline is then applied to HPAIV H5N1 (chapter 5) and H1N1 (chapter 6) where I present the results of the analysis and evaluate my methods. I want to show whether `HMMER` is applicable to correctly classify influenza A sequences and how the classification accuracy can be improved.

2 The Influenza Virus

2.1 Virus Genetics

In this thesis I analyse gene sequences of different influenza A subtypes. The aim is to develop methods for the identification and classification of subfamilies of influenza A types based on their genetic similarity of the hemagglutinin gene. To provide background information, in this chapter the properties of the virus and its gene sequences are explained.

DNA and Genes

In 1944 Oswald Avery proved that nucleic acids and not, as one previously believed, proteins carry the genetic information. Today it is known that DNA (deoxyribonucleic acid) and RNA (ribonucleic acid) are polymers of single nucleotides. They only differ in the nucleobase that is attached to each sugar. The DNA bases are adenine (A), cytosine (C), guanine (G) and thymine (T). One can divide them into pyrimidine (C, T) and purine (A, G) bases depending on their chemical structure. This is why each gene sequence can be written as a word using the DNA alphabet $\mathcal{A} = \{$A, C, G, T$\}$; $w = \mathcal{A}^n$ is a word of length n. These four letters are the basis of genes, chromosomes and the genetic material in general. Their information provides the templates for building proteins and RNAs necessary for structural and biochemical processes [6].

Genome Evolution

Although DNA replication is very accurate, it is, like other complex biological processes, not free of mistakes. Some of these mistakes cause a non-viable cell or organism, others may have no or even a positive effect. There are small-scale mutations that affect only a few nucleotides and cause a change in the base sequence and large-scale mutations that affect the whole chromosomal structure. The former one can be subdivided into two different types.

1. **Substitution**: This is a mutation that causes a change of just one nucleobase in the sequence. One can distinguish between transitions (change within pyrimidines or purines; C$\leftrightarrow$T or A$\leftrightarrow$G; four possibilities) and transversions

(change between pyrimidines and purines; eight possibilities). Despite the fact that there are twice as much possibilities for transversion, transitions are observed more frequently in sequences.

2. **Indel**: This is a combination of the words insertion and deletion. The first one means to add one or more extra nucleotides while the second one is a loss of single nucleotides or small segments. Both mutations can alter the reading frame if the length of the removed or added sequence is not a multiple of three (called frame shift).

For genetically important DNA, such as DNA encoding for a gene, these mutations are rarely inherited. But although they can be harmful, they are also essential as a driving force of evolution. They can effect genetic differences among individuals, emerging of new species as well as adapting to a new environment [6].

2.2 Influenza A

Viruses are known to be infectious particles, also called virions. Their genome consists of either DNA or RNA which can be single- or double-stranded, linear, circular or segmented depending on the virus type.

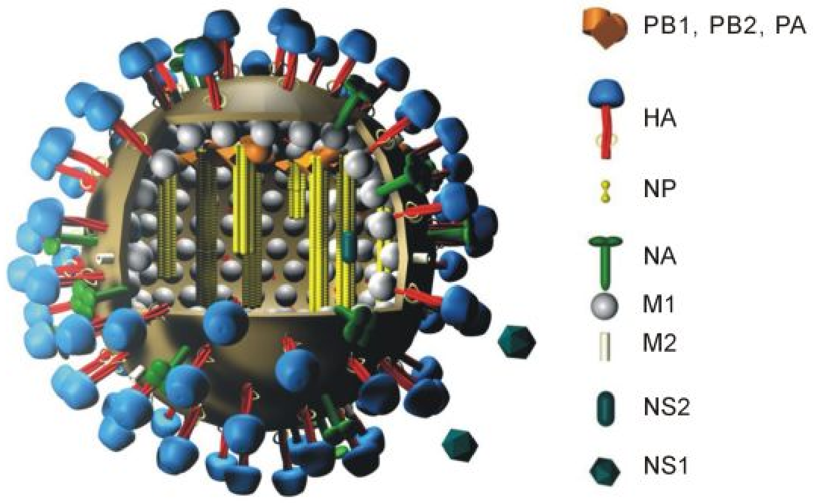


Figure 2.1: Physical structure of the influenza A virus

Structure

Influenza A, the biggest of the three different influenza genera (A, B, C), is a genus of the Orthomyxoviridae which includes coated viruses with negative sensed, single-stranded, segmented RNA as genome. In this context, negative sensed means that the RNA needs to be transcribed into complementary mRNA before protein synthesis can start. Figure 2.1 shows the physical structure of these viruses.

Since they have no mitochondria or ribosomes and thus no own metabolism, influenza A viruses depend on living cells they infect. These hosts are needed for protein synthesis and reproduction of the virus. This is why they are referred to as intracellular parasites which reprogram the metabolism of the host and use it for their own purpose.
All influenza A viruses are enveloped by a lipid membrane with included proteins which play an important part in viral attachment and cell lysis. These surface proteins are hemagglutinin (HA) and neuraminidase (NA). The inside of the virus contains a nucleocapsid with the whole genome on eight RNA-segments. These strands code for eleven proteins needed during the amplification cycle. Table 2.1 lists up the eight segments with their associated proteins.

segment	**protein**	**length in bp**
1	polymeraseprotein (PB2)	2,277
2	polymeraseprotein (PB1)	2,271
3	polymeraseprotein (PA)	2,148
4	hemagglutinin (HA)	1,698
5	nucleoprotein (NP)	1,494
6	neuraminidase (NA)	1,407
7	matrix proteins (M1, M2)	756 resp. 291
8	non-structural proteins (NS1, NS2)	690 resp. 363

Table 2.1: Overview of the segments of the influenza A virus and their coded proteins [7]

The two integral surface proteins mentioned before are of special interest because they have a great influence on the interaction with the host during the life cycle of the virus. Furthermore, the classification system of Influenza A bases on the HA- and NA-characteristic. Currently there are 17 different HA- and 10 NA-types known [8]. Their combination determines the notation A/HxNy of the virus subtype (e.g. H1N1, H5N1, H3N2). Hemagglutinin is essential for binding the receptor molecule on the host's cellular surface and causes the fusion of viral and cellular membrane. It is composed of three identical monomers which also have two different subunits (HA1 and HA2) themselves. In contrast, neuraminidase has an influence on the release of the virus through destruction of the host's cell membrane.

A special property of Influenza A is their ability to do genetic re-assortment. This means that two viruses can swap RNA-segments and create combinations with a new antigenic behaviour. But this can only occur when there is a double infection in some host. Viruses evolve very rapidly and are able to adapt themselves continually to their environment because of their short generation time, their high reproduction and their quite simple structure. Essentially, there are three processes which are important for the development of new successful subtypes:

- high mutation rate of the RNA-viruses,
- adaption to the host
- and genetic re-assortment.

The three subtypes listed above (H1N1, H3N2 and H5N1) are the most rife ones. The first and second one are human influenza viruses. H1N1 was responsible for the "Spanish flu" in 1918 and the "swine flu" pandemic in 2009. The subtype H3N2 caused the "Hong Kong flu" in 1968 [9]. In contrast to that, H5N1, better known as avian influenza or "bird flu", has not yet been proved to be human pathogenic. Only the bird-adapted strand of H5N1 which is named HPAIV (*highly pathogenic avian influenza virus*) can be transmitted from birds to humans in individual cases[1].

2.3 Classification of H5N1

Influenza A viruses are divided into subtypes based on their different surface proteins hemagglutinin and neuraminidase. Every sequence in the database gets a name containing:

- the virus type A/B/C,
- the host (if it is not human),
- the geographical place of the isolation,
- the number of the isolate,
- the year
- and the subtype HxNy.

[1]http://www.who.int/influenza/human_animal_interface/H5N1_cumulative_table_archives/en/index.html (July 26, 2012)

But this is not the only possibility for a classification. In recent years, the focus was concentrated on the HPAIV of H5N1. Scientists have monitored the circulation of this virus from region to region. Because it still spreads, infects different hosts and gets more divers, it became difficult to control its geographical expansion and rapid evolution. Somehow it seemed necessary to develop a consistent clade classification based on genetic diversity to better understand the virus' behaviour [10]. Since influenza is able to do genetic re-assortment, the first problem was to find the right sequence for comparison. But the specific H5 HA gene *A/goose/Guangdong/1996 H5N1* remained present until today and could be used as common ancestor sequence. The new nomenclature system should enable:

1. "a unified system to be developed to facilitate the interpretation of sequence/ surveillance data from different laboratories,
2. the labelling of clades by geographical reference to be replaced by a more representative system,
3. the phylogenetic tree to be expanded in the future
4. and a starting point to be established to develop a more extensive system in the near future that takes into consideration antigenic variation and re-assortment into multiple genotypes [10]".

This hopefully would help to better understand and predict the spread of the virus.

The HPAIV Nomenclature Study (2008)

To develop this system, a *H5N1 Evolution Working Group* was convened by the WHO (World Health Organisation), the OIE (World Organisation for Animal Health) and the FAO (Food and Agriculture Organisation) at the *Options for the Control of Influenza VI Conference* in June 2007. It lasted more than a year until the results were published [11]. All available nearly full length HA sequences (length ≥1,600 nucleotides) that have evolved from the ancestor sequence *A/goose/Guangdong/1996 H5N1* were taken from the public databases (Genbank National Center for Biotechnology, Los Alamos National Laboratories). After that, these sequences were grouped into distinct virus clades based on their phylogeny and their homology of the HA gene. The conditions that constitute a cluster were set up by the *H5N1 Evolution Working Group.*

Definition 2.3.1 (Clade/Cluster). *Each clade met the following criteria:*

1. *"maintain previously designated clade numbers where possible [11]",*
2. *"sharing of a common (clade-defining) node in the phylogenetic tree,*

3. *monophyletic grouping with a bootstrap value of ≥60% at the clade defining node (after 1,000 neighbour joining bootstrap replicates)*

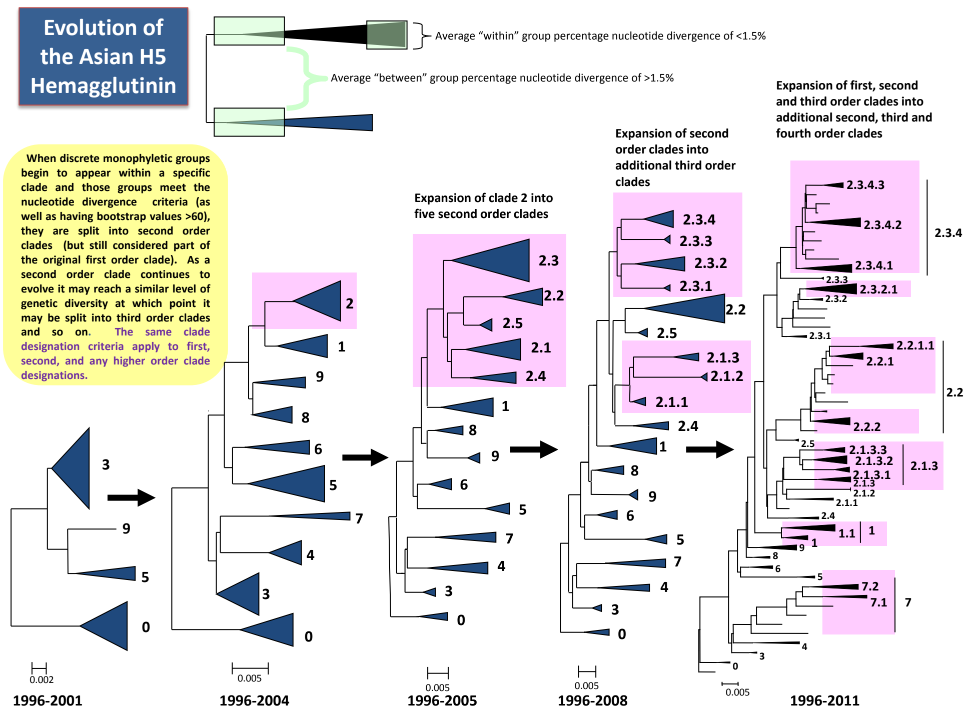
4. *and average percentage pairwise nucleotide distance between and within clades of >1.5% and ≤1.5% respectively; using K80 distance [10]".*

Clades were designated if there were at least four sequences which shared a common node in a monophyletic group, otherwise they were labelled as outliers [11].

By applying this method, ten distinct clades had been found and labelled as first order clades (0–9). Within these major clades the sequences continued to evolve and define new sub-lineages if they met the same clade criteria listed above. They were named as second or third order clade. For example, the first order clade (2) was divided into five second order clades (2.1), (2.2), (2.5), (2.4) and (2.3) which was also divided into four third order clades (2.3.1), (2.3.2), (2.3.3) and (2.3.4).

The HPAIV Nomenclature Study (2011)

The current analysis of 2011 used 2,947 HA gene sequences. 1,637 of them were sequenced from 2008 to 2010 and the number of new sequences is still increasing.


When discrete monophyletic groups begin to appear within a specific clade and those groups meet the nucleotide divergence criteria (as well as having bootstrap values >60), they are split into second order clades (but still considered part of the original first order clade). As a second order clade continues to evolve it may reach a similar level of genetic diversity at which point it may be split into third order clades and so on. The same clade designation criteria apply to first, second, and any higher order clade designations.

Evolution of the Asian H5 Hemagglutinin
Average "within" group percentage nucleotide divergence of <1.5%
Average "between" group percentage nucleotide divergence of >1.5%
Expansion of clade 2 into five second order clades
Expansion of second order clades into additional third order clades
Expansion of first, second and third order clades into additional second, third and fourth order clades
1996-2001
1996-2004
1996-2005
1996-2008
1996-2011
0.002
0.005
2.3.4.3
2.3.4.2
2.3.4.1
2.3.4
2.3.3
2.3.2.1
2.3.2
2.3.1
2.2.1.1
2.2.1
2.2.2
2.2
2.1.3.3
2.1.3.2
2.1.3.1
2.1.3
2.1.2
2.1.1
2.5
2.4
2.3
2.1
1.1
7.2
7.1
0
1
2
3
4
5
6
7
8
9

Figure 2.2: Evolution of the Asian H5 hemagglutinin (http://www.who.int [10])

Since the last nomenclature update in 2008, 13 new clades were identified. On the one hand many clades that have already diverged into third order clades form new fourth order clades, e.g. (2.3.4) splits up into (2.3.4.1), (2.3.4.2) and (2.3.4.3). But on the other hand many older clades have not been detected for years (0, 2.1.1, 2.1.2, 2.3.1, 2.3.3, 2.4, 3, 4, 5, 6, 8 and 9) and it seems likely that they "have been supplanted by new clades and become inactive [10]". Figure 2.2 illustrates the ongoing evolution of the HA ancestor sequence *A/goose/ Guangdong/1996 H5N1* from 1996 to 2011. Expectedly, the numbering system is directly linked to the HA phylogeny and so removes geographical designations.

The *H5N1 Evolution Working Group* has the aim to repeat these studies every few years to keep the HPAIV evolution under surveillance.

3 Models for Sequence Analysis

To handle biological sequence data mathematically, theoretical models are required to describe the data and their properties. In bioinformatics many models have been developed to analyse sequence data and simulate biological processes. Some of these models are used in my thesis and explained below for better understanding.

3.1 Alignments

As one is given biological sequences such as DNA, RNA or amino acids over a finite alphabet Σ, a frequently asked question is if the sequences are related. To compare this kind of data, the sequences are arranged as pairwise or multiple alignments. Moreover, a score is assigned to evaluate if an alignment occurs because the sequences are related or just by chance. This score depends on the differences and similarities between the sequences. However, each alignment that is calculated remains a best guess according to the parameters of the scoring system [12] [13].

3.1.1 Pairwise Alignments

Given two sequences $Y^{(1)}, Y^{(2)} \in \Sigma = \{A, C, G, T\}$ they are written one above the other in a way to maximize the number of residue pairs that match. While doing this, the order of the residues in both sequences is preserved but gaps can be introduced (gap-character: $-$). These gaps represent the insertions and deletions (indels) that occurred during evolution.

Definition 3.1.1 (Pairwise Alignment [13]). *An alignment between* $Y^{(1)} = Y^{(1)}(1), Y^{(1)}(2), \ldots$ *and* $Y^{(2)} = Y^{(2)}(1), Y^{(2)}(2), \ldots$ *is a matrix* $A = \begin{pmatrix} Y^{(1)'} \\ Y^{(2)'} \end{pmatrix}$ *with* $Y^{(1)'}, Y^{(2)'} \in \Sigma \cup \{-\}$ *and with the following properties:*

- *there are two rows, one for* $Y^{(1)'}$ *and one for* $Y^{(2)'}$*,*
- *the left-to-right ordering of the sequences* $Y^{(1)}$ *and* $Y^{(2)}$ *must be preserved but gaps are allowed at any position,*
- $|Y^{(1)'}| = |Y^{(2)'}| = L$ *,* L*: alignment length,*

- $(Y^{(1)'}(i), Y^{(2)'}(i))$, $i = 1, \ldots, L$ *are called alignment columns*
- *and* $\nexists\, i : Y^{(1)'}(i) = Y^{(2)'}(i) = -$.

There are three different states of alignment columns that can emerge when comparing two sequences; match, mismatch and gap (insertion or deletion). See below: $(N, N' \in \Sigma)$

$$\text{match: } \begin{pmatrix} N \\ N \end{pmatrix} \;\; ; \text{ mismatch: } \begin{pmatrix} N \\ N' \end{pmatrix} \;\; N \neq N' \; ; \text{ gap: } \begin{pmatrix} - \\ N \end{pmatrix} \;\; \text{or} \;\; \begin{pmatrix} N \\ - \end{pmatrix} \; .$$

When inserting an arbitrary number of gaps, one can design alignments with match and gap states only. Nevertheless, such an alignment is biological little likely because it disregards the occurrence of substitutions during evolution. E. g.,

the two sequences $Y^{(1)} = ATGCCT$ and $Y^{(2)} = AACCCTA$ can be aligned using

$$\text{only match and gap states:} \quad A = \begin{pmatrix} A & T & G & - & - & C & C & T & - \\ | & & & & & | & | & | & \\ A & - & - & A & C & C & C & T & A \end{pmatrix}$$

$$\text{or considering mismatches:} \quad A = \begin{pmatrix} A & T & G & C & C & T & - \\ | & & & | & | & | & \\ A & A & C & C & C & T & A \end{pmatrix} .$$

One wants to find the alignment that reflects the true relationship of the sequences best. For this reason, it is necessary to somehow weight the different states according to their biological meaning.

Definition 3.1.2 (Scoring System [13])**.** *This scoring system usually rewards matches and penalises mismatches and gaps. The scores are stored in a so called scoring matrix* $M = (m_{ij})_{i,j \in \Sigma \cup \{-\}}$; $N, N' \in \Sigma$

- $m(N, N) > 0$,
- $m(N, N') < m(N, N) \quad N \neq N'$,
- $m(N, -) < 0, \;\; m(-, N) < 0$.

The score $S(A)$ for an alignment $A = \begin{pmatrix} Y^{(1)'} \\ Y^{(2)'} \end{pmatrix}$ can be calculated by summing up the scores for each column while the alignment columns are assumed to be independent:

$$S(A) = \sum_{i=1}^{L} m(Y^{(1)'}(i), Y^{(2)'}(i)) \; . \tag{3.1.1}$$

3.1.2 Multiple Sequence Alignments (MSA)

Sequence alignments have a wide area of application in bioinformatics. For example, to draw a conclusion about the evolution from a common ancestor, to identify conserved structures or to build profiles for particular sequence families. They are often an initial step for further, more complex analyses. Multiple sequence comparison on m sequences $Y^{(1)}, \ldots, Y^{(m)}$ means that similar residues among the sequences are aligned together in columns such as $|Y^{(1)\prime}| = \cdots = |Y^{(m)\prime}|$ [12]. Like with pairwise alignments these columns are treated as independent so that the score of the alignment A of length L can be written as follows:

$$S(A) = \sum_{i=1}^{L} S(A_i) \,,$$

where A_i is the i-th column. A standard method to score each column is the SP-score (sum of pairs score) where summing up all the pairwise substitution scores in the column

$$S(A_i) = \sum_{j<k} m(Y^{(j)}(i), Y^{(k)}(i)) \ .$$

Here $Y^{(j)}(i)$ is the i-th residue in the j-th sequence and $m(a,b)$ is the value from the scoring matrix M. Given m sequences the multiple alignment is wanted that gets the maximum score.

3.2 Markov Models

There are many processes that follow strict deterministic rules but also many that involve chance. In the latter case the behaviour of the process is determined by a random process. A hidden Markov[1] model (abbr. HMM) is a general form of a probabilistic model for sequences of symbols [12]. They are based on Markov processes which I will introduce first.

3.2.1 Markov Chains

A Markov chain describes a random process in which the current state of a sequence depends only on the previous state [15].

[1]Andrei Andreyevich Markov (June 14, 1856-July 20, 1922), Russian mathematician [14]

Definition 3.2.1 (Markov Chain)**.** *A stochastic process $X = \{X_i\}_{i=1,2,\dots}$ on a finite, discrete state space S is called a Markov chain if and only if it holds the following property, also known as Markov property: $\forall\, i > 0$ and $\forall\, x_1, x_2, \dots, x_{i+1} \in S$*

$$\begin{aligned} P(X_{i+1} = x_{i+1} | X_1 = x_1, \dots, X_i = x_i) &= P(X_{i+1} = x_{i+1} | X_i = x_i) \\ &= P(X_{i+1} | X_i) \quad . \end{aligned} \tag{3.2.1}$$

The state sequence is called homogeneous if the transitions $P(X_{i+1} = s \mid X_i = r)$, $r, s \in S$, do not depend on i, otherwise inhomogeneous. In other words, the behaviour of a system is just influenced by the current symbol r and the probability $P(X_{i+1} = s \mid X_i = r) = t_{rs},\ r, s \in S$. For a homogeneous Markov chain t_{rs} is the probability to make a transition from state r to s.
It is called transition probability and the matrix $T = (t_{rs})_{r,s \in S}$ is called the transition matrix. By applying equation (3.2.1) repeatedly, one gets the probability of the whole state sequence $X = X_1, \dots, X_L$ [12]:

$$\begin{aligned} P(X) &= P(X_L, X_{L-1}, \dots, X_1) \\ &= P(X_L | X_{L-1}, \dots, X_1) \cdot P(X_{L-1} | X_{L-2}, \dots, X_1) \cdot \dots \cdot P(X_1) \\ &= P(X_1) \cdot t_{X_1 X_2} \cdot \dots \cdot t_{X_{L-1} X_L} \\ &= P(X_1) \cdot \prod_{i=2}^{L} t_{X_{i-1} X_i} \quad . \end{aligned} \tag{3.2.2}$$

3.2.2 Hidden Markov Models (HMM)

In recent years the sequencing technology has been improved continuously and also the amount of sequence data has been increased. As a result, it became more and more difficult for the employees of the laboratories to keep up with the sequencing and to analyse each sequence (DNA, RNA or amino-acid) experimentally.

This is where computational methods get involved. In many cases this means just to put the right label of a property or a function on each residue of the sequence. Assumptions have to be made about these unknown labels. To detect the most likely ones, hidden Markov models are used, “probabilistic models of linear sequence labelling problems [16]”. They have a wide area of application in bioinformatics; for example in gene identification, detection of CpG islands, sequence alignments, profile search or regulatory site identification.

Definition 3.2.2 (Hidden Markov Model (HMM) [15]). *A HMM is a stochastic model which consists of a Markov chain $X = \{X_i\}_{i=1,2,...}$ on a state space S (see Def. 3.2.1) and a sequence of random variables $Y = \{Y_i\}_{i=1,2,...} \in \Sigma$ finite. Both hold the equations*

$$\begin{gathered} P(X_i|X_1,\dots,X_{i-1},Y_1,\dots,Y_{i-1}) = P(X_i|X_{i-1}) \\ \textit{and} \\ P(Y_i|X_1,\dots,X_i,Y_1,\dots,Y_{i-1}) = P(Y_i|X_i) \ . \end{gathered} \tag{3.2.3}$$

- *Y is the emission sequence; Y_i is the i-th emission,*
- *X is the state sequence; X_i is the i-th state,*
- *Σ is the emission alphabet,*
- *S is the state space.*

The result of this process are two strings of information; the sequence of states $X_1, X_2, \dots$ and the emission sequence $Y_1, Y_2, \dots$. In typically applications only the emission sequence Y is observed while the state sequence X is not observed ("hidden"). A very descriptive example is the following one (Example 3.2.1).

Example 3.2.1 (A toy HMM: 5' splice site recognition [16]). *The example of a 5' splice site recognition problem is illustrated in Figure 3.1 where the HMM is represented as a graph. The DNA sequence contains an exon followed by an intron and somewhere in between a 5' splice site. This site has a determined structure. It can only be at an A or G. Looking at the sequence in Figure 3.1 there are 14 possibilities.*

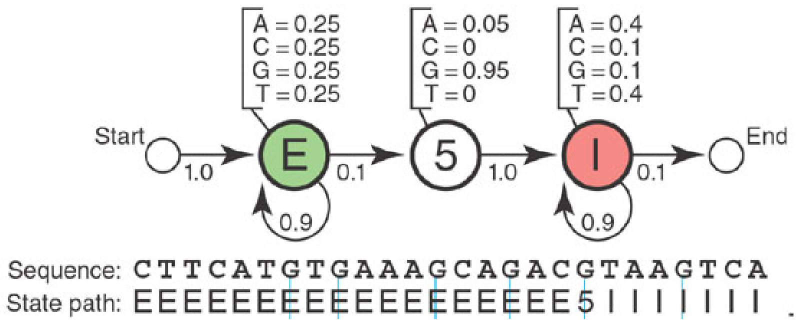


Figure 3.1: A toy HMM for 5' splice site recognition

The model has three states: E (exon), 5 (5'SS) and I (intron). Each of them has its own emission probability of observing A, C, G or T. To make the system *complete, a start and end state are added (compare equation (3.3.4)). The switches between the states are given through transition probabilities. In each state a symbol is emitted according to the state's emission probabilities and the next state to visit is chosen by the state's transition probabilities. Only the sequence can be observed while the state path is a hidden Markov chain. The questions now is: What is the state path and what is the most possible position for the splice site?*

Searching with HMMs

There are different ways to determine the most probable state path X^* given the emission sequence Y. The most common methods use the Viterbi or the Forward recursion [12].

The Viterbi algorithm chooses a X^* that maximizes the probability of X given Y, $P(X|Y)$, using Viterbi recursion.

Definition 3.2.3 (Viterbi Recursion)**.** *Assume the probability for the best path ending in $r \in S$ with observation i is known to be $v(r, i)$.*
Recursion:

$$v(s, i+1) = e_{sY_{i+1}} \max_{r \in S} \; v(r, i) \; \cdot t_{rs} \quad , \; s \in S \, . \tag{3.2.4}$$

Through the recursion for $i = 1, 2, \ldots, n$ X^* is found. This state path X^* is important for the HMM training (see section 4.2.2) when the alignment of the query sequence and the profile is required.

With the Forward equations the probability to see the sequence $Y_1, \ldots, Y_i$ when being in state r at time i is calculated [12].

Definition 3.2.4 (Forward Recursion)**.** *Let $f(r, i)$ be the probability to be in state r at position X_i and having observation $Y_1, \ldots, Y_i$.*
Recursion:

$$f(s, i+1) = e_{sY_{i+1}} \sum_{r \in S} f(r, i) \, t_{rs} \; . \tag{3.2.5}$$

The tool I use for classification (`HMMER` [5], section 3.4) applies the forward recursion to calculate $P(Y|\mathcal{H})$ and thus calculate the scores (see section 3.4.3).

3.3 Profile HMMs for Sequence Families

Hidden Markov models can be generalized.

Definition 3.3.1 (generalized HMM (GHMM) [15]). *A GHMM is defined by a distribution*

$$P_{GHMM}(X,Y) = \prod_{i=1}^{L} t_{X_{i-1}X_i} \cdot e_{X_i Y(b_{i-1},b_i]} \ , \qquad (3.3.1)$$

where

- *Y is a sequence,*
- *t_{uv} are transition probabilities for states u and v,*
- *e_{us} are emission probabilities for a state u and a string s,*
- *$Y(b_{i-1}, b_i] = Y[b_{i-1}+1 \dots b_i]$ is the sequence of the i-th segment of parse X.*

It is no longer assumed that a state emits a single symbol. The model is extended so that in each state a string can be emitted. A special case of such a probabilistic model are the profile HMMs (abbr. pHMM). Here the string s in Def. 3.3.1 has length 1 or 0 whether the actual state emits a symbol or not. Profile HMMs capture patterns of a set of sequences and express them as statistical features. This profiles can be taken to identify similarity or relationship between a single sequence and a sequence family. I use profile HMMs to assign sequences of Influenza A to the subfamilies of the specific influenza subtype for the purpose of classification. In the following sections I will introduce the pHMM architecture and explain how their parameters are estimated and how sequences are scanned against the profiles.

3.3.1 Profile HMM Architecture

Considering the possible structure of a multiple sequence alignment the architecture of a profile HMM for DNA sequences can be built [12]. For every consensus column in the alignment (a column with a minimum fraction of nucleotides) a match state $M_1, ..., M_n$ is defined and they are linked $M_i \to M_{i+1}$. Each state has its own emission distribution

$$\begin{aligned} &e_{M_i q} = P(Y_j = q | X_i = M_i) \ , \\ &q \in \{A, C, G, T\} \ \ , \ i \in \{1, \dots, n\} \ \ , \ j \in \{1, \dots, |Y| = L\} \ . \end{aligned} \qquad (3.3.2)$$

They model the occurrence of the residues $q \in \{A, C, G, T\}$ allowed in consensus column i. Further states need to be added to model indels. The approach is to treat insertions and deletions separately and to allow them at each position in the alignment. A set of $n+1$ insert states $I_0, \dots, I_n$ between two adjoined match states allow for insertion of arbitrary length. The inserts get their own emission distribution $e_{I_i q}$, too:

$$\begin{aligned} &e_{I_i q} = P(Y_j = q | X_i = I_i) \ , \\ &q \in \{A, C, G, T\} \ \ , \ i \in \{1, \ldots, n\} \ \ , \ j \in \{1, \ldots, |Y| = L\} \ . \end{aligned} \tag{3.3.3}$$

It must be possible to make a transition $M_i \to I_i$, $I_i \to M_{i+1}$ and a transition $I_i \to I_i$ to itself. At last n delete states $D_1, ..., D_n$ (silent states) are added that do not emit a residue. Therefore, state D_i has transitions $M_{i-1} \to D_i$, $D_i \to D_{i+1}$ and $D_i \to M_{i+1}$. Now every pair of match states can be linked while using the delete states between them.
Finally, the question in which particular state $s \in S$ the model might start or end must be resolved. So an extra beginning state $\mathcal{B} = X_0$ with a starting probability π and similarly an end state $\mathcal{E} = X_{n+1}$ are added. Begin and end satisfy:

$$\begin{aligned} &P(X_1 = s) = t_{\mathcal{B}s} = \pi_s \quad \text{and} \\ &P(\mathcal{E} \mid X_n = s) = t_{s\mathcal{E}} \ \ , \ \forall s \in \{M_n, I_n, D_n\} \ . \end{aligned} \tag{3.3.4}$$

The model architecture for the pHMM is summed up in Figure 3.2. Each position $i = 0, 1, \ldots, n$ gets own transition probabilities and every state $M_1, \ldots, M_n$ and $I_0, \ldots, I_n$ gets an own emission distributions.
The five parameters of the profile HMM $\mathcal{H} = (S, T, \Sigma, E, \pi)$ are:

1. the states S of the model (here: $S = \{M, I, D\}$),

2. transition probabilities t_{rs} for the Markov chain (one transition matrix T for each position $i = 1, \ldots, n$):

$$T = (t_{rs})_{r,s \in S}, \quad t_{rs} = P(X_{i+1} = s \mid X_i = r), \quad \sum_{s \in S} t_{rs} = 1 \ \ ,$$

3. the symbol alphabet Σ of the emitted sequence, $Y_i \in \Sigma$, $i = 1, \ldots, L$, (with DNA $\Sigma = \{A, C, G, T\}$),

4. emission distribution e_{sq} (one for each match and insert state $(M_1, \ldots, M_n)$, $(I_0, \ldots, I_n)$):

$$E = e_{sq} = P(Y_i = q \mid X_i = s), \ s \in S \ , \ q \in \Sigma \ , \ \sum_{q \in \Sigma} e_{sq} = 1$$

5. and the starting probabilities π with $\pi_s = P(X_1 = s)$, $s \in S$, assuming to start in an initial state $X_0 = 0$.

The two sequences, the emission sequence Y and the hidden state sequence X, constitute a pHMM $\mathcal{H}$. Their joint probability can be calculated using equation (3.3.5) [12]:

$$P(X,Y \mid \mathcal{H}) = t_{0X_1} \prod_{i=1}^{L} e_{X_iY_i}\, t_{X_iX_{i+1}} \ . \tag{3.3.5}$$

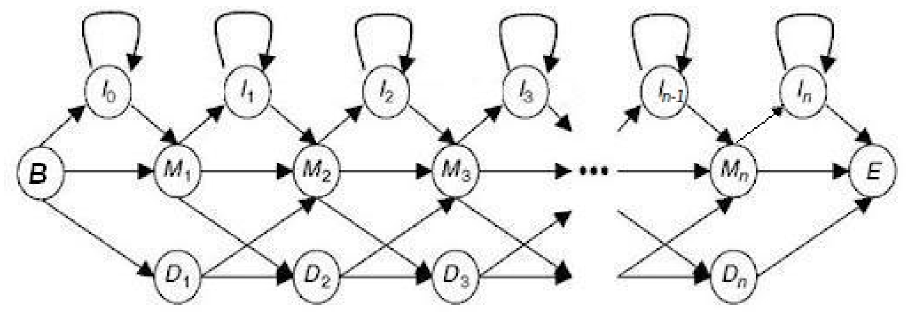


Figure 3.2: Profile HMM architecture (http://what-when-how.com/bioinformatics/ hidden-markov-models-and-neural-networks- bioinformatics/)

Any given emission sequence over the alphabet Σ can be represented as a possible path from left to right through the graph in Figure 3.2.

3.3.2 Deriving a Profile HMM from a MSA

Assume an optimal multiple alignment of m observed sequences $Y^{(1)}, \dots, Y^{(m)}$ with the underlying state sequences $X^{(1)}, \dots, X^{(m)}$, one wants to create a profile HMM on them [12].

The length of the model is set by the number of columns in the MSA that are assigned to match states (e.g. if for a given p more than p% of the entries in a column are no gap). For ungapped regions with only match states the fraction of each residue per position can be counted. Such an approach is called a position specific scoring matrix (PSSM) but does not take into account insertions and deletions.
To estimate the probability parameters for T and E, the number of transitions and emissions per position gives the following maximum likelihood (ML) estimators.
Set

- $\mathcal{T}'_{rs}$= number of transitions from r to s; $r, s \in S$
- and $\mathcal{E}'_{sq}$= number of emissions q when being in state s; $s \in S$, $q \in \Sigma$.

Although all $\mathcal{T}'_{rs}$ or $\mathcal{E}'_{sq}$ defined for $\mathcal{H}$ are theoretical possible, there could be some that are not observed because the training set is insufficient. It is therefore predefine that

- no transition probability of the allowed transitions will be set to zero (every state is possible at any position in the model)
- and no emission probability will be set to zero (every symbol could be observed anywhere).

This lack of information in the data can be corrected by adding predetermined pseudo counts $c > 0$. Getting:

$$\mathcal{T}^*_{rs} = \mathcal{T}'_{rs} + c_{rs} \quad \text{and} \quad \mathcal{E}^*_{sq} = \mathcal{E}'_{sq} + c_{sq}.$$

For example, all c_{rs} and c_{sq} could be set to a constant $c \in \mathbb{N}$ or set to $c = 1$.
From that follow the ML estimators for the transitions in T and the emissions in E:

$$t_{rs} = \frac{\mathcal{T}^*_{rs}}{\sum\limits_{s' \in S} \mathcal{T}^*_{rs'}} \quad \text{and} \quad e_{sq} = \frac{\mathcal{E}^*_{sq}}{\sum\limits_{q' \in \Sigma} \mathcal{E}^*_{sq'}} \quad . \tag{3.3.6}$$

3.4 HMM-Profiles in HMMER

For the classification of the HA gene sequences I apply the `HMMER 3.0` [5] tool for “biological sequence analysis using profile hidden Markov models [17]”. The tool enables to build HMM profiles (`hmmbuild`) and to scan sequences against a profile database (`hmmscan`). In the following section I will explain the `HMMER` architecture and how a sequence is scanned against a profile.

3.4.1 HMMER Architecture

The model implemented in `HMMER` is the “Plan7” architecture. So called from the seven possible transitions of matrix T [17] with

$$T = \begin{pmatrix} t_{MM} & t_{MI} & t_{MD} \\ t_{IM} & t_{II} & - \\ t_{DM} & - & t_{DD} \end{pmatrix} \quad .$$

The section of the model composed of the states M, I, D, B and E (see Figure 3.2) and this seven transitions form the main model. It controls every data dependent features; match states for modelling consensus positions in the alignment, insert and delete states for modelling insertions and deletions relative to consensus. Additionally, there are special states (S, N, C, T, J) that control the algorithm dependent

features of the model. This includes all special cases that can appear. The J state makes sure that the model can be reloaded in a loop from E to B through an arbitrary number of unaligned residues. This might happen if a special protein domain occurs repeatedly. In contrast, N and C allow for unaligned sequence data before and after the match, respectively. To complete the model, S and T are added as start and terminate position (Fig. 3.3) [18].
Summarized:

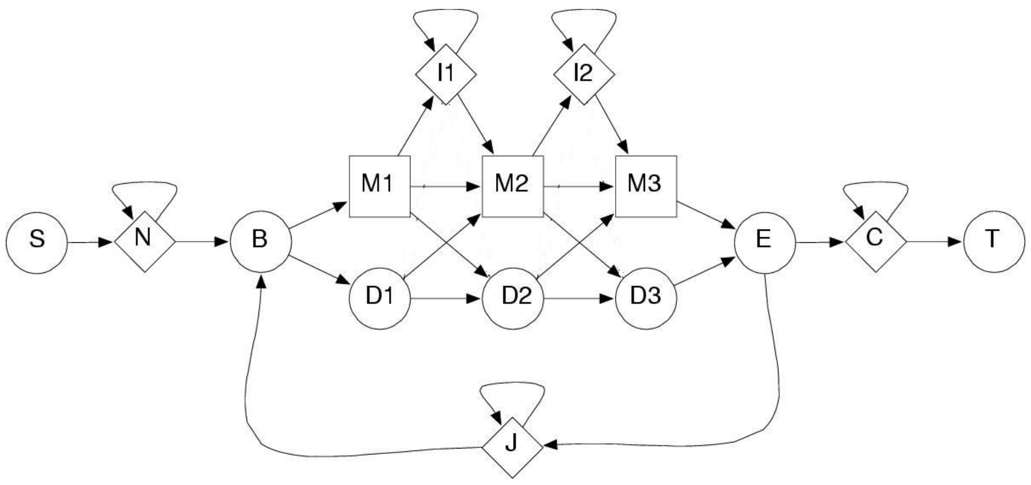


Figure 3.3: Plan7 model architecture (biomedcentral.com)

- M: Match state; has $|\Sigma|$ emission probabilities,
- I: Insert state; has $|\Sigma|$ emission probabilities,
- D: Delete state; non-emitter,
- S: Start position; non-emitter,
- N: N-terminal unaligned sequence state; emits on transition with $|\Sigma|$ emission probabilities,
- B: Begin state for entering the main model; non-emitter,
- E: End state for exiting the main model; non-emitter,
- C: C-terminal unaligned sequence state; emits on transition with $|\Sigma|$ emission probabilities,
- T: Terminate position; non-emitter,

- J: Joining segment unaligned sequence state; emits on transition with $|\Sigma|$ emission probabilities.

The HA sequences in my datasets of H5N1 and H1N1 are very closely related and especially within a subfamily cluster the average percentage pairwise nucleotide distance is $\leq 1.5\%$ (Definition 2.3.1). As a result, the multiple alignments of the clusters contain hardly any indels (compare section 5.1, Table 5.2). This is why they play no important part and can be ignored to simplify the model. In this case, there is only one transition left ($t_{MM} = 1$) and one emission distribution for each match state. The joint probability of X and Y can be calculated with equation (3.4.1):

$$\begin{aligned} P(X, Y \mid \mathcal{H}) &= t_{0X_1} \prod_{i=1}^{L} e_{X_i Y_i}\, t_{X_i X_{i+1}} \\ &= \prod_{i=1}^{L} e_{X_i Y_i} \quad . \end{aligned} \tag{3.4.1}$$

3.4.2 pHMMs in HMMER - hmmbuild

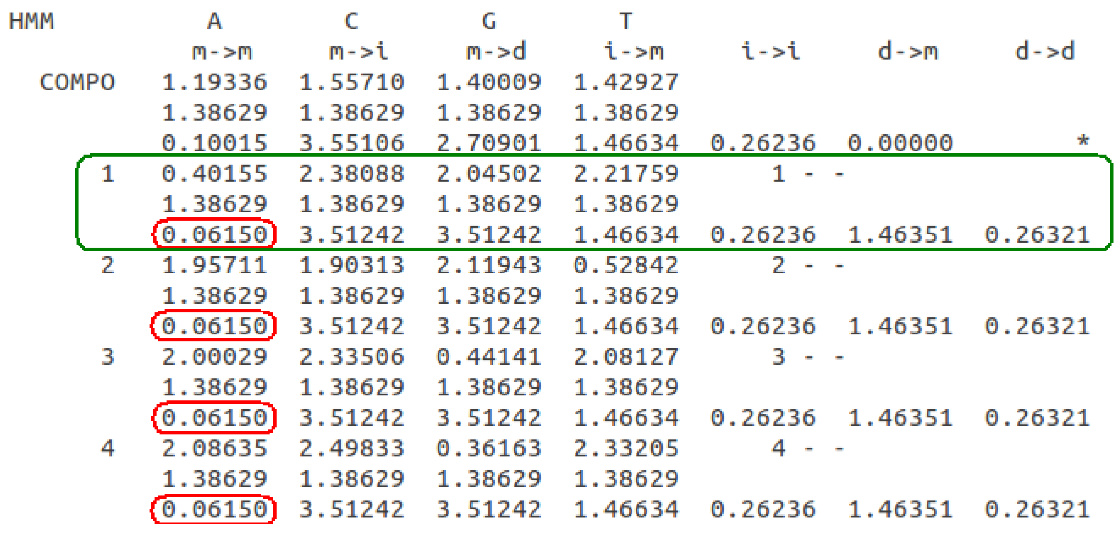
The program `hmmbuild` in `HMMER` is used to construct profiles from a MSA. The function call "`hmmbuild` *hmmfile inputmsa*" creates a pHMM of the aligned input sequences *inputmsa* as explained in section 3.3.2 and stores the profile in *hmmfile*.

HMMER Profile HMM File

The HMM output file consists of two different sections [17]. For one thing there is a header where the most important technical features are listed. And for another thing there is a mandatory main model section below that holds all parameters of the profile (Fig. 3.4 shows the main model section of a HMM file).

The model section is a table initiated by the word HMM. After that follows the description of the columns (the DNA alphabet A, C, G, T for the emission parameters and all possible transitions m→m, ..., d→d). For each consensus column of the input alignment there are three lines now (green frame), starting with an added begin node (COMPO) of the core model. The first line is the *match emission line* with the node number, the emission probabilities per symbol and annotations for this node. The next line is the *insert emission line* with one probability per symbol. Finally the *state transition line* shows the probabilities for the seven transitions $M_k \to M_{k+1}, I_k, D_{k+1}; I_k \to M_{k+1}, I_k; D_k \to M_{k+1}, D_{k+1}$ for node k. To avoid very small values when multiplying many probabilities, `HMMER` stores all probability parameters *par* of the HMM as the negative common logarithm:

$$par_{HMM} = -\log_{10} par\ .$$



```
HMM          A        C        G        T
           m->m     m->i     m->d     i->m     i->i     d->m     d->d
  COMPO   1.19336  1.55710  1.40009  1.42927
          1.38629  1.38629  1.38629  1.38629
          0.10015  3.55106  2.70901  1.46634  0.26236  0.00000        *
      1   0.40155  2.38088  2.04502  2.21759      1 - -
          1.38629  1.38629  1.38629  1.38629
          0.06150  3.51242  3.51242  1.46634  0.26236  1.46351  0.26321
      2   1.95711  1.90313  2.11943  0.52842      2 - -
          1.38629  1.38629  1.38629  1.38629
          0.06150  3.51242  3.51242  1.46634  0.26236  1.46351  0.26321
      3   2.00029  2.33506  0.44141  2.08127      3 - -
          1.38629  1.38629  1.38629  1.38629
          0.06150  3.51242  3.51242  1.46634  0.26236  1.46351  0.26321
      4   2.08635  2.49833  0.36163  2.33205      4 - -
          1.38629  1.38629  1.38629  1.38629
          0.06150  3.51242  3.51242  1.46634  0.26236  1.46351  0.26321
```

Figure 3.4: Example `HMMER` profile HMM file

So they can be summed up easily when the score is calculated (see section 3.4.3). In the HMM file in Figure 3.4 one can see that the state transition line has its highest probability with the $m \rightarrow m$ transition (red frame) and the values are all equal. As mentioned previously, indels play no important part with such closely related sequences. Therefore, the parameters of the match emission line make almost all the difference between the profiles of the clusters.

3.4.3 Classification with pHMMs - hmmscan

A typical application for profile HMMs is to detect a potential relationship between a profile of a sequence family and a single sequence. To obtain a significant hit, the matches need to be scored.

hmmscan

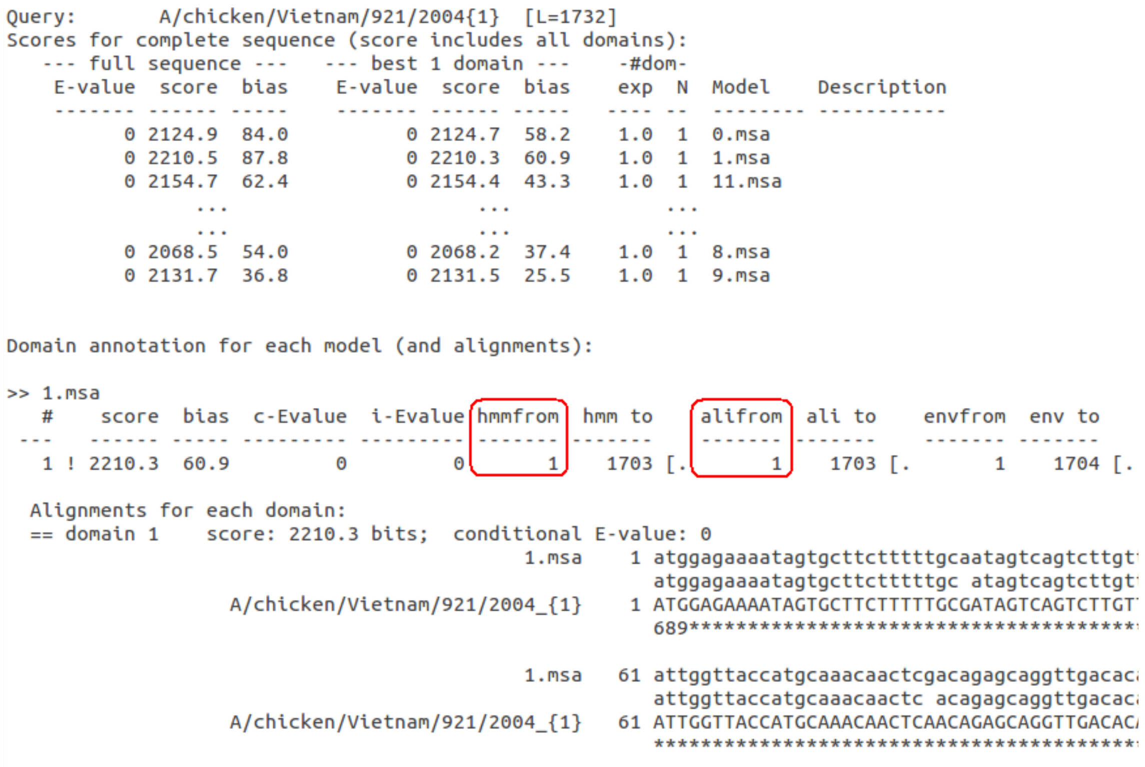
After building a profile database, `hmmscan` searches a sequence against it. This sequence is known as the query and the database as the target [17]. The output (Fig. 3.5) is a text file with a ranked list of scores against the profiles with the most significant matches to the query. Below, there are the multiple alignments of the sequence to each profile. The alignments also contain information about starting and ending points of the model (in Fig. 3.5 starting points hmmfrom: 1 and alifrom: 1; in the red frames).

Log-Odds Score

A bit score for each profile is the so called log-odds score; the logarithm of the ratio of the sequence probability according to the *null* model probability [17].

$$S_k(Y) = \log \frac{P(Y \mid \mathcal{H}_k)}{P(Y \mid null)} \qquad k = 1, \ldots, K \quad , \tag{3.4.2}$$

where k is the subtype (cluster) and Y the query sequence.



```
Query:       A/chicken/Vietnam/921/2004{1}  [L=1732]
Scores for complete sequence (score includes all domains):
   --- full sequence ---   --- best 1 domain ---    -#dom-
    E-value  score  bias    E-value  score  bias    exp  N  Model     Description
    ------- ------ -----    ------- ------ -----   ---- --  --------  -----------
          0 2124.9  84.0          0 2124.7  58.2    1.0  1  0.msa
          0 2210.5  87.8          0 2210.3  60.9    1.0  1  1.msa
          0 2154.7  62.4          0 2154.4  43.3    1.0  1  11.msa
             ...                     ...                ...
             ...                     ...                ...
          0 2068.5  54.0          0 2068.2  37.4    1.0  1  8.msa
          0 2131.7  36.8          0 2131.5  25.5    1.0  1  9.msa


Domain annotation for each model (and alignments):

>> 1.msa
   #    score  bias  c-Evalue  i-Evalue hmmfrom  hmm to    alifrom  ali to    envfrom  env to
 ---   ------ ----- --------- --------- ------- -------    ------- -------    ------- -------
   1 ! 2210.3  60.9         0         0       1    1703 [.       1    1703 [.       1    1704 [.

  Alignments for each domain:
  == domain 1    score: 2210.3 bits;  conditional E-value: 0
                                 1.msa    1 atggagaaaatagtgcttctttttgcaatagtcagtcttgt
                                            atggagaaaatagtgcttcttttgc atagtcagtcttgt
         A/chicken/Vietnam/921/2004_{1}    1 ATGGAGAAAATAGTGCTTCTTTTTGCGATAGTCAGTCTTGT
                                            689****************************************

                                 1.msa   61 attggttaccatgcaaacaactcgacagagcaggttgacac
                                            attggttaccatgcaaacaactc acagagcaggttgacac
         A/chicken/Vietnam/921/2004_{1}   61 ATTGGTTACCATGCAAACAACTCAACAGAGCAGGTTGACAC
                                            *****************************************
```

Figure 3.5: Example `hmmscan` output
The table with the scores and the alignment of the model and the test sequence below.

The probability $P(Y \mid \mathcal{H}_k)$ is computed using forward recursion (equation (3.2.5)) by summing over all state paths through the model k from begin (B) to end (E) that generate Y. The alignment of the sequence and the model is determined using the Viterbi recursion (equation (3.2.4)). It computes the most probable state path X^* in $\mathcal{H}$ to generate Y. Each symbol of the sequence is related to a match or insert state along the path.

Instead, the *null* model supports the non-homology hypothesis. It is a Markov chain with

$$S = \{G, F\} \quad \text{and}$$
$$T = (t_{rs})_{r,s \in \{G,F\}} = \begin{pmatrix} \frac{L}{L+1} & 1 - \frac{L}{L+1} \\ 0 & 1 \end{pmatrix} ,$$

where L is the model length. The G state generates a random sequence with an emission probability distribution of 0.25 for each nucleotide. To complete the model, a dummy end state F is added like the T state in the Plan7 architecture which emits no symbol [3].
When ignoring the indels, the only parameters important for calculating the score are the match emission line and the $m \to m$ transition. Assuming a fixed value for this transition, only the match emission values need to be considered.

$$P_k(Y_i, i) \; > 0 \qquad k = 1, \dots, K \;\; , \qquad Y_i \in \{A, C, G, T\} \; , \quad i = 1, \dots, L \;\; ,$$

is the probability to see Y_i when being on position i in model k.
In this case the score looks as follows:

$$\begin{aligned} S_k(Y) &= \log \frac{\prod_{i=1}^{L} P_k(Y_i, i)}{\prod_{i=1}^{L} P_{null}(Y_i, i)} \\ &= \sum_{i=1}^{L} \log \; P_k(Y_i, i) \; - \; \sum_{i=1}^{L} \log \; 0.25 \\ & k = 1, \dots, K \;\; , \quad Y_i \in \{A, C, G, T\} \; . \end{aligned} \tag{3.4.3}$$

Considering the properties of the log function, both sums are negative. And because the term for the *null* model is fixed, maximizing the score is equivalent to minimizing

$$\left| \sum_{i=1}^{L} \log \; P_k(Y_i, i) \right| \; . \tag{3.4.4}$$

This fact is important in the profile training in section 4.2.2 when correcting the HMM parameters.

3.5 Models of DNA Evolution

Even though an alignment provides information about evolutionary relationship between the sequences, it does not describe the substitution processes in DNA, RNA or amino acid sequences through time. In order to compute phylogenetic trees or to understand the evolutionary process, genetic distances need to be introduced. The simplest way is counting the number of different sites in an alignment. This observed distance, also called p-distance [15], is the relative frequency of nucleotide differences between two sequences

$$d(Y^{(1)}, Y^{(2)}) = \frac{\#\ \text{differences}}{|\text{alignment}|} \quad . \tag{3.5.1}$$

The p-distance is easy but not very useful in sequence analysis because it underestimates the true genetic distance and becomes saturated when the number of mutations is large [15]. For that reason a more descriptive approach is required.

A model of sequence evolution can be understood as a continuous time Markov chain. There are a lot of substitution models to analyse evolution. They can be very complex when considering more and more parameters and conditions [19]. To model sequence evolution, a substitution rate matrix Q must be specified:

$$\begin{aligned} &Q = (q_{ij})_{i,j \in \{A,C,G,T\}} \\ &\qquad \text{with } q_{ij}\text{: rate at which base } i \text{ goes to base } j, \\ &q_{ii} = -\sum_{\{j | j \neq i\}} q_{ij} \quad . \end{aligned} \tag{3.5.2}$$

The distribution vector of the base frequencies $\pi = (\pi_A, \pi_C, \pi_G, \pi_T)$ can be determined through

$$\pi = \lim_{t \to \infty} P(t) \ = \ \lim_{t \to \infty} e^{Qt} \quad .$$

From that rate matrix Q a transition matrix $P(t) = (p_{ij}(t))_{i,j \in \{A,C,G,T\}}$ can be calculated via matrix exponential (equation (3.5.3)) where $p_{ij}(t)$ is the probability to get from state i to j after time t [15]:

$$P(t) = e^{Qt} = \sum_{n=0}^{\infty} Q^n \frac{t^n}{n!} \quad . \tag{3.5.3}$$

3.5.1 JC69 Model (Jukes-Cantor 1969)

The simplest model of DNA sequence evolution is the Jukes-Cantor model, also known as the one-parameter model [19]. This is because it is assumed that on the one hand the base frequencies are equal ($\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$) and on the other hand that all substitutions are equal. This substitution rate per time t is the one parameter α ($\alpha \in \mathbb{R}, \alpha > 0$) [15] so that:

$$q_{ij} = \begin{cases} -\frac{3}{4}\,\alpha, & i = j \\ \\ \frac{1}{4}\,\alpha, & i \neq j \end{cases} \quad . \tag{3.5.4}$$

If the distances are determined, the associated transition probabilities in matrix P can be computed using equation (3.5.3):

$$P_{ij}(t) = \begin{cases} \frac{1}{4} + \frac{3}{4}e^{-\alpha t}, & i = j \\ \\ \frac{1}{4} - \frac{1}{4}e^{-\alpha t}, & i \neq j \end{cases} \quad . \tag{3.5.5}$$

The more time passes the greater is the probability that nucleotides had changed. In contrast, if t is near to zero, P_{ii}(t) is very close to 1 and P_{ij}(t) is very close to 0, for $j \neq i$. Finally, phylogenetic distances can be assigned. Considering a fixed site of the sequence, the probability of observing a mutation is

$$\sum_{j \neq i} P_{ij}(t) = \frac{3}{4}(1 - e^{-2\alpha t}) \quad ,$$

where $\alpha \cdot t$ is the time required to get from one sequence to another sequence in a phylogenetic tree.
Let p be the observed p-distance (equation (3.5.1)). One gets the genetic distance as the expected number of changes along the branch (branch length):

$$p = \frac{3}{4}(1 - e^{-2\alpha t}) \implies d = \alpha t = -\frac{1}{2}\,\ln\,(1 - \frac{4}{3}\,p) \ . \tag{3.5.6}$$

This model is simple but not very close to biological reality because it does not take into account that nucleotide frequencies as well as substitution rates can be very different [19].

3.5.2 K80 Model (Kimura 1980)

A better way is Kimura's two-parameter model [19] of 1980.

In genetics, transitions are prone to occur more frequently than transversions (see section 2.1) and so it is a good approach to distinguish between them and allow inequality of rates. The parameters of this two-parameter model are the transition and transversion rates α and β. All base frequencies stay the same as with the JC69 model; $\pi_A = \pi_C = \pi_G = \pi_T = \frac{1}{4}$. This is why the whole model can be summed up in Figure 3.6. Obviously, there is one possible transition at rate α and two different transversions at rate β for each nucleotide. This means a transition/transversion rate $R = \alpha/(2\beta)$ and thus a total rate of change $\alpha + 2\beta = 1$. The probabilities that the result turns out to be a transition or transversion in time t can be derived explicitly [19]:

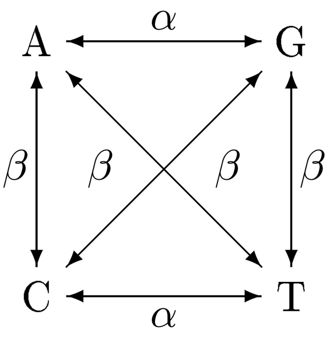

$$\begin{tikzcd}
A \arrow[r, leftrightarrow, "\alpha"] \arrow[d, leftrightarrow, "\beta"'] \arrow[dr, leftrightarrow, "\beta"] & G \arrow[d, leftrightarrow, "\beta"] \arrow[dl, leftrightarrow, "\beta"'] \\
C \arrow[r, leftrightarrow, "\alpha"'] & T
\end{tikzcd}$$

Figure 3.6:
K80 model
(www.biomedcentral.com)

$$P_{ij}(t) = \begin{cases} \frac{1}{4} - \frac{1}{2}\exp(-\frac{2R+1}{R+1}\,t) + \frac{1}{4}\exp(-\frac{-2}{R+1}\,t), & \textit{transition} \\ \\ \frac{1}{2} - \frac{1}{2}\exp(-\frac{2}{R+1}\,t), & \textit{transversion} \end{cases} . \qquad (3.5.7)$$

To calculate the K80 distance, a similar procedure to the JC69 model can be used. In that model it is distinguished between two observations when comparing two sequences; the proportion of sites with transitions p_1 and the proportion of sites with transversions p_2. The distance can be interpreted as the time that passed since the sequences got separated and it is determined with equation (3.5.8)

$$d = -\frac{1}{2}\ln(1 - 2p_1 - p_2) - \frac{1}{4}\ln(1 - 2p_2)\ . \qquad (3.5.8)$$

When building the clusters I take the K80 distance, like in the WHO study, to calculate distances that are used for the phylogenetic analysis.

There are lots of further models which assume four different base frequencies or distinguish between each single substitution rate; e.g. F81 (J. Felsenstein 1981) or GTR (general time reversible) model [19]. Nevertheless, using distances also has weaknesses. All these distance models treat each position in the sequence equally. But there might be positions that are strongly conserved and more important for the phylogenetic relationship than the variable ones. It could be a future task to use a better suitable model.

4 Methods

To implement a complete classification work-flow for the subfamily identification and classification of influenza A subtypes, a number of different methods had to be developed and applied. This includes methods for clustering the sequences as well as methods for the classification based on HMM profiles.

4.1 Cluster Methods

The aim of classification schemes is to organize a large dataset for better understanding and revealing information more efficiently. As a result, the data are summarized by a number of homogeneous groups of objects, called clusters. These clusters capture patterns of similarities and differences in the set [20]. Clustering is important these days when analysing the data of large databases (e.g. sequence databases).
A more formal definition: Clustering means to describe group structures in datasets.

Definition 4.1.1 (Cluster and Clustering [20]). *A cluster is defined as a set of objects cohesive in a way that objects within are more similar to each other than to outer objects. Given a dataset D, a clustering is to divide D into disjoint sets or clusters $\{D_1, \ldots, D_k\}$ such as*

$$D_i \cap D_j = \emptyset \ \ \forall i, j \in \{1, \ldots, k\}, \ i \neq j \quad \textit{and} \quad \bigcup_{i=1}^{k} D_i = D \ .$$

Before revealing a cluster structure for D, two aspects need to be specified; a criterion to measure the similarity of the data and a method to assign the objects to clusters [20].

Cluster methods can be divided into different groups, e.g. connectivity based, centroid, distribution or density models. The choice of the model depends on the properties of the data and the purpose of the analysis. An assignment criterion is often formulated in terms of similarity or dissimilarity between the objects, e.g. distance measures. In my datasets of H5N1 and H1N1 I have several thousands of closely related DNA sequences. Depending on the cluster definition (Def. 2.3.1) set by the *H5N1 Evolution Working Group*, I examine two different approaches. The first one is a hierarchical clustering considering all conditions in the definition. The second one is a faster medoid-based greedy-method that only takes the distance criterion into account.

4.1.1 Hierarchical Clustering

In hierarchical clustering, the data are represented as a tree to constitute the relation among the objects. On the basis of that tree there are different methods how clusters can be built [20]:

1. agglomerative clustering: starting from smaller clusters and gradually merging them into parental nodes and

2. divisive clustering: splitting greater clusters into smaller ones.

I use agglomerative clustering where a hierarchy is given through a constructed phylogenetic tree of the input sequences.

Bootstrapping

Building the hierarchy tree includes a bootstrap analysis which is a method to statistically estimate the reliability of some grouping, e.g. in a phylogenetic tree [19]. Assuming that one is given a tree T for a set of m sequences. Although the topology of the relationship can be deduced from T, the reliabilities of these clades need to be measured. For this purpose, sub-samples of the input set are taken in the following way.
Choose a sequence $s \in \{s^{(1)}, \dots, s^{(m)}\}$ at random and repeat this procedure m times (a sequence can be chosen several times). With this sequences a new tree T_{new} is built. After that, each clade of the original tree T is compared to T_{new}. If a clade is present in both trees, its score is increased by one (+1), otherwise not (+0), and a new bootstrap cycle is started. To get a convictive result, this procedure is repeated several times. Finally, every clade is assigned a value of reliability as the frequency of occurrence $\in (0\%, 100\%]$. It seems comprehensible that clades with a high value (e.g. 85%) are more reliable than those with a low value (e.g. 20%).

clusterboot.r

My clustering approach bases on the clade defining linkage criteria (Def. 2.3.1). A similarity criterion and a hierarchy are given through the matrix of the pairwise distances and the consensus tree.

The required distance matrix of the DNA sequences is computed using the K80 distance (section 3.5.2; [11]) and is needed to ensure that the average pairwise distance among the sequences within a clade is ≤1.5% [10]. The consensus tree is determined through 1,000 bootstrapping replicates as explained above. As defined by the *H5N1 Evolution Working Group*, I build a consensus tree where all clades exceed a threshold of 60%. The clusters are then built by starting from singleton clusters and sequentially merging them while the distance criterion is still satisfied.

Explanation

The algorithm `clusterboot.r`, written in the statistical language `R`, [21] works as follows. The pseudo-code with the steps of the algorithm is given in Alg. 4.1.1. As input a multiple alignment (MSA) of all sequences in the dataset is required. In step (I) the matrix and the tree are prepared. More exactly, the matrix D of the pairwise distances is computed (`R` function *dist.dna* from package ape [22]). Next comes the bootstrap analysis with 1,000 neighbour joining replicates to determine the consensus tree T_{con} (`R` functions *nj, boot.phylo, consensus* from package ape [22]) and after that the clusters are built (II). Using Figure 4.1 as example, the procedure works as follows. Every sequence (every leaf in the consensus tree T_{con}) forms an own cluster (a to f in Fig. 4.1(A)). The leaves that share a distinct common node are taken and it is examined if the distance criterion *crit* is satisfied (equation (4.1.1)). The criterion ensures that the pairwise average distance among the sequences of one cluster is ≤ 0.015. To avoid inhomogeneity, it is additionally required that the maximum distance between two sequences of the clusters that should be merged is ≤ 0.035.

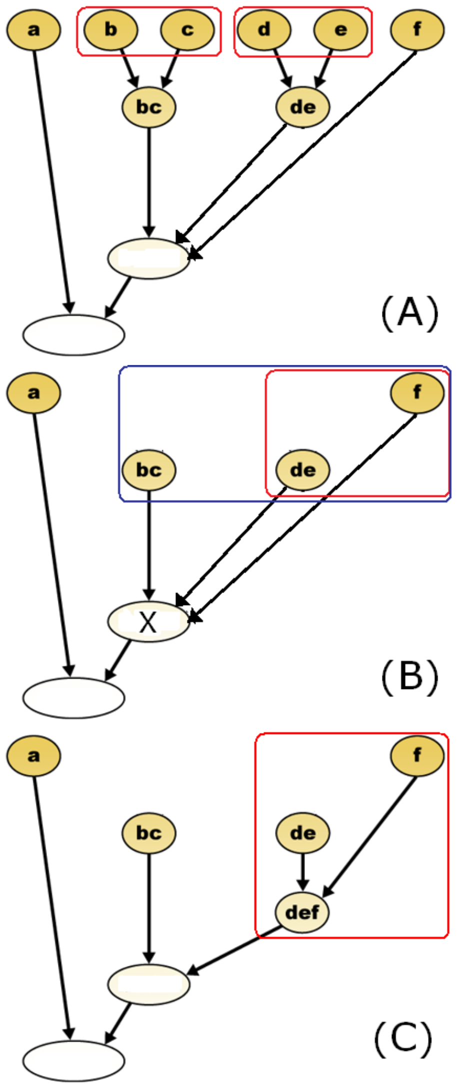


Figure 4.1: Hierarchical clustering (building the clusters)

Definition 4.1.2 (*crit*). *Set*

- Seq_i, Seq_j: *clusters,*
- $s_i \in Seq_i$ *and* $s_j \in Seq_j$,
- $|Seq_i|, |Seq_j|$: *number of sequences,*
- $d(s_i, s_j) \in D$: *distance of two sequences in distance matrix D.*

Then the criterion crit is defined as

$$crit = \begin{cases} \frac{1}{|Seq_i| \cdot |Seq_j|} \sum\limits_{s_i \in Seq_i; s_j \in Seq_j} d(s_i, s_j) \quad \leq 0.015 & \textit{and} \\ \max\limits_{s_i \in i; s_j \in j} d(s_i, s_j) \quad \leq 0.035 \quad . & \end{cases} \tag{4.1.1}$$

If *crit* is satisfied, the sequences are merged; (*bc*) and (*de*) in Figure 4.1(A) (red frames). The sequences are stored in their parental node and the leaves are deleted (Fig. 4.1(B)). After that, the algorithm continues working on the next level (from the leaves to the root of the tree). If a node has more than two leaves like node X in Figure 4.1(B) (blue frame), it could happen that all sequences together do not meet the distance criterion. In this case the criterion is only tested for two leaves. In Figure 4.1(B) it would be possible to merge (*de* and *f*), (*bc* and *f*) and (*bc* and *de*); here (*de* and *f*) are merged (red frame in Fig. 4.1(C)). A pair of leaves is only merged if *crit* is satisfied. Each step decreases the number of nodes in the tree until no further merging is possible under *crit* and every sequence belongs to a particular cluster.
As output one gets a table where each sequence is labelled with a cluster number. From that I take only those clusters with ≥ 4 sequences (set by the *H5N1 Evolution Working Group*). The other sequences are labelled as outliers.

The algorithm is very slow because it takes a long time to compute the bootstrap replicates and the consensus tree. Especially when the input MSA consists of several thousand sequences. For comparison, with the H5N1 dataset it requires nearly a weeks to determine the clusters. To avoid this long runtime of the algorithm, I try a faster greedy approach that uses only the matrix of the pairwise distances.

Algorithm 4.1.1: CLUSTERBOOT.R(MSA)

```
INPUT: MSA of all sequences
OUTPUT: clustering (table with the sequence names and cluster labels)

comment: prepare the distance matrix and the consensus tree
do { $D \leftarrow dist.dna\ (MSA)$                                          $(I)$
   { $T \leftarrow nj\ (D)$
   { $T_1, \ldots, T_{1000} \leftarrow boot.phylo\ (T)$
   { $T_{con} \leftarrow consensus\ (T_1 \ldots T_{1000}, p = 0.6)$

comment: build the clusters level by level
while $crit$ is satisfied for leaves $i$ and $j$                              $(II)$
  do { merge $i$ and $j$ into their parental node
     { delete $i$ and $j$
```

4.1.2 Medoid based Greedy-Clustering

With medoid based clustering the set D is separated into k non-overlapping clusters represented by k central sequences $(m_1^*, \ldots, m_k^*)$ of the dataset, called medoids [20]. In my application the medoids are defined as follows with $d(m_i^{(j)}, m_i^{(l)})$ as the pairwise distance of the sequences $m_i^{(j)}$ and $m_i^{(l)}$ from cluster i.

Definition 4.1.3 (Medoid [20])**.** *Given a cluster $D_i = \{m_i^{(1)}, m_i^{(2)}, \ldots\}$. An entity $m_i^* \in D_i$ will be referred to as its medoid if it minimizes the sum of distances to all other elements of D_i*

$$\sum_{m_i^{(j)} \in D_i} d(m_i^*, m_i^{(j)}) = \min_{m_i^{(l)} \in D_i} \sum_{m_i^{(j)} \in D_i} d(m_i^{(l)}, m_i^{(j)}) \ . \qquad (4.1.2)$$

A greedy heuristic means to make the current optimal choice in each step no matter whether it leads to the global optimum or not. In a greedy manner a particular sequence is taken and all neighbours within a specific distance radius are set as one cluster. After that, the next unassigned sequence is taken until all sequences belong to a cluster.

clustergreedy.r

The algorithm `clustergreedy.r` (Alg. 4.1.2) written in `R` [21] also takes the MSA of all sequences as input. The following steps are executed.

At first, a distance matrix D is computed and each sequence is referred to as a single cluster Cl_i (I). For each sequence $s' \in MSA$ the number of neighbour sequences inside a distance radius of 0.015 is determined with:

$$c_{s'} = |\{d(s', s) \leq 0.015\}|_{s \in MSA} \ .$$

The clusters (=sequences) are arranged in decreasing order starting with the sequence that has the most neighbours (II). After that, the greedy-clusters are built as follows (III). Starting from the sequence with the most neighbours, all singleton neighbour sequences ($dist \leq 0.015$) that have not been assigned to another cluster yet are merged into one cluster. When all sequences are assigned to a cluster the medoid sequences for each cluster are computed (Def. 4.1.3).

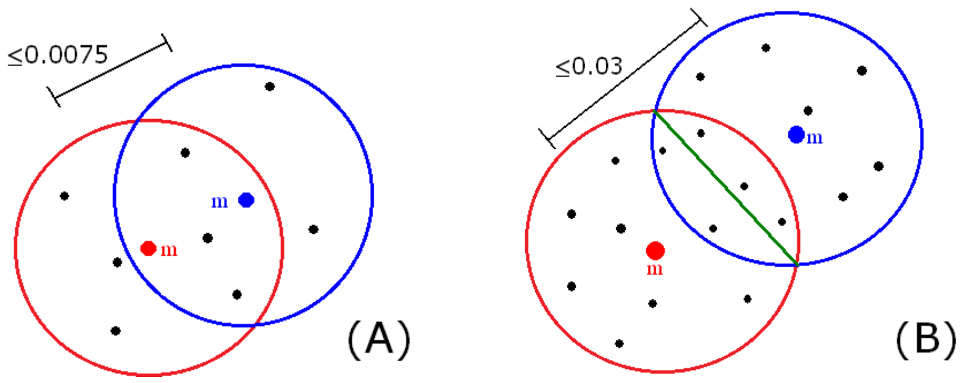


Figure 4.2: Adjustment cases in `clustergreedy.r`
A: cluster merging
B: nearest neighbour method

Based on this medoids, the clusters are adjusted. If two medoids m_i^*, m_j^* have a distance ≤ 0.0075, the clusters are joined (IV) and the medoid of this new cluster $Cl_i = Cl_i \cup Cl_j$ is recomputed. In Figure 4.2(A) the blue and red cluster would be merged. If two medoids m_i^*, m_j^* have a distance ≤ 0.03, there could be sequences $s \in \{Cl_i \cup Cl_j\}$ that meet the distance criterion to both clusters. These sequences are then assigned to their nearest medoid (V). The green line in Figure 4.2(B) separates these sequences into the blue and red cluster. From the output clustering table I take only those cluster with ≥ 4 sequences while the other sequences are labelled as outliers.

This algorithm only bases on the distances between the medoids. Depending on the distribution of the sequences the pairwise average distance within a cluster could be $> 1.5\%$. However, this is an approach trying to approximately reproduce the clustering in `clusterboot.r` considering easier and faster methods.

Algorithm 4.1.2: CLUSTERGREEDY.R(MSA)

```
INPUT: MSA of all sequences
OUTPUT: clustering (table with the sequence names and cluster labels)

comment: prepare the sequences
do { $D \leftarrow dist.dna(MSA)$
     $Cl_i \leftarrow$ each sequence $s'$                                   $(I)$
     for each $s' \in MSA$
     compute $c_{s'}$
     sort sequences in descending order regarding $c_{s'}$                  $(II)$

comment: compute clusters with greedy method
for each cluster $Cl_i$
  do { if $|Cl_i| = 1$                                                   $(III)$
         then merge it with all singleton
              clusters with distance $\leq 0.015$
  compute medoids $m_i^*$

comment: correct clusters
for each pair $(m_i^*, m_j^*), i \neq j$ of medoids
  do { if $d(m_i^*, m_j^*) \leq 0.0075$                                   (IV)
         then join the clusters
       recompute medoid
for each pair $(m_i^*, m_j^*), i \neq j$ of medoids
  do { if $d(m_i^*, m_j^*) \leq 0.03$                                     $(V)$
         then assign sequences $s$ to their nearest medoid $m_i^*$ or $m_j^*$
```

Both clustering algorithms are constructed in a way to fulfil the clade defining criteria and to approximate the original clusters of HPAIV best. It is not said that this is the method of choice to get meaningful clusters and maybe another approach based on the variance within a cluster would be more appropriate.

4.1.3 Comparing two Clusterings

Given two clusterings $C = \{C_1, \ldots, C_r\}$ and $\tilde{C} = \{\tilde{C}_1, \ldots, \tilde{C}_s\}$ over the same dataset $D = \{d_1, \ldots, d_n\}$. The Rand-index is a similarity measure to describe the co-clustering of pairs of objects [23]. For the $\frac{n \cdot (n-1)}{2}$ pairs of D one can define a 2×2 contingency table.

$C\|\tilde{C}$	# pairs in the same cluster	# pairs in different clusters
# pairs in the same cluster	a	b
# pairs in different clusters	c	d

Table 4.1: Contingency table for the Rand-index [23]
a and d are the agreements
b and c are the disagreements

Definition 4.1.4 (Rand-Index)**.** *The Rand-Index ($RI \in [0,1]$) is defined as*

$$RI = \frac{a+d}{a+b+c+d} \ , \tag{4.1.3}$$

- *0: no agreements between C and $\tilde{C}$,*
- *1: identical clusterings C and $\tilde{C}$.*

This measure can be used to evaluate the two clustering algorithms.

4.2 Classification Methods

4.2.1 Cross-Validation

To examine the performance of the classification model, a so called *cross-validation* can be applied. Given a dataset D it is divided into non overlapping training and test set. The training set is used to estimate the parameters of the model. With the test set the performance of that model is evaluated. One way to realise this is the *leave-one-out cross-validation.* As its name implies, in every step a single element of the dataset is removed and used as test set to validate the model afterwards while the other elements are used as training set [24].

In my application the aim is to determine the accuracy of the influenza A subfamily classification using hidden Markov profiles. The available dataset to test the model contains a number of clusters and within each cluster a different number of sequences. After building the HMM profiles, I examine the accuracy of my classification model with *leave-one-out cross-validation.*
For an arbitrary cluster c I have m_c pairs of a sequence s and the cluster cl it belongs to as reference:

$$\{(s^{(t)}, cl^{(t)})\}_{t=1}^{m_c} \ ,$$

where $cl^{(t)}$ is the cluster assignment made by the *H5N1 Evolution Working Group* (H5N1) (Table 5.1) or by a cluster algorithm (H1N1) (Table 6.1). This kind of assignment is called *supervised learning* where unlabelled sequences are classified using previously labelled sequences as a reference [25].

Explanation

The algorithm `HMMcrossval.pl` (Alg. 4.2.1) implemented in `perl`[1] requires a file with all possible clusters $c \in C$ as input.

Algorithm 4.2.1: HMMcrossval.pl(C)

```
INPUT: text file with all cluster names
OUTPUT: table with the classification results

comment: preparation
for each $c \in C$                                                    $(I)$
  do { $MSA_c \leftarrow$ MUSCLE $(Seq_c)$
       sub fasta2stockholm $(MSA_c)$

comment: loop through the clusters
for each $c_i \in C$
  do { $\mathcal{H}_c \leftarrow$ hmmbuild $(MSA_c)$,   $c \in C, c \neq c_i$      $(II)$

       comment: leave-one-out cross-validation
       for each $s_{c_i}^{(t)}$  $t = 1, \ldots, m$   $; m = |Seq_{c_i}|$        $(III)$
         do { $\mathcal{H}_{c_i} \leftarrow$ hmmbuild $(MSA_{c_i} \setminus \{s_{c_i}^{(t)}\})$
              hmmscan $(\mathcal{H}, s_{c_i}^{(t)})$                              $(IV)$
              $\tilde{c} \leftarrow \text{argmax}_{c \in C}$  $S_c(s_{c_i}^{(t)})$
              PRINT output $(s_{c_i}^{(t)}, c_i, \tilde{c})$
```

Before the cross-validation starts, a preparation step is performed (I). For every cluster $c \in C$ multiple alignments MSA_c are built. To get these alignments, I use a popular alignment tool, called `MUSCLE` [26]. It requires a sequence file in FASTA format and also gives the output MSA in the same format. But the problem is that the function `hmmbuild` in `HMMER` can only read STOCKHOLM files. Therefore, the subroutine `fasta2stockholm` converts the FASTA alignment into STOCKHOLM format. After that the clusters $c_i \in C$ are chosen one by one and the profiles $\mathcal{H}_c$ for ,$c \in C, c \neq c_i$, are built (II). Now a cross-validation is performed for the sequences of the clusters ($s_{c_i} \in Seq_{c_i}$).

[1]`perl` 5, version 12, subversion 4 (v5.12.4)

In every cross-validation step the sequences of c_i, $Seq_{c_i} = \{s_{c_i}^{(1)}, \dots, s_{c_i}^{(m)}\}$, are split into two complementary sets. A single sequence $s_{c_i}^{(t)}$ as test sequence and the remaining sequences, $MSA_{c_i} \setminus \{s_{c_i}^{(t)}\}$, as training set (III). The model parameters for $\mathcal{H}_{c_i}$ are estimated on the training set. Finally, the test sequence $s_{c_i}^{(t)}$ is scanned against the profile database $\mathcal{H} = \{\mathcal{H}_c\}_{c \in C}$ with the `HMMER` function `hmmscan` (IV). It is assigned to that model $\tilde{c}$ where it gets the highest score (compare equation (3.4.2)) with $\tilde{c}$ defined as follows:

$$\tilde{c} \in \operatorname{argmax}_{c \in C} \quad S_c(s_{c_i}^{(t)}) \,. \tag{4.2.1}$$

The output file is a table with the sequence name $s_{c_i}^{(t)}$, the reference cluster c_i and the classification result $\tilde{c}$ in each row.

The programming language `perl` is convenient for reading and manipulating text files (like the HMM profile and the `hmmscan` output). What takes the most time are the system calls for the other programs (`MUSCLE, HMMER`).

Runtime Analysis

Assume

- k = number of clusters in file C,
- $m_{all} = \sum\limits_{c \in C} m_c$ = total number of sequences in all cluster.

The algorithm runs the procedures

- `MUSCLE`: k times,
- `hmmbuild`: $k \cdot (k-1) + m_{all}$ times,
- `hmmscan`: m_{all} times.

Summarized: $k^2 + 2m_{all}$.

This number of calls can be reduced when several sequences are taken at one time and used as test set to evaluate the model. For example one sequence of each cluster.

4.2.2 Training the HMM Files

The accuracy of this predictive model depends on the parameters of the profiles $\mathcal{H}_c$. Whether or not they are usable to distinguish between the different models and allow a correct classification. The emission and transition probabilities for $\mathcal{H}_c$ are

estimated using the MSA of the cluster c. One can easily imagine that large clusters (e.g. with 100 sequences) provide more information to estimate the parameters than small clusters do (e.g. with 10 sequences). In addition, the added pseudo-counts (section 3.3.2) have a greater influence on small than on large clusters. This problem of estimating parameters on small datasets is well known. The aim of this training approach is to adjust the profiles to the sequences of the clusters individually and thus improve the classification accuracy.
I implement a modified cross-validation with an additional inner loop where the HMM training is performed.

The Training Algorithm - HMMtraining.pl

As with `HMMcrossval.pl` I have m_c sequences with known classification as a reference for an arbitrary cluster c:

$$\{(s^{(t)}, cl^{(t)})\}_{t=1}^{m_c} \; .$$

In `HMMcrossval.pl` the sequences are assigned to that profile where they get the highest score. The new algorithm `HMMtraining.pl` executes a profile training every time a classification is wrong. As input the algorithm requires the names of all possible clusters $c \in C$ in a text file. The pseudo-code of `HMMtraining.pl` is given in Alg. 4.2.2.

Algorithm 4.2.2: HMMtraining.pl(C)

```
INPUT: text file with all cluster names
OUTPUT: trained HMM database $\mathcal{H}^*$

comment: preparation
for each $c \in C$                                                    $(I)$
  do
    $MSA_c \leftarrow$ MUSCLE $(Seq_c)$
    sub fasta2stockholm $(MSA_c)$
    $\mathcal{H}_c \leftarrow$ hmmbuild $(MSA_c)$

comment: a loop through all clusters
for each $c_i \in C$                                                  $(II)$
  do
    for each $s_i^{(t)}$ , $t = 1, \dots, m$ ; $m = |Seq_{c_i}|$           $(III)$
      do
        $(var, \eta) \leftarrow$ sub hmmer $(s_i^{(t)})$                      $(IV)$

        comment: training (correct the $\mathcal{H}$ parameters)
        if $\eta > 0$                                                  $(V)$
          then
            $\mathcal{H}_{c_i} \leftarrow$ sub hmmcorrect $(\mathcal{H}_{c_i}, var, \eta)$
            $\mathcal{H}_{\tilde{c}} \leftarrow$ sub hmmcorrect $(\mathcal{H}_{\tilde{c}}, var, -\eta)$
```

Explanation

In the first section (I) a preparation step is performed. A multiple alignment of each cluster $c_i \in C$ is built by `MUSCLE` and these files are converted into STOCKHOLM format using the subroutine `fasta2stockholm`. After that, the HMM profiles , $\mathcal{H}_{c_i}, c_i \in C$, are built and the cross-validation can start.

In a loop the sequences of the current cluster $c_i \in C$ are used to train the profiles (II). The sequences $s_i^{(t)} \in Seq_{c_i} = \{s_i^{(1)}, \ldots, s_i^{(m)}\}$ are taken one after the other and stored as training sequence (III). For this sequence the subroutine `hmmer` is called (IV) where the function `hmmscan` (section 3.4.3) is executed. The sequence $s_i^{(t)}$ is scanned against all profiles $\mathcal{H}_c$, $c \in C$ and it is tested whether $s_i^{(t)}$ gets the highest score with $\mathcal{H}_{c_i}$ or not (compare equation (4.2.1)), i.e. whether

$$\text{argmax}_{c\in C} \;\; S_c(s_i^{(t)}) = c_i \; .$$

If not (that means $\tilde{c} \neq c_i$), the classification is wrong for the training sequence $s_i^{(t)}$ and the HMM parameters for the right and the wrong model (c_i and $\tilde{c}$) are adjusted. The following information, needed to adjust the HMM parameters, is retrieved from the `hmmscan` output (η and other variables var):

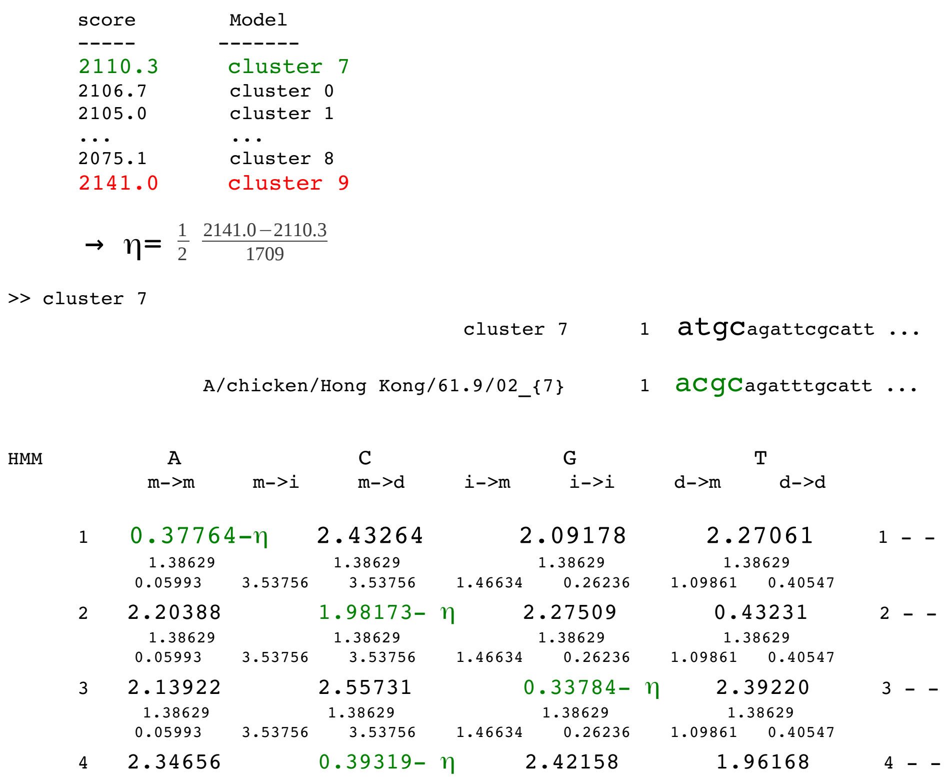
- η: This correction parameter is computed from three values:
  1. the score difference between the models $\mathcal{H}_{\tilde{c}}$ and $\mathcal{H}_{c_i}$
  $$S_{\tilde{c}}(s_i^{(t)}) - S_{c_i}(s_i^{(t)}) \; ,$$
  2. the length of the model $\mathcal{H}_{c_i} = |\mathcal{H}_{c_i}|$,
  3. the factor $\frac{1}{2}$ because there are two models (c_i and $\tilde{c}$) to adjust
  $$\eta = \frac{1}{2} \cdot \frac{S_{\tilde{c}}(s_i^{(t)}) \; - \; S_{c_i}(s_i^{(t)})}{|\mathcal{H}_{c_i}|} \; .$$

  This is the correction value for the profiles.
- $start_{c_i}$: Especially if $s_i^{(t)}$ is shorter than $|\mathcal{H}_{c_i}|$, it could happen that the assignment to $\mathcal{H}_{c_i}$ does not start at position one. So it is necessary to store the start position in order to have a starting point for the HMM parameter correction.
- The alignment of the consensus sequence for $\mathcal{H}_{c_i}$ and the sequence $s_i^{(t)}$:
  $$\mathcal{H}_{c_i} \| s_i^{(t)} \; .$$

  This is needed to change the right parameters in the HMM file.

- $\tilde{c}$: The name of the wrong assigned model for the correction as well as
- $start_{\tilde{c}}$ and $\mathcal{H}_{\tilde{c}} \| s_i^{(t)}$ for model $\tilde{c}$ (as for c_i).

In the case of a wrong classification the returned value for η is positive and a parameter correction for $\mathcal{H}_{c_i}$ and $\mathcal{H}_{\tilde{c}}$ is required (V). This is performed for each model individually in the subroutine `hmmcorrect` which takes the previously stored values (η, var) and the HMM file as input. Two corrections are done now. On the one hand the match emission parameters of $\mathcal{H}_{c_i}$ are decreased by η which means that the probabilities in the model are increased, and on the other hand the match emission parameters of $\mathcal{H}_{\tilde{c}}$ are increased. Figure 4.3 illustrates how to change the parameters of the $\mathcal{H}_{c_i}$ file (the right model).



```
score         Model
-----        -------
2110.3        cluster 7
2106.7        cluster 0
2105.0        cluster 1
...           ...
2075.1        cluster 8
2141.0        cluster 9
```

$$\rightarrow \eta = \frac{1}{2}\,\frac{2141.0-2110.3}{1709}$$

```
>> cluster 7
                                         cluster 7      1  atgcagattcgcatt ...

                         A/chicken/Hong Kong/61.9/02_{7}      1  acgcagatttgcatt ...

HMM         A                C                G                T
          m->m     m->i     m->d     i->m     i->i     d->m     d->d

      1  0.37764-η       2.43264          2.09178        2.27061       1 - -
          1.38629         1.38629          1.38629        1.38629
         0.05993   3.53756   3.53756   1.46634   0.26236   1.09861   0.40547
      2  2.20388         1.98173- η       2.27509        0.43231       2 - -
          1.38629         1.38629          1.38629        1.38629
         0.05993   3.53756   3.53756   1.46634   0.26236   1.09861   0.40547
      3  2.13922         2.55731          0.33784- η     2.39220       3 - -
          1.38629         1.38629          1.38629        1.38629
         0.05993   3.53756   3.53756   1.46634   0.26236   1.09861   0.40547
      4  2.34656         0.39319- η       2.42158        1.96168       4 - -
```

Figure 4.3: Correcting the parameters of $\mathcal{H}_{c_i}$

Computing η; the $\mathcal{H}_{c_i} \| s_i^{(t)}$ alignment and their associated parameters in the file (the green entries in the alignment and the match emission line).

When correcting the right profile $\mathcal{H}_{c_i}$, the algorithm starts at position $start_{c_i}$ (in Fig. 4.3 at position 1). The alignment $\mathcal{H}_{c_i} \| s_i^{(t)}$ is taken column by column (j=1, …) to assess whether there is a match between the model $\mathcal{H}_{c_i}$ and the sequence $s_{c_i}^{(t)}$ or not. If there is a match column j, the match emission value for the nucleotide n that is observed in the sequence, $s_i^{(t)}(j) = n$, is decreased by η (match state m, emission n)

$$e_{mn} - \eta \ .$$

The same procedure is done for the wrong assigned profile $\mathcal{H}_{\tilde{c}}$ but with the difference that the match emission values are decreased by $-\eta$ which means they are increased by η

$$e_{mn} + \eta \ .$$

As a result, the parameters of both HMM profiles are adjusted in a way that $s_i^{(t)}$ now gets a higher score to $\mathcal{H}_{c_i}$ and a lower score to $\mathcal{H}_{\tilde{c}}$.

$$\begin{gathered} S^*_{c_i}(s_i^{(t)}) > S_{c_i}(s_i^{(t)}) \\ \text{and} \\ S^*_{\tilde{c}}(s_i^{(t)}) < S_{\tilde{c}}(s_i^{(t)}) \ , \end{gathered} \tag{4.2.2}$$

with S^* as the score against the corrected model. After this procedure has been repeated for the sequences s of all clusters $c \in C$, one gets a trained HMM database $\mathcal{H}^* = \{\mathcal{H}^*_c\}_{c \in C}$ as output. This database can be applied in a classification model.

This way of profile training is applicable because the sequences are very closely related and thus almost exclusively match states are observed. So the insertion and transition parameters can be left out when correcting the HMMs. But the training method only describes one possibility to perform the parameter correction. There are many further ideas that can be taken into account:

- The correction parameter η can be damped by taking only a fraction of it to avoid large correction steps ($p \cdot \eta, p \in (0, 1]$). This p could be interpreted as learning rate.
- Whether a sequence s is assigned to the right model after HMM training strongly depends on the order in which the clusters for the profile training are chosen. The last profile for the training has the best chances to get enhanced while the parameter corrections for the first profiles could already have been cancelled out. To avoid this bias:
    1. The number of training iterations can be modified. For example, specifying the number of training iterations or performing the training until a termination criterion is satisfied.

2. Also the order to chose the clusters for the profile training can be set at random.

- Equal residues and substitutions in the alignment $\mathcal{H}_c \| s$ can be treated differently. For example, adding individual rates of η to the match emission values.

Evaluation of the HMM Training

With the algorithm `HMMtraining.pl` HMM profiles can be created that are trained on the input sequences. But to evaluate if the trained profile database $\mathcal{H}^*$ leads to a better or worse performance of the classification, the accuracy of this predictive model has to be determined. For that task an independent test set is required. This set is not included in the profile training and can be used to evaluate the HMM training. This is performed with a modified version of the `HMMtraining.pl` algorithm. This new evaluation algorithm `HMMtraineval.pl` requires the cluster names file C and the number of evaluation rounds r as input.

In every evaluation step $t = 1, \dots, r$ a test set of at most one sequence is removed from each cluster, $s_i^{(t)}$, $c_i \in C$ (I). A sequence that has been used as test sequence in a previous round cannot be chosen again (this applies to clusters with $< t$ sequences) (II). The remaining sequences, $Seq_{c_i} \setminus \{s_i^{(t)}\}$, $c_i \in C$, of each cluster form the training sets to build the alignments MSA_{c_i} and the profiles $\mathcal{H}_{c_i}$ and perform the HMM training from `HMMtraining.pl`. To bound the runtime of the training procedure, a maximum of 50 sequences of each cluster ($s_i^{(j)},\ j = 1, \dots, 50,\ s_i^{(j)} \neq s_i^{(t)}$) is used for the profile training (III). As in `HMMtraining.pl` the training sequences $s_i^{(j)}$ are scanned against the profiles and the HMMs are corrected if the classification for the j-th training sequence in cluster c_i is wrong (IV). After the HMMs have been trained on all clusters $c_i \in C$, the test sequences $s_i^{(t)}$, $c_i \in C$, are scanned against the corrected HMM database $\mathcal{H}^* = \{\mathcal{H}_c^*\}_{c \in C}$ (V) and the classification results are stored in the output file. For each test sequence the sequence name $s_i^{(t)}$, the original cluster c_i and the classification result after HMM training $\tilde{c}$ are stored.

Although the algorithm does not test the classification accuracy for all sequences in the cluster, it provides a good approximation of the performance of this profile training in `HMMtraining.pl` (Alg. 4.2.2). The training loop in `HMMtraineval.pl` (III-IV) is a slightly modified form of the algorithm `HMMtraining.pl` with the difference that the HMMs are trained on at most 50 sequences of each cluster.

Algorithm 4.2.3: HMMtraineval.pl(C, r)

```
INPUT: text file with all cluster names, number of evaluation rounds
OUTPUT: table with the classification results

comment: number of evaluation rounds
for each $t = 1, \dots, r$
do
    comment: remove test sequences and prepare HMM database
    for each $c_i \in C$
    do
        if $|Seq_{c_i}| > t$                                                    $(I)$
        then
            remove test sequence $s_i^{(t)}$
            $MSA_{c_i} \leftarrow$ MUSCLE $(Seq_{c_i} \setminus \{s_i^{(t)}\})$
            sub fasta2stockholm $(MSA_{c_i})$
            $\mathcal{H}_{c_i} \leftarrow$ hmmbuild $(MSA_{c_i})$
        else                                                                   $(II)$
        then
            $MSA_{c_i} \leftarrow$ MUSCLE $(Seq_{c_i})$
            sub fasta2stockholm $(MSA_{c_i})$
            $\mathcal{H}_{c_i} \leftarrow$ hmmbuild $(MSA_{c_i})$

    comment: HMM training
    for each $c_i \in C$
    do
        for each $s_i^{(j)}$, $j = 1, \dots, 50$, $s_i^{(j)} \neq s_i^{(t)}$                  $(III)$
        do
            $(var, \eta) \leftarrow$ sub hmmer $(s_i^{(j)})$

            comment: training (correct the $\mathcal{H}$ parameters)
            if $\eta > 0$                                                      $(IV)$
            then
                $\mathcal{H}_{c_i} \leftarrow$ sub hmmcorrect $(\mathcal{H}_{c_i}, var, \eta)$
                $\mathcal{H}_{\tilde{c}} \leftarrow$ sub hmmcorrect $(\mathcal{H}_{\tilde{c}}, var, -\eta)$

    comment: evaluation with the test sequences
    for each $c_i \in C$
    do
        hmmscan $(\mathcal{H}^*, s_i^{(t)})$                                         $(V)$
        $\tilde{c} \leftarrow \text{argmax}_{c \in C} \; S_c(s_i^{(t)})$
        PRINT output $(s_i^{(t)}, c_i, \tilde{c})$
```

4.2.3 ROC-Analysis

ROC is the abbreviation for *Receiver Operating Characteristic* and a method to evaluate the accuracy of a statistical test for a binary classifier system [24].

In my application I use ROC-analysis to find optimal thresholds (henceforth referred to as cutoffs) to support the HMM profile based classification. With the previous algorithm every sequence can be assigned to one of the models, even unrelated sequences from another virus or organism. For example, the sequence *A/swine/Costa Rica/000125-15/2010* of H1N1 can be scanned against the models of H5N1 and outputs a highest score against one of them (model 7 with score 706.2). But the scores for H5N1 HA sequences against the models are all in the region of 2,000. Therefore, a criterion is necessary to distinguish between sequences belonging to that subfamily and other sequences. The extended classification pipeline contains two steps:

1. A sequence s is scanned against the profile database and a model with highest score is determined,
$$\tilde{c} \leftarrow \text{argmax}_{c \in C} \;\; S_c(s) \;\; .$$
2. If the score exceeds the specific cutoff for the model,
$$S_{\tilde{c}}(s) \geq cut_{\tilde{c}} \;\; ,$$
then the sequence should be assigned to $\tilde{c}$.

The ROC analysis to identify these cutoffs includes the following steps [24]:

1. dichotomization,
2. 2×2 contingency table,
3. ROC-curve,
4. optimization.

During the cross-validation (without or with additional HMM training) every sequence is scanned against every profile (`HMMcrossval.pl` (III) and `HMMtraineval.pl` (V)). This table of scores can be obtained from the algorithms and be used as input file.

1. Dichotomization

In order to get a binary classifier I assign 0/1-values to each sequence of the input file.

The input is a table with tree columns; the original cluster of the sequence, the name of the profile it is scanned against and the score to that profile. Another column is attached with values of 1 and 0. 1 if the sequence is scanned against the original model and 0 otherwise, e.g. Table 4.2.

Sequence	Profile	Score	0/1
2.1.3	2.1.3	2198.7	1
2.1.3	2.1.2	2118.1	0
...	...	...	...

Table 4.2: ROC input table with GS

This prior knowledge of the true classification is referred to as *binary gold standard (GS)*.

2. 2×2 contingency table

For every pair of a sequence s and a model $\mathcal{H}_c$ there is score $S_c(s)$. Consequently, a 2×2 contingency table can be set for each cluster. According to a fixed cutoff *cut* the sequences can be divided into four groups. The sequences that belong to the cluster and exceed the cutoff *cut* (true positive) or not (false negative) and the sequences that do not belong to the cluster and exceed *cut* (false positive) or not (true negative). For such a *cut* the contingency table is shown below.

	GS (Positive)	GS (Negative)	Total
Test +	TP	FP	$\#\{score \geqslant cut\}$
Test -	FN	TN	$\#\{score < cut\}$
Total	n^1	n^0	$n = n^0 + n^1$

Table 4.3: ROC 2×2 contingency table [24]
TP=true positive, FP=false positive, FN=false negative, TN=true negative

To measure the accuracy of the classification, the sensitivity can be defined as the fraction of sequences that exceed *cut* among all sequences of that cluster:

$$Se = \frac{TP}{TP + FP} \; .$$

Furthermore, the specificity is the fraction of sequences that do not belong to the cluster among all sequences that do not exceed *cut*:

$$Sp = \frac{TN}{TN + FN} \; .$$

For an arbitrary cluster c^* I consider all possible values for $cut \in [a, b]$; choosing a as the lowest and b the highest score of a sequence $s \in Seq_{c^*}$ against the model c^*:

$$a = \min_s \ S_{c^*}(s) \quad , \quad b = \max_s \ S_{c^*}(s) \ .$$

I set up the contingency tables for all values *cut* between two consecutive scores in the sorted list L of scores against c^*:

$$L = \{a = S^1_{c^*}, S^2_{c^*}, \ldots, S^r_{c^*} = b\} \ , \qquad cut = \frac{S^i_{c^*} + S^{i-1}_{c^*}}{2} \qquad i = 2, \ldots, r \ . \tag{4.2.3}$$

For all contingency tables belonging to a specific cutoff *cut* I determine the sensitivity and specificity. These values are needed to draw the ROC-curve.

3. ROC-curve

The ROC-curve is a graphical plot of false-positive rate (1-specificity) on the x-axis versus sensitivity (true-positive rate) on the y-axis considering all possible values for the cutoff *cut* [24]. In a discrete case like mine one gets pairs (x, y) of FP-rate and sensitivity from the contingency tables for every chosen *cut* in equation (4.2.3) and thus a set of points in a graph. As can be seen in Figure 4.4, there are different outcomes of the ROC-curve to draw conclusion about the accuracy of the test.

- A curve near to the 45° line (yellow line) indicates a random process because the frequency of TP and FP is almost equal. This line is called line of equality.
- Values below the line of equality indicate a worse than random performance.
- An ideal result would be a curve that lies maximal above the line of equality, concurrent to the blue point P(0,1) $\rightarrow$ 100% correct classification.
- A satisfying result is given through the curve of black points for cluster 2.2.1.1 of H5N1 in Figure 4.4. These are the pairs (x, y) for each cutoff *cut*. With a high *cut* one gets a low FP-rate but also a low TP-rate. If *cut* is reduced, the TP-rate generally increases. But with some *cut* also the FP-rate increases.
- The optimum point considering the optimization criterion in equation (4.2.5) is the red point on the curve.

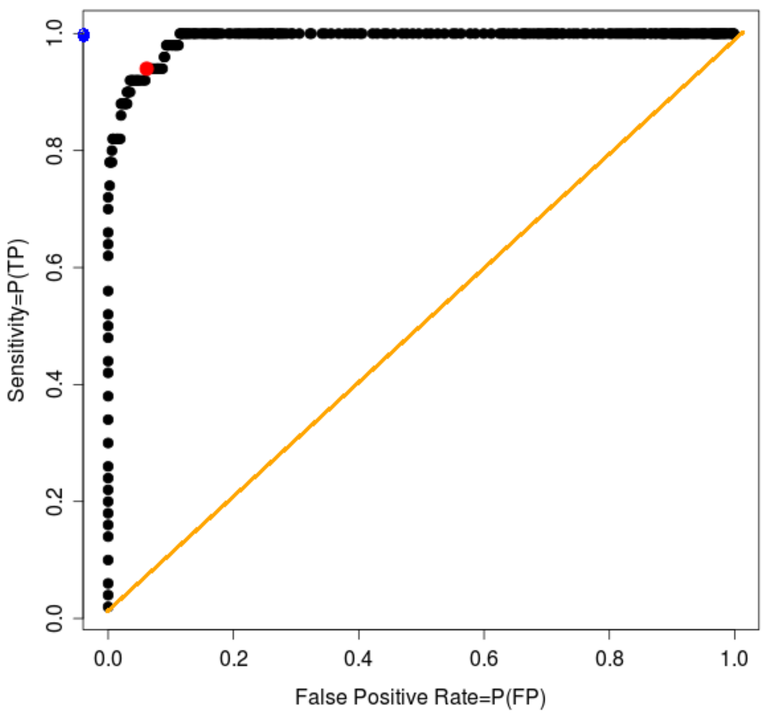


Figure 4.4: ROC-curve result: for cluster 2.2.1.1 (H5N1)

What is required is the cutoff value *cut** that explains the data best according to a defined optimization criterion.

4. Optimization

To calculate the optimum *cut** value for the classification model, an optimization criterion must be set first [24].
One possibility would be to determine the highest accuracy. This is the fraction of correct assignments in the contingency table. It is defined as follows:

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \ . \tag{4.2.4}$$

Another method, often applied in medical diagnostics, is to find the minimum distance of the ROC-curve and the point P(0,1):

$$(1 - \frac{TP}{TP + FN})^2 + (1 - \frac{TN}{TN + FP})^2 = (1 - Se)^2 + (1 - Sp)^2 \ . \tag{4.2.5}$$

Or cutoffs could be required that ensure a defined sensitivity or specificity of the test.

The choice which criterion to use depends on the aim in the application of the test. I need the cutoffs for the distinction of H5N1 from other influenza subtypes or new H5N1 clades. Because the classification should hold a high sensitivity I determined cutoffs that meet the condition of 99% sensitivity, at least:

$$cut^* = \max_{cut} \{Se(cut) \geq 0.99\} \ .$$

5 Application to HPAIV (H5N1)

To determine the performance of the methods introduced in chapter 4, I need a dataset which is suitable for that purpose. The classification for this set must be known so that it can be used as a reference to validate the test results. An available set are the HA gene sequences from the HPAIV (*highly pathogenic avian influenza virus*) of H5N1 (section 2.3) where a consistent nomenclature already exists. On the one hand these labels can be used to examine if the clustering algorithms reproduce the original clusters and on the other hand to evaluate the classification results.

5.1 Data Selection

The phylogenetic tree of all sequences with the associated cluster labels (attachment: 201101_h5fulltree.pdf [10]) was published in a WHO study of 2011 (section 2.3).

cluster label	**# sequences**	**cluster label**	**# sequences**
0	120	2.1.2	31
1	313	2.1.3	27
3	27	2.2.1	302
4	12	2.2.2	37
5	22	2.3.1	13
6	3	2.3.2	58
7	29	2.3.3	5
8	3	2.3.4	230
9	30	2.1.3.1	13
1.1	56	2.1.3.2	100
2.2	453	2.1.3.3	4
2.4	19	2.2.1.1	81
2.5	14	2.3.2.1	69
7.1	11	2.3.4.1	2
7.2	4	2.3.4.2	8
2.1.1	35	2.3.4.3	32

Table 5.1: Clusters of HPAIV (H5N1)
(32 clusters + 29 outliers $\Rightarrow$ 2,192 sequences)

The sequences themselves are listed in sequence databases. As data source for the

HA sequences I use the public NCBI Influenza Virus Resource [8]. In Table 5.1 the currently existing clusters are summarized. The second column includes the number of sequences in the respective cluster that were available at NCBI. Although the study of 2011 contains 2,947 sequences, only 2,192 of them were listed at NCBI (data collection: November 15th, 2011). These sequences are henceforth referred to as 2011H5N1 dataset I use for the analysis. To illustrate the relationship of the HPAIV clades, their phylogenetic tree is given in Figure 2.2 in section 2.3.

The HA sequences of Influenza H5N1 are very closely related because they represent different variants of the same gene for the same influenza subtype. But my clustering and classification methods are all based on genetic distances and sequence similarities. This point makes it more difficult to separate the sequences into distinct clusters that are suitable to receive good classification results with the HMM profiles.

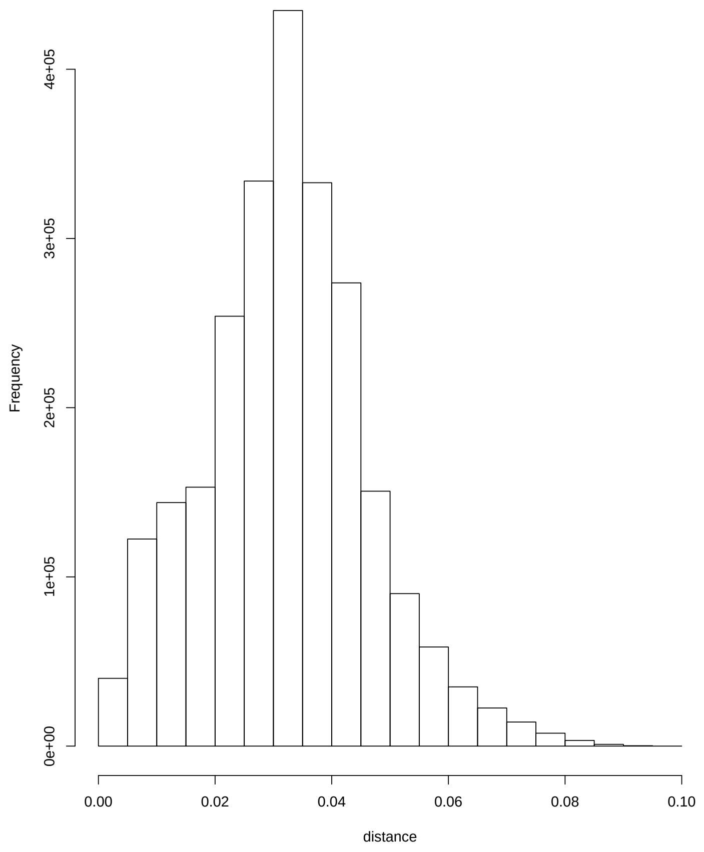


Figure 5.1: Histogram of the pairwise phylogenetic distances in the 2011H5N1 dataset [R package ape [22]], K80 distance

The histogram in Figure 5.1 illustrates the distribution of the pairwise distances among the sequences of my 2011H5N1 dataset. The minimum distance is 0 among sequences that belong to the same cluster where the average pairwise distance within is already ≤ 0.015. In contrast, the maximum distance is 0.0951 between a sequence from cluster 7.1 and one from 2.3.2.1. As was expected, these are different clusters that do not share a common ancestor clade except the root of the tree (common ancestor sequence; section 2.3). The average distance is at 0.0322. Considering that all sequences evolved from the common ancestor sequence, it is remarkable that within 15 years distances of nearly 0.1 occurred.

Another fact that is associated with the close relationship of the sequences is the small number of gaps in the multiple alignments of the clusters. As mentioned previously in section 3.4.1, the insert and delete parameters can be ignored in the profile training and only the match emission values are taken into account when correcting the HMMs.
To give a reason for this simplification, I compute the proportion of gaps for every cluster individually. The approach I use is to cut the ends of the alignments (deleting terminal alignment columns that contain gaps). These gaps occur through the varying sequence lengths in the cluster (from 1,600 to 1,780 nucleotides). One cannot distinguish between real indels or missing data through sequencing. Therefore, all terminal gaps are assumed to be incomplete sequence data. I only count the number of inner gaps that are interpreted to be results of insertions and deletions during evolution.
Table 5.2 shows the number and proportion of inner gaps per cluster. As expected, all clusters have less than 1% gaps and most clusters contain hardly any gaps. This serves as evidence to apply the simplified HMM training.

cluster	**# gaps**	**proportion**
0	341	0.0018
1	87	0.0002
3	0	0
4	0	0
5	0	0
6	0	0
7	88	0.0019
8	0	0
9	1	≈ 0
1.1	0	0
2.2	9	≈ 0
2.4	48	0.0015
2.5	31	0.0013
7.1	36	0.0019
7.2	3	0.0004
2.1.1	3	≈ 0
2.1.2	3	≈ 0
2.1.3	3	≈ 0
2.2.1	301	0.0006
2.2.2	0	0
2.3.1	144	0.0065
2.3.2	168	0.0018
2.3.3	0	0
2.3.4	681	0.0018
2.1.3.1	0	0
2.1.3.2	0	0
2.1.3.3	48	0.007
2.2.1.1	246	0.0019
2.3.2.1	174	0.0015
2.3.4.1	0	0
2.3.4.2	0	0
2.3.4.3	0	0

Table 5.2: Gap analysis for HPAIV (H5N1)

5.2 Cluster Analysis

The two algorithms `clusterboot.r` (Alg. 4.1.1) and `clustergreedy.r` (Alg. 4.1.2) are applied to the 2011H5N1 dataset. The objective of this algorithmic approach is to reproduce the reference clusters in Table 5.1.

5.2.1 clusterboot.r

The clustering of `clusterboot.r` identifies 31 clusters plus outlier sequences (summed up in cluster 32). To compare this outcome to the reference clustering, I set up a contingency table where each row represents an original cluster of HPAIV and each column a computed cluster (attachment: originalvsboot_H5N1.csv). This table shows how the sequences of the original cluster are sorted into groups through this algorithm. All rows and columns in this matrix are arranged in a way to maximize the sum of the entries on the main diagonal. The applied method is to iteratively take the maximum of the remaining matrix and put it on the main diagonal. In Figure 5.2 this matrix is illustrated as a levelplot where the colour intensity increases with the number of sequences in a matrix entry. Ideally, there would be a one-to-one relation between the original and the new clusters. This would result in a square matrix with just one entry per row or column on the main diagonal.

As can be seen in Figure 5.2, this nearly holds true for eight clusters: 2.4, 7.1, 2.1.3, 2.3.2, 2.3.4, 2.3.2.1, 2.3.4.3 and 5. Nevertheless, many clusters have been subdivided into sub-clusters or merged with other sequences.

1. Clusters that are split into sub-clusters of similar size:
   - 2.2, 1, 2.3.4, 0, 4, 2.2.1.1, 1.1, 9, 7, 2.1.1.
2. Clusters that share a common ancestor clade and are merged:
   - 2.2 & 2.2.1,
   - 2.3.4 & 2.3.4.2 & 2.3.3 & 2.3.1,
   - 2.1.3.1 & 2.1.3.2 & 2.1.3.3,
   - 2.2 & 2.2.2,
   - 2.1.1 & 2.1.2.
3. But also the first order clades 1 & 6 (& outliers) as well as 3 & 8 & 9 are merged although they do not belong to the same clade.

To evaluate this clustering approach and measure the similarity of both clusterings, a number of criteria can be used. Assuming $C = \{c_{ij}\}_{i=1,\dots,n\,;\;j=1,\dots,m}$ as the contingency table of the two clusterings with n rows (reference clusters) and m columns (boot clusters). The average number of entries per row and column can be calculated. Set $z := |\{c_{ij} \neq 0\}_{i=1,\ldots,n\,;\; j=1,\ldots,m}|$ as the number of matrix entries in C that are unequal zero. Then

$$\begin{aligned} \text{average per row} &= \frac{1}{n} \cdot z \quad \text{and} \\ \text{average per column} &= \frac{1}{m} \cdot z \;. \end{aligned} \tag{5.2.1}$$

For `clusterboot.r` one gets 1.88 per row and 1.94 per column respectively. The fraction of entries on the main diagonal is 69%. As an indicator for the accuracy between this two clusterings the Rand-index (section 4.1.3) is calculated as 0.6471.

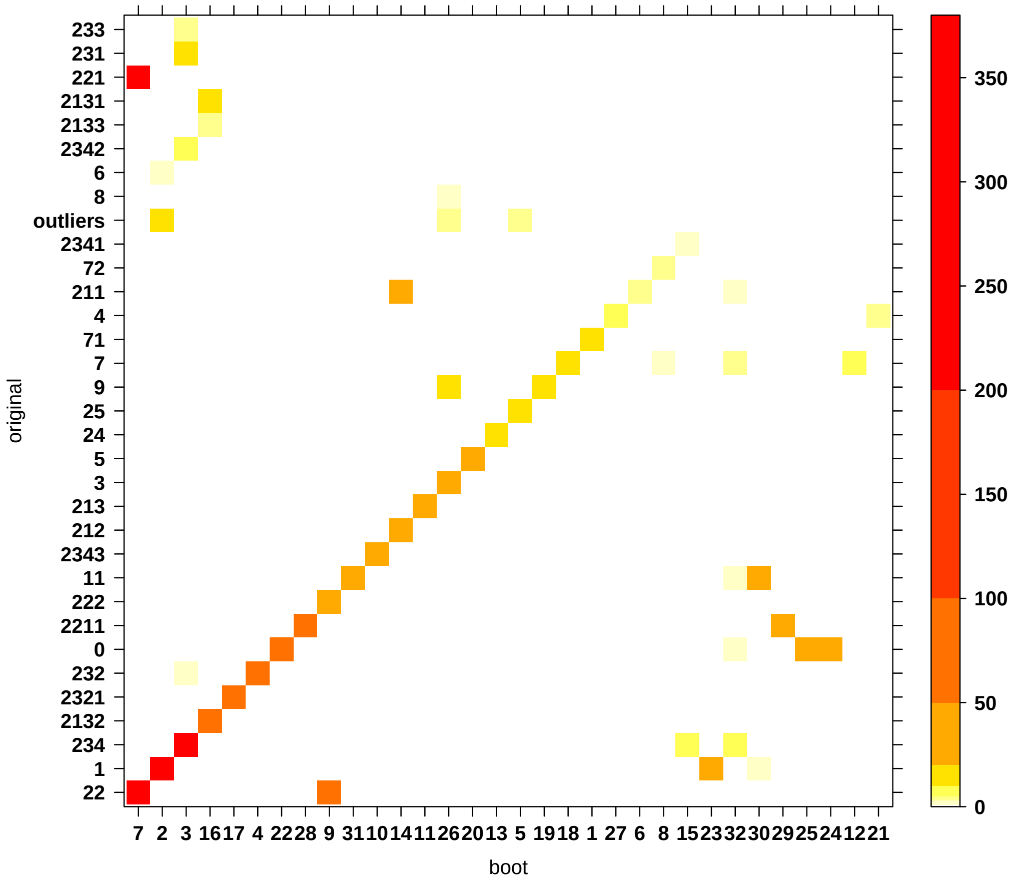


Figure 5.2: Levelplot of original versus boot clusters (H5N1)

5.2.2 clustergreedy.r

With the second algorithm `clustergreedy.r` the sequences are assigned differently. 30 clusters plus outliers (summed up in cluster 31) are detected. This clustering can also be compared to the reference shown in a contingency table (attachment: originalvsgreedy_H5N1.csv). As for the previous clustering, this table is illustrated as a levelplot with one row for each original cluster and one column for each cluster assigned through the greedy algorithm (Figure 5.3).

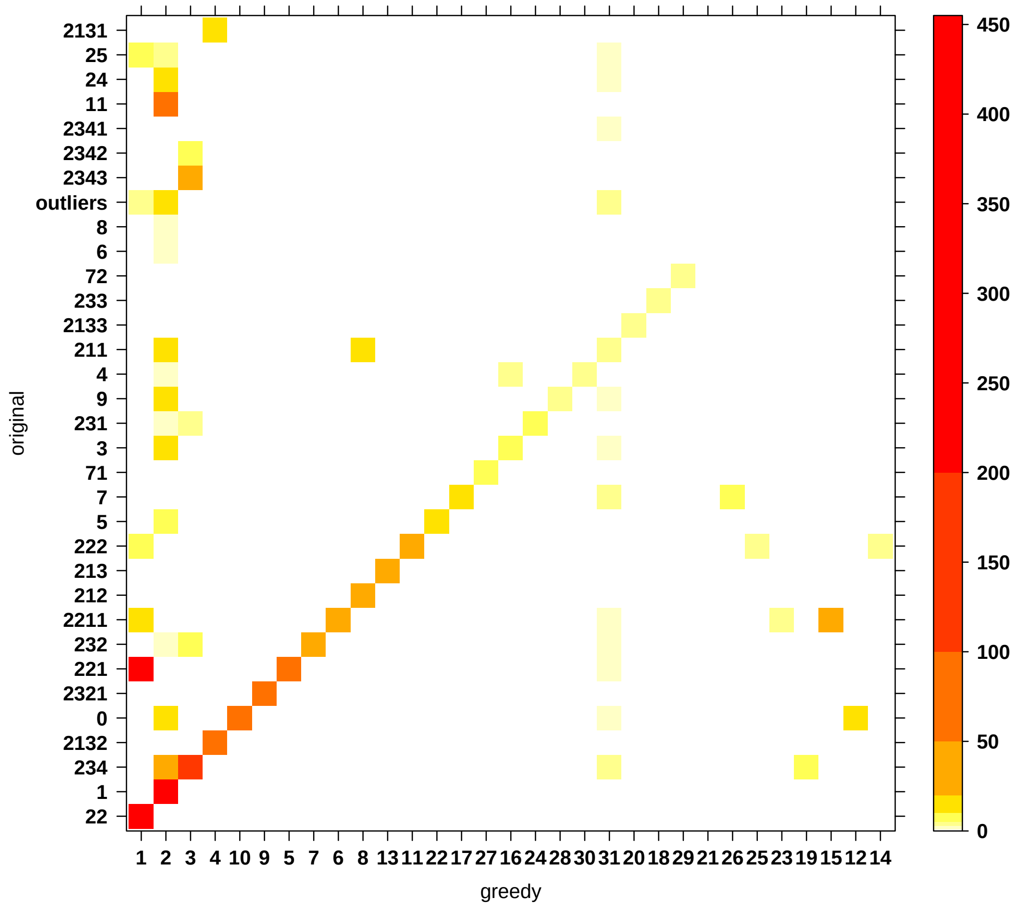


Figure 5.3: Levelplot of original versus greedy clusters (H5N1)

The original clusters that are clearly identified (with one entry per row and column in Figure 5.3) are 7.1, 7.2, 2.1.3, 2.3.3, 2.1.3.3 and 2.3.2.1. The greedy method also splits or merges sequences of different HPAIV clusters but worse than `clusterboot.r` does. Especially the greedy clusters 1, 2 and 31 consist of many sequences with different reference cluster labels.

- Greedy cluster 1 contains sequences of 2.2, 2.5, 2.2.1, 2.2.2 and 2.2.1.1 which are second, third and fourth order clades mixed.
- Greedy cluster 2 contains sequences of 0, 1.1, 2.4, 2.5, 2.1.1, 1, 2.3.4, 3, 5, 6, 8 and 9. This is not just a mixture of first, second and third order clades but also a mixture of different first order clades that do not share a common ancestor except the ancestor sequences of the whole dataset.
- Greedy cluster 31 contains the sequences that are labelled as outliers. But most of these sequences belong to various reference clusters of HPAIV.

Comparing the greedy clusters to the reference, one gets the following values. The average number of entries in each row and column is 2.33 and 2.52, respectively, using the equations (5.2.1). A proportion of 68% of the sequences are on the main diagonal. When the clusterings of the original clusters and the greedy results are compared the Rand-index is 0.5512.

5.2.3 Comparison

The two algorithms use different approaches to cluster the same set of sequences; a more complex one based on a tree topology and a faster greedy method. To evaluate which algorithm is more suitable to reproduce the reference clustering, they can be compared. The criteria are summed up in Table 5.3 below.

	clusterboot	**clustergreedy**
clearly identified clusters	8	6
aver. entries per row	1.88	2.33
aver. entries per column	1.94	2.52
sequences on the main diagonal	69%	68%
Rand-index (RI)	0.6471	0.5512

Table 5.3: Comparison: `clusterboot.r` - `clustergreedy.r` (H5N1)

No matter which criterion is chosen, the algorithm `clusterboot.r` comes closer to the reference clustering. The average number of entries per row and column show that the original clusters are detected more successfully with `clusterboot.r` than with `clustergreedy.r`. Another argument is the higher Rand-index that indicates a better agreement between reference and boot clusters. However, the differences in the performance of the two methods are not as distinct as expected. Although the greedy method does not take into account the criterion of a common ancestor node in a phylogenetic tree, it is not much worse. Compared with each other, the Rand-index of both clusterings is 0.6312 which shows how different the results of both approaches are in fact when clustering the same dataset. It must be regarded, however, that the bootstrapping analysis in `clusterboot.r` requires a long time. With the 2011H5N1 dataset it lasts nearly a week while `clustergreedy.r` needs just one hour to compute the clusters.

Discussion

There are some possible reasons for these clear differences between the reference clusters and the computed ones:

- Because the raw data of the study and my 2011H5N1 dataset are not identical (755 sequences are missing in my set), it is nearly impossible to compute exactly the same clusters. I get another consensus tree after the 1,000 bootstrapping replicates and this causes different assignments through the algorithm `clusterboot.r`.
- Some clusters were corrected manually by the *H5N1 Evolution Working Group*. At cluster 7, for example, the sequence *Ck/Shanxi/2/2006* was added which causes a higher average distance (>1.5%) within the cluster [11]. So those clusters do not satisfy the distance criterion any more and cannot be reproduced with my algorithmic approaches.
- The authors of the studies in 2008 and 2011 use a lot of different programs (ClustalW, Mega) and methods (neighbour-joining, maximum-likelihood, Bayesian) to determine the multiple alignment and the consensus tree. Whereas I just use the functions for sequence analysis implemented in `R` [21] and the criteria in the cluster definition (Def. 2.3.1) to compute the clustering.
- The greedy method does not take into account the phylogeny of the sequences in order to save computational time. But as a result, information gets lost to determine the correct clusters.
- The cluster assignment of the *H5N1 Evolution Working Group* is not necessarily correct. Maybe there are already mistakes in the sequence labelling.

5.3 Classification with Markov Profiles

For each subfamily cluster of HPAIV H5N1 a HMM profile is built. As can be seen in Table 5.1, there are 32 clusters of first, second, third and fourth order clades. But my classification algorithms (chapter 4) can only handle clusters with ≥ 3 sequences. When splitting a cluster $\{s^{(1)}, s^{(2)}\}$ of two sequences into test $\{s^{(1)}\}$ and training set $\{s^{(2)}\}$ no MSA can be built except the trivial one on just a single sequence. This is why the `hmmbuild` function in `HMMER` cannot build a profile for that cluster. Therefore, cluster 2.3.4.1 has to be excluded from the analysis and the classification methods can only be tested for 31 clusters. These 31 profiles define the HMM database to scan sequences against for the purpose of classification. I apply two classification methods, test their performance and determine the prediction error; one without and one with additional profile training. The results are written down in contingency tables (attachment: HMMcrossval_H5N1.csv, HMMcrossval-cut_H5N1.csv and HMMtrainevalcut_H5N1.csv). To evaluate if a classification is right, I compare the results to the cluster assignments made by the *H5N1 Evolution Working Group*. These assigned labels are taken as reference.

5.3.1 Without additional HMM Training

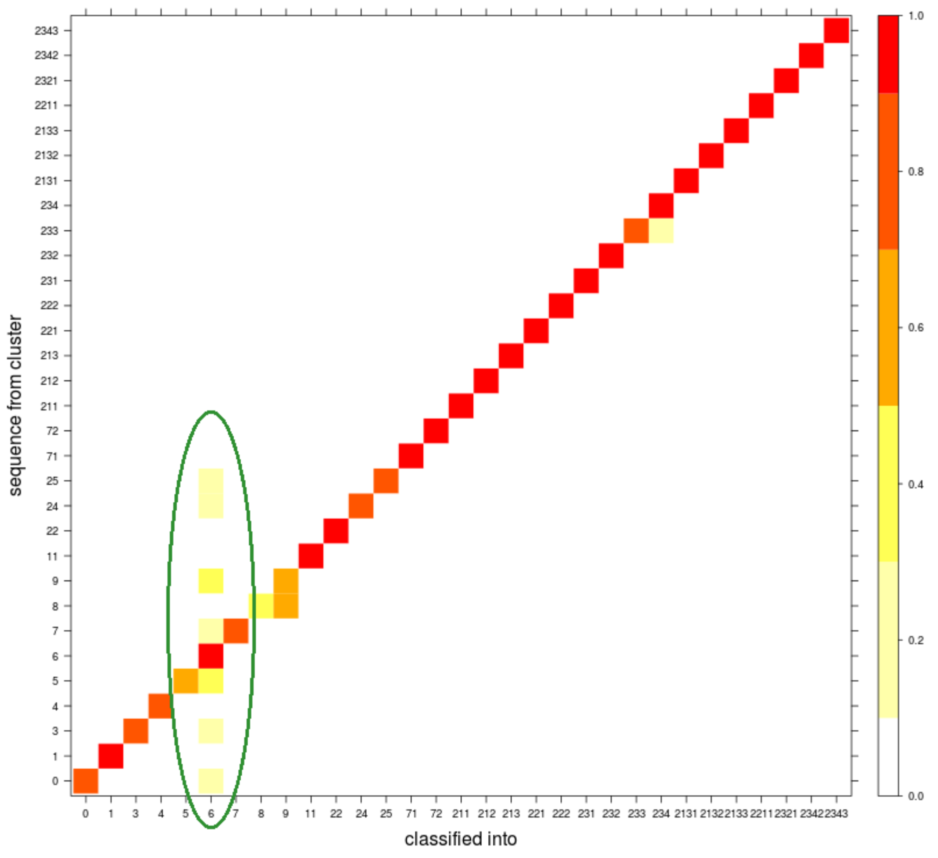


Figure 5.4: Levelplot for the `HMMcrossval.pl` results (H5N1)

The Algorithm 4.2.1 (`HMMcrossval.pl`) is used to classify the sequences with the standard profile database. This is the simple cross-validation without training where each sequence of the 31 clusters (in total 2,161 sequences) is scanned against the profiles and assigned to that model with the highest score. The levelplot in Figure 5.4 is the graphical representation of the classification results (HMMcrossval_H5N1.csv). It shows where the sequences of the reference clusters (the row labels on the y-axis) are assigned to (labels on the x-axis). Each coloured entry represents a percentage of the cluster. Ideally, there would just be entries on the main diagonal which would indicate a 100% correct classification. The clusters where this occurs are 6, 1.1, 7.2, 2.1.3, 2.2.1, 2.2.2, 2.1.3.1, 2.1.3.2, 2.1.3.3, 2.2.1.1, 2.3.2.1, 2.3.4.2 and 2.3.4.3.
In total the classification results in 2,045 right and 116 wrong assignments which is an accuracy of 94.6%. This performance is satisfying for a predictive model but the number of wrong classifications in each cluster varies in size from zero up to more than 50% of the sequences (e.g. cluster 8 with 2 wrong classifications out of 3).

Discussion

Especially in some smaller clusters like 3, 5, 7, 8, 9, 2.4 and 2.5 the classification seems to be more difficult. Nonetheless, bigger clusters (0 and 2.2) are prone to wrong classifications, too. For the smaller clusters the reason can be that the input MSA for `hmmbuild` contains too little information to build a meaningful profile. This is the problem of insufficient training data.

Another analytic approach is to have a look at the misclassified sequences and where they are assigned to. It is noticeable that 69 of the 116 wrong classifications are to cluster 6 which contains only three sequences itself. This feature is highlighted in Figure 5.4 with a green frame.

To give an explanation, one can take a closer look at the HMM profiles. Because the MSA of the three sequences in cluster 6 shows a high rate of conservation (more than 98% conserved nucleotides), the match emission probabilities for those nucleotides in the HMM profile are very high. As a result, sequences belonging to clusters that are closely related to cluster 6 could be misclassified easier. This applies especially for other first order clades shown in the table besides.

cluster	**assigned to 6**
0	26
1	8
3	5
5	7
9	12

Another more technical reason that can be taken into account is associated with the profile length. The HMMs are built using the MSAs of the clusters. However, these alignments vary in length depending on the length of the sequences in each respective cluster. For example, the profile of cluster 6 has length 1,779 while that of cluster 8 only has length 1,647 (the average length of the other profiles is 1,706). For profile 6 the score is calculated over more positions and additionally the emission probabilities in the HMM are very high. In this case the score to profile 6 can exceed the score to the original model when a sequence is scanned against the profile database. This is also why especially long sequences are more likely to being classified to cluster 6. The five sequences of cluster 3 that are assigned to cluster 6, for example, have a length from 1,749 to 1,780. In order to verify my assumption that the inconsistent HMM lengths are one of the main error sources, I construct a test example for cluster 0.

Example 5.3.1 (Error source: varying HMM lengths).
I take the profile of cluster 0 which has a length of 1,704 and the sequence A/goose/China/F3/2004 (HA) from cluster 0 with a length of 1,709 (>1,704) as query. To get a HMM database for classification, I edit the profile while sequentially removing a position from the end of the file. As a result I have 8 profiles (hmm0-0 , ..., hmm0-7) from length 1,704 to 1,697 the query can be scanned against. As expected, the scores are decreasing with decreasing profile length.

model	**score**
hmm0-0	2158.3
hmm0-1	2157.0
hmm0-2	2154.8
hmm0-3	2153.9
hmm0-4	2152.2
hmm0-5	2151.8
hmm0-6	2150.5
hmm0-7	2149.1

This example is a simplification. In my analysis the probabilities of the profiles are not equal but very similar because of the close relationship of the sequences. Furthermore, the models vary in length up to 132 positions and not just seven. But it seems very likely that long sequences can be misclassified to the longer model of cluster 6. A further reason for the misclassifications can be the editing by the biologists after constructing the clusters (section 5.2.3), e.g. in clusters 0 and 7. But also sequencing errors or errors in the input alignments which have an impact on the HMM profiles cannot be disregarded either.

New Approach

To avoid this error source of inconsistent model lengths, I consider the following solution.

For each cluster I take the MSA that is used to build the HMM profile and edit it. I cut off all alignment columns before the start codon ATG. This is applicable because all sequences represent a gene and therefore should possess this codon. Beginning with that start position, I take the subsequent 1,698 columns if existing because this is the general length of the HA gene (see Table 2.1). Consequently, no input

alignment for `hmmbuild` is longer than 1,698 positions.
My ideas behind this correction of HMM lengths are:

- to get HMMs of an equal length,
- to use just the DNA section for the classification that belongs to the HA gene (ATG → 1,698 nucleotides) and
- to exclude alignment errors especially in the terminal alignments columns from the analysis.

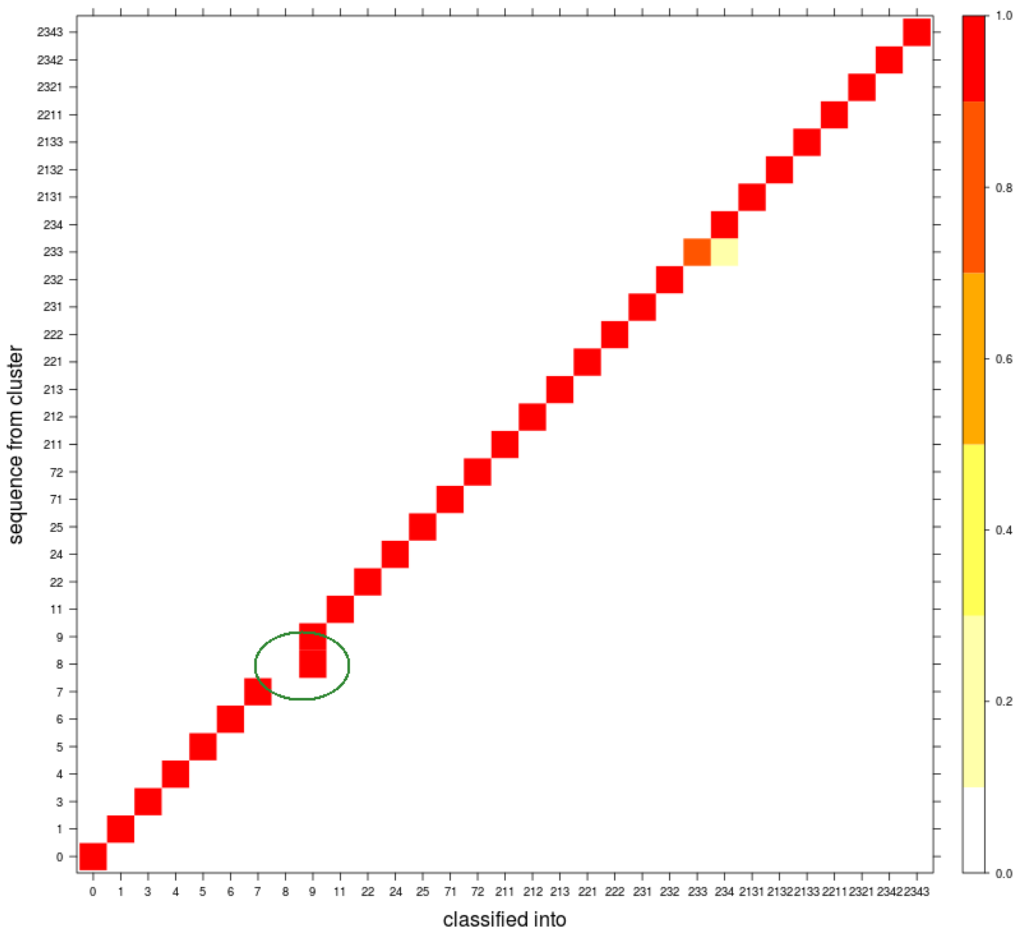


Figure 5.5: Levelplot for the `HMMcrossval.pl` results after correcting the HMM lengths (H5N1)

This method of cutting the input alignments with the objective to get similar profile lengths improves the classification accuracy from 94.6% to 97%. The results are again illustrated as a levelplot in Figure 5.5. The number of wrong assignments can be reduced from 116 to 66. More precisely, no sequence is misclassified to cluster 6 again but all sequences of cluster 8 are assigned to cluster 9 now (green frame in Fig. 5.5). The reason could be that the MSA for cluster 8 has a length of 1,625 after cutting. Thus the score is calculated over less positions than before. All remaining wrong classifications are to closely related clusters (e.g. 7→7.2, 2.2→ 2.2.2, 2.1.2→2.1.1). If these sequences are examined individually, one can see that some possess a specific property.

cluster	**sequence**	$dist_{av}(s)$
0	A/duck/Vietnam/1/2005	0.0403
5	A/chicken/Jilin/hp/2003	0.0279
7.1	A/chicken/Vietnam/NCVD-016/2008	0.0288
2.3.3	A/duck/Hunan/69/2004	0.0174

Table 5.4: Selection of misclassified sequences of H5N1 with a maximum average distance to the other sequences in their cluster

The average distance of a sequence s to the other sequences in their cluster, $Seq\backslash\{s\}$, is maximal. This distance is defined as

$$dist_{av}(s) = \frac{1}{|Seq| - 1} \sum_{s' \in Seq} d(s, s') \quad , \tag{5.3.1}$$

where $d(s, s')$ are the entries in the distance matrix D (the same matrix as for the clustering in section 4.1.1 and 4.1.2). Therefore, these sequences s are different and do not fit the profile as well as the remaining sequences in their cluster. This might lead to misclassification. The sequences in Table 5.4 have such a special role in their cluster which can also be observed in the phylogenetic tree of the cluster. Figure 5.6 shows the trees of the clusters 7.1 and 2.3.3 and the sequence with the maximum average distance in the red frame. It seems comprehensible that the correct classification of those sequences is difficult.

Unfortunately, all three sequences in cluster 8 have a length far smaller than 1,698 so that the full information of the gene cannot be used for the classification. As a result, the HMM profiles of the clusters are not of the same length but very long HMMs can be shortened. At this point it could be useful to introduce a normalized score to compensate still varying HMM lengths. According to equation (4.2.1) in section 4.2.1 the score needs to be divided through the length of the profile it is scanned against:

$$\tilde{c} \in \operatorname{argmax}_{c \in C} \; \frac{S_c(s_{c^*}^{(t)})}{|\mathcal{H}_c|} \; . \tag{5.3.2}$$

This method does not lead to a further improvement, an accuracy of 96%, but the results are similar to those with limiting the profile length to $\leq$ 1,698 positions. It is noticeable that many misclassifications are now not to cluster 6 but to cluster 8 (31 out of 86). But this almost exclusively affects the other first order clades.
Another approach would be to choose the shortest profile and bring all other profiles to that length L with

$$L = \min_{c \in C} |\mathcal{H}_c| \quad . \tag{5.3.3}$$

So all profile lengths are equal and only the DNA region of the HA gene is considered (ATG $\rightarrow L$ nucleotides). But the results show that the accuracy is even worse than without any correction of the profile length; 94.5%.

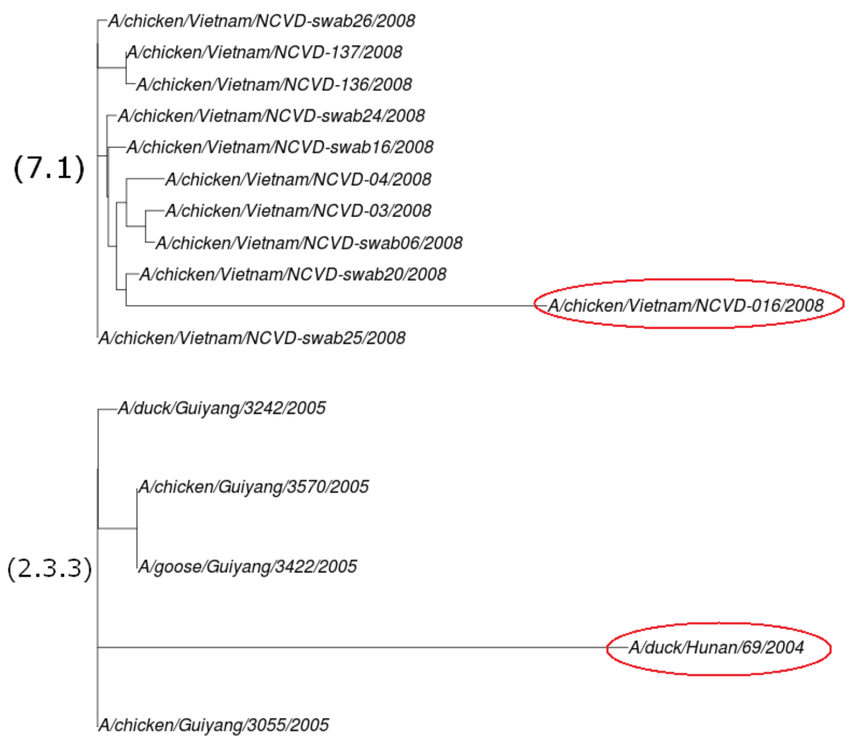

(7.1)
A/chicken/Vietnam/NCVD-swab26/2008
A/chicken/Vietnam/NCVD-137/2008
A/chicken/Vietnam/NCVD-136/2008
A/chicken/Vietnam/NCVD-swab24/2008
A/chicken/Vietnam/NCVD-swab16/2008
A/chicken/Vietnam/NCVD-04/2008
A/chicken/Vietnam/NCVD-03/2008
A/chicken/Vietnam/NCVD-swab06/2008
A/chicken/Vietnam/NCVD-swab20/2008
A/chicken/Vietnam/NCVD-016/2008
A/chicken/Vietnam/NCVD-swab25/2008
(2.3.3)
A/duck/Guiyang/3242/2005
A/chicken/Guiyang/3570/2005
A/goose/Guiyang/3422/2005
A/duck/Hunan/69/2004
A/chicken/Guiyang/3055/2005

Figure 5.6: Phylogenetic trees of clusters 7.1 and 2.3.3 (H5N1) [R package ape [22]], K80 distance

To adjust the profiles to their sequences and further improve the classification accuracy, an additional HMM training is performed. Apparently, the profiles with the corrected length to ≤ 1,698 positions (ATG → 1,698) obtain the best results. This is why all further analyses in chapter 5 and 6 make use of this method.

5.3.2 With additional HMM Training

The training in `HMMtraining.pl` (Alg. 4.2.2) creates profiles that are adjusted to the sequences of the clusters individually. This should lead to a better classification and improve the performance of the predictive model.

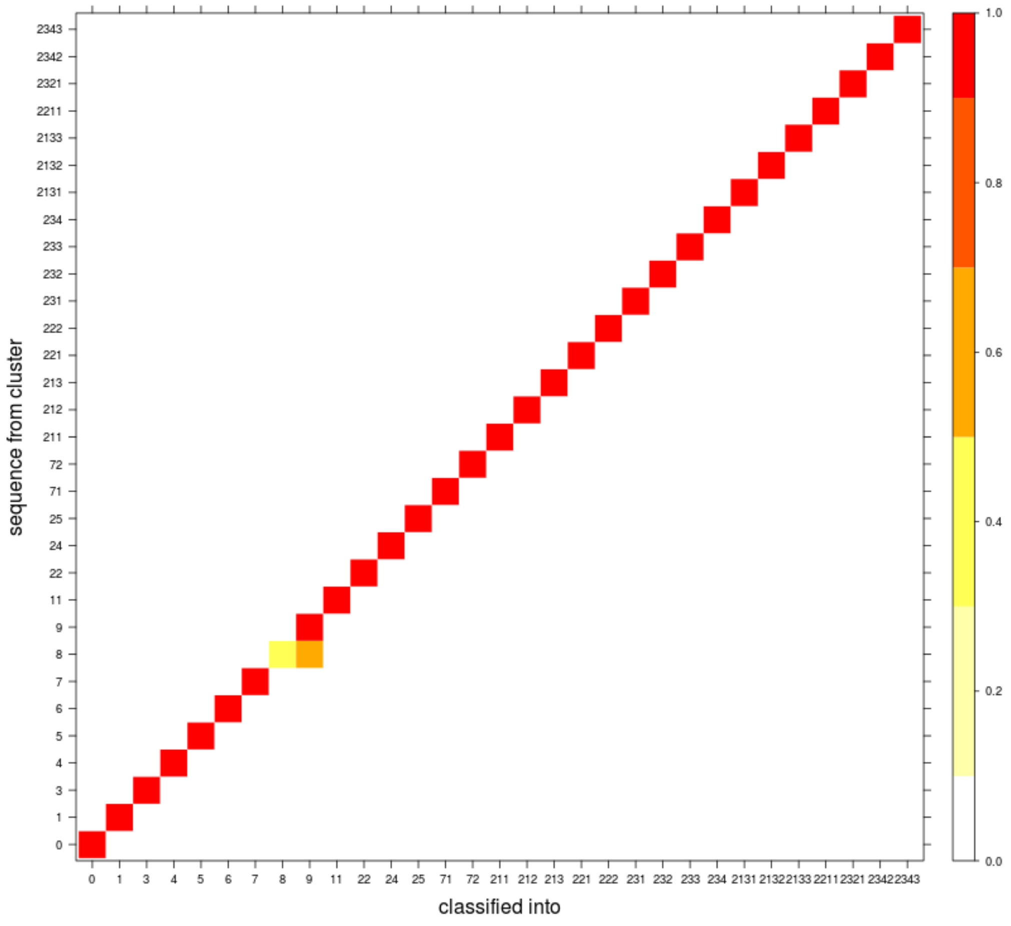


Figure 5.7: Levelplot for the `HMMtraineval.pl` results after correcting the HMM lengths (H5N1)

With the evaluation algorithm (`HMMtraineval.pl`, Alg. 4.2.3) this can be tested. I perform an evaluation with 50 test rounds which leads to a total number of 879 evaluation samples for the profile training. This is not exactly the same set as without the additional training as only 50 sequences of each cluster are tested (10 clusters have > 50 sequences). The classification results are again illustrated as a levelplot (Fig. 5.7) where the classification accuracy for each cluster and the wrong classifications are shown. Compared to the results without training (after correcting the HMM lengths) the right classification rate is increased by 2.2 percentage points from 97% to 99.2% which implies 872 right and 7 wrong assignments. These misclassifications are all to closely related clusters (compare Table 5.5). For the three sequences of cluster 0, 7.1 and 2.3.3 from Table 5.4 the classification is right after the profile training.

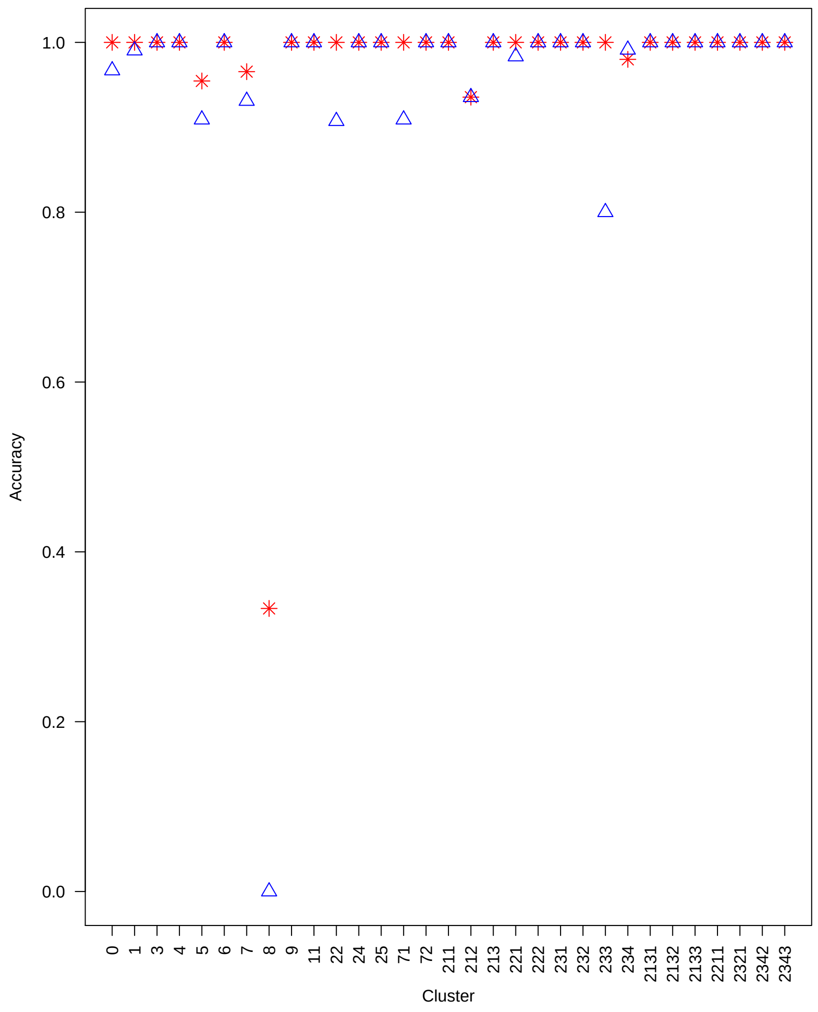


Figure 5.8: Comparing the classification accuracies for each cluster respectively (both after correcting the HMM lengths); blue triangles: without; red crosses: with additional training

Through the additional training the accuracy in every cluster remains constant or is improved except cluster 2.3.4. The reason could be that the number of evaluated test sequences per cluster in `HMMtraineval.pl` is limited to 50. Without the additional training two sequences are misclassified and the accuracy in this cluster is $\frac{228}{230}$. One of these sequences is among the 50 test sequences and the accuracy after the training is $\frac{49}{50}$ ($< \frac{228}{230}$). The total number of misclassifications is not increasing. In Figure 5.8 the accuracy without and with additional training for every cluster are compared to each other. The blue triangles are the accuracies without and the red crosses with additional training.

The two further methods with the normalised scores and the equal profile length obtain no further improvement of the classification accuracy; 99.1% and 98.7% respectively.

Discussion

The training algorithm works well and gives good results but there are still misclassifications after the HMM training. One reason for that could be the way the `hmmscan` method in `HMMER` and my training approach work. If `HMMER` used the Viterbi algorithm (section 3.2.2) to calculate the score, it would simply sum up the transition and emission values of the HMM file that generate the query sequence. In this case the additive changes of the HMM parameters would cancel out the whole score difference. However, `hmmscan` uses the Forward algorithm (section 3.2.2) with the result that the score difference cannot be passed fully when changing the match emission values by η (section 4.2.2). Another reason could be that the enhancement of an arbitrary HMM file X to a training sequence (emission values $-\eta$) can be partly changed back when another training sequence is misclassified to model X (emission values $+\eta'$). This could lead to wrong assignments after the training.

cluster	sequence	assigned to
5	A/chicken/Jilin/hp/2003	0
7	A/environment/Hunan/1-35/2007	7.2
8	A/chicken/Hong Kong/61.9/2002	9
8	A/chicken/Hong Kong/86.3/2002	9
2.1.2	A/chicken/Pangkalpinang/BPPV3/2004	2.1.1
2.1.2	A/turkey/Kedaton/BPPV3/2004	2.1.1
2.3.4	A/duck/Guangxi/5457/2005	2.3.4.2

Table 5.5: Remaining misclassifications after additional HMM training

The still remaining misclassified sequences after cutting and training the profiles are listed in Table 5.5. Maybe they would be correctly classified if the profile training in `HMMtraineval.pl` used not only 50 training sequences from each cluster to adjust the HMM parameters but more or all available sequences. In addition, to avoid a deterioration of the classification performance after the profile training, the number of training iterations could be increased or the cluster order for the profile training could be changed.

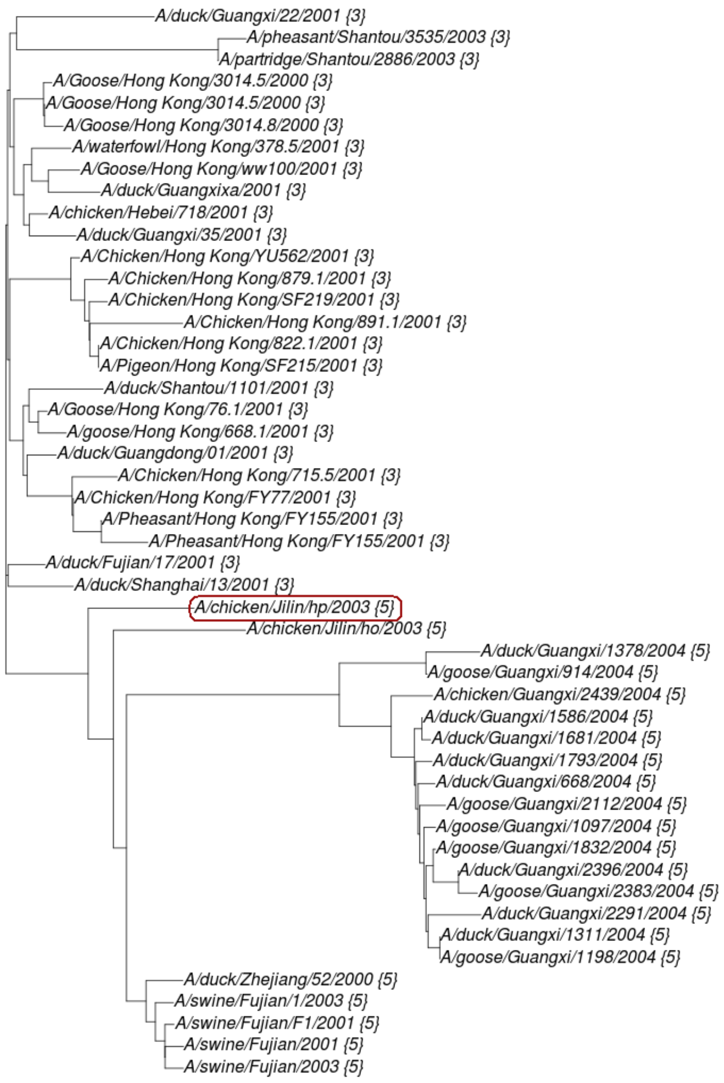

A/duck/Guangxi/22/2001 {3}
A/pheasant/Shantou/3535/2003 {3}
A/partridge/Shantou/2886/2003 {3}
A/Goose/Hong Kong/3014.5/2000 {3}
A/Goose/Hong Kong/3014.5/2000 {3}
A/Goose/Hong Kong/3014.8/2000 {3}
A/waterfowl/Hong Kong/378.5/2001 {3}
A/Goose/Hong Kong/ww100/2001 {3}
A/duck/Guangxixa/2001 {3}
A/chicken/Hebei/718/2001 {3}
A/duck/Guangxi/35/2001 {3}
A/Chicken/Hong Kong/YU562/2001 {3}
A/Chicken/Hong Kong/879.1/2001 {3}
A/Chicken/Hong Kong/SF219/2001 {3}
A/Chicken/Hong Kong/891.1/2001 {3}
A/Chicken/Hong Kong/822.1/2001 {3}
A/Pigeon/Hong Kong/SF215/2001 {3}
A/duck/Shantou/1101/2001 {3}
A/Goose/Hong Kong/76.1/2001 {3}
A/goose/Hong Kong/668.1/2001 {3}
A/duck/Guangdong/01/2001 {3}
A/Chicken/Hong Kong/715.5/2001 {3}
A/Chicken/Hong Kong/FY77/2001 {3}
A/Pheasant/Hong Kong/FY155/2001 {3}
A/Pheasant/Hong Kong/FY155/2001 {3}
A/duck/Fujian/17/2001 {3}
A/duck/Shanghai/13/2001 {3}
A/chicken/Jilin/hp/2003 {5}
A/chicken/Jilin/ho/2003 {5}
A/duck/Guangxi/1378/2004 {5}
A/goose/Guangxi/914/2004 {5}
A/chicken/Guangxi/2439/2004 {5}
A/duck/Guangxi/1586/2004 {5}
A/duck/Guangxi/1681/2004 {5}
A/duck/Guangxi/1793/2004 {5}
A/duck/Guangxi/668/2004 {5}
A/goose/Guangxi/2112/2004 {5}
A/goose/Guangxi/1097/2004 {5}
A/goose/Guangxi/1832/2004 {5}
A/duck/Guangxi/2396/2004 {5}
A/goose/Guangxi/2383/2004 {5}
A/duck/Guangxi/2291/2004 {5}
A/duck/Guangxi/1311/2004 {5}
A/goose/Guangxi/1198/2004 {5}
A/duck/Zhejiang/52/2000 {5}
A/swine/Fujian/1/2003 {5}
A/swine/Fujian/F1/2001 {5}
A/swine/Fujian/2001 {5}
A/swine/Fujian/2003 {5}

Figure 5.9: Joint phylogenetic tree of clusters 3 and 5 (H5N1) [R package ape [22]], K80 distance

This type of analysis could be used to identify wrong assigned reference labels. The remaining wrong classifications after profile training can be considered to be wrong labels in the reference clustering. This can be observed in a joint phylogenetic tree of the reference cluster and the wrong assigned cluster. An example is given in Figure 5.9 where the joint phylogenetic tree of the clusters 3 and 5 is shown. As one can see, the misclassified sequence *A/chicken/Jilin/hp/2003* in cluster 5 (red frame) is closely arranged to cluster 3. This can be an indicator that the cluster label for this sequence is maybe wrong.

Altogether, the accuracy of the classification can be improved when correcting the length of the input MSA (ATG $\rightarrow \leq$ 1,698 nucleotides) and again with additional HMM training. From 94.6% without training and without correcting the MSAs up to 99.2% after training and correcting the MSAs. This is summed up in Table 5.6 where also the two other methods with the normalised scores and the equal profile length are listed.

	without additional training (2,161 sequences)	with additional training (879 sequences)
no further approaches	94.6%	98.6%
cutting the MSA	97%	99.2%
cutting + normalise	96%	99.1%
cutting to equal length	94.5%	98.7%

Table 5.6: Summary of the classification accuracy (H5N1)

5.3.3 Score Calculation

Every sequence s gets a score against each profile when s is scanned against the HMM database. By applying a ROC-analysis for the scores of each cluster, as described in section 4.2.3, one can calculate optimum cutoffs cut^* concerning a defined optimization criterion. Typical ROC-curves for one of the clusters are given in Figure 5.10. Both curves show the outcomes of the analysis for cluster 2.2.1 with the difference that the left one uses the scores without and the right one the scores with additional HMM training. As was expected through the good classification results, both curves indicate a high accuracy.

-	cluster 2.2.1	not cluster 2.2.1
$\geq cut^*$	271	268
$< cut^*$	31	1,591

-	cluster 2.2.1	not cluster 2.2.1
$\geq cut^*$	49	50
$< cut^*$	1	779

Table 5.7: Contingency tables without and with additional training for cluster 2.2.1
without: cut^*=2167.5 with: cut^*=2135

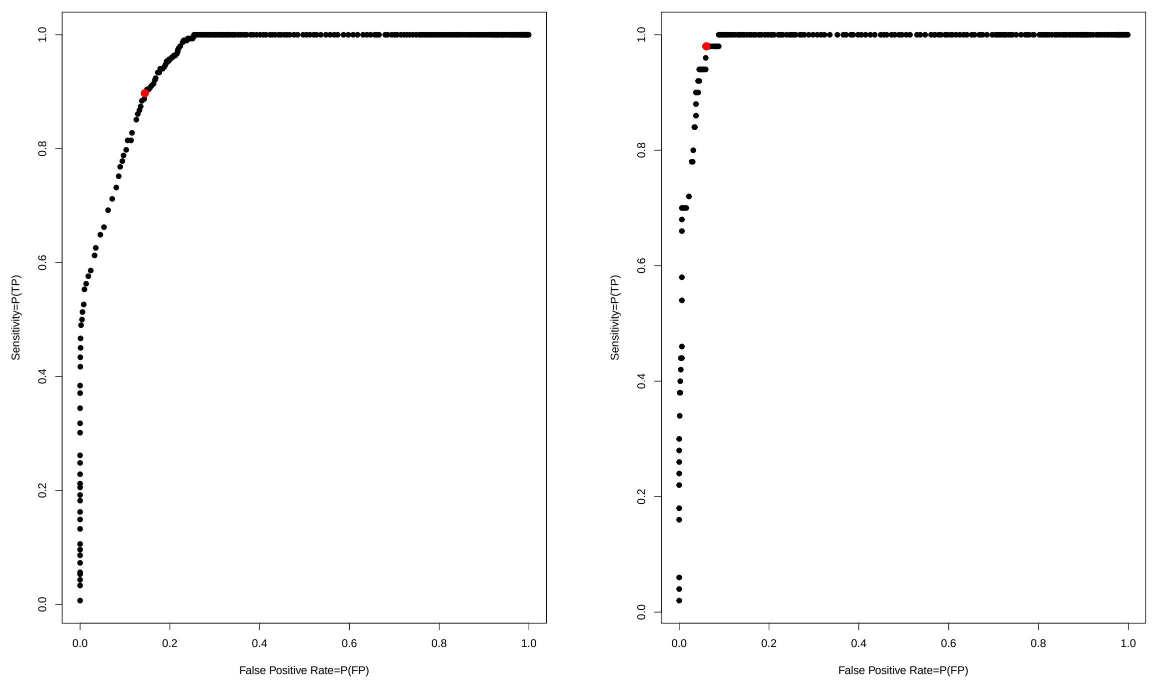


Figure 5.10: ROC-curves for cluster 2.2.1 (without additional training and cutting the MSA $\rightarrow$ with additional training + cutting the MSA

The red point in each curve represents the optimum cutoff according to the criterion in equation (4.2.5). Table 5.7 shows the associated contingency tables for the optimal cutoffs without and with additional training. In detail, taking this contingency tables the sensitivity and specificity are improved through the training. Sensitivity from 89.7% to 98% and specificity from 85.6% to 94% respectively.

When comparing both figures to each other, one can see that after the training the curve is closer to the optimum point P(0,1). The distance between this point and the red point on the curve can be calculated using the following equation:

$$dist(P, cut^*) = \sqrt{(1 - Se)^2 + (1 - Sp)^2} \ .$$

The resulting distances are 0.177 without and 0.0635 with additional training. This reflects the improvement of the classification with this HMM training.
According to equation (4.2.3) optimum cutoffs cut^* for the *two step classification* (see section 4.2.3) can be determined.

5.3.4 The Use of Cutoffs

As explained in section 4.2.3, every sequence, no matter if it belongs to H5N1 or not, could be assigned to one of the models of HPAIV H5N1. This can be avoided when adding cutoffs as a second step for the classification. In order to distinguish between a sequence of HPAIV and unrelated sequences, I define score cutoffs cut_i^* for every model $i = 1, \dots, k$ that have a sensitivity of at least 99% for model i (equation 4.2.3).

To test the accuracy of this *two-step-classification* (HMM database + cutoffs), I compile a cutoff-test dataset. It contains sequences of all different HN-types if they were available at NCBI [8]. In total 2,584 sequences:

- all sequences from the cross-validation used to determine the cutoffs → 2,161 sequences,
- five sequences of any other HN-type (if available) → 418 sequences,
- five NA sequences of H5N1.

For the classification test I use the trained HMM database that is determined through `HMMtraining.pl` and the cutoffs calculated with the scores of `HMMtraineval.pl`. The classification of this dataset results in:

	H5N1 (HA)	not H5N1 (HA)	total
cutoff exceeded	2,146	1	2,147
cutoff not exceeded	15	422	437
total	2,161	423	2,584

Table 5.8: Results of the *two-step classification*

The one sequence that is mistakenly assigned to H5N1 (HA) is *A/duck/Eastern China/031/2009 (HA)* of H5N5 and is assigned to cluster 2.3.4. The accuracy of the classification can be measured by either taking the "H5N1 (HA)" or the "not H5N1 (HA)" samples into account. If considering the overall 2,161 H5N1 HA sequences, the sensitivity is at 99.3%, as required when calculating the cutoffs and the specificity at 99.8% (see equations in section 4.2.3). The right classification rate is 99.4%.

6 Application to H1N1

The results for H5N1 in the previous chapter show that the applied methods are well suitable to identify and classify HPAIV H5N1 subfamilies. But the question is if they are universally applicable to other HN-types of Influenza A. A good test example are the sequences of H1N1. This subtype is of current interest in science and medicine, since it is the cause of the seasonal flu pandemics. This is why there is a sufficient amount of sequence data available that can be used for the analysis.

6.1 Data Selection

From the NCBI Influenza Virus Resource [8] I select all H1N1 HA gene sequences with a length of at least 1,600 nucleotides because this is the threshold that is used for H5N1. This set of sequences is referred to as 2012H1N1 dataset (collection date: February 7th, 2012) and contains 4,183 sequences. Because these sequences also represent samples of the same gene for one influenza subtype, they are closely related. The distribution of the pairwise distances among the sequences is shown in Figure 6.1. Compared to Figure 5.1 of H5N1, there is a huge peak at distance 0. As will be seen in section 6.2, this can be explained by the large cluster 7 (2,414 sequences) which has many distances of 0 among its sequences. Summarized, the minimum distance is 0 and the maximum is 0.3408 at an average of 0.1182. The values for the maximum and the average distance exceed those for H5N1. Possible reasons could be:

- All sequences of the 2011H5N1 dataset share a common ancestor sequence *A/ goose/Guangdong/1996 H5N1* from 1996 (set by the *H5N1 Evolution Working Group* (section 2.3)) whereas the 2012H1N1 dataset contains all available H1N1 (HA) sequences (the oldest *A/swine/Iowa/15/1930 H1N1* from 1930). $\Rightarrow$ The sequences of H1N1 have been collected over a period of more than 80 years (H5N1 only 15 years). This longer time of evolution might result in higher nucleotide distances among the sequences.
- The sequences of the 2012H1N1 dataset were chosen by me and not by experts and also maybe the selection criteria of H5N1 are unsuitable for H1N1.
- I cannot name a common ancestor sequence in my dataset or even ensure that

all H1N1 sequences evolved from such a sequence. It is more likely that within 80 years genetic re-assortment (section 2.2) has taken place.

- Perhaps the mutation rates of both HN subtypes are different.

To apply my classification method with the simplified HMM training without considering indels, I must ensure that the proportion of inner gaps in the MSAs of the clusters is negligibly small. At this point I presuppose the results of the clustering which are explained in the following section. For the clusters 1 to 54 in Table 6.1 the gap fraction lies between 0 and 0.0073 with an average of 0.0003. This serves as evidence to ignore the indels when applying the additional profile HMM training.

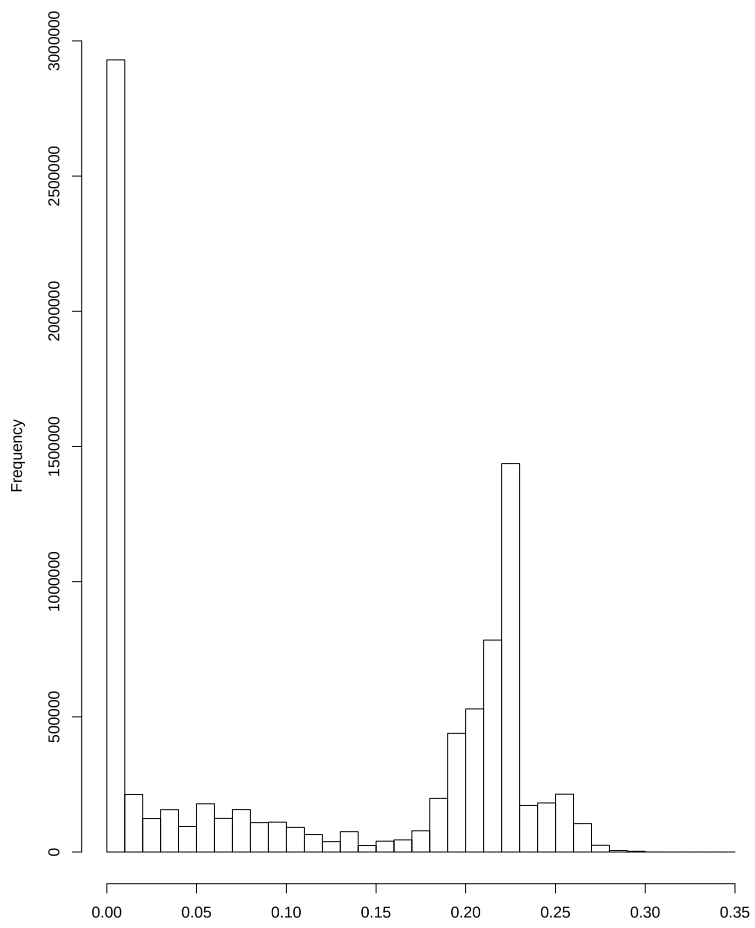


Figure 6.1: Histogram of the pairwise phylogenetic distances in the 2012H1N1 dataset [R package ape [22]], K80 distance

6.2 Cluster Analysis

Both cluster algorithms (`clusterboot.r`, Alg. 4.1.1 and `clustergreedy.r`, Alg. 4.1.2) are applied to the 2012H1N1 dataset with the objective to identify clusters of possible H1N1 subfamilies. Because the former algorithm has a better result with the 2011H5N1 dataset, I use the H1N1 clusters determined with `clusterboot.r` as a reference labelling for the analysis.

6.2.1 Clusters of H1N1 (with clusterboot.r)

cluster label	# sequences	cluster label	# sequences
1	10	28	11
2	10	29	15
3	19	30	15
4	604	31	4
5	9	32	4
6	6	33	6
7	2,414	34	10
8	40	35	4
9	11	36	12
10	121	37	26
11	193	38	5
12	7	39	4
13	11	40	8
14	30	41	44
15	12	42	6
16	50	43	7
17	6	44	5
18	38	45	4
19	7	46	4
20	4	47	10
21	12	48	4
22	6	49	8
23	14	50	5
24	6	51	6
25	16	52	9
26	13	53	71
27	12	54	45

Table 6.1: Clusters of H1N1 identified with `clusterboot.r`
(54 clusters + 150 outliers ⇒ 4,183 sequences)

In summary, 54 clusters plus 150 outlier sequences (summed up in cluster 55) were identified through the algorithm `clusterboot.r` (compare Table 6.1). These clusters vary significantly in their size, from a minimum of 4 to a maximum of 2,414 sequences. At first appearance this may seem surprisingly. But if the clusters are examined individually, one can see that all sequences in cluster 7 are from 2009 to 2011 and the sequences from cluster 4 are from 2005 to 2009.

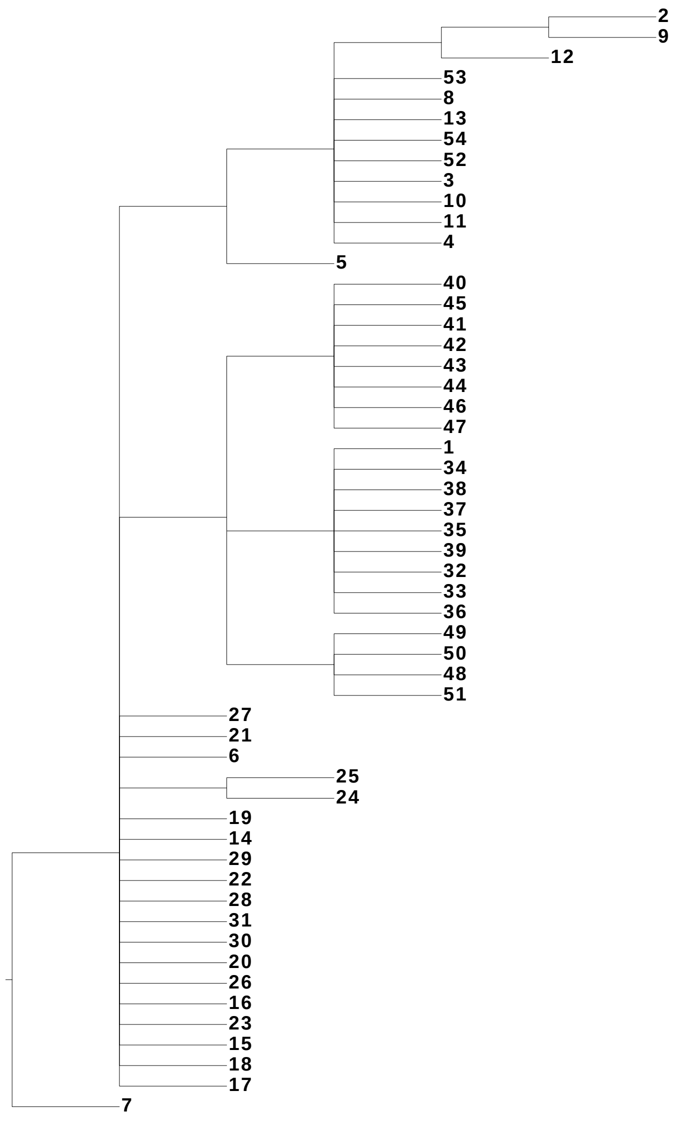


Figure 6.2: Phylogram of the H1N1 clusters (`clusterboot.r`)

There are different reasons for that. On the one hand the amount of sequence data is increasing continually[1]. On the other hand much data was collected and added to the databases during the H1N1 pandemic 2009/2010[2] and these sequences form a huge cluster 7. Generally, more influenza samples are collected and sequenced when a special HN-subtype causes a pandemic. A phylogram of the identified clusters is given in Figure 6.2. This tree can be taken to evaluate whether the wrong assignments of the classification are to closely related clusters or not.

6.2.2 Comparison

The results of both cluster algorithms can be compared to each other.

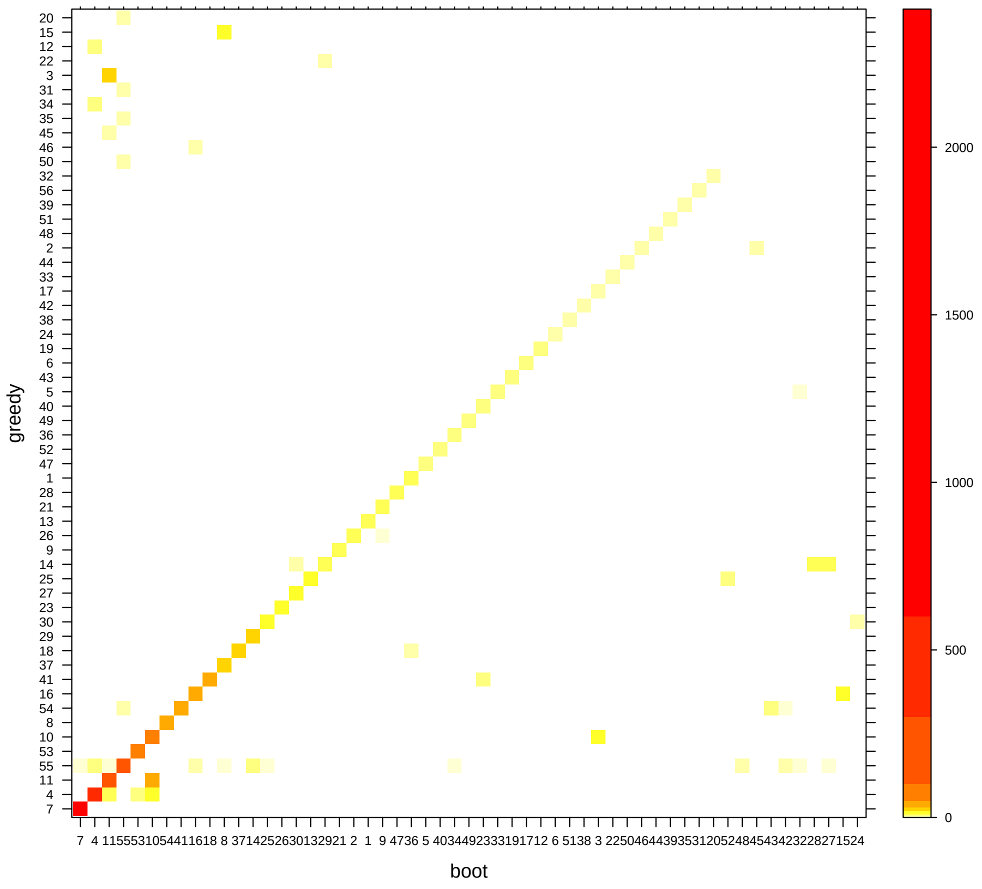


Figure 6.3: Levelplot of boot versus greedy clusters (H1N1)

[1]http://www.nature.com/scitable/nated/content/45068/nucleotide_growth_FULL.gif (July 15, 2012)

[2]http://www.cdc.gov/h1n1flu/cdcresponse.htm (July 15, 2012)

The algorithm `clusterboot.r` identified 54 clusters plus outlier sequences and `clustergreedy.r` 55 clusters plus outliers. The contingency table where the clusters of the first algorithm are matched against the clusters of the second one is given in the attachment (bootvsgreedy_H1N1.csv). As with H5N1, this table is illustrated as levelplot (Fig. 6.3) that shows where the sequences are assigned through both clusterings. The additional clusters 55 and 56 for boot and greedy represent the outlier sequences.
To evaluate the two clusterings and to measure their similarity, the Rand-index can be computed. As distinct from the comparisons for H5N1, the index is very high at 0.9832. This indicates that both algorithms nearly identified the same clusters in this dataset. Furthermore, the fraction of entries on the main diagonal is approx. 92.8%. One main reason for that is the high match of both large clusters that make up the biggest part of the sequences. A perfect relation between boot and greedy clusters with just one entry per row and column is given for 11 boot clusters and nearly satisfied for 7 other clusters.

6.3 Classification with Markov Profiles

With the 2012H1N1 dataset a cross-validation without and with additional profile training is performed (attachment: HMMcrossvalcut_H1N1.csv, HMMtrainevalcut_H1N1.csv). As well as for H5N1 the lengths of the HMM profiles can be considered as a cause for misclassification. The profile lengths for H1N1 vary from 1,668 to 1,777 with an average length of 1,731. To avoid this error source and get profiles of similar lengths, the alignments are cut before the profiles are built. For that task the same method as with H5N1 is used. The MSAs and HMMs are built for the clusters in Table 6.1. These labels of cluster assignment are taken as reference to evaluate the accuracy of the classification methods.

6.3.1 Without additional HMM Training

The simple cross-validation (`HMMcrossval.pl`, Alg. 4.2.1) is applied to the H1N1 clusters. To save computational time, the number of classified test sequences in each cluster is limited to 100. This results in a set of 1,096 evaluated sequences. However, the HMMs for this classification are still built on all sequences in the cluster.

This cross-validation without additional training results in 1,072 right and 24 wrong classifications which is a classification accuracy of 97.8%. The full contingency table of the results is given in the attachment. Therefore, the levelplot in Figure 6.4 shows the graphical representation of the results and illustrates the classification accuracy.

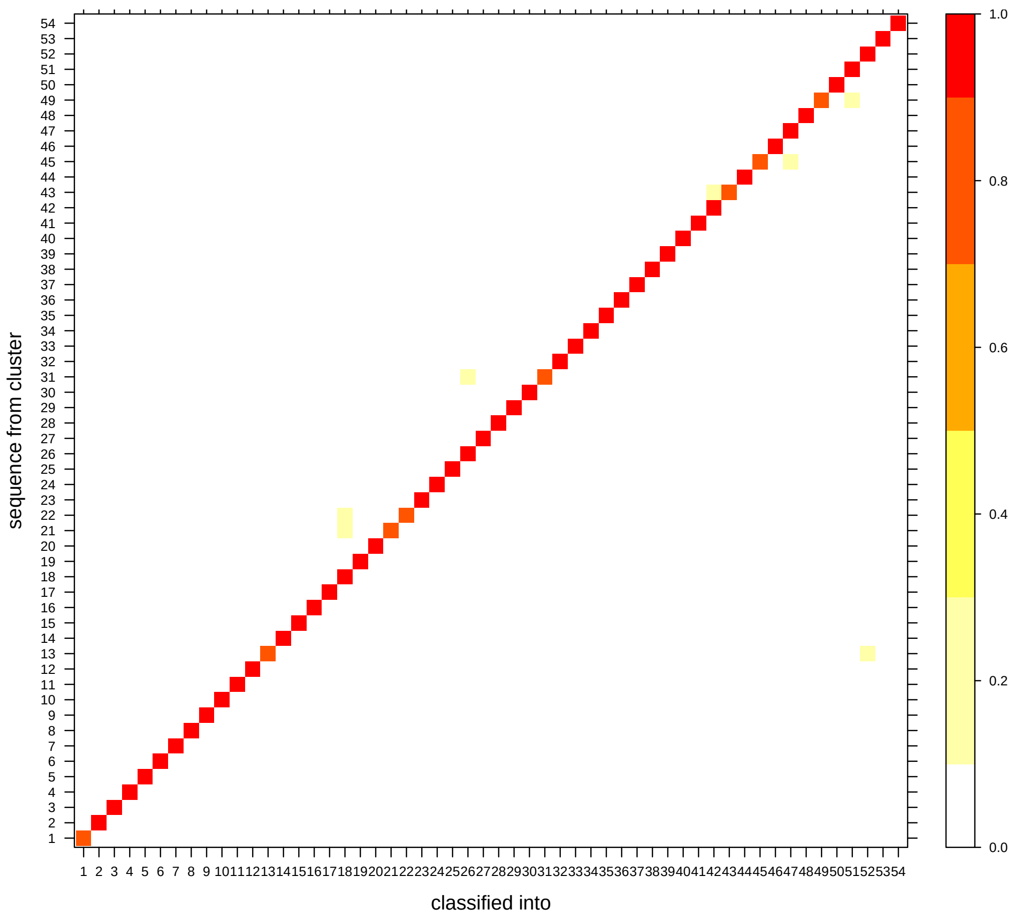
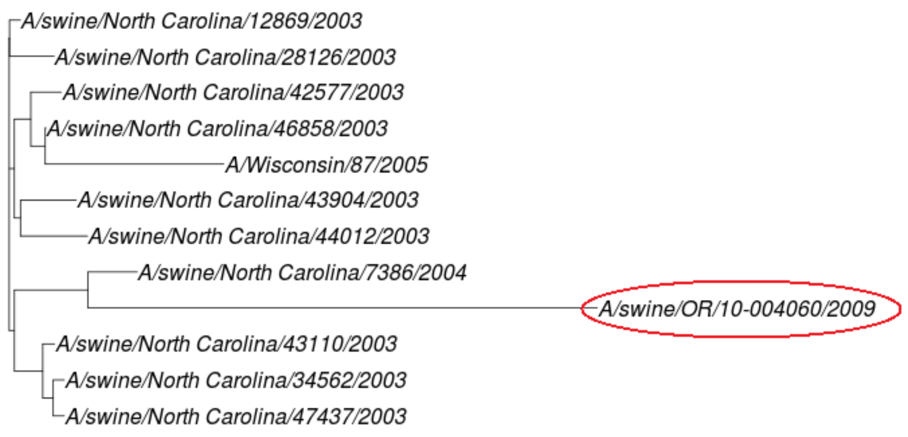


Figure 6.4: Levelplot for the `HMMcrossval.pl` results after correcting the HMM lengths (H1N1)

Discussion

These 24 misclassifications do not occur more frequently in small or big clusters. They are distributed among clusters of different size and furthermore, all wrong assignments are to closely related clades. This feature can be examined when comparing the misclassifications to the phylogram of the H1N1 clades (Fig. 6.2). There is no cluster like cluster 6 for H5N1 where the greatest proportion of wrong assignments is located. But a possible reason for the misclassifications can be an erroneous clustering through the algorithm which leads to ambigious profiles. It is also conceivable that the cluster definition for H5N1 is not applicable for H1N1 and thus maybe another clustering approach is advisable. At last, errors in the input alignment or in the sequences themselves cannot be excluded either.

(15)


A/swine/North Carolina/12869/2003
A/swine/North Carolina/28126/2003
A/swine/North Carolina/42577/2003
A/swine/North Carolina/46858/2003
A/Wisconsin/87/2005
A/swine/North Carolina/43904/2003
A/swine/North Carolina/44012/2003
A/swine/North Carolina/7386/2004
A/swine/OR/10-004060/2009
A/swine/North Carolina/43110/2003
A/swine/North Carolina/34562/2003
A/swine/North Carolina/47437/2003

(22)

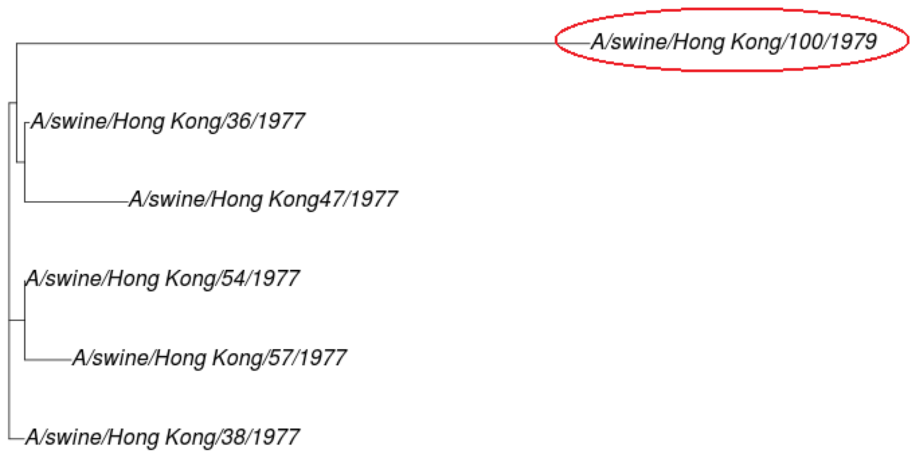

A/swine/Hong Kong/100/1979
A/swine/Hong Kong/36/1977
A/swine/Hong Kong47/1977
A/swine/Hong Kong/54/1977
A/swine/Hong Kong/57/1977
A/swine/Hong Kong/38/1977

Figure 6.5: Phylogenetic trees of clusters 15 and 22 (H1N1) [`R` package ape [22]], K80 distance

As previously with H5N1, the remaining misclassifications can be examined individually. Some of these sequences (see Table 6.2) possess the property of maximum average distance in their cluster as defined in equation (5.3.1). By way of illustration, the phylogenetic trees of cluster 15 and 22 are shown in Figure 6.5 with the specific sequence in the red frame. This property was also observable for some sequences of H5N1.

cluster	sequence	average distance
9	A/Phila/1935	0.0267
13	A/Hickox/1940	0.034
15	A/swine/OR/10-004060/2009	0.0344
21	A/swine/Kyoto/3/1979	0.0297
22	A/swine/Hong Kong/47/1977	0.0242
23	A/swine/Iowa/2/1987	0.0312
49	A/duck/Shimane/188/1999	0.031

Table 6.2: Selection of misclassified sequences (H1N1)

6.3.2 With additional HMM Training

The HMM training evaluation with `HMMtraineval.pl` (Alg. 4.2.3) for H1N1 is performed with 50 evaluation rounds. This results in 880 test sequences to determine the accuracy of the classification. From these sequences 859 are assigned to the right cluster and 21 sequences are misclassified which is an accuracy of 97.6%. This value is slightly lower than without additional profile training. The levelplot of the classification results is given in Figure 6.6.

cluster	before training	after training	misclassifications
4	$\frac{99}{100} = 99\%$	$\frac{49}{50} = 98\%$	1 × cluster 53
10	$\frac{97}{100} = 97\%$	$\frac{48}{50} = 96\%$	3 × cl. 11 and 2 × cl. 11 resp.
53	$\frac{70}{71} \approx 98.6\%$	$\frac{49}{50} = 98\%$	1 × cluster 4

Table 6.3: Correlation between the accuracy and the number of evaluated sequences

Discussion

For most clusters the classification accuracy remains constant or is improved through the additional HMM training. But especially for some bigger clusters (> 50 sequences) the accuracy is decreased. These clusters are listed in Table 6.3. The accuracy after training the profiles also depends on how the algorithm `HMMtraineval.pl` choses the test sequences. If the previously misclassified sequences are among the

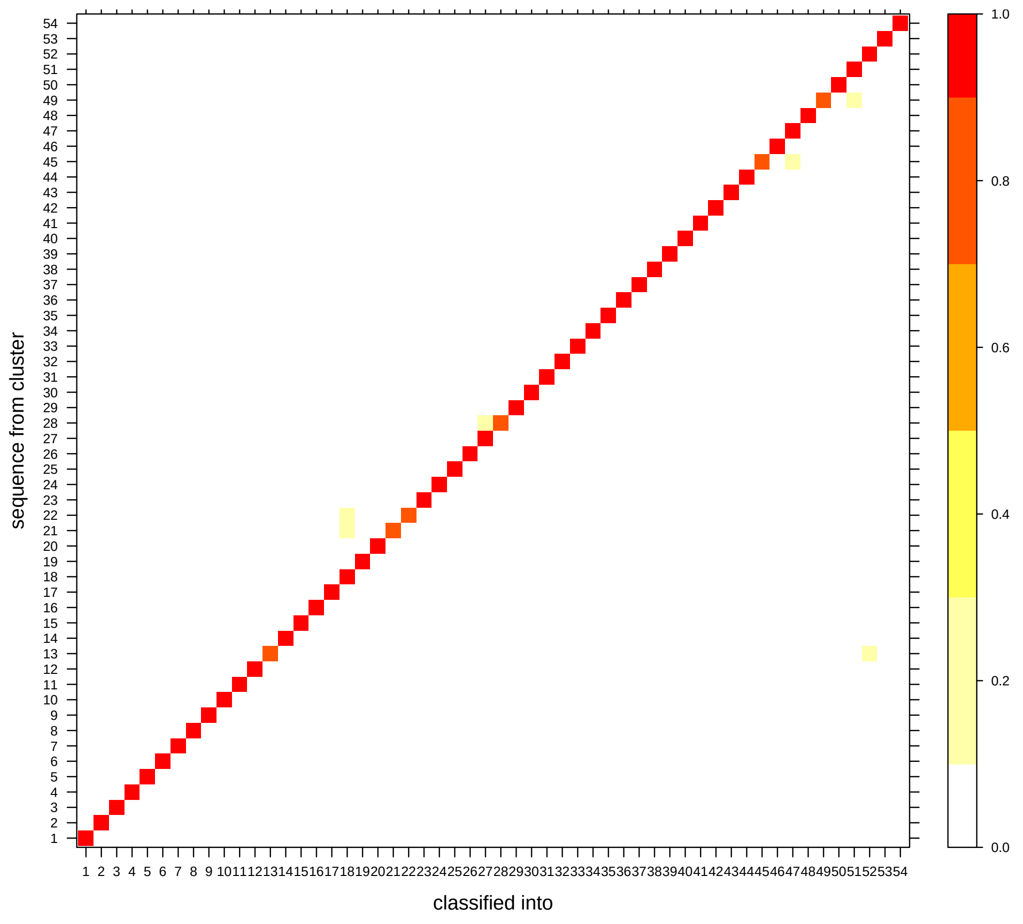


Figure 6.6: Levelplot for the `HMMtraineval.pl` results after correcting the HMM lengths (H1N1)

first 50 sequences in the cluster, they can again be assigned to the wrong cluster. Because the total number of evaluated sequences in each cluster is limited to 50, the accuracy can be worse after the HMM training although the number of misclassifications itself is not increased. In cluster 10, for example, this number is even reduced (two of the tree are among the first 50 sequences). However, the total number of test sequences is smaller so the accuracy gets worse (compare Table 6.3). Maybe if another subset of these clusters (e. g. sequence 51 – 100) would be used for the analysis, the evaluation results were different.

The only cluster where the accuracy is really getting worse through the training algorithm is cluster 18 where the number of misclassifications increases from 2 to 3. One reason could again be that parameter corrections ($+\eta$, $-\eta'$) cancel each other out. To avoid this, the number of training iterations can be set on a higher value

(> 1) or the cluster order for the profile training needs to be changed.
Many of the still misclassified sequences possess a maximum average distance in their cluster again. But from all sequences in Table 6.2 those from cluster 9 and 23 are correctly classified after training. So the additional profile training leads to a reduction of the wrong assignments in some clusters.

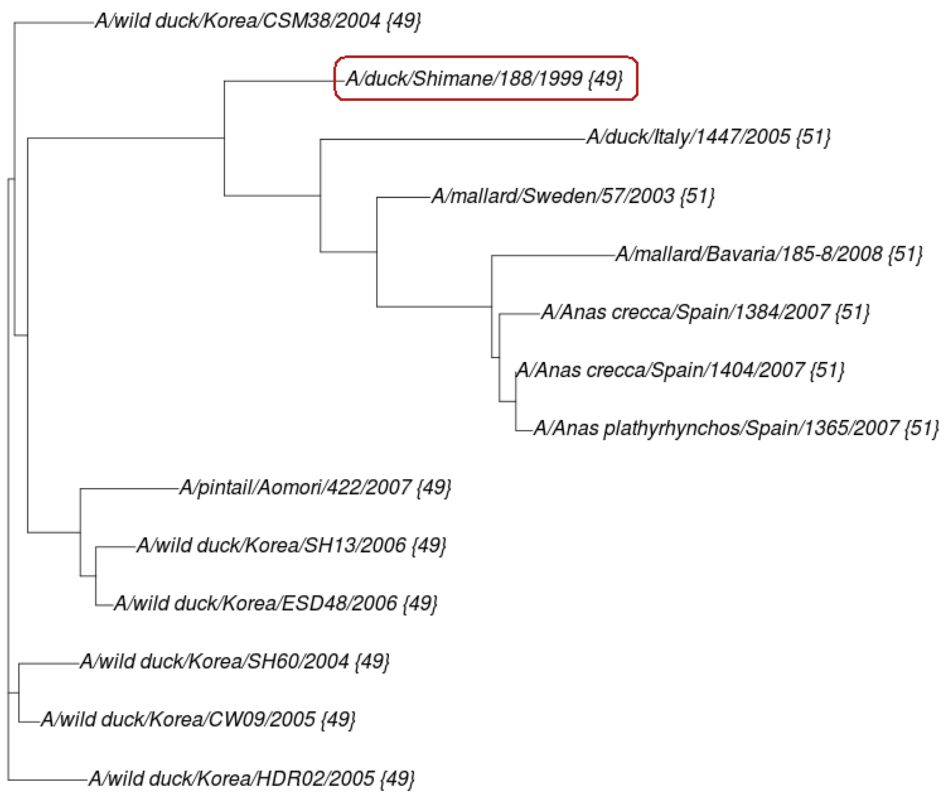

A/wild duck/Korea/CSM38/2004 {49}
A/duck/Shimane/188/1999 {49}
A/duck/Italy/1447/2005 {51}
A/mallard/Sweden/57/2003 {51}
A/mallard/Bavaria/185-8/2008 {51}
A/Anas crecca/Spain/1384/2007 {51}
A/Anas crecca/Spain/1404/2007 {51}
A/Anas plathyrhynchos/Spain/1365/2007 {51}
A/pintail/Aomori/422/2007 {49}
A/wild duck/Korea/SH13/2006 {49}
A/wild duck/Korea/ESD48/2006 {49}
A/wild duck/Korea/SH60/2004 {49}
A/wild duck/Korea/CW09/2005 {49}
A/wild duck/Korea/HDR02/2005 {49}

Figure 6.7: Joint phylogenetic tree of clusters 49 and 51 (H1N1) [R package ape [22]], K80 distance

As well as for the H5N1 clusters, these remaining mistakes can be interpreted to be wrong assignments in the reference clustering. This feature can be illustrated in the joint phylogenetic tree of the reference cluster and the cluster it is assigned to. Figure 6.7 shows such a tree for the clusters 49 and 51. It can be seen that the sequence *A/duck/Shimane/188/1999* from cluster 49 in the red frame is closely arranged to cluster 51 and could therefore be an incorrect labelling.

In summary, it can be said that the training for H1N1 is applicable. Some sequences that are misclassified before are correctly assigned after the profile training. The differences in the accuracies of the large clusters mainly occurs because of the different test sequences for `HMMcrossval.pl` and `HMMtraineval.pl`. Maybe the additional training should be performed for all 1,096 sequences that are used without training. All results for H1N1 with the different approaches are summed up in Table 6.4.

	without additional training (1,096 sequences)	with additional training (880 sequences)
no further approaches	96%	96.3%
cutting the MSA	97.8%	97.6%

Table 6.4: Summary of the classification accuracy (H1N1)

7 Conclusion

In my thesis I present a complete work-flow for the subfamily identification and classification of influenza A subtypes. On the one hand this includes methods for clustering the sequences and on the other hand classification methods based on hidden Markov profiles. The experiments with H5N1 and H1N1 show that especially this classification method is well applicable and reaches a high accuracy.

7.1 The Clustering

The first step in the analysis is the clustering considering on a number of predefined criteria. My approach is to integrate the criteria into an algorithm that identifies the correct clusters. To evaluate my two clustering algorithms (`clusterboot.r` and `clustergreedy.r`), I take the cluster assignments for H5N1 set by the *H5N1 Evolution Working Group* as reference. From the output of the cluster algorithms it is recognizable that the clustering mostly keeps sequences together that belong to the same cluster but many clusters are split or merged. Although `clusterboot.r` is a little better than `clustergreedy.r` (see section 5.2), there are still significant differences to the reference clustering. This can have various reasons:

- different initial datasets for the analysis (my 2011H5N1 set is not identical with the HPAIV H5N1 reference set),
- manual adjustment of the reference clusters by the *H5N1 Evolution Working Group*,
- the use of different programs and/or methods (e.g. maximum likelihood, Bayesian) and
- errors in the reference cluster assignments.

Nevertheless, the algorithm `clusterboot.r` is applied to identify subfamily clusters for H1N1. But the possibility that the clade defining criteria for H5N1 are maybe unsuitable for H1N1 cannot be ruled out. In summary, it can be said that both algorithms do not really achieve what is requested, namely to reproduce the reference clustering.

It could be a future task to determine a consistent clustering method and to set individual criteria for each influenza A subtype. For that purpose, experts should be consulted to set criteria that are biological meaningful.

7.2 The Classification

The second step in the analysis is the classification where I use a profile based method; the `HMMER 3.0` tool. For every subfamily cluster of an influenza A subtype I build a hidden Markov profile. The sequences are scanned against the profiles and assigned to that model where they get the maximum score. This classification method has an accuracy of 94.6% (H5N1) and 96% (H1N1). For a predictive model this is a good result but when taking a closer look on the misclassifications one can recognize some features.

1. The profiles vary in lengths and especially long sequences are misclassified to longer models.
2. The correct assignment to smaller clusters seems to be more difficult than to larger ones.
3. Many of the misclassified sequences have a maximum average distance to the other sequences in their cluster.
4. The sequence data are not perfect. many sequences are far shorter than 1,698 nucleotides. So the full length and thus the whole information of the gene is not available for the classification.

The first point, the varying HMM lengths, comes about because the genes sequences in the databases have a length from 1,600 to 1,780. This is the result of the sequencing methods that were used and the threshold set for the data selection. To exclude this error source of varying profile lengths, I perform a correction of the input MSA length by cutting away the ends of the alignments. As a result, just the HA-gene section of the sequence (ATG $\rightarrow$ 1,698 nucleotides) is used for the classification. I test the classification with this "cut HMMs" on H5N1 and H1N1 and get right classification rates of 97% and 97.8% which is an improvement of the former results.

To correct the remaining misclassifications, another approach that bases on the sequences and the associated profile parameters can be considered. The aim now is to improve the classification accuracy by manipulating the parameters of the HMMs. This can only work because `HMMER` is insensitive to parameter editing as long as the file format is kept. So I implement a training algorithm `HMMtraining.pl` and an algorithm `HMMtraineval.pl` to evaluate the training that adjust the match emission parameters to the sequences of the clusters. Both approaches can be put together (cutting the MSA + additional training the HMM files). With both methods the accuracy is 99.2% for H5N1 and 97.6% for H1N1.

Because some profiles are shorter than 1,698 positions, one can try to compensate this still varying HMM length. One possibility is to introduce normalized scores (see equation (5.3.2)) and another approach is to shorten the alignments to the length of the shortest profile (see equation (5.3.3)). But for H5N1 the accuracies are not improved through this methods (compare table 5.6).

As a second criterion in the classification I suggest cutoffs to avoid misassignment of unrelated sequences. This *two step classification* gets good results (section 5.3.4). Such an approach could be helpful in identifying new sublineages of different influenza types. Another objective that could be pursued is the identification of re-assortant viruses through an entire Influenza A HMM profile database.

7.3 Prospects

There are still a lot of approaches for improvement and development on this topic that can be considered. The first one might be to improve the training and thus the classification accuracy. Suggestions are:

- The correction steps of the HMM parameters can be varied ($p \cdot \eta$, $p \in (0, 1]$) when defining p as learning rate for the profiles.
- Referring to the former point the number of "correction iterations" can be increased (≥ 1). So the profiles are trained on the sequences several times in succession.
- Additionally, the order to choose the clusters for the profile training can be set at random.
- One can distinguish between conserved and variable positions in the alignment of the profile and the training sequence. When adjusting the profile parameters one can add a different fraction of η to them.

Another point that can be attempt is to use only a section of the full HA gene. As already mentioned in section 2.2, HA is composed of three identical monomers and each of them consists of two different subunits (HA1 and HA2). It would be interesting to see if the classification works with just one monomer or just with the HA1 or HA2 section.

But before the analysis of some sequences can start, it must be ensured that the input sequences are suitable. One cannot exclude that the available data in the public databases show sequencing errors or have a wrong annotation. For example long parts with only Ns or missing nucleotides or sequences that do not belong to the specific HN-type could exist. This leads to erroneous multiple alignments and this affects the HMM profiles. Therefore, the data selection should be more specific. Perhaps it would be reasonable to apply some kind of a sequence filter that selects unsuitable sequence data from the input.

Final Conclusion

This training is well applicable to H5N1 and H1N1. It is surprising that `HMMER` files can be manipulated so very easily. Maybe this kind of training is not only applicable to Influenza A but to other sequences. A consistent profile database could be established to standardise the nomenclature of influenza A sequences.

Maybe this kind of training can be expanded to other Influenza A HN-types in future to establish a complete profile database. This might also be applied to the NA gene. This would lead to a database for monitoring the Influenza A evolution and expansion or for identifying re-assortant viruses.

Attachment CD

List of Algorithms

List of Tables

List of Figures

Bibliography

[1] http://www.smcm.edu/rivergazette/archives/winter09/outbreak.html, July 28, 2012.

[2] http://www.nobelprize.org/nobel_prizes/medicine/laureates/1958/lederberg-bio.html, July 28, 2012.

[3] http://www.thesun.co.uk/sol/homepage/, July 25, 2012.

[4] http://www.thetimes.co.uk/tto/news/, July 25, 2012.

[5] http://hmmer.janelia.org/, April 30, 2012.

[6] Rolf Knippers. *Molekulare Genetik.* Georg Thieme Verlag, 2006.

[7] J. Haß and S. Matuszewski. Diversität von Influenza A subtyp H1N1. Master's thesis, Ernst-Moritz-Arndt Universität Greifswald, 2010.

[8] Y. Bao, P. Bolotov, D. Dernovoy, B. Kiryutin, L. Zaslavsky, T. Tatusova, J. Ostell, and D. Lipman. The influenza virus resource at the national center for biotechnology information. *J. Virol.*, 2008.

[9] S. Modrow, D. Falke, and U. Truyen. *Molekulare Virologie.* Spektrum Akademischer Verlag, Heidelberg, 2003.

[10] WHO. Updated unified nomenclature system for the highly pathogenic H5N1 avian influenza viruses. *www.who.int*, October 2011.

[11] WHO/OIE/FAO H5N1 Evolution Working Group. Toward a unified nomenclature system for highly pathogenic avian influenza virus (H5N1). *http://wwwnc.cdc.gov/eid/article/14/7/07-1681.htm*, July 2008.

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

[13] Mario Stanke. Vorlesung Genomanalyse, WS 2011/12.

[14] http://www-history.mcs.st-andrews.ac.uk/biographies/markov.html, May 29, 2012.

[15] Mario Stanke. Vorlesung Bioinformatik, WS 2011/12.

[16] Sean R. Eddy. What is a hidden markov model? *Nature Biotechnology*, 22:1315–1316, 2004.

[17] Sean R. Eddy. *HMMER User's Guide*. HMMER Development Team Janelia Farm Research Campus, Ashburn VA 20147 USA, March 2010.

[18] Martin Gollery. *Handbook of Hidden Markov Models in Bioinformatics*. Chapman & Hall/CRC, Boca Raton, 2008.

[19] Joseph Felsenstein. *Inferring Phylogenies*. Sinauer Associated, Inc., Sunderland, Massachusetts, 2004.

[20] Boris Mirkin. *Clustering for Data Mining*. Chapman & Hall/CRC, Boca Raton, 2005.

[21] R Development Core Team. *R: A Language and Environment for Statistical Computing (Version 2.1.13)*. R Foundation for Statistical Computing, Vienna, Austria, 2011. ISBN 3-900051-07-0.

[22] E. Paradis, J. Claude, and K. Strimmer. APE: analyses of phylogenetics and evolution in R language. *Bioinformatics*, 20:289–290, 2004.

[23] B. Schuster. *Statistik für Human- und Sozialwissenschaftler*. Springer-Verlag Berlin Heidelberg, 2010.

[24] K.H. Zou, A. Liu, A.I. Bandos, L. Ohno-Machado, and H.E. Rockette. *Statistical Evaluation of Diagnostic Performances*. Chapman & Hall/CRC, Boca Raton, 2012.

[25] Adam L. Bazinet and Michael P. Commings. A comparative evaluation of sequence classification programs. *BioMed Central*, 2012.

[26] R.C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, 2004.

Danksagung

Zuallererst möchte ich mich bei Prof. Mario Stanke von der Ernst-Moritz-Arndt Universität Greifswald und Dr. Mario Ziller vom Friedrich-Loeffler-Institut auf Riems bedanken, für die Betreuung meiner Diplomarbeit sowie für viele hilfreiche Anmerkungen und Denkanstöße.

Ein großes Dankeschön geht auch and den "5th Floor & Friends". Vielen Dank für eure Motivation, das Korrekturlesen, die lustigen Kaffeerunden und die familiäre Atmosphäre und dass ihr mich nicht habt vergessen lassen, wie viel Spaß eine Diplomarbeit machen kann.

Abschließend möchte ich all jenen Menschen danken, die mich durch fünf wunderbare Studienjahre begleitet haben; meiner Familie und meinen Freunden, die immer ein offenes Ohr und eine Schulter zum Anlehnen für mich haben.

Danke!