Accurate and fast taxonomic profiling of microbial communities

DAMON SHAHRIVAR

Master’s Degree Project
Stockholm, Sweden 2015

XR-EE-KT 2015:001
Abstract

With the advent of next generation sequencing there has been an explosion of the size of data that needs to be processed, where next generation sequencing yields basepairs of DNA in the millions. The rate at which the size of data increases supersedes Moores law therefore there is a huge demand for methods to find meaningful labels of sequenced data. Studies of microbial diversity of a sample is one such challenge in the field of metagenomics. Finding the distribution of a bacterial community has many uses for example, obesity control. Existing methods often resort to read-by-read classification which can take several days of computing time in a regular desktop environment, excluding genomic scientists without access to huge clusters of computational units.

By using sparsity enforcing methods from the general sparse signal processing field (such as compressed sensing), solutions have been found to the bacterial community composition estimation problem by a simultaneous assignment of all sample reads to a pre-processed reference database.

The inference task is reduced to a general statistical model based on kernel density estimation techniques that are solved by existing convex optimization tools. The objective is to offer a reasonably fast community composition estimation method. This report proposes, clustering as a means of aggregating data to improve existing techniques run-time and biological fidelity. Use of convex optimization tools to increase the accuracy of mixture model parameters are also explored and tested. The work is concluded by experimentation on proposed improvements with satisfactory results.

The use of Dirichlet mixtures is explored as a parametric model of the sample distribution where it is deemed that the Dirichlet is a good choice for aggregation of k-mer feature vectors but the use of Expectation Maximization is unfit for parameter estimation of bacterial 16s rRNA samples.

Finally, a semi-supervised learning method found on distance based classification of taxa has been implemented and tested on real biological data with high biological fidelity.
Abstract

Nya tekniker inom DNA-sekvensering har givit upphov till en explosion på data som finns att tillåta. Nästa generations DNA-sekvensering genererar baspar som sträcker sig i miljonerna och mängden data ökas i en exponentiell takt, vilket är varför det finns ett stort behov av ny skalbar metodik som kan analysera kvantitiv data för att få ut relevant information. Den bakteriella artfördelning av ett provrörs en sådan problemställning inom meta-genomik, vilket har flera tillämpningsområden som exempelvis, studier av fettma. I dagsläget så är den vanligaste metoden för att få ut artfördelningen genom att klassifiera DNA-strängarna av bakterierna, vilket är en tidskrävande lösning som kan ta upp emot ett dygn för att processera data med hög upplösning. En snabb och tillförlitlig lösning skulle därför tillåta fler forskare att ta del av nästa generations sekvensering och analysera dess data som i sin tur skulle ge upphov till mer innovation inom området.

Alternativa lösningar med inspiration från signalbehandling har hittats som nyttjar problemställningens glesa natur genom användning av Compressed Sensing. Svar hittas genom att simultant tilldela strängar till en för-processerad referensdatabas. Problemställningen har förenklats till en statistisk modell av provrörs med ickeparametrisk estimering för att implicit få ut fördelningen av bakteriearter med hjälp av konvex optimering.

Denna rapport föreslår användningen av klastrerings för aggregering av data för att förbättra tillförlitligheten av svaren och minska tiden för beräkning av dessa. Användningen av parametriska modeller, Dirichlet fördelningen, har utforskas där rapporten har kommit fram till att antaganden för lämpligheten av denna som ett medel att aggregera k-mer vektorer är rimliga men att parameterestimeringen med Expectation Maximization ej fungerar väl i samband med Dirichlet och en omskrivning av α parameteren skulle behövas i vektorrymden som späns av 16S rRNA genen.

Slutligen så har distansbaserad tilldelning av bakterier testats på data från verklig biologisk kontext med väldigt hög noggranhet.
Acknowledgements

I would like to express my sincere gratitude to my supervisor Saikat Chatterjee for his support and engagement through the learning process of this master thesis. You’ve been a mentor, a friend and thanks to you, I have been brought into contact with the fulfilling world of research which has inspired and given me the confidence to pursue my ideas and dreams. Furthermore, I would like to thank David Koslicki for his help and the most appreciated feedback of efficient coding in Matlab with the use of vectorized methods and instructions on how to set up a good experimentation work flow in a meta-genomic setting.
List of Figures

1. Frequency counting algorithm for 2-mers . . . . . . . . . . . . . . . . . . . 5
2. Three dimensional plots of Dirichlet distribution . . . . . . . . . . . . . . . 6
3. Simplex region of the Dirichlet . . . . . . . . . . . . . . . . . . . . . . . . . . 7
4. Mean square error of K-means . . . . . . . . . . . . . . . . . . . . . . . . . . 8
5. Illustration of K-means algorithm on Old Faithful data set . . . . . . . . . 9
6. A flow-chart of full ARK system . . . . . . . . . . . . . . . . . . . . . . . . . . 22
7. ARK-Quikr VD on real data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
8. ARK-SEK VD on real data . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
9. Mock communities: VD convergence of ARK methods . . . . . . . . . . . . 30
10. ARK-SEK: Reconstruction error at Taxonomic ranks . . . . . . . . . . . . 31
11. Simulated data: VD decrease and run time increase . . . . . . . . . . . . . 32
12. Simulated data: Mean VD error and Mean execution time . . . . . . . . . . 32

List of Tables

1. Time and VD measured for different training matrices . . . . . . . . . . . . 33
2. Optimization impact on reconstruction error . . . . . . . . . . . . . . . . . . 35
# Contents

1 Introduction
  1.1 Thesis scope and contributions .................................. 1
  1.2 Previous work .................................................. 2
  1.3 Problem description ........................................... 3

2 Theoretical background
  2.1 Law of large numbers ........................................... 4
  2.2 Compressive sensing ............................................ 4
  2.3 K-mer feature extraction ....................................... 5
  2.4 Dirichlet distribution ......................................... 6
  2.5 Clustering ....................................................... 7
    2.5.1 K-means .................................................... 7
    2.5.2 Expectation Maximization ................................. 8

3 Method
  3.1 Training data .................................................. 12
  3.2 Simultaneous supervised classification ....................... 13
    3.2.1 Problem modelling ....................................... 13
    3.2.2 Aggregate reads by K-means to improve accuracy .......... 15
    3.2.3 Improving computation time by reducing dimensionality . 16
    3.2.4 Alternate Optimization to improve accuracy ............. 17
  3.3 Semi-supervised learning ..................................... 17
    3.3.1 Expectation Maximization with Dirichlet Mixtures ..... 18

4 Implementation
  4.1 Optimization problem ......................................... 19
  4.2 K-means clustering ............................................ 19
  4.3 Simultaneous supervised classification ....................... 20
    4.3.1 Aggregation of reads by k-means ........................ 20
    4.3.2 Dimension reduction of training data ................... 23
    4.3.3 Alternate optimization .................................. 24
  4.4 Semi-supervised learning .................................... 25
    4.4.1 EM with Dirichlet Mixtures ............................... 25
    4.4.2 Semi-supervised K-means ................................. 25

5 Results
  5.1 Experimentation setup ......................................... 27
  5.2 Data sets ...................................................... 27
  5.3 Simultaneous supervised classification ....................... 28
    5.3.1 ARK results for mock communities data .................. 28
    5.3.2 ARK results for simulated data .......................... 31
    5.3.3 Results for dimension reduction of training data ...... 33
    5.3.4 Results for alternate optimization ...................... 33
    5.3.5 Results for combined system .............................. 36
  5.4 Semi-supervised learning .................................... 36
    5.4.1 EM with Dirichlet Mixtures ............................... 36
    5.4.2 Semi-supervised k-means .................................. 37
6 Discussion and conclusion

6.1 Closing remark
1 Introduction

The study of DNA sequences is essential in the study of life. With the discovery of sequencing methods such as Sanger sequencing, scientists had the ability to elucidate genetic information for any biological system. The true diversity of the microbial world was mostly uncharted since organisms present in a sample had to be cultivated for successful observation of its genetic information [2].

New sequencing technologies have been developed, also called Next-Generation Sequencing, these allow genomic scientists to extract genetic information at an unprecedented level of detail [33]. This high resolution of genetic information adds an insight into complex microbial populations with low abundance species which paves the way for greater understanding of bacterial communities in diverse environments [23, 44].

Large scientific initiatives to map the human microbiota [43] actively use the diversity as a means to analyze bacterial communities function in human health related studies [27, 46]. The use of microbial diversity isn’t limited to human health but many other domains where microbes fulfill a role in its given eco-system such as soil analysis for sustainable agriculture [28] and wastewater treatment [40]. It is needless to say that the implications of developing robust and reliable methods to assess the microbial diversity of different samples is profound and of importance for the scientific community.

To access the diversity of a sample, the 16S rRNA gene is used to identify different taxa, by reason of an universal distribution among bacteria and its slow evolutionary changes over time making it fit for classification [24, 13].

1.1 Thesis scope and contributions

This thesis is ultimately based on the experimental implementation and data of the SEK paper [14], the subject of a large international collaboration. The work of Chatterjee et al was based on the mixture model representation of a sample put forth in [35] and thus the scope of the exploratory analysis is based on this fundamental assumption. The main theme of the thesis investigates possible uses of machine learning techniques to improve the reconstruction accuracy more specifically, use of clustering to improve resolution of data and exploring use of parametric models in k-mer feature space.

Underlying intuition and theory was provided by S. Chatterjee for the improvements on SEK, Section 3.2.2, 3.2.3 and 3.2.4. In silico experimentation in Section 5.3.2 were performed by D. Koslicki. K-mer generating code was supplied from previous SEK experimentation software. All Matlab implementations part of this thesis are the work of the author with input from S. Chatterjee on the underlying mathematical theory and feedback by D. Koslicki on vectorized methods. The design and implementation of Section 4.4.2 was done independently by the author. Methods developed, more precisely the improvements on SEK, in this thesis have been subject of an international collaboration with J. Corrander 1, A. W. Walker 2, M. Vehkaperä 3, D. Koslicki 4, Y. Lan 5, L. J.

1Dept of Mathematics and Statistics, University of Helsinki, Finland
2Microbiology Group, Rowett Institute of Nutrition and Health, University of Aberdeen, UK
3Dept of Signal Processing, Aalto University, Finland
4Dept of Mathematics, Oregon State University, Corvallis, USA
5Dept of Physics, Tsinghua Unviversity, Beijing, China
1.2 Previous work

The high-throughput approach focuses on producing for each sample a large number of reads covering certain variable part of the 16S rRNA gene, which enables an identification and comparison of the relative frequencies of different taxa present across samples. Depending on the characteristics of the samples, the bacteria involved and the quality of the acquired sequences, the taxa may correspond to species, genera or even higher levels of hierarchical classification of the variation existing in the bacterial kingdom. Given the exponential increase of read sets produced per sample in typical applications, a huge need for fast inference methods to assign meaningful labels to the sequence data, a modern problem which has already received considerable attention [45, 35, 29, 39]. As of today massive amount of reads can be available and need to be analysed.

Many approaches to the bacterial community composition estimation problem use 16S rRNA amplicon sequencing where a large amount of moderate length (around 250-500 bp) reads are produced from each sample and then either clustered or classified to obtain estimate of any particular taxon. In the clustering approach reads are grouped into taxons by either distance-based or probabilistic methods [11, 19, 16], such that the actual taxonomic labels are assigned to the clusters afterwards by classification of their consensus sequences to a reference database. The Bayesian estimation of bacterial communities (BeBAC) method [16] was shown to provide high biological fidelity by employing maximum likelihood based clustering framework along-with stochastic search and sequence alignment. However the high accuracy of BeBAC comes at a heavy computational price such that it is required to run for several days in a computing-cluster environment for large read sets.

Another manner to solve the bacterial estimation problem is the classification approach, which is based on using a reference database directly to assign reads to meaningful labels representing biological variations. Methods for the classification of reads have been based either on homology using sequence similarity or on genomic signatures in terms of oligonucleotide composition. Examples of homology-based methods include MEGAN [26, 38] and phylogenetic analysis [44, 34, 31, 41]. A widely used solution is the Ribosomal Database Project’s (RDP) classifier which is based on a naïve Bayesian classifier (NBC) that assigns a label explicitly to each read produced for a particular sample [45, 17]. Despite the computational simplicity of NBC, the RDP classifier may still require several days to process a data set in a desktop environment. Given this challenge, considerably faster methods based on principle of aggregation of

---

6Illumina Cambridge Ltd., Chesterford Research Park, Essex, UK
7MRC Tropical Epidemiology Group, London School of Hygiene and Tropical Medicine, London, UK
8Dept of Communication Theory, KTH Royal Institute of Technology, Stockholm, Sweden
reads have been proposed, for example Taxy [35], Quikr [29] and recently SEK [14].

1.3 Problem description
The task at hand is to determine the taxonomic profile (composition of different species) of a microbial community with high gap accuracy in a computing environment. To solve this, an easily available measurement data set (sample) and reference data are at the disposal of this thesis work. The reference data set is labelled and contains perfect length sequences of known bacteria that have been cultivated in a lab. The measurement data set has variable length sequences that are unlabelled, it is not known if a particular sequence is any specific bacteria. The measurement data set could be from any type of sample, soil, human gut, marina and etc. The inference task is to determine the proportion of every species in the measurement or sample. The scope of the project will be to improve existing methods such as [14] and propose new methods. The project will not be concerned with biological interpretation of the results produced by the proposed methods, as these are outside the domain of expertise of the author.
2 Theoretical background

2.1 Law of large numbers

The theorem states that, the average of the result of an experiment is close to the expected value, as the number of trials increases the sample mean converges to the distributional mean. In mathematical notation:

\[ \bar{X}_n = \frac{1}{n} \sum_{i=1}^{n} X_i \rightarrow \mu, \text{ as } n \rightarrow \infty. \]

Given a large enough set of data it is possible to estimate the true distributional mean and variances with the use of sample mean.

2.2 Compressive sensing

Prior to compressed sensing it has been widely accepted that to reconstruct any given signal the sample rate has to be at least twice that of the highest frequency of the given signal, also known as the Nyquist rate. Signals that aren’t naturally band-limited are bounded by other measures to fulfill a required resolution. In general, data acquisition is done by sampling at uniform time intervals with a frequency higher or equal than that of the Nyquist rate.

The concept of compressive sensing tells us differently, a signal can be reconstructed with far less samples. This is possible due to two key properties, sparsity and incoherence. Sparsity simply means that any given signal with length of \( n \) can be represented with \( k \) nonzero coefficients where \( k << n \). Many natural signals have a concise representation in the proper basis, the signals are very spread out in the convenient domain but when expanded in an orthonormal basis the small coefficients of the vectors are discarded without much observable loss. The vector is now sparse in a strictly sparse meaning, that most of the coefficients are zero. This is the underlying theory for most lossy compressors, to adaptively encode the most significant coefficients and throw away the remaining ones. In compressive sensing there is this notion of coherence of basis. A sensing matrix that senses the input signal into a sparse representation and then the representation basis that transforms this sparse representation. The coherence measures the correlation between these two, any element that has high coherence is correlated. The coherence is measured by:

\[ \mu(\Phi, \Psi) = \sqrt{n} \cdot \max_{1 \leq k, j \leq n} |\langle \varphi_k, \psi_j \rangle|. \]

A more simple explanation, coherence measures the largest correlation between any two elements of \( \Phi \) and \( \Psi \). In compressive sensing the elements of interest are those with low coherence and the intuition behind this could be explained as followed, any coefficient that isn’t correlated to any other coefficient cannot be reconstructed with any other information except its own, so by throwing away highly correlated coefficients one throws away information that is reconstructable by other coefficients.

Reconstruction is done by \( l_1 \)-minimization of the underdetermined linear system:

\[ f^* = \Psi \cdot x^* \]
where $x^*$ is the reconstructed signal. Since $l_1$-minimization is a non trivial problem unlike the $l_2$ norm, there exists various methods for signal reconstruction with compressed sensing principles [12]. The reader is referred to [18] for further reading about compressed sensing.

### 2.3 K-mer feature extraction

Previous work in genomic science, strings of DNA have often been compared by their similarity in specific regions of genes. However this requires some a priori knowledge of specific bio-markers. A general framework has been devised to capture the statistics of a genetic signal in a concise and intuitive manner while still retaining variability amongst the taxa. This is done by capturing the different frequencies of fixed length patterns or permutations of strings in a signal of genetic information called k-mers(oligonucleotides), see Figure 1. Each position in a k-mer feature vector corresponds to a particular permutation and the number of occurrences of that particular pattern/permutation. The counting of the patterns is done by choosing a window size, this window will slide from the beginning of the sequence till the end.

It will capture different k-mers and increment the counter as its sliding through the genetic signal. The window size determines the dimension of the k-mer feature vectors. A bigger window size allows for more variability, more permutations which results in a sparser matrix. As the variance between the different species increase so will the classification accuracy of the result, where each species becomes more separable at the cost of huge memory allocation. A single base-pair can take form in four different nucleic acids \{A C G T\} and therefore the dimension of a k-mer feature vector can be counted to $4^k$ or in general mathematical notation, $x_n \in \mathbb{R}_+^{4^k \times 1}$. By employing 4 basepair wide window in k-mer extraction, the effective dimension of the data reaches $4^4 = 256$. 

![Figure 1: Frequency counting algorithm for 2-mers, Figure 3 from [20]](image-url)
\( \alpha_k = 0.1 \)

\( \alpha_k = 1 \)

\( \alpha_k = 10 \)

Figure 2: Three dimensional plots of Dirichlet distribution, where the horizontal axes are coordinates in the simplex region and vertical axis corresponds to the density values. The plots are from Figure 2.5 [7].

2.4 Dirichlet distribution

The Dirichlet distribution, often denoted Dir(\( \alpha \)), is a continuous multivariate probability distribution that is parametrized by the vector \( \alpha \in \mathbb{R}^+ \). The role of the Dirichlet is often as a prior distribution in Bayesian statistical modelling. An intuitive explanation of the Dirichlet is that it’s a distribution of distributions where the dimension represents rival events or categories. A way to put this explanation into a real world context, suppose that there exist \( N \) samples that are generating \( K \)-dimensional multinomially distributed vectors, however each source \( K \) is parametrized differently from the others. The union of the vectors can be modelled by a \( K \)-dimensional Dirichlet distribution. Since the vector elements that are fed into the Dirichlet machine are probabilities, the domain of the Dirichlet is itself a probability distribution and thus one may view the Dirichlet as the uncertainty of the actual distribution that generated the data point. The Dirichlet probability density function is expressed as

\[
\text{Dir}(\alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \prod_k p_k^{\alpha_k - 1}
\]

(1)

where the entries \( p_k \in [0, 1] \) and \( \|p\|_1 = 1 \).

Parameter Intuition To understand the role of \( \alpha \), it is good to remember that the Dirichlet is often used as a prior distribution as previously mentioned. In the context of priors it is often desirable to have a uniform probability for all components in a mixture model, this is achieved by setting the \( \alpha_k \) to 1 as viewed in Figure 2b. The \( \alpha \) can be understood by the following representations

\[
m = \mathbb{E}[p_k] = \frac{\alpha_k}{s}
\]

(2)
2.5 Clustering

2.5.1 K-means

Clustering is the process of identifying groups of data points in a high-dimensional setting. The resulting grouping of data points is called clusters, intuitively one may view a cluster as a group of data points whose inter-point distances are small compared to the distances of data points outside of the cluster. To begin variable $\mu_k$ will be introduced to represent the mean of the $k$th cluster. The centroids are assigned in a manner to minimize the sum of squares of distances of each data point to its closest centroid vector $\mu_k$. K-means is a hard clustering algorithm meaning that a data point can only be part of a single cluster therefore it is convenient to have a binary indicator variable $r_{nk} \in \{0,1\}$ where it describes whether the data point $x_n$ is assigned to the $k$th cluster. Now the sum of square distances that was mentioned which is also known as the squared-error distortion measure in literature, is defined as:

$$D = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2.$$ 

The objective is to minimize $D$ which can be done in an iterative procedure where each iteration is involved in two steps. First and foremost, an initial value to $\mu_k$ is assigned, this can be done with either random sampling or splitting technique which will be discussed in depth in the implementation section. In the first step, $D$ is minimized with respect to $r_{nk}$ by keeping $\mu_k$ fixed. This is done by nearest neighbour classification of data points, each data point is compared to each cluster centroid $\mu_k$ and assigned to the closest.

$$r_{nk} = \begin{cases} 
1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|^2 \\
0 & \text{otherwise}
\end{cases}$$
The second step, $\mu_k$, is updated by keeping $r_{nk}$ fixed. In other words, $\mu_k$ is re-estimated based on the current grouping of data points, this is done by basic mean computation of the data points assigned to each cluster.

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

After these two steps, the squared-error distortion measure is updated with the new parameters. This is repeated until convergence, where the change between two iterations is below a certain threshold set by the user. The K-means algorithm doesn’t guarantee a global optimal solution. The clustering procedure is illustrated with M-steps and E-steps explained in Figure 5 on page 9.

### 2.5.2 Expectation Maximization

**Mixture of Gaussians** To motivate the expectation maximization, a good foundation in the role of latent variables in mixture models is required. The gaussian mixture model is written as weighted gaussian distributions in the form

$$p(x) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \Sigma_k).$$  \hspace{1cm} (3)

$K$ is the dimension of the binary random variable $z$ where a particular element $z_k$ is equal to 1 and all others are 0, $z_k$ satisfies $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. The joint distribution $p(x, z)$ is defined in terms of the marginal $p(z)$ and conditional distributions $p(x|z)$. The marginal distribution is specified in terms of the weights $\pi_k$ such that

$$p(z_k = 1) = \pi_k$$

where $0 \leq \pi_k \leq 1$ and $\sum_k \pi_k = 1$. Since $z$ is in 1-of-$K$ representation, the distribution can be rewritten as

$$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}.$$  \hspace{1cm} (4)
Figure 5: Illustration of K-mean algorithm using the re-scaled Old Faithful data set, graphics are from Figure 9.1 [7]. A is the initial step where the crosses $\mu_k$ are initialised, b is the first E step and c the subsequent M step. The M and E steps are repeated through d - i.
The conditional distribution \( p(x|z_k = 1) \) denotes the probability for data points \( x \) to have been generated by the \( k \)th gaussian component, the conditional distribution is also gaussian and can thus be written in the form

\[
p(x|z) = \prod_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)^{z_k}.
\]

Now the joint distribution \( p(x, z) \) can be obtained, by using the product rule of probability, as

\[
p(x) = \sum_z p(z)p(x|z) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k).
\]

What is observed is that (3) can be found by an equivalent reformulation involving latent variables. When \( x \) is several data points \( x_n \) then there is a binary latent variable \( z_n \) for each data point. Another important equation to point out is the conditional distribution

\[
p(z_k = 1|x) = \frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(x|z_j = 1)}.
\]

The equation (7) can be understood as the responsibility that component \( k \) assumes given the observation \( x \).

**Parameter Estimation** Suppose that there is the observation \( X = \{x_1, \ldots, x_N\} \) and the objective is to parametrize this set of data by mixtures of gaussians. The observation \( X \) has the dimensionality of \( N \times D \) where \( N \) is the number of data points and the \( D \) is the dimension of the data. The latent variables are denoted by the matrix \( Z \) with dimensionality \( N \times K \), where \( z_{nk} \) denotes the responsibility of the \( k \)th component to the \( n \)th data point of \( X \). Evaluation of model parameters are done by maximizing the total probability for every single data point. This is done by the use of maximum likelihood formulation where the probability of the parameters corresponding to the data is written in the form

\[
\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left( \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right).
\]

The natural logarithm is a monotonically increasing function and will simplify the expression for further manipulation which is why from now on, many formulations will be put under the logarithm function. To find the maximum of (8), its derivative with respect to the mean \( \mu_k \) is set to 0

\[
0 = \sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \Sigma_k^{-1} (x_n - \mu_k).
\]

It is observed that the posterior probabilities from (7) appear on the right hand side of the equation, these will from now on be denoted by \( z_{nk} \) for convenience. Equation (9) can be rearranged to get an expression for \( \mu_k \) by multiplying with \( \Sigma_k \)

\[
\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} x_n
\]
where $N_k$ is defined as

$$N_k = \sum_{n=1}^{N} z_{nk},$$

where it can be interpreted as the amount of data points assigned to cluster $k$. The mean $\mu_k$ is computed by a simple weighted average over $x$. The covariance matrix is computed in a similar manner by taking the estimated weighted mean and square it

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} z_{nk} (x_n - \mu_k)(x_n - \mu_k)^T.$$  \hfill (11)

The last parameter to compute is the weight $\pi_k$, this is done by maximizing (8) with the added constraint of $\sum_k \pi_k = 1$. Maximization with regard to constraints are done with Lagrange multiplier and the quantity to be maximized by differentiation with respect to $\pi_k$,

$$\ln p(X | \pi, \mu, \Sigma) + \lambda \left( \sum_{k=1}^{K} \pi_k - 1 \right)$$ \hfill (12)

which equates to

$$0 = \sum_{n=1}^{N} \pi_k N(x_n | \mu_k, \Sigma_k) + \lambda = \sum_{j} \pi_j N(x_n | \mu_j, \Sigma_j) + \lambda.$$ \hfill (13)

If from (13), you multiply with $\pi_k$ on both sides and sum over $k$. The reader will notice that $\lambda = -N$ due to the constraint imposed on $\pi_k$. By re-arranging, the equation for estimation of the mixing coefficients is found to be

$$\pi_k = \frac{N_k}{N}.$$ \hfill (14)

With the formulations presented an alternating algorithm, much like the k-means, is devised and can be viewed in algorithm 1 on page 12.
Algorithm 1 EM for Gaussian Mixtures

1: Initialize the means \( \mu_k \), covariances \( \Sigma_k \) and mixing coefficients \( \pi_k \) and evaluate the initial value of the log likelihood.

2: **E step** Evaluate the responsibilities \( Z \), using the current parameter values

\[
z_{nk} = \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \quad \triangleright \text{Eq (7)}.\]

3: **M step** Re-estimate the parameters using the current responsibilities

\[
\begin{align*}
\mu_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N z_{nk} x_n \quad \triangleright \text{Eq (10)}, \\
\Sigma_k^{\text{new}} &= \frac{1}{N_k} \sum_{n=1}^N z_{nk} (x_n - \mu_k^{\text{new}})(x_n - \mu_k^{\text{new}})^T \quad \triangleright \text{Eq (11)}, \\
\pi_k &= \frac{N_k}{N} \quad \triangleright \text{Eq (14)}.
\end{align*}
\]

4: Evaluate the log likelihood

\[
\ln p(X|\mu, \Sigma, \pi) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\} \quad \triangleright \text{Eq (9)},
\]

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

3 Method

3.1 Training data

The reference database has perfect genetic sequences whose length in base-pairs are denoted by \( L_d \). An overly simplistic explanation, each k-mer feature vector is a building block where the objective is to rebuild the sample with these different blocks. The number of each block which is a positive real number \([0, 1]\) will determine the proportion of a known species.

**Aggregation by mean** In the methods [29, 35], the whole genetic signal is captured in one single vector by computing the frequencies of different k-mers as explained in Section 2.3. A reference database with \( m \) taxa \( D = d_1, \ldots, d_m \) would in that case yield the training matrix

\[
X = [x_1, x_2, \ldots, x_m], \quad (15)
\]

where each taxon is processed independently and stored in a training matrix in column wise fashion.

**SEK Training matrix** In [14] an alternate training matrix is proposed, instead of computing the mean of the entire genetic signal. Its k-mer frequencies are split into multiple vectors where sequence data is read in fragments of length \( L_w \) constituting a separate k-mer feature vector. These fragments are read in sliding window manner where the window is slid by \( L_p \) positions from the beginning till the end. The total number of k-mer feature vectors can be computed to \( N \approx \left\lfloor \frac{L_d - L_w}{L_p} \right\rfloor \). The advantage of splitting the genetic sequence into multiple vectors is the retained variability which is believed to help when fitting
the sample distribution with the reference data. Variable $L_w$ determines the k-mer statistics captured by a single feature vector, the choice of this parameter should be done with respect to the length of reads since the objective is to have training data that matches the characteristics of the reads of sample. Finally, column dimension of $X$ is linearly proportional to $L_p$. The reader may have noticed that gap of overlaps occur when $L_w > L_p$ which results in highly correlated feature vectors. It is thus advised to choose $L_p$ with care to maximize efficiency of memory use without losing performance. If maximum information content (variability) is desired then $L_p$ should be set to 1. The different taxa are represented as

$$X = [X_1 X_2 \ldots , X_m] \in \mathbb{R}^{4^k \times N},$$

≡ $[x_1 x_2 \ldots , x_n]$, (16)

where $X_m \in \mathbb{R}^{4^k \times N_m}$ corresponds to the matrix with all of the k-mer feature vectors of $m$th taxon. The training matrix $X$ has $N$ columns where these are the sum of the concatenated taxon specific training matrices, $\sum_{m=1}^{M} N_m = N$, and $x_n \in \mathbb{R}^{4^k \times 1}$ which corresponds to the $n$th feature vector of the full training matrix $X$.

### 3.2 Simultaneous supervised classification

The concept of simultaneous classification in bacterial community composition stems from the work [35], where Meinicke et al proposed a mixture model based profiling of taxa. Instead of doing read specific classification such as the case with the RDP classifier, the whole sample is modelled as a composition of oligonucleotides (k-mers). The computation task is then to reconstruct the sample by weighing different organism specific oligonucleotides from a reference database. The underlying theory is very attractive for this problem statement by skipping the classification of reads which could potentially be enumerated to the millions and reconstructing the sample directly, obtaining taxa weights implicitly.

#### 3.2.1 Problem modelling

The mixture model based approach has attracted considerable attention for various different implementations [35, 14, 4, 3, 47, 29] to estimate taxonomic profile of a microbial sample. This section will present the underlying mathematical formulation of these implementations.

The probability of the $m$th taxon will be denoted by $p(C_m)$ and $x_{ml}$ represents the $l$th feature vector of the $m$th taxon which can now be expressed as the conditional mixture density function

$$p(x|C_m) = \sum_{l=1}^{N_m} \alpha_{ml} p_m(x|\Theta_{ml}),$$

where $\alpha_{ml} \geq 0, \sum_{l=1}^{N_m} \alpha_{ml} = 1$. Each feature vector $x_{ml}$ is drawn from an unknown distribution and is thus assumed to be it’s own mean value from stan-
standard kernel density methods [7]. For convenience \( p_{ml} \) could be any parametric or nonparametric distribution.

The reads are unlabelled and modelled as a composition of known species in reference

\[
p(x) = \sum_{m=1}^{M} p(C_m) p(x|C_m),
\]

\( p(C_m) \) is the proportion of taxon \( m \) in a sample where \( p(C_m) \geq 0 \) and \( \sum_{m=1}^{M} p(C_m) = 1 \). The inference task is to compute \( p(C_m) \) therefore it is possible to make use of the first order of moments to evaluate the sample mean of measurement data \( p(x) \)

\[
E[x] = \int x p(x) \, dx \in \mathbb{R}^{4 \times 1}
\]

\[
= \sum_{m=1}^{M} p(C_m) \int x p(x|C_m) \, dx
\]

\[
= \sum_{m=1}^{M} p(C_m) \int x \sum_{l=1}^{N_m} \alpha_{ml} p_{ml}(x|x_{ml}, \Theta_{ml}) \, dx
\]

\[
= \sum_{m=1}^{M} p(C_m) \sum_{l=1}^{N_m} \alpha_{ml} \int p_{ml}(x|x_{ml}, \Theta_{ml}) \, dx
\]

\[
= \sum_{m=1}^{M} p(C_m) \sum_{l=1}^{N_m} \alpha_{ml} x_{ml}
\]

it's evident that the distributional mean can be approximated by multiplying each column of the training matrix \( X \) with its corresponding weights, \( \alpha_{ml} p(C_m) \). Since the sample mean is already given, a framework has now been devised to solve for these weights.

With the assumption that there is enough reads, the sample mean can be assumed to be the distributional mean

\[
E[x] = \sum_{n=1}^{N} \gamma_n x_n = X \gamma.
\]

The variable \( \gamma \) can be viewed as an auxiliary variable to help with the indexing of \( \alpha_{ml} \) and \( p(C_m) \)

\[
\gamma = [\gamma_1 \gamma_2 \cdots \gamma_N]^t \in \mathbb{R}^{N \times 1},
\]

\[
\gamma_n \triangleq \gamma_{n(m,l)} = p(C_m)\alpha_{ml}
\]

with the following properties

\[
\sum_{n(m,1)}^{n(m,N_m)} \gamma_n = p(C_m) \sum_{l=1}^{N_m} \alpha_{ml} = p(C_m),
\]

\[
\sum_{n=1}^{N} \gamma_n = \|\gamma\|_1 = 1.
\]
The measurement data is processed by standard k-mer feature extraction. The feature vectors are generated and the sample mean of these are computed which is denoted by $\mu \in \mathbb{R}^{4^k \times 1}$. The computation task can now be viewed as the linear problem

$$\mu = X \gamma + n \in \mathbb{R}^{4^k \times 1}. \quad (20)$$

Variable $n$ is additive noise and solving of Eq. (20) for $\hat{\gamma}$ happens under constraints

$$\hat{\gamma} \geq 0, \quad \|\hat{\gamma}\|_1 = \sum_{n=1}^{N} \hat{\gamma} = \sum_{m=1}^{M} \hat{p}(C_m) = 1. \quad (21)$$

Since Eq. (20) is under-determined $4^k < N$ and in general such systems have infinitely many solutions. The constraints (21) will decrease the solution space of the problem and result in a convex optimization task. Estimation of $p(C_m)$ is done by summing up ranges of $\hat{\gamma}$ that partake the $m$th taxon,

$$\hat{p}(C_m) = \sum_{n(m,1)}^{n(m,N_m)} \hat{\gamma}_n. \quad (22)$$

### 3.2.2 Aggregate reads by K-means to improve accuracy

In the previous section, it was seen how the bacterial community composition could be found by solving a linear system of mean values from reference data set and the mean of measurement data set by minimizing the $l_1$ norm. In this section an optimization to the existing problem model is proposed.

Shannon entropy tells us that the information gain is proportional to the uncertainty of a particular outcome and uncertainty generally increases with more variation, requiring several bits to be represented. This intuition indicates that more variability of the measurement data is captured if it would be represented by several vectors rather than the single one which is currently employed in existing methods.

Taxy, Quikr and SEK use sample mean vector of k-mer feature vectors computed from reads as the main input; the mean vector contains the composition information. These three methods do not use the reads in any other way; once the mean vector of k-mer feature vectors is computed, the reads are no more in use. The proposed principle is to segregate the set of k-mers into subsets, compute mean vector for each subset, employ an estimation method (such as Taxy, Quikr or SEK) to find the composition of the subset, and finally fuse the solutions into one for the full data set. The much cited K-means clustering algorithm is suggested due to its algorithmic simplicity, its computationally inexpensive for a reasonable number of $Q$ clusters.

The assumption is that it can be used for a large data set to be partitioned into subsets, providing a set of mean vectors to divide the feature space into $Q$ non-overlapping regions. This is called codebook generation in vector quantization, originally from signal processing and coding.

The new proposed concept of k-mer aggregation is reformulated as

$$p(x) = \sum_{q=1}^{Q} P_q p(x | x \in \mathcal{R}_q) \quad (23)$$
the feature space of the measurement data is divided into $Q$ non-overlapping regions $R_q$ such that $\bigcup_{q=1}^{Q} R_q = \mathbb{R}^d$. Voronoi region/cluster probability is defined as $P_q \overset{\triangle}{=} Pr(x \in R_q)$ such that $\sum_{q=1}^{Q} P_q = 1$. In practice $P_q$ is determined by dividing number of data points within a cluster by the total number of data points:

$$P_q \approx \frac{\text{number of feature vectors in } R_q}{\text{total number of feature vectors}}. \quad (24)$$

Continuing from (18) and rewrite it to include the Voronoi regions

$$p(x|x \in R_q) = \sum_{m=1}^{M} p(C_m|x \in R_q) \cdot p(x|C_m, x \in R_q),$$

which models the bacterial community of the $Q$th region. If the proportions of taxa is known for each region $R_q$ then the weighted sum of the conditional probabilities $p(C_m|x \in R_q)$ is equal to

$$p(C_m) = \sum_{q=1}^{Q} P_q \cdot p(C_m|x \in R_q). \quad (25)$$

The conditional mean vector is found for each region $R_q$

$$E[x|x \in R_q] = \int x \cdot p(x|x \in R_q) \, dx = \sum_{m=1}^{M} p(C_m|x \in R_q) \int x \cdot p(x|C_m, x \in R_q) \, dx. \quad (26)$$

From Section 3.2.1, second step from (19), it is known that $p(C_m)$ can be inferred with existing composition estimation methods [14, 35, 29] given the sample mean $\mu_q$ and the assumption that $\mu_q \approx E[x|x \in R_q]$.

### 3.2.3 Improving computation time by reducing dimensionality

Previously, clustering was proposed as a means to aggregate k-mer information. This idea was put in perspective as a way of improving classification accuracy by increasing the dimensionality of the input data. This clustering comes at a computational cost that is tied to the number of reads in measurement data, leading to run times that aren’t feasible for data sets with millions of reads. In the SEK model, it is proposed to not aggregate reference data and keep all k-mers feature vectors intact. The hypothesis is that the performance of SEK comes from this retained variability [14] of all taxa in the training matrix $X$.

Two assumptions can be drawn from performing k-means on the SEK training matrix.

1. With aggregation, the performance will take a hit in terms of VD but computation times will improve.

2. The new training matrix will increase the dimension of the training data and thus give a performance boosts at the cost of slightly added run times for Quikr and Taxy.
Aggregation of reference data is done with the loss of variability, the number of clusters will determine how much variance of training data is kept and the appearance of the training matrix. In composition estimation methods of [29, 35] the linear systems are over determined. By clustering reference it is possible from user input to set the different properties of the whole classification system. The estimation method of [14] has a complexity tied to the dimension of the data $O(4^k I^3)$ where $I << 4^k$ and the number of feature vectors are $4^k << N$.

In [14], a comparison of solvers, Chatterjee et al makes the remark that the computational complexity of $P_{sec}^{+}$ is tied to the column dimension $O(N^3)$, thus with an overdetermined matrix $X \in \mathbb{R}^{4^k \times N}$ where $N << 4^k$ is more preferable to use conventional convex solving systems such as [22] to solve composition problems. The loss of variability could be accounted for by employing high dimensional k-mers.

### 3.2.4 Alternate Optimization to improve accuracy

The proposed improvements on the original problem formulation have so far been concerned with the dimensionality of training and input data. In the mixture model (18), the variable $\alpha_{ml}$ is the weight of the different k-mer feature vectors, in biological terms it is the amplification of a variable sequence region. By clustering reference data, much of the k-mers are aggregated. The problem is that there is no certainty that the biological interpretation agrees with the intuitive mathematical one. Therefore an alternate optimization technique is proposed. After the community composition estimation is done, the composition solution $\hat{p}(Cm)$ is fixed to re-estimate the $\alpha_{ml}$. This is repeated until no change is observed in the taxonomic composition. If the reader recalls (20), the vector $\gamma$ is the joint distribution of $\alpha_{ml}$ and $p(Cm)$. After the initial estimation of $\hat{\gamma}(C)$, $\alpha_{ml}$ is re-estimated by keeping taxa proportions $p$ fixed

$$\hat{\alpha}^{\text{new}} = \arg \min_{\alpha} \|\mu - Xp^{\text{old}}\alpha\|_2$$

s.t. $\alpha_m \geq 0, \|\alpha_m\|_1 = 1,$

(27)

where the vector $\alpha_m$ is the cluster probabilities of the $m$th taxon. When new cluster weights have been obtained, the taxa proportions are found with respect to the new cluster probabilities

$$\hat{\alpha}^{\text{new}} = \arg \min_{p} \|\mu - X\alpha^{\text{old}}p\|_2$$

s.t. $p \geq 0, \|p\|_1 = 1.$

(28)

These two alternating steps are done until no change in taxa proportions are detected

$$\|\hat{\gamma} - p\|_2^2 < \epsilon,$$

(29)

where $\epsilon$ is a user defined tolerance level.

### 3.3 Semi-supervised learning

Semi-supervised learning is a class of learning techniques where you make use of unlabelled data for training. The problem formulation in the scope of this project, a reference database is given with previously discovered bacterial species.
In BeBAC [16], the authors solve the problem by first performing several layers of unsupervised clustering, then match the cluster signatures with similar species in the training data. However it is possible to reduce the computation time by starting the clustering much closer to the globally optimal solution. As the reference database is also in k-mer feature space and with the use of the training data it is possible to initialize unsupervised techniques to yield more preferable solutions.

### 3.3.1 Expectation Maximization with Dirichlet Mixtures

As the k-mer feature space is inherit with deficient covariance matrices, which eliminates modelling with mixtures of Gaussians, an alternate parametric model is preferred. The k-mer vectors are vectors of frequencies and as the Dirichlet distribution is a continuous parametric model of how proportions vary, it is hypothesised that it will be a fitting way for parametrizing the underlying data.

Prior to normalizing the count-vector in the k-mer feature extraction scheme, a pseudo count is added to avoid zero-values in the matrix which will permit log transformations, this way of pre-processing has been put to use in [1] for the same purpose. The necessity of this transformation arises from current Dirichlet estimation methods [36] where an added constraint of $0 < \alpha$ takes place.

If the reader recalls, the Expectation Maximization algorithm was presented in Section 2.5.2 with respect to the gaussian as the underlying parametric model. The role of latent variables in mixture models were explored in conjunction with simple estimation techniques for model parameters and the alternate optimization to maximize the entropy/likelihood function of all the data points. The Dirichlet Mixture model

$$p(x|\alpha) = \sum_k \pi_k \text{Dir}(x_n|\alpha_k)$$

and the latent variable formulation as presented in [32]

$$\pi_{nk} = E[z_{nk}] = \frac{\pi_k \text{Dir}(x_n|\alpha_k)}{\sum_j \pi_j \text{Dir}(x_n|\alpha_j)}. \quad (31)$$

The latent variable is estimated in the E step and in the M step the mixing coefficients $\pi_k$ and $\alpha$ are estimated.

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \pi_{nk} \quad (32)$$

Following Eq. 28 in [9], Eq. 15 in [32] and Eq. 8 in [36], $\alpha$ is estimated as

$$\psi(\alpha_{jk}) - \psi(\sum_k \alpha_{jk}) = \frac{\sum_{n=1}^{N} \pi_{nj} \log pk_n}{\sum_{n=1}^{N} \pi_{nj}} \quad (33)$$

where $k$ denotes the dimension and $j$ the Dirichlet component. The $\psi(\cdot)$ is the digamma function $\psi(x) = \delta \log \Gamma(x)/\delta x$. 

18
4 Implementation

4.1 Optimization problem

In Section 3.2.1, the community estimation problem was formulated as a linear system of equations where the inference task was reduced to an optimization problem. Solving of (20) is no trivial task and different compressive sensing algorithms have been found \[29, 14\] to solve
\[
\hat{\gamma} = \arg \min_{\gamma} \| \mu - X \gamma \|_2,
\]
subject to constraints (21). The optimization task is a constrained least squares and a quadratic program \[10\], solvable by convex optimization tools such as \texttt{cvx} \[22, 21\].

In \[14\], Chatterjee et al makes the observation in the community estimation problem that $\hat{\gamma}$ is inherently sparse. Natural explanations follow from the fact that the conditional density (17) only induce a few vectors in the feature space, resulting in mostly zero-value or negligible $\alpha_{ml}$. This sparsity is unstructured in $\gamma$. Also worth elucidating is that in practice, most measurement data contains only a sub set of the expected taxa and thus some values of $p(C_m)$ will turn out to be zero which consequently results in stretches of zeros in the $\gamma$ vector.

Another observation is that the taxonomic composition to highly under-determined systems (20) will result in a solution to (34) where it is sparse due to the constraint $\| \gamma \|_1 = 1$ imposed.

It is possible to reduce the time complexity of the estimation process by using sparsity promoting algorithms. Orthogonal matching pursuit is an iterative greedy algorithm \[42\] that has been extended for various applications and constraints as it uses a least squares approach to solve under-determined linear systems. One of many extensions to it is the OMP$^+_{sek}$ which was devised to fulfill the given constraints (21) of the bacterial community composition problem \[14\].

4.2 K-means clustering

A much cited K-means algorithm called Linde-Buzo-Gray (LBG) algorithm used in vector quantization \[30\] (or source coding literature). The LBG has many variants. In one variant, the algorithm starts with $Q = 1$, one single cluster and then slowly split the dense and high probability clusters to end up with a high $Q$, such that it does not deviate significantly from exponentially decaying bit rate versus coding distortion (R/D) curve. View Algorithm 2 for pseudo-code.

\begin{algorithm}
\caption{Linde-Buzo-Gray - Part 1}
1: $X \in \mathbb{R}^{d \times N}$, $C^{(0)} = X$, $p = p(C^{(0)})$
2: while length$(p) < Q$ do
3: \hspace{1cm} $C^{new}, p^{new} \leftarrow \text{LBG}(X, C^{old}, p^{old})$ \hspace{1cm} $Q$ is the desired number of clusters.
4: \hspace{1cm} $p^{new}$ \hspace{1cm} Increments the number of clusters.
5: end while
\end{algorithm}

The most computationally intensive step of the LBG algorithm, is the nearest neighbour classification of data points. In optimal clustering case, this step
Algorithm 2 Linde-Buzo-Gray - Part 2

1: function LBG($X$, $C$, $p$) \> $p \in \mathbb{R}^{1 \times Q}$
2: $D^{(0)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \min_q \|x_n - c_q^{(0)}\|^2$
3: $i \leftarrow \arg \max_p$ \> index of the cluster with highest probability
4: Split largest cluster:
   \begin{align*}
   Q &\leftarrow Q + 1 \\
   c_i^{(0)} &\leftarrow (1 + \epsilon) c_i^{(0)} \\
   c_Q^{(0)} &\leftarrow (1 - \epsilon) c_i^{(0)}
   \end{align*}
5: $t \leftarrow 0$ \> $t$ is the iteration counter
6: repeat
7: $Q(x_n) \leftarrow \arg \min_q \|x_n - c_q^{(t)}\|^2$ \> Nearest-neighbour classification
8: $p_q^{(t+1)} \leftarrow \frac{1}{N} \sum_{Q(x_n) = c_q^{(t)}} 1$ \> Update cluster weight
9: $c_q^{(t+1)} \leftarrow \frac{\sum_{Q(x_n) = c_q^{(t)}} x_n}{\sum_{Q(x_n) = c_q^{(t)}} 1}$ \> Update codebook vector
10: $t \leftarrow t + 1$
11: $D^{(t)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \|x_n - Q(x_n)\|^2$
12: until $\|D^{(t-1)} - D^{(t)}\|_1 \leq \epsilon$
13: return $C$, $p$
14: end function

is repeated at least twice for each splitting process, initialization, once for incrementing $Q$ and another time for re-estimation of cluster mean. In practice this number will increase and for each clustering step this will be repeated which can be time consuming if a high number of clusters is desired. Therefore another variant of the LBG is designed. By employing random sampling to immediately create $Q$ sub sets of the data set, the number of nearest neighbour classifications are effectively reduced. This is done by random selection of $Q$ data points in the feature space and assign data points to clusters by nearest neighbour rule. This will result in optimal case of two nearest neighbour computations for $Q$ regions, the downside is the random nature of the clustering which cannot be reproduced or ensured to have even cluster probabilities. Pseudo-code for the randomized version of LBG can be viewed in Algorithm 3.

### 4.3 Simultaneous supervised classification

#### 4.3.1 Aggregation of reads by k-means

The ARK algorithm can be implemented by following steps.

a. Divide full test set into $Q$ subsets; The region $\mathcal{R}_q$ corresponds to the $Q$th subset.

b. For the $Q$th subset, compute $P_q$ and sample mean $\mu_q$.

c. For the $Q$th subset, apply a composition estimation method which requires the main input as $\mu_q$; estimate $p(C_m | x \in \mathcal{R}_q)$.

d. Estimate $p(C_m)$ by $p(C_m) = \sum_{q=1}^{Q} P_q \cdot p(C_m | x \in \mathcal{R}_q)$. 
Algorithm 3 Random init LBG

1: procedure LBG2(\(X, Q, T\))  \(\triangleright X \in \mathbb{R}^d \times N\)
2: \(C \leftarrow Q\) randomly selected row-vectors from \(X\)
3: \(D^{(0)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \arg \min_q \|x_n - c_q\|^2\)
4: \(t \leftarrow 0\) \(\triangleright\) Initialize iteration counter
5: repeat
6: \(Q(x_n) \leftarrow \arg \min_q \|x_n - c_q\|^2\) \(\triangleright\) Nearest cluster \(c_q\) to point \(x_n\)
7: \(c_q^{(t+1)} \leftarrow \frac{\sum_{Q(x_n) = c_q^{(t)}} x_n}{\sum_{Q(x_n) = c_q^{(t)}} 1}\) \(\triangleright\) Update mean values of clusters
8: \(t \leftarrow t + 1\) \(\triangleright\) Increment counter
9: \(D^{(t)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \|x_n - Q(x_n)\|^2\)
10: until \((\|D^{(t-1)} - D^{(t)}\|_1 \leq \epsilon)\) or \((t \geq T)\)
11: end procedure

Challenges: The natural challenges are as follows.

1. How many regions? That means, what is \(Q\)?
2. How to create the subsets? That means, how to form \(R_q\)?

The above points are inherent to any subset forming algorithm, more generally to any clustering algorithm. Further, finding optimum regions (or clusters) require alternative optimization techniques. Given a pre-defined \(Q\), typically a K-means algorithm does two optimization steps in alternate. They are: (a) with the knowledge of a set of representation vectors (also called code vectors) forming new clusters by nearest neighbour rule (or forming new subsets from full dataset), (b) finding the set of cluster representation vectors (called codebook) with the knowledge of clusters; the optimal representation vector is the mean vector if square Euclidean distance is used for nearest neighbour rule. The K-means algorithm starts with an initialization of set of representation vectors and runs alternating optimization until convergence in the sense of average square Euclidean distance does not show a significant reduction trend.

In ARK, two strategies are used to the two challenges, as follows.

1. Optimal strategy: Start with \(Q = 1\). That means the existing standard approach which uses mean vector of the full test set as the aggregator. Then \(Q = 2\) is set for LBG algorithm, see Algorithm 2, that uses Euclidean distance as the distortion measure. Initialization is done by a standard split approach where the mean vector is perturbed. Finally for \(Q = 2\), \(\{R_q\}_{q=1}^{2}\) is formed and \(p(C_m)\) is estimated. The highest probable cluster is always split as the initialization and use of the LBG algorithm to find optimized clusters. Increasing of \(Q\) is stopped if estimation of \(p(C_m)\) for \(Q\) and \((Q-1)\) remain almost same, that means when a saturation in estimated \(p(C_m)\) is observed. In practice, the \(l_1\) norm between \(p(C_m)\) for \(Q\) and \((Q-1)\) is used as a stopping condition, if it is less than the stopping threshold; in mathematical notation \(\sum_{m=1}^{M} \|p(C_m)|_Q - p(C_m)|(Q-1)\| < \epsilon\), with a user defined choice of \(\epsilon\).

This optimized strategy is believed to provide consistent performance improvement with increase in \(Q\). Further the increment in number of clus-
System matrix generation
computation
Computing composition
Reads
On−line computation
Reference (training sequences)
Off−line computation
(test sequences)

Initialization: \( Q = 1 \)

\( \mu_q \), \( P_q \) computation

\( p(C_m|k \in R_q) \) computation by Taxy or Quikr or SEK

Computation composition proportion \( p(C_m) \)

Until stopping condition met

\( Q = Q + 1 \)

\( Q = Q_{\text{max}} \)

At this point, it must be mentioned that nothing can be said about global optimality (or absolute consistency in performance improvement), and there is no absolute guarantee that estimation of \( p(C_m) \) is bound to improve with increase in \( Q \). But, overall, a higher \( Q \) is expected to perform reasonably better than a much lower \( Q \).

2. Non-optimal strategy: For a very large testset, see Algorithm 3, a predetermined \( Q \) is used with a random choice of \( Q \) representation vectors, which are initially \( Q \) random data points in the test set. Then divide the full test set into \( Q \) subsets by nearest neighbour rule, and compute the set of \( Q \) mean vectors \( \{\mu_q\} \), and cluster probabilities \( \{P_q\} \). It is worth noting that this strategy has the advantage of being able to start at a user defined number of clusters. The disadvantage of it is the random nature of the clustering, there exists no correlation between the different cluster partitions meaning this method cannot be employed in the same manner as with the optimal strategy since convergence is not guaranteed. Generally for this type of method, a very high number of clusters are chosen directly with the assumption that the number of clusters are sufficient and the clustering will capture most of the variability in the data.

Finally, a flow-chart of the full ARK-SEK system depicting the associated on-line and off-line computations is shown in Figure 6 on page 22.
4.3.2 Dimension reduction of training data

Aggregation of reference data with k-means have the same natural clustering challenges that were shed to light in Section 4.3.1. One of the advantages and also disadvantages of reference clustering is that the aggregation is taking place off-line, prior to the actual composition estimation. This can be problematic as there is no way to ensure if an optimal number of clusters have been achieved in the same manner presented in the ARK formulation. Choice of clusters will have to be ad hoc, a system property set by the user or possibly in regard to the variability of the greatest spanning feature space $X_m$. The advantage of performing clustering offline, is that the computation time will not count against the time for estimation of the taxonomic profile. The offline computation is done only once and for this reason, more complex and better performing clustering algorithms could be employed. But for the sake of continuity, in the scope of this report, the two variants of LBG that were put forth in Section 4.2 will be utilized.

Reducing the column dimension of the training matrix $X$ can be done in the following steps:

a. Perform clustering on $X_m$, the feature space of the $m$th taxon, with either algorithm 2 or 3 to form the set of non-overlapping regions $\{R_q\}$.

b. Reform the feature space of $X_m$ as the set of mean vectors computed from $\{R_q\}$ in column wise fashion $X_m = [\mu_1 \mu_2 \ldots \mu_q]$. Concatenate the new $X_m$ to the training matrix $X = [X_1 X_2 \ldots X_m]$.

c. Save the cluster weights in the transformation matrix $T \in \mathbb{R}^{(M \times Q) \times M}$.

The role of the transformation matrix is to avoid the computation step of Eq. (22) since it is computationally ineffective. It is preferred to vectorize this operation by matrix multiplication. The transformation matrix $T$ for $M = 3$ (number of taxon) and $Q = 2$ (number of clusters) would assume the shape

$$T = \begin{bmatrix} \alpha_{1,1} & 0 & 0 \\ \alpha_{1,2} & 0 & 0 \\ 0 & \alpha_{2,1} & 0 \\ 0 & \alpha_{2,2} & 0 \\ 0 & 0 & \alpha_{3,1} \\ 0 & 0 & \alpha_{3,2} \end{bmatrix}. $$

(35)

With the introduction of $T$, the role of $\gamma$ is rendered obsolete and the linear system can be expressed as

$$\mu = X \ T \ p + n \in \mathbb{R}^{k \times 1},$$

(36)

where $p \in \mathbb{R}^{M \times 1}$ is the column vector with the probabilities $p(C_m)$. It is important to point out that the use of $\alpha_{ml}$ to express the cluster weights is not correct and is infused with error. The interpretation of $\alpha_{ml}$ is the amplification of a variable sequence region and its probability of occurrence in the sample. Since the $\alpha_{ml}$ is computed from reference, it is not possible with high certainty to assert its correctness and performance loss may occur from this fact.
4.3.3 Alternate optimization

The optimization of mixture parameters $\alpha_{ml}$ and $p(C_m)$ takes place after initial estimation of the taxonomic composition. Optimization occurs by solving the equations (27) and (28) in alternating steps. Re-estimation of $\alpha_{ml}$ is problematic in the sense that it is time consuming. This reasoning is motivated by the taxon specific constraints imposed. There exists no solvers that can solve a linear equation with different constraints for specific stretches of the inferred vector. One might argue that $\|\alpha\|_1 = M$ but this doesn’t imply that $\|\alpha_m\|_1 = 1$ is satisfied. Despite these hurdles, it is possible to split the estimation into $M$ parts that can be performed independent of each other, where the cluster weights for each taxon are found $\alpha_m$. After initial estimation of $p(C_m)$, alternating optimization can be performed in the following steps:

a. Estimation of $\alpha_m$ happens by solving

$$\arg \min_{\alpha_m} \|E[x] - X_m T_m \alpha_m\|_2$$
$$\text{s.t. } \|\alpha_m\|_1 = 1, \alpha_m \geq 0.$$ (37)

The diagonal matrix $T \in \mathbb{R}^{Q \times Q}$ is plugged with the a priori probability $p(C_m)$ of the $m$th taxon. This step will have to be performed once for each taxon and the results are concatenated in the column vector $\alpha$.

b. New composition solution $p(C_m)$ is found by solving

$$\arg \min_{p} \|E[x] - X A T p\|_2$$
$$\text{s.t. } \|p\|_1 = 1, \ p \geq 0.$$ (38)

The diagonal matrix $A \in \mathbb{R}^{(Q \times M) \times (Q \times M)}$ is plugged with all of the cluster weights $\alpha_{ml}$ which are summed by matrix multiplication of transformation matrix $T$ whose entries constitute of $\{0, 1\}$. Structure of $T$ can be reviewed in Eq. (35). This multiplication takes place to sum the ranges $\alpha_{ml}$ for each cluster to efficiently compute $p(C_m) \sum_{l=1}^{Q} \alpha_{ml}$.

Convergence appears when no reduction of variance in composition is observed, $\|p^{(t-1)} - p^{(t)}\|_1 < \epsilon$. As a measure of pre-caution an iteration counter is implemented where the looping is stopped after $I$ iterations which is ad hoc, defined by user needs. Furthermore there is no assumption that a much higher iteration will yield a significantly better solution.

Another aspect to keep in mind, by this strategy (solving in parts), the $\alpha_m$ is inferred from the full sample mean which contains information of the other taxa. It is believed that this error will be negligible as the same erroneous process is repeated for each taxon. Optimistically, in certain circumstances the feature spaces of the taxa could be disjoint in a way that resulting reconstruction will have a greater residual vector. In ordinary cases this would be an unwelcome side-effect but since the purpose is to implicitly find the $m$th taxon’s cluster weights by reconstructing its feature space, this is only positive.
4.4 Semi-supervised learning

Previously clustering has been put to use as a mean of aggregating information. Another way of finding community composition is by clustering the species data points and use the cluster probabilities as the final species proportions. The role of the clustering in this sense, to classify data points by similarity either in probabilistic setting with respect to a certain parametric model or by euclidean distance measure. Probabilistic clustering deals with overlapping data points in an accurate way but with an imposed computational cost. Two methods have been proposed in the semi-supervised learning framework.

4.4.1 EM with Dirichlet Mixtures

For Dirichlet parameter estimation, the fastfit toolbox was used [37]. It has been used to initialize the EM algorithm by fitting reference species data to single Dirichlet distribution and inverting the digamma function which is done numerically, this is the scope of fastfit methods used. Prior to training Dirichlet distribution, a pseudo-count of 1 was added to the non-normalized k-mer feature vectors. The Dirichlet taxon feature space will be denoted by $X^*_m$. The whole process can be summerized as described in algorithm 4.

**Algorithm 4 Dirichlet Mixture Estimation**

1: $x \in \mathbb{R}^{N \times 4^k}, X^*_m \in \mathbb{R}^{N \times 4^k}$
2: Initialization:
   - Train $M$ Dirichlet distributions $\text{Dir}(\alpha_m | X^*_m)$ on taxon feature space.
   - Run SEK with $X$ to estimate mixing coefficients $\pi_k$ of the Dirichlet Mixture.
3: repeat
4:   E Step, compute responsibilities $z_{nk} = \frac{\pi_k \text{Dir}(x_n | \alpha_k)}{\sum_{j} \pi_j \text{Dir}(x_n | \alpha_j)}$
5:   M Step, compute Mixture parameters $\pi_k = \frac{1}{N} \sum_{n=1}^{N} z_{nk}$
   - $\psi(\alpha_{jk}) = \psi(\sum_k \alpha_{jk}) + \frac{\sum_{n=1}^{N} z_{nk} \log p_k}{\sum_{n=1}^{N} z_{nk}}$ \(\triangleright\) This requires inverting the digamma function.
6: until Sum of all responsibilities $z_{nk}$ are maximized
7: Print mixing coefficients $\pi$ as final composition.

4.4.2 Semi-supervised K-means

The semi-supervised K-means is an re-iteration of the EM with Dirichlet proposition. With the Dirichlet model, data points are given a probability to each Dirichlet which is pre-trained on the reference data base. This probability is the basis for measuring similarity. Given the added complexity of the Dirichlet and the increased computation, a more simple approach with the same intuition is implemented. Clustering is done with respect to standard density kernel methodology(non-parametric) and the assumption that taxa are separable in high dimensional k-mer feature space, where $k \geq 4$. Initialization is done in the same manner as with the Dirichlet but the pre-training involves computing
mean vectors from reference database unlike with the Dirichlets, where estimation of $\alpha$ took place. With the obtained mean vectors, a K-means clustering procedure is initialized where the initial centroids are those computed from the training data.

The task is now reduced to an alternate optimization problem formulation where the objective is to track how many of the data points belonging to the $m$th cluster which corresponds to the $m$th taxon. The full algorithm can be viewed in Algorithm 5.

**Algorithm 5 Semi-supervised K-means**

1: $x \in \mathbb{R}^{N \times 4k}, X_m \in \mathbb{R}^{N_m \times 4k}$ $\triangleright$ $x$ is the test data and $X_m$ is the training data.

2: Initialization:
   - $c_{ml} \leftarrow \text{LBG2}(X_m, Q, I)$ $\triangleright$ $Q$ is the number of clusters desired.
   - $c_m \leftarrow \frac{1}{N} \sum_{n=1}^{N} \text{arg min}_{ml} \|x_n - c_{ml}\|^2$ $\triangleright$ Sample mean of $X_m$ if $Q = 1$.
   - $D(0) \leftarrow \frac{1}{N} \sum_{n=1}^{N} \|x_n - c_{ml}\|^2$ $\triangleright$ Initialize iteration counter

3: $t \leftarrow 0$

4: repeat
   - $Q(x_n) \leftarrow \text{arg min}_{ml} \|x_n - c_{ml}\|^2$ $\triangleright$ Nearest cluster $c_{ml}$ to point $x_n$
   - $c^{(t+1)}_{ml} \leftarrow \frac{\sum_{Q(x_n) = c^{(t)}_{ml}} x_n}{\sum_{Q(x_n) = c^{(t)}_{ml}} 1}$ $\triangleright$ Update mean values of clusters
   - $t \leftarrow t + 1$ $\triangleright$ Increment counter
   - $D^{(t)} \leftarrow \frac{1}{N} \sum_{n=1}^{N} \|x_n - Q(x_n)\|^2$
   - until ($\|D^{(i-1)} - D^{(i)}\|_1 \leq \epsilon$ or $i \geq I$)

5: Print cluster probabilities $p(C)$ as final composition.

The choice of $Q$ and $I$ are ad hoc, mainly determined by user choice or complexity reasons. Worth noting that this solution is greatly dependant on the number of taxon in sample. In practice, training data sets of thousands taxa are used, this would result in equally many nearest neighbour computations which is expected to be very time consuming. A way to reduce the complexity is to only perform necessary computations on taxa that are believed to be present, by performing for example SEK and only include non-zero abundancies.
5 Results

5.1 Experimentation setup

Computational resources All of the testing apart from those conducted with the simulated data were performed on a stationary computer with a 3.2 GHz processor and 8 GB memory. The in silico testing was performed with desktop workstation with an Intel Core i7 4930K processor and 64Gb of RAM. Least-squares with non-negativity problems were solved with cvx [22, 21] and the built in Matlab function lsqnonneg(). Dirichlet estimations were solved by the fastfit toolbox [37, 36].

Performance measure As a quantitative performance measure, variational distance (VD) is used to compare between known proportions of taxons \( \mathbf{p} = [p(C_1), p(C_2), \ldots, p(C_m)]^T \) and the estimated proportions \( \mathbf{\hat{p}} = [\hat{p}(C_1), \hat{p}(C_2), \ldots, \hat{p}(C_M)]^T \). The VD is defined as

\[
VD = 0.5 \times \| \mathbf{p} - \mathbf{\hat{p}} \|_1 \in [0, 1].
\]

A low VD indicates more satisfactory performance. Results from SEK, Quikr and Taxy are compared for real biological data (mock communities data).

5.2 Data sets

Mock communities data For experiments on real biological data, the mock microbial communities database developed in [25] was used. The database is called even composition Mock Communities (eMC) for chimeric sequence detection where the involved bacterial species are known in advance. Three regions (V1-V3, V3-V5, V6-V9) of the 16S rRNA gene of the composition eMC were sequenced using 454 sequencing technology in four different sequencing centers. The experimentation involved focused on the V3-V5 region datasets, since these have been earlier used for evaluation of the BeBAC method (see Experiment 2 of [16]). The test dataset consists of 91240 short length reads from 21 different species. The length of reads has a range between 450-550 bp and the bacterial community composition is known at the species level.

Simulated data For a more diverse test setting, ARK methods were tested with simulated data which consists of 180 different 16S rRNA data sets targeting V1-V2 regions or V3-V5, using the shotgun/amplicon read simulator Grinder [5]. Read-length distributions were set to be one of the following: fixed at 250bp or normally distributed at 450bp ± 50bp. Read depth were chosen to be 10K, 100K or 250K. Three different diversity values were chosen (50, 100, and 500) at the species level and abundance was modelled by one of the following three distributions: uniform, linear, or power-law. Chimera percentages were set to either 5% or 35%. Balzer model was used to simulate homopolymer errors [6] and copy bias was included while length bias wasn’t. Reference data used for the simulated data testing is identical to RDP’s NBC training set 7 and consists of 10046 sequences covering 1813 genera.
5.3 Simultaneous supervised classification

For testing of the ARK-methods, two clustering approaches were chosen as described in the implementation section, one deterministic and one random sampling k-means were used. The deterministic clustering technique requires running the estimation and splitting methods K times for K clusters while the random sampling version allows one to directly run the ARK method with K clusters.

5.3.1 ARK results for mock communities data

The shortest read length for this data set is 450bp so choice of $L_w$ was set to 450bp and $L_p = 1$ for maximum variability. Experiments were conducted with for $k$ equal to 4, 5 and 6. The choice of parameters for the estimators of SEK and Quikr were in their original parameter setting proposed during testing of these in respective papers. Figure 9 shows the decreasing VD as a function of the number of clusters with the use of the deterministic strategy proposed in
Figure 8: Comparison of ARK-SEK, SEK and BEBaC against ground truth. BEBaC, ARK-SEK and SEK provide VD performance 0.0038, 0.0103 and 0.0305 respectively.
Figure 9: Use of optimal deterministic clustering strategy on mock communities data and max number of clusters set to 32. ARK-SEK never converged but this is due to choice of $\epsilon = 5 \times 10^{-4}$. ARK Quikr and Taxy converged with 14 and 18 clusters. The fastest of the three was ARK-Quikr with 62 seconds, then ARK-Taxy with 147 seconds and lastly comes ARK-SEK with 169 minutes.

Section 4.3.1. The final cluster probabilities were spread very even, meaning that the clustering was retaining the information content of the sample data. Performance results overall were good, at lowest the VD would drop to 0.010 or less which if you put in perspective is closing the gap of the state-of-art method BeBAC with its VD of 0.0038, see Figure 8. Quikr couldn’t recreate the low-abundance species prior to clustering, Figure 7. Both Taxy and Quikr assume 16S rRNA statistics are stationary hence the small training matrix which results in difficulties to detect variance in overlapping or similar species, *S. aureus* and *S. epidermidis* are still missclassified amongst each other.
5.3.2 ARK results for simulated data

All plots are of the mean variational distance error over all 180 experiments. The k was set to 6, optimal setting for Quikr and the rest of the parameters were in their original setting. The difficulty with this test set is the extra number of taxon in the reference data, unlike the mock communities where only 21 species were present in sample and reference, the taxa aren’t known in advance for a given test. From initial testing of simulated data and Mock Communities data set it was evident that an increase of clusters brings an improvement in reconstruction accuracy. The decrease of error behaved in a power law kind of curve as, Figure 11, in relation to the clusters resulting in insignificant boost in reconstruction error in relation to the run-time with use of the deterministic approach. All the in silico testing used the random sampling version of K-means with maximum number of clusters set to 75 to keep the computation times under an hour. For both the SEK and Quikr methods, ARK caused a significant decrease in error. The performance boost from ARK for Quikr was much more significant than SEK, see Figure 12. Apart from the improved reconstruction accuracy, the ARK methodology also increases the precision for a high number of taxon which is the case when finding compositions at the genus level, Figure 10.
Figure 11: Results for the random K-means clustering on the simulated data. (a) The left panel: Mean VD error at the genus level as a function of the number of clusters. Note the improvement that ARK contributes to each method. (b) The right panel: Mean execution time increase (factor given in comparison to running SEK or Quikr in the absence of ARK) as a function of number of clusters. The dashed line represents a line with slope 1.

Figure 12: Comparison of the underlying algorithms with and without ARK. Results are for the random K-means clustering on the simulated data when fixing the number of clusters to 75. (a) The left subfigure: Mean VD error at the genus level. (b) The right subfigure: Box and whisker plot of the individual simulated sample execution times. Mean execution times for Quikr and ARK Quikr were 1.75 seconds and 4.71 minutes, while for SEK and ARK SEK they were 21.26 seconds and 19.21 minutes respectively.
5.3.3 Results for dimension reduction of training data

Testing was done on mock communities data with methods, Quikr, SEK and Taxy. These however had modified training data sets. The SEK training matrix was clustered for each taxon as explained in Section 4.3.2. The random-sampling LBG algorithm was used for the clustering, see Algorithm 3 with its iteration counter limit set to 25. Cluster size chosen were 4, 8, 16, 32 and 64 respectively, meaning the column dimension of the training matrix $X$ was effectively reduced to 84, 168, 336, 672 and 1344, this from 121412. Variable $k$ was chosen to 4, 5 and 6. The results from Table 1 are expected, the reconstruction time is indeed reduced by clustering the reference data. The discrepancy of Taxy’s computation time against Quikr can be explained that the latter is implemented with Matlab's built in solver whilst Taxy is using cvx.

Overall the VD performance is worse than expected, to determine the source of this error, same test with 5-mer data on Quikr was conducted but without plugging the cluster probabilities into the transformation matrix $T$, Eq. (35), the results are displayed in Table 1d.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Seconds</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.6883</td>
<td>0.3658</td>
</tr>
<tr>
<td>8</td>
<td>0.8174</td>
<td>0.4307</td>
</tr>
<tr>
<td>16</td>
<td>2.0172</td>
<td>0.4630</td>
</tr>
<tr>
<td>32</td>
<td>3.3993</td>
<td>0.4825</td>
</tr>
<tr>
<td>64</td>
<td>7.5553</td>
<td>0.4895</td>
</tr>
</tbody>
</table>

(a) SEK with 5-mer data.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Seconds</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.0102</td>
<td>0.3658</td>
</tr>
<tr>
<td>8</td>
<td>0.0155</td>
<td>0.4307</td>
</tr>
<tr>
<td>16</td>
<td>0.0490</td>
<td>0.4630</td>
</tr>
<tr>
<td>32</td>
<td>0.1200</td>
<td>0.4825</td>
</tr>
<tr>
<td>64</td>
<td>0.5072</td>
<td>0.4895</td>
</tr>
</tbody>
</table>

(b) Quikr with 5-mer data.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Seconds</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.4940</td>
<td>0.3674</td>
</tr>
<tr>
<td>8</td>
<td>0.9707</td>
<td>0.4313</td>
</tr>
<tr>
<td>16</td>
<td>2.9126</td>
<td>0.4631</td>
</tr>
<tr>
<td>32</td>
<td>9.7578</td>
<td>0.4825</td>
</tr>
<tr>
<td>64</td>
<td>24.7534</td>
<td>0.4895</td>
</tr>
</tbody>
</table>

(c) Taxy with 5-mer data.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Seconds</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>0.0109</td>
<td>0.1398</td>
</tr>
<tr>
<td>8</td>
<td>0.0158</td>
<td>0.0892</td>
</tr>
<tr>
<td>16</td>
<td>0.0490</td>
<td>0.0579</td>
</tr>
<tr>
<td>32</td>
<td>0.1172</td>
<td>0.0471</td>
</tr>
<tr>
<td>64</td>
<td>0.4972</td>
<td>0.0253</td>
</tr>
</tbody>
</table>

(d) $\alpha_{ml} = 1$ in matrix $T$.

Table 1: Time and VD measured for different training matrices.

The performance boost does indeed confirm that the error source was originating from the poor $\alpha_{ml}$ values.

5.3.4 Results for alternate optimization

Testing of the alternate optimization continues where the experiments for reference clustering left off with the exception that all convex optimizations are solved by cvx. Choice of parameters are exactly the same as with Taxy. Maximum number of iterations possible is set to 20 and testing will look at the impact of the number of iterations and if the alternate optimization does indeed improve the overall results. The training data will be clustered with the deterministic LBG variant. Table 2d, page 35, shows a clear pattern, the optimization converges rather quickly, within 5 iterations, which is positive but the rate of change doesn’t reduce. At times the algorithm will get stuck between two
modes where it would alternate indefinitely if not stopped explicitly. There is also no clear connection into how much of an increase of performance one might expect rather the result will improve but the degree to which it does is unclear. The disparity of the VD performances between 4 and 5-mers can only be explained by the information content captured by the clustering, the cluster setting just happens to be better with 4-mers and 64 clusters. Worth noting is that the improvement here is with the erroneous $\alpha_{ml}$. In the last experiment conducted with the reference clustering, the convex optimization formulation was changed by excluding $T$ and solving $X\gamma$ directly which gives the solver more freedom. The role of $T$ without the cluster probabilities, was to sum $\gamma$. Good results were observed, better than SEK despite the tall training matrix. These good results are deteriorated when the alternate optimization is applied, in fact from a VD of 0.0309, with 4-mers and 64 clusters, to a VD of 0.1604. It leads to believe that this optimization method, although theoretically sound, doesn’t correspond well in practice.
<table>
<thead>
<tr>
<th>ΔP</th>
<th>VD</th>
<th>ΔP</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4008</td>
<td>0.4578</td>
<td>0.3615</td>
<td>0.4710</td>
</tr>
<tr>
<td>0.2079</td>
<td>0.4087</td>
<td>0.2309</td>
<td>0.4102</td>
</tr>
<tr>
<td>0.1880</td>
<td>0.3971</td>
<td>0.0723</td>
<td>0.4086</td>
</tr>
<tr>
<td>0.5622</td>
<td>0.1998</td>
<td>0.0680</td>
<td>0.3958</td>
</tr>
<tr>
<td>0.1691</td>
<td>0.1777</td>
<td>0.2943</td>
<td>0.3134</td>
</tr>
<tr>
<td>0.0808</td>
<td>0.1614</td>
<td>0.1155</td>
<td>0.3309</td>
</tr>
<tr>
<td>0.0661</td>
<td>0.1652</td>
<td>0.0762</td>
<td>0.3446</td>
</tr>
<tr>
<td>0.0559</td>
<td>0.1665</td>
<td>0.0586</td>
<td>0.3350</td>
</tr>
<tr>
<td>0.0602</td>
<td>0.1610</td>
<td>0.0595</td>
<td>0.3452</td>
</tr>
<tr>
<td>0.0597</td>
<td>0.1657</td>
<td>0.0593</td>
<td>0.3350</td>
</tr>
<tr>
<td>0.0579</td>
<td>0.1625</td>
<td>0.0600</td>
<td>0.3459</td>
</tr>
<tr>
<td>0.0552</td>
<td>0.1652</td>
<td>0.0597</td>
<td>0.3349</td>
</tr>
<tr>
<td>0.0552</td>
<td>0.1686</td>
<td>0.0603</td>
<td>0.3463</td>
</tr>
<tr>
<td>0.0607</td>
<td>0.1635</td>
<td>0.0614</td>
<td>0.3346</td>
</tr>
<tr>
<td>0.0536</td>
<td>0.1631</td>
<td>0.0619</td>
<td>0.3465</td>
</tr>
<tr>
<td>0.0546</td>
<td>0.1632</td>
<td>0.0621</td>
<td>0.3345</td>
</tr>
<tr>
<td>0.0552</td>
<td>0.1639</td>
<td>0.0626</td>
<td>0.3466</td>
</tr>
<tr>
<td>0.0543</td>
<td>0.1630</td>
<td>0.0626</td>
<td>0.3345</td>
</tr>
<tr>
<td>0.0568</td>
<td>0.1649</td>
<td>0.0627</td>
<td>0.3466</td>
</tr>
<tr>
<td>0.0578</td>
<td>0.1600</td>
<td>0.0637</td>
<td>0.3343</td>
</tr>
</tbody>
</table>

(a) 4-mer: 64 clusters per taxon. (b) 4-mer: 32 clusters per taxon.

<table>
<thead>
<tr>
<th>ΔP</th>
<th>VD</th>
<th>ΔP</th>
<th>VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4908</td>
<td>0.2692</td>
<td>0.6089</td>
<td>0.1976</td>
</tr>
<tr>
<td>0.3032</td>
<td>0.2161</td>
<td>0.2879</td>
<td>0.1458</td>
</tr>
<tr>
<td>0.0973</td>
<td>0.2035</td>
<td>0.1045</td>
<td>0.1286</td>
</tr>
<tr>
<td>0.0245</td>
<td>0.2026</td>
<td>0.0271</td>
<td>0.1400</td>
</tr>
<tr>
<td>0.0008</td>
<td>0.2028</td>
<td>0.0126</td>
<td>0.1355</td>
</tr>
<tr>
<td>0.0003</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1390</td>
</tr>
<tr>
<td>1.0152e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1354</td>
</tr>
<tr>
<td>3.7312e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
<tr>
<td>2.1348e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1354</td>
</tr>
<tr>
<td>8.0097e-07</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
<tr>
<td>1.3766e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1353</td>
</tr>
<tr>
<td>1.6003e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
<tr>
<td>2.4180e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1353</td>
</tr>
<tr>
<td>2.7274e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
<tr>
<td>1.2342e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1354</td>
</tr>
<tr>
<td>1.3640e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
<tr>
<td>2.1128e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1353</td>
</tr>
<tr>
<td>2.4752e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1390</td>
</tr>
<tr>
<td>2.5006e-05</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1353</td>
</tr>
<tr>
<td>1.8325e-06</td>
<td>0.2027</td>
<td>0.0114</td>
<td>0.1391</td>
</tr>
</tbody>
</table>

(c) 5-mer: 64 clusters per taxon. (d) 5-mer: 32 clusters per taxon.

Table 2: Observation of convergence behaviour of alternate optimization and the impact on performance. VD reduction stabilizes around 5 iterations.
5.3.5 Results for combined system

Ultimately, a final system is put together where all the proposed improvements are combined. The objective of this experiment is to see how good of a reconstruction rate can be achieved with what time. Only the optimal setting will be presented as previous sections have already covered the function of the parameters, this section will only be concerned with the whole.

The maximum number of iterations allowed for the alternating optimization was set to 5. Clustering was done in deterministic manner on both read and reference. Number of clusters was set to either 32, 64 or 128, in various permutations of read and reference. Cluster probabilities were plugged to the transformation matrix $T$. Testing were performed solely on 4 and 5-mers as anything past this would use too much memory and consume time.

For 4-mer data and 64 clusters on both read and reference, the resulting VD was 0.0071, total online time was 25 minutes and the most time consuming part was the optimization step that took 23 minutes. Same experiment although without the optimization was tested with result of 0.0633 in 39 seconds without the pre-clustering meaning that the optimization does indeed have an impact on the faulty $\alpha_{ml}$ values. Online estimation time was effectively halved down to 15 minutes by having set 32 clusters on reads and 64 on reference, the VD wasn’t affected by much, yielding a result of 0.0090. Same experiment but with 128 clusters on read would give a minimal change on VD, 0.0094 at a computation time of 20 minutes.

Finally, same test was carried out but without using the faulty $\alpha_{ml}$. These were set to 1 and the \texttt{cvx} solver was re-formulated to find an exact solution for all of $\gamma$ where the corresponding stretches were then summed to obtain taxa proportions, like SEK but in an overdetermined linear system. The corresponding result of the changes were a VD of 0.0127 with the time of 32 minutes, excluding the pre-clustering time. Furthermore, by upping the dimension of the data to 5-mers (64 clusters on read and reference) the following results came to light, VD of 0.0083 and online computation time of 30 minutes, this leads to believe that the increase of k-mer is unnecessary.

5.4 Semi-supervised learning

5.4.1 EM with Dirichlet Mixtures

Prior to actual implementation of the expectation maximization algorithm, the distribution fit was tested. Quality of distribution fit was done by classification, where each data point of the reads were fed into each Dirichlet to give a probability and the max probability would then be the classified species. The fit of the distribution is also proven with the precision of the Dirichlet for taxon $A.\text{baumannii}$ which reached to 1.9053e+03. In another context this would be worrying but as the training data consisted of fragments from a noise free, perfect length genetic sequence, it is believed that overfitting of representative data isn’t an issue.

The results from classification also showed that the choice of Dirichlets was a well founded choice, nearly every single data point was correct, the score was a VD of 0.0033 in 7 seconds by using 4-mer data. Unfortunately this is as far as it would get. The clustering algorithm couldn’t update the mixing coefficients in a satisfactory manner, they would ruin the initial estimate from SEK at
each iteration. The reason behind this turnout is believed to be due to poor $\alpha$ estimation at the M-step.

5.4.2 Semi-supervised k-means

As the discovery of this algorithm came to be late in the project during the actual report writing hence the testing would only include mock communities data by measuring VD and time. The parameters of choice are k and the number of clusters for each reference taxon. The number of clusters determine the major part of the computations as this multiple determines the number of nearest neighbour classifications to be done. The testing involved k equal to 4 and 5, with 1, 2 and 4 clusters. By employing only the means from each taxon with 4-mers, the VD was 0.0163 at 1.3 seconds, with 2 and 4 clusters per taxon the VD improved to 6.0281e-04 (1.4 seconds) and 5.0416e-04 (1.6 seconds). An interesting development, the final composition had only a single non-zero centroid per taxon, from the initial 4 clusters. When increasing the dimension, so does the distances between the taxa which is emphasized by the fact that the VD improves to BeBAC levels, VD of 0.0012 when increasing k to 5 and setting number of clusters to 1. With the increase of k comes an increase in run-times as the computation increased to 5.8 seconds.
6 Discussion and conclusion

This report has tested various improvements to methods that try to reconstruct the sample distribution with reference data and implicitly infer taxa proportions. Clustering has been proposed as a means of aggregating reads of sample and kmer feature vectors of reference. Testing was conducted on real biological data resulting in increased biological fidelity at a cost of increased computational time, a feasible tradeoff as the total process still takes less than 30 minutes to provide satisfactory results. The in silico testing provides a more diverse and challenging task as methods that try to reconstruct the sample distribution (simultaneous supervised classification) tend to include low-abundance taxon in the final composition which explains the VD discrepancy from the mock communities testing to the simulated data. However in a realistic setting it is often known or it can be assumed what bacteria are present in a sample so the results from the in silico tests are just a means to confirm the general improvement for all possible cases.

Furthermore, unforeseen results came to light, clustering not only affects the general accuracy but also does so with an increased number of taxon in training matrix. This is believed to stem from the fact that clusters tend to naturally separate taxon, reducing overlapping taxa and provide a more sparse expression of the sample data which in turn eases reconstruction of sample distribution for the various estimation methods.

In addition, an alternate optimization technique was tested that gave reliable improvement in the right context. Impact on low and high abundance species must be evaluated to find out when the optimization should take place in order to reduce run-times. Also, due to the sparse nature of the problem, it is possible to optimize run-times further by including sequences and taxon with non-zero probabilities in re-estimation steps.

Ultimately, all these methods were combined to a system that yields results which were slightly better than the ARK-methods but were achieved reliably whilst using less memory, being scalable for high number of reads and no sparsity assumption (overdetermined system).

The report has also explored the use of Dirichlet distribution as a parametric model with mixed success. The Dirichlet is indeed a good fit despite the added noise with the pseudo-count. Reasoning to why the soft clustering didn’t work, recent findings have come to light that the EM estimation of Dirichlet Mixtures is a complicated procedure and a more simple approach has been suggested in [8], where convergence is achieved slower but simplifies the M step by reparametrization. As it is now, the inverting of the digamma function is done numerically which can also be said about the initial estimation of $\alpha$ parameters, these are found with Newton-Raphson.

Finally, a simplistic distance based method was implemented and tested with eyebrow raising results which needs to be tested in a more diverse setting. Further testing with simulated datasets will tell the growth of computation times as the number of taxon increases in training data. Other important questions to be answered are key aspects such as diverging cluster means and their impact on final classification. Can the final centroid be assumed to belong to the same species if the cluster mean has diverged from the initial mean values or is a cutoff rate needed to be implemented and how would this be done? Similarity and cutoff rates have been proposed in previous research and there is an ongoing
debate on when a taxon differs from another.

6.1 Closing remark

The methods of the research conducted were mainly involved with manipulating dimension of training and test data with the use of clustering for aggregation. It has been concluded that the training data of SEK is the reason for its superior performance. Various shapes of the training matrix and read data determined the final results greatly, however the research hasn’t been very concerned with the actual properties of the data itself. How are the different behaviours of taxa feature spaces and how do they impact the final reconstruction. Is there an alternative way of representing the current training data, such as taking ordering of basepairs in consideration. These are questions that are believed to improve existing methods even further.

A discovery was made in the work, in k-mer feature space it would seem that taxa create natural clusters and that distance based classification could possibly work well. The run-time of this approach is tied to the number of taxon in reference, a proper initialization method to find out present taxa needs to be explored to implement a reliable and computationally efficient solution for simulated dataset testing.
References


