Novel machine learning methods for cancer sequencing analysis



Yun Feng

Jesus College

Nuffield Department of Medicine

University of Oxford

United Kingdom

A thesis presented for the degree of

Doctor of Philosophy in Clinical Medicine

July 19, 2021

Novel machine learning methods for cancer sequencing analysis

Yun Feng
Jesus College

A thesis presented for the degree of
Doctor of Philosophy in Clinical Medicine

July 19, 2021

Abstract

Heterogeneity is arguably one of the most important hallmarks of cancer which contributes to its drug resistance property. Cancer heterogeneity is the consequence of an evolutionary process during its development. Understanding cancer evolution will thus benefit the drug development field as well as clinical treatment of cancer.

In this thesis, I developed three novel approaches based on machine learning for the analysis of cancer evolution using genomics data.

In the work related to mutation signature, I developed a novel approach using convolution filtering to detect new signatures. I showed that this new approach works better than existing method under certain condition and could be applied to longer nucleotide sequences.

In the work for copy number evolution, I developed a reinforcement learning approach to uncover the evolution history. I showed that this approach could recover the history more accurately than existing methods. It is also the first reinforcement learning approach developed in the field for cancer evolution.

In the work for high dimensional genomic data analysis, I developed an approach which jointly carries out dimension reduction and clustering. I showed that this method could recover the underlying latent structure of genomic sequencing data more accurately than existing methods and this method could be used for scRNA-seq data analysis as well as many other fields.

These novel methods developed could be used for more cancer genomic data to gain better understanding of the cancer evolution process in the future.

Acknowledgements

First and foremost, I would like to offer my sincere gratitude to my supervisors Professor Christopher Yau and Professor David Church who offered me generous support and invaluable suggestions during my DPhil. I would also like to extend my thanks to my co-supervisor Professor Ian Tomlinson and Professor Christopher Holmes who provided helpful guidance on my work.

I also want to thank China Scholarship Council and Cancer Research UK Oxford Center for their generous funding support. I would like to thank Nuffield Department of Medicine and Jesus College at University of Oxford for supporting my research.

I am grateful for Professor Ahmed Ahmed for giving me invaluable advice and providing me with sequencing data during my transfer and confirmation of status. I would like to thank Professor Gerton Lunter who acted as my transfer assessor.

I would also like to thank Professor Ke Yuan who acted as my examiner for this work and Dr Edward Morrissey who acted as my confirmation assessor and internal examiner. They have provided helpful feedback on my work.

I would like to thank for the support that I received from lab members. They have provided me with valuable feedback on my work.

Declaration

I here declare that no part of this thesis has been accepted or submitted for any degree, diploma, certificate or any other qualification in this university or elsewhere. The results presented herein are my own.

Contents

List of Abbreviations **ix**

List of Figures **x**

List of Tables **xix**

1 Introduction to machine learning in cancer sequencing analysis **1**

1.1 Molecular mutational processes 2

1.2 Introduction to machine learning methods for studying cancer sequencing data 5

1.3 Related work 8

1.3.1 Mutational signature detection 8

1.3.2 Copy number variation detection 10

1.3.3 Joint clustering and dimension reduction for high dimensional sequencing data 13

1.4 Thesis structure 15

1.5 Math symbols 16

2 Convolutional filtering for mutation signature discovery **17**

2.1 Mutation signatures 17

2.2 Methodology background 19

2.2.1 Convolution filtering 19

2.2.2 Discrete convolution in human genome 21

2.2.3 Convolution neural network 22

2.2.4 Cosine similarity 24

2.2.5 Related work 25

2.2.5.1 Non-negative matrix factorization methods 25
2.2.5.2 Extending mutation signatures to longer nucleotide sequences: `pmsignature` 27
2.2.5.3 My contribution 30
2.3 Convolution based mutation signature detection: `convSig` 30
2.4 Model variants 32
2.4.1 Exponential transformation 32
2.4.2 ReLU transformation 34
2.5 Deep convolutional filtering 35
2.6 Experiments 37
2.6.1 Simulation study 37
2.6.2 More complex simulations 41
2.6.3 XERXES clinical trial data 43
2.6.3.1 Deep convolutional filtering 46
2.7 Discussion 48
2.8 Appendix 51

3 A reinforcement learning approach for learning copy number evolution in cancer 55
3.1 Copy number evolution in cancer 55
3.2 Motivation 57
3.2.1 Related work 58
3.3 Reinforcement learning 60
3.3.1 Introduction 60
3.3.2 Formal explanation 61
3.3.3 Applications 64
3.4 Model 65
3.4.1 Definitions 65
3.4.2 Reinforcement learning approach 66
3.4.2.1 Q-learning 66
3.4.2.2 Action space 68
3.4.2.3 Training samples 69
3.4.2.4 Reward function 69

3.4.3 Implementation . . . 70
3.4.3.1 Partitioning the action space . . . 70
3.4.3.2 Real-world physical constraints . . . 71
3.4.3.3 Permutation invariance . . . 71
3.4.3.4 Switch and residual representation . . . 73
3.4.3.5 Pseudocode . . . 74
3.4.3.6 Parameter settings . . . 74
3.4.3.7 Identifiability of the trajectories . . . 76
3.5 Experiments . . . 76
3.5.1 Heuristic benchmarks . . . 76
3.5.2 Synthetic data experiments . . . 78
3.5.3 TCGA colorectal cancer . . . 80
3.5.4 Exploratory analysis on genome duplicated cell line data . . . 82
3.6 Discussion . . . 86
3.7 Appendix . . . 89

4 A curvature regularization based method for joint dimension reduction and clustering 95

4.1 Background . . . 95
4.1.1 Methodology . . . 97
4.1.1.1 Clustering . . . 98
4.1.1.2 Dimension reduction . . . 102
4.1.1.3 Joint dimension reduction and clustering . . . 106
4.1.2 My contribution . . . 108
4.2 Motivation of the method . . . 108
4.2.1 Simple case study: linear model . . . 109
4.2.2 Two step EM algorithm: towards a nonlinear model . . . 111
4.3 Curvature regularized minimization . . . 112
4.3.1 Dimension reduction with curvature regularization . . . 112
4.3.2 Multiple curves . . . 114
4.3.3 Softmin approximation . . . 114
4.3.4 Optimisation . . . 114
4.3.4.1 EM algorithm . . . 114

4.3.4.2 Finite difference approximation . . . 115
4.3.5 Model selection . . . 117
4.3.6 Pseudocode . . . 119
4.4 Results . . . 119
4.4.1 Simple data examples . . . 120
4.4.2 Spiral data . . . 120
4.4.3 3D spiral data . . . 122
4.4.4 Choosing K: number of curves . . . 123
4.4.5 COVID-19-ICU data . . . 124
4.4.6 Single cell sequencing data: fibroblast development . . . 128
4.4.7 Single cell sequencing data: blood cell development . . . 130
4.5 Discussion . . . 133
4.6 Appendix . . . 136
4.6.1 Proofs . . . 136
4.6.2 Local update for tangent vectors . . . 140

5 Discussions and conclusions **142**
5.1 Overview . . . 142
5.2 Further direction . . . 144
5.3 Conclusion . . . 145

References **146**

List of Abbreviations

BVP	Boundary Value Problem (for ordinary differential equations)
CIN	Chromosome Instability
CNV	Copy Number Variation
CNN	Convolution Neural Network
CNP	Copy Number Profile
CRC	Colon Rectal Cancer
EM	Expectation Maximisation
HMM	Hidden Markov Model
LOH	Loss Of Heterogeneity
MDP	Markov Decision Process
MPAL	Mixed Phenotype Acute Leukemia
NMF	Non-negative Matrix Factorization
PCA	Principal Component Analysis
scRNA	Single Cell RNA
SNV	Single Nucleotide Variation
TCGA	The Cancer Genome Atlas
WGD	Whole Genome Duplication

List of Figures

1.1 **Machine learning applications on PubMed searching by keywords** (A)Number of publications related to machine learning in biomedical research each year. (B) Number of publications related to machine learning in cancer research each year. 6

1.2 **Methods for detecting CNA** (A)Paired end mapping: the distance between the pair of reads with CNA is different from others. (B) Spilt reads: a read is mapped to different part of the genome (C) The number of reads covering a region with CNA is different from other part (D)Assembly based: the assembly of reads could directly show the CNA. 11

2.1 **Overview of convolutional filtering model for mutation signature identification.** (A) Schematic of standard matrix factorization approach for mutational signature discovery. (B) Mutation processes can be modelled as convolutional filters that screen the genome. (C) These convolution filters each has an affinity for certain sequence contexts based on their filtering coefficients. High affinity sub-sequences will lead to an increased probability that a mutation will occur. (D) A cancer genome can be considered as the output after a number of mutational filters have passed across the genome. The learning task is to take the set of observed mutations and their local sequence context and to learn the most plausible set of mutational filters that could have given rise to these observations. 20

2.2 `pmsignature` (A):Naive Bayesian assumption: the probability of a pentanucleotide sequence to be mutated by the signature is determined by the product of relative preference (area of the base). (B) `pmsignature` assumes a mixture model for different signatures. The total number of mutations are assigned to each mutation signature. . 29

2.3 **Discrete convolution in human genome.** Compute the convolution product θ of a filter F of size 3bp and nucleotide sequence denoted by T. For filters of size 3bp, only the immediate flanking bases on each side has been taken into convolution. 31

2.4 **3-base results on simulations.** The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 3-bp nucleotide sequences. 39

2.5 **Estimating the number of signatures**. (A) Test and training error for `convSig` with the exponential transformation. (B) Test and training error for `convSig` with the ReLU transformation. 40

2.6 **5-base results on simulations.** The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 5 bp nucleotide sequences. 40

2.7 **3-base results on complex simulations.** The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 3-bp nucleotide sequences. `EMu` is not included as it only outputs one signature which means it fails to capture the difference between signatures possibly due to small sample size. 41

2.8 **Estimating the number of signatures**. (A)Number of signatures detected by `convSig` with the exponential transformation. (B)Number of signatures detected by `convSig` with the ReLU transformation. . . 42

2.9 **5-base results on complex simulations.** The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 5 bp nucleotide sequences. . . . 43

2.10 **Statistics of mutations in XERXES data.** Number of deletions, duplications, insertions, inversions, simple somatic mutations and translocations in each sample. 44

2.11 **Deep convolutional filtering.** (A) Distribution of output probabilities from the deep convolutional model for each sequence nucleotide sequence to be in a highly mutated region. The highly mutated (red) nucleotide sequences and low mutation regions (blue) are well separated in the trained model. (B) Average gradient of the signatures generating mutations in highly mutated regions. (C) Average gradient of the signature generating mutations in low mutation regions. (D) Average gradient of the signatures having preference for either low or high mutation regions. (E) Difference in output from first convolution layer between nucleotide sequences in highly mutated regions and in low-mutation regions. (F) Weights for all filters in the first convolution layer. 47

3.1 **Cell division.** (a) In normal cell division, a cell synthesises an exact copy of its DNA and shares it amongst two daughter cells. (b) Faulty cell divisions, DNA replication or chromosomal segregation can lead to abnormal cellular chromosomal content (aneuploidy). Paternally and maternally-derived chromosomes are shaded blue and red respectively. 57

3.2 **Copy number evolution.** Illustrating how two different sequences of copy number alterations events can lead to the same copy number profile. The actual evolution history is not observed, only the read out of the number of copies of each genomic region. 59

3.3 **Markov Decision Process.** At the current state s, an action a is taken according to some strategy $\pi(a|s)$. The reward $r(s, a)$ is thus determined by the current state s and actions taken a. After the actions taken, the state changed to a new state s' and the decision process continues as the previous state. 62

3.4 **Modelling copy number evolution.** A Cartoon illustration of a sequence of genomic alteration events associated with the transformation of a normal genome into a cancer genome. Copy number calling methods estimate copy number profiles from array- or sequencing-based data. My contribution is to develop a model `RLEvolution` that uses the copy number profiles to infer the evolutionary sequence. B Modelling the evolution of the genomic state sequence as a discrete Markov decision process requires and find the optimal policy $\pi(a|s)$. . 66

3.5 **Action space.** The action space is partitioned into three hierarchical sub-components, the first sub-action determines if a single chromosome should be chosen for alteration, a whole-genome doubling should occur or the evolutionary trajectory should end. If a single chromosome is chosen, the method must then decide what type of alteration to induce and where the copy number alteration begins. Finally, the end point of the alteration must be chosen. I use three convolutional neural networks to model the probability distribution over actions. . . 68

3.6 **A deep convolutional neural network model for computation of $\pi_\phi(a|s)$.** Because of the large search space for actions, the parametric model $\pi_\phi(a|s)$ is separated into three parts: $\pi_{\phi_1}(c|s)$, $\pi_{\phi_2}(sp|s, c)$ and $\pi_{\phi_3}(ep|s, sp, c)$. The $\pi_{\phi_1}(c|s)$ is built with the permutation invariant structure and with the switch to distinguish the cases with WGD and the case without WGD. Both $\pi_{\phi_2}(sp|s, c)$ and $\pi_{\phi_3}(ep|s, sp, c)$ will be based on the results of the other neural nets. Finally, the output of these three neural nets combined with the rule system, produces the conditional probability that's needed $\pi_\phi(a|s)$. 70

3.7 **Model Architecture.** This figure shows the main architecture of the model $\pi_\phi(a|s) \propto \exp(Q_\phi(s,a))$. First, on the left hand side, it shows the basic structure of the switch structure in partitioning of the action space. The symbols used are the classically used convolution(Conv), max-pooling and fully-connected linear(FC) layers. There are activation layers(ReLU) in between. Res indicates the residual representation. On the top right corner, it shows the combination of two small units with the switch to form a sub-unit of the action space. Finaly, on the bottom right corner, it shows the final combination of all sub-units together with the rule system (real world physical constraint) to form the full model. 75

3.8 **Example showing naive over-estimation of evolutionary trajectory length.** Four copy number alteration events are required to transform a normal genome to the final altered genome. However, if each sub-region is assumed to be affected by independent CNA events to achieve the final altered genome then six steps are required. This example also illustrates why `RLEvolution` would be better than benchmark method `WCND`. As `RLEvolution` only needs 3 steps to recover the copy number for this chromosome (except WGD which is shared by all chromosomes), however, `WCND` will need 4 steps as follows: a whole chromosome gain, a loss on the second segment (blue segment), a gain on the last 3 segments and finally a loss on the second last segment (orange segment). 77

3.9 **Synthetic experiments.** (A) A simulated copy number profile consisting of multiple CNA events and WGD. (B) Inferred actions trajectory for the sample in (A). (C) Inferred trajectory lengths for simulated CNPs with and without WGD. (D) Relative timing of inferred WGD events compared to ground truth. 79

3.10 **Compare Trajectory Similarity.** (A) The heatmap shows the correlation/similarity between the CNP of the ground truth (simulated) and CNP inferred by `RLEvolution` after each step. The correlation is computed by the number of loci with the same copy number divided by the total number of loci. The black box are the CNAs(actions) that are identical between the ground truth and the inferred CNAs by `RLEvolution`. The green line is the WGD event. (B) The boxplot of similarity of the trajectory by different methods. The trajectory similarity is computed by the number of identical actions (black boxes in graph A) between ground truth and inferred CNAs, divided by the total number of inferred CNAs (number of rows in graph A). 80

3.11 **TCGA CRC data analysis.** (A) Violin plot of average genome-wide copy number against inferred WGD status. (B) Number of actions required to generate each tumour copy number profile. (C) Timing of WGD relative to other CNA events. (D) Reconstruction accuracy. 81

3.12 **Cell Line Data** The cell line was split into two samples, and one of them has been treated with taxol. The DNA is extracted from these two samples and sequenced. 83

3.13 **Copy number profile of cell line data** (A) Copy number profile after filtering and cleaning of kuromachi cell line being treated with taxol. (B) Copy number profile after filtering and cleaning of kuromachi cell line without treatment. (C) The difference between two copy number profiles shown in (A)(B) 83

3.14 **Deconvoluted trajectory for copy number evolution by `RLEvolution`.** Deconvoluted step for the sample treated (A) with taxol and (B) without treatment. The y-axis is the timing of the step in the deconvoluted trajectory and x-axis is the genome position. The color legend is the same as the one for Fig.3.9B. The deconvoluted trajectory is the reverse of the evolution history, so that the history of evolution goes from top to bottom, and a gain in the deconvoluted trajectory is actually a loss in the evolution process. All red regions showed the loss in copy number evolution. 84

3.15 **Similarity between deconvoluted trajectories by `RLEvolution`** I compute the similarity between the CNP of two samples after removing the effect of each step in the trajectory. The color of each box indicates the similarity. The black box are the CNAs(actions) that are identical between the ground truth and the inferred CNAs by `RLEvolution`. The green line is the WGD event. The color of the heatmap indicate the similarity between two genomes at this step. . . 85

3.16 **Similarity between deconvoluted trajectories by `WCND`.** It is the same as Fig.3.15 except that the trajectory is computed by `WCND`. . . 86

4.1 **Dimension reduction and clustering.** (A) Dimension reduction: the data is shown in the top panel. It contains 2 dimensions. Each dot indicates a sample/single cell. The dimension reduction is shown in the bottom panel that the 2 dimensional data could be represented by a 1 dimensional curve. (B) Clustering: the data is formed by two disjoint cluster of points. The clustering process is to find the sample/single cells belonging to each cluster as shown in the bottom panel. (C)Joint dimension reduction and clustering: the data consist of two disjoint clusters and each cluster could be represented by a 1 dimensional curve. The task is to find the sample/single cells belonging to each curve and find the representation of each curve. . . 98

4.2 **Simple example results.** This figure shows the fitted curves (colored lines) on simulated data (black dots). Half circles are simulated to have radius of 1 and the noise is set to follow a normal distribution with mean 0 and variance 0.05. The algorithms manage to infer the correct underlying curves in all cases with 0,1,2 and 3 intersection points. 121
4.3 **Spiral Data** 122
4.4 **Spiral Data with Baseline method** The color of the dots indicate the reported pseudotime A. Results from method `iWMM`. B. Results from method `DRjCC`. 123
4.5 **Three Dimensional Spiral Data** 123
4.6 **Choose the number of clusters.** This figure illustrates the effect of shrinkage prior on the selection of the number of clusters to use. From top to bottom, I simulated two half circles with radius 1 and distances in y-axis 0,0.2,0.4,0.6 respectively. From left to right, the algorithm starts with $K = 4$ clusters, and reduces the number of clusters under the force of penalty. For the first and second row, the K gets finally reduced to 1. For the third and fourth row, the number of clusters gets reduced to $K = 2$, and it stops reducing the number of clusters. 125
4.7 **COVID-19-ICU Data** A. Component 1 and 2 are from the results of local linear embedding. Two colored curves are fitted by my method. B.Two curves correspond very well to the patients being/not being in ICU at the time of the measurement is taken. C. The red dots are measurements of patients who are not in ICU at the time taken but being sent to ICU later on. D. The color correspond to the length of the time after the patient has been admitted to the hospital. 127
4.8 **Fibroblast development scRNA-seq data** A. Two components of x,y axis are computed by local linear embedding. Two trajectories are detected by my method. Black dots are single cells. B.The colored dots indicate the ground truth of temporal order and it confirms these two development trajectories. 129

4.9 **Pseudotime results on fibroblast scRNA-seq data** The pseudotime has been binned to three groups: early, mid, and late, with equal cells in each group for visualisation. A. The color correspond to the results of `Curd`. A. The color correspond to the results of `DPT`. C. The color correspond to the results of `DRjCC` D. The color correspond to the results of `iWMM` 130

4.10 **Results on scRNA-seq leukemia data set.** I applied my method to single cell scRNA-seq data for both healthy people and leukemia patients. (A) Black dots indicate single blood cells of healthy people, and red dots indicate those of leukemia patients. The two blue lines are the two curves output by `Curd`. Two axis are computed by UMAP method. (B) The Venn graph for all up-regulated genes reported by author's method and `Curd` using the two curves. 131

4.11 **Comparison between local and global update.** (A) The initial loss for both methods is the same at 142. The loss decreases slower by using local update. (B) The blue curve is the output of `Curd` and the black dots are simulated data points. The blue curves are learned by `Curd` using global update and from left to right, the training iterations are 1, 8, 200 respectively. (C) The blue curves are learned by `Curd` using local update and the training iterations are 1, 40, 100, 950 from left to right 141

List of Tables

2.1 Distribution of the composition around mutated base. Position indicates the position of the base relative to the mutated base. Each cell indicates the proportion of the base at such position. 28

2.2 Cosine similarity of detected versus four Alexandrov mutational signatures for 3-base nucleotide sequences in the XERXES clinical sequence data. The 4 signatures correspond to No.1, 6, 10, 18 of the 30 signatures reported[1]. 45

2.3 Similarity between methods on 5-base nucleotide sequences for XERXES clinical data 45

2.4 Similarity between truncated 5-base signature to 3-base signature. . . 46

3.1 Parameters for Q function 75

4.1 Dimension reduction results for COVID-19 data 128

4.2 Clustering results for COVID-19 data 128

4.3 Bootstrapping results on fibroblast development data 131

4.4 Up-regulated genes 134

Chapter 1

Introduction to machine learning in cancer sequencing analysis

In this thesis, I have developed novel machine learning tools for analysing cancer sequencing data, including somatic mutation, copy number alteration and gene expression data, from DNA and RNA sequencing data. The motivation of the study is the assumption that heterogeneity among cancer cells is a result of an evolutionary processes[2, 3] and machine learning methods can be developed to study the history of such evolutionary processes from cancer sequencing data. In this thesis, mutational processes are used to refer to all molecular processes that change the structure of the genome, such as simple somatic mutations and copy number alterations. I developed machine learning methods based on the changed genome structure to understand the evolution history caused by these mutational processes. Hopefully, with the help of these novel tools and vast amount of cancer data, it is possible to gain a better understanding of the processes how cancer occur and to develop new therapeutic strategies.

Since epigenetic modifications are not the focus of the study here, they are not taken into consideration during the development of the methods. This is mainly because the study of epigenetic modifications requires special sequencing techniques[4] and their inclusion will inevitably increase the complexity of the models.

In this first chapter, I give a brief introduction to the molecular mutational processes

that operate in cancer, and to the machine learning methods previously used to study these processes. More details of the background can be found at the beginning of each chapter, while here I aim to give an overview of the topics covered in this thesis and their connections.

1.1 Molecular mutational processes

Most cancers are caused by somatic mutations[5]. These processes that generate mutations are called mutational processes, and these are usually the result of intrinsic infidelity of the machinery involved in DNA replication [6], exogenous exposure to mutagens[7–9], defective DNA repair pathways[10] and DNA modifications[11]. In some types of cancers, it has been shown that a substantial proportion of mutations are caused by exogenous mutagen exposures[1]. For example, in cutaneous melanoma, solar UV radiation tends to cause breaks in the structure of double strand DNA by producing dimmers in the adjacent pyrimidine and thus resulting in $CC \rightarrow TT$ mutations[12]. However, current understanding of the causes for many other mutational process is still limited[13].

These mutations have potential to make the genome unstable, by causing dysfunction of important genes involved in DNA replication or repair pathways, such as DNA polymerase, APC, TP53 and others[14]. These early mutations are the so-called driver mutations. Due to the instability of the genome, additional mutations occur, these often differ among patients and also among cells in the same patient. These so-called passenger mutations are not thought to contribute to the key attributes of cancer[5]. Typically, the number of passenger mutations is 1000 times more of the number for driver mutation in a single patient[14]. However, these two types of mutations together give rise to the cancer genome heterogeneity[2, 3]. Cancer heterogeneity here refers to the observation that different cancer cells within the same patient show distinct morphological and phenotypic characteristics, including gene expression, cellular morphology, replication, metabolic processes, motility, and metastatic potential[15].

Cancer evolution is a process that the cancer cells undergo, under the pressure of genetic, epigenetic and micro-environmental selection. Cancer evolution results in

cancer heterogeneity, an important factor contributing to therapeutic resistance to anticancer agents[5]. The true biological process of cancer evolution can be far more complex than the one described above. However, as the focus of this thesis is on the development of computational methods, readers are referred to other reviews for more details of cancer evolutionary processes[3, 16].

It is thus of strong interest to cancer researchers how the mutational processes leave their mark on the genome and what the essential mutations involved in cancer development are. The knowledge of these problems will also contribute to drug development and cancer treatment through blocking the effect of these essential mutations.

As a computational biologist, I try to answer these questions from the analysis of cancer sequencing data of clinical patients through machine learning approaches. However, because the real biological system is so complex, I only focused on three particular sub-field in the study of cancer evolution, including (1) mutation signature detection for simple somatic mutations, (2) deconvolution of copy number evolution history and (3) joint dimension reduction and clustering for single cell sequencing data. They will be discussed in details in each of the following result chapters.

The simplest form of mutations are called single nucleotide variation, also known as point mutation. In single nucleotide variations, only single base substitution, e.g. $C \to G$ is considered. For this single nucleotide mutation, all the flanking bases of the site mutated are not affected by the mutation and thus stay the same as before, unlike some other mutation, e.g. insertion or deletion where a fragment of DNA is inserted or deleted. In previous studies about mutation signatures, most attention has been paid to the the trinucleotide context, including the site being mutated and the adjacent 3' and 5' flanking bases. The two flanking bases could be any of the 4 bases (A,C,G,T) and the site being mutated could be any of the following 6 possibilities: $T \to A, C, G$ or $C \to A, G, T$. The Mutations such as $A \to C$ will be equivalent to $T \to G$ because of the reverse complementary property of human genome and thus they are not considered here. In total, there are $4 \times 4 \times 6 = 96$ number of different types of mutations. These different types of mutations do not occur with equal probability and there are some special patterns of their occur-

rence probability. These special patterns are defined as mutation signatures[13]. They arise mainly because that the sequential pattern of the local composition of genome will form a special chemical structure, and thus make it vulnerable to attacks by some certain chemical or physical mutagens. The study of these patterns could potentially reveal unknown mutagens or shed light on the chemistry behind some known mutagens[1]. Further details on computational methods for mutational signatures will be provided in the next section as well as chapter 2.

A different type of mutations that will be studied in this thesis are copy number variations (CNV). In cancer, it is called somatic CNV or in other words, copy number alterations (CNA). In normal human cells, each gene should contain exactly two copies, with one from father and one from mother. However, in cancers, because of the activity of mutational processes and DNA repair processes, the copy number of genes may change[17]. One example is a special type of mutation called whole genome duplication (WGD), where the copy number of each gene gets doubled[18]. Such whole genome duplication widely occurs in different types of cancer and is associated with poor prognosis[19]. Besides, there are also other types of copy number mutations which only affect a small proportion of the human genome, including whole chromosome loss and gain, and partial chromosome changes[20]. Partial chromosome changes are also called focal CNAs. Except for these somatic copy number variations, there are also germline copy number variation which are regions with different copy number among healthy population[21]. These germline copy number variations are carried over from parental generations and they are a valuable source for uncovering the history of human migration, adaptation and range expansions[21]. These germline variants are not the focus of my study and are generally excluded by comparing cancer tissues against a control sample from the same patient, e.g. blood sample. Further details on copy number detection will be provided in the next section and copy number evolution as well as whole genome duplication will be discussed in detail in chapter 3.

In chapter 4, the dimension reduction and clustering methods for cancer single cell sequencing are studied. Dimension reduction methods are often used to find low dimensional latent structure in a high dimensional data set[22]. For example, the

temporal structure among cancer cells is a latent low dimension (1 dimension) structure for high dimensional gene expression data[23]. Dimension reduction methods for recovery of the temporal structure are particular useful for fields where the collection of time series data is challenging or prohibitive. For example, for many cancer patients, only one biopsy sample is collected in many studies. Clustering methods are often used to find the subtype latent structure. Such clustering method are widely used in cancer sequencing data analysis to discover molecular subtypes of the cancer[24]. Further details about dimension reduction and clustering for cancer genomics analysis will be discussed in the following section as well as chapter 4.

1.2 Introduction to machine learning methods for studying cancer sequencing data

Machine learning is the study of computer algorithms that improve itself automatically through experience. Machine learning algorithms learn from collected real world data and help understand complex patterns. They are applied to various types problems in biomedical research, from transcription factor binding site prediction to pharmaceutical applications[25, 26]. For example, in transcription factor binding site detection, researchers make use of human labelled genomic sequence data, and apply machine learning algorithms, including support vector machine[27] and deep learning approaches[28], to find common patterns between transcription factor binding sites and use these to predict if a new genomic sequence is a transcription factor binding site or not[29, 30].

Different from traditional statistical methods used in these areas, e.g. t-test or linear regression, machine learning algorithms have certain advantages in analysing data with highly complex structure and large quantity[31] because machine learning models usually allow more complex patterns in the model assumption. Thus, the increase of popularity of machine learning approaches in biomedical research, including cancer studies is getting obvious(Fig.1.1).

Machine learning algorithms can be roughly classified into supervised learning and unsupervised learning algorithms[31]. For supervised learning, the goal is to predict

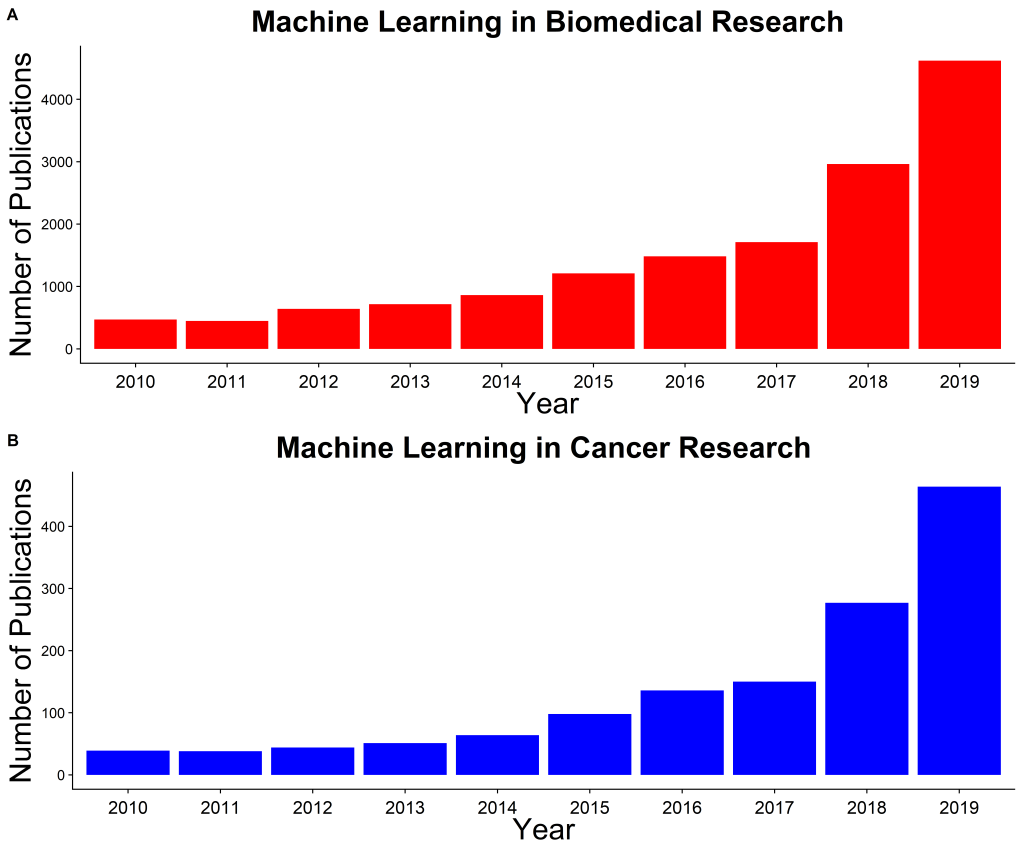


Figure 1.1. **Machine learning applications on PubMed searching by keywords** ***(A)Number of publications related to machine learning in biomedical research each year. (B) Number of publications related to machine learning in cancer research each year.***

the outcome for future data point. For example, to predict if a genomic sequence is a transcription binding site[32] and to predict if a patient have certain type of disease based on his/her MRI or CT scans[33, 34] are certainly supervised learning problems. One of the most important characteristics of supervised learning algorithms is that labeled data are essential in order to train the model.

However, unsupervised learning does not need labeled data for training. It is often used as a way to discover the latent structure within the data available. For example, clustering algorithms have been widely used in single cell RNA-seq data, which could potentially give indication of subtype structure within the tissue[35, 36]. Besides, unsupervised learning is extremely widely used in the study of cancer evolution. It is most suitable for the type of data where it is difficult to do human labelling. For example, in cancer evolution, the phylogeny structure is of great interest, whereas it is difficult to design a standard experimental method to reconstruct the structure

truthfully due to the complexity of cancer evolution processes[37–39]. Thus, supervised learning algorithms are not applicable and unsupervised learning algorithms have been developed for these problem[40–42].

Machine learning algorithms have fostered the growth of genomic data analysis as well as many other aspects in biomedical research[25, 26]. However, the study of biomedical data has, in return, fostered the development of new machine learning methods as well. For example, advanced causal inference methods are developed because of the clinical need to understand the causal relationship between drug treatment and patients recovery[43, 44].

I am particularly interested in applications of machine learning in cancer evolution study using sequencing data. There are several reasons for using machine learning to study cancer evolution through sequencing data. First of all, cancer sequencing data are highly complex and of very high dimension, which make it difficult to be analysed by traditional statistical methods, e.g. linear regression, due to a fact called curse of dimensionality in statistics[45]. Besides, most traditional statistical methods make simple assumptions. For example, in linear regression, the relationship between response variable and covariate variable is assumed to be linear. These classic statistic methods are not good at catching complex patterns. Secondly, with the increasing amount of data available, due to the decreasing of sequencing cost[46], machine learning methods will become more robust and results from machine learning methods will become more reliable. Thirdly, unlike many other areas in biomedical research, which has been studied for a long time by many researchers, e.g. biomedical image analysis[47] and genomic element prediction[48], cancer evolution is relatively a less well studied sub field. Many topics in this field is of great interest to oncologists but the current methods have certain limitations in them. There is still room for improvement in this sub field.

Here in this introduction chapter, I aim to provide some introduction to the topics in the field of cancer evolution which I have studied in details in the following chapters. These related methods will again be discussed in further details in the following chapters with formal math definitions.

1.3 Related work

1.3.1 Mutational signature detection

Mutation signature is one of the first fields that I am going to study in detail in chapter 2. As mentioned earlier, mutational signature is directly related to the mutational processes or chemicals in the living environment of a patient. The understanding of mutation signatures, and potentially the biological mechanism behind, will help clinicians to make decisions on the treatment of patients[49].

There are many widely used algorithms for deciphering the mutation signatures, including `WTSI-NMF`[50], `EMu`[51] and `pmsignature`[52]. They differ in the model and method that they use, but through careful mathematical derivation, it could be shown that the essential part of their model is quite similar[53].

$$\begin{aligned} \boldsymbol{X} &\approx \boldsymbol{\Theta P} \\ \boldsymbol{\Theta} &\geq \boldsymbol{0} \\ \boldsymbol{P} &\geq \boldsymbol{0} \end{aligned} \tag{1.1}$$

Most of these methods use non-negative matrix factorization(NMF) as the core part of the algorithm[50, 54]. As shown above, in NMF, there are two matrices to be learned. One is the mutation signature matrix $\boldsymbol{\Theta}$, which contains the features of different mutation signatures. This feature has a biological interpretation as it is assumed to be proportional to the probability of a mutation caused by a mutational process associated with such signature. Another matrix is the mutation activity matrix $\boldsymbol{P}$, which contains the mutation rate of each mutation signature in each samples. The mutation rate here is defined as number of mutations per sample. The machine learning methods used are un-supervised learning matrix factorization algorithms. It minimizes the distance metric between the multiplication product of these two matrices and the true mutation matrix that contain the number of different types of mutations in each sample $\boldsymbol{X}$. Namely, NMF tries to minimize the difference between the left hand side and the right hand side of the equation(1.1). The non-negative constraints refers to the biological fact that mutational processes

only create new mutations but not make mutations disappear, so that every value should be non-negative in the matrices.

In the original work of `WTSI-NMF`[50], the distance metric is the Frobenius norm, which corresponds to a probability model that the error in the number of mutations follows a normal distribution. This method has laid the foundation for many following methods and provides a benchmark in the field.

However, as the number of mutation is certainly an integer number, the Poisson distribution could be a better choice than normal distribution. (In normal distribution, the number is not necessarily an integer.) Thus, in the `EMu`, the distance metric is the Kullback-Leiber divergence, which corresponds to a probability model that error follows the Poisson distribution[51].

In another modified version, called `BayesNMF`, a Dirichlet Process distribution is used as a prior distribution to make it easier to select the number of signatures[55]. Dirichlet Process prior is a non-parametric distribution widely used in machine learning for feature selection[56]. Similarly, in `signeR`[57], a prior distribution is used. However, in `signeR`, the prior distribution is determined by the empirical Bayes method[58], and thus the prior distribution will vary depending on the data provided.

Though non-negative assumption is quite important in the model in order to account for the fact that mutational processes only introduce new mutations but do not remove existing mutations, in `SomaticSignatures`, it provides a way to use other matrix factorization methods from machine learning field including principal component analysis (PCA) to see if there are new mutation signatures[59]. The downside of this type of problem is that it is difficult to find biological interpretation of the negative values in the matrices.

There are also some methods which add modifications to the non-negative matrix factorization method to achieve some particular goal. For example, there is a method trying to recover sparse mutation signatures called `SNMF`[60]. For sparse signatures, an additional assumption has been made in these models that a mutation process has strong preference for a specific type of mutation, and thus will not cause different types of mutations, e.g. only causing $ACG \rightarrow AGG$ but not any other mutation. In

`SNMF`, an L1 penalty term has been added, which plays a similar role as in LASSO to encourage sparse signatures[60].

All the methods above more or less rely on the algorithms of matrix factorization and are all designed for the analysis of mutation signatures associated with trinucleotide context, including the site being mutated and two flanking bases. They could be potentially applied for mutation signatures with longer nucleotide sequence, however for this purpose, much more samples will be required because more base pairs would make the inference algorithm far less robust. Thus, it will be challenging to apply these NMF based methods to study mutation processes that have preference for certain mutations in a pentanucleotide context, e.g. $CACGT \rightarrow CATGT$. A method called `pmsignature` has looked at this issue, and used a probabilistic model with Naive Bayes assumption to solve this problem[52]. However, because of such probabilistic model used, it is particularly difficult to choose the number of signatures for `pmsignature`.

As mentioned, in the field of mutation signature detection, most existing methods have only looked at mutations with 3 bases. Thus, one potential future development will be in extending it to more bases. Besides, all the methods mentioned above formulate the mutation signature problem as an unsupervised machine learning problem. Thus, I am going to study the problem in a totally different view. I will formulate the mutation signature detection problem as a supervised machine learning problem, for which a classification model could be designed.

1.3.2 Copy number variation detection

Copy number variation is one of the major type of mutations in cancer[20]. It roughly accounts for 12% of the total mutations detected in cancer[61]. In normal genome, each region should contain exactly one copy from mother and one copy from father. If there is a CNA, the number of copies for certain regions will change. It arises for multiple reasons. For example, during DNA repair, a non-homologous end joining could result in a copy number loss, while homologous directed repair could result in a loss for one allele and a gain for the sister allele, which will result in an event called loss of heterogeneity (LOH) in patients without WGD[62].

There are multiple ways of detecting CNA in cancer genome. Generally speaking, these can be divided in to five categories based on the specific sequencing technique used.

- Paired end mapping: for a pair of reads, if the distance in between is significantly different from other pairs, it is considered to have either copy number gain/loss in between
- Split read: for a read, if it maps to different parts of the genome, then the region in between of these two parts are considered to be lost
- Read depth: for a genome region, if the number of reads covering this region is significantly different from others, this region is considered to have CNA
- Assembly-based: assemble reads from sequencing to contigs and map contigs to the genome to see if there is CNA
- Combinatorial approaches: The combination of methods above

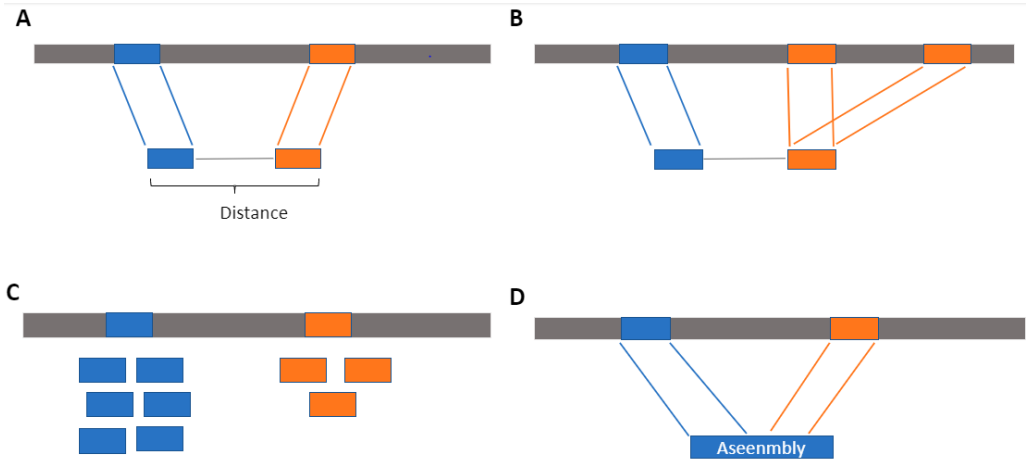


Figure 1.2.* Methods for detecting CNA *(A)Paired end mapping: the distance between the pair of reads with CNA is different from others. (B) Spilt reads: a read is mapped to different part of the genome (C) The number of reads covering a region with CNA is different from other part (D)Assembly based: the assembly of reads could directly show the CNA.

Most of studies choose to use one of the sequencing approaches mentioned followed by a carefully designed statistical or machine learning caller to find CNA in cancer genome[63, 64]. As there is no human labeled copy number data, all these methods belong to the unsupervised learning methods.

For example, in one of the read depth based methods, the Expectation-Maximization[65] method is used to assign reads to similar regions[66]. In traditional read depth based method, one of the difficulty is that there are regions on the genome that are similar and may only differ by one or two bases. For reads from these similar regions but not containing these different bases, it is hard to determine which region this read is coming from making CNA calling very inaccurate. In this method, an Expectation-Maximization algorithm is used and the probability of which region that this read is coming from is computed. The basis, on which probability could be computed, is from the reads that covering the region with different bases.

Another difficulty in CNA calling is how to define the segmentation. It is usually hard to find where the break-point of CNA is accurately. For example, in read depth methods, for two adjacent regions, if they have different number of reads covering them, it could be explained as it is a sampling error, i.e. random noise. However, it could also be explained as these two regions have different number of copies in the mutated genome. In other words, at least one of the region contain a CNA. In many methods based on machine learning, a Hidden Markov Model (HMM) is used[67, 68]. In HMM, an assumption is made that for regions next to each other, there is a large probability that have the same copy number and the difference in read depth is caused by random noise; but also that there is a small probability that this is caused by CNA in one of the regions. Based on this probability model, the posterior probability that these two regions contain the same copy number and different copy number are then compared. In the final result, the model outputs the result which has higher posterior probability[67]. Besides, in models using HMM, it is much easier to incorporate the subclone structure, and it is possible to compute the CNAs in major clones and minor clones, as well as the cell fraction of minor clones[67].

Multiple cancer samples from one patient pose a particular challenge for the analysis of CNAs. In order to analyse them jointly, a reasonable assumption is often introduced: different samples from the same patient should contain many CNAs in common, i.e. with the same break points. Methods based on HMM and mixture model have been developed to improve the statistical power of these joint analyses[69, 70].

As copy number alteration calling is mostly done by unsupervised machine learning algorithms, its performance relies a lot on the soundness of the probability model.

With all these methods being developed, most researchers have only focused on the process of mapping the sequencing data to the copy number profile and very few computational methods looked at the evolution process of such copy number profile. In chapter 3, I am going to study the process of copy number evolution in detail with a new kind of machine learning method called reinforcement learning[71], which will be introduced in details later on.

1.3.3 Joint clustering and dimension reduction for high dimensional sequencing data

With the development of high-throughput sequencing techniques, i.e. the so-called omics methods, it is much easier than before to get genomic/functional genomic data for a cancer tissue[72, 73]. For example, cancer transcriptomics data from RNA-seq method can be used to study the gene expression level in cancer cells[74]. These data are typically of very high-dimension, which poses a particular challenge for data analysis tools. In the previous example, a single RNA-seq study could give the expression level of thousands of genes and therefore in order to compare two samples, it is necessary to compare the expression level of these thousands of genes. In statistics, it is not a robust way to compare the genes one by one in order to study the relationship between cells.

This high dimensional data poses a particular challenge for classic statistical methods due to a fact called curse of dimensionality mentioned earlier[31]. It is been shown that with the dimension of the data increasing, the probability that the result of statistical method, including linear regression, will hold true decrease exponentially[45]. Thus, a class of machine learning method called dimension reduction is often used to tackle such problem.

Besides, in many cases, omics data of several different samples are available and the goal is to find out the relationship between samples. In other words, the objective is to find if there are subtype structures within the data such that some samples

belong to the same molecular subtype with certain properties, e.g. expression level of certain genes, shared in common while other samples outside the subtype do not have such property. The method in machine learning for this particular problem is called clustering method[31].

As mentioned above, clustering and dimensional reduction methods from machine learning are designed to study such latent structure within the data. There are many well developed methods from machine learning field and have been successfully applied them to cancer sequencing data[75].

Full details of these methods and their statistical models will be discussed in the chapter 4. Here, I will briefly discuss about how they have been applied in cancer research to disentangle cancer heterogeneity due to the evolution process as an example.

For clustering, it assumes that the samples could be classified into subtypes, so that samples in the same cluster will share similar patterns while samples in different clusters will differ significantly. This method has been used to study the subtype of cancer cells or cancer patients in previous studies[24]. Besides, such subtype structure may be correlated with the subclone structure during the evolution of cancer, so clustering methods have also been applied to study the phylogeny structure[40–42].

For dimension reduction, it is assumed that there is an underlying lower (typically 1 to 3) dimensional structure among all data points. For example, this low dimension structure could be either time (1 dimensional) during the development or the position of the cell in the body (3 dimensional). These dimension reduction methods are thus developed to uncover such low dimensional structure[22]. The most widely used method is of course principal component analysis (PCA)[76, 77] which is a linear method, but nonlinear methods have also gained popularity later on for biomedical research[22]. For cancer evolution, a special application of dimension reduction algorithm is the pseudotime study. In pseudotime studies, it is assumed that the evolution of cancer follows a temporal order and by reducing the feature of cancer samples to one dimension, the temporal order of these cancer samples could be recovered[23].

The assumptions made by clustering and dimension reduction methods seem quite natural and fit well with the cancer evolution topics in many cases[78]. However, the real world biological samples are often highly complex and it is very likely that both the subtype structures and the low dimensional temporal structure will be present in the same sample. Thus, I am going to look into this aspect and try to combine clustering and dimension reduction tool together in chapter 4.

1.4 Thesis structure

This thesis is going to have the following structure.

- Chapter 1 Introduction of the foundation about the areas I have developed methods for
- Chapter 2 Novel methods for mutational signatures
- Chapter 3 Deep reinforcement learning approach for copy number evolution in cancer
- Chapter 4 A curvature regularized multi curve dimension reduction method with applications in cancer genomics
- Chapter 5 Discussion and conclusions

1.5 Math symbols

$f(t), g(t), h(t)$	continuous functions of independent variable t
$p(x)$	probability mass function of x
f, g	discrete functions
F^i	discrete filter function with four channels for convolution
T_j	discrete nucleotide sequence function with four channels to be convoluted
$\mathbb{Z}, \mathbb{N}$	all integers and all natural numbers
$\mathbb{R}$	all real values
i, j, k, n	subscript/superscript to indicate one element of the set
N, K, J	integers, usually used to indicate the size of training samples/number of models and etc
$\theta, \psi, \phi, \alpha, \beta, \eta$	parameters of the model
λ, ρ	hyperparameter, e.g. the strength of regularization
x, y, z	data points and variables of the model
s	state (for reinforcement learning)
a	action (for reinforcement learning)
τ	trajectory
$\pi(a\|s)$	policy (for reinforcement learning)
$\vec{x}$	vector
$\boldsymbol{X}$	matrix
$\mathcal{X}$	space,set
∇	gradients
$*$	convolution operation
$< \vec{a}, \vec{b} >$	inner product of $\vec{a}, \vec{b}$
$\sum, \prod$	summation, product

Chapter 2

Convolutional filtering for mutation signature discovery

2.1 Mutation signatures

All cancers contain mutations in their genome, and mutations accumulate during the evolution and life history of cancer[5]. As mentioned in the chapter 1, some mutations, so-called driver mutations, give cancer cells selective advantage during expansion and metastasis, while most other mutations, so-called passenger mutations do not have any effect on the fitness of cancer. For example, many mutations in the tumour suppressor gene TP53 are driver mutations in multiple cancer types[79–81]. Typically, the number of passenger mutations in the same sample is 1000 times more then that of the driver mutations as mentioned previously[14].

There are different types of mutations, e.g. insertions, deletions and others, while here I only focus on the simplest form of the mutations: single base substitutions.

A single DNA strand could be seen as a long sequence of code consisting of 4 different bases $\{A, C, G, T\}$. Genomes for all living creatures except virus, consists of two DNA strands which forms a double helix structure. The strands are in opposite direction in the sense that their codes are read in the direction from 5' end to 3' end , and each base on these strands forms a Watson-Crick base pair (bp): $\{A : T, C : G\}$ (5' and 3' are defined according to the linkage of the nucleotide, which is a direction determined biologically). This combination forms the reverse-

complementary property for DNA. Thus, a single strand actually encodes most of the genome information except epigenetic modifications and this, in terms of mutations, a $CTT \rightarrow CGT$ base substitution, is equivalent to $AAG \rightarrow ACG$ substitution on the other strand.

During oncogenesis, there are sometimes mutational processes, the consequence of which are manifest as defined alterations in DNA sequence. These patterns of defined alterations in DNA sequence that sometimes occur in certain types of cancer due to exposure to the mutagen are called mutational signatures[13]. Such sequential patterns of mutations have long been of interest to cancer biologists[54]. For example, $C \rightarrow A$ mutations are frequently observed in lung cancer, and these were later determined to be caused by smoking through clinical data[7]. It is common to see $GG \rightarrow TT$ mutations in melanoma and other skin cancers and it has been later shown to be associated to ultraviolet radiation[8]. The study of these sequential patterns of mutations, the so-called mutation signatures, contribute to the understanding of the mutagenic processes that lead to cancer.

Following the first work based on non-negative matrix factorization, i.e. `WTSI-NMF`, to capture mutation signatures for cancer data[50, 54], the signatures in the context of trinucleotide sequence has been studied broadly and more systematically[82]. Many of the discovered mutation signatures have been verified by clinical data and assigned to particular mutagens[83, 84]. Therefore, in most literature, mutation signature is defined as patterns of mutations on trinucleotide sequences.

As mentioned in chapter 1, there are a number of computational methods for mutation signature detection using genomic sequencing data[53]. Most of these methods only focus on the trinucleotide context.

Also as mentioned previously, the standard approach for identifying mutation signatures has been popularized by the work[1], where a non-negative matrix factorization (NMF) is used to decompose a $96 \times J$ count matrix (of all the trinucleotide mutations identified from the sequence data of J samples) into two low-rank $96 \times K$ and $K \times J$ matrices. Here K is the number of mutation signatures[50, 55]. The first factor matrix contains the so-called mutation signatures, i.e. the pattern of mutations, whilst the second contains the mutational rate of each signature for each sample (Figure

2.1A). One limitation is that the trinucleotide context is somewhat chosen arbitrary. Indeed previous research showed that mutational processes might be related to more extended sequence contexts[85]. A previous method called `pmsignature` has looked at this problem while most others have ignored this issue. Furthermore, most methods do not implement model selection and estimate the number of signatures K[52]. This parameter is usually pre-defined.

Here, I propose a novel approach for mutation signature identification based on convolutional filtering. The premise is that each mutational process can be described as a convolutional filter which scans the genome. Each filter has coefficients which gives it more affinity to certain sequence types. When a filter encounters a sequence that it has high affinity for, there is an increase probability that a mutation will occur here. The inference task is to examine the set of mutations detected from genome sequencing of a cancer (the type and its sequence context) and to infer the set of mutational filters (the coefficients) that is likely to have given rise to that observed data (Figure 2.1B-D). This approach is therefore fundamentally very different to NMF of mutation count matrices and has a qualitatively more mechanistic link to the biochemical processes that might produce these mutations.

2.2 Methodology background

2.2.1 Convolution filtering

Convolution is a classic concept from signal processing, where for two functions $f(t)$ and $g(t)$, the convolution product would be $(f*g)(t) = \int f(t-t')g(t')dt'$. The result of convolution could represent the local similarity between these two functions at some point t, which will be clarified later on. In biology, many molecular processes associated with the genome, e.g. DNA replication and DNA repair, are based on the local structure of the DNA and the structure of the interacting proteins or chemicals. Therefore, it is natural to make assumptions that genomic sequences which have similar local structures will be mutated in a similar way. These considerations mean that the convolution product could be a promising method to use to study the local similarity between genomic sequences.

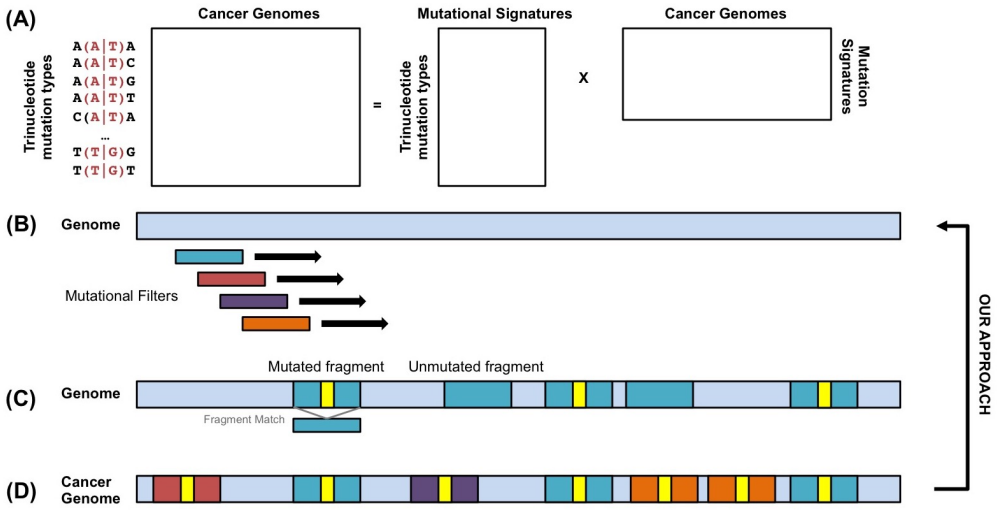


Figure 2.1.* Overview of convolutional filtering model for mutation signature identification.** ***(A) Schematic of standard matrix factorization approach for mutational signature discovery. (B) Mutation processes can be modelled as convolutional filters that screen the genome. (C) These convolution filters each has an affinity for certain sequence contexts based on their filtering coefficients. High affinity sub-sequences will lead to an increased probability that a mutation will occur. (D) A cancer genome can be considered as the output after a number of mutational filters have passed across the genome. The learning task is to take the set of observed mutations and their local sequence context and to learn the most plausible set of mutational filters that could have given rise to these observations.

For two functions, $f(t)$ and $g(t)$, the integral

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t - t')g(t')dt'$$

is defined as the *convolution* of $f(t)$ and $g(t)$ where $*$ denotes the convolution operator.

And in practice, a discrete fashion of convolution is often used.

$$(f * g)_k = \sum_i f_{k-i} g_i$$

f and g here are a sequence of real numbers, or discrete functions equivalently, i.e. f and g consist of f_i and g_i respectively, where $i \in \mathbb{Z}$.

Next, it can be shown that convolution operation is often used as a way to measure the local similarity between $\tilde{f}(t) = f(-t)$ and $g(t)$. As seen from the previous

definition, following equations could be derived.

$$\begin{aligned}(f * g)(t) &= \int_{-\infty}^{\infty} f(t - t')g(t')dt' \\ &= \int_{-\infty}^{\infty} \tilde{f}(t' - t)g(t')dt' \\ &=< \tilde{f}(t' - t), g(t') > \\ &=< \tilde{f}(t'), g(t' + t) >\end{aligned}$$

As shown above, where $\tilde{f}(t) = f(-t)$, the convolution could also seen as the inner product of two functions, which will measure the similarity between them. In math, the similarity between two functions is often defined as an inner product. Therefore, for each t the convolution product $(f * g)(t)$ is measuring the similarity between the shifted filter function $\tilde{f}(t' - t)$ and the other function $g(t')$ to be convoluted.

In practice, the filter function will have finite support, i.e. there exists some $I \in \mathbb{R}$, $\forall t'$, such that $|t'| > I, f(t') = 0$. Therefore, the convolution product is comparing the similarity of $f(t')$ and $g(t' + t)$ centered around t, where $-I + t < t' + t < I + t$. This gives the similarity between $f(t')$ and the local structure of $g(t'') = g(t' + t)$ around t. In other words, if the filter function $f(t)$ has finite support, the convolution product is representing the local similarity between $g(t)$ and $f(t)$.

2.2.2 Discrete convolution in human genome

In human genomics, the local structure of DNA sequence determines its function, and therefore, convolution operation will be a suitable mathematical method to measure this local DNA structure.

As stated above, convolution is carried out in a discrete format. Consider the human genome as a function detailed below. This function maps each locus of the genome to a particular base. Thus, the domain of this special function is discrete, i.e. $g : \mathcal{N} \to \{A, C, G, T\}$. Here, $\mathcal{N}$ indicates all loci on the genome. This is a technique commonly used in computer vision that the output of four bases is treated as signals in four different channel. Thus, $g_i = (g_i^A, g_i^C, g_i^G, g_i^T)$ could be seen as a vector with 4 outputs, i.e. $g : \mathcal{N} \to \{0, 1\}^4$. The output in each channel will indicate if such base

exists at the corresponding position or not. For example, for the 1st position on the genome, i.e. the 5' end of the chromosome 1, if it is a base A, then $g_1 =< 1, 0, 0, 0 >$, as the first channel indicates base A, which exists for the 1st locus, while all other bases do not. And if it is a base C, G, T, then $g_1 =< 0, 1, 0, 0 >$, $g_1 =< 0, 0, 1, 0 >$ or $g_1 =< 0, 0, 0, 1 >$ respectively.

This method of viewing the human genome as a discrete function is called one-hot encoding and it is has been used previously to investigate the application of convolution operation for human genome[86].

It is then necessary to modify the discrete filters $f : \mathcal{N} \to \mathcal{R}^4$ to the same 4 channels in order to carry out convolution operations.

The convolution product between them is the sum of the convolution in four channels of A, C, G, T. More specifically, if a position i in genome with base $b \in \{A, C, G, T\}$ is denoted by g_i^b, which is simply one of the dimensions of g_i, and if the same definition is applied to f_i^b, the convolution output is shown below.

$$
\begin{aligned}
(f * g)_i &= \sum_{b \in \{A,C,G,T\}} (f^b * g^b)_i \\
&= \sum_{b \in \{A,C,G,T\}} \sum_k f_{i-k}^b g_k^b \\
g_i^b &= \begin{cases} 1, & \text{if the } i\text{th locus on genome is base b} \\ 0, & \text{otherwise} \end{cases} \\
f_i^X &\in \mathbb{R}
\end{aligned}
$$

As f^b and g^b are one dimensional discrete functions, the convolution is carried out as described in the previous sections. The values of g^b are determined by genome sequences while f^b are filter weights to be inferred by machine learning algorithms.

2.2.3 Convolution neural network

Convolution neural networks(CNN) have been widely used in computer vision as well as genomic element classification, e.g. predicting Long non-coding RNA[87]. It is based on the conjugate of a few parametric functions, which finally forms a

hierarchical structure called neural networks. In neural networks, the input x is transformed by multiple functions sequentially to the the output.

$$
\begin{aligned}
y_1 &= f_1(x) \\
y_2 &= f_2(y_1) \\
&\dots \\
y_H &= f_H(y_{H-1})
\end{aligned}
$$

In the above model, y_i indicates the output of each layer/function, and y_H is the last output of the model, which is the loss that is going to be minimised. For some layers, $f_i(y_{i-1})$, they may contain parameters to be learned/optimised. To achieve this goal, one could use a gradient based optimization algorithm and the only ingredient needed is the gradient with respect to the parameter. To get such gradient, the chain rule for derivative is used.

$$
\begin{aligned}
\nabla y_H &= 1 \\
\nabla y_{H-1} &= \nabla_{y_{H-1}} f_k(y_{H-1}) \nabla y_H \\
&\dots \\
\nabla y_1 &= \nabla_{y_1} f_2(y_1) \nabla y_2 \\
\nabla x &= \nabla_x f_1(x) \nabla y_1 \\
\nabla \theta_i &= \nabla_{\theta_i} f_i(y_{i-1}) \nabla y_i
\end{aligned}
$$

Here, θ_i is the parameter associated with the ith function $f_i(y_{i-1})$. $\nabla_{y_{i-1}} f_i(y_{i-1})$ and $\nabla_{\theta_i} f_i(y_{i-1})$ indicate the partial derivative with respect to the input y_{i-1} and the parameter of this layer θ_i respectively. And finally, ∇x is the gradient with respect to the input data.

CNN are neural networks where the functions used are within some special cases, including convolution, ReLU, MaxPooling and linear, which are described below[28].

A convolution layer is defined as above, $conv(x) = \theta * x$ where θ are parameters to

be learned.

$$conv(x, \theta)_t = (\theta * x)_t = \sum_{i=0}^{T} \theta_i x_{t-i}$$

Here, θ are parameters to be learned for the model. Another way to look at this is that θ is a discrete valued function with θ_i, where $\theta_i = 0$, $\forall i < 0$ or $i > T$.

ReLU functions are nonlinear activation functions, which truncate the negative input to 0 while keeping the positive part unchanged.

$$ReLU(x) = max(x, 0)$$

The reason that such ReLU transformation is widely used in deep learning is that it not only introduces nonlinearity into the model, but it is also very easy to compute the forward and backward pass. The forward pass of ReLU transformation, i.e. the evaluation of the value of the function, simply truncates all negative values to zero and the backward pass also blocks the gradient from going into these nodes while not changing the gradient for those positive nodes[28].

MaxPooling reduces the dimension of the input and only retains the value of the largest one, as follows.

$$MaxPooling(x_1, x_2, x_3, x_4) = max(x_1, x_2, x_3, x_4)$$

In terms of forward and backward passing, it is quite similar to ReLU that it is easy to compute. What's more, Max-Pooling layers bring sequential information which are not originally in the same position together[28].

2.2.4 Cosine similarity

The traditional way of defining a mutation signature is to define a vector containing values proportional to the probability for each mutation to cause different types of mutations[50]. Therefore, all values in the vector are non-negative and they often sum up to one.

To compare mutation signatures detected by different methods, a metric is needed

to measure the similarity between the signature vectors. Though the L2 distance (Euclidean distance) is the most commonly used metric, it is not commonly used for signature vectors[50, 52] for the reason mentioned below.

Ideally, when comparing two mutation signatures, the norm of the vector should not contribute to the similarity. For example, for a constant λ, $\vec{x}$ and $\vec{x}' = \lambda\vec{x}$ actually represent the same signature vector by definition, so ideally the metric will produce the same value for $\vec{x}$ and $\vec{x}'$ when comparing to another signature vector $\vec{y}$, i.e $dist(\vec{x}, \vec{y}) = dist(\vec{x}', \vec{y})$.

It could be checked that L2 distance does not have such a property and it is better to use an alternative metric called cosine similarity for bench-marking[50].

$$\cos(\vec{x}, \vec{y}) = \frac{< \vec{x}, \vec{y} >}{\|\vec{x}\|_2 \|\vec{y}\|_2}$$

Here $< \vec{x}, \vec{y} >= \vec{x}^T\vec{y}$ is the inner product between two vectors, and it satisfies the requirement that it is invariant to the norm of the vectors. This could be seen as a modification of L2 distance by normalising every signature vector to have norm 1. Thus the cosine similarity can be seen as normalised L2 distance.

$$\begin{aligned} \|\frac{\vec{x}}{\|\vec{x}\|_2} - \frac{\vec{y}}{\|\vec{y}\|_2}\|_2^2 &= 2 - \frac{< \vec{x}, \vec{y} >}{\|\vec{x}\|_2 \|\vec{y}\|_2} \\ &= 2 - \cos(\vec{x}, \vec{y}) \end{aligned}$$

2.2.5 Related work

2.2.5.1 Non-negative matrix factorization methods

As discussed, previous work in this field has mostly focusing on the trinucleotide context of DNA point mutations. Namely, for each locus on the genome that is mutated, the local structure of it is formed by adding the $3'$ and $5'$ flanking bases of it. And therefore, as mentioned above, the total number of types of mutations is $6(substitutions) \times 4(5'bases) \times 4(3'bases) = 96$

In previous methods[50, 54] for mutation signature analysis, a latent feature model is proposed and a non-negative matrix factorization (NMF) method is used for

inference. Here I will give a brief introduction of one of the most popular methods in this field, and many of other methods are built based on this method by adding Bayesian assumption or by using different metric for matrix factorization.

In the following, the number of different types of mutations is denoted by $N = 96$, the number of patient samples is denoted by J and X_{nj} with $n \in \{1, 2, ...N\}$ and $j \in \{1, 2, ...J\}$ denotes the number of mutations of the nth type, e.g. $AGT \rightarrow ACT$, in the jth sample. They are combined into a matrix $\boldsymbol{X}$. By assumption, there are K different types of mutational processes in nature. For each mutational process, it has a signature vector $\vec{\theta_k}$ which indicates the probability that it causes different types of mutations. For example, θ_{nk} denotes the probability for the kth mutation signature to cause a mutation of type n. These signature vectors are put into a matrix $\boldsymbol{\Theta}$, called signature matrix. For each sample, it has a mutational rate vector $\vec{p_j}$, which indicates the total number of mutation caused by each mutation signature for the jth sample. In other words, p_{kj} will indicate the total number of mutations caused by the kth mutation signature in the jth sample, and they are put into a matrix $\boldsymbol{P}$ called mutational rate matrix. Due to the assumption made, every entry in matrix $\boldsymbol{P}$ and $\boldsymbol{\Theta}$ are non-negative values.

Thus, the number of mutation of certain type for a particular sample X_{nj} should be the combination of mutations caused by all mutation signatures.

$$X_{nj} \approx \sum_k \theta_{nk} p_{kj}$$

$$\boldsymbol{X} \approx \boldsymbol{\Theta P}$$

$$\boldsymbol{\Theta} \geq \boldsymbol{0}, \boldsymbol{P} \geq \boldsymbol{0}$$

To solve such problem, it is been converted to an optimization problem.

$$\begin{aligned} \min \quad & d(\boldsymbol{X}, \boldsymbol{\Theta P}) \\ \text{s.t} \quad & \boldsymbol{\Theta} \geq \boldsymbol{0} \\ & \boldsymbol{P} \geq \boldsymbol{0} \end{aligned}$$

Here $d(\boldsymbol{X}, \boldsymbol{\Theta P})$ could be any type of metric that measures the difference between

two matrices. In most classic methods, L_2 distance has been applied, while in others, Kullback-Leiber distance has been used, which has a probabilistic interpretation the X_{nk} follows a Poisson distribution with mean $\sum_k \theta_{nk} p_{kl}$.

The inference algorithm used is a coordinate descent method, where in each iteration, one of $\boldsymbol{\Theta}$ or $\boldsymbol{P}$ is fixed and the other one is chosen to be the value that minimises of the loss function based on the current values of the fixed matrix.

This method is widely applicable and has been applied to multiple cancer genomic data[13, 50].

2.2.5.2 Extending mutation signatures to longer nucleotide sequences: `pmsignature`

Despite the popularity of such method, it does come with a few drawbacks. One of which is that it only considers one flanking base on each side of the mutated base making the mutated nucleotide sequences contain only 3 bases in total. It is quite natural to consider if different signatures could be found when the number of bases being considered is extended to more than 3 bases. For example, mutation signatures based on a pentanucleotide context have been studied[6]. Besides, there is also other evidence showing that extra base pairs will potentially be informative for mutation signatures[52, 85]. It has been reported that except for the immediate flanking bases, the bases which are next to the immediate flanking bases have a disproportional distribution of the compositions. As shown in Table.2.1, the position ± 1 indicates the immediate flanking bases and ± 2 indicates the bases next to the immediate flanking bases. It is clear that for the -2 position, there is a strong preference for A and T, which might be associated to the mutation signatures.

Though it is possible to formulate this problem as a NMF just the same as the case where 3 bases are considered, it is not a good idea to do so as only limited number of mutations per sample is detected (≈ 3000 for whole genome sequencing with 30X depth). For example, if two flanking bases on both sides are considered, then in total, the number of different types of mutation will be 4^4(flanking bases) $\times$ 6(substitutions) $= 1536$. Thus, in most samples, mutations of certain types are never observed, and to make things worse, the number of different types of mutation

Position	A	C	G	T
-2	0.34	0.14	0.14	0.38
-1	0.57	0.11	0.17	0.15
+1	0.45	0.05	0.24	0.25
+2	0.21	0.20	0.25	0.34

Table 2.1. Distribution of the composition around mutated base. Position indicates the position of the base relative to the mutated base. Each cell indicates the proportion of the base at such position.

grows exponentially with the number of flanking bases being considered. As a result, the NMF will become unstable due to the small size of samples compared to the number of parameters in the model. This phenomena is one example of curse of dimensionality in machine learning mentioned earlier[45].

With this being said, new assumptions must come in for the study of mutation signature when considering more than 3 bases. One method that tries to solve this problem is called `pmsignature`. This uses a Naive Bayesian assumption on the probability that a nucleotide sequence is being mutated (Fig.2.2). The probability of a mutational signature to cause mutations within a pentanucleotide context is the product of the preference for each base by the signature.

$$
\begin{aligned}
p(TTCCC- > TTTCC) &= p_1(T)p_2(T)p_3(C- > T)p_4(C)p_5(C) \\
p(TTCTC- > TTTTC) &= p_1(T)p_2(T)p_3(C- > T)p_4(T)p_5(C) \\
\frac{p(TTCCC- > TTTCC)}{p(TTCTC- > TTTTC)} &= \frac{p_4(C)}{p_4(T)}
\end{aligned}
$$

As shown above, the probability for a mutational process with certain signature to cause a mutation $TTCCC- > TTTCC$, i.e. $p(TTCCC- > TTTCC)$ is determined by the product of the preference for each bases at each position, which are parameters of the model to be learned. For mutations that of similar type and context, e.g. $TTCCC- > TTTCC$ and $TTCTC- > TTTTC$, the difference in the probability of being mutated by the same signature is solely determined by the po-

sition where the base is different. In this example, it is the fourth base which have the preference of $p_4(C)$ and $p_4(T)$ respectively. Thus, in total, to determine the probability of different types of mutations $X_1X_2X_3X_4X_5 -> X_1X_2YX_4X_5$ caused by a single mutational process, which is the definition of mutation signature here, it is then only necessary to learn the parameters of $p_n(X_i)$ for $i = 1, 2, 4, 5$ and $p_3(X_3 -> Y)$. There are 4 bases for $X_i \in \{A, C, G, T\}$ when $i = 1, 2, 4, 5$, and 2 bases for X_3 (due to the reverse complementary structure) and 3 bases for Y (because $Y \neq X_3$). In total, there is only $4 + 4 + (2 \times 3) + 4 + 4 = 22$ parameters needs to be learned from data, which is much smaller than $4^4 \times 6 = 1536$.

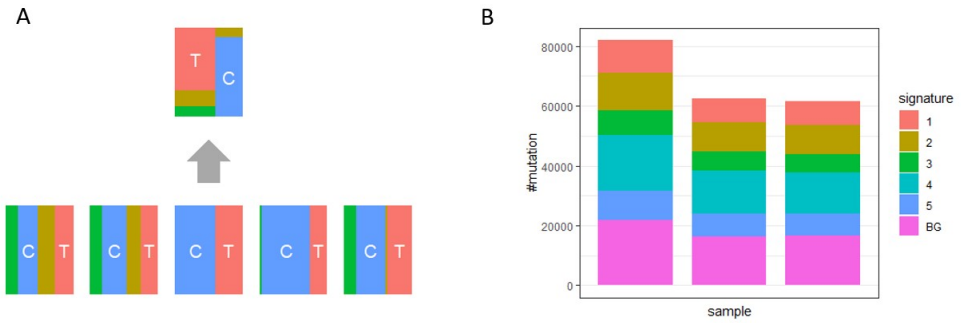


Figure 2.2. `pmsignature` ***(A):Naive Bayesian assumption: the probability of a pentanucleotide sequence to be mutated by the signature is determined by the product of relative preference (area of the base). (B)*** `pmsignature` ***assumes a mixture model for different signatures. The total number of mutations are assigned to each mutation signature.***

Besides, `pmsignature` has made a probabilistic model for mutation generating process: the mutations come from a mixture distribution caused by different mutational process, different mutational processes have certain probabilities causing each type of in different patients.

Compared to other method used for mutation signature detection with extended nucleotide sequence[50], the advantage of `pmsignature` is that it has a clear probability model which will make the inference algorithm robust.

However, the biggest problem with `pmsignature` is that it is not clear how to choose

the number of signatures and it has to be determined before the training starts.

To tackle such problem, I developed a new method called `convSig` which could be applied to longer nucleotide sequence. Because it is been built on a supervised problem, it is possible to apply cross-validation to choose the number of signatures[31]. The model of `convSig` is based on the convolution operation with a sound biological interpretation, while the model of `pmsignature` is built on the Naive Bayesian assumption which does not have a biological support.

2.2.5.3 My contribution

My contribution to this problem are listed below.

- I formulate the problem as a supervised classification problem, which allows people to use all types of classification algorithms being developed before in machine learning area.
- I solve the problem of choosing number of signatures by cross-validation.
- With deep neural network approaches, I extend the analysis to longer nucleotide sequences than existing tools.
- I show that my model provides a better view of the biological mutational process on the genome.

2.3 Convolution based mutation signature detection: `convSig`

I will assume that the genome is a sequence of H bases consisting of $H - L$ overlapping L nucleotide sequences. There are $N = 4^L$ unique nucleotide sequences (in case of 3 bases, these are $\{AAA, AAC, ..., TTG, TTT\}$) which can be encoded using a one-hot encoding matrix (see Figure 2.3). It is then denoted by w_n where $n \in \{1, 2, ..N\}$ the total count of the presence of the n-th nucleotide sequence for all types of nucleotide sequences in the non-mutated genome. Given the mutated cancer genome, it is denoted by $X_{n,y}$ the number of nucleotide sequences of type n that have been mutated to type $y \in \{1, 2, 3\}$, which is one of the other 3 bases.

I will assume for exposition that the data consists of only a single normal-cancer genome pair but the following discussion extends trivially to multiple samples.

The mutations are assumed to be driven by K mutational processes. For the k-th mutational process, where $k \in \{1, 2, ...K\}$, it is assumed the weights corresponding to its mutational filter is denoted by $F^k \in \mathbb{R}^{Q \times 4}$ where Q corresponds to the filter length. (In this chapter, except for deep model, $Q = L$ will hold.) Its preference for the n-th nucleotide sequence type T_n is denoted by $\theta_{nk} = f(T_n, F^k) = h(T_n * F^k)$ where $h(x)$ is a transformation/activation function detailed later and $*$ is the convolution operator. The probability that each mutational processes causes a mutation is p_k (the mutational rate), and the probability for different types of mutation is denoted by $M^k_{x,y}$, where $x \to y$ denotes the mutation type (e.g. $C \to G$). I will always assume that the base in the centre of the nucleotide sequence is mutated. The total probability that a nucleotide sequence of type n, with central base x, to be mutated to y is $\sum_k p_k \theta_{nk} M^n_{x,y}$ (see Figure 2.3).

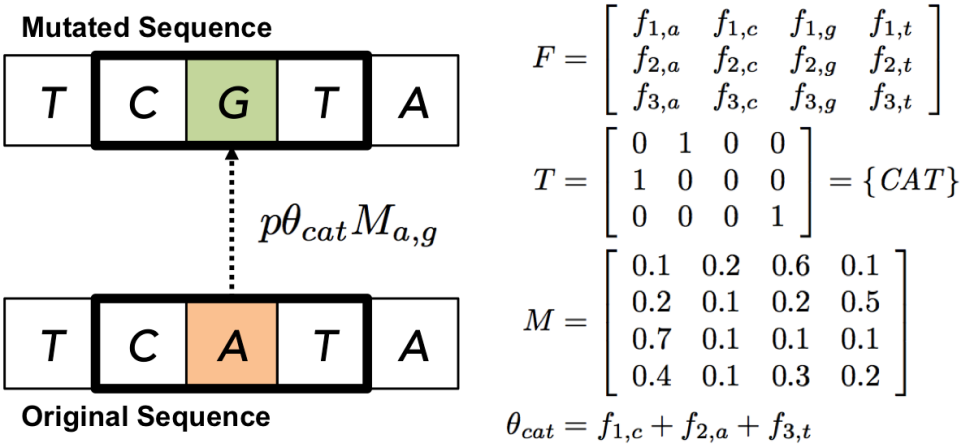


Figure 2.3.* Discrete convolution in human genome. *Compute the convolution product θ of a filter F of size 3bp and nucleotide sequence denoted by T. For filters of size 3bp, only the immediate flanking bases on each side has been taken into convolution.

Here, it is aimed to use a machine learning method to learn the mutational rate, mutational filter weights and mutational types $\{p_k, F^k, M^k_{x,y}\}$ given data $\{w_n, X_{n,y}\}$. I define the minimisation of negative log likelihood function as shown below. The first term is the negative log likelihood for mutated bases and the second term is

that for unmutated.

$$\min \quad -\sum_{n=1}^{4^L}\left[\underbrace{\sum_{y=1}^{3} X_{n,y}\log\left(\sum_{k=1}^{K} p_k\theta_{nk}M^k_{\mathrm{mid}(n),y}\right)}_{\text{Mutated}} - \underbrace{(w_n - \sum_y X_{n,y})\log(1-\sum_k\sum_y p_k\theta_{nk}M^k_{x,y})}_{\text{Unmutated}}\right] \tag{2.1}$$

$$\begin{aligned}
&\approx -\sum_{n=1}^{4^L}\left[\underbrace{\sum_{y=1}^{3} X_{n,y}\log\left(\sum_{k=1}^{K} p_k\theta_{nk}M^k_{\mathrm{mid}(n),y}\right)}_{\text{Mutated}} - \underbrace{w_n\sum_{k=1}^{K} p_k\theta_{nk}}_{\text{Unmutated}}\right] \\
&\text{s.t.} \quad \forall k, \sum_{n=1}^{4^L}\theta_{nk} = 1, \forall k, x, \sum_{y=1}^{3} M^k_{x,y} = 1 \\
&\qquad \forall n, k, \theta_{nk} = h(T_n * F^k)
\end{aligned} \tag{2.2}$$

where mid(n) denotes the unmutated middle base of the n-th nucleotide sequence type and it is assumed here that the mutational rate is typically very small ($10^{-6} \sim 10^{-4} \ll 1$) so the number of mutated nucleotide sequences is expected to be much less than the total number, i.e. $X_{n,y} \ll w_n$. Thus, in the second line the following approximations apply $w_n - \sum_y X_{n,y} \approx w_n$ and $\log(1 - \sum_k \sum_y p_k\theta_{nk}M^k_{x,y}) \approx -\sum_k \sum_y p_k\theta_{nk}M^k_{x,y}$.

2.4 Model variants

I explored two versions of my model using an exponential and rectifier transformation for $\theta_{nk} = h(T_n * F^k)$. Both ensure positivity for θ_{nk} which defines the attraction of the k-th mutation process to the n-th nucleotide sequence type. Each transformation requires specific approaches for statistical inference which I will detail in the following.

2.4.1 Exponential transformation

Theorem 2.1. *The local optimum for maximum likelihood parameter estimation can be computed using an expectation-maximization algorithm by iterating over the*

following 4 steps:

$$z_k^{n,y} = \frac{p_k \theta_{nk} M^k_{mid(n),y}}{\sum_k p_k \theta_{nk} M^k_{mid(n),y}}, \tag{2.3}$$

$$M^k_{x,y} = \frac{\sum_{n:mid(n)=x} X_{n,y} z_k^{n,y}}{\sum_{n:mid(n)=x} \sum_y X_{n,y} z_k^{n,y}}, \tag{2.4}$$

$$p_k = \frac{\sum_n \sum_y X_{n,y} z_n^{,y}}{\sum_n w_n \theta_{nk}}, \tag{2.5}$$

$$\eta^k_{l,b} = \frac{P^k_{l,b}/W^k_{l,b}}{\sum_{b'} \frac{P^k_{l,b'}}{W^k_{l,b'}}}, \tag{2.6}$$

where l goes from 1 to L for each filter and other variables are defined as:

$$\eta^k_{l,b} = \exp(F^k_{l,b}) / \sum_b \exp(F^k_{l,b}),$$

$$\theta_{nk} = \prod_l \eta^k_{l,b},$$

$$P^k_{l,b} = \sum_{n:l(n)=b} \sum_y X_{n,y} z_k^{n,y},$$

and

$$W^k_{l,b} = \sum_{n:l(j)=b} w_n \prod_{l' \neq l} \eta^k_{l',b'}$$

where $n : l(n) = b$ indicates all nucleotide sequences with lth base being b. mid is the base being mutated, which is by convention in the middle of the nucleotide sequence.

The proof for it is in the appendix section.

The `convSig` model using exponential transformation is quite similar to the model for `pmsignature`. The variable $\eta_{l,b}$ in Theorem 2.1 is equivalent to the $p_l(b)$, i.e. probability for the lth base of the mutated nucleotide sequence to be base b for `pmsignature`. The difference is in the loss function used. The loss for `pmsignature` is shown below. For `pmsignature`, they only have the mutated term for loss (2.1) but not the unmutated part, with an additional optimisation constraint that $\sum_k p_k = 1$. This loss function shown below will very likely decrease with more number of signatures used and thus makes the choice of the number of signatures a difficult

task for `pmsignature`.

$$\begin{aligned}
\min \quad & -\sum_{n=1}^{4^L}\left[\sum_{y=1}^{3} X_{n,y}\log(\sum_{k=1}^{K} p_k\theta_{nk}M^k_{\text{mid}(n),y})\right] \\
\text{s.t.} \quad & \forall k, \sum_{n=1}^{4^L}\theta_{nk}=1, \forall k,x, \sum_{y=1}^{3} M^k_{x,y}=1 \\
& \forall n,k,\theta_{nk}=h(T_n * F^k) \\
& \sum_k p_k = 1
\end{aligned}$$

2.4.2 ReLU transformation

Now I will discuss the rectifier transformation function, i.e. $h(x)=\max(0,x)$, while keep every thing else the same as before. The reason for considering this alternative transformation is that though the exponential function matches the requirement for non-negativity and being monotonic, however, it never goes exactly to zero. The rectifier allows for the possibility of obtaining sparse results (more zero entries in F^k) that potentially simplifies the biochemical interpretation of the mutation process.

This modification makes the inference considerably more challenging. Here, I proposed a heuristic method to solve a relaxed version of the optimization problem such that, whereas before it is required that $\theta_{nk}=h(T_n * F^k)$, and now this constraint is relaxed to $\|\theta_{nk}-h(T_n * F^k)\|_2 \leq \epsilon$ and form a Lagrange dual with respect to this constraint. This is similar to the idea used in alternating direction method of multipliers (ADMM)[88], but the problem here is highly non-convex, which makes it difficult to solve. I still use a coordinate descent method to optimize over each term:

$$\begin{aligned}
\min . & -\sum_n\left[\sum_y X_{n,y}\log\left(\sum_k p_k\theta_{nk}M^k_{x,y}\right) - w_n\sum_k p_k\theta_{nk}\right] \\
& +\frac{1}{2}\lambda\sum_{nk}\|\theta_{nk}-h(T_n * F^k)\|_2^2 \\
\text{s.t.} \quad & \forall i, \sum_n \theta_{nk}=1, \forall k,x, \sum_y M^k_{x,y}=1
\end{aligned}$$

Theorem 2.2. *The following conditional update for θ_{nk} minimizes the conditional*

log likelihood when all other variables are fixed:

$$\theta_{nk} = \max(0, \theta^*_{nk}),$$

where

$$\gamma_{nk} = h(T_n * F^k)$$

,

$$\alpha^k_{n,y} = \frac{\sum_{k' \neq k} p_{k'} \theta_{nk'} M^{k'}_{x,y}}{p_k M^k_{x,y}},$$

$$\omega_{nk} = w_n p_k$$

and

$$\sum_y \frac{X_{n,y}}{\alpha^k_{n,y} + \theta^*_{nk}} - \lambda\theta^*_{nk} + \lambda\gamma_{nk} - \omega_{nk} = v,$$

where v varies to ensure $\sum_j \theta_{nk} = 1$ with different values for θ^*_{nk}.

The proof is detailed in the appendix section.

Other parameters, i.e. p_k and $M^k_{x,y}$ can be updated as before under the exponential transformation and the only problem left is to find the optimal value for convolution filters F^k.

The optimisation (2.1) with respect to the filter weights F^k could be simplified into the following form. To solve this problem, Newton's method can be applied[88].

$$\min \quad \sum_j \frac{1}{2} \left[\theta_{nk} - h(T_n * F^k)\right]^2$$

2.5 Deep convolutional filtering

Some cancers exhibit mutational patterns in which there are genomic regions where mutations tend to occur very often, with the density of mtations greater than 1 per 1000 base pair (1Kbp), while in other regions, the density is less than 1 per 1 million base pair (1Mbp)[54]. In the case of highly mutated regions, the sequence context driving the local mutational rate may extend more than a few base pairs flanking the mutations. Previous approaches and the model I have described so far

are not suitable to explore larger sequence nucleotide sequences since they require enumerating every possible nucleotide sequence as a feature. However, in the case of the convolutional model, it is possible to adopt the principles underlying deep convolutional network to propose a multi-layered solution to examining long nucleotide sequence mutation signatures.

Here, I use the previous model as guidance for the construction of a deep convolutional model. Note that the original problem I wish to solve was to minimize the objective (2.1).

In a deep learning model, a modification is made. Here, y takes value in $\{1, 2, 3, 4\}$ compared to $\{1, 2, 3\}$, and when $y = 4$, it indicates the nucleotide sequence has not been mutated. By this modification, it is possible to merge the unmutated part in the problem (2.1) into the first term. This makes the following math look simpler. Therefore, it is possible to transform the problem (2.1) into the following optimisation problem

$$\begin{aligned} \min . & -\sum_n \left[\sum_y X_{n,y} \log \left(\sum_k p_k \theta_{nk} M^k_{x,y} \right) \right] \\ \text{s.t.} \quad & \forall k, \sum_n \theta_{nk} = 1 \\ & \forall k, x, \sum_y M^k_{x,y} = 1 \\ & \forall n, \sum_y \sum_k p_k \theta_{nk} M^k_{x,y} = 1 \end{aligned}$$

In this case, $\theta_{nk} = h(T_n * F^k)$ now denotes multiple convolutional layers: in this examples, I will use four repeated layers of convolution, transformation and max-pooling (F^k indicates the weights and biases in all these layers), and due to such a construction, the size of F^k only grows logarithmically with respect to the length of nucleotide sequences, while it grows exponentially for methods based on NMF and linearly for `pmsignature`. Inference for deep networks are usually solved by first-order (sub)gradient methods in which constraints can be tricky to implement. I use the ADAM method to numerically optimize this model[89].

For this model, there are three constraints in total. The first one is to prevent θ_{nk}

and p_k from becoming interchangeable, and this can equivalently be prevented by adding regularization penalties for convolution parameters in the model. For the second constraint, I defined a new variable $a_y^k = p_k \times M_{x,y}^k$, and to ensure that a_y^k is non-negative (and therefore $M_{x,y}^k$), a ReLU layer is used to truncate any non-negative value. The final constraint is enforced by adding a normalization layer at last.

Since the purpose for this model is to infer mutational processes rather than prediction, the model is not as huge as traditionally used in computer vision[90]. There are 8, 12, 16 and 20 kernels for each convolution layer respectively from bottom to top, and the filter lengths are set to 8 for all convolution layers and the pooling is on size of 4.

For previous models, the interpretation for model parameters is quite straight forward, whereas for the deep model, the interpretation of mutation signatures is more challenging due to the deep convolutional structure of the filters. I thus adopt an gradient approach which is often used in the field of computer vision for understanding information contained in higher layers of CNN models[91].

2.6 Experiments

I performed a series of numerical studies to understand the performance of the proposed approach for mutation signature identification. I begin with a comparison based on simulated sequence count data using 3bp and 5bp nucleotide sequence sizes respectively. The former makes it possible to make comparisons across existing algorithms whilst the latter demonstrates the ability to go beyond the common state-of-the-art. I then repeat the exercise on real whole genome sequencing data. Finally, I illustrate the potential utility of deep convolutional mutational filters for identifying long-range mutational signature patterns.

2.6.1 Simulation study

I simulated sequence count data based on the generative model in the `pmsignature` approach[52]. I chose `pmsignature` as it shows good performance and has been

successfully applied to clinical studies[92, 93] for the analysis of mutation signatures associated with longer nucleotide sequences. The methodology adopted by `pmsignature` utilises a mixed membership (topic model) based approach.

In the first simulation, I generated mutated trinucleotides and used five mutational signatures (No. 1, 2, 13, 17 and 21) previously reported[1] with random noise. As in the assumption of `pmsignature`, the mutations caused by the same signature will follow a Naive Bayesian assumption, so that I only leave the major peaks in the graph of mutation signature that follow this structure. For example, for signature 17, I use the major peak and assume this signature only cause $CTT \rightarrow CGT$ mutations. And for signature 1, there are 4 major peaks and as they follow the Naive Bayesian assumption, so that I leave all of them , which means it only causes $XCG \rightarrow XTG$ mutations, where X could be any of the 4 bases.

I set the mutational rate of the background signature which has equal probability for all types of mutations to be 10^{-6}, and that of other signatures to be some random number uniformly sampled between 0 and 2×10^{-5}. The stochastic error was simulated to follow a Poisson distribution, and thus, in total, there were approximately 4,000 mutations per sample, which is in the range of what is observed in whole genome sequencing of non-hypermutated cancers. I simulated 10 cancer samples per dataset, and the experiment is repeated for 5 times.

I applied the methods, both exponential and ReLU transformations of `convSig`, to each dataset independently. I tested versions of `convSig` using both the exponential and ReLU transformations. I also compared to existing algorithms, including: `BayesNMF`, `EMu` and `pmsignature`. I used cosine similarity between the estimated and simulated mutation signatures, as a criterion for comparison. All algorithms were run with the number of mutation signatures to be detected equal to the truth.

Figure 2.4 shows that `pmsignature` naturally has the best performance since the data was simulated under its model, however, `convSig` using the exponential transformation has comparable performance. Using the ReLU transformation, `convSig` has comparable performance to `BayesNMF` and `EMu`.

Next, I examined whether it is possible to use `convSig` to identify the number of

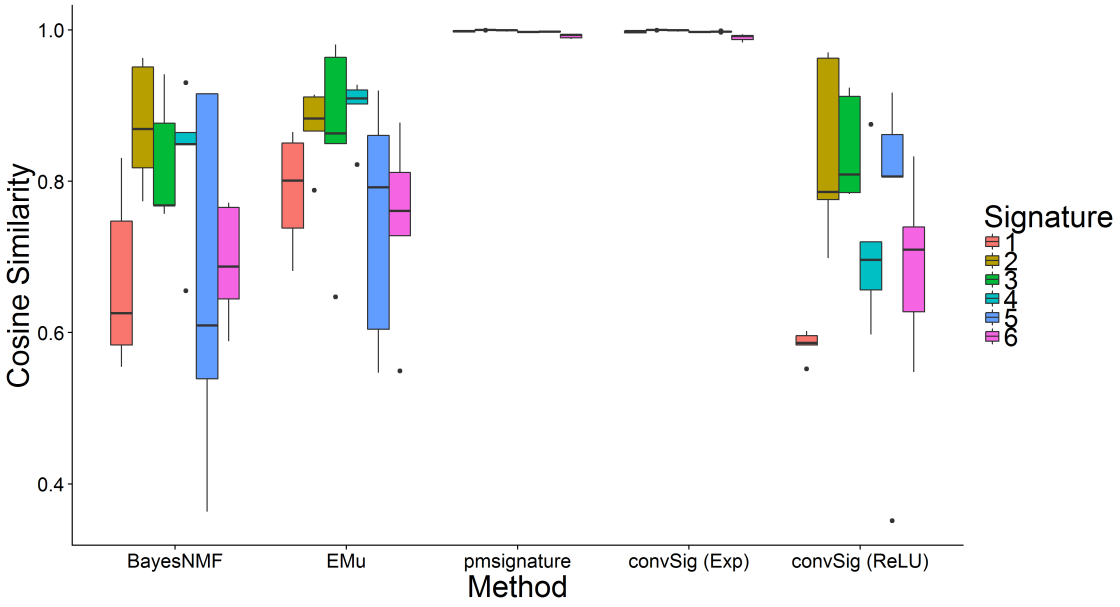


Figure 2.4.* 3-base results on simulations. *The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 3-bp nucleotide sequences.

mutation signatures present in the simulated data using a cross-validation procedure. Figure2.5 shows that the decrease in the relative training and test loss tapers off at approximately the true number of signatures (5 signature+1 backgroud noise). This suggests that `convSig` is able to identify the number of mutation signatures to a high degree of accuracy.

I next simulated nucleotide sequence that were 5bp in length. This is substantially more difficult than the 3bp case and only `pmsignature` is designed to handle this scenario among the methods compared to. I incorporated three mutation signatures and one background signature, but kept other simulation parameters the same as in the previous simulation.

Figure2.6 shows `pmsignature` exhibits the best performance in terms of mutation signature recovery since the data was generated from its assumed mode. However, using the exponential transformation, `convSig` performed adequately. For ReLU transformation, `convSig` could not identify the three mutation signatures. If the underlying mutational rate is increased by 100X, the ReLU model gives a satisfactory result, which indicates it may only work with sufficient number of mutations. Note that the ReLU model induces sparsity through its transformation which may not be

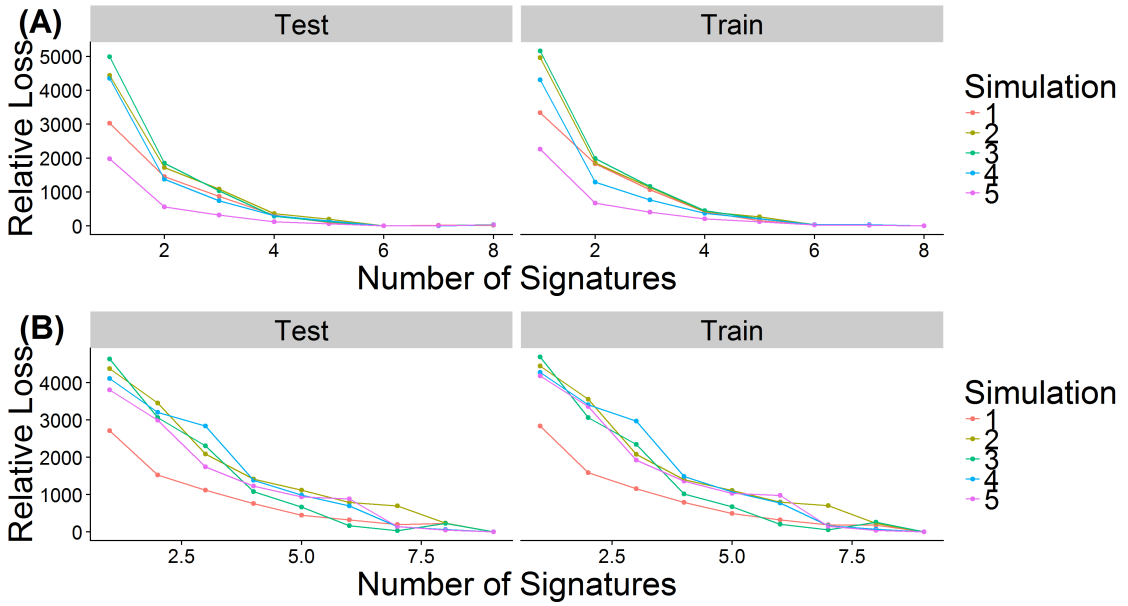


Figure 2.5. **Estimating the number of signatures.** ***(A) Test and training error for*** `convSig` ***with the exponential transformation. (B) Test and training error for*** `convSig` ***with the ReLU transformation.***

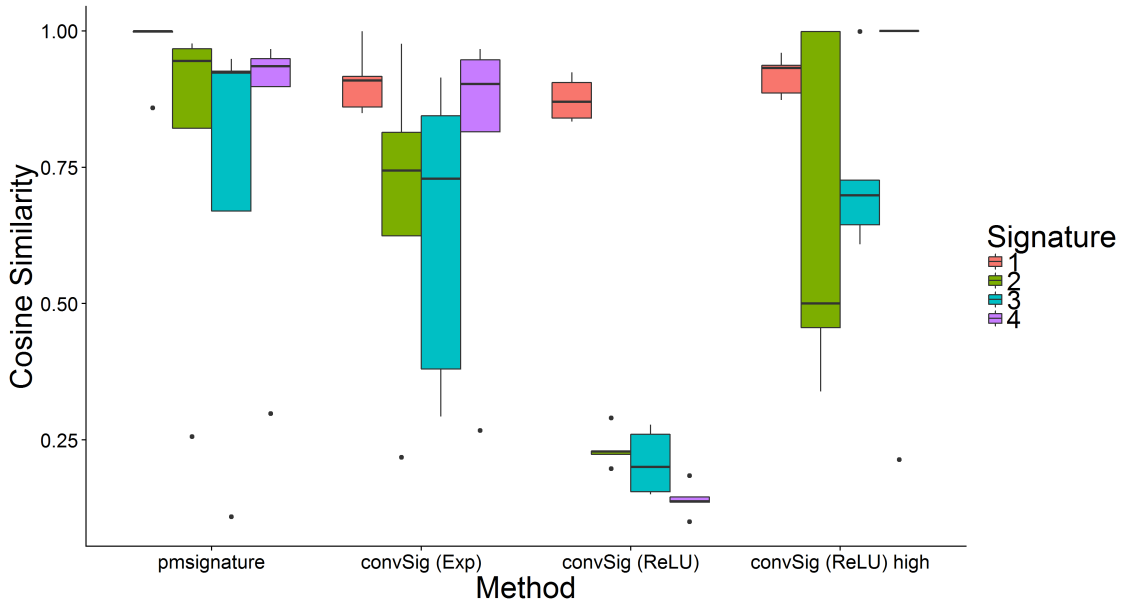


Figure 2.6. **5-base results on simulations.** ***The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 5 bp nucleotide sequences.***

applicable in this simulation.

2.6.2 More complex simulations

As shown in the previous examples, I simulated data using the assumption based on `pmsignature` method, and showed that in all these cases, this method achieved satisfactory results among different existing methods, although `pmsignature` still did the best among all methods.

I then go on to simulate data based on the assumption of `EMu`. In this simulation study, I removed the Naive Bayesian assumption made by `pmsignature`, and use the reported signatures from literature exactly as they are.

I set the mutational rate of the background signature to be 10^{-6}, and that of other signatures to be some random number uniformly sampled between 0 and 2×10^{-5}. The stochastic error was simulated to follow a Poisson distribution. I simulated 10 cancer samples per dataset, and the experiment is repeated for 20 times.

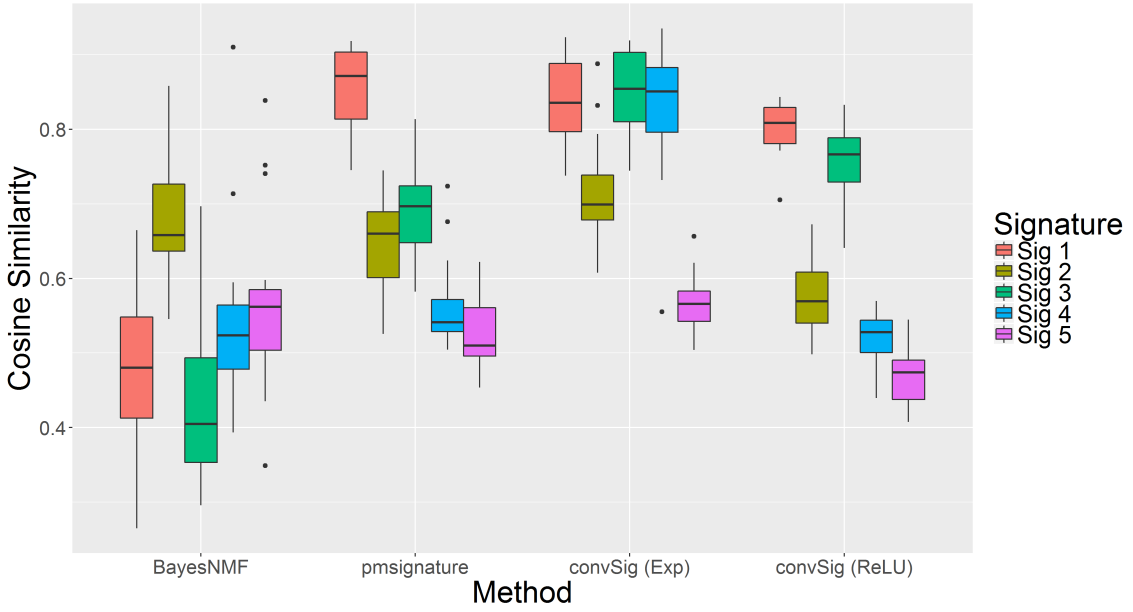


Figure 2.7.* 3-base results on complex simulations. *The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 3-bp nucleotide sequences.* `EMu` *is not included as it only outputs one signature which means it fails to capture the difference between signatures possibly due to small sample size.

The results showed (Fig.2.7) that for 3 base signatures, my method `convSig` using exponential transformation was better than `pmsignature` in reconstruction of three signatures (Sig 2-4) and achieved similar results in the rest two (Sig 1 and Sig 5).

`EMu` results was removed from the Fig.2.7 because `EMu` only outputs 1 signature, which indicated that it failed to capture the differences between signatures for this task and made it meaningless to compare it to other methods.

I again tested the performance of `convSig` at choosing the correct number of signatures from these simulated samples, which is a challenging task for `pmsignature`[52]. It showed (Fig.2.8) that the exponential transformation of `convSig` could in most cases, correctly infer the number of signatures, which equals 5 in all cases. And in all cases, the number of signatures detected are between 4-6. However, the ReLU transformation of `convSig` is less accurate at recovering the number of signatures and the reported results are between 3-7.

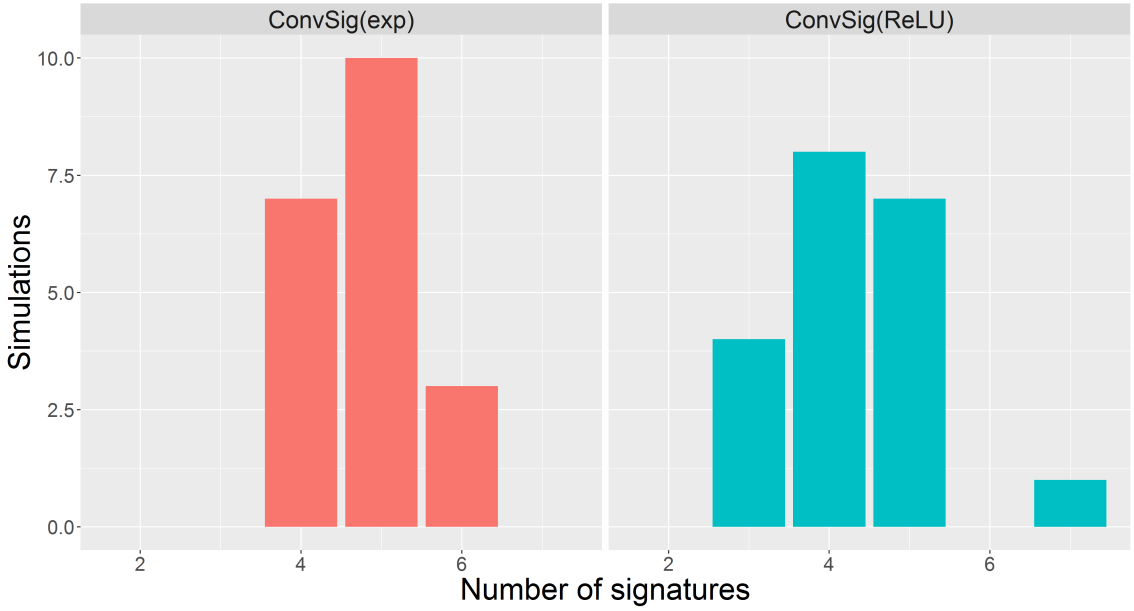


Figure 2.8. **Estimating the number of signatures.** ***(A)Number of signatures detected by*** `convSig` ***with the exponential transformation. (B)Number of signatures detected by*** `convSig` ***with the ReLU transformation.***

I also simulated data on 5 base signatures similarly. The results(Fig.2.9) showed that the exponential transformation of the method `convSig` was better at reconstructing of 4 signatures (Sig 1-4) and achieved similar results on the last one (Sig 5). And the ReLU transformation was better at 4 signatures (Sig 1-3 and Sig 5), and achieved similar results on Sig 4.

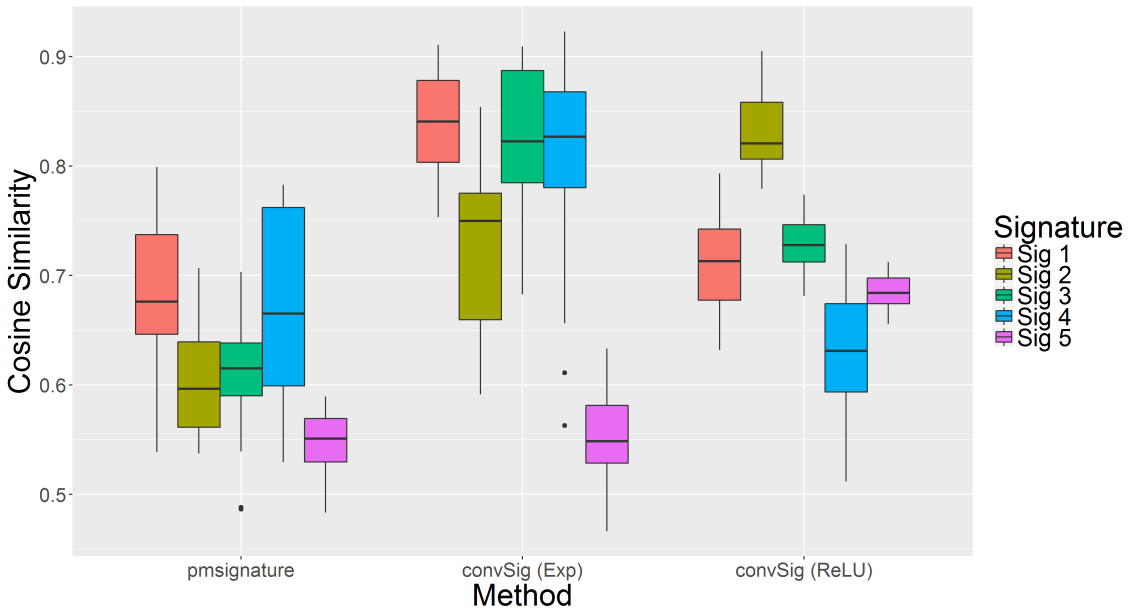


Figure 2.9. **5-base results on complex simulations.** ***The cosine similarity between computed and simulated mutation signatures for different mutation signature identification approaches across 5 simulations each involving 10 simulated cancers using 5 bp nucleotide sequences.***

2.6.3 XERXES clinical trial data

I next applied `convSig` to whole genome sequencing data from the XERXES clinical trial[1][2]. This Phase 1/2 clinical trial involves the collection of patient tumour samples to study the effect of chemotherapy and radiotherapy in the treatment of rectal cancer patients. I examined 22 samples in total. The data were collected from 6 patients with 3-4 samples for each patient at different time during their treatment.

The experimental steps are listed below. First, DNA was extracted from tumour regions and blood samples using the Qiagen DNeasy kit. Sequencing library is then prepared by NEBNext DNA kit. Next, DNA fragments were sequenced using standard protocols on the Illumina HiSeq 2000, with the FastQC program used to check the sequencing quality of raw data. The sequencing depth was 30x and it has been assessed using the GATK package[94]. Mutation calling was carried out by the Platypus tool[95]. Reported mutations were filtered by the following requirement: (1) there were greater than 10x depth in tumour sample; (2) either there were fewer

[1]http://www.cancerresearchuk.org/about-cancer/find-a-clinical-trial/a-study-looking-giving-biological-therapy-as-well-as-combined-chemotherapy-and-radiotherapy-before-surgery-cancer-back-passage-xerxes

[2]https://www.evidence.nhs.uk/

than 40x depth and no mutant reads in the the normal sample or greater than 40× depth and fewer than 1 mutant read in the normal sample (3) the allele frequency is greater than 1%.

Fig.2.10 shows a brief summary of the number of different mutations in the samples.

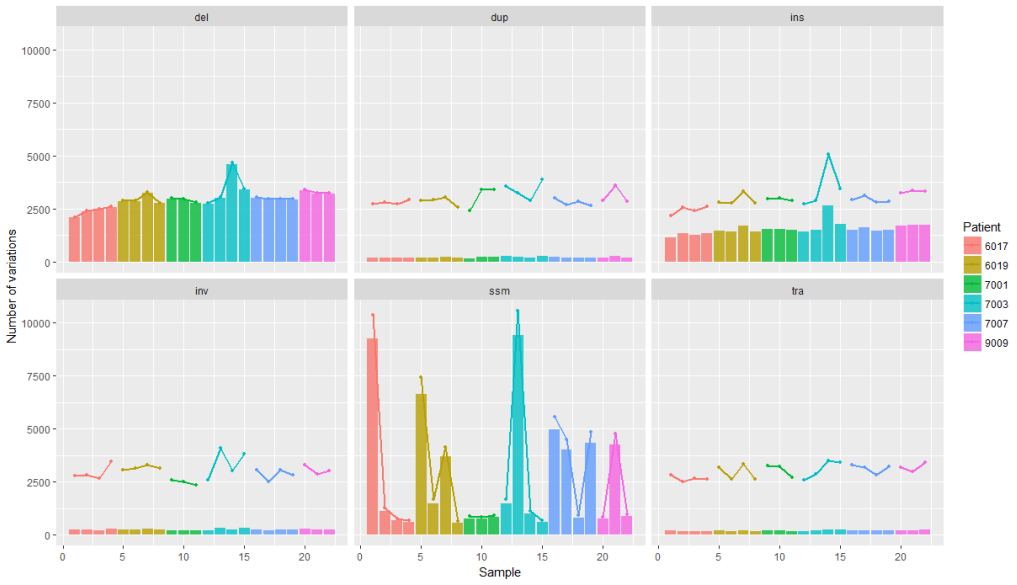


Figure 2.10.* Statistics of mutations in XERXES data. *Number of deletions, duplications, insertions, inversions, simple somatic mutations and translocations in each sample.

In order to establish a ground truth for benchmarking on this real data set, I used the 30 trinucleotide mutation signatures from[1]. I used penalised regression based on LASSO method, which is commonly used for feature selection in statistics[31], with non-negative constraint to estimate the contribution of these 30 signatures to the trinucleotide distribution in the XERXES trial patients. I found that only 4 (No.1, 6, 10, 18) of the 30 signatures[1] were present in the XERXES patients and used them as a ground truth to test how each method recovers them.

Table 2.2 shows `convSig` using both exponential transformation and ReLU function outperforms the other methods on real data. `EMu` fails to deconvolve the mutational profile and only outputs a single background signature. Both `BayesNMF` and `pmsignature` under-performs `convSig`. Interestingly, whilst the use of the ReLU transformation in `convSig` led to relatively worse performance in the simulation study, it is the best performing on real data. This is likely due to the fact that, in simulation, the data is idealized and noise is assumed to be Poisson distributed.

Table 2.2. Cosine similarity of detected versus four Alexandrov mutational signatures for 3-base nucleotide sequences in the XERXES clinical sequence data. The 4 signatures correspond to No.1, 6, 10, 18 of the 30 signatures reported[1].

	Signature			
Method	1	2	3	4
BayesNMF	0.76	0.32	0.47	0.48
EMu	NA	NA	NA	NA
pmsignature	0.75	0.74	0.69	0.49
convSig (Exp)	0.73	0.77	0.73	**0.94**
convSig (ReLU)	**0.85**	**0.85**	**0.76**	0.77

In real sequencing data, the noise processes and the mutation signatures are more complex and the sparsity inducing properties of the ReLU transformation maybe help to regularize the output.

Table 2.3. Similarity between methods on 5-base nucleotide sequences for XERXES clinical data

Methods	Cosine Similarity
pmsignature & Exp	0.67, 0.78, 0.63, 0.58, 0.60
Exp & ReLU	0.74, 0.45, 0.81, 0.62, 0.74
ReLU & pmsignature	0.68, 0.52, 0.63, 0.55, 0.41

In the case of 5-bp nucleotide sequence analysis, there are no reference signatures from which to determine a benchmark. However, it is possible to compare the results of `convSig` method to that of `pmsignature` to determine concordance. I found there was good similarity between the signatures found by all these algorithms (Table 2.3). The `convSig` variants shared the most similarity in terms of the five signatures discovered.

Besides, I truncated the discovered 5-bp mutation signature to 3-bp mutation sig-

nature and compare it to the 3-bp signature detected by the same method. I found that results are generally consistent. The value in Table.2.4 indicates the similarity between the truncated signature and the closest 3-bp signature. The extra 2-bp from the 5-bp signature is informative as there is an imbalance between the convolution filter weights. For example, among 3 of the 5-bp signatures found by `convSig` (2 of the signatures found by `pmsignature`), there is a strong preference for the base A at the 5' end.

Table 2.4. Similarity between truncated 5-base signature to 3-base signature.

Methods	Cosine Similarity
Exp	0.90, 0.91, 0.99, 0.89, 0.91
ReLU	0.85, 0.64, 0.64, 0.70, 0.97
pmsignature	0.98, 0.91, 0.90, 0.75, 0.80

2.6.3.1 Deep convolutional filtering

Finally, I evaluated the application of deep convolutional filtering for mutation signature identification across large nucleotide sequences of 1Kbp. This analysis would enable the assessment of mutagenic processes that affect larger genomic regions, for example, mutation hotspots in which multiple mutations occur within close proximity to each other[54].

To assess whether a deep `convSig` could distinguish between highly mutated and lowly mutated genomic regions, I empirically separated all mutated 1Kbp nucleotide sequences in the XERXES dataset into one of two classes: high and low, where high indicates nucleotide sequences that lie in a highly mutated region. A highly mutated region is defined as any region where the distance between two mutations is less than 1Kbp. I trained a deep `convSig` model mentioned before using the XERXES sequence data.

Figure 2.11A shows that after training, the deep `convSig` model was able to separate the high and low mutation density regions from one another suggesting that the classification model is working properly. I identified that the twenty outputs of

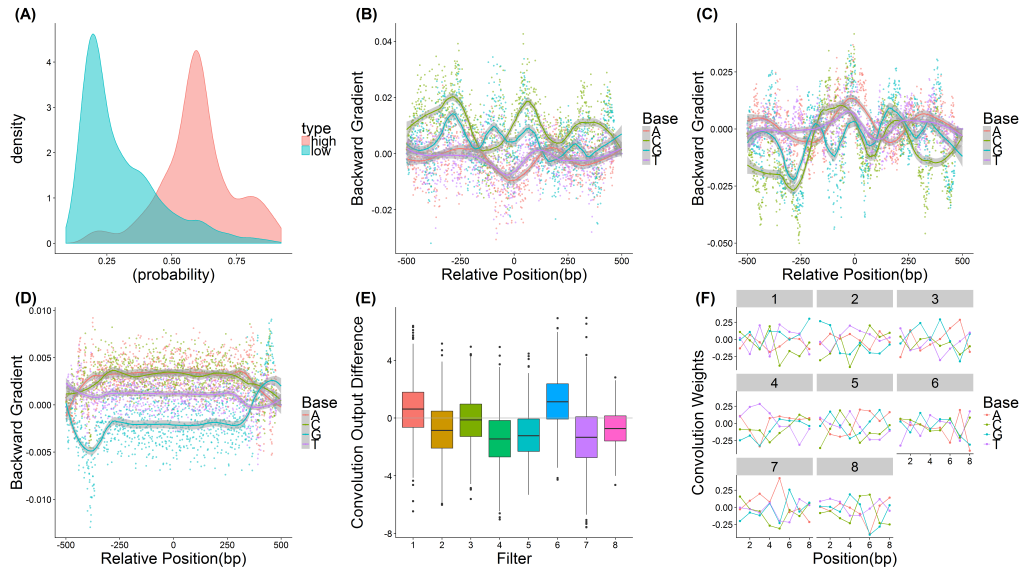


Figure 2.11.* Deep convolutional filtering.** ***(A) Distribution of output probabilities from the deep convolutional model for each sequence nucleotide sequence to be in a highly mutated region. The highly mutated (red) nucleotide sequences and low mutation regions (blue) are well separated in the trained model. (B) Average gradient of the signatures generating mutations in highly mutated regions. (C) Average gradient of the signature generating mutations in low mutation regions. (D) Average gradient of the signatures having preference for either low or high mutation regions. (E) Difference in output from first convolution layer between nucleotide sequences in highly mutated regions and in low-mutation regions. (F) Weights for all filters in the first convolution layer.

the last convolution layer could be divided into three classes, using hierarchical clustering. I qualitatively divided these into outputs that are associated with highly mutated regions, those associated with low mutation regions, and those that do not have any bias for either regime.

To visualise the 1Kbp mutational signatures, I backpropagated the output of the model to the input nucleotide sequences. I took the average of the gradients for input nucleotide sequences which are in the highly mutated regions as the mutation signatures[91]. This average gradient is shown in Figure 2.11B-D. It is shown that mutation signatures associated with highly mutated regions are enriched for CG content at the point of mutation but also 300 bp downstream and upstream of the mutation(Fig.2.11B). For signatures associated with mutations in low mutation density region, the form of the gradients is essentially an inverted version of the signature associated with high mutation density regions(Fig.2.11C) with low GC content instead at these respective locations. Mutational signatures which have no

preference for high or low mutation regions have a uniform gradient(Fig. 2.11D), which indicates it is a random noise.

Finally, I investigated the first convolution layer of the full CNN in further details. As mentioned before, in the first convolution layer, I have implemented 8 filters. I first took the difference between the output of this layer between the two different classes of mutations(Fig. 2.11E) and found that filter 1 and 6 (slightly above 0) tend to give higher signals for mutations in highly mutated regions. Then, I compared the weights of these 8 convolution filters after training (Fig2.11F). It turned out to be quite challenging to draw any strong conclusion from these wrights because of the noise, but it is still possible to gain some weak insight. For example, for the 6th filter which tends to give higher signal for mutation in highly mutated region, there is a light preference for local structure $GT\{C|G\}\{C|G\}$. The might partially explain the preference for CG rich content for the 1Kbp signature mentioned above.

To sum up, interpretation of the 1kb signature posed a particular challenge. In the above paragraphs, I have illustrated that the classification model is working properly as shown in Fig.2.11A that the two classes are well separated. Fig.2.11B and Fig.2.11C together concluded that for mutations in highly mutated region, there is an enrichment of CG bases in the ± 250 region. However, it is quite difficult to infer why such enrichment will exist from computational method alone. Further experimental studies will be required to verify the findings related to this deep model .

2.7 Discussion

I have presented a novel convolutional filtering based approach for the extraction of mutation signatures from genome sequencing data. In this method, the mutation signature detection problem is posed as a classification problem which is different from previous NMF methods[50]. Such formulation allows the easy choosing of number of signatures, which is a major problem for previous probabilistic based method `pmsignature`[52]. Besides, the convolution filtering idea has a clear biological model that it is based on the local structure of DNA sequence and interaction molecules.

In the simulation studies based on assumptions of `pmsignature`, my methods `convSig` have demonstrated comparable performance to it and achieved better performance than other methods[51, 52, 55]. In more complex simulation studies using assumptions based on `Emu`[51] and clinical findings[1], `convSig` have demonstrated better performance.

In the study of a clinical data set, my methods have demonstrated better performance at reconstructing the previous reported signatures[1] on trinucleotide context and showed concordance findings with existing method on pentanucleotide context where the benchmark is not available.

These results indicate that the use of convolution operation could truthfully capture the characteristics of molecular mutational processes.

I have also demonstrated the use of the deep version of `convSig` to detect mutation signatures associated with 1Kbp nucleotide sequences. This signature has showed a preference for GC rich content for $\pm 300bp$, though more studies are required to verify this finding.

For future direction, this method makes it possible to identify regions of the genome that are particularly vulnerable to some mutagenic process, which is of strong interest in previous studies[51, 54]. For my method, mutation signatures/filters learnt by `convSig` could be applied back to the genome. By sliding the mutation signatures/filters across the genome, the probability for each locus to be mutated by such mutation process could be computed. Then, the region which are more likely to be mutated by certain mutation process associated with some signature could be determined by this method.

The model also enables simple transfer learning. Whilst other methods such as non-negative matrix factorization (NMF) give mutational signatures in the form of relative mutation type frequencies, `convSig` outputs filter coefficients expressing the affinity of a mutational process for certain genomic sequences. Trained filters can be readily applied to new genomes and used to assess the presence of that mutational process in the new data. In contrast, some form of regression analysis would be required to project new data on to the low-rank factor matrices from NMF[96]. For

example. in cases where the number of sample is limited, by using this transfer learning method, the mutational rate p_k could be learned, with all other variables kept unchanged in the optimisation problem (2.1).

Moreover, this method could be applied to existing cohorts, such as TCGA[97] and potentially `convSig` could discover more mutation signatures or add attributes to the known 3-bp signatures detected by classic `NMF` methods.

However, there are also certain limitations of `convSig` which could be area for future improvement.

First of all, the exponential and ReLU transformation that keeps the mutation signature positive may not be the best choices. These two functions are chosen mainly because of computational concern. It is easier to develop optimisation algorithms with these two transformation functions. However, these two functions may not be the best functions for biological concern. In other words, there might be mutation signatures that do not follow the rule of either exponential or ReLU transformation in nature. Therefore, for future research, more transformation functions that keeps the mutation signature positive could be explored, e.g. the `softplus` function which gained its popularity in deep learning in recent years[98]. With more powerful computational resources, the mix of different transformation functions could also be studied. For example, one could combine exponential and ReLU transformation in a single study, so that some signatures may follow the rule of exponential transformation while others could follow ReLU transformation.

Secondly, for deep version of `convSig`, the way that it is used to extract the mutation signature could also be improved. For example, in computer vision, a similar approach based on back-propagate gradients called `TCAV` could be used here as well[99]. The biggest advantage of using `TCAV` than using simple back-propagate gradients is that it offers a hypothesis-testing framework so that it is more robust than other approaches. However, for `TCAV`, it requires much more computational time which makes this task challenging.

Finally, for the mutation signatures beyond trinucleotide context, it requires experimental approaches to verify the results found by computational approaches. For

trinucleotide mutation signatures, it has been verified by multiple studies[9, 84, 100]. However, for signatures with longer context, it is going to be much harder. For the discovery of the potential signatures found by the deep model, it will be interesting if such CG rich region is potentially related to some molecular process on the genome.

Overall, I believe the approach here to provide an exciting new approach to genome sequence analysis, with the potential for future development in the directions mentioned above.

2.8 Appendix

Proof for Theorem 2.1. First, augment the model by introducing an auxiliary variable $z_k^{n,y}$ which is the conditional probability that the ith mutation process results in the mutation of nucleotide sequence type j to y. This results in a modified minimization problem as follows.

$$\begin{aligned} \min \quad & -\sum_n \left[\sum_y X_{n,y} \sum_k z_k^{n,y} \log \frac{p_k \theta_{nk} M_{x,y}^k}{z_k^{n,y}} - w_n \sum_k n p_k \theta_{nk} \right] \\ \text{s.t.} \quad & \sum_k z_k^{n,y} = 1, z_k^{n,y} \geq 0 \\ & \theta_{nk} = \exp(T_n * F^k) \end{aligned}$$

If a coordinate descent is carried out for $p_k, z_k^{n,y}$ and $M_{x,y}^k$, by taking derivatives of the Lagrange dual with respect to each term and set to zero, the relevant M-update steps could be derived respectively for Theorem 2.1.

For the M-update for the filter weights, F^k, they are optimised on an element-by-element basis, $F_{l,b}^k$. By taking the derivative of the Lagrange dual to zero, it is possible to obtain

$$\sum_{n:\mathrm{l}(n)=b} \left[\sum_y X_{n,y} z_k^{n,y} - (w_n p_k + \mu_k) \theta_{nk} \right] = 0 \tag{2.7}$$

where μ_k is the Lagrange multiplier.

Then, I defined two variables as follow.

$$\eta_{l,b}^{k} = \frac{\exp(F_{l,b}^{k})}{\sum_{b} \exp(F_{l,b}^{k})}$$
$$C_k = \prod_{l} \sum_{b} \exp(F_{l,b}^{k})$$

It could then be shown that:

$$\theta_{nk} = C_k \prod_{l} \eta_{l,b}^{k} \tag{2.8}$$

By combining equation (2.8) and (2.7), it is possible to get the following equation.

$$\eta_{l,b}^{k} = \frac{\sum_{n:\mathrm{l}(n)=b} \sum_{y} X_{n,y} z_{k}^{n,y}}{C_k \left[\mu_k + (p_k \sum_{n:\mathrm{l}(n)=b} w_n \prod_{l' \neq l} \eta_{l',b'}^{k})\right]}$$

Since the optimisation with respect to $\eta_{l,b}^{k}$ is a convex problem where Slater's condition holds, there will be a unique solution for which KKT condition holds. Furthermore, as $\sum_{n} \theta_{nk} = C_k = 1$. Therefore, the update rule for $\eta_{l,b}^{k}$ could be simplified into the following.

$$\eta_{l,b}^{k} = \frac{\sum_{n:\mathrm{l}(n)=b} \sum_{y} X_{n,y} z_{k}^{n,y}}{\mu_k + (p_k \sum_{n:\mathrm{l}(n)=b} w_n \prod_{l' \neq l} \eta_{l',b'}^{k})}$$

Since every $\eta_{l,b}^{k}$ is monotonically decreasing with respect to μ_k, there is a unique solution which guarantees $\sum_{b} \eta_{l,b}^{k} = 1$.

In most cases, w_n is non-zero, and the optimization is bounded below. After the introduction of auxiliary variables, the new problem is also bounded below. And if p_k and $\eta_{l,b}^{k}$ is updated iteratively by some root finding algorithm, it will finally converge, for each update will only decrease the loss. When it converges p_k and $\eta_{l,b}^{k}$ will arrive at some fixed point. And at such time, the following condition could be verified.

$$\begin{aligned}
&\sum_n [\sum_y X_{n,y} z_k^{n,y} - (w_n p_k + \mu_k)\theta_{nk}] \\
=&\sum_n \sum_y X_{n,y} z_k^{n,y} - (p_k \sum_n w_n \theta_{nk}) - \mu_k \\
=&0
\end{aligned}$$

This indicates $\mu_k = 0$ at the fixed point. Therefore, the update for (2.8) could finally be simplified into the following step.

$$\begin{aligned}
\eta_{l,b}^k =& \frac{\sum_{n:\mathrm{l}(n)=b} \sum_y X_{n,y} z_k^{n,y}}{p_k \sum_{n:\mathrm{l}(n)=b} w_n \prod_{l' \neq l} \eta_{l',b}^k} \\
\propto& \frac{P_{l,b}^k}{W_{l,b}^k}
\end{aligned}$$

By normalizing $\eta_{l,b}^k$, the third update in Theorem 2.1 is derived and the fourth update still holds at the fixed point.

□

Proof for Theorem 2.2. For a particular mutational process, the original target function can be simplified as follow.

$$\begin{aligned}
\min \quad & \sum_n \left[\sum_y -X_{n,y} \log(\alpha_{n,y}^k + \theta_{nk}) + \omega_{nk}\theta_{nk} \right] + \frac{1}{2}\lambda \sum_{nk} (\theta_{nk} - \gamma_{nk})^2 \\
\text{s.t.} \quad & \sum_n \theta_{nk} = 1, \theta_{nk} \geq 0
\end{aligned}$$

If Lagrange dual of this optimization problem and KKT conditions is considered, it is possible to get $\theta_{nk} = 0$ or $\mu_{nk} = 0$, and also the following equation.

$$\sum_y \frac{X_{n,y}}{\alpha_{n,y}^k + \theta_{nk}} - \lambda\theta_{nk} + \lambda\gamma_{nk} - \omega_{nk} + \mu_{nk} = v$$

(v is a dual variable associated with constraint $\sum_n \theta_{nk} = 1$ and μ_{nk} are dual variables associated with $\theta_{nk} \geq 0$.)

Thus, if $\theta_{nk} \neq 0$, then $\mu_{nk} = 0$, and $g(\theta_{nk}) = \sum_y \frac{X_{n,y}}{\alpha^k_{n,y}+\theta_{nk}} - \lambda\theta_{nk} + \lambda\gamma_{nk} - \omega_{nk} = v$.

Because $g(x)$ is monotonically decreasing to negative infinity with respect to θ_{nk}, there is a unique solution to both v and θ_{nk}. Thus, it is possible to use some root finding algorithms to find the update for θ_{nk}.

□

Chapter 3

A reinforcement learning approach for learning copy number evolution in cancer

3.1 Copy number evolution in cancer

In humans, each cell contains 23 pairs of chromosomes, for a total of 46 which contains the DNA that encodes its genome. Twenty-two of these pairs (chr1-22), called autosomes, reside in both males and females but the final pair, the sex chromosomes (chrX/Y), differ between males (with one X and one Y) and females (with two X chromosomes). Normal cell division results in a parent cell giving rise to an exact clone of itself with the same DNA sequence and copy number. The parent cell first duplicates its entire complement of chromosomes, before the process of chromosome segregation(Fig.3.1a).

In cancer cells, mutations as well as exposure to exogenous mutagens can give rise to a loss of effective control of the DNA replication, DNA repair and cellular division process[17, 62]. This can give rise to chromosome segregation errors that cause cells to acquire abnormal numbers of chromosomes or the partial loss or gain of chromosomal segments, a phenomenon referred to as copy number alterations (Fig.3.1b)[61]. Therefore, whilst in a normal human cell, it is expected to find one paternally and one maternally derived copy of most regions of the genome, the copy number profile (CNP) of a cancer often deviate from this norm. An ongoing and persistent high

frequency of these errors (chromosomal instability (CIN)) is predicted to profoundly impact tumor evolution and therapy response[101, 102]. The way how CNA could be detected has been discussed previously in the introduction chapter and I will focus on other aspects of copy number evolution here.

A special event arises when a cell receives double the normal genome content - this might arise due to a failure to separate into two daughter cells. This process, formally known as *tetraploidization*, is often referred to as whole genome doubling (WGD) in more recent studies [103]. Whether a tumour sample contains cells with WGD or not can be roughly estimated based on its average copy number[104], but some copy number callers can be used to estimate if there is WGD based on their probabilistic model as well[67].

Among all cancer types, WGD occurs on average in 30% of the samples. However, this varies substantially. In cancers like germ cell tumor it may be as high as 60% and in cancers like malignant gastrointestinal neuroectodermal tumor it can be as low as 5%[104, 105].

Cancers in which cells have undergone WGD are often more aggressive and difficult to treat [18]. It is suspected that such properties arise because the cells carry abnormalities which are permissive for WGD and cells with WGD may better withstand insults because of more copies of genes involved in response. However, it is still not clear how (1) the impact of WGD on prognosis is affected by the therapy given to the patients previously, (2) WGD leads to better or worse[106] response to different therapies, including systemic, targeted, and immuno-therapies, and (3) whether unique therapeutic vulnerabilities may be caused by WGD, and whether it is due to the WGD event itself or it is due to the subsequent CNAs introduced by WGD[105]. Consequently understanding if and when WGD occurs is of importance for prognosis.

The estimation of the timing of CNA events, including WGD, poses a particular challenge[107]. In previous studies, it has typically been estimated by analysis of loss of heterogeneity(LOH), under the assumption that a loss will result in LOH if it happens before WGD[18]. If point mutation data is available, it is also possible to time the WGD event by using the variant allele frequency computed for point muta-

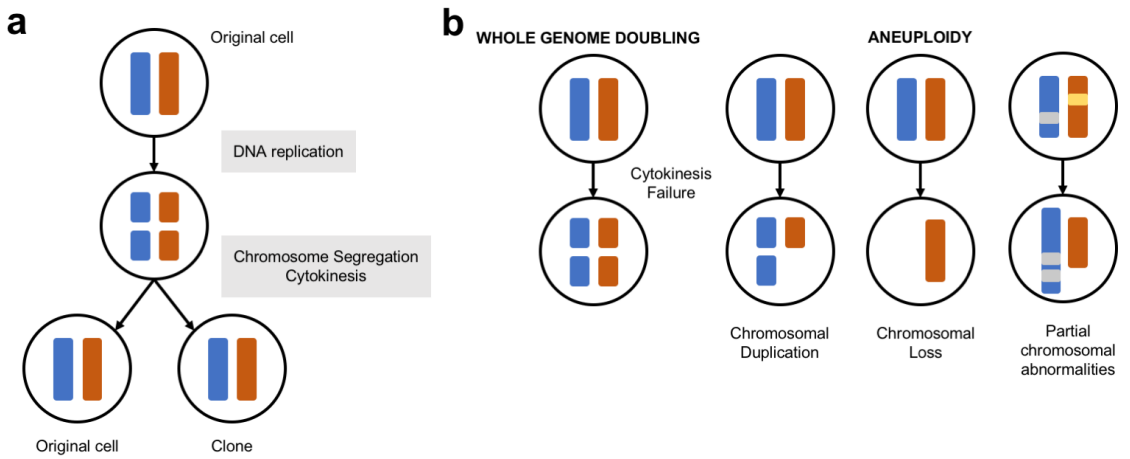


Figure 3.1.* Cell division.** ***(a) In normal cell division, a cell synthesises an exact copy of its DNA and shares it amongst two daughter cells. (b) Faulty cell divisions, DNA replication or chromosomal segregation can lead to abnormal cellular chromosomal content (aneuploidy). Paternally and maternally-derived chromosomes are shaded blue and red respectively.

tions where mutations with higher allele frequencies are considered to happen early in the history of cancer evolution[108]. A computational approach called `WCND`[109] based on a linear programming model is also proposed to estimate the timing of CNA events as well, which will be discussed in further details.

In this chapter, I have developed a novel machine learning method called `RLEvolution` to retrieve the evolution history of copy number alterations in cancer data using only copy number data. This method is based on the reinforcement learning algorithm[71] and it models the cancer evolution process in a more complex way than `WCND`.

3.2 Motivation

As mentioned in introduction chapter, modern DNA sequencing technologies, in combination with appropriate copy number analysis models[66, 67, 70], allow the number of copies of most genomic region to be estimated [110], except regions such as repetitive or poorly mapped regions. However, the sequencing-derived read out gives the current copy number state but does not directly indicate the sequence of events that lead to the observation. Fig.3.2 illustrates two distinct evolutionary paths that involve sequences of copy number alteration events. The first sees a whole paternal chromosome duplication leading to extra copies of the regions A, B and C.

Subsequently, a sub-chromosomal loss occurs leading to the loss of region B. The second evolutionary trajectory sees a sub-chromosomal gain of region A, followed by a gain of C. Thus, it is possible for there to be more than one evolutionary path leading to the same final copy number profile as illustrated here.

The objective of this work is to use collections of sequencing-derived cancer copy number profiles to infer the sequence of copy number altering events that occurred to give rise to those perturbed cancer genomes. While there are many algorithms for estimating copy number profiles in heterogeneous and single cell cancer genomes from raw sequencing data [67, 111–113], there have been limited number of working attempts[109, 114] to directly infer the sequence of evolutionary events that have led to the observed CNP. Other studies have instead relied on heuristics, for example, by using the rule that tumours were considered to have undergone WGD if greater than 50% of their genome had a major copy number (the more frequent allele in a given segment) greater than or equal to two[115].

The contribution of this work in this chapter is to introduce a fully algorithmic approach based on *deep reinforcement learning*[71], which will be described in detail below, to address this need in evolutionary cancer modelling. I cast the problem as a genomic state evolution in time governed by a Markov decision process that is solved using reinforcement learning. I show that this novel method can infer evolutionary trajectories from CNPs more accurately than heuristic and existing approaches[109] using simulated data, and provide a quantitative basis behind qualitative evolutionary features previously identified from real cancer data sets (Fig.3.4A).

3.2.1 Related work

The most similar prior method addressing the problem of timing different CNAs is `WCND` (weighted copy number distance)[109], which is based on the previous work of un-weighted copy number transformation[114]. In the un-weighted copy number transformation, the copy number evolution process is modelled as two phases. In the first phase, all CNAs are losses and in the second phase, all CNAs are gains. Then, the goal of timing CNAs becomes identifying the CNAs belonging to the first phases and the CNAs belonging to the second phase. The reason for this two phase

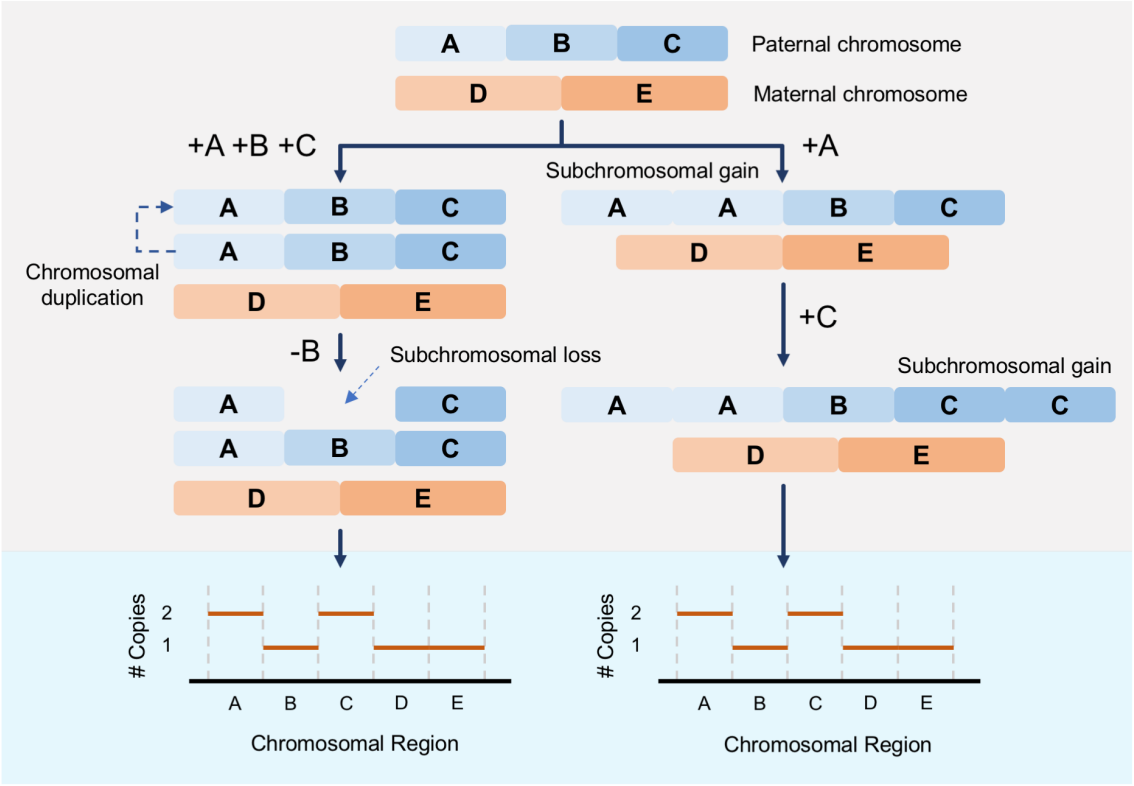


Figure 3.2.* Copy number evolution.** ***Illustrating how two different sequences of copy number alterations events can lead to the same copy number profile. The actual evolution history is not observed, only the read out of the number of copies of each genomic region.

modelling is because of a mathematical concern. It is shown that this problem could be converted to a linear programming optimisation problem and thus solved by a standard convex optimisation solver by using this two phase model[114].

`WCND` is an improved version of the model mentioned above. First, it gives different weights to chromosome CNAs and focal CNAs in the linear programming problem. Secondly, it increases the number of phases to 3, so that it has a phase containing only losses, a phase containing only gains and a final phase again containing only losses. These two modifications make the model more realistic[109].

The limitation of these methods is obvious. The simplification of copy number evolution process to only 2 or 3 phases is crude and does not have biological support for doing so. Besides, whole genome duplication event is not considered in both methods. These limitations motivated development of a new method based on reinforcement learning to circumvent these limitations.

3.3 Reinforcement learning

3.3.1 Introduction

Reinforcement learning is a branch of machine learning where the goal is to train an algorithm to take actions in order to get maximum reward under a specific environment[116]. Reinforcement learning became popular first through the application of AlphaGO[117], where the computer was taught to play a game GO. Imagine a situation where a human beginner learns to play GO. There are two alternatives: (1) follow a human expert in the field, and replicate his/her decision of which move to play. This is also called imitation learning[118]. (2) find a teacher in the field, play the game yourself, and ask the teacher to evaluate if you are making the correct decisions or not, based on his/her experiences of watching others play. If not, ask the teacher to tell you how you might improve yourself.

To elaborate, in the first method, the expert is called a "policy network", and in the second method, the teacher is called a "value network" in reinforcement learning. The policy network is responsible for playing the games based on what it believes to be the best policy/strategy. The value network is responsible for evaluating/judging how good each move is, which could be the change of the probability that the player is going to win at the end after making this move. Both value network and policy network are implemented as a deep neural network with parameters to be learnt.

However, in order to train a computer to play GO, neither an 'expert' nor a 'teacher' is available before training. To address this issue, the policy network and value network with parameters are randomly initialised. During training, the parameters of policy network are first fixed unchanged, and the parameters of value network are updated so that it learns to predict how likely the player is going to win by following the current strategy of the policy network. Then, the parameters of the value network are fixed unchanged, and the parameters of the policy network are updated to make better movements under the guidance of the value network. These two schemes are repeated for multiple times, and hopefully when it converges, both policy network and value network become optimal. This algorithm is called actor-critic algorithm[119] and has been successfully applied to AlphaGO. However, in this

algorithm, two deep neural networks have to be trained which is computationally intensive.

In another algorithm of reinforcement learning called Q-learning[120, 121], which is the algorithm going to be used in this chapter, the policy network is defined to be always choosing the best action identified by the value network. Namely, the player traverse all possible actions that could be taken at the current stage, and takes the one that the value network considers to be the best action. Therefore, it is not necessary for the computer to learn the parameters for the policy network. The training for the value network is simplified to compute the probability that the computer is going to win under this policy.

In the case of copy number evolution, the policy network will identify the latest CNA in the evolution while the value network will evaluate through a probability model how likely this choice by the policy network is correct.

3.3.2 Formal explanation

Reinforcement learning is also known as dynamic programming and is traditionally used in control systems to develop robots that can accomplish specific tasks[116]. Modern deep neural nets have brought greater power to reinforcement learning as they could approximate much more complex functions[28].

Reinforcement learning is defined on a Markov decision process(Fig.3.3). In a Markov decision process, there are two important elements: state s and action a. A state $s \in \mathcal{S}$ is an element that contains all information about the current situation so that the future state will be independent from previous states given the current state. For example, in the study of copy number evolution, the state is defined as the CNP. An action $a \in \mathcal{A}$ is an element that could change the current state. For example, in copy number evolution, an action is defined by an CNA. Here $\mathcal{S}$ is called the state space and $\mathcal{A}$ is called the action space. Based on state and action, there are two functions defined: a reward function $r(s, a)$ which measures how preferable this action a is under state s and a transition probability function $p(s_{t+1}|s_t, a_t)$ which is the probability that the state at time $t + 1$ will be s_{s+1} given state s_t and action a_t at time t. In the case of cancer evolution, the transition probability function is

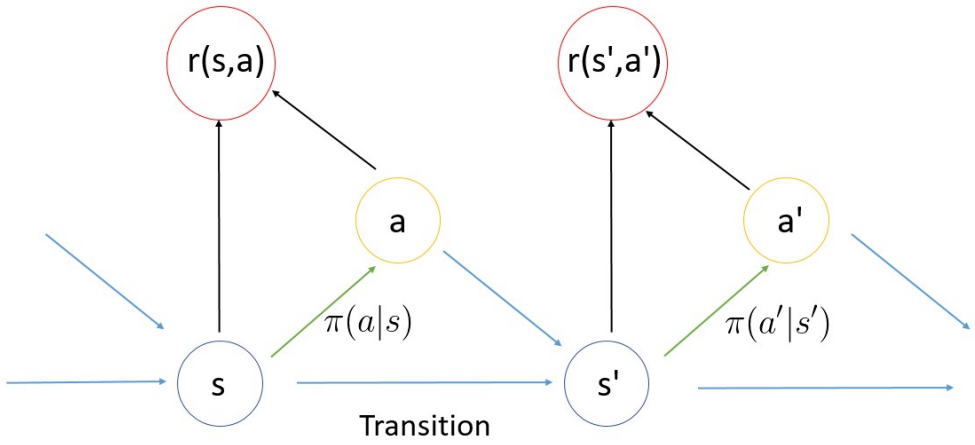


Figure 3.3.* Markov Decision Process.** ***At the current state s***, an action*** a ***is taken according to some strategy*** $\pi(a|s)$***. The reward*** $r(s,a)$ ***is thus determined by the current state*** s ***and actions taken*** a***. After the actions taken, the state changed to a new state*** s' ***and the decision process continues as the previous state.***

not needed as the transition in cancer evolution is mathematically a deterministic process. In other words, by knowing the current state, i.e. the CNP and the last action, i.e. the CNA, it is possible to work out the CNP before this last CNA directly. For example, if this last CNA is a chromosome gain for chromosome 9, the CNP before this CNA could be found by simply subtracting one copy for all loci on chromosome 9 for the current CNP. This transition probability function is needed for other applications, e.g. autonomous driving, to take into the probability that due to mechanical error, the car doesn't follow the action taken by the driver[122]. Finally, a policy $\pi(a|s)$ is defined to be the probability of choosing action a given state s.

The goal is to find an optimal policy $\pi(a|s)$, i.e. a way to choose a specific action depending on the state, which will finally give the maximum reward, i.e. $\sum_t r(s_t, a_t)$[116].

The mathematical formulation of reinforcement learning problem goes as follows, where $\pi_\phi(a|s)$ is a parametric policy function to be learned. This $\pi_\phi(a|s)$ is indeed the policy network mentioned above.

$$\max_{\phi} \quad E_{a_t \sim \pi_\phi(a_t|s_t)} \sum_{t=0}^{T} r(s_t, a_t)$$

This target function could be simplified to give the following Bellman equation (3.1). The function $V_\phi(s)$ is then the value network mentioned above.

$$\begin{aligned} &\max_{\phi} \quad E_{a_0 \sim \pi_\phi(a_0|s_0)} \left[r(s_0, a_0) + E_{a_i \sim \pi_\phi(a_i|s_i)} \sum_{i=1}^{T} r(s_i, a_i) \right] \\ &V_\phi(s_0) \coloneqq E_{a_0 \sim \pi_\phi(a_0|s_0)} \left[r(s_0, a_0) + E_{a_i \sim \pi_\phi(a_i|s_i)} \sum_{i=1}^{T} r(s_i, a_i) \right] \\ \implies &V_\phi(s_0) = E_{a_0 \sim \pi_\phi(a_0|s_0)} \left[r(s_0, a_0) + V_\phi(s_1) \right] \end{aligned} \tag{3.1}$$

There are multiple ways to solve the reinforcement problem. As in many applications, a human expert is provided and the model is trained to copy the policy of human expert, i.e. imitation learning as mentioned above. Though this gives successful results in some cases[123], the need for a human expert necessarily limits the size for training samples.

Another algorithm called actor-critic has also been widely used in the field of reinforcement learning[117, 124, 125]. In actor-critic algorithm, the value network will evaluate each state and output the expectation for rewards after the state, the so-called reward-to-go, depending on the current policy of the model. Then, the policy network will decide, based on the output of the value network, which action to take as to maximize the reward, and then improve the policy to take. Though, this type of method works in many cases, to train two deep neural network i.e. both $\pi_\phi(a|s)$ and $V_\phi(s)$, pose a big challenge.

In Q-learning[120], the training of policy network is omitted, as one could choose the action by traversing all actions possible to take, and choose the best one determined by the value network. The Bellman equation (3.1) can then be converted to the following.

$$\begin{aligned} \pi_\phi(a_t|s_t) = 1 \quad \text{if} \quad a_t = \arg\max Q_\phi(s_t, a_t) \\ Q_\phi(s_t, a_t) \coloneqq r(s_t, a_t) + V_\phi(s_{t+1}) \\ \implies Q_\phi(s_t, a_t) = r(s_t, a_t) + max_{a_{t+1}} Q_\phi(s_{t+1}, a_{t+1}) \end{aligned} \tag{3.2}$$

Here, a Q-function is introduced and it evaluates the expected reward to get after the current state s_t by taking action a_t, which is called reward-to-go. And then, the policy is simply defined to always choose the action a_t that maximized the reward-to-go determined by the Q-function. In Q-learning, the objective is to keep the new version of Bellman equation (3.2) hold true, by solving the following optimisation function.

$$\min_\phi \quad E_{s_t,a_t}[Q_\phi(s_t, a_t) - r(s_t, a_t) - max_{a_{t+1}} Q_\phi(s_{t+1}, a_{t+1})]^2$$

3.3.3 Applications

Reinforcement learning has been widely used in different areas. For example, in autonomous driving, people are using reinforcement learning algorithms to train agents to drive on its own[122–124]. Also, in many computer games, agents trained by reinforcement learning are beating humans these days[126, 127]. There are also attempts trying to apply reinforcement learning to biomedical research field. For example, there is an approach which tries to apply reinforcement learning to fragment assembly for sequencing data[128]. In this work, fragment assembly is posed as a complex optimisation problem and Q-learning algorithm mentioned above is used to solve it. However, most of these applications lie in the field where there is a clear goal and a set of defined rules, either by physical laws or by human definitions, about how it behaves. For cancer evolution, the most difficult part is that there is neither of these.

In this application of reinforcement learning to cancer genome evolution, I have defined the goal of the learning algorithm through a probabilistic model, so that there is a clear objective for the algorithm.

3.4 Model

3.4.1 Definitions

A *copy number profile* (CNP) is a sequence s_0 of N non-negative integer indicating the number of copies for each position on the genome across different chromosomes. In the method I have developed, 22 pairs of autosomes and 50 loci for each chromosome are considered. This choice of 50 loci per chromosome is mainly out of the concern for computational efficiency. In total, there are $44\ chromosomes \times 50 = 2200$ loci on the genome considered, and thus CNP is a vector of size 2200, where each integer in the vector indicate the number of copies for the corresponding locus on the genome. The CNP is the result of an unknown number T sequential copy number alteration (CNA) events, $A = \{a_T, \ldots, a_1\}$, that transforms a normal diploid genomic state s_T to s_0. I define a trajectory τ_T as the collection of genomic states $\{s_T, \ldots, s_1, s_0\}$ (or equivalently $\{a_T, \ldots, a_1, s_0\}$ since s_t is defined by the action a_t taken on the previous state s_{t-1}) and therefore the probability of a trajectory, $P(\tau_T) = p(s_T)\prod_{t=1}^{T} p(a_{T-t+1})$. Here, for a particular cancer CNP S, I am interested in the probability of a particular trajectory τ_T among all trajectories τ'_T that transform s_T into $s_0 = S$ via some A', namely $P(\tau_T|S) \coloneqq P(\tau_T|\tau_T \in \{\tau'_T : s_0 = S\}) = \frac{P(\tau_T)}{\sum_{\tau'_T:s_0=S} P(\tau'_T)}$, if τ_T transforms s_T into $s_0 = S$, otherwise it is 0.

I therefore approach this from a *backwards-in-time* perspective, where I seek to find a trajectory from the observed CNP back to the original normal, unaltered genomic state.

I am interested in viewing this as a *Markov decision process* (MDP) where I wish to learn a *policy* $\pi(a|s)$ that defines the probability distribution over CNA events (actions) given a particular genomic state. This means that, for a given CNP s, $\pi(a|s)$ represents the probability of a CNA a happening as the latest event during evolution after s_1, where a transforms s_1 to $s_0 = s$. This process is shown in Fig 3.4B. However, since it is not obvious how to get $\pi(a|s)$ from the previously defined $P(\tau_T)$, I solve this problem through *reinforcement learning* [71].

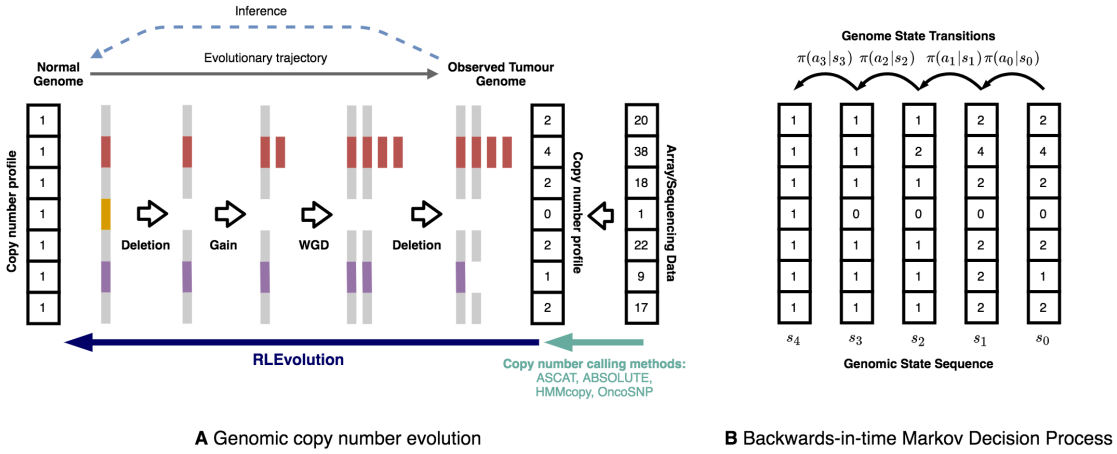


Figure 3.4. **Modelling copy number evolution.** ***A Cartoon illustration of a sequence of genomic alteration events associated with the transformation of a normal genome into a cancer genome. Copy number calling methods estimate copy number profiles from array- or sequencing-based data. My contribution is to develop a model*** `RLEvolution` ***that uses the copy number profiles to infer the evolutionary sequence. B Modelling the evolution of the genomic state sequence as a discrete Markov decision process requires and find the optimal policy*** $\pi(a|s)$***.***

3.4.2 Reinforcement learning approach

The goal is to find an optimal trajectory τ_T^*, with the highest probability $\tau_T^* = \arg\max_{\tau_T} P(\tau_T|s_0)$ for a given CNP. This will require identifying an optimal policy $\pi^*(a|s)$ for which I will use a Q-learning approach [120].

To simplify the math notations, I first introduce a special action, denoted by END, which indicates the end of the MDP (namely, the start of the cancer evolution process). Thus, it is possible to write the probability in the following form as $P(\tau_T) = \prod_{t=0}^{T} q(a_{T-t+1}, s_{T-t})$, where $a_{T+1} = \text{END}$, $s_{T+1} = s_T$ and $q(\text{END}, s_T) = p(s_T)$, $q(a_{T-t+1}, s_{T-t}) = p(a_{T-t+1})$. Therefore the action a_T is always to convert state s_{T-1} into the normal genomic state s_T. Notice that the definition of $P(\tau_T|S)$ is not changed.

3.4.2.1 Q-learning

The probability of each trajectory sampled according to this policy could be defined as $\Pi(\tau_T|s_0) = \prod_{t=0}^{T} \pi(a_{t+1}|s_t)$ and $a_{T+1} = \text{END}$. Therefore, if $\pi(a|s)$ for every action-state pair (a, s) is known, it will be possible to calculate $\Pi(\tau_T|s_0)$ in linear time with respect to the length T of the trajectory τ_T.

It is aimed to find a policy which leads to a distribution over trajectories $\Pi(\tau_T|S)$ that is close to the true distribution of evolution trajectory $P(\tau_T|S)$. However, as I will outline later, the action-state space in this setting is extremely large, and I will try to find a parametric model to approximate it. I define a parametric model $\pi_\phi(a|s)$ with parameters ϕ that outputs the probability for each CNA a, and the conditional probability for each trajectory as $\Pi_\phi(\tau_T|s) = \prod_{t=0}^{T} \pi_\phi(a_t|s_t)$. The objective is therefore to minimise the Kullback-Leibler divergence between the approximating and true distributions through the parameters ϕ: (Notice that here it is not difficult to change $P(\tau_T|s)$ to $P(\tau_T)$ without changing the optimization problem, as long as τ_T is a trajectory transforming a normal genome s_T into s.)

$$\min_{\phi} -\mathrm{E}_s \left[\sum_{\tau_T} \Pi_\phi(\tau_T|s) \log \frac{P(\tau_T|s)}{\Pi_\phi(\tau_T|s)} \right] \tag{3.3}$$

This problem can be treated as a reinforcement learning problem by defining the reward to be the log-likelihood of each CNA, i.e. $r(s_t, a_t) = \log q_(a_t, s_t)$ and solve through a Q-learning algorithm according to the following theorem.

Theorem 3.1. *If the gradient of* $\log(\pi_\phi(a|s))$ *is bounded, the following optimization problem:*

$$\min_{\phi} \quad \mathbb{E}_s \mathbb{E}_a \left[Q_\phi(s,a) - r(s,a) - \mathrm{softmax}(Q_\phi(s',a')) \right]^2 \tag{3.4}$$

where

$$\begin{aligned} Q_\phi(s,a) &= C_\phi(s) + \log \pi_\phi(a|s) \\ C_\phi(s) &= -KL(\pi_\phi(\tau_s|s), p(\tau_s|s)) \\ \mathrm{softmax}(Q_\phi(s',a')) &= \log \left(\sum_{a'} \exp^{Q_\phi(s',a')} \right) \qquad (3.5) \\ r(s_t,a_t) &= \log q(a_t,s_t) \qquad (3.6) \end{aligned}$$

is equivalent to solving Eq. 3.3 in the sense that solving this optimisation problem is the same as finding the critical point for Eq. 3.3, when the action is not the special action, i.e. $a \neq END$. *Here* $KL(p,q)$ *is the Kullback-Leibler divergence between* p

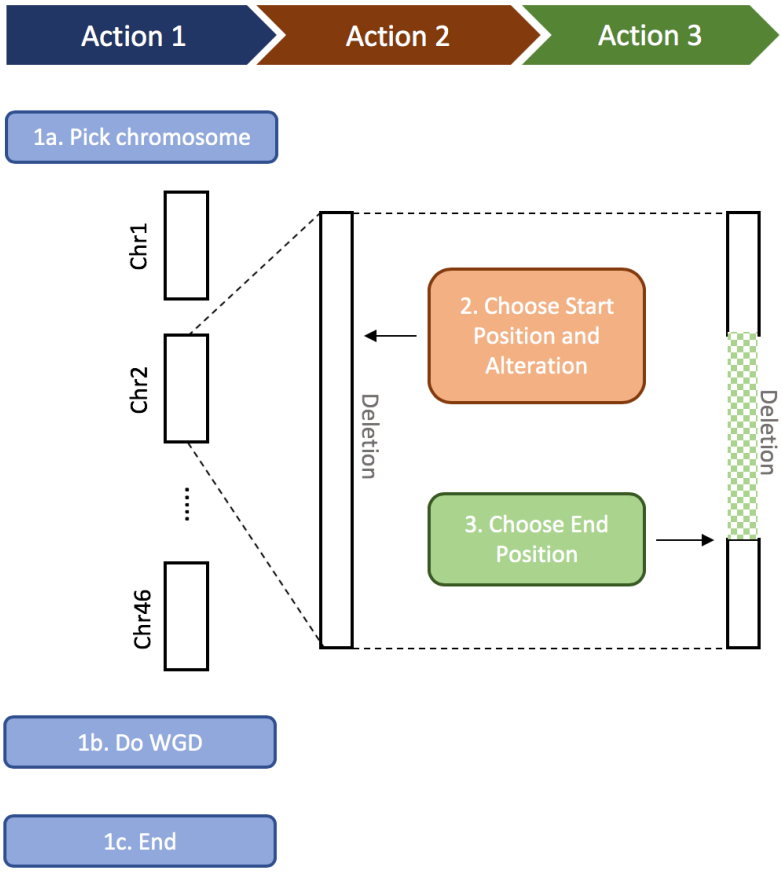


Figure 3.5.* Action space.** ***The action space is partitioned into three hierarchical sub-components, the first sub-action determines if a single chromosome should be chosen for alteration, a whole-genome doubling should occur or the evolutionary trajectory should end. If a single chromosome is chosen, the method must then decide what type of alteration to induce and where the copy number alteration begins. Finally, the end point of the alteration must be chosen. I use three convolutional neural networks to model the probability distribution over actions.

and q.

The full proof for the theorem is attached in the appendix section.

3.4.2.2 Action space

The action space includes all possible copy number alteration events for a chosen chromosome, whole genome duplication, and the special action END indicating the end of the MDP (Fig.3.5). For the experiments, I divide each chromosome uniformly into 50 sub-regions. Therefore, there are 1,225 ($= 50 \times 49/2$) possible CNA events with distinct starting and endpoints. Ignoring the sex chromosomes, there are 44 ($= 22 \times 2$) chromosomes and each CNA event could involve a deletion or gain of a

chromosomal segment. Hence, in total, the size of the action space is on the order of $44 \times 1,225 \times 2 = 107,800$ (even larger if finer resolution of CNA events is desired).

3.4.2.3 Training samples

The loss function requires expectations over copy number profiles s and actions a to compute equation (3.4). In Theorem 3.1, no specific distribution is required for the expectation, as all distributions provide a tight upper bound for the true target (at global minimum 0). Thus it is possible to use all possible distributions over these CNP as long as the distribution has a broad support. This is critical since known distributions for these are unavailable but this insight means it is still possible to train the model. I sampled from a probability distribution over CNPs with a hierarchical construction such that the number of CNAs needed to form the CNP, i.e. $p(T)$, has a geometric distribution. The decay rate $= 0.98$ is used, so on average, it is expected that CNPs will contain $\frac{1}{1-0.98} = 50$ number of CNAs. Therefore, the probability of sampling CNPs with differing number of CNAs is proportional to 0.98^T. Then, given T, it is possible to sample CNAs from the action space as outlined previously. Thus, the total number of CNPs is $50 \times 107,800 \approx 5,000,000$.

3.4.2.4 Reward function

The reward assigned to each CNA event was computed according to findings from genomic landscape studies of copy number alterations in cancer [20]. Specifically, it has been found that the probability of interstitial CNAs decay with their length, while whole chromosome and chromosome arm changes are distinct event types with their own probabilities. Here, I used summary statistics from these studies to set the probability of interstitial CNAs $\log q(a_t, s_t) \propto \frac{1}{L}$ in Equation (3.6)), but the probability for whole chromosome or chromosome arm CNAs to fixed values that are $\frac{1}{4}$ and $\frac{1}{3}$ of the largest probability for the interstitial CNAs. It is of course also possible to optimise these choices as part of the learning process, however, the marginal statistics provided by the literature [20] made this empirically unnecessary.

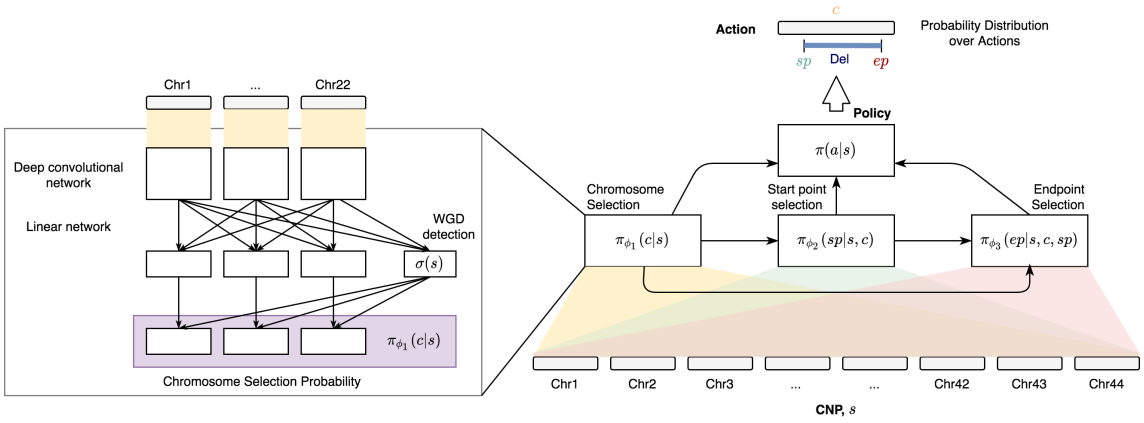


Figure 3.6. **A deep convolutional neural network model for computation of $\pi_\phi(a|s)$. *Because of the large search space for actions, the parametric model $\pi_\phi(a|s)$ is separated into three parts: $\pi_{\phi_1}(c|s)$, $\pi_{\phi_2}(sp|s,c)$ and $\pi_{\phi_3}(ep|s,sp,c)$. The $\pi_{\phi_1}(c|s)$ is built with the permutation invariant structure and with the switch to distinguish the cases with WGD and the case without WGD. Both $\pi_{\phi_2}(sp|s,c)$ and $\pi_{\phi_3}(ep|s,sp,c)$ will be based on the results of the other neural nets. Finally, the output of these three neural nets combined with the rule system, produces the conditional probability that's needed $\pi_\phi(a|s)$.***

3.4.3 Implementation

The reinforcement learning algorithm developed is called `RLevolution` here. The following outlines specific key features of the system.

3.4.3.1 Partitioning the action space

I partition the policy model into three sub-components and model each with separate deep convolutional neural networks (Fig.3.5). The first network selects whether the action will affect a particular chromosome, cause a WGD or trigger the end of the evolutionary process. If a chromosome is chosen, the second network outputs the type of CNA and the starting point for that CNA along the chromosome. Finally, the third network outputs the endpoint for the CNA.

An action on a chromosome is denoted by $a = (c, sp, g, ep)$, where $c \in \{1 - -44, a_0, a_{WGD}\}$ determines whether the action affects one of the 44 chromosomes, terminates the MDP or triggers a WGD, $sp \in \{1, \ldots, n\}, ep \in \{\min(i+1, n), \ldots, n\}$ denote the start and end point of the alteration and $g \in \{-1, +1\}$ is the alteration type respectively. I have by design a hierarchical construction,where ϕ_1, ϕ_2, ϕ_3 are subsets of the trainable parameter ϕ. $\pi_\phi(a|s) = \pi_{\phi_1}(c|s)\pi_{\phi_2}(sp, g|s, c)\pi_{\phi_3}(ep|s, g, sp, c)$ and $\sum \pi_{\phi_2}(sp, g|s, c) = 1$, $\sum \pi_{\phi_3}(ep|s, c, sp, g) = 1$ for actions other than the spe-

cial actions a_0 and a_{WGD}. Then, for the Q-function of these actions, there is the following relationship:

$$\begin{aligned} Q_\phi(s,a) = & Q_{\phi_1}(s,c) \\ & + Q_{\phi_2}(s,c,sp,g) - \text{softmax}_{sp,g}(Q_{\phi_2}(s,c,sp,g)) \\ & + Q_{\phi_3}(s,sp,g,c,ep) - \text{softmax}_{ep}(Q_{\phi_3}(s,sp,g,c,ep)) \end{aligned} \tag{3.7}$$

where the softmax terms are included to enforce the sum-to-one constraints.

This structure makes it easier to compute the softmax for $Q(s,a)$, because of the fact that $\text{softmax}_a Q(s,a)$, which equals $\log\left(\sum_c \exp^{Q_\phi(s,c)} + \exp^{Q_\phi(s,a_0)} + \exp^{Q_\phi(s,a_{WGD})}\right)$. Thus, it is not necessary to traverse all possible actions a in order to calculate such softmax of Q-function.

This structure is illustrated in Fig.3.6.

3.4.3.2 Real-world physical constraints

From any genomic state, there maybe some CNAs or actions which are physically impossible to take. For example, in this backwards-in-time model, a genomic region which has an odd copy number cannot have arisen directly from a WGD previously since the latter would always result in even copy numbers after completion. In order to prevent these actions a from occurring, rather than setting a large penalty (negative reward) for them, I associate a rule system for the policy. This is equivalent, statistically speaking, to placing a strong prior distribution on the policy, $p'_\phi(a|s) \propto p_{RL}(a|s)\pi_\phi(a|s)$ where $p_{RL}(a|s)$ quantifies this rule system. It is a very strong prior such that for all impossible actions, I set $p_{RL}(a|s) = 0$, which creates a mask for these actions. During training gradients will not flow through these actions either.

3.4.3.3 Permutation invariance

In genome annotation, chromosomes are assigned a numerical value, e.g. Chr1, Chr2, etc. However, these labels are arbitrary and, as a consequence, if a given copy number profile is picked, the copy number state of two chromosomes are ex-

changed, say Chr1 and Chr2, the inferred evolutionary trajectory should be exactly the same other than the labels for these two chromosomes have been exchanged, i.e. $\pi_{\phi_1}(c_1|s) = \pi_{\phi_1}(c_2|s')$, where s and s' are CNPs with the copy number state of chromosome c_1 and c_2 being changed.

In this work, the chromosome selection policy network $\pi_{\phi_1}(c|s)$ is built using convolutional networks with several layers of convolution, ReLu transformation and max-pooling at the bottom and fully connected linear layers at the top. However, in order to account for permutation invariance of the chromosomes, the following theorem is applied. The proof is attached in the appendix section as well.

Theorem 3.2. *For any linear operations (including convolution)* $\boldsymbol{A}$*, if it has block permutation invariant property, i.e.* $\boldsymbol{AP}\vec{x} = \boldsymbol{PA}\vec{x}, \forall \vec{x}$*, where* $\boldsymbol{P}$ *is any block permutation matrix, it necessarily has structure* $\boldsymbol{A} = \boldsymbol{1}\, diag(\boldsymbol{B}) + diag(\boldsymbol{C})$*, where* $\boldsymbol{1}$ *is a block matrix of the same block size filled with identity matrix, and* $diag(\boldsymbol{B})$ *and* $diag(\boldsymbol{C})$ *are block diagonal matrices.*

$$\boldsymbol{P_{ij}} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{I} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{0} & \cdots & \boldsymbol{I} \end{pmatrix}, \quad \boldsymbol{1} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \end{pmatrix}$$

$\boldsymbol{I}$ is the identity matrix with the block size the same as each chromosome, while $\boldsymbol{B}$ and $\boldsymbol{C}$ are any matrices with the same block size. Moreover, for nonlinear operations $h(\vec{x})$ that only depends on values within the same block, including ReLU activation and Max-pooling, the conjugate operations $\boldsymbol{A}h(\vec{x})$ and $h(\boldsymbol{A}\vec{x})$ stay block permutation invariant.

This structure is enforced in $\pi_{\phi_1}(c|s)$ which ensure permutation invariance with respect to chromosome labelling and also substantially reduces the number of parameters in the final fully connected linear layers (in the networks, this gives rise to $\approx 800\times$ fewer parameters). Since for convolutional neural networks, the fully connected linear layer is the most expensive layer in terms of number of parameters, this structure reduces both storage requirements and the training time for the algorithm.

Besides, such block structure does not prevent different chromosomes having different probabilities for CNA. For example, if chromosome c_1 is going to have a larger probability then other chromosomes c. Then $r(s, a_1)$ for actions a_1 on chromosome c_1 will be larger than other actions by a constant μ_1, i.e. $r(s, a_1) = r(s, a) + \mu_1$. Thus, it is possible to add this constant to the blocks related to c_1 on the output of the neural network, by $Q'_\phi(s, a_1) = Q_\phi(s, a_1) + \mu_1$. Then, if we take this new $Q'_\phi(s, a_1)$ into loss function (3.4), it is easy to check the loss function on Q_ϕ stays the same.

3.4.3.4 Switch and residual representation

The structure mentioned above for $\pi_\phi(a|s)$ has difficulty distinguishing samples with and without WGD because of the major difference between these two scenarios. To solve this problem, I implemented a special structure as follows, where ϕ_a, ϕ_b, ϕ_c are also subsets of the trainable parameter ϕ. (Since ϕ_a, ϕ_b, ϕ_c and ϕ_1, ϕ_2, ϕ_3 are different ways of subsetting the same parameter ϕ, they are not mutually disjoint.)

$$
\begin{aligned}
Q_\phi(s, a) &= [1 - \sigma(s)]Q_{\phi_a}(s, a) + \sigma(s)Q_{\phi_b}(s, a) \\
\sigma(s) &= \text{sigmoid}\left[(\text{ave}(s) - 1.5) + \text{res}_{\phi_c}(s)\right] \\
\text{sigmoid}(x) &= [1 + \exp(-x)]^{-1}
\end{aligned}
$$

Here, it is aimed to use a switch or gated unit, $\sigma(s)$, to learn if the CNP s is one with WGD or not. The function ave(s) calculates the average copy number of s and it is usually a good indicator for most CNP s. This ave(s) helps training at the beginning when $\sigma(s)$ doesn't know how to distinguish between CNPs with and without WGD. However, I add a residual function $\text{res}_{\phi_c}(s)$ to the ave(s) to shift the results when the average copy number is not a good indicator for WGD. Therefore, for $\sigma(s)$, all training parameters are in the $\text{res}_{\phi_c}(s)$ part and this is a way of residual representation.

3.4.3.5 Pseudocode

A pseudocode is provided as Algorithm 3.1.

Algorithm 3.1: `RLEvolution`

Result: Deep Neural Net: Q_ϕ

Random initialization of ϕ, step length ρ
while *Current Epoch* $<$ *Target Epoch* **do**
    Sample CNP s from the geometric distribution mentioned in Data Set section and a single CNA a uniformly;
    Compute next CNP s': apply the changes caused by CNA a to the sample CNP s;
    Compute reward $r(s, a)$ according to equation (3.6);
    Compute Q-function values: $Q_\phi(s, a)$ according to equation (3.7) and $\text{softmax}_{a'} Q_\phi(s', a')$ according to (3.5);
    Compute loss: $L = [Q_\phi(s, a) - r(s, a) - \text{softmax}_{a'} Q_\phi(s', a')]^2$;
    Take a gradient step on ϕ to minimize loss: $\phi = \phi - \rho \nabla_\phi L$
end

3.4.3.6 Parameter settings

The parameters for the architecture in Fig.3.7 are listed below in Table.3.1. Notice that though the parameter is quite large for ϕ_2 and ϕ_3, many of the output will be inactivated by the Rule system, because many of the loci on the genome will not contain any CNA and thus won't be considered. Thus, the rule system could play as the dropout regularization in training. There is no parameter in the residual part to be learnt because it is fully determined.

The model is optimised with Adam with learning rate 1×10^{-4}, $\beta = (0.9, 0.99)$, $\epsilon = 1 \times 10^{-6}$, weight decay$= 1 \times 10^{-3}$. Batch size is set to 30 and I used 3×10^6

Model Unit	Conv	FC
Switch	channel=[20,40,80] length=[3,3,5]	80× 1
$\Pi_{\phi_{1,a}}$,$\Pi_{\phi_{1,b}}$	[80,120,160] length=[5,3,3]	800× 44
$\Pi_{\phi_{2,a}}$,$\Pi_{\phi_{2,b}}$	[20,40,80,10] length=[5,5,5,5]	500 × 100
$\Pi_{\phi_{3,a}}$,$\Pi_{\phi_{3,b}}$	[20,40,10] length=[5,5,5]	500 × 50

Table 3.1. Parameters for Q function

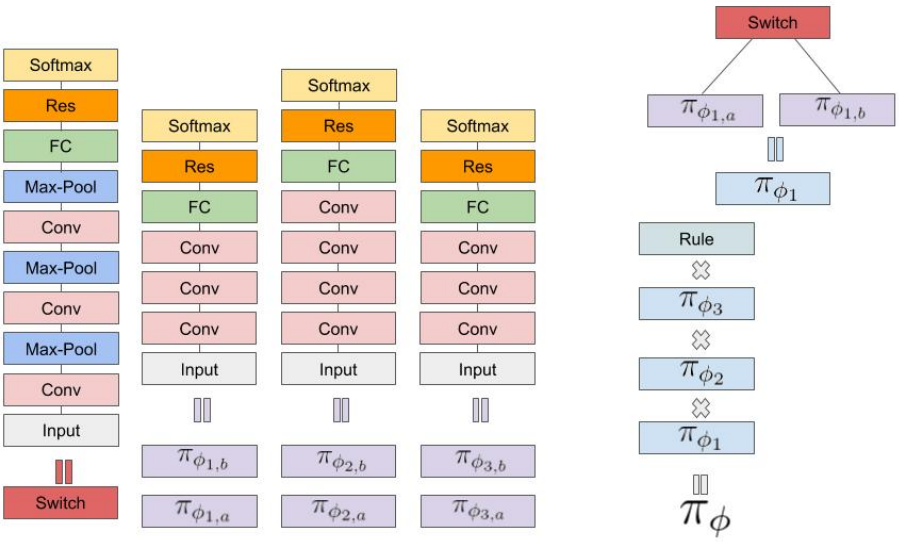


Figure 3.7. **Model Architecture.** ***This figure shows the main architecture of the model*** $\pi_\phi(a|s) \propto \exp(Q_\phi(s,a))$***. First, on the left hand side, it shows the basic structure of the switch structure in partitioning of the action space. The symbols used are the classically used convolution(Conv), max-pooling and fully-connected linear(FC) layers. There are activation layers(ReLU) in between. Res indicates the residual representation. On the top right corner, it shows the combination of two small units with the switch to form a sub-unit of the action space. Finaly, on the bottom right corner, it shows the final combination of all sub-units together with the rule system (real world physical constraint) to form the full model.***

iterations for training.

3.4.3.7 Identifiability of the trajectories

The identifiability of the trajectories is determined by the overlapping region of the CNA events. For example, WGD will have overlapping region with all other CNAs. Thus, the timing for WGD is fully identifiable with respect to other CNAs. A gain on chromosome 8 does not have any overlap region with a loss on chromosome 7 and therefore their relative timing is not identifiable. However, these two separate CNAs could be timed if they have different timing with respect to WGD. For example, if the gain on chromosome 8 happens before WGD and the loss on chromosome 7 happens after WGD, then for sure the gain on chromosome 8 happens before the loss on chromosome 7.

3.5 Experiments

3.5.1 Heuristic benchmarks

Except for `WCND` mentioned above[109], I also produced two benchmark techniques based on heuristics for comparison as well.

In the first (Heur1), I treat each continuous copy number segment as an individual CNA event and thus the number of evolutionary actions needed to generate a CNP is simply a function of the number of breakpoints in the CNP. This is naive in the sense that if, for example, in Fig.3.8 only four copy number alteration events are required to generate a given copy number profile. A heuristic based on counting each region as being affected by independent CNA events would overestimate the number of events required - in this case six steps instead of four. This approach consequently leads to an upper bound estimate of the number of copy number alteration events.

With this heuristic method, in order to extract the trajectory of copy number evolution, I simply go through every locus one by one. If the locus contains copy number other than 1, I find the nearest next locus on the same chromosome with different copy number as the end point for this CNA. If there is no such locus, then the end point is the end of this chromosome. If the copy number is more than 1, I consider this CNA to be a loss and gain otherwise.

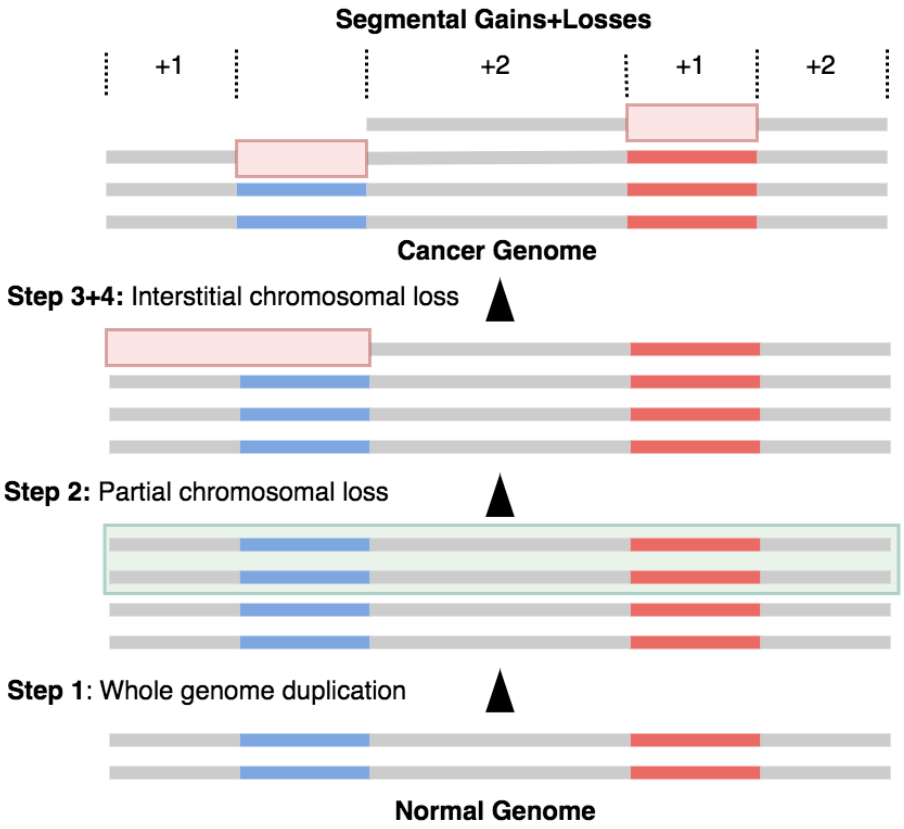


Figure 3.8. **Example showing naive over-estimation of evolutionary trajectory length.** ***Four copy number alteration events are required to transform a normal genome to the final altered genome. However, if each sub-region is assumed to be affected by independent CNA events to achieve the final altered genome then six steps are required. This example also illustrates why*** `RLEvolution` ***would be better than benchmark method*** `WCND`***. As*** `RLEvolution` ***only needs 3 steps to recover the copy number for this chromosome (except WGD which is shared by all chromosomes), however,*** `WCND` ***will need 4 steps as follows: a whole chromosome gain, a loss on the second segment (blue segment), a gain on the last 3 segments and finally a loss on the second last segment (orange segment).***

In the second approach (Heur2), I examine the average genome-wide copy number of each tumour. If the average copy number is greater than 1.7, which is the average of copy number for samples with WGD from TCGA data [104], I classify the tumour as having undergone WGD and change the copy number baseline accordingly for subsequent copy number measurement. I then consider each chromosome arm and if the average copy number of that arm is above or below the genome-wide copy number by more than 50% then it is then determined that chromosome arm has been duplicated or lost. This method would approximately account for WGD and allow the measurement of CNAs relative to this state.

Then, to extract the trajectory of evolution, I first compute the average of copy number of the whole CNP, and if it is above 1.7, it is considered to be a sample with WGD. I set the baseline for samples with WGD to be 2 copy and without WGD to be 1 copy. I then go through each chromosome first to check if there is whole chromosome change or not. If the average copy number of this chromosome is 50% larger/smaller than the baseline, it is considered that there is a whole chromosome loss or gain respectively. For samples with WGD, I go through every locus one by one and check if it contains odd number copy numbers. For a locus with odd copy number, I find the nearest next locus on the same chromosome with different copy number as the end point for this CNA. If there is no such locus, then the end point is the end of this chromosome. If the copy number is larger than 2 (the baseline), it is considered that this CNA is a loss, otherwise it is considered to be a gain. If then all loci contain even copy numbers, it is considered that the next action is a WGD and divide all copy numbers by 2 and this CNP finally becomes a CNP without WGD. For CNP without WGD, I go through each chromosome first to check if there is whole chromosome change or not as mentioned. After the chromosome checks, I go through each loci one by one, and determine the remaining CNAs similar to what I did for Heur1.

The limitation of these heuristics is that they cannot account for sequential sequences of CNAs that may overlap one another. This is the problem that `RLEvolution` uniquely solves and the following experiments demonstrate this effect.

3.5.2 Synthetic data experiments

I first conducted a simulation experiment in which I generated artificial copy number profiles. I simulated 100 samples in total, with half having a WGD event at some time point during the evolution. On average, I simulated approximately 50 copy number alterations for each sample and the probability for each type of CNA used in simulation comes from previous studies of clinical data [20]. The time point of WGD was also randomly selected but concentrated at around the 15th event during simulated evolution, following evidence that WGD is usually an early event in cancer [104, 107].

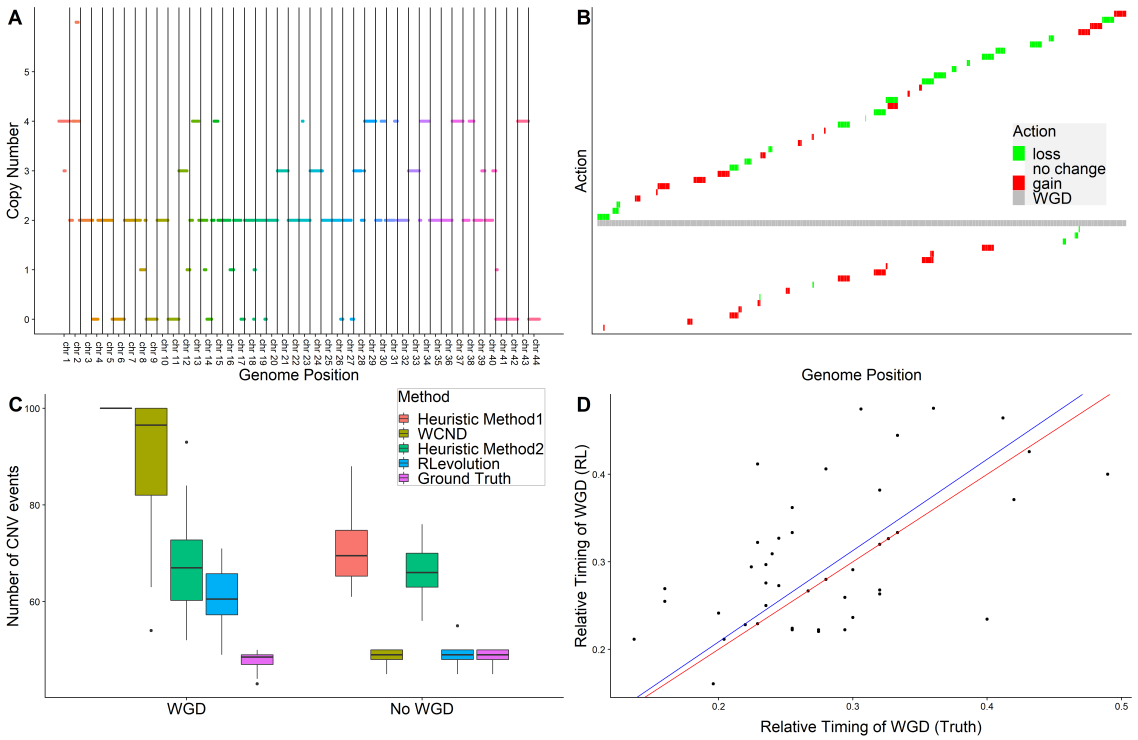


Figure 3.9. **Synthetic experiments.** ***(A) A simulated copy number profile consisting of multiple CNA events and WGD. (B) Inferred actions trajectory for the sample in (A). (C) Inferred trajectory lengths for simulated CNPs with and without WGD. (D) Relative timing of inferred WGD events compared to ground truth.***

I applied `RLEvolution` to the simulated samples and an example is shown in Fig.3.9A. `RLEvolution` is able to infer and report a trajectory consisting of all the CNA events (actions) that modified an otherwise normal genome into the observed abnormal cancer genome (Fig.3.9B). When I examined the length of the trajectories inferred using `RLEvolution` in comparison to the ground truth and the other approaches detailed previously. Fig.3.9C shows that, in the absence of WGD, both `RLEvolution` and `WCND` were able to correctly infer the number of evolutionary events in multiple instances whilst the heuristic methods substantially overestimate. In the presence of more complex WGD-affected samples, `RLEvolution` over-estimates the number of CNA events but substantially less so than the heuristic I and `WCND`. Importantly, Fig.3.9D shows that the relative timing of the inferred WGD event given by `RLEvolution` correlates strongly with the true timing. Note the heuristic methods described cannot be used to determine the relative timing of the WGD event.

I next examined if the exact sequence of actions predicted by `RLEvolution` matched the true sequence. Fig.3.10A illustrates an example step-to-step comparison for one

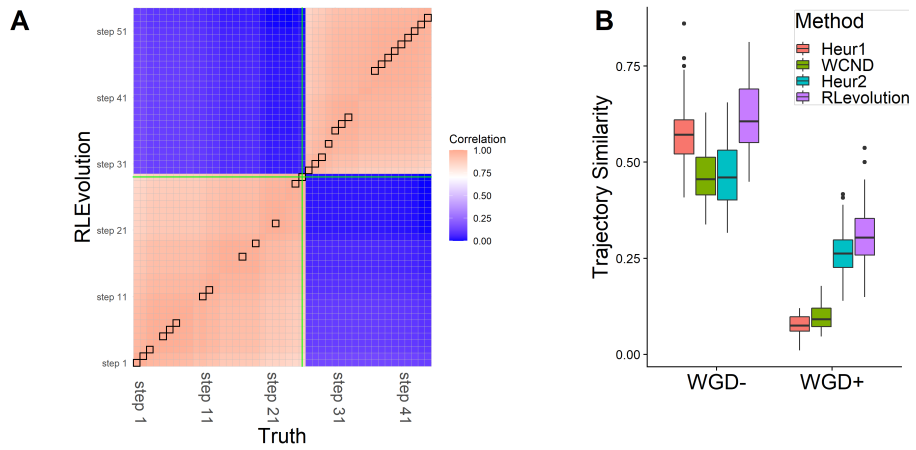


Figure 3.10. **Compare Trajectory Similarity.** ***(A) The heatmap shows the correlation/similarity between the CNP of the ground truth (simulated) and CNP inferred by*** `RLEvolution` ***after each step. The correlation is computed by the number of loci with the same copy number divided by the total number of loci. The black box are the CNAs(actions) that are identical between the ground truth and the inferred CNAs by*** `RLEvolution`***. The green line is the WGD event. (B) The boxplot of similarity of the trajectory by different methods. The trajectory similarity is computed by the number of identical actions (black boxes in graph A) between ground truth and inferred CNAs, divided by the total number of inferred CNAs (number of rows in graph A).***

simulation. Since `RLEvolution` does not estimate the exact number of steps required precisely, I compared the CNP after every `RLEvolution` action against each step of the ground truth CNPs and looked at the correlation between the CNPs. It could be observed that `RLEvolution` maintains strong consistency with the ground truth CNPs even though the exact sequence of actions is not precisely the same. In general, it is observed that some long copy number changes are fragmented by `RLEvolution` into distinct separate changes leading to overestimates of the number of steps. Fig.3.10B shows that the CNPs from `RLEvolution` maintain greater similarity to the ground-truth trajectories than CNPs derived using the other methods.

3.5.3 TCGA colorectal cancer

I next analysed 380 copy number profiles from colorectal cancers (CRCs) from The Cancer Genome Atlas (TCGA)[97]. The copy number data is collected by Affymetrix SNP 6.0 microarrays and the CNP is computed by `oncosnp`[67]. I used `RLEvolution` to infer if and when WGD occurred for each tumour and to determine its timing relative to other copy number alteration events.

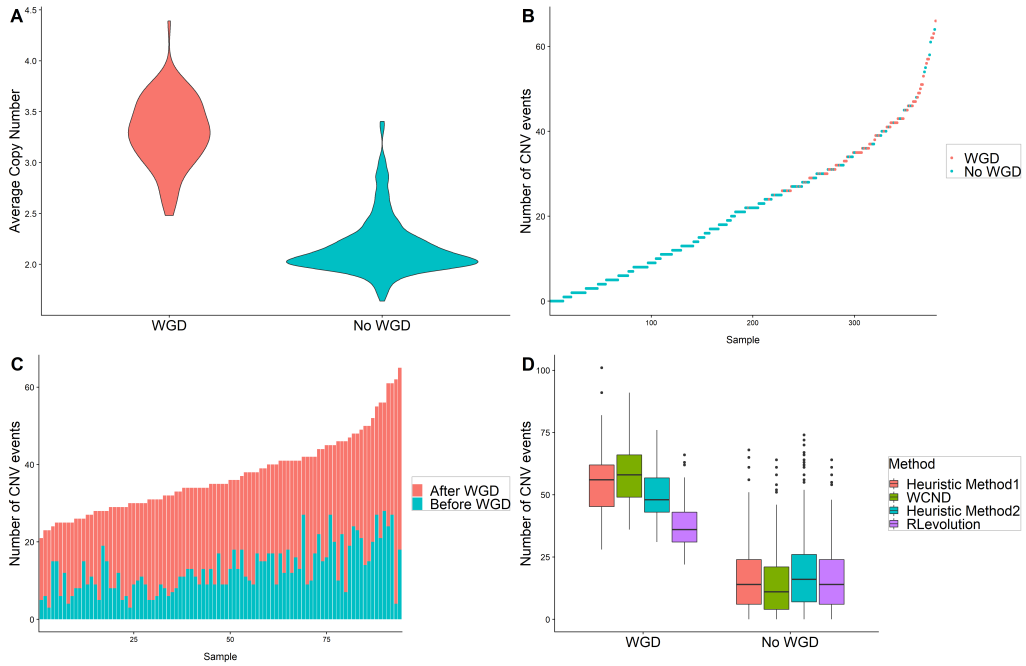


Figure 3.11.* TCGA CRC data analysis.** ***(A) Violin plot of average genome-wide copy number against inferred WGD status. (B) Number of actions required to generate each tumour copy number profile. (C) Timing of WGD relative to other CNA events. (D) Reconstruction accuracy.

In such data, CRCs typically fall into two distinct classes, tumours which exhibit hypermutability known as microsatelite instability (MSI) or those which show chromosomal instability (CIN) with many chromosomal abnormalities [101]. Tumours in the latter class typically possess much more complex CNPs than the former. Some of these CNPs suggest that whole genome duplication has occurred at some point in the evolution of the tumour.

Fig.3.11A shows the average genome-wide copy number of each TCGA CRC against inferred WGD status by `RLEvolution`. As expected, tumours with WGD possess a higher average copy number compared to those which have not undergone WGD. I then examined the number of actions required to generate each CNP against average genome-wide copy number and number of breakpoints. Fig.3.11B shows that genomically complex tumours with higher average copy number and breakpoints require more CNAs in order to be formed. Although it is not possible to know for certain, these results provide evidence `RLEvolution` that is inferring evolutionary histories that are intuitively consistent with the tumour data.

I then examined the timing of WGD estimated from trajectories found by `RLEvolution`.

This suggested that WGD is a relatively early event in the evolution of the tumours and tends to occur at a consistently similar time during evolution (approximately 10th event). The consistent timing suggests that the causal factors for the WGD event are common in colorectal cancers and appear to push certain cancer genomes into a mitotic crisis. This finding is consistent with previous finding by using other approaches[18].

Finally I compared the number of actions required by `RLEvolution` compared to the other breakpoint-based approaches (Fig.3.9D). In the absence of WGD all methods yielded trajectories of similar length, while in samples with WGD, `RLEvolution` tends to do slightly better than other approaches.

3.5.4 Exploratory analysis on genome duplicated cell line data

Here, I have applied `RLEvolution` to copy number data from cultured cancer cell line (kuramochi cell line) to show a potential application of this method. The cell line was split into two samples, and one of them has been treated with taxol, a chemotherapy drug (Fig.3.12). The drug is known to inhibit spindle function and results in chromosome aberrations. As a consequence, cells treated with taxol should contain more CNA than the untreated cells, whereas they should still share a significant number of other CNAs.

I called the CNP of both samples by `oncosnp`. Then, to make the deconvolution process easier for the method, I cleaned and filtered the copy number profile with the following procedures: 1). I filtered out all loci on the genome where the total copy number is larger than 6; 2). I filtered out the fragments shorter than 2Mb (0.07% of human genome) that have different copy numbers between these two samples; 3) I sampled 50 loci with uniform distribution for each chromosome to form the final copy number profile for my method. The final CNP of two samples and their difference is shown in Fig.3.13.

I then feed the copy number profile of both samples into my `RLEvolution` model and get the trajectories for copy number evolution history(Fig.3.14). Note that the

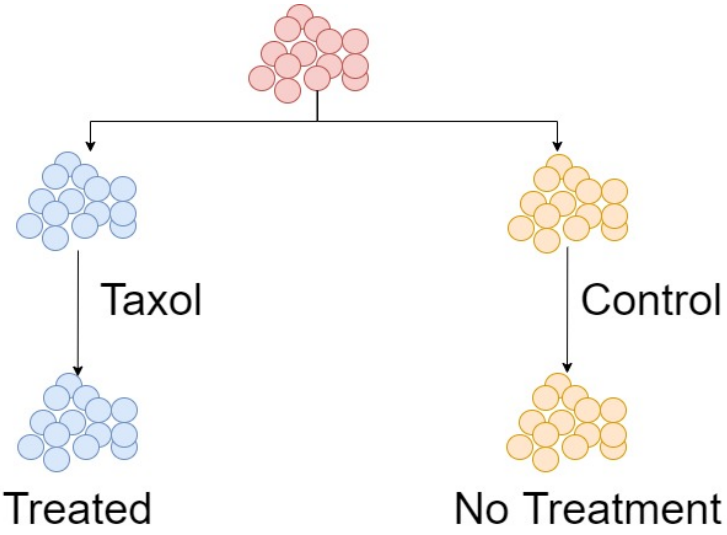


Figure 3.12. **Cell Line Data** ***The cell line was split into two samples, and one of them has been treated with taxol. The DNA is extracted from these two samples and sequenced.***

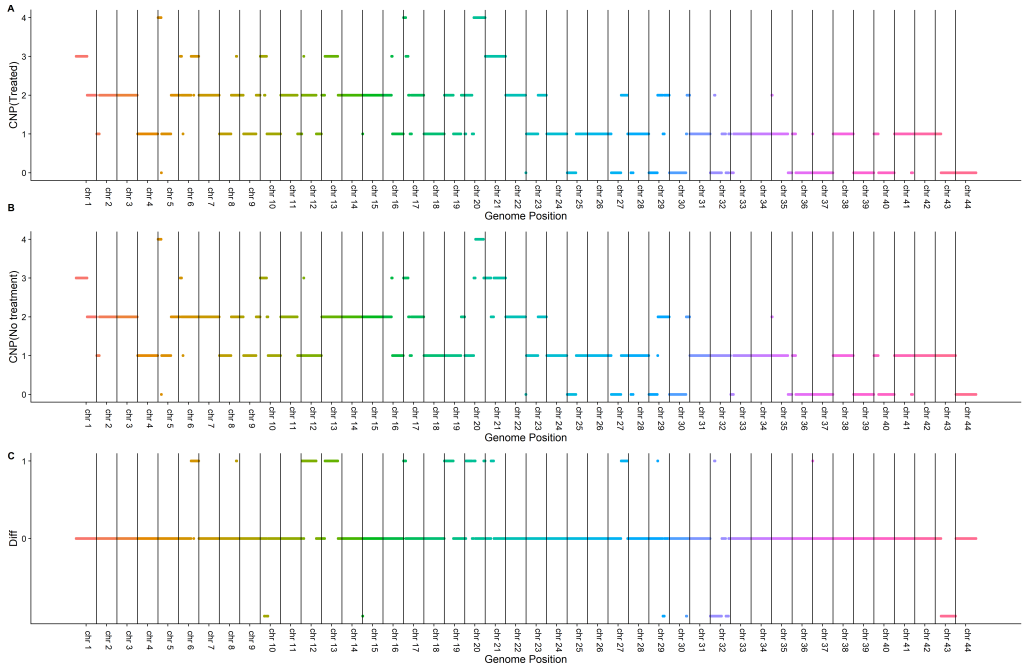


Figure 3.13. **Copy number profile of cell line data** ***(A) Copy number profile after filtering and cleaning of kuromachi cell line being treated with taxol. (B) Copy number profile after filtering and cleaning of kuromachi cell line without treatment. (C) The difference between two copy number profiles shown in (A)(B)***

deconvoluted trajectory actually is the reverse for copy number evolution. Thus step 1 is the last CNA in the evolution. As expected, most CNAs happened after WGD and most of these ones are losses (gains in the deconvoluted trajectories) after WGD[18, 107]. Besides, the sample treated with taxol contains more CNAs as

expected.

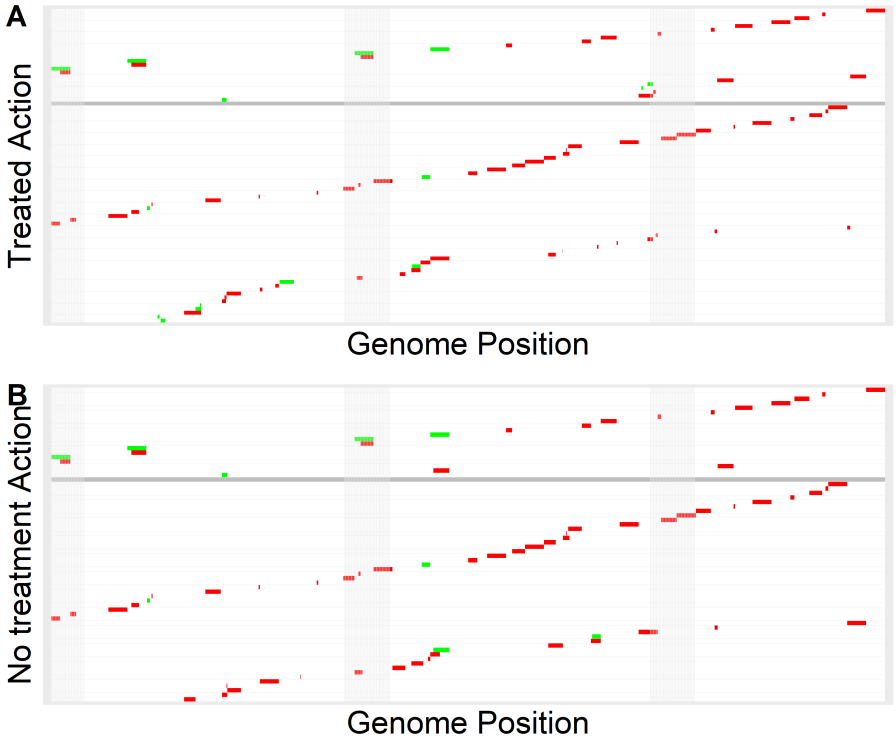


Figure 3.14. **Deconvoluted trajectory for copy number evolution by** `RLEvolution`. ***Deconvoluted step for the sample treated (A) with taxol and (B) without treatment. The y-axis is the timing of the step in the deconvoluted trajectory and x-axis is the genome position. The color legend is the same as the one for Fig.3.9B. The deconvoluted trajectory is the reverse of the evolution history, so that the history of evolution goes from top to bottom, and a gain in the deconvoluted trajectory is actually a loss in the evolution process. All red regions showed the loss in copy number evolution.***

Finally, I compared the similarity of these two trajectories. The similarity is defined to be the percent of genome positions that contain the same copy numbers between two samples. it could be shown from Fig.3.15 that these two trajectories share a lot of common steps. From the results, the divergence in the evolution of copy number happens at step 19 for the sample without treatment and step 25 for the sample with treatment. (The line of black boxes in Fig.3.15 ends at y axis 19 and x axis 25.) The divergence between 51-53 for the sample without treatment and 58-64 for treated sample could be either due to the sequencing error or the wrong timing caused during deconvolution by `RLEvolution`. (The line of black boxes in Fig.3.15 is fragmented into 2 part in y axis 51-53 and x axis 58-64.) This is potentially a direction which needs further improvement in the future. However, these CNAs which are not correctly timed is short compared to the full trajectory.

It can also be computed that the number of CNAs shared between these two samples accounts for 70% of the CNAs in total in the sample without treatment (number of black boxes divided by number of rows) and accounts for 60% of the treated sample (number of black boxes divided by number of columns). This showed that the trajectories inferred by `RLEvolution` have most of CNAs in common between the two samples of the same origin.

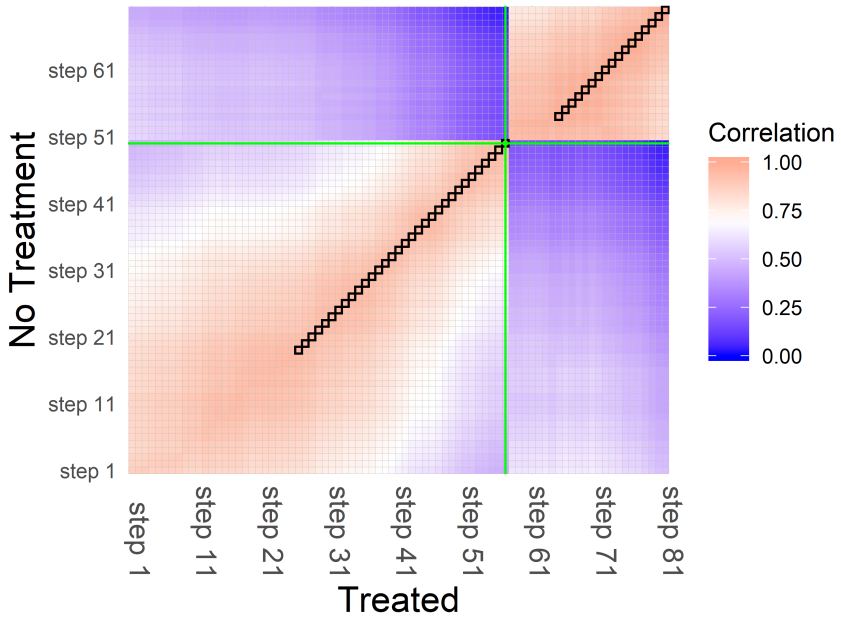


Figure 3.15.* Similarity between deconvoluted trajectories by `RLEvolution` *I compute the similarity between the CNP of two samples after removing the effect of each step in the trajectory. The color of each box indicates the similarity. The black box are the CNAs(actions) that are identical between the ground truth and the inferred CNAs by* `RLEvolution`*. The green line is the WGD event. The color of the heatmap indicate the similarity between two genomes at this step.

In comparison, I applied `WCND` to this data set as well, while keeping all other procedures, including data pre-processing, and similarity computation un-changed. It is shown (Fig.3.16) that the two trajectories inferred by `WCND` share little in common, with 29% for sample without treatment and 31% for the treated sample.

Thus, this concludes that my method `RLEvolution` could recover more similarities between these two samples of the same origin than existing method `WCND`.

Besides, this data also gives us a detailed example showing why `RLEvolution` would work better. As shown in Fig.3.13AB, for these two samples, chromosome 1 has mostly 3 copies except for a short region on q arm. As shown in Fig.3.14,

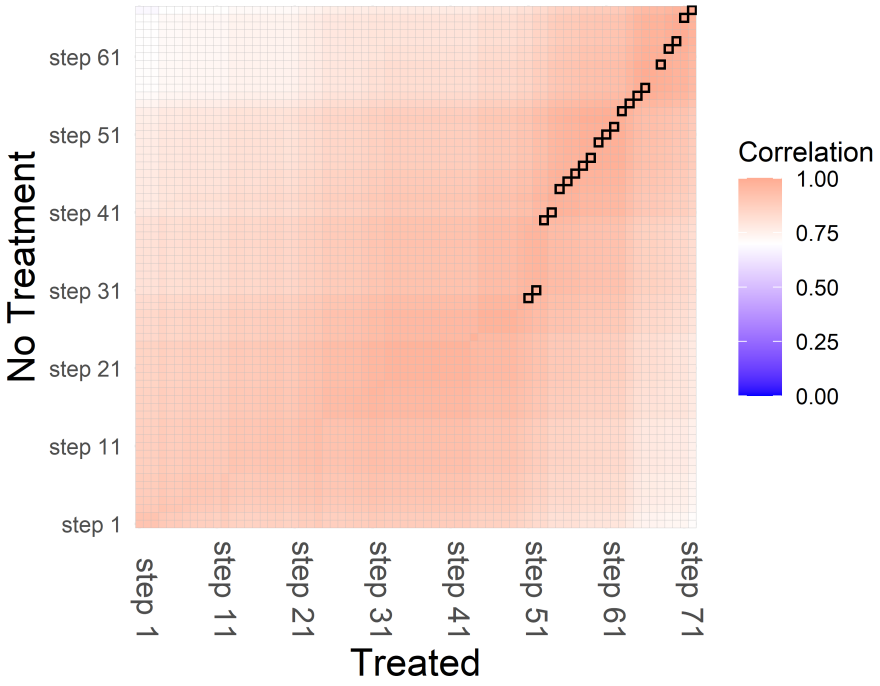


Figure 3.16. **Similarity between deconvoluted trajectories by `WCND`.** ***It is the same as Fig.3.15 except that the trajectory is computed by*** `WCND`**.**

`RLEvolution` explains this as: a gain for whole chromosome 1 and then a loss on the short region of q arm, followed by WGD and a loss on the remaining region of chromosome 1. However, for `WCND`, it explains the copy number of chromosome 1 as a gain on the short region of q arm and 2 gains on the remaining region. As it is been reported that full chromosome change usually has a slightly higher probability of occurring[20], it is quite likely that the trajectory found by `RLEvolution` is closer to the truth.

3.6 Discussion

I have described a fully-algorithmic approach for learning the evolutionary trajectories of cancers from whole-genome DNA copy number profiles. It is the first reinforcement learning method in the field for cancer evolution. This model-free reinforcement learning based approach makes it possible to solve a large state-space

Markov Decision Process that underlies the genetic evolution model.

I showed from synthetic experiments that `RLEvolution` is able to infer evolutionary trajectories which are superior than ad-hoc heuristic approaches as well as `WCND`. From real cancer data I developed inferences that are consistent with prior qualitative observations[18, 107]. Finally, I showed with an example how `RLEvolution` could be applied to real-world experimental data and in this example, it is verified that `RLEvolution` generates trajectories that are more consistent with the experimental design than `WCND`. Further experimental work will be required to validate my findings but this approach represents a capability that is better than the one existing in the computational biology literature. It is also a platform for further innovation for the study of copy number evolution.

Here, I only showed the study of `RLEvolution` on CRC data. It could be applied to more types of cancer in the future. The results of such studies could potentially contribute to the understanding of the different biological mechanism of copy number evolution among different types of cancer.

This method could also be used to compute the similarity of two CNPs by using the similarity metric in Fig.3.10A. Such similarity could then be used to reconstruct phylogeny tree of cancer evolution from copy number data similar to the application of `WCND`[109]. The traditional approaches using euclidean distance tend to be heavily affected by WGD [129] and the similarity computed by `RLEvolution` is potentially less affected by the presence of WGD.

Besides, for special cases where the initial state of a sample is not a completely normal genome , one could still use `RLEvolution` to get an approximated trajectory. The way to do so is to compare the CNP to the initial state and extract all the CNAs. Next, one could manually construct a CNP with all these CNAs being applied to a completely normal genome, and then apply `RLEvolution`. This is certainly only a heuristic method and doesn't guarantee success for all cases.

There are also several improvements that could be made in future for `RLEvolution`.

The improvement of scalability will be required to achieve single-nucleotide resolution and the incorporation of point mutations alongside the chromosomal alterations.

Currently in the model, 50 loci for a chromosome is considered, due to the lack of scalability of the model. Theoretically, if the number of loci per chromosome considered increases, the result will be more accurate. However, in the current state of `RLEvolution`, the number of parameters for the deep neural network is proportional to the number of loci and the training time will increase if the number of parameters grows. This factor thus limits the scalability of the model. In the future, it is possible to limit the number of parameters by using special neural net architectures which are more parameter efficient[130].

Second, the performance `RLEvolution` heavily depends on the performance of the copy number caller used. As described before, in order to use `RLEvolution`, a CNP needs to be generated by a copy number caller. However, sequencing data of cancer samples are highly noisy. For example, the cancer sample may not be pure, as there might be cells from subclones and even cells from normal tissues mixed in the sample. Some copy number callers have taken this particular problem into consideration[67]. Thus, it would be interesting to compare the results of `RLEvolution` by using different copy number callers.

Third, as illustrated in the final example (Fig.3.15), the output of `RLEvolution` did not recover the ground truth with full accuracy. Thus, improving the performance of the `RLEvolution` is also important. The Q-learning algorithm used in `RLEvolution` is a classic algorithm in the field of reinforcement learning, however there are many more advanced algorithms developed in recent years[131–134]. It will be beneficial to retrain the `RLEvolution` with these advanced algorithms and evaluate model accuracy.

Overall, in this chapter, I have developed a method `RLEvolution`, to the best of my knowledge the first application of reinforcement learning in the study of cancer evolution. It has shown to be better than simple heuristics and previous method `WCND`, though it still shows potential for improvement. I believe that this approach will contribute to a better understanding of copy number evolution in cancer in future.

3.7 Appendix

Proof for Theorem 3.1. I first simply the target function for the first problem as follows.

$$
\begin{aligned}
&- \mathrm{E}_s \sum_{\tau_T} \Pi_\phi(\tau_T|s) \log \frac{P(\tau_T|s)}{\Pi_\phi(\tau_T|s)} \\
=&- \mathrm{E}_s \sum_{\tau_T} \Pi_\phi(\tau_T|s) \log \frac{P(\tau_T)}{\Pi_\phi(\tau_T|s)} + Const \\
=&- \mathrm{E}_s \sum_{\tau_T} \left[\prod_{i=0}^{T} \pi_\phi(a_{i+1}|s_i)\right] \sum_{j=0}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j)] + Const \\
=&- \mathrm{E}_s \sum_{\tau_T} \sum_{j=0}^{T} \left[\prod_{i=0}^{T} \pi_\phi(a_{i+1}|s_i)\right] [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j)] + Const
\end{aligned}
$$

As defined in main text, $P(\tau_T) = p(s_T) \prod_{t=1}^{T} p(a_{T-t+1}) = \prod_{t=0}^{T} q(a_{T-t+1}, s_{T-t})$ and $r(s_i, a_{i+1}) = \log q(a_{i+1}, s_i)$ here.

This is quite similar to the simplification steps used in classic reinforcement learning, whereas here everything is in *backwards-in-time* and the reward has a special probability interpretation.

I then calculate the gradient of the first optimization problem as follows.

$$
\begin{aligned}
&-\nabla_\phi \mathrm{E}_s \sum_{\tau_T} \sum_{j=0}^{T} \left[\prod_{i=0}^{T} \pi_\phi(a_{i+1}|s_i) \right] [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
&= -\mathrm{E}_s \sum_{j=0}^{T} \sum_{\tau_T} \left[\prod_{i=0}^{T} \pi_\phi(a_{i+1}|s_i) \right] \sum_{k=0}^{j} [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))][r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
&= -\mathrm{E}_s \sum_{k=0}^{T} \sum_{\tau_T} \left[\prod_{i=0}^{T} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \sum_{j=k}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
&= -\mathrm{E}_s \sum_{k=0}^{T} \sum_{\tau_T} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))][r(s_k, a_{k+1}) - \log(\pi_\phi(a_{k+1}|s_k))] \\
&\quad -\mathrm{E}_s \sum_{k=0}^{T} \sum_{\tau_T} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&\qquad \left[\prod_{i=k+1}^{T} \pi_\phi(a_{i+1}|s_i) \right] \sum_{j=k+1}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
&= -\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{T+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))][r(s_k, a_{k+1}) - \log(\pi_\phi(a_{k+1}|s_k))] \\
&\quad -\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&\qquad \sum_{a_{k+2} \ldots a_{T+1}} \left[\prod_{i=k+1}^{T} \pi_\phi(a_{i+1}|s_i)) \right] \sum_{j=k+1}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))]
\end{aligned}
$$

Notice that the second term in the last equation exists only when $k+1 <= T$, which means $a_{k+1}! = \text{END}$. Here I define:

$$
\begin{aligned}
C_\phi(s_{k+1}) &\coloneqq \sum_{a_{k+2} \ldots a_{T+1}} \left[\prod_{i=k+1}^{T} \pi_\phi(a_{i+1}|s_i) \right] \sum_{j=k+1}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
&= \sum_{a_{k+2} \ldots a_{T+1}} \Pi_\phi(\tau_{T-k}|s_{k+1}) \left[\log \frac{P(\tau_{T-k}|s_{k+1})}{\Pi_\phi(\tau_{T-k}|s_{k+1})} + \log \left(\sum_{\tau'_{T-k}:s_0=S} P(\tau'_{T-k}) \right) \right]
\end{aligned}
$$

Thus, $C_\phi(s_{k+1})$ could be seen as the KL divergence between the un-normalised theoretical and parametric distribution of all trajectories τ_{T-k} that transforms normal genome s_T to s_{k+1}, which is also the so-called reward-to-go.

Next, it is possible to further simplify the results as follows.

$$
\begin{aligned}
\nabla_\phi = &-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{T+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))][r(s_k, a_{k+1}) - \log(\pi_\phi(a_{k+1}|s_k))] \\
&-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&\sum_{a_{k+2} \ldots a_{T+1}} \left[\prod_{i=k+1}^{T} \pi_\phi(a_{i+1}|s_i)) \right] \sum_{j=k+1}^{T} [r(s_j, a_{j+1}) - \log(\pi_\phi(a_{j+1}|s_j))] \\
= &-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&[r(s_k, a_{k+1}) - \log(\pi_\phi(a_{k+1}|s_k)) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1})] \\
= &-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&[r(s_k, a_{k+1}) - \log(\pi_\phi(a_{k+1}|s_k)) - C_\phi(s_k) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1})]
\end{aligned}
$$

Here I define:

$$
Q_\phi(a_{k+1}, s_k) \coloneqq \log(\pi_\phi(a_{k+1}|s_k)) + C_\phi(s_k)
$$

It is can be verified that:

$$
\mathrm{softmax}(Q_\phi(a_{k+1}, s_k)) = \log \left(\sum_{a_{k+1}} \exp^{Q_\phi(a_{k+1}, s_k)} \right) = C_\phi(s_k)
$$

Then there is:

$$
\begin{aligned}
\nabla_\phi = &-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&[r(s_k, a_{k+1}) - Q_\phi(a_{k+1}, s_k) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1})] \\
= &-\mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \\
&[r(s_k, a_{k+1}) - Q_\phi(a_{k+1}, s_k) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) \mathrm{softmax}(Q_\phi(a_{k+2}, s_{k+1}))]
\end{aligned}
$$

Now, it is hypothesised that the number of action is bounded by N, and the model

is properly designed that the gradient is bounded, i.e. $\|\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))\| \leq L, \forall s, a$. Thus, by Holder's inequality (when the sum is countably infinite) that $[\sum_i a_i b_i]^2 \leq (\sum_i a_i^2)(\sum_i b_i^2)$, the following result could be derived. Such result indicates when minimizing the loss function in softmax Q-learning, it is equivalent to minimizing the upper bound for norm of gradient for this KL distance function. This also shows why in Q-learning I don't need to sample from the policy network, which makes such method off-policy.

$$
\begin{aligned}
\|\nabla_\phi\|^2 = & \left\{ \mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] [\nabla_\phi \log(\pi_\phi(a_{k+1}|s_k))] \times \right. \\
& \left. [r(s_k, a_{k+1}) - Q_\phi(a_{k+1}, s_k) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1})] \right\}^2 \\
\leq & \left\{ \mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_{i+1}|s_i) \right] L \times \right. \\
& \left. [r(s_k, a_{k+1}) - Q_\phi(a_{k+1}, s_k) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1})] \right\}^2 \\
\leq & \, \mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} \left[\prod_{i=0}^{k} \pi_\phi(a_i|s_i)) \right]^2 L^2 \times \\
& \mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} [r(s_k, a_{k+1}) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1}) - Q_\phi(a_{k+1}, s_k)]^2 \\
\leq & \, L^2 \times \mathrm{E}_s \sum_{k=0}^{T} \sum_{a_1 \ldots a_{k+1}} [r(s_k, a_{k+1}) + \mathbb{1}(a_{k+1} \neq \mathrm{END}) C_\phi(s_{k+1}) - Q_\phi(a_{k+1}, s_k)]^2 \\
\propto & \, \mathrm{E}_{s,a} [r(s, a) + \mathbb{1}(a \neq \mathrm{END}) C_\phi(s') - Q_\phi(a, s)]^2
\end{aligned}
$$

In the last line, I used a trick that s_k and a_{k+1} are simple samples from the distribution of trajectory, and I simply sample a random s to be s_k and a random CNA a to be a_{k+1}, with $s' = s_{k+1}$ is a CNP that will be transformed to s_k by a_{k+1}.

And it follows that, if $\pi^*(a|s) = \frac{q(a,s) \sum_{\tau'_T : s_0 = S'} P(\tau'_T)}{\sum_{\tau_T : s_0 = S} P(\tau_T)}$, where a is a CNA that transforms S' to S and $\pi^*(a = \mathrm{END}|s) = \frac{q(a=\mathrm{END}, s)}{\sum_{\tau_T : s_0 = S} P(\tau_T)}$, the target function for the second optimization gets 0. It is also quite easy to check that $\pi^*(a|s)$ is a probability distribution over all actions including the special one and $\Pi^*(\tau_T|S) = P(\tau_T|S)$. Since the target function is a quadratic function, which is certainly greater or equal to 0,

it indicates that $\pi^*(a|s)$ will still be a global minimum for the second optimization problem as well.

It could also be noticed that with such optimal solution $Q^*(a,s) = \log(\pi^*(a|s)) + C_\phi(s)$, there is $Q^*(a = \text{END}, s) = \log(\pi^*(a = \text{END}|s)) + C_\phi(s) = \log(q(a = \text{END}, s)) = p(s_T = s)$.

Thus, in the training procedure, I am actually dealing with the following (approximation) to the true problem, with the global optimum unchanged.

$$\begin{aligned} \min_{\phi} \quad & \mathrm{E}_s \mathrm{E}_a \left[Q_\phi(s,a) - r(s,a) - \text{softmax}(Q_\phi(s',a')) \right]^2 \\ \text{s.t.} \quad & Q_\phi(a = \text{END}, s) = \log(q(a = \text{END}, s)) \end{aligned}$$

Due to the constraint, I don't need to sample $a = \text{END}$, and thus in the target function, I only need to consider cases when $a \neq \text{END}$.

It is easier to understand how I am doing the training when looking at the last equation. However, it is not obvious how I get rid of $\pi_\phi(a|s)$ from the last equation. Thus, in theorem 3.1, I am still including the formula with $\pi_\phi(a|s)$

□

Proof for Theorem 3.2. I first show that $\boldsymbol{A}$ necessarily has such structure.

Case 1: if $\boldsymbol{A_{ii}} \neq \boldsymbol{A_{ij}}$, it is possible to find a vector $\vec{x}$ where $\vec{x}_i \neq 0$ and 0 everywhere else. Then, for permutation matrix $\boldsymbol{P} = \boldsymbol{P_{ij}}$ that only permutes block i, j, $(\boldsymbol{AP}\vec{x})_j = \boldsymbol{A}_{jj}\boldsymbol{x}_i \neq \boldsymbol{A}_{ii}\vec{x}_i = (\boldsymbol{PA}\vec{x})_j$.

Case 2a: if $\boldsymbol{B}_{ij} \neq \boldsymbol{B}_{ik}$, where i, j, k are mutually different, it is possible to find a vector where $\vec{x}_j \neq \vec{x}_k$ and 0 elsewhere, with a permutation matrix $\boldsymbol{P} = \boldsymbol{P}_{jk}$ that only permutes block j, k, $(\boldsymbol{APx})_i = \boldsymbol{B}_{ij}x_k + \boldsymbol{B}_{ik}x_j \neq \boldsymbol{B}_{ij}\vec{x}_j + \boldsymbol{B}_{ik}\vec{x}_k = (\boldsymbol{PA}\vec{x})_i$.

Case 2b: if $\boldsymbol{B}_{ik} \neq \boldsymbol{B}_{jk}$, where i, j, k are mutually different, it is possible to find a vector where $\vec{x}_k \neq 0$ and 0 elsewhere, with a permutation matrix $\boldsymbol{P} = \boldsymbol{P}_{ij}$ that only permutes block i, j, $(\boldsymbol{AP}\vec{x})_i = \boldsymbol{B}_{ik}\vec{x}_k \neq \boldsymbol{B}_{jk}\vec{x}_k = (\boldsymbol{PA}\vec{x})_i$.

Case 2c: if $\boldsymbol{B}_{ij} \neq \boldsymbol{B}_{jk}$, where $i, j, k$are mutually different, then there is either

$\boldsymbol{B}_{ij} \neq \boldsymbol{B}_{ik}$ or $\boldsymbol{B}_{ik} \neq \boldsymbol{B}_{jk}$, which is covered in case 2a and 2b.

Case 2d: if $\boldsymbol{B}_{ij} \neq \boldsymbol{B}_{kl}$, where i, j, k are mutually different, and j, k, l are mutually different, there is either $\boldsymbol{B}_{ij} \neq \boldsymbol{B}_{jk}$ or $\boldsymbol{B}_{jk} \neq \boldsymbol{B}_{kl}$, which is covered in case 2c.

And next, it is possible to check, when $\boldsymbol{A} = \boldsymbol{1}\mathrm{diag}(\boldsymbol{B}) + \mathrm{diag}(\boldsymbol{C})$, there is $(\boldsymbol{A}\vec{x})_i = (\boldsymbol{C}\vec{x})_i + \sum_k (\boldsymbol{B}\vec{x})_k$ and $(\boldsymbol{A}\vec{x})_j = (\boldsymbol{C}\vec{x})_j + \sum_k (\boldsymbol{B}\vec{x})_k$. By setting $\boldsymbol{P} = \boldsymbol{P}_{\boldsymbol{ij}}$, i.e. exchanging i,j on the left hand side, it is equivalent to exchanging i,j on the right hand side. And since all permutation matrices $\boldsymbol{P}$ could be represented by the product of $\boldsymbol{P}_{\boldsymbol{ij}}$, $\forall \boldsymbol{P}, \boldsymbol{PA} = \boldsymbol{AP}$ holds.

Finally, The nonlinear case follows immediately because $f(x) = f(y), \text{if } x = y$ for any nonlinear function $f(x)$. Note that this is only a sufficient condition but not a necessary condition here, i.e. $f(x) = f(y)$ doesn't imply $x = y$.

□

Chapter 4

A curvature regularization based method for joint dimension reduction and clustering

4.1 Background

With advances in biotechnology, it is possible to profile the transcriptome of individual cells, a method referred to as single-cell RNA-sequencing (scRNA-seq)[135]. The accumulated scRNA-seq data provides an opportunity to investigate the composition and dynamics of mixed cell populations in cancer samples. For example, scRNA-seq data has been used to identify different molecular subtypes in ovarian cancer[136, 137], head and neck cancer[138] and colorectal cancer[139]. Besides, such data has also provided an opportunity to investigate the biological processes in unprecedented detail, e.g. the differentiation process during fibroblast reprogramming[140] and the evolution of lung cancer cells under drug treatment[141].

Despite the great amount of information provided by scRNA-seq data, it is not easy to draw robust conclusions from such data by using traditional approaches, e.g. linear regression due to curse of dimensionality as mentioned previously. Besides, the data collected from scRNA-seq contains substantial amount of random error caused during library preparation and sequencing, which again increases the complexity in the analysis of such data[78, 142, 143].

Consequently, a dimension reduction method is often used before other analysis, including differentially expressed gene discovery and clustering analysis for the discovery of molecular subtypes[144]. The classic method in scRNA-seq analysis is principal component analysis (PCA) because of its simplicity [145, 146]. However, because of the linearity constraint in PCA, it sometimes fails to capture the true low dimensional structure in the data and thus other non-linear methods have been applied[147, 148], discussed in detail later. Except for statistical robustness, another advantage of reducing the dimension to either 2 or 3 is that the cells can be plotted in either a 2d or 3d graph after dimension reduction, so that it is possible to have some visual understanding of the general structure between cells, i.e. the cells having similar gene expression patterns and the cells having different gene expression patterns[149]. One particular type of dimension reduction method in scRNA-seq data analysis falls into the field of pseudotime algorithms, where the data will be reduced to only 1 dimension. This final dimension is hypothesised to be associated with physical time for the single cells, and therefore, this pseudotime algorithm could be used to analyse temporal changes of single cells in processes such as development or cancer evolution[150].

For cancer evolution, typically only one sample is collected from each patient, which makes it a challenge to find out temporal information. However, it is still possible to use dimension reduction or pseudotime algorithms to get such information in the following two ways. Firstly, as a consequence of cancer evolution, in the single sample collected, there are usually tumour cells belonging to the major clone and sub clones. The transition between major and sub clones contain the information of temporal evolution during cancer progress, which might be detected by the algorithms. Secondly, in a cohort study, there are usually patients in different stage of the cancer development. Thus, applying the algorithms to the cohort, could potentially identify the temporal evolution of the tumour cells.

While dimension reduction tools are helpful to unravel low dimensional latent structure among single cells, clustering methods are often used to identify latent subtype structure in the data set. For example, one classic clustering method called K-means has been applied to intestinal scRNA-seq data to identify molecular subtypes[151].

Such subtype structure information for cancer studies is critical as it may enable personalised and different treatment of patients depending on subtype compositions[152]. Hierarchical clustering methods have also been developed for single cell data, which have the potential to handle large data set because of their scalability[153]. Except for the subtype structure, such hierarchical clustering methods could also potentially provide information about the phylogeny structure during the evolution of cancer[75]. The technical details of these clustering methods will be further discussed in the next section.

For these methods, dimension reduction and clustering analysis are carried out separately. However, evidence has been provided that joint learning could largely improve the performance of machine learning algorithms[154]. Fig.4.1 illustrates how joint dimension reduction and clustering works. One existing method which does this is called `DRjCC`[155] which combines PCA and NMF (detailed later). Another method called `iWMM`[156] also tries to address this problem, by using a mixture model for clustering, and a Gaussian process latent variable model for dimension reduction. However, the inference for this method is based on Markov Chain Monte Carlo(MCMC) which takes large number of iterations to converge[157]. In this chapter, I developed a new method `Curd` combining dimension reduction and clustering to analyze the latent low dimensional and subtype structure between single cells based on a nonparametric curvature model, which is substantially different from the linear model used in `DRjCC` and the Gaussian process model in `iWMM`. Though only single cell data are mentioned here, this method could be applied to more general problems where dimension reduction and clustering are required beyond the scope of cancer evolution research. I will illustrate this idea with examples in the results section.

4.1.1 Methodology

In this section, I describe the clustering and dimension reduction methods commonly used for single cell analysis. In the following part, I will denote each data point in the form of a vector as $\vec{x_j}$, where $j \in \{1, 2, 3...J\}$ and J is the total number of data points. As assumed, $\vec{x_j}$ will be vectors of moderately high dimension, usually on the scale of several thousand.

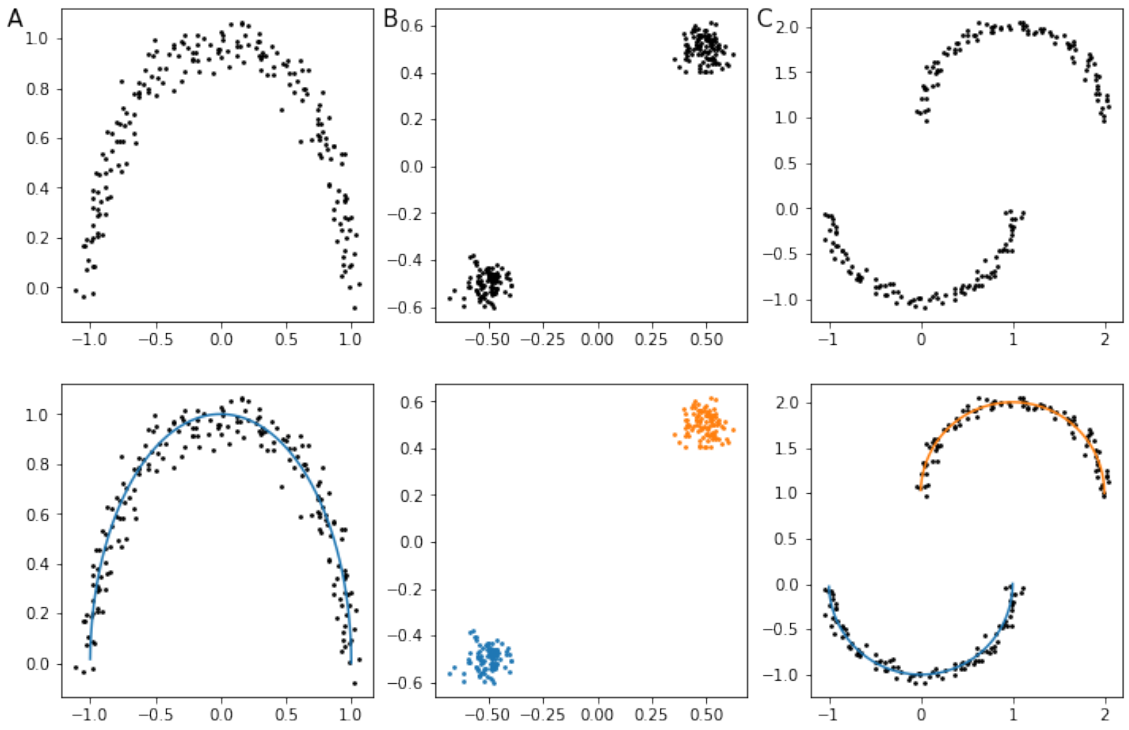


Figure 4.1.* Dimension reduction and clustering. *(A) Dimension reduction: the data is shown in the top panel. It contains 2 dimensions. Each dot indicates a sample/single cell. The dimension reduction is shown in the bottom panel that the 2 dimensional data could be represented by a 1 dimensional curve. (B) Clustering: the data is formed by two disjoint cluster of points. The clustering process is to find the sample/single cells belonging to each cluster as shown in the bottom panel. (C)Joint dimension reduction and clustering: the data consist of two disjoint clusters and each cluster could be represented by a 1 dimensional curve. The task is to find the sample/single cells belonging to each curve and find the representation of each curve.

4.1.1.1 Clustering

As mentioned in the introduction chapter, clustering methods are a class of unsupervised learning methods widely used in data analysis. Unlike classification problems, where data come with a label, in clustering, there is not any label for data. However, in clustering methods, it is assumed that there are certain patterns in the distribution of the data, $p(\vec{x_j})$. For some data points, $j \in C_1$, they are assumed to follow a regular and commonly used distribution $p_{C_1}(\vec{x_j})$, while for some other $j \in C_2$, they are assumed to follow another regular and commonly used distribution $p_{C_2}(\vec{x_j})$, where C_1 and C_2 are two disjoint subsets of the full data set $\{1, 2, 3...J\}$. It is assumed that there are K disjoint subsets denoted by C_k respectively. And finally they combine to form a complex distribution.

$$p(\vec{x}) = \sum_k w_k p_{C_k}(\vec{x}),$$
$$\sum_k w_k = 1,\ \forall k\ w_k > 0,$$

This is usually the assumption behind most clustering methods, where w_k is the probability that a data point comes from cluster C_k. Methods that use this definition of probability in the loss function and maximize the log-likelihood are called *mixture models*. For example, in a Gaussian mixture model[158] shown below, $\forall j \in C_k : \vec{x_j} \sim Normal(\vec{\mu_k}, \boldsymbol{\Sigma_k})$, where $Normal(\vec{\mu_k}, \boldsymbol{\Sigma_k})$ is a Gaussian/Normal distribution with mean $\vec{\mu_k}$ and covariance matrix $\boldsymbol{\Sigma_j}$. K-means clustering[159] obeys the same model, although the optimization method is different from the maximum likelihood estimation used for Gaussian mixture model. For a Gaussian mixture model, Expectation-Maximization[65] is used to find the maximum likelihood estimate and the algorithm is guaranteed to converge. However, for K-means clustering, a heuristic method which iteratively update the cluster assignment, i.e. the composition of C_k and the center for each cluster, i.e. $\vec{\mu_k}$ is used[159]. The biggest advantage of K-means and Gaussian mixture models are low time latency so that they are easily applied to large and high dimensional data sets[75].

$$p(\vec{x}) = \sum_k w_k p_{C_k}(\vec{x})$$
$$\forall j \in C_k : \vec{x_j} \sim Normal(\vec{\mu_k}, \boldsymbol{\Sigma_k})$$

Density based clustering methods[160] have also been developed for scRNA-seq data[161, 162]. In such methods, the cluster assignment of each data points is determined to be the mode, i.e. the local maximum, in density function $p(\vec{x})$ associated the data. Namely, data points close to the same mode of the distribution will be clustered together. In practice, though it is difficult to get the true density function $p(\vec{x})$, it can be approximated from the empirical distribution estimated from the data. The biggest advantage of density clustering is that it is very flexible with regard to the shape of the cluster, i.e. the data in the same cluster could be from some distribution that is not Gaussian[75].

A typical method of density based clustering is summarised below[160]. For each data point $\vec{x_j}$, it will be moved following the gradient of density function to the mode, i.e. the local maximum. As shown below, the trajectory that it is going to be moved is denoted by $t \in [0, \infty) : \tau_j(t)$. The starting point of the trajectory is $\tau_j(0) = \vec{x_j}$, and the direction that the data point moves is the gradient of the density function $\nabla p(\tau_j(t))$. The final destination of the trajectory is marked as $dest(\vec{x_j})$. For all points arrive at the same final destination $\forall j : dest(\vec{x_j}) = \vec{\mu_k}$ are put into the same cluster C_k.

$$\begin{aligned} \tau_j(0) &= \vec{x_j} \\ \nabla_t \tau_j(t) &= \nabla_{\tau_j(t)} p(\tau_j(t)) \\ dest(\vec{x_j}) &= \lim_{t \to \infty} \tau_j(t) \\ C_k &= \{j : dest(\vec{x_j}) = \vec{\mu_k}\} \end{aligned}$$

Another class of clustering methods widely used in scRNA-seq analysis is hierarchical clustering[163], where a clustering tree is generated. In hierarchical clustering different data points are assigned to the same cluster if the distance between them is small. Subsequently different clusters are combined to form a larger cluster if the distance between the clusters is small, until there is only a single cluster containing all data points. This single final cluster is seen as the root node of the tree. The data points are the leaf nodes and the clusters generated through this process are internal nodes of the tree. The algorithm works as Algo.4.1 shown below. Different hierarchical clustering algorithms use different distance metrics. For example, if both nodes have only one data point belonging to it, i.e. they are both leaf nodes, the distance between these two nodes will be the distance between these two data vectors, e.g. Euclidean distance, Manhattan distance. If there is more than one node in one of the cluster, i.e. this is an internal node, then the distance between two nodes could be either the minimum or maximum distance between all pairs of data points with one from each node. The reason why hierarchical clustering method is popular in biomedical research, including scRNA-seq analysis, can be summarised as follows: (1) it is fast to run; (2) it allows fitting flexible shapes of the clusters; (3) it enforces a hierarchical relationship among data points which is potentially

associated with the phylogeny structure[75].

Algorithm 4.1: Hierarchical Clustering

Input: data $\vec{x_j}$ as leaf nodes, number of clusters K, number of data points J,

Initialise: Count=J

while Count>K **do**

Compute the pairwise distance of all nodes

Merge two nodes with smallest distance into one internal node

All data points belonging to these two nodes now belong to the new node

Count=Count-1

end while

All data belong to the same node are assigned with the same cluster label

Another popular method called spectral clustering, will be discussed in the following section. Though it is a clustering method, it actually relies on spectral method for dimension reduction and then a clustering method, such as K-means mentioned above is used based on the features after dimension reduction[164].

Finally, with the recent development of deep neural networks, deep clustering methods have gained in popularity. Most deep clustering method are based on the work of autoencoder, which is a method for dimension reduction[165–167]. In autoencoder, there is an encoding function $g(\vec{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^q$ and a decoding function $f(\vec{x'}) : \mathbb{R}^q \rightarrow \mathbb{R}^d$. Typically, q is much smaller than d and both of these functions are deep neural networks. In autoencoder, it tries to minimize the reconstruction loss after encoding and decoding shown below. Thus the encoding part, $g(\vec{x})$ could be seen as a dimension reduction method. For deep clustering, it is attempted to do clustering based on the output of the encoder function $g(\vec{x})$. For example, it is possible to introduce K-means clustering and learn the K-means cluster assignment simultaneously with the autoencoder[166]. It is also possible to use a Gaussian mixture model on the output of pre-trained encoder network $g(\vec{x})$[167]. Deep clustering has considerable potential in analysing complex and high dimensional data, e.g. image data, whereas the training of deep clustering models typically requires longer time[167].

$$\min_{f(\vec{x'}),g(\vec{x})} \quad \|\vec{x} - f(g(\vec{x}))\|_2^2$$

4.1.1.2 Dimension reduction

Dimension reduction is also an unsupervised learning problem, where there is assumed to be some latent low dimensional structure in the distribution for the high dimensional data, e.g. temporal structure in scRNA-seq data.

The data is assumed to follow a distribution of the form:

$$\vec{x} \sim Normal(f(\vec{x'}), \boldsymbol{I})$$

Here, $\vec{x} \in \mathbb{R}^d$, $\vec{x'} \in \mathbb{R}^q$ and $\boldsymbol{I}$ is a d-dimensional identity matrix. It is assumed that $d \gg q$. The function $f(\vec{x'})$ is a mapping from $\mathbb{R}^q$ to $\mathbb{R}^d$, and it determines the low dimensional structure $\vec{x'}$ behind the distribution of high dimensional data $\vec{x}$. The problem for dimension reduction is to find out the corresponding low dimensional $\vec{x_j'}$ for each high dimensional data point $\vec{x_j}$.

In dimension reduction, one widely used method is *principal component analysis* (PCA), where $f(\vec{x'}) = \boldsymbol{A}\vec{x'} + \vec{b}$ is a linear function[76]. For PCA, maximum likelihood estimation is also used and the optimization problem can be solved by eigenvalue decomposition of the data covariance matrix $\boldsymbol{\Sigma}$ shown below. The first k eigenvectors of the covariance matrix $\boldsymbol{\Sigma}$ (ranked by eigenvalues from largest to smallest) are the first k principal components. (Note that there are different ways to formulate the PCA algorithm, e.g. by using the singular value decomposition[76, 77] and the way explained above is only an attempt to make it easier to be related to the novel method `Curd` developed later on.)

$$\boldsymbol{\Sigma} = \sum_j (\vec{x_j} - \vec{\mu})(\vec{x_j} - \vec{\mu})^T$$
$$\vec{\mu} = \frac{\sum_j \vec{x_j}}{J}$$

PCA is easy and fast to run, but as it is only a linear dimension reduction method, it has certain limitations when the relationship between data points become complex.

Thus nonlinear dimension reduction methods are also widely used in biomedical research.

In many classes of nonlinear dimensional reduction methods, it is attempted to find a mapping from $\vec{x}$ to $\vec{x'}$ directly, such that $\vec{x_j'} = g(\vec{x_j})$ the distance between $\vec{x_i'}$ and $\vec{x_j'}$ is similar to that of $\vec{x_i}$ and $\vec{x_j}$.

Multidimensional scaling[168], attempts to directly minimise the difference between the pairwise distance between data points shown below. This loss function measures the difference of the distance between data points in the original high dimensional space, i.e. $\|\vec{x_i} - \vec{x_j}\|$ and the distance in the new low dimensional space, i.e. $\|\vec{x_i'} - \vec{x_j'}\|$.

$$\sum_i \sum_j (\|\vec{x_i} - \vec{x_j}\| - \|\vec{x_i'} - \vec{x_j'}\|)^2$$

Similarly, in spectral methods[169], it is attempted to make the two points closer in lower dimension, if they are close in their original dimension. As shown below, $Kernel(\vec{x_i}, \vec{x_j})$ is a kernel function which measures the similarity (the reverse of distance) between two points $\vec{x_i}$ and $\vec{x_j}$ in the original space -the larger this number, the closer the two data points. Thus, through the optimisation problem shown below, for data points closer to each other, they should have smaller distance in lower dimensional representation as $\vec{x_i'}$ and $\vec{x_j'}$.

$$\min_{\vec{x_i'}, \vec{x_j'}} \quad \sum_i \frac{\sum_j Kernel(\vec{x_i}, \vec{x_j}) \|\vec{x_i'} - \vec{x_j'}\|^2}{\sum_j Kernel(\vec{x_i}, \vec{x_j})}$$

Both MDS and spectral method focus on the global structure of all the data points available and thus might be difficult to focus on the local data structure[168, 169]. Besides, it is not clear how to choose the proper kernel function for spectral method.

Similar to the idea of matching the pair wise distance between the original data points $\vec{x}$ and the low dimensional representation $\vec{x'}$, there are also methods trying to focus only on the local structure of the data, i.e. the relationship between closer data points, while ignoring the relationship between data points far away from each other in the original high dimensional space.

For example, in local linear embedding[170, 171], it is attempted to map each data points $\vec{x_j}$ to a low dimensional point $\vec{x_j'}$ such that they can be represented by the same linear combination of other adjacent points as shown below. Here, $\vec{x_i}$ and $\vec{x_j}$ are data points in close neighbourhood and w_{ij} is the weight computed based on these adjacent data points $\vec{x_i}$ of the data point $\vec{x_j}$. The first equation shown below is used to compute the weights w_{ij} (through linear regression), while the second one is used to compute the lower dimension representation $\vec{x_j'}$.

$$\vec{x_j} = \sum_{i \neq j} w_{ij} \vec{x_i}$$
$$\vec{x_j'} = \sum_{i \neq j} w_{ij} \vec{x_i'}$$

Compared to other nonlinear dimension reduction methods, the biggest advantage of local linear embedding is that it is quite robust. The results won't be influenced by the random initialisation too much.

Another class of methods called stochastic neighbour embedding (SNE)[172], also tries to achieve this goal of matching the distance of adjacent data points. In this method, the probability of $\vec{x_j}$ being generated by a Normal distribution centered at $\vec{x_i}$ as p_{ij} is computed first. Then, the probability of $\vec{x_j'}$ being generated by a Normal distribution centered at $\vec{x_i'}$ is denoted by p'_{ij}. The goal of this method is to minimise the KL divergence between these two probabilities as shown below.

$$p_{ij} = \frac{\exp(-\|\vec{x_i} - \vec{x_j}\|^2)}{\sum_j \exp(-\|\vec{x_i} - \vec{x_j}\|^2)}$$
$$p'_{ij} = \frac{\exp(-\|\vec{x_i'} - \vec{x_j'}\|^2)}{\sum_j \exp(-\|\vec{x_i'} - \vec{x_j'}\|^2)}$$
$$\min_{\vec{x'}} \quad \sum_i \sum_j -p_{ij} \log \frac{p'_{ij}}{p_{ij}}$$

For SNE, because of the distribution used, it is likely to put the distant points together, which is known as the crowding problem[173]. Thus, there is a modified version of SNE based on different distribution, i.e. t-SNE[173] which is based on student-t distribution and the p'_{ij} is modified as shown below. However, such choice

of student-t distribution is more of a numerical concern to solve the problem of crowding points rather than based on a statistical model. Therefore, a similar method called UMAP which is also widely used for scRNA-seq data analysis[174] has been proposed. It is based on similar idea as t-SNE but the probability p_{ij} is then computed in a different way guided by the theory of geometry[175].

$$p'_{ij} = \frac{(1 + \|\vec{x'_i} - \vec{x'_j}\|^2)^{-1}}{\sum_j (1 + \|\vec{x'_i} - \vec{x'_j}\|^2)^{-1}}$$

Among these methods derived from SNE, t-SNE is arguably the most widely used method in scRNA-seq analysis, mainly because it is most likely to truthfully recover the underlying structure among cell[176]. However, t-SNE often takes long time to run and it is sensitive to the initialisation value for the variables, compared to local linear embedding[176]. UMAP is a relatively new method (developed in 2019), but it achieved similar performance as t-SNE, and it is been shown to be slightly faster and the results are more reproducible than t-SNE in scRNA-seq data analysis[177].

The sub-field of pseudotime analysis in single cell analysis is a special type of dimension reduction where the underlying low dimension is $q = 1$[150]. (Note that not all pseudotime algorithms could be seen as dimensional reduction methods, e.g. tree-based [178] and graph-based [179] pseudotime algorithms.) `DPT` (diffusion pseudotime) is a tree-based pseudotime algorithm[180]. It uses a cell picked by the user as the root cell and compute the number of steps it takes to go from the root cell to another cell as the pseudotime for that time. In each step. the probability of going from one cell to another cell is defined similar to the one used in SNE method according to the gene expression of these two cells. Thus, this method is good at capturing non-linear relationship between the single cells. However, this method requires the single cells to be connected(two cells are connected if the probability of going from one cell to another cell is larger than 0). This requirement can't be satisfied if the cells contain quite different populations. Besides, the choice of the root cell is also a difficult task.

4.1.1.3 Joint dimension reduction and clustering

For the methods mentioned above, they only focus on one type of the latent structure, i.e. either subtype structure or low dimension structure. For some data, which will be shown in the result section, both subtype and low dimension structure might exist. Thus, joint dimension reduction and clustering will be required to study such data. Besides, as evidence shown, in machine learning algorithms, joint learning could potentially increase the performance of the models[154].

This is a less well-studied field for scRNA-seq analysis and the most relevant method here is `DRjCC`[155]. In this method, a non-convex optimisation is proposed as the target by combining the target function of PCA and the target function of non-negative matrix factorization (NMF) with a penalty term as shown below. $\boldsymbol{X}$ is the matrix containing all the gene expression data from scRNA-seq. In $\boldsymbol{X}$, each row indicates a gene and each column indicate a single cell. The PCA part, i.e. $\|\boldsymbol{X} - \boldsymbol{ZA}\|_2^2$ is responsible for dimension reduction while the NMF part, i.e. $\|\boldsymbol{A} - \boldsymbol{BF}\|_2^2$ is responsible for clustering. $\|\boldsymbol{A}\|_1$ indicates the sum of the absolute value of all entries in matrix $\boldsymbol{A}$ and ρ_1, ρ_2 are hyper-parameters for penalty terms preventing the model from overfitting. Though NMF is not traditionally used for clustering, it is used heuristically here mainly because of the computational concern. The authors of this method chose the the row number with the largest value in each column of matrix $\boldsymbol{F}$ to be the cluster of the single cell associated with that column.

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{A},\boldsymbol{B},\boldsymbol{F}} \quad & \|\boldsymbol{X} - \boldsymbol{ZA}\|_2^2 + \|\boldsymbol{A} - \boldsymbol{BF}\|_2^2 \\ & + \rho_1\|\boldsymbol{A}\|_1 + \rho_2 tr(\boldsymbol{FLF}^T) \\ \text{s.t.} \quad & \boldsymbol{B} \geq \boldsymbol{0}, \boldsymbol{F} \geq \boldsymbol{0} \end{aligned}$$

This method however has certain limitations: (1) it is still using a linear method for dimension reduction; (2) the clustering part is also a linear function which doesn't have a statistical model supporting its utility in clustering; (3) it is not clear how to choose the number of clusters for it.

Besides, there is also a method called `iWMM` aimed at the task[156] from the machine learning community. The method `iWMM` is built on the work of Gaussian Process

Latent Variable Model(GPLVM)[181]. In GPLVM all data points $\vec{x}$ are combined into a data matrix $\boldsymbol{X}$, i.e. each row of $\boldsymbol{X}$ represents a data point. The goal is to find the lower dimension representations $\vec{x'}$ for all data points that maximise the loss function shown below, where Σ_{ij} is the ith row and jth column of the covariance matrix $\boldsymbol{\Sigma}$. This loss function is proportional to the likelihood function using Gaussian Process model[182]. In GPLVM, $\vec{x'_j}$ is called the latent variable and the distribution of it is assumed to be a standard Gaussian distribution, i.e. $p(\vec{x'_j}) = Gaussian(\vec{0}, \boldsymbol{I})$. Namely, the distribution for the variables $\vec{x_j}$ are assumed to follow Gaussian distribution with the covariance determined by the latent variables $\vec{x'_j}$.

$$\begin{aligned} \max_{\vec{x'}} \quad & |\boldsymbol{\Sigma}|^{-\frac{d}{2}} \exp(-\boldsymbol{X}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{X}) \prod_j p(\vec{x'_j}) \\ \text{s.t.} \quad & \Sigma_{ij} = \exp(\|\vec{x'_i} - \vec{x'_j}\|_x^2) \end{aligned}$$

In the model of `iWMM`, the GPLVM is mostly unchanged and this part is responsible for dimension reduction. To take clustering into consideration, the latent variables $\vec{x'}$ are assumed to follow a mixture model as mentioned above.

$$\begin{aligned} p(\vec{x'}) &= \sum_k w_k p_{C_k}(\vec{x}) \\ p_{C_k}(\vec{x}) &= Normal(\vec{\mu_k}, \boldsymbol{\Sigma_k}) \end{aligned}$$

The biggest advantage of `iWMM` is that it has a sound statistical model to support nonlinear dimension reduction. However, because the inference is carried out through MCMC method, it takes many number of iterations for the algorithm to converge. Besides, `iWMM` is not an ideal method for dimension reduction or clustering for further supervised learning method. As for `iWMM`, on the test data set, it is not clear how to map the new data points to the low dimensional embedding.

Therefore, I have developed my own method for joint dimension reduction and clustering called `Curd` based on curvature penalisation.

In my method, it is restricted that the dimension after reduction is only 1, which is

similar to many pseudotime algorithms[150]. Thus, in stead of using $\vec{x'}$ for notation, I use a one dimension variable $t \in \mathbb{R}$. This variable t is also called pseudotime, because it could potentially capture the temporal latent structure in the data $\vec{x}$. In this case, the mapping from t to data points $\vec{x}$, i.e. $\vec{x_j} = f(t_j)$, is called a curve mathematically.

4.1.2 My contribution

- My method simultaneously performs dimension reduction and clustering
- My method is essentially nonparametric with the Frenet Frame form, so that it is more flexible than many parametric models, including linear models
- My method gives an explicit representation of the function $f(t)$ so that for new data points, it is easy to find the low dimension embedding for them
- My method uses the concept of curvature from differential geometry to prevent overfitting
- My method has an interesting interpretation as a physical model
- I provide an alternative way to do model selection (choosing the number of clusters) for the method

4.2 Motivation of the method

The problem of joint dimension reduction and clustering can be generalized as fitting multiple curves $f_k(t), k \in \{1, 2, ..., K\}$ (1 dimensional manifold) to high dimensional data $\vec{x}$. Each data point $\vec{x_j},\ j \in \{1, 2, ..., J\}$ comes from a particular function $f_k(t)$ where $j \in C_k$ and C_k is a cluster/subset of the original data, i.e. $C_k \subset \{1, 2, ..., J\}$. For each data point, there is also a pseudotime t_j associated with it, i.e. $\vec{x_j} \approx f_k(t_j)$. Therefore the problem can be converted to a regression problem as below if both C_k and t_j are known.

$$\min_{f_k(t)} \quad \sum_k \sum_{j \in C_k} \|\vec{x_j} - f_k(t_j)\|_2^2$$

However, the difficulty of such problem comes from the fact that the class allocation C_k and low dimensional representation t_j are unobserved. From pure regression point of view, these variables could be learnt from the model as below.

$$\min_{f_k(t)} \quad \sum_k \min_{C_k} \sum_{j \in C_k} \min_{t_j} \|\vec{x_j} - f_k(t_j)\|_2^2 \tag{4.1}$$

Unfortunately, this function is hard to optimize because of the two inner minimization functions introduced. A relaxation to this problem is to change the minimisation to softmin, where softmin is defined as below and $y_i \in \mathbb{R}$.

$$\text{softmin}\left(\{y_i\}_{i\in\{1,2,\ldots,N\}}\right) = -\log\left(\sum_i e^{-y_i}\right)$$

It is easy to check that this softmin will converge to the minimisation step if the differences between different y_i gets larger. The use of softmin to approximate the minimization is similar to the idea of using probabilistic cluster allocation in Gaussian mixture model[158].

Then, by using this approximation, the problem of joint dimension reduction and clustering can be converted to the following form.

$$\min_{f_k(t)} \sum_j -\log\left(\sum_k \int e^{-\|\vec{x_j} - f_k(t)\|_2^2} \mathrm{d}t\right) \tag{4.2}$$

Similar to all other pseudotime algorithms for scRNA-seq, the pseudotime here is identifiable up to monotonic transformation.

4.2.1 Simple case study: linear model

In order to see what the model is doing, it is important to understand the linear cases, where $f_k(t) = \vec{\alpha_k}t + \vec{\beta_k}$. It is worthy noting that any monotonic change of variable $h(t)$ and $f_k'(t) = f_k(h(t))$ will represent the same curve (with different velocity, physically speaking). Thus it doesn't lose any generality to introduce a constraint $\|\vec{\alpha_k}\|_2 = 1$. If this constraint is plugged into (4.2), it could be transformed into the following problem.

$$\begin{aligned}\min_{\vec{\alpha_k},\vec{\beta_k}} \quad & \sum_j -\log\left(\sum_k \int e^{-\|\vec{x_j}-\vec{\alpha_k}t-\vec{\beta_k}\|_2^2}\mathrm{d}t\right) \\ =& \sum_j -\log\left(\sum_k e^{-\|\vec{\beta_k}-\vec{x_j}\|_2^2+[\vec{\alpha_k}^T(\vec{\beta_k}-\vec{x_j})]^2}\int e^{-(t-\vec{\alpha_k}^T\vec{x_j}+\vec{\alpha_k}^T\vec{\beta_k})^2}\mathrm{d}t\right) \\ \propto& \sum_j -\log\left(\sum_k e^{-\|\vec{\beta_k}-\vec{x_j}\|_2^2+[\vec{\alpha_k}^T(\vec{\beta_k}-\vec{x_j})]^2}\right)\end{aligned}$$

Then, this optimization problem could be solved by the Expectation-Maximization algorithm[65]. The E-step is to solve the following optimization problem with introduced auxiliary variable z_j^k as shown below.

$$\begin{aligned}\min_{z_j^k} \quad & -\sum_j\sum_k \left[z_j^k \log y_j^k - z_j^k \log(z_j^k)\right] \\ s.t. \quad & z_j^k \geq 0, \sum_k z_j^k = 1 \\ & y_j^k = e^{-\|\vec{\beta_k}-\vec{x_j}\|_2^2+[\vec{\alpha_k}^T(\vec{\beta_k}-\vec{x_j})]^2}\end{aligned}$$

This implied the update step for z_j^k as follows.

$$z_j^k = \frac{y_j^k}{\sum_k y_j^k}$$

In M-step, it is left to solve the following problem with z_j^k fixed for each curve k.

$$\min_{\vec{\alpha_k},\vec{\beta_k}} \quad \sum_j z_j^k\|\vec{\beta_k}-\vec{x_j}\|_2^2 - z_j^k[\vec{\alpha_k}^T(\vec{\beta_k}-\vec{x_j})]^2$$

The optimisation related to $\vec{\beta_k}$ can be done through the following step, where c is any constant.

$$\begin{aligned}\nabla_{\vec{\beta_k}} =& (I-\vec{\alpha_k}\vec{\alpha_k}^T)[(\sum_j z_j^k)\vec{\beta_k} - (\sum_j z_j^k\vec{x_j})] = \vec{0} \\ \implies \vec{\beta_k} =& \frac{\sum_j z_j^k\vec{x_j}}{\sum_j z_j^k} + c\vec{\alpha_k}\end{aligned}$$

This means that the intercept $\vec{\beta_k}$ could be chosen arbitrarily along the direction of

$\vec{\alpha_k}$. However, since the optimal value for $\vec{\alpha_k}$ is not known at the moment, it would suffice to set $c = 0$, and $\vec{\beta_k}$ becomes the weighted mean of all data points $\vec{x_j}$.

Then, by fixing the values of $\vec{\beta_k}$, it is possible to get the optimal value for $\vec{\alpha_k}$ through the following optimisation.

$$\begin{aligned} \min_{\vec{\alpha_k}} \quad & -\vec{\alpha_k}^T \boldsymbol{\Sigma_k} \vec{\alpha_k} \\ \text{s.t.} \quad & \|\vec{\alpha_k}\|_2 = 1 \\ & \boldsymbol{\Sigma_k} = \sum_j z_j^k (\vec{x_j} - \vec{\beta_k})(\vec{x_j} - \vec{\beta_k})^T \end{aligned}$$

This implies that $\vec{\alpha_k}$ is the eigenvector with respect to the largest eigenvalue of the weighted covariance matrix $\boldsymbol{\Sigma_k}$.

These three steps form a coordinate descent method for the linear case. It could be learnt from this algorithm that: (1) if z_j^k is known beforehand, then it becomes PCA model[76]; (2) if $\vec{\alpha_k} = \vec{0}$, it becomes a Gaussian mixture model where z_j^k is the cluster allocation[158]. Therefore, the optimisation for z_j^k corresponds to clustering and the optimisation for $\vec{\alpha_k}$ and $\vec{\beta_k}$ becomes dimension reduction with PCA method. Thus, this model provides a way to do clustering and dimension reduction jointly.

4.2.2 Two step EM algorithm: towards a nonlinear model

Going from the linear model to non linear models, a two step Expectation-Maximisation described below is used. By introducing two auxiliary variables z_j^k and $p_j^k(t)$ to the optimisation problem (4.2), it could be converted to the following form.

$$\min_{f_k(t), z_j^k, p_j^k(t)} \quad \sum_j \sum_k -z_j^k \int \left[p_j^k(t)(-\|\vec{x_j} - f_k(t)\|_2^2) - p_j^k(t) \log(p_j^k(t)) \right] \mathrm{d}t + z_j^k \log(z_j^k) \tag{4.3}$$

$$s.t. \quad z_j^k \geq 0, \ \sum_k z_j^k = 1, \ p_j^k(t) \geq 0, \ \int p_j^k(t) \mathrm{d}t = 1$$

Then, two expectation steps become as follows.

$$p_j^k(t) = \frac{e^{-\|\vec{x_j} - f_k(t)\|_2^2}}{\int e^{-\|\vec{x_j} - f_k(t)\|_2^2} \mathrm{d}t} \tag{4.4}$$

$$z_j^k = \frac{\int e^{-\|\vec{x_j} - f_k(t)\|_2^2} \mathrm{d}t}{\sum_k \int e^{-\|\vec{x_j} - f_k(t)\|_2^2} \mathrm{d}t} \tag{4.5}$$

With these two E-steps introduced, the optimization becomes a regression problem as below, and then a proper regression model for $f_k(t)$ used, e.g. polynomial regression [31].

$$\min_{f_k(t)} \quad \sum_j \int z_j^k p_j^k(t) \|\vec{x_j} - f_k(t)\|_2^2 \mathrm{d}t$$

Here, it is possible to fit each curve $f_k(t)$ individually. And this is exactly what I did for my method `Curd`.

4.3 Curvature regularized minimization

From now on, I will formally introduce my new method for joint dimension reduction and clustering called `Curd`.

4.3.1 Dimension reduction with curvature regularization

To recapitulate, for data points $\vec{x_j} \in \mathbb{R}^d$, $j \in \{1, 2, 3..J\}$, the dimension reduction process is shown as the optimization problem (4.6). Here, $t_j \in \mathbb{R}$ is the one dimensional representation for each x_j and $f(t) : \mathbb{R} \to \mathbb{R}^d$ is any arbitrary function. $\alpha(t)$ and $\beta(t)$ is the tangent and normal(curvature) vector of the curve $f(t)$. Thus, the curve $f(t)$ is fully determined by the initial points $\vec{f^0}$ and tangent functions $\alpha(t)$.

$$\begin{aligned} \min_{f(t),\alpha(t),\beta(t),t_j} \quad & \sum_j \|\vec{x_j} - f(t_j)\|_2^2 + \rho \int \|\beta(t)\|_2^2 dt \\ \text{s.t.} \quad & f(t) = \vec{f^0} + \int_0^t \alpha(t')dt' \\ & \alpha(t) = \vec{\alpha^0} + \int_0^t \beta(t')dt' \\ & \alpha(t)^T \beta(t) = 0 \\ & \|\vec{\alpha^0}\|_2 = 1 \end{aligned} \tag{4.6}$$

Besides, there happens to be a physical interpretation to the model (4.6), which may assist in understanding the math formulas used. Remind that the elastic potential energy of a spring is defined as below, where κ_1 is some constant coefficient and E_1, E_2 are two end points of the spring: $U_1 = \frac{1}{2}\kappa_1\|E_1 - E_2\|_2^2$.

The potential energy of a elastic rod is defined similarly where $\beta(t)$ is the curvature of the rob at point t and κ_2 is some constant coefficient:$U_2 = \frac{1}{2}\int_t \kappa_2\|\beta(t)\|_2^2 dt$.

Then the whole model (4.6), where $\rho = \frac{\kappa_2}{\kappa_1}$, could be seen as a system with an elastic rod $f(t)$ and a few points $\vec{x_j}$ attached to springs in space $\mathbb{R}^d$. There are J springs where one end of them are attached to $\vec{x_j}$ and the other end has been attached to $f(t_j)$. There is no external force to the system.

The solution to the model is the stable state of the system. Namely, it is the state when the total energy is minimized. And thus, the solution is the same as the solution to the boundary value problem (BVP) shown below. the Lagrange dual variable $F(t)$ could be explained as the internal force along the rod and the boundary condition (4.7) are indicating that there is no external force so that the force on the two ends of the rod should equal 0.

Theorem 4.1. *The solution to the problem (4.6) could be computed by the following BVP, where $F(t)$ is the Lagrange dual variable.*

$$
\begin{aligned}
&\nabla_t f(t) = \alpha(t) \\
&\nabla_t \alpha(t) = 1/\rho[\boldsymbol{I} - \alpha(t)\alpha(t)^T]F(t) \\
&\nabla_t F(t) = 1/\rho[(\boldsymbol{I} - \alpha(t)\alpha(t)^T)F(t)F(t)^T\alpha(t)] \\
&\qquad + \sum_{j:t_j<t} \vec{x_j} - f(t_j) \\
&\textit{Boundary Condition:} \quad F(0) = 0,\ F(T) = 0 \qquad (4.7) \\
&\sum_j \vec{x_j} - f(t_j) = 0 \\
&\textit{Parameter Condition:} \quad \alpha(t_j)^T[\vec{x_j} - f(t_j)] = 0 \qquad (4.8)
\end{aligned}
$$

The minimization with respect to t_j here could be seen as the attached point $f(t_j)$ could slide along the rod. Thus, at the balanced state, the force $\vec{x_j} - f(t_j)$ should

be perpendicular to the tangent of the rod $\alpha(t_j)$ as in (4.8). The proof for this BVP is attached in the appendix section.

4.3.2 Multiple curves

In the work of `Curd`, multiple curves are considered jointly. Namely, for each data point $\vec{x_j}$, it is necessary to assign it to a single curve $f_k(t)$, $k \in \{1, 2, ..., K\}$. And for all points assigned to the same curve $f_k(t)$, I applied the dimension reduction as shown in (4.9) with the same constraint as (4.6). Besides, as t_j is not known, they are the variable to be minimised.

$$\min_{f_k(t), t_j, C_k} \quad \sum_k \min_{C_k} \sum_{j \in C_k} \min_{t_j} \|\vec{x_j} - f_k(t_j)\|_2^2 + \rho \int \|\beta_k(t)\|_2^2 dt \tag{4.9}$$

4.3.3 Softmin approximation

The original optimization problem (4.9) is essentially the problem (4.1) with a penalty term added. Thus, the softmin approximation could be used, and the problem (4.9) could be converted to the following form.

$$\begin{aligned} \min_{f_k(t)} \quad & -\sum_j \log \sum_k \int \exp\left(-\|\vec{x_j} - f_k(t)\|_2^2\right) dt \\ & + \rho \int \|\beta_k(t)\|_2^2 dt \end{aligned} \tag{4.10}$$

4.3.4 Optimisation

From the constraint for original problem (4.6), it is necessary to find the optimal value of the starting location $\vec{f_k^0}$, and the tangents $\alpha_k(t)$ of every curve $f_k(t)$.

4.3.4.1 EM algorithm

The first step to simplify problem (4.10), expectation-maximization method is used by introducing auxiliary variable $w_{jk}(t)$. Notice that, this is essentially combining the steps (4.4) and (4.5) mentioned before, i.e. $w_{jk}(t) = z_j^k p_j^k(t)$.

$$w_{jk}(t) = \frac{\exp(-\|\vec{x_j} - f_k(t)\|_2^2)}{\sum_k \int \exp(-\|\vec{x_j} - f_k(t')\|_2^2) dt'} \tag{4.11}$$

Then, the optimization is simplified into the following form.

$$\min_{f_k(t)} \quad \sum_{i,j} \int w_{jk}(t) \|\vec{x_j} - f_k(t)\|_2^2 dt + \rho \int \|\beta_j(t)\|_2^2 dt \tag{4.12}$$

The optimization of f_k^0 with all other variables fixed is simply an unconstrained quadratic programming and could be done as below.

$$\vec{f_k^0} = \sum_j \int w_{jk}(t)[\vec{x_j} - \int^t \alpha_k(t')dt']dt \tag{4.13}$$

4.3.4.2 Finite difference approximation

To compute the integral in (4.12), a finite difference method is applied. The total number of interval is denoted by I and the finite difference $\Delta t = t_{i+1} - t_i$.

$$\begin{aligned} \min_{f_k(t)} \quad & \sum_{i,j,k} \sum_{i'>i} w_{jk}(t_{i'}) \|\vec{x_j} - f_k^{-i}(t_{i'}) - \alpha_k(t_i)\Delta t\|_2^2 \\ & + \rho \sum_i \|\frac{\alpha_k(t_i) - \alpha_k(t_{i-1})}{\Delta t}\|_2^2 \Delta t \end{aligned} \tag{4.14}$$

$$\begin{aligned} \text{where} \quad & f_k(t_{i'}) = f_k^{-i}(t_{i'}) + \alpha_k(t_i)\Delta t \\ & \|\alpha_k(t_i)\|_2 = 1 \end{aligned} \tag{4.15}$$

Then, the optimization with respect to each $\alpha_k(t_i)$ in (4.14) becomes a quadratic programming with constraint (4.15).

To optimise the tangents $\alpha_k(t_i)$, it may be noticed that the norm of all tangents have not been changed and therefore, every update step is equivalent to multiply the old value of tangent vector by an orthonormal matrix. Namely, it is possible to set all $\alpha_k(t_{i'}) = \boldsymbol{P_{ki}}\alpha_k(t_{i'}),\ \forall i' \geq i$, where $\boldsymbol{P_{ki}}^T \boldsymbol{P_{ki}} = \boldsymbol{I}$. Then, the optimization problem becomes essentially the following.

$$\begin{aligned}
\min_{\boldsymbol{P_{ki}}} \quad & \sum_{i'>i,j} w_{jk}(t_{i'})\|\vec{x_j} - f_k^{i-}(t_{i'}) - \boldsymbol{P_{ki}} f_k^{i+}(t_{i'})\Delta t\|_2^2 \\
& + \rho\|\frac{\boldsymbol{P_{ki}}\alpha_k(t_i) - \alpha_k(t_{i-1})}{\Delta t}\|_2^2 \Delta t \\
& = tr(\boldsymbol{P_{ki}}\boldsymbol{X_{ki}})
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
\text{where} \quad & f_k^{i-}(t_{i'}) = \vec{f_k^0} + \sum_{i''=0}^{i-1} \alpha_k(t_{i''})\Delta t \\
& f_k^{i+}(t_{i'}) = \sum_{i''=i}^{i'-1} \alpha_k(t_{i''})\Delta t \\
& \boldsymbol{X_{ki}} = \sum_{i'>i,j} w_{jk}(t_{i'}) f_k^{i+}(t_{i'})[\vec{x_j} - f_k^{i-}(t_{i'})]^T \Delta t \\
& \qquad + \frac{\rho}{\Delta t}\alpha_k(t_i)\alpha_k(t_{i-1})^T
\end{aligned} \tag{4.17}$$

As all orthonormal matrix could be seen as a product of several Householder matrices, $\boldsymbol{H} = \boldsymbol{I} - 2\frac{\vec{n}\vec{n}^T}{\|\vec{n}\|_2^2}$, each time the value could be updated recursively by $\boldsymbol{P'_{ki}} = \boldsymbol{H_{ki}}\boldsymbol{P_{ki}}$. Then the problem (4.16) becomes (4.18). This indicates the vector $\vec{n_{ki}}$ for Householder matrix should be simply the largest eigenvector of matrix $\boldsymbol{P_{ki}}\boldsymbol{X_{ki}}$. And thus, the algorithm for computing the largest eigenvalue could be applied[183].

$$\begin{aligned}
\min \quad & -\vec{n_{ki}}^T \boldsymbol{P_{ki}}\boldsymbol{X_{ki}}\vec{n_{ki}} \\
\text{s.t.} \quad & \|\vec{n_{ki}}\|_2 = 1
\end{aligned} \tag{4.18}$$

This algorithm for optimising $\alpha_k(t)$ has been shown in Algorithm.4.2. The derivation of the steps as well as the proof of convergence for this algorithm is included in the appendix. The main idea here is to guarantee that the loss does not go up in each update step, i.e. making it a loss descent method for optimisation.

Note that without using the idea of Householder matrix, it is still possible to derive an update rule for tangent vectors on an one-by-one basis. However, this algorithm will converge slowly in practice. This update rule is discussed in the appendix of this chapter.

Algorithm 4.2: Update Curve Tangents for a particular curve $\alpha_k(t)$

Input: data $\vec{x_j}$, regularization ρ, auxiliary weight $w_{jk}(t_i)$, old curve tangents $\alpha_k(t_i)$, finite element size Δt
for $i = 1$ **to** I **do**
  Compute $\boldsymbol{X_{ki}}$ according to (4.17).
  Select a random vector $\vec{n}$ with norm 1.
  Compute Householder matrix $\boldsymbol{H_{ki}} = \boldsymbol{I} - 2\vec{n}\vec{n}^T$.
  Modify $\boldsymbol{X'_{ki}} = \boldsymbol{H_{ki}}\boldsymbol{X_{ki}}$
  Make it symmetric and positive semi-definite $\boldsymbol{X'_{ki}} = \frac{\boldsymbol{X'_{ki}} + \boldsymbol{X'_{ki}}^T + 2\|\boldsymbol{X'_{ki}}\|\boldsymbol{I}}{2+2\|\boldsymbol{X'_{ki}}\|}$
  Compute $\vec{y} = \boldsymbol{X'_{ki}}\vec{n}$
  Normalize $\vec{n'} = \frac{\vec{y}}{\|\vec{y}\|_2}$
  Compute Householder matrix $\boldsymbol{H'_{ki}} = \boldsymbol{I} - 2\vec{n'}\vec{n'}^T$
  Update all $\alpha_k(t_{i'}) = \boldsymbol{H'_{ki}}\boldsymbol{H_{ki}}\alpha_k(t_{i'})$ with $i' \geq i$.
end for

4.3.5 Model selection

Here, I introduce a method of choosing the number of curves, i.e. K, for `Curd`.

A popular method for choosing the number of clusters is group LASSO, where a L2 regularization is put on the parameter for each group and drives them to zero[184]. However, it could be checked that in the problem of (3.5), because of the constraint $\sum_k z_j^k = 1$, the group LASSO penalty $\lambda\sqrt{\sum_j (z_j^k)^2}$ will not have such effect. For example, if curve k and curve k' are quite similar, which means one curve is redundant in this setting, then $z_j^k = 1$ and $z_i^{k'} = 0$ will give the same loss as $z_j^k = 0.5$ and $z_i^{k'} = 0.5$ for group LASSO. This means that the group LASSO model has no preference to choose to use only one curve.

This is because the conflicting force between group LASSO and sum to 1 constraint as mentioned. Thus, it is desirable to have some penalty that has little penalty effect when the z_j^k is large while has large effect when it is close to zero. Statistically speaking, it is necessary to have a prior distribution that concentrates at zero but also has a long tail. This new prior distribution/penalty of choice is shown below.

$$\begin{aligned} \min_{z_j^k} \quad & \sum_j \sum_k -z_j^k \log \frac{e^{-d_j^k}}{z_j^k} - \log\left(1 + e^{-\sqrt{\sum_j (z_j^k)^2}}\right) \\ \text{where} \quad & d_j^k = \int \|\vec{x_j} - f_k(t)\|_2^2 dt \end{aligned} \tag{4.19}$$

It could then be shown this prior is similar to group LASSO. However, It could be checked that when $\sqrt{\sum_j (z_j^k)^2}$ is small, the penalty term λ^k will be large, while when $\sqrt{\sum_j (z_j^k)^2}$ is large, the penalty term λ^k will be small. Thus, it has the preference to drive the curve with fewer data points associated with it to be unused, i.e. driving z_j^k to 0 for all j. (Note that, when optimizing z_j^k, λ^k is considered to be a fixed constant.)

$$\begin{aligned} \min_{z_j^k} \quad & \sum_j \sum_k -z_j^k \log \frac{e^{-d_j^k}}{z_j^k} + \lambda^k \sqrt{\sum_j (z_j^k)^2} \\ \text{where} \quad & \lambda^k = \frac{e^{-\sqrt{\sum_j (z_j^k)^2}}}{1 + e^{-\sqrt{\sum_j (z_j^k)^2}}} \end{aligned} \tag{4.20}$$

The technique used to derive the optimisation target shown above is detailed below by introducing an auxiliary variable λ^k.

$$\begin{aligned} & -\log\left(1 + e^{-\sqrt{\sum_j (z_j^k)^2}}\right) \\ = & -\log\left((1-\lambda^k)\frac{1}{1-\lambda^k} + \lambda^k \frac{e^{-\sqrt{\sum_j (z_j^k)^2}}}{\lambda^k}\right) \\ \leq & \ (1-\lambda^k)\log(1-\lambda^k) + \lambda^k \log \lambda^k + \lambda^k \sqrt{\sum_j (z_j^k)^2} \end{aligned} \tag{4.21}$$

The equality for (4.21) holds when $\lambda^k = \frac{e^{-\sqrt{\sum_j (z_j^k)^2}}}{1+e^{-\sqrt{\sum_j (z_j^k)^2}}}$.

It could then be checked that, by using the new penalty (4.19), in the previous example when curve k and k' share similar parameters, setting $z_j^k = 1$ and $z_j^{k'} = 0$, will give a smaller penalty $-\log(2 + 2e^{-\sqrt{J}})$ than the case setting $z_j^k = z_j^{k'} = 0.5$, where the penalty is $-2\log(1 + e^{-\sqrt{J}/2})$ as below.

$$\begin{aligned} & e^{-\sqrt{J}} > 0 \\ \implies & (1 + e^{-\sqrt{J}}) > 2e^{-\sqrt{J}/2} \\ \implies & 2(1 + e^{-\sqrt{J}}) > (1 + e^{-\sqrt{J}/2})^2 \\ \implies & -\log(2 + 2e^{-\sqrt{J}}) < -2\log(1 + e^{-\sqrt{J}/2}) \end{aligned}$$

Algorithm 4.3: `Curd`

Input: data $\vec{x_j}$, regularization ρ, finite difference size Δt, number of curves K
Randomly initialize: auxiliary weight $w_{jk}(t_i)$, curve tangents $\vec{\alpha_k}(t_i)$, curve starting point $\vec{f_k^0}$
repeat
  Compute $w_{jk}(t_i)$ according to (4.11)
  for $k = 1$ **to** K **do**
    Update curve starting point $\vec{f_k^0}$ according to (4.13)
    Update curve tangents using Algo.4.2.
  end for
until Convergence: The decrease in loss function (4.10) is smaller than some threshold

This indicates that the new penalty will drive the model to select only one curve when these two curves share similar parameters.

4.3.6 Pseudocode

The pseudo code for the whole training process is shown in Algo.4.3.

The most important hyper parameter for `Curd` is the regularization ρ in Algo.4.3. This hyper parameter is generally chosen by cross validation and practically, as long as it is on the same scale, the result doesn't change much. However, because this optimisation problem is highly nonconvex, there are a lot of local optimal solutions. Thus, it is necessary to try multiple random initialisation and choose the one which gives the minimum loss.

4.4 Results

In the results, I am going to first start with some examples of simulated data and then move to real-world data.

The simulated data are based on previous literature, where they have been used qualitatively to check if the dimension reduction methods could successfully unravel the underlying latent structure or not[185–187]. These simulated data have 2 or 3 dimensions, so it is quite straight forward to have a visual understanding of how `Curd` fit the curves through the data points.

Then, I tested the method on a COVID data set which comes with features that could be used as ground truth for clustering and dimension reduction tasks. Thus, this data set is ideal to test the performance of the method.

Finally, I tested the method on a scRNA-seq data to show its performance and on anther scRNA-seq to showcase one possible application, i.e. finding differentially expressed genes, of such method for scRNA-seq data analysis.

4.4.1 Simple data examples

To test the performance of my method, I simulated some simple data examples. These examples are all two dimensional data, which makes it possible to have a clear illustration about how the curves learned by my method fit into the original data.

Here, the first example are two separate half circles. The radius of the circles are both 1 and the noise are simulated to follow normal distributions with 0 mean and variance equal 0.05 in both dimensions. This is a very classic example used in dimension reduction to illustrate the utility of the methods[185]. Then, in the following 3 cases, I make the data set more complex.

In the second case, I have changed one half circle to a straight line of length 2 with the same noise, which goes across the remaining half circle.

In the third one, I kept two circles but moved their position so that they cross each other on the two tails, which makes it a bit more complex than the first case.

And finally, in the last example, I introduced an extra half circle, and two of them are crossing each other and the last one is separate from the other two. This is actually the combination of case 1 and case 3.

It showed (Fig.4.2) that my method succeeded with all these simple examples.

4.4.2 Spiral data

I then simulated more complex spiral data, which is also used previously to check the performance of dimension reduction tools[186]. The data are simulated according

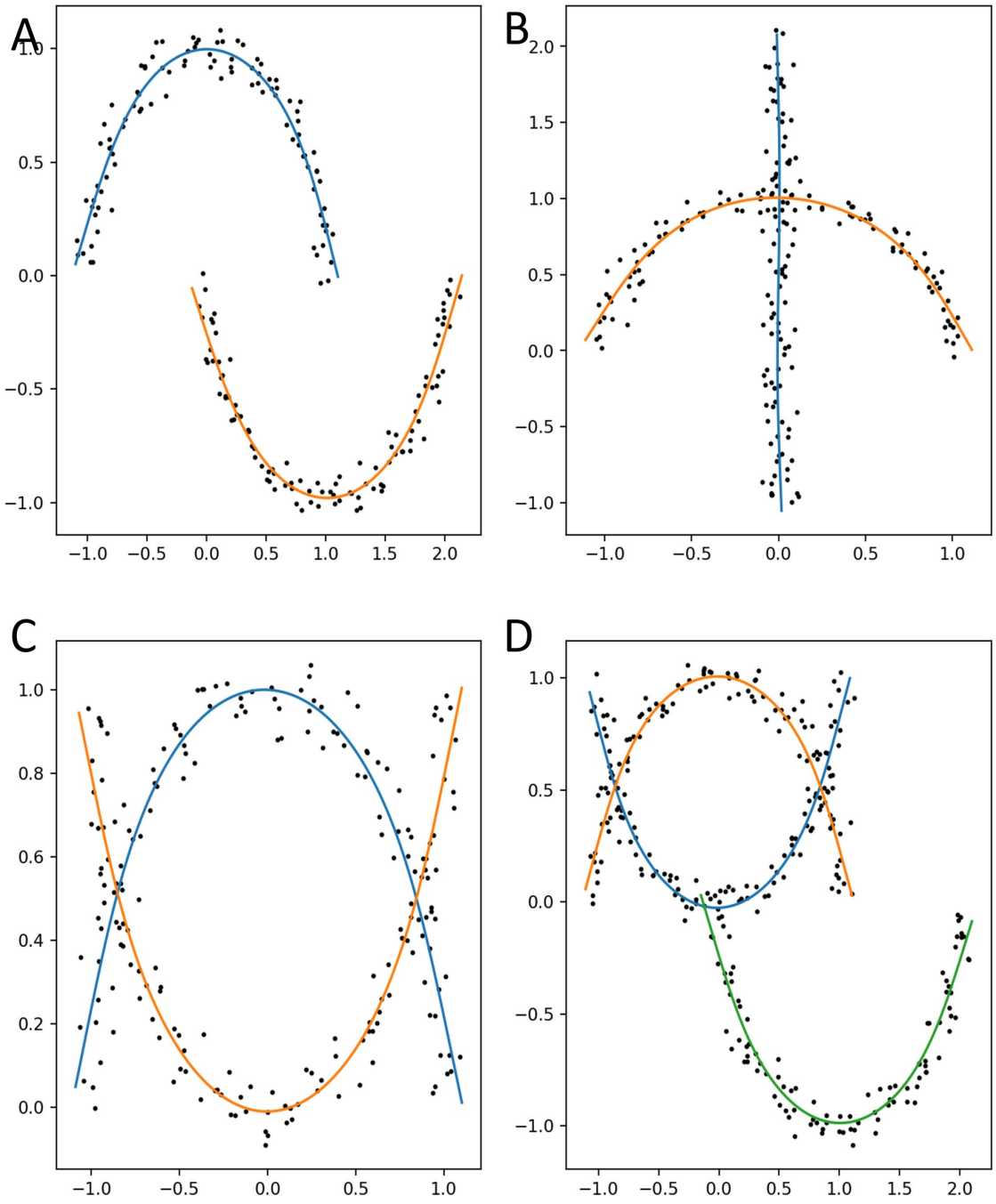


Figure 4.2.* Simple example results.** ***This figure shows the fitted curves (colored lines) on simulated data (black dots). Half circles are simulated to have radius of 1 and the noise is set to follow a normal distribution with mean 0 and variance 0.05. The algorithms manage to infer the correct underlying curves in all cases with 0,1,2 and 3 intersection points.

to the spiral functions $r = r_0 + vt$ and $\theta = \theta_0 + \omega t$ in the polar coordinate system. And then it is converted back to x-y coordinate by $x = r\cos(\theta)$ and $y = r\sin(\theta)$

In the first example, I set $r_0 = 0.2$, $v = 12.5$ $\omega = 1$, $\theta_0 = 0$ and t goes from 0 to 1. Noise is added in both x and y direction following normal distribution with 0.05

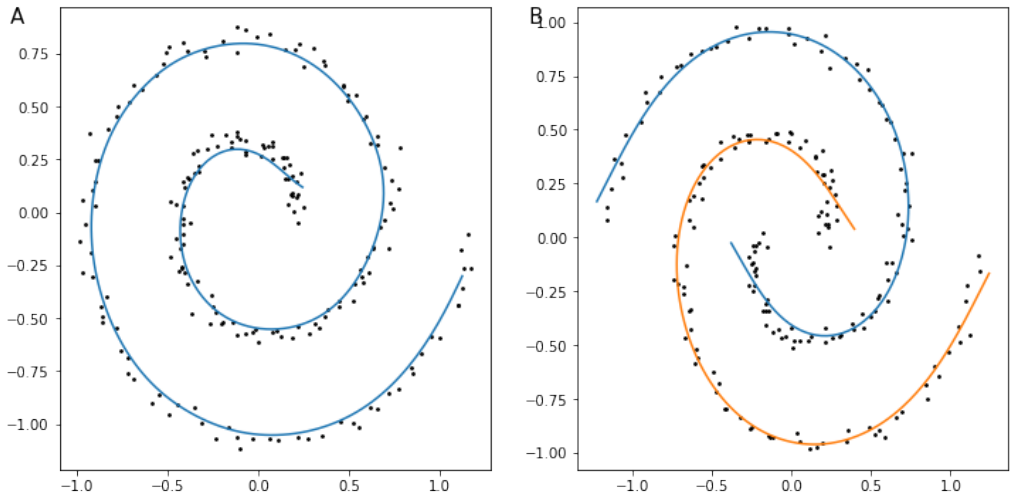


Figure 4.3. **Spiral Data**

variance.

In the second example, I made the task harder by using two convoluted spirals. I set $r_0 = 0.2$, $v = 6.3$ $\omega = 1$ and t goes from 0 to 1 for both spirals. For one of the spiral, $\theta_0 = 0$ and for the other $\theta_0 = 3.1$. Noise is added in both x and y direction following normal distribution with 0.05 variance.

It showed (Fig.4.3) that `Curd` could successfully recover the underlying structure of both 1 spiral and 2 convoluted spirals with different initial angle θ_0.

I have also applied the baseline methods, i.e. `iWMM` and `DRjCC` on the convoluted spiral data and the results are shown in Fig.4.4. (Note that for these two methods, they don't report the parametric form of the curve so that it is not possible to draw the curve explicitly like what I did for the result from `Curd`.) The correlation between the pseudotime reported from this two method and the ground truth is 0.77 and 0.85 respectively, which is much lower than 0.97 for the pseudotime reported by `Curd`.

4.4.3 3D spiral data

I have also tested my method against spiral data in higher dimension, which is again traditionally used to test the performance of dimension reduction methods[187]. I have simulated spiral in the 3 dimensional space with function $x = \cos(\omega t + \theta_0)$, $y = \sin(\omega t + \theta_0)$ and $z = vt$.

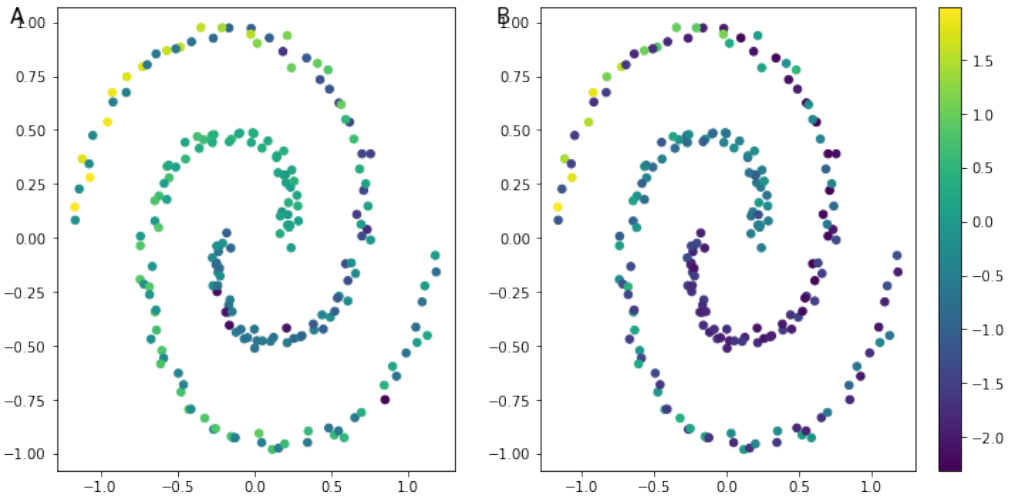


Figure 4.4. **Spiral Data with Baseline method** ***The color of the dots indicate the reported pseudotime A. Results from method*** `iWMM`***. B. Results from method*** `DRjCC`**.**

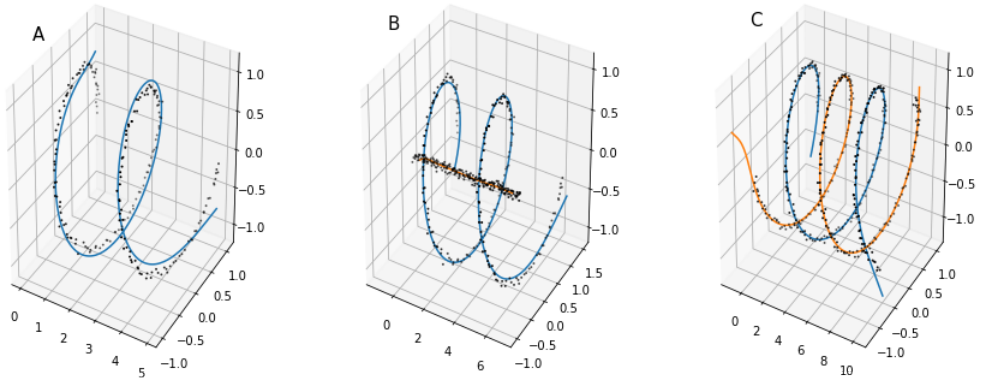


Figure 4.5. **Three Dimensional Spiral Data**

In the first example (Fig.4.5A), $\omega = 12.5$, $\theta_0 = 0$, $v = 5$ and t goes from 0 to 1. In the second example (Fig.4.5B), a straight line with $x = 0$, $y = 0$ and $z = 7t - 1$ is added. In the final example (Fig.4.5C), there are two spirals with $\omega = 9.4$, $v = 10$ and one of them has $\theta_0 = 0$, while the other has $\theta_0 = 3.1$. The noise is introduced in x and y direction following normal distribution with variance 0.05 as before.

It is shown from the result (Fig.4.5) that `Curd` could successfully recover the lower dimensional structure for all these three cases involving 3d spirals.

4.4.4 Choosing K: number of curves

As mentioned before, I proposed a new method using a modified version of group LASSO method for choosing K, which is a challenging task for previous methods[155].

Here, in this example, I showed the utility of using new method to choose the number of curves/clusters. I simulated two half circles with radius 1 and variance 0.1 in both x and y axis, and moved one of the half circle off by 0, 0.2, 0.4 and 0.6 respective in the y axis. It could be seen from the results (Fig.4.6) that when the two half circles co-locate, i.e. when the offset is 0, `Curd` successfully finds that there is only 1 curve. When the offset is small and two half circles are close to each other, i.e. the off-set is 0.2, `Curd` fails to distinguish these two half circles and only output one half circle. And when the offset is significantly large, i.e. the offset is between 0.4-0.6, the two half circles are detected by `Curd`. This showed that when the distance between two cluster is significant larger than the noise, i.e. off set 0.4 compared to noise 0.1, the method proposed could successfully identify the two clusters.

4.4.5 COVID-19-ICU data

I then applied my method to a real world data for exploratory analysis. The data is downloaded from Kaggle data set "COVID-19 - Clinical Data to assess diagnosis"[1]. It contains the body measurement data of patients sent to hospitals because of COVID-19[188]. The data is preprocessed so that no privacy information could be identified. As detailed in the following paragraphs, this data set contains information which could be used as benchmark for both clustering and dimension reduction tasks. Thus, this data is used to test the performance of `Curd` on these two tasks.

This data set contains 231 features in total. The measurement features within these features include the mean value, median value, minimum value, max value, difference and relative difference value between the min and max of the measurements. To make the task simpler, only the mean value of the measurements is used. Thus, there are in total 41 variables which could be roughly categorized into three parts: (1) response variables (used as ground truth for clustering): whether the patient is sent to ICU or not at the time when the measurement is taken; (2) time variables (used as ground truth for dimension reduction): at the time when the body measurement is taken, how long it is after the patient has been first admitted to hospital; (3) prediction variable, respiratory rate, body temperature, age and etc. of patients. The goal here is to understand the relationship between the prediction variable and

[1] https://www.kaggle.com/Sirio-Libanes/covid19

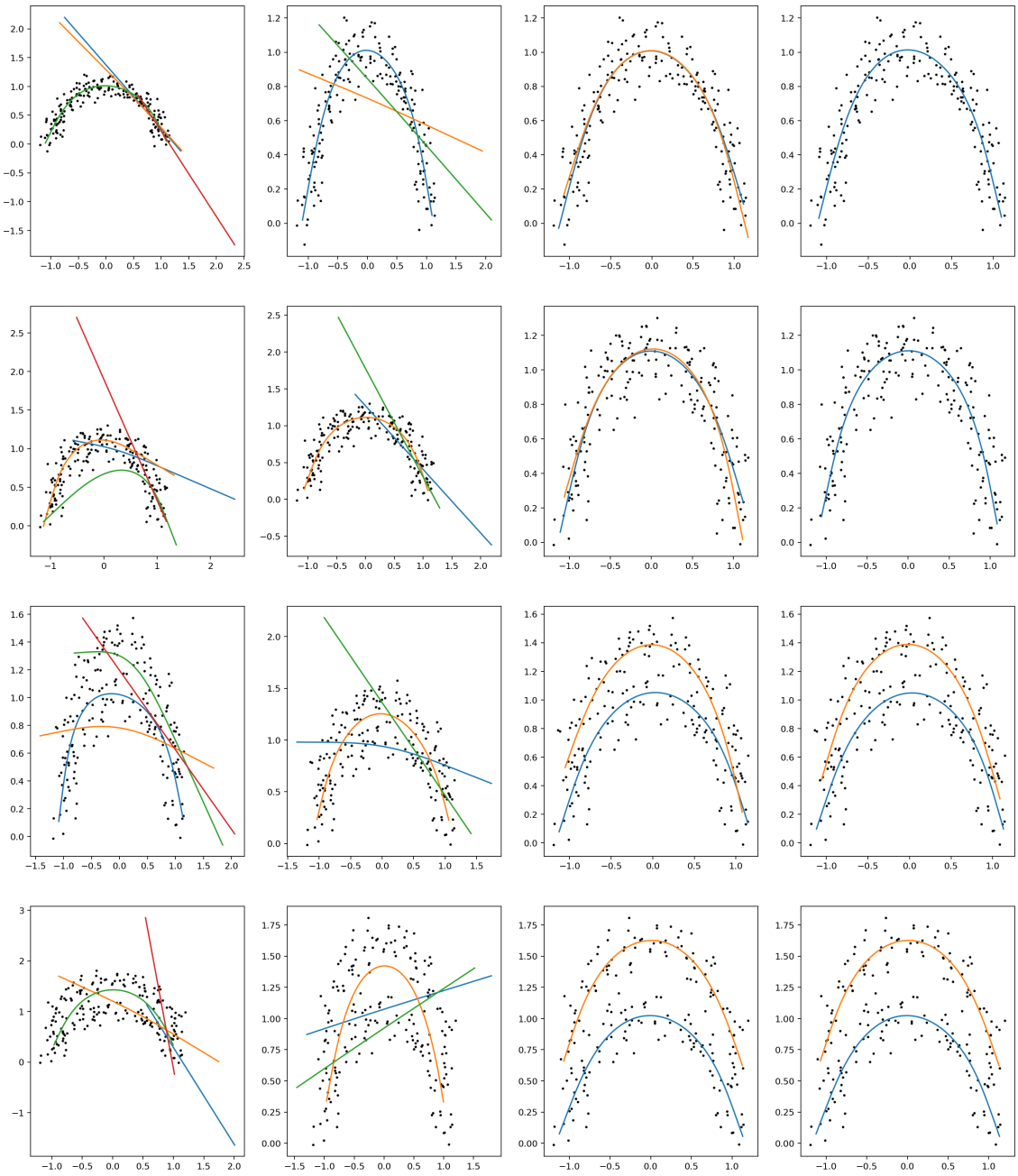


Figure 4.6. **Choose the number of clusters.** ***This figure illustrates the effect of shrinkage prior on the selection of the number of clusters to use. From top to bottom, I simulated two half circles with radius 1 and distances in y-axis 0,0.2,0.4,0.6 respectively. From left to right, the algorithm starts with*** $K = 4$ ***clusters, and reduces the number of clusters under the force of penalty. For the first and second row, the*** K ***gets finally reduced to 1. For the third and fourth row, the number of clusters gets reduced to*** $K = 2$***, and it stops reducing the number of clusters.***

response variable, i.e. if there is any difference between patients sent to ICU and those who are not. As it has already been shown that age is strongly correlated with the response variable[189], the age information has been excluded for analysis.

There are 1925 data points in total. However, after removing measurements which contain missing value, there are 545 data points left. For each patient, 1 to 4 data points is available depending on the time that the patient stayed in the hospital.

I first applied local linear embedding to extract two components for visualisation and then applied `Curd` to the dimension reduced data. My method `Curd` reported two trajectories (Fig.4.7A) detected. It could be clearly seen that these two trajectories match the response variable that whether this patient is in ICU or not at the time the body test being done (Fig.4.7B). The clustering accuracy based on the ICU information is 0.64. This result showed that the clustering result of `Curd` reflected patients physical conditions.

In Fig.4.7C, it shows patients who are not admitted to ICU at the time when all prediction variables are measured, but admitted to ICU at a later time are mostly located at one end point of both trajectories. In Fig.4.7D, it shows patients who are in the early stage (within 4 hours of admitted to hospital) and mid stage (between 4-12 hours of admitted to hospital) are also located at this end of the trajectories. The correlation between the pseudotime detected and the physical time is 0.28. This concluded that, for data points lying on one end of both curves, they are related to patients in their early stay in the hospital, while for data points lying on the other ends for both curves, they are associated with patients in their late stage in hospital. This separation of patients in early/mid stage from patients in later stage on two end of the trajectories showed that the 1 dimensional pseudotime structure of the two trajectories matches the physical time of patients' stay in the hospital.

This example shows that `Curd` has the potential to recover real world physical temporal structure from the feature variables in a clinical data set.

To verify the conclusion, I applied `Curd` as well as other clustering[158, 159, 161, 163, 167, 169] and dimension reduction[76, 156, 168, 170, 173] to the original data set. To compute the standard deviation, bootstrapping[190] is used here. In every bootstrap, 200 data points are randomly sampled and the experiment is repeated 5 times. The experiment is carried out on the original 38 dimensions after removing the age feature and the two features used as ground truth for comparison. For clustering task, whether the patient is in ICU or not is used as the ground truth

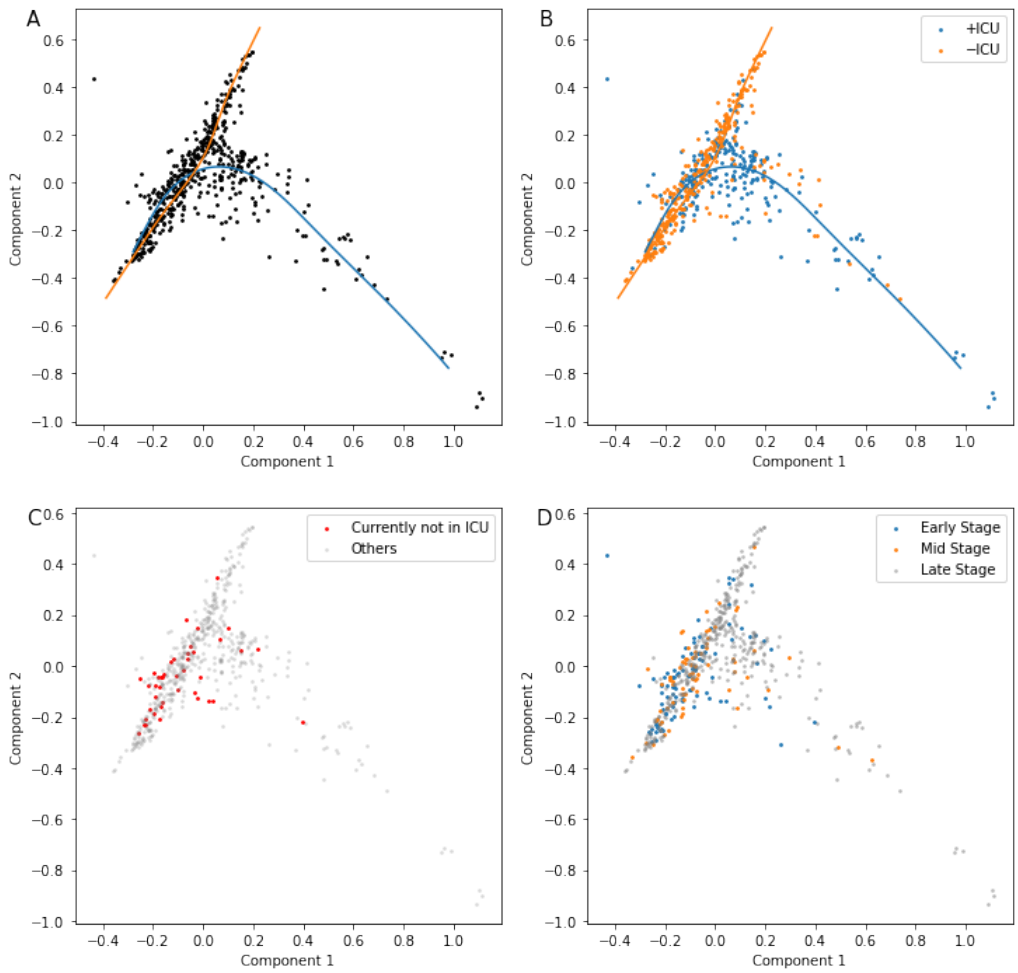


Figure 4.7.* COVID-19-ICU Data *A. Component 1 and 2 are from the results of local linear embedding. Two colored curves are fitted by my method. B.Two curves correspond very well to the patients being/not being in ICU at the time of the measurement is taken. C. The red dots are measurements of patients who are not in ICU at the time taken but being sent to ICU later on. D. The color correspond to the length of the time after the patient has been admitted to the hospital.

to compute the accuracy and it is specified that the number of cluster is 2 for all clustering methods. For dimension reduction task, the physical time that the patient stayed in the hospital when the measurements are taken is used as the ground truth to compute the correlation against the single dimension reported by the dimension reduction methods.

The bootstrapping results (Table.4.1,Table.4.2) confirmed that, compared to other commonly used dimension reduction and clustering tools, `Curd` could successfully identify the underlying latent structure within this COVID-19-ICU data set with clustering accuracy around 0.65, and 0.21 correlation with the physical time for

Method	`Curd`	`iWMM`	`PCA`
Correlation	$\mathbf{0.21 \pm 0.04}$	0.19 ± 0.05	0.16 ± 0.02
Method	`MDS`	`t-SNE`	`LLE`
Correlation	0.09 ± 0.06	0.16 ± 0.12	0.18 ± 0.14

Table 4.1. Dimension reduction results for COVID-19 data

Method	`Curd`	`iWMM`	`Kmeans`	`SpectralClust`
Accuracy	$\mathbf{0.66 \pm 0.04}$	0.63 ± 0.03	0.61 ± 0.06	0.64 ± 0.05
Method	`HierarchicalClust`	`DensityClust`	`GaussianMixture`	`DeepClust`
Accuracy	0.53 ± 0.02	0.55 ± 0.02	0.59 ± 0.05	0.60 ± 0.04

Table 4.2. Clustering results for COVID-19 data

dimension reduction task.

It is worth noting that the results only indicate that `Curd` achieved comparable performance but not necessarily better results compared to other methods.

4.4.6 Single cell sequencing data: fibroblast development

I then applied my method `Curd` to scRNA-seq data from samples related to fibroblast development and compare its performance to existing method[155, 156, 180]. This scRNA-seq data were collected for the study of lineage reprogramming from fibroblast cells to neuron cells. The fibroblast cells were isolated from mice to be cultured in lab and treated with Ascl1 to trigger their reprogramming towards neural cells. The cultured cells were collected on day 0, 2, 5, 20 and 22 respectively and their RNA has been extracted for scRNA-seq[140].

The data has been used previously for evaluating the pseudotime algorithms for scRNA-seq data analysis[150] and the physical time for cell collection is used as the ground truth for such data. There are 355 cells with expression data for 3379 genes in total contained in the data set.

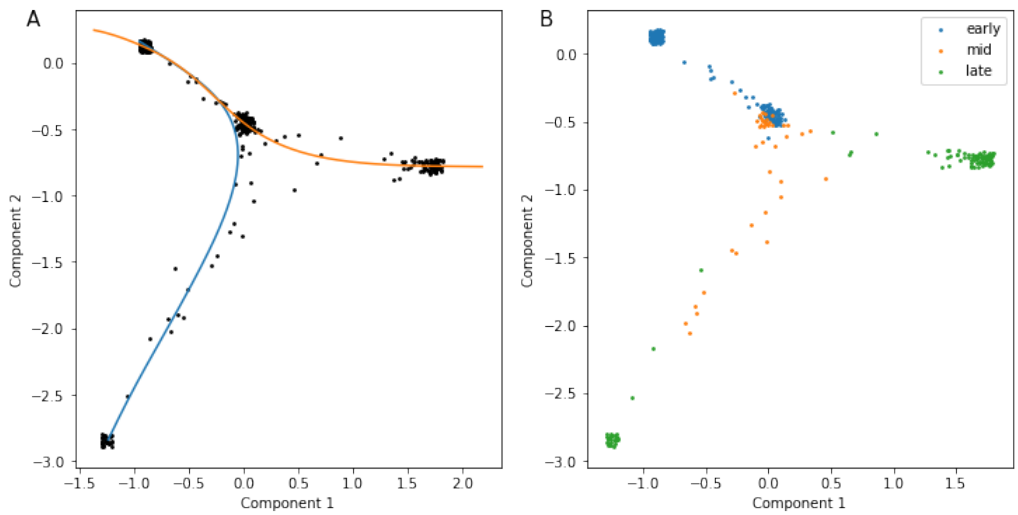


Figure 4.8. **Fibroblast development scRNA-seq data** ***A. Two components of x,y axis are computed by local linear embedding. Two trajectories are detected by my method. Black dots are single cells. B.The colored dots indicate the ground truth of temporal order and it confirms these two development trajectories.***

For visualisation, two components are computed by local linear embedding[170] and each dot indicates a single cell (Fig.4.8). On such dimension reduced data, `Curd` reported 2 trajectories, confirmed by using other pseudotime algorithms[150]. The two colored curves in Fig.4.8A are the two trajectories detected by `Curd`. The right part of the figure shows the ground truth of temporal information. This figure illustrated that the cells start from the early stage (day0 and day2), start to diverge during the mid stage (day5) and end in two different late stages (day 20 and day 22).

Then, I applied `Curd`, `DRjCC`[155], `iWMM`[156] and a pseudotime algorithm `DPT`[180], to the original data set and checked the pseudotime reported by the methods. I removed genes whose standard deviation among all cells are smaller than 3 because training on the original data is time consuming, leaving 192 genes for further analysis. The result of `Curd` gives a correlation of 0.67 and the result of `iWMM` gives a slightly lower correlation of 0.50. However the result of `DPT` and `DRjCC` give a correlation score of about 0.22 to the ground truth. This is also shown from Fig.4.9, where the result of `Curd` capture the underlying structure better than the other three methods. This indicates that `Curd` is better at capturing the underlying temporal information in scRNA-seq data than these existing methods.

To measure the stability of the model, bootstrapping method was used. In each

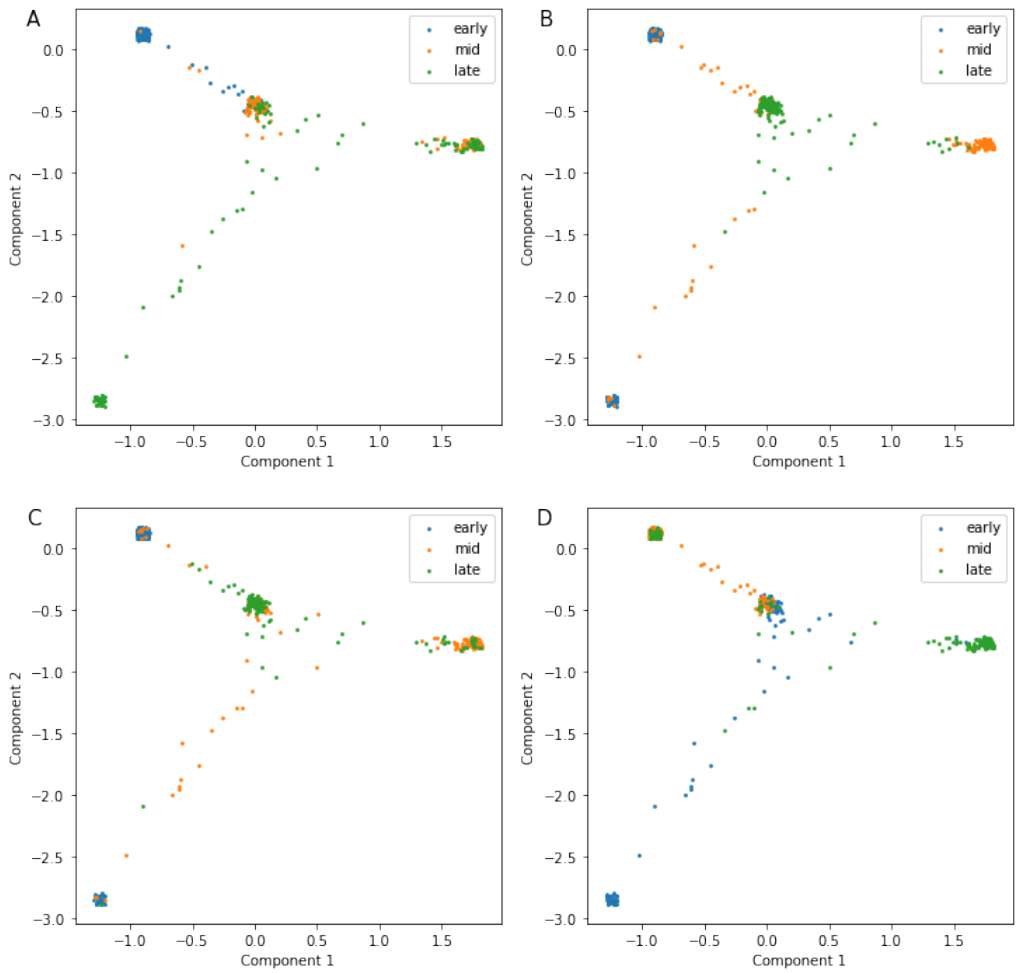


Figure 4.9. **Pseudotime results on fibroblast scRNA-seq data** ***The pseudotime has been binned to three groups: early, mid, and late, with equal cells in each group for visualisation. A. The color correspond to the results of*** `Curd`***. A. The color correspond to the results of*** `DPT`***. C. The color correspond to the results of*** `DRjCC` ***D. The color correspond to the results of*** `iWMM`

bootstrapping, I randomly sampled 300 samples and the experiment is repeated 5 times to compute standard deviation. The results (Table.4.3) again confirmed that the pseudotime found by `Curd` is better correlated with the ground truth.

4.4.7 Single cell sequencing data: blood cell development

At the end, I applied my method to a single cell study to illustrate how the method could be used for scRNA-seq analysis. The results are based on a single cell study for normal blood and leukemia cells[144].

The authors have sequenced the RNA expression for thousands of single cells from the blood of healthy people and blood of mixed phenotype acute leukemia (MPAL)

Method	`Curd`	`DPT`	`DRjCC`	`iWMM`
Correlation	$\mathbf{0.68 \pm 0.02}$	0.42 ± 0.23	0.13 ± 0.08	0.55 ± 0.08

Table 4.3. Bootstrapping results on fibroblast development data

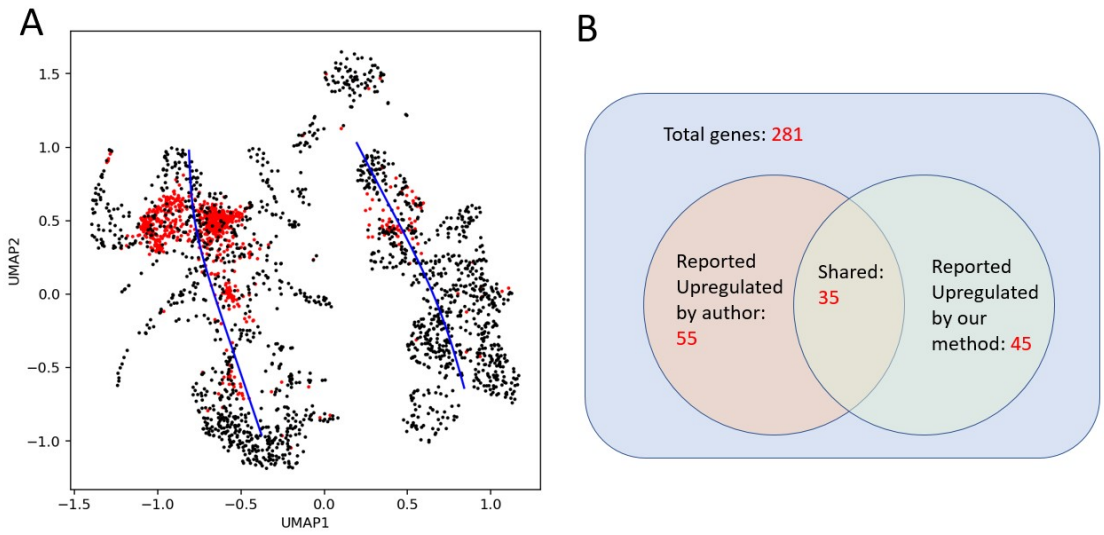


Figure 4.10.* Results on scRNA-seq leukemia data set. *I applied my method to single cell scRNA-seq data for both healthy people and leukemia patients. (A) Black dots indicate single blood cells of healthy people, and red dots indicate those of leukemia patients. The two blue lines are the two curves output by* `Curd`*. Two axis are computed by UMAP method. (B) The Venn graph for all up-regulated genes reported by author's method and* `Curd` *using the two curves.

patients. One of the goal from the study is to find the differentially expressed genes between MPAL patients and healthy people, which might be a potential driver for the disease.

The data set contains 3558 single cells for normal blood and 1805 single cells for MPAL. In total, there are 281 genes whose expression is detected for most cells and got large variance from scRNA-seq.

Unlike bulk cell sequencing, where the average expression of MPAL patients cells is compared to that of healthy people, in single cell sequencing, it is not clear that for each cell from MPAL patients, which cell in health population should be compared to it.

As shown in Fig.4.10A, the red dots indicate the cells from MPAL patients and black ones indicate cells from health populations. In the original publication, the region crowded by red MPAL cells were determined to be hematopoietic stem cell (HSC) populations, which was verified by other experimental methods, including cell surface markers[144]. This indicates that these MPAL cells are most similar to the HSC cells in normal tissues. Thus, if the average expression of MPAL cells is compared to the average gene expression from healthy people, or the expression of MPAL cells are compared to cells other than HSC cells from healthy people, it is likely that the differentially expressed genes reported later on are not involved in cancer development, but instead involved the normal development of blood cells.

Given the similarity between MPAL cells and HSC cells in normal tissues, in order to find the differentially expressed genes in MPAL cells, the authors compared each cell from the MPAL patient to a cell from the healthy population closest to it (based on Euclidean distance). And here, by using `Curd`, it is possible to propose an alternative method.

As in `Curd`, the assumption is that there are different types of transitions going on in the blood cell population, one of them might be the blood development process while the other one is not known for now. My method `Curd` first separates cells into two sub groups which are undertaking these two different transitions and assigns a pseudotime for each cell. I then compared each cell from MPAL population (red dots in Fig.4.10A) to cells that received the same pseudotime in the same transition process from the healthy population (black dots in Fig.4.10A) to get the differentially expressed genes. I used a threshold of being up-regulated in 40% of the cells in MPAL compared to normal blood cells as a potential up-regulated gene, which is the same threshold used by the authors of the original study. I then compared the differentially expressed genes identified by this method based on `Curd` to the ones found in the original study.

I found that (Fig.4.10B) there is strong correlation between the two methods, as 35 out of 45 (approximately 80%) genes identified as up-regulated by my method were reported up-regulated by the authors gene list containing 55 genes in total. The number of genes being studied was 281. This showed that my method is largely

consistent with the approach proposed by the authors.

For the 10 genes identified by my method but not by the original study, there is evidence linking them to oncogenesis (Table.4.4), which suggests these could be false negatives missed by the authors method. For example, the SEPT6 gene, encodes the septin GTPase, mutation of which has been previously reported in leukemia[191]. RAC2 which encodes a protein within the Ras superfamily, has also been shown to be associated with leukemia in mouse models[192].

One of the genes identified as up-regulated in the original study but not by my method, is ITGA6, a membrane protein within the integrin alpha chain family. Although this is up-regulated in 70% of the MPAL cells by the method in the original study, it has been shown to be closely related to the differentiation of normal blood cells as hypothesised before[193]. This indicates this gene might be a false positive.

Clearly without ground truth, it is hard to draw firm conclusions about which method is indeed better. However the evidence shown above suggests that my method at least provides a promising alternative approach to that used by the authors to find differentially expressed genes for scRNA-seq data analysis. Combining my method with the existing method may lead to a more complete understanding of the differentially expressed genes.

4.5 Discussion

In this chapter, I developed a new method called `Curd` which combines dimension reduction and clustering. This method has a sound statistical foundation and allows for non-linear relationships between the low dimensional structure and high dimensional data. I also proposed a method for automatically choosing the number of clusters which is a difficult task in previous method[155] and showed its utility.

I showed with simulated experiments that this new method is able to accomplish more complex tasks than the one previously used in literature[186, 187]. This indicates that the curvature penalty and the softmin approximation works properly that it captures the general shape of the curves while not overfitting on the random noise. On a real world data related to COVID, I showed that this new method could

Gene Symbol	Up regulated ratio	Literature
SEPT6	0.46	[194]
TMEM123	0.46	[195]
TAGLN2	0.43	[196]
AIF1	0.43	[197]
TMSB4X	0.42	[198]
TSC22D3/GILZ	0.41	[199]
PTPRC	0.41	[200]
SERPINB1	0.40	[201]
RAC2	0.40	[202]
ARHGDIB	0.40	[203]

Table 4.4. Up-regulated genes

successfully uncover the underlying subtype and temporal structure. This confirmed that `Curd` could uncover the underlying cluster and low dimensional structure better than other methods. This is potentially the benefit of joint learning the dimension reduction and clustering[154]. Finally, I used two scRNA-seq data set to illustrate how `Curd` to be used to this type of data. In the first example, `Curd` is compared to the existing methods on finding the latent temporal structure. It showed that the result of `Curd` matches the ground truth better than existing methods. In the second example, `Curd` is used for the development of a new method to detect differentially expressed genes from scRNA-seq data. This new method shows that it is consistent with the approach proposed previously[144], and my method could potentially used to find the false negative genes not detected by the existing approach. And this might be because `Curd` truly capture the temporal order of the cells involved in blood cell differentiation. Thus removing the effect introduced by such temporal change through `Curd` could hopefully capture the truly differentially expressed genes between MPAL cell and normal blood cells.

I believe this new method has potential to contribute to the scRNA-seq data analysis as well as other fields in the future. For example, this method could be used to do interpolation for scRNA-seq data. As in scRNA-seq data, there are often missing values for some cells, for example due to sequencing error or error in library preparation. Thus, it is sometimes necessary to fill-in these missing values by values from other cells[204]. This process in scRNA-seq analysis is called interpolation. With the new method `Curd`, it is possible to use the values from the cells which belong to the same cluster and have similar low dimensional values as the cells containing missing values. This method could potentially make the interpolation process more accurate.

For the analysis of COVID data, if allowed, it will be interesting to check the predictive power of the finding. For example, as mentioned before, it is possible to map new coming data to the trajectory found by `Curd`. Thus, if the relevant data related to the patients become available in the future, it is possible to check which trajectory the new coming belongs to. And thus, it is possible to compute the probability that the newly admitted patient is going to be sent the ICU in the future. If proven useful, such approach could be used to help clinicians allocate the beds and devices necessary for the patients in ICU.

However, this method certainly has its limitations which could be optimised in the future work. First of all, scalability is certainly an issue for the current version of `Curd`. Because of the finite difference method used, the training time of `Curd` is often quite long as mentioned. In the future version of `Curd`, it may be possible to develop distributed algorithm for the optimisation of the tangent variables to reduce the training time[205].

Second, similar to the `DRjCC` method[155], it is not easy to determine the hyper-parameter used for the penalty term. Traditionally in machine learning, such hyper-parameter could be determined by cross-validation[31]. However, as K-fold cross-validation will require the method to be run for K times, it increases the time for training the model. Thus, in the future version of `Curd`, one interesting direction is to develop the method for choosing hyper parameter by estimating some properties of the data. For example, the penalty parameter could be set to be proportional

to the L2 norm of the variance of the data points. However, this hypothesis needs further simulation studies to check its utility. Besides, as the penalty parameter is related to the variance, it might be possible to learn such parameter from data using the update rule similar to the one used for Gaussian mixture model[158].

Finally, though it is proven that the solution of the optimisation target of `Curd` is a BVP, it is not used in practice. The main reason is that computing the solution of BVP in high dimensional spaces is often more time consuming than doing the optimisation steps mentioned. However, with recent development in deep learning with ordinary differential equation layers [206, 207], it is promising to check if such deep learning approach could be applied to such problem and reduce the time complexity.

Overall, I believe that the new method `Curd` has its potential to benefit the community of scRNA-seq data analysis as well as many other areas.

4.6 Appendix

4.6.1 Proofs

Proof for Theorem 4.1. To start the proof, I first assume that the pseudotime t_j is a fixed and known parameter to the system. Here, I use $\mu(t)$, $\omega(t)$ and $\gamma(t)$ to indicate the dual variables with respect to three equality constraints respectively. And set $F(t) = -\int_t \omega(t')dt'$.

$$\begin{aligned}\text{Lagrange dual } &= \sum_j \|\vec{x_j} - f(t_j)\|_2^2 + \rho \int \|\beta(t)\|_2^2 dt \\ &- \int \mu(t)[f(t) - \vec{f^0} - \int_0^t \alpha(t')dt']dt \\ &- \int \omega(t)[\alpha(t) - \vec{\alpha^0} - \int_0^t \beta(t')dt']dt \\ &- \int \gamma(t)\alpha(t)^T\beta(t)dt\end{aligned}$$

Take gradients with respect to three original variables $f(t)$, $\alpha(t)$ and $\beta(t)$ and get the following relation.

$$f(t_j) - \vec{x_j} - \mu(t_j) = 0 \tag{4.22}$$

$$\sum_{j:t_j>t} \mu(t_j) - \omega(t) - \beta(t)\gamma(t) = 0 \tag{4.23}$$

$$\rho\beta(t) + \int_t \omega(t) - \alpha(t)\gamma(t) = 0 \tag{4.24}$$

By multiplying $\alpha(t)$ to equation (4.24), it is easy to get the following.

$$\gamma(t) = -\alpha(t)^T F(t)$$
$$\beta(t) = \frac{1}{\rho}\left[\boldsymbol{I} - \alpha(t)\alpha(t)^T\right] F(t)$$

Be inserting the above equation for $\beta(t)$ into equation (4.23) and apply the equation (4.22), it is possible to get the following.

$$\begin{aligned}\nabla_t F(t) &= \frac{1}{\rho}\left[(\boldsymbol{I} - \alpha(t)\alpha(t)^T)F(t)F(t)^T\alpha(t)\right] + \sum_{j:t_j>t}\mu(t_j) \\ &= \frac{1}{\rho}\left[(\boldsymbol{I} - \alpha(t)\alpha(t)^T)F(t)F(t)^T\alpha(t)\right] + \sum_{j:t_j>t}[f(t_j) - \vec{x_j}]\end{aligned}$$

By taking gradient with respect to f_0 and α_0, the following two equations could be shown.

$$\sum_j \mu(t_j) = 0$$
$$\int \omega(t)dt = 0$$

Thus, the boundary condition could be found out as below. And the condition $F(T) = 0$ follows naturally from its definition.

$$\sum_j \vec{x_j} - f(t_j) = \sum_j \mu(t_j) = 0$$
$$F(0) = \int \omega(t)dt = 0$$

Now, to check the parameter condition related to t_j, it could be shown directly be taking the gradient of the loss shown below to 0.

$$\nabla_{t_j} = \alpha(t_j)^T[\vec{x_j} - f(t_j)] = 0$$

□

Proof of convergence for Algo.4.2. To prove Algo.4.2 will converge when used together with the update for other steps, I will prove that the loss associated with it will not increase.

The optimisation problem (4.16) could be simplified to have the following form.

$$\begin{aligned} \min_{\boldsymbol{P}} \quad & \boldsymbol{PX} \\ \text{s.t.} \quad & \boldsymbol{P}^T\boldsymbol{P} = \boldsymbol{I} \end{aligned}$$

To update the value of $\boldsymbol{P}$, I proposed finding a Householder matrix $\boldsymbol{H} = \boldsymbol{I} - 2\vec{n}\vec{n}^T$ with $\|\vec{n}\|_2 = 1$ and update its value to $\boldsymbol{P'} = \boldsymbol{HP}$.

The optimisation problem above could be then converted to the following form.

$$\begin{aligned} \min_{\boldsymbol{H_1}} \quad & \boldsymbol{H_1}\boldsymbol{H_2}\boldsymbol{PX} \\ \text{s.t.} \quad & \boldsymbol{H_1} = \boldsymbol{I} - 2\vec{n_1}\,\vec{n_1}^T \\ & \boldsymbol{H_2} = \boldsymbol{I} - 2\vec{n_2}\,\vec{n_2}^T \end{aligned}$$

Here, $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$ are introduced to have the same initialised value for $\vec{n_1}$ and $\vec{n_2}$, so that initially $\boldsymbol{H_1}\boldsymbol{H_2} = \boldsymbol{I}$. This introduction of two Householder matrices is to guarantee that the initial loss is the same . Thus, it is only necessary to show that the update of $\vec{n_1}$ will decrease the loss and the update $\boldsymbol{P'} = \boldsymbol{H_1}\boldsymbol{H_2}\boldsymbol{P}$ will also decrease the loss.

The optimisation thus become the following.

$$\begin{aligned} \min_{\vec{n_1}} \quad & -\vec{n_1}^T\boldsymbol{A}\vec{n_1} \\ \text{s.t.} \quad & \|\vec{n_1}\|_2 = 1 \\ & \boldsymbol{A} = \boldsymbol{H_2}\boldsymbol{PX} \end{aligned}$$

Without changing the solution, it is equivalent to set the optimisation target as following. This change of target is necessary to keep the $\boldsymbol{A'}$ positive definite and $\|\boldsymbol{A}\|$ could be any matrix norm.

$$\begin{aligned} \min_{\vec{n_1}} \quad & -\vec{n_1}^T \boldsymbol{A'} \vec{n_1} \\ \text{s.t.} \quad & \|\vec{n_1}\|_2 = 1 \\ & \boldsymbol{A'} = \frac{\boldsymbol{A} + \boldsymbol{A}^T + 2\|\boldsymbol{A}\|\boldsymbol{I}}{2 + 2\|\boldsymbol{A}\|} \end{aligned}$$

Thus, it remains to show the update step shown below, which is traditionally used for computing the eigenvector with the largest eigen[183], will not make the loss increase.

$$\vec{n_1'} = \frac{\boldsymbol{A'}\vec{n_1}}{\|\boldsymbol{A'}\vec{n_1}\|_2}$$

It is then equivalent to show the following.

$$\vec{n_1}^T \boldsymbol{A'}^3 \vec{n_1} > \|\boldsymbol{A'}\vec{n_1}\|_2^2 \vec{n_1}^T \boldsymbol{A'} \vec{n_1}$$

By using the eigenvalue decomposition, it is defined that $\boldsymbol{A'} = \boldsymbol{Q}^T \boldsymbol{\Lambda} \boldsymbol{Q}$ and $\vec{n_1} = \boldsymbol{Q}^T \vec{n_3}$, where $\|\vec{n_3}\|_2 = 1$. Then, it is left to show the following, where λ_i is the ith eigenvalue of $\boldsymbol{A'}$ and n_{3i} is the ith entry of $\vec{n_3}$

$$\begin{aligned} \vec{n_3}^T \boldsymbol{\Lambda}^3 \vec{n_3} &> \|\boldsymbol{\Lambda}\vec{n_3}\|_2^2 \vec{n_3}^T \boldsymbol{\Lambda} \vec{n_3} \\ \sum_i \lambda_i^3 n_{3i}^2 &> \left(\sum_i \lambda_i^2 n_{3i}^2\right)\left(\sum_i \lambda_i n_{3i}^2\right) \end{aligned}$$

As $n_{3i}^2 \geq 0$, $\sum_i n_{3i}^2 = 1$, and $\forall i,\ \lambda_i \geq 0$, it is possible to verify the following.

$$
\begin{aligned}
&\sum_i \lambda_i^3 n_{3i}^2 \\
&= \left(\sum_i \lambda_i^3 n_{3i}^2\right)\left(\sum_i n_{3i}^2\right) \\
&= \left(\sum_i \lambda_i^2 n_{3i}^2\right)\left(\sum_i \lambda_i n_{3i}^2\right) + \sum_{i,j}(\lambda_i^3 + \lambda_j^3 - \lambda_i^2\lambda_j - \lambda_i\lambda_j^2) n_{3i}^2 n_{3j}^2 \\
&= \left(\sum_i \lambda_i^2 n_{3i}^2\right)\left(\sum_i \lambda_i n_{3i}^2\right) + \sum_{i,j}(\lambda_i + \lambda_j)(\lambda_i - \lambda_j)^2 n_{3i}^2 n_{3j}^2 \\
&\geq \left(\sum_i \lambda_i^2 n_{3i}^2\right)\left(\sum_i \lambda_i n_{3i}^2\right)
\end{aligned}
$$

The inequality holds as long as there are two eigenvalues of $\boldsymbol{A'}$ not equal and $\vec{n_1}$ has some projection value in the direction of the eigenvectors corresponding to these two eigenvalues.

□

4.6.2 Local update for tangent vectors

The optimisation problem (4.14) could be converted to the following form, which is essentially a linear target function with norm constraint for each $\alpha_k(t_i)$ with all other parameters fixed.

$$
\begin{aligned}
\min_{\alpha_k(t_i)} \quad & -\sum_{i,j,k}\sum_{i'>i} w_{jk}(t_{i'})[\vec{x_j} - f_k^{-i}(t_{i'})]\alpha_k(t_i)\Delta t - \rho[\alpha_k(t_{i-1}) + \alpha_k(t_{i+1})]\frac{\alpha_k(t_i)}{\Delta t} \\
\text{s.t.} \quad & f_k(t_{i'}) = f_k^{-i}(t_{i'}) + \alpha_k(t_i)\Delta t \\
& \|\alpha_k(t_i)\|_2 = 1
\end{aligned}
$$

From the optimisation problem above, a simpler update rule could be derived as following.

$$
\alpha_k(t_i) = \frac{\sum_j \sum_{i'>i} w_{jk}(t_{i'})[\vec{x_j} - f_k^{-i}(t_{i'})](\Delta t)^2 + \rho[\alpha_k(t_{i-1}) + \alpha_k(t_{i+1})]}{\| \sum_j \sum_{i'>i} w_{jk}(t_{i'})[\vec{x_j} - f_k^{-i}(t_{i'})](\Delta t)^2 + \rho[\alpha_k(t_{i-1}) + \alpha_k(t_{i+1})]\|_2} \tag{4.25}
$$

Here, I name the update rule (4.25) as local update and the update rule in Algo.4.2 as global update.

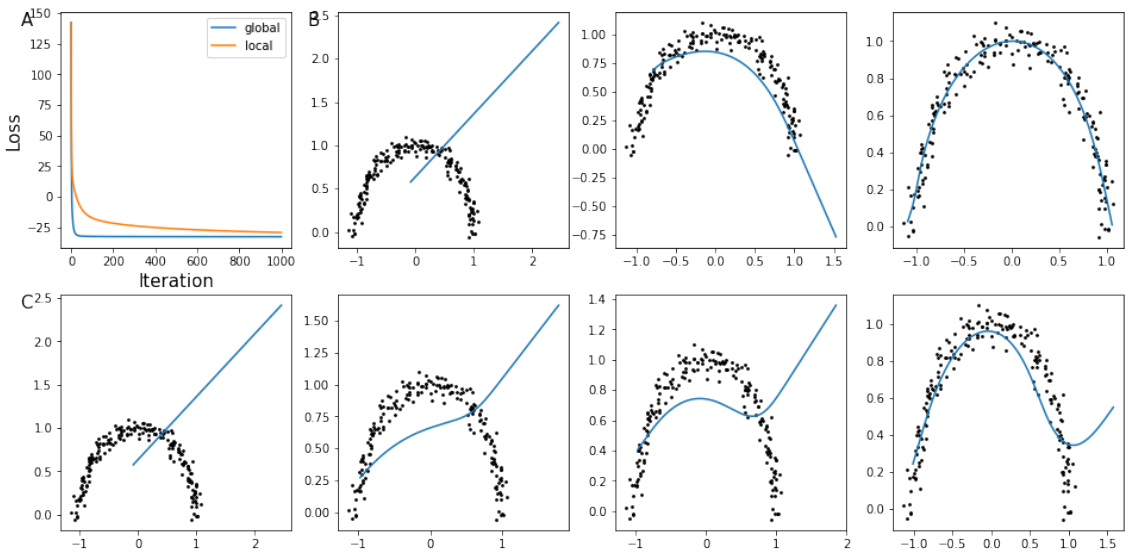


Figure 4.11. **Comparison between local and global update.** ***(A) The initial loss for both methods is the same at 142. The loss decreases slower by using local update. (B) The blue curve is the output of*** `Curd` ***and the black dots are simulated data points. The blue curves are learned by*** `Curd` ***using global update and from left to right, the training iterations are 1, 8, 200 respectively. (C) The blue curves are learned by*** `Curd` ***using local update and the training iterations are 1, 40, 100, 950 from left to right***

The biggest disadvantage of the local update is that its convergence is quite slow. In local update, the value of $\alpha_k(t_i)$ is heavily influenced by the adjacent tangent vectors while in global update, most adjacent tangent vectors are updated simultaneously. This has been shown in Fig.4.11.

In Fig.4.11, both local and global update are used to fit a half circle simulated with radius 1 and noise with variance 0.05. It shows(Fig.4.11A) that the loss decreases slower using local update. Besides, while global update could find the optimal solution very quickly(Fig.4.11B), the local update rule will change the value of tangents very slowly because of the penalty term related to adjacent tangents(Fig.4.11C).

Chapter 5

Discussions and conclusions

5.1 Overview

In this thesis, I have developed three novel methods based on machine learning algorithms for applications in cancer genomics. These methods were aimed at making better use of the genomic sequencing data to improve the understanding of cancer evolution process, which could potentially facilitate the process of drug development, clinical diagnosis and cancer treatment.

In chapter 2, I developed a novel method for mutation signature detection called `convSig`. The method is built on convolutional filtering technique and I have made necessary adaptations to bring it to the field of mutation signature detection. I have compared it to existing method[51, 52, 55] on simulated data and showed it achieved comparable performance in all cases. I have applied the methods to a clinical data set and showed that `convSig` could successfully recover the reported signatures compared to other methods. I also applied it to study mutation signatures associated with much longer nucleotide sequence (1Kb) in high mutation density region and detected a potential mutation signature.

In chapter 3, I developed a novel method for deconvolution of copy number evolution called `RLEvolution`. This method is also the first application of reinforcement learning in cancer evolution study. I have mathematically formulated the problem of copy number evolution as a reinforcement learning problem based on previous studies[20].

I have also developed a novel architecture of deep neural network that matches with permutation invariant structure the human chromosomes. I have showed the performance of `RLEvolution` is better than heuristic methods and an existing approach named `WCND`[109]. I have also applied the `RLEvolution` to TCGA CRC data[**?**] and showed its findings as consistent with previous studies using different methods[18]. I finally applied `RLEvolution` to two cell line samples from the same origin and showed that the trajectory discovered by `RLEvolution` share more common CNVs than that discovered by `WCND`.

In chapter 4, I have developed a method `Curd` which jointly carries out dimension reduction and clustering. This approach could be applied to scRNA-seq data as well as many other data. I first showed that such method could successfully accomplish complex tasks classically used[185–187] for the test of dimension reduction tools. I then showed that the clustering performance of `Curd` is as good as the best of other popular clustering tools[158, 159, 163, 169, 208]. In the COVID-19 data set, the two trajectories found by `Curd` is shown to be highly correlated with patient condition and the 1 dimensional pseudotime detected by `Curd` has a good correlation with the physical time. Next, I compared `Curd` to the method `DRjCC`[155] and `iWMM`[156] which carry out dimension reduction and clustering jointly for scRNA-seq analysis and a pseudotime algorithm `DPT`[180]. I showed that `Curd` got better results in the scRNA-seq data related to fibroblast development. Finally, I illustrated a potential application of `Curd` to detect the differentially expressed genes from scRNA-seq data and showed that it provides consistent and potentially complementary results to the methods used before[144].

The contribution of this thesis is mainly the development of these novel machine learning methods, which can be applied to more data sets in the future. All codes with these methods are submitted as supplementary materials with this thesis and they will be made freely online after the relevant work is published.

Furthermore, formulating the problems involved in the study of cancer evolution as a mathematical problem will facilitate the growth of machine learning methods in this field. In chapter 2, the mutation signature detection has been formulated into a supervised learning problem which is substantially different from the previous

unsupervised matrix factorization formulation[50]. In chapter 3, the copy number evolution problem is formulated as a reinforcement learning problem which is again different from the previous transition distance based method[109].

Except for the two aspects mentioned above related to methodology, the results found by the application the these methods could also contribute to the related field.

In chapter 2, the 1Kb mutation signature related to the high mutation density region could be of interest for further verification. It may shed light on the mutational process generating such high density region. In chapter 3, the finding that WGD is an early event during copy number evolution confirmed previous finding through different methods[18]. In chapter 4, the finding related to COVID-19 data set could potentially used to predict the number of ICU beds needed and the differentially expressed genes between leukemia cells and normal blood cells could be of interest for further functional studies.

5.2 Further direction

The specific future direction for each method is discussed at the end of each chapter and here I only briefly summarize the common scheme among these methods.

First of all, because of the different design of my methods compared to existing approaches, the application of them to more data set will be a direction where potential new findings could be made. For `convSig`, it could be applied to the clinical mutation data set to find new mutation signatures with different neucleotide sequence context. For `RLEvolution`, it could be applied to the copy number variation data set to find the common patterns involved in copy number evolution. For example, the results of `RLEvolution` might shed light on the copy number events before WGD, which could be the potential cause for WGD. For `Curd`, it could be applied to more scRNA-seq data to find out the molecular subtype as well as the 1 dimensional temporal relationship between single cells.

Secondly, though demonstrated their value in the relevant field, the methods could be optimised in certain directions, especially in model accuracy and scalability im-

provement. The accuracy of the deep models could be optimised by using potentially more complex model architectures developed in recent years, which have been proven to achieve state-of-the-art performance in other field[209, 210]. The scalability of the model is a difficult task and potential improvement could be made by using distributed algorithms[205].

Finally, development of new methods based on the mathematical formulation posed in each chapter will also be a valuable direction for further studies. For mutation signature detection, as this problem is re-formulated into a supervised learning problem, more classification models could be applied to this problem. For copy number evolution, as it has been posed as a reinforcement learning problem, more deep reinforcement learning models could also be developed in this field. For jointly dimension reduction and clustering, through the technique of softmin approximation, it has been formulated as a regression problem. Thus, more regression methods could be applied to this field.

5.3 Conclusion

Overall, in this thesis, I have provided three novel approaches which could give better understanding of the cancer evolution process using genomic data. My methods form a platform for further studies and could motivate more machine learning methods in the field.

References

[1] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Samuel AJR Aparicio, Sam Behjati, Andrew V Biankin, Graham R Bignell, Niccolo Bolli, Ake Borg, Anne-Lise Børresen-Dale, et al. Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421, 2013.

[2] Adriana Amaro, Silvana Chiara, and Ulrich Pfeffer. Molecular evolution of colorectal cancer: from multistep carcinogenesis to the big bang. *Cancer and Metastasis Reviews*, 35(1):63–74, 2016.

[3] Russell Schwartz and Alejandro A Schäffer. The evolution of tumour phylogenetics: principles and practice. *Nature Reviews Genetics*, 18(4):213, 2017.

[4] Christoph Grunau, SJ Clark, and André Rosenthal. Bisulfite genomic sequencing: systematic investigation of critical experimental parameters. *Nucleic acids research*, 29(13):e65–e65, 2001.

[5] Douglas Hanahan and Robert A Weinberg. Hallmarks of cancer: the next generation. *cell*, 144(5):646–674, 2011.

[6] N. J Haradhvala, J Kim, Y. E Maruvka, P Polak, D Rosebrock, D Livitz, J. M Hess, I Leshchiner, A Kamburov, and K. W and Mouw. Distinct mutational signatures characterize concurrent loss of polymerase proofreading and mismatch repair. *Nature Communications*, 9(1):3183–3196, 2018.

[7] Pierre Hainaut and Gerd P Pfeifer. Patterns of p53 g to t transversions in lung cancers reflect the primary mutagenic signature of dna-damage by tobacco smoke. *Carcinogenesis*, 22(3):367–374, 2001.

[8] Gerd P Pfeifer, Young-Hyun You, and Ahmad Besaratinia. Mutations induced by ultraviolet light. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 571(1):19–31, 2005.

[9] Sam Behjati, Gunes Gundem, David C Wedge, Nicola D Roberts, Patrick S Tarpey, Susanna L Cooke, Peter Van Loo, Ludmil B Alexandrov, Manasa Ramakrishna, Helen Davies, et al. Mutational signatures of ionizing radiation in second malignancies. *Nature communications*, 7(1):1–8, 2016.

[10] Nicola J Curtin. Dna repair dysregulation from cancer driver to therapeutic target. *Nature Reviews Cancer*, 12(12):801–817, 2012.

[11] Michael J Waring. Dna modification and cancer. *Annual review of biochemistry*, 50(1):159–192, 1981.

[12] Holly Soehnge, Allal Ouhtit, and ON Ananthaswamy. Mechanisms of induction of skin cancer by uv radiation. *Front Biosci*, 2:D538–D551, 1997.

[13] Adrian Baez-Ortega and Kevin Gori. Computational approaches for discovery of mutational signatures in cancer. *Briefings in bioinformatics*, 20(1):77–88, 2019.

[14] Matthew H Bailey, Collin Tokheim, Eduard Porta-Pardo, Sohini Sengupta, Denis Bertrand, Amila Weerasinghe, Antonio Colaprico, Michael C Wendl, Jaegil Kim, Brendan Reardon, et al. Comprehensive characterization of cancer driver genes and mutations. *Cell*, 173(2):371–385, 2018.

[15] Andriy Marusyk and Kornelia Polyak. Tumor heterogeneity: causes and consequences. *Biochimica et Biophysica Acta (BBA)-Reviews on Cancer*, 1805(1):105–117, 2010.

[16] Marco Gerlinger, Nicholas McGranahan, Sally M Dewhurst, Rebecca A Burrell, Ian Tomlinson, and Charles Swanton. Cancer: evolution within a lifetime. *Annual review of genetics*, 48:215–236, 2014.

[17] Adam Shlien and David Malkin. Copy number variations and cancer. *Genome medicine*, 1(6):62, 2009.

[18] Sally M Dewhurst, Nicholas McGranahan, Rebecca A Burrell, Andrew J Rowan, Eva Grönroos, David Endesfelder, Tejal Joshi, Dmitri Mouradov, Peter Gibbs, Robyn L Ward, et al. Tolerance of whole-genome doubling propagates chromosomal instability and accelerates cancer genome evolution. *Cancer discovery*, 4(2):175–185, 2014.

[19] Craig M Bielski, Ahmet Zehir, Alexander V Penson, Mark TA Donoghue, Walid Chatila, Joshua Armenia, Matthew T Chang, Alison M Schram, Philip Jonsson, Chaitanya Bandlamudi, et al. Genome doubling shapes the evolution and prognosis of advanced cancers. *Nature genetics*, 50(8):1189–1195, 2018.

[20] Rameen Beroukhim, Craig H Mermel, Dale Porter, Guo Wei, Soumya Raychaudhuri, Jerry Donovan, Jordi Barretina, Jesse S Boehm, Jennifer Dobson, Mitsuyoshi Urashima, et al. The landscape of somatic copy-number alteration across human cancers. *Nature*, 463(7283):899, 2010.

[21] Mehdi Zarrei, Jeffrey R MacDonald, Daniele Merico, and Stephen W Scherer. A copy number variation map of the human genome. *Nature reviews genetics*, 16(3):172–183, 2015.

[22] Rabia Aziz, CK Verma, and Namita Srivastava. Dimension reduction methods for microarray data: a review. *AIMS. Bioengineering*, 4(1):179–197, 2017.

[23] John E Reid and Lorenz Wernisch. Pseudotime estimation: deconfounding single cell time series. *Bioinformatics*, 32(19):2973–2980, 2016.

[24] Anguraj Sadanandam, Costas A Lyssiotis, Krisztian Homicsko, Eric A Collisson, William J Gibb, Stephan Wullschleger, Liliane C Gonzalez Ostos, William A Lannon, Carsten Grotzinger, Maguy Del Rio, et al. A colorectal cancer classification system that associates cellular phenotype and responses to therapy. *Nature medicine*, 19(5):619, 2013.

[25] Robert Burbidge, Matthew Trotter, B Buxton, and Sl Holden. Drug design by machine learning: support vector machines for pharmaceutical data analysis. *Computers & chemistry*, 26(1):5–14, 2001.

[26] Mufti Mahmud, Mohammed Shamim Kaiser, Amir Hussain, and Stefano Vassanelli. Applications of deep learning and reinforcement learning to biological data. *IEEE transactions on neural networks and learning systems*, 29(6):2063–2079, 2018.

[27] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

[28] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[29] Dustin T Holloway, Mark Kon, and Charles De Lisi. Integrating genomic data to predict transcription factor binding. *Genome informatics*, 16(1):83–94, 2005.

[30] Zhen Shen, Wenzheng Bao, and De-Shuang Huang. Recurrent neural network for predicting transcription factor binding sites. *Scientific reports*, 8(1):1–10, 2018.

[31] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[32] Martha L Bulyk. Computational prediction of transcription-factor binding site locations. *Genome biology*, 5(1):201, 2003.

[33] Elaheh Moradi, Antonietta Pepe, Christian Gaser, Heikki Huttunen, Jussi Tohka, Alzheimer's Disease Neuroimaging Initiative, et al. Machine learning framework for early mri-based alzheimer's conversion prediction in mci subjects. *Neuroimage*, 104:398–412, 2015.

[34] Damini Dey, Sara Gaur, Kristian A Ovrehus, Piotr J Slomka, Julian Betancur, Markus Goeller, Michaela M Hell, Heidi Gransar, Daniel S Berman, Stephan Achenbach, et al. Integrated prediction of lesion-specific ischaemia from quantitative coronary ct angiography using machine learning: a multicentre study. *European radiology*, 28(6):2655–2664, 2018.

[35] Laetitia Marisa, Aurélien de Reyniès, Alex Duval, Janick Selves, Marie Pierre Gaub, Laure Vescovo, Marie-Christine Etienne-Grimaldi, Renaud Schiappa,

Dominique Guenot, Mira Ayadi, et al. Gene expression classification of colon cancer into molecular subtypes: characterization, validation, and prognostic value. *PLoS medicine*, 10(5), 2013.

[36] E Melo Felipe De Sousa, Xin Wang, Marnix Jansen, Evelyn Fessler, Anne Trinh, Laura PMH De Rooij, Joan H De Jong, Onno J De Boer, Ronald Van Leersum, Maarten F Bijlsma, et al. Poor-prognosis colon cancer is defined by a molecularly distinct subtype and develops from serrated precursor lesions. *Nature medicine*, 19(5):614, 2013.

[37] Marc J Williams, Benjamin Werner, Chris P Barnes, Trevor A Graham, and Andrea Sottoriva. Identification of neutral tumor evolution across cancer types. *Nature genetics*, 48(3):238, 2016.

[38] Bernard J Crespi and Kyle Summers. Positive selection in the evolution of cancer. *Biological Reviews*, 81(3):407–424, 2006.

[39] Sheli L Ostrow, Ruth Barshir, James DeGregori, Esti Yeger-Lotem, and Ruth Hershberg. Cancer evolution is associated with pervasive positive selection on globally expressed genes. *PLoS genetics*, 10(3), 2014.

[40] Andrew Roth, Jaswinder Khattra, Damian Yap, Adrian Wan, Emma Laks, Justina Biele, Gavin Ha, Samuel Aparicio, Alexandre Bouchard-Côté, and Sohrab P Shah. Pyclone: statistical inference of clonal population structure in cancer. *Nature methods*, 11(4):396, 2014.

[41] Wei Jiao, Shankar Vembu, Amit G Deshwar, Lincoln Stein, and Quaid Morris. Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC bioinformatics*, 15(1):35, 2014.

[42] Amit G Deshwar, Shankar Vembu, Christina K Yung, Gun Ho Jang, Lincoln Stein, and Quaid Morris. Phylowgs: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome biology*, 16(1):35, 2015.

[43] Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the*

34th International Conference on Machine Learning-Volume 70, pages 3076–3085. JMLR. org, 2017.

[44] Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029, 2016.

[45] Mario Köppen. The curse of dimensionality. In *5th Online World Conference on Soft Computing in Industrial Applications (WSC5)*, volume 1, pages 4–8, 2000.

[46] Kate Dreher, Mireille Khairallah, Jean-Marcel Ribaut, and Michael Morris. Money matters (i): costs of field and laboratory procedures associated with conventional and marker-assisted maize breeding at cimmyt. *Molecular Breeding*, 11(3):221–234, 2003.

[47] Rangaraj M Rangayyan. *Biomedical image analysis*. CRC press, 2004.

[48] Steven JM Jones. Prediction of genomic functional elements. *Annu. Rev. Genomics Hum. Genet.*, 7:315–338, 2006.

[49] Reuben S Harris. Cancer mutation signatures, dna damage mechanisms, and potential clinical implications. *Genome Medicine*, 5(9), 2013.

[50] Ludmil B Alexandrov, Serena Nik-Zainal, David C Wedge, Peter J Campbell, and Michael R Stratton. Deciphering signatures of mutational processes operative in human cancer. *Cell reports*, 3(1):246–259, 2013.

[51] Andrej Fischer, Christopher JR Illingworth, Peter J Campbell, and Ville Mustonen. Emu: probabilistic inference of mutational processes and their localization in the cancer genome. *Genome biology*, 14(4):R39, 2013.

[52] Yuichi Shiraishi, Georg Tremmel, Satoru Miyano, and Matthew Stephens. A simple model-based approach to inferring and visualizing cancer mutation signatures. *PLoS genetics*, 11(12):e1005657, 2015.

[53] Adrian Baez-Ortega and Kevin Gori. Computational approaches for discovery of mutational signatures in cancer. *Briefings in bioinformatics*, 2017.

[54] Serena Nik-Zainal, Ludmil B Alexandrov, David C Wedge, Peter Van Loo, Christopher D Greenman, Keiran Raine, David Jones, Jonathan Hinton, John Marshall, Lucy A Stebbings, et al. Mutational processes molding the genomes of 21 breast cancers. *Cell*, 149(5):979–993, 2012.

[55] S Kasar, J Kim, R Improgo, G Tiao, P Polak, N Haradhvala, MS Lawrence, A Kiezun, SM Fernandes, S Bahl, et al. Whole-genome sequencing reveals activation-induced cytidine deaminase signatures during indolent chronic lymphocytic leukaemia evolution. *Nature communications*, 6:8866, 2015.

[56] David M Blei and Michael I Jordan. Variational inference for dirichlet process mixtures. *Journal of Bayesian Analysis*, 1(1):121–143, 2006.

[57] Rafael A Rosales, Rodrigo D Drummond, Renan Valieris, Emmanuel Dias-Neto, and Israel T Da Silva. signer: an empirical bayesian approach to mutational signature discovery. *Bioinformatics*, 33(1):8–16, 2017.

[58] J. S. Maritz and T. Lwin. Empirical bayes methods. *Mathematical Gazette*, 74(467), 1989.

[59] Julian S Gehring, Bernd Fischer, Michael Lawrence, and Wolfgang Huber. Somaticsignatures: inferring mutational signatures from single-nucleotide variants. *Bioinformatics*, 31(22):3673–3675, 2015.

[60] Yuan Gao and George Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[61] Min Zhao, Qingguo Wang, Quan Wang, Peilin Jia, and Zhongming Zhao. Computational tools for copy number variation (cnv) detection using next-generation sequencing data: features and perspectives. *BMC bioinformatics*, 14(11):S1, 2013.

[62] Philip J Hastings, James R Lupski, Susan M Rosenberg, and Grzegorz Ira. Mechanisms of change in gene copy number. *Nature Reviews Genetics*, 10(8):551–564, 2009.

[63] Jan O Korbel, Alexander Eckehart Urban, Jason P Affourtit, Brian Godwin, Fabian Grubert, Jan Fredrik Simons, Philip M Kim, Dean Palejev, Nicholas J Carriero, Lei Du, et al. Paired-end mapping reveals extensive structural variation in the human genome. *Science*, 318(5849):420–426, 2007.

[64] Zhengdong D Zhang, Jiang Du, Hugo Lam, Alex Abyzov, Alexander E Urban, Michael Snyder, and Mark Gerstein. Identification of genomic indels and structural variations using split reads. *BMC genomics*, 12(1):375, 2011.

[65] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60, 1996.

[66] Zhanyong Wang, Farhad Hormozdiari, Wen-Yun Yang, Eran Halperin, and Eleazar Eskin. Cnvem: copy number variation detection using uncertainty of read mapping. *Journal of Computational Biology*, 20(3):224–236, 2013.

[67] Christopher Yau. Oncosnp-seq: a statistical approach for the identification of somatic copy number alterations from next-generation sequencing of cancer genomes. *Bioinformatics*, 29(19):2482–2484, 2013.

[68] Sergii Ivakhno, Tom Royce, Anthony J Cox, Dirk J Evers, R Keira Cheetham, and Simon Tavaré. Cnaseg—a novel framework for identification of copy number changes in cancer from second-generation sequencing data. *Bioinformatics*, 26(24):3051–3058, 2010.

[69] Alberto Magi, Matteo Benelli, Seungtai Yoon, Franco Roviello, and Francesca Torricelli. Detecting common copy number variants in high-throughput sequencing data by using jointslm algorithm. *Nucleic acids research*, 39(10):e65–e65, 2011.

[70] Günter Klambauer, Karin Schwarzbauer, Andreas Mayr, Djork-Arne Clevert, Andreas Mitterecker, Ulrich Bodenhofer, and Sepp Hochreiter. cn. mops: mixture of poissons for discovering copy number variations in next-generation sequencing data with a low false discovery rate. *Nucleic acids research*, 40(9):e69–e69, 2012.

[71] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[72] O Alejandro Balbin, John R Prensner, Anirban Sahu, Anastasia Yocum, Sunita Shankar, Rohit Malik, Damian Fermin, Saravana M Dhanasekaran, Benjamin Chandler, Dafydd Thomas, et al. Reconstructing targetable pathways in lung cancer by integrating diverse omics data. *Nature communications*, 4(1):1–13, 2013.

[73] Kumardeep Chaudhary, Olivier B Poirion, Liangqun Lu, and Lana X Garmire. Deep learning–based multi-omics integration robustly predicts survival in liver cancer. *Clinical Cancer Research*, 24(6):1248–1259, 2018.

[74] Hanchen Xu, Chunyan Wang, Haiyan Song, Yangxian Xu, and Guang Ji. Rna-seq profiling of circular rnas in human colorectal cancer liver metastasis and the potential biomarkers. *Molecular cancer*, 18(1):8, 2019.

[75] Raphael Petegrosso, Zhuliu Li, and Rui Kuang. Machine learning and statistical methods for clustering single-cell rna-sequencing data. *Briefings in bioinformatics*, 21(4):1209–1223, 2020.

[76] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[77] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[78] Minzhe Guo, Hui Wang, S Steven Potter, Jeffrey A Whitsett, and Yan Xu. Sincera: a pipeline for single-cell rna-seq profiling analysis. *PLoS computational biology*, 11(11):e1004575, 2015.

[79] Ahmed Ashour Ahmed, Dariush Etemadmoghadam, Jillian Temple, Andy G Lynch, Mohamed Riad, Raghwa Sharma, Colin Stewart, Sian Fereday, Carlos Caldas, Anna DeFazio, et al. Driver mutations in tp53 are ubiquitous in high grade serous carcinoma of the ovary. *The Journal of pathology*, 221(1):49–56, 2010.

[80] Dara L Aisner, Lynette M Sholl, Lynne D Berry, Michael R Rossi, Heidi Chen, Junya Fujimoto, Andre L Moreira, Suresh S Ramalingam, Liza C Villaruz, Gregory A Otterson, et al. The impact of smoking and tp53 mutations in lung adenocarcinoma patients with targetable mutations—the lung cancer mutation consortium (lcmc2). *Clinical Cancer Research*, 24(5):1038–1047, 2018.

[81] Scott E Kern and Jordan M Winter. Elegance, silence and nonsense in the mutations literature for solid tumors. *Cancer biology & therapy*, 5(4):349–359, 2006.

[82] Serena Nik-Zainal, Ludmil Alexandrov, David Wedge, Peter Campbell, and Mike Stratton. Signatures of mutagenesis in mammalian cells. *Mutagenesis*, 29(6):500–500, 2014.

[83] Alexandrov, B. Ludmil, Young Seok Ju, Haase, Kerstin, Van Loo, Peter, Martincorena, I?igo, and Nik-Zainal. Mutational signatures associated with tobacco smoking in human cancer. *Science*, 2016.

[84] Kin, Chan, A Steven, Roberts, J Leszek, Klimczak, F Joan, Sterling, Natalie, and Saini. An apobec3a hypermutation signature is distinguishable from the signature of background mutagenesis by apobec3b in human cancers. *Nature Genetics*, 2015.

[85] Michael Krawczak, Edward V Ball, and David N Cooper. Neighboring-nucleotide effects on the rates of germ-line single-base-pair substitution in human genes. *The American Journal of Human Genetics*, 63(2):474–488, 1998.

[86] Allen Chieng Hoon Choong and Nung Kion Lee. Evaluation of convolutionary neural networks modeling of dna sequences using ordinal versus one-hot encoding method. In *2017 International Conference on Computer and Drone Applications (IConDA)*, pages 60–65. IEEE, 2017.

[87] Fan Wang, Pranik Chainani, Tommy White, Jin Yang, Yu Liu, and Benjamin Soibam. Deep learning identifies genome-wide dna binding sites of long noncoding rnas. *RNA biology*, 15(12):1468–1476, 2018.

[88] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[89] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[90] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[91] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[92] Mutational signatures in esophageal adenocarcinoma define etiologically distinct subgroups with therapeutic relevance. *Nature Genetics*, 2016.

[93] M Hollstein, L B Alexandrov, C P Wild, M Ardin, and J Zavadil. Base changes in tumour dna have the power to reveal the causes and evolution of cancer. *Oncogene*, 2016.

[94] Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, et al. The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303, 2010.

[95] Tom Grant et al. *Platypus*. CSIRO PUBLISHING, 2007.

[96] Rachel Rosenthal, Nicholas McGranahan, Javier Herrero, Barry S Taylor, and Charles Swanton. Deconstructsigs: delineating mutational processes in single tumors distinguishes dna repair deficiencies and patterns of carcinoma evolution. *Genome biology*, 17(1):31, 2016.

[97] The Cancer Genome Atlas Network. Comprehensive molecular characterization of human colon and rectal cancer. *Nature*, 487(7407):330, 2012.

[98] Huizhen Zhao, Fuxian Liu, Longyue Li, and Chang Luo. A novel softplus linear unit for deep convolutional neural networks. *Applied Intelligence*, 48(7):1707–1720, 2018.

[99] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[100] Serena Nik-Zainal, Helen Davies, Johan Staaf, Manasa Ramakrishna, Dominik Glodzik, Xueqing Zou, Inigo Martincorena, Ludmil B Alexandrov, Sancha Martin, David C Wedge, et al. Landscape of somatic mutations in 560 breast cancer whole-genome sequences. *Nature*, 534(7605):47–54, 2016.

[101] Christoph Lengauer, Kenneth W Kinzler, and Bert Vogelstein. Genetic instability in colorectal cancers. *Nature*, 386(6625):623, 1997.

[102] L Ottini, M Falchetti, R Lupi, P Rizzolo, V Agnese, G Colucci, V Bazan, and A Russo. Patterns of genomic instability in gastric cancer: clinical implications and perspectives. *Annals of Oncology*, 17:vii97–vii102, 2006.

[103] Neil J Ganem, Zuzana Storchova, and David Pellman. Tetraploidy, aneuploidy and cancer. *Current opinion in genetics & development*, 17(2):157–162, 2007.

[104] Travis I Zack, Steven E Schumacher, Scott L Carter, Andrew D Cherniack, Gordon Saksena, Barbara Tabak, Michael S Lawrence, Cheng-Zhong Zhang, Jeremiah Wala, Craig H Mermel, et al. Pan-cancer patterns of somatic copy number alteration. *Nature genetics*, 45(10):1134, 2013.

[105] Craig M Bielski, Ahmet Zehir, Alexander V Penson, Mark TA Donoghue, Walid Chatila, Joshua Armenia, Matthew T Chang, Alison M Schram, Philip Jonsson, Chaitanya Bandlamudi, et al. Genome doubling shapes the evolution and prognosis of advanced cancers. *Nature genetics*, 50(8):1189–1195, 2018.

[106] Anastasia Y Kuznetsova, Katarzyna Seget, Giuliana K Moeller, Mirjam S de Pagter, Jeroen ADM de Roos, Milena Dürrbaum, Christian Kuffer, Stefan

Müller, Guido JR Zaman, Wigard P Kloosterman, et al. Chromosomal instability, tolerance of mitotic errors and multidrug resistance are promoted by tetraploidization in human cells. *Cell cycle*, 14(17):2810–2820, 2015.

[107] Clemency Jolly and Peter Van Loo. Timing somatic events in the evolution of cancer. *Genome biology*, 19(1):95, 2018.

[108] Moritz Gerstung, Clemency Jolly, Ignaty Leshchiner, Stefan C Dentro, Santiago Gonzalez, Daniel Rosebrock, Thomas J Mitchell, Yulia Rubanova, Pavana Anur, Kaixian Yu, et al. The evolutionary history of 2,658 cancers. *Nature*, 578(7793):122–128, 2020.

[109] Ron Zeira and Benjamin J Raphael. Copy number evolution with weighted aberrations in cancer. *Bioinformatics*, 36(Supplement_1):i344–i352, 2020.

[110] H Nakagawa, CP Wardell, M Furuta, H Taniguchi, and A Fujimoto. Cancer whole-genome sequencing: present and future. *Oncogene*, 34(49):5943, 2015.

[111] Eric Talevich, A Hunter Shain, Thomas Botton, and Boris C Bastian. Cnvkit: genome-wide copy number detection and visualization from targeted dna sequencing. *PLoS computational biology*, 12(4):e1004873, 2016.

[112] Daniel C Koboldt, Qunyuan Zhang, David E Larson, Dong Shen, Michael D McLellan, Ling Lin, Christopher A Miller, Elaine R Mardis, Li Ding, and Richard K Wilson. Varscan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing. *Genome research*, 22(3):568–576, 2012.

[113] Jarupon Fah Sathirapongsasuti, Hane Lee, Basil AJ Horst, Georg Brunner, Alistair J Cochran, Scott Binder, John Quackenbush, and Stanley F Nelson. Exome sequencing-based copy-number variation and loss of heterozygosity detection: Exomecnv. *Bioinformatics*, 27(19):2648–2654, 2011.

[114] Ron Zeira, Meirav Zehavi, and Ron Shamir. A linear-time algorithm for the copy number transformation problem. *Journal of Computational Biology*, 24(12):1179–1194, 2017.

[115] Craig M Bielski, Ahmet Zehir, Alexander V Penson, Mark TA Donoghue, Walid Chatila, Joshua Armenia, Matthew T Chang, Alison M Schram, Philip Jonsson, Chaitanya Bandlamudi, et al. Genome doubling shapes the evolution and prognosis of advanced cancers. *Nature genetics*, 50(8):1189, 2018.

[116] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[117] Elizabeth Gibney. Deepmind algorithm beats people at classic video games. *Nature*, 518(7540):465–466, 2015.

[118] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[119] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[120] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[121] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

[122] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[123] Jiakai Zhang and Kyunghyun Cho. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

[124] Xin Xu, Chuanqiang Lian, Jian Wang, HG He, and Dewen Hu. Actor–critic reinforcement learning for autonomous control of unmanned ground vehicles. *Science Robotics*, 354:42–47, 2016.

[125] David Muse and Stefan Wermter. Actor-critic learning for platform-independent robot navigation. *Cognitive Computation*, 1(3):203–220, 2009.

[126] Kai Arulkumaran, Antoine Cully, and Julian Togelius. Alphastar: An evolutionary computation perspective. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 314–315, 2019.

[127] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[128] Maria-Iuliana Bocicor, Gabriela Czibula, and Istvan-Gergely Czibula. A reinforcement learning approach for solving the fragment assembly problem. In *2011 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 191–198. IEEE, 2011.

[129] Nicholas Navin, Jude Kendall, Jennifer Troge, Peter Andrews, Linda Rodgers, Jeanne McIndoo, Kerry Cook, Asya Stepansky, Dan Levy, Diane Esposito, et al. Tumour evolution inferred by single-cell sequencing. *Nature*, 472(7341):90, 2011.

[130] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5, 2001.

[131] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[132] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[133] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning.

In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[134] Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017.

[135] Antoine-Emmanuel Saliba, Alexander J Westermann, Stanislaw A Gorski, and Jörg Vogel. Single-cell rna-seq: advances and future challenges. *Nucleic acids research*, 42(14):8845–8860, 2014.

[136] Zhiyuan Hu, Mara Artibani, Abdulkhaliq Alsaadi, Nina Wietek, Matteo Morotti, Tingyan Shi, Zhe Zhong, Laura Santana Gonzalez, Salma El-Sahhar, Mohammad KaramiNejadRanjbar, et al. The repertoire of serous ovarian cancer non-genetic heterogeneity revealed by single-cell sequencing of normal fallopian tube epithelial cells. *Cancer cell*, 37(2):226–242, 2020.

[137] Woosung Chung, Hye Hyeon Eum, Hae-Ock Lee, Kyung-Min Lee, Han-Byoel Lee, Kyu-Tae Kim, Han Suk Ryu, Sangmin Kim, Jeong Eon Lee, Yeon Hee Park, et al. Single-cell rna-seq enables comprehensive tumour and immune cell profiling in primary breast cancer. *Nature communications*, 8(1):1–12, 2017.

[138] Sidharth V Puram, Itay Tirosh, Anuraag S Parikh, Anoop P Patel, Keren Yizhak, Shawn Gillespie, Christopher Rodman, Christina L Luo, Edmund A Mroz, Kevin S Emerick, et al. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell*, 171(7):1611–1624, 2017.

[139] Huipeng Li, Elise T Courtois, Debarka Sengupta, Yuliana Tan, Kok Hao Chen, Jolene Jie Lin Goh, Say Li Kong, Clarinda Chua, Lim Kiat Hon, Wah Siew Tan, et al. Reference component analysis of single-cell transcriptomes elucidates cellular heterogeneity in human colorectal tumors. *Nature genetics*, 49(5):708–718, 2017.

[140] Barbara Treutlein, Qian Yi Lee, J Gray Camp, Moritz Mall, Winston Koh, Seyed Ali Mohammad Shariati, Sopheak Sim, Norma F Neff, Jan M Skotheim,

Marius Wernig, et al. Dissecting direct reprogramming from fibroblast to neuron using single-cell rna-seq. *Nature*, 534(7607):391–395, 2016.

[141] Ashley Maynard, Caroline E McCoach, Julia K Rotow, Lincoln Harris, Franziska Haderk, D Lucas Kerr, A Yu Elizabeth, Erin L Schenk, Weilun Tan, Alexander Zee, et al. Therapy-induced evolution of human lung cancer revealed by single-cell rna sequencing. *Cell*, 182(5):1232–1251, 2020.

[142] Vladimir Yu Kiselev, Tallulah S Andrews, and Martin Hemberg. Challenges in unsupervised clustering of single-cell rna-seq data. *Nature Reviews Genetics*, 20(5):273–282, 2019.

[143] Hannah A Pliner, Jay Shendure, and Cole Trapnell. Supervised classification enables rapid annotation of cell atlases. *Nature methods*, 16(10):983–986, 2019.

[144] Jeffrey M. Granja, Sandy Klemm, Lisa M. Mcginnis, Arwa S. Kathiria, Anja Mezger, M. Ryan Corces, Benjamin Parks, Eric Gars, Michaela Liedtke, and Grace X. Y. and Zheng. Single-cell multiomic analysis identifies regulatory programs in mixed-phenotype acute leukemia. *Nature Biotechnology*.

[145] Emma Pierson and Christopher Yau. Zifa: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome biology*, 16(1):1–10, 2015.

[146] Vladimir Yu Kiselev, Kristina Kirschner, Michael T Schaub, Tallulah Andrews, Andrew Yiu, Tamir Chandra, Kedar N Natarajan, Wolf Reik, Mauricio Barahona, Anthony R Green, et al. Sc3: consensus clustering of single-cell rna-seq data. *Nature methods*, 14(5):483–486, 2017.

[147] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019.

[148] Dmitry Kobak and Philipp Berens. The art of using t-sne for single-cell transcriptomics. *Nature communications*, 10(1):1–14, 2019.

[149] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature methods*, 16(3):243–245, 2019.

[150] Wouter Saelens, Robrecht Cannoodt, Helena Todorov, and Yvan Saeys. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, 2019.

[151] Dominic Grün, Anna Lyubimova, Lennart Kester, Kay Wiebrands, Onur Basak, Nobuo Sasaki, Hans Clevers, and Alexander Van Oudenaarden. Single-cell messenger rna sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251–255, 2015.

[152] Manish Pratap Singh, Sandhya Rai, Ashutosh Pandey, Nand K Singh, and Sameer Srivastava. Molecular subtypes of colorectal cancer: An emerging therapeutic opportunity for personalized medicine. *Genes & Diseases*, 2019.

[153] Peijie Lin, Michael Troup, and Joshua WK Ho. Cidr: Ultrafast and accurate clustering through imputation for single-cell rna-seq data. *Genome biology*, 18(1):59, 2017.

[154] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016.

[155] Wenming Wu and Xiaoke Ma. Joint learning dimension reduction and clustering of single-cell rna-sequencing data. *Bioinformatics*, 36(12):3825–3832, 2020.

[156] Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. Warped mixtures for nonparametric cluster shapes. *arXiv preprint arXiv:1206.1846*, 2012.

[157] Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

[158] Carl Rasmussen. The infinite gaussian mixture model. *Advances in neural information processing systems*, 12:554–560, 1999.

[159] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[160] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(3):231–240, 2011.

[161] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.

[162] Liu Yaohui, Ma Zhengming, and Yu Fang. Adaptive density peak clustering based on k-nearest neighbors with aggregating strategy. *Knowledge-Based Systems*, 133:208–220, 2017.

[163] Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1):86–97, 2012.

[164] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001.

[165] Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[166] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

[167] Ryan McConville, Raul Santos-Rodriguez, Robert J Piechocki, and Ian Craddock. N2d:(not too) deep clustering via clustering the local manifold of an autoencoded embedding. *arXiv preprint arXiv:1908.05968*, 2019.

[168] Michael AA Cox and Trevor F Cox. Multidimensional scaling. In *Handbook of data visualization*, pages 315–347. Springer, 2008.

[169] Christine Bernardi and Yvon Maday. Spectral methods. *Handbook of numerical analysis*, 5:209–485, 1997.

[170] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[171] Zhenyue Zhang and Jing Wang. Mlle: Modified locally linear embedding using multiple weights. *Advances in neural information processing systems*, 19:1593–1600, 2006.

[172] Geoffrey Hinton and Sam T Roweis. Stochastic neighbor embedding. In *NIPS*, volume 15, pages 833–840. Citeseer, 2002.

[173] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[174] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 37(1):38–44, 2019.

[175] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[176] Shiquan Sun, Jiaqiang Zhu, Ying Ma, and Xiang Zhou. Accuracy, robustness and scalability of dimensionality reduction methods for single-cell rna-seq analysis. *Genome biology*, 20(1):269, 2019.

[177] Etienne Becht, Leland McInnes, John Healy, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology*, 37(1):38–44, 2019.

[178] Kelly Street, Davide Risso, Russell B Fletcher, Diya Das, John Ngai, Nir Yosef, Elizabeth Purdom, and Sandrine Dudoit. Slingshot: cell lineage and pseudotime inference for single-cell transcriptomics. *BMC genomics*, 19(1):477, 2018.

[179] F. Alexander Wolf, Fiona K. Hamey, Mireya Plass, Jordi Solana, Joakim S. Dahlin, Berthold Göttgens, Nikolaus Rajewsky, Lukas Simon, and Fabian J. Theis. Paga: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome Biology*, 20(1), 2019.

[180] Laleh Haghverdi, Maren Büttner, F Alexander Wolf, Florian Buettner, and Fabian J Theis. Diffusion pseudotime robustly reconstructs lineage branching. *Nature methods*, 13(10):845, 2016.

[181] Neil Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16:329–336, 2003.

[182] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[183] Justin Solomon. *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. 2015.

[184] Lukas Meier, Sara Van De Geer, and Peter Bühlmann. The group lasso for logistic regression. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 70(1):53–71, 2008.

[185] Umut Ozertem and Deniz Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Learning Research*, 12(34):1249–1286, 2011.

[186] Umut Ozertem and Deniz Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Learning Research*, 12(34):1249–1286, 2011.

[187] Slav Kirov and Dejan Slepčev. Multiple penalized principal curves: Analysis and computation. *Journal of Mathematical Imaging and Vision*, 59(2):234–256, 2017.

[188] Fei Zhou, Ting Yu, Ronghui Du, Guohui Fan, Ying Liu, Zhibo Liu, Jie Xiang, Yeming Wang, Bin Song, Xiaoying Gu, et al. Clinical course and risk factors for mortality of adult inpatients with covid-19 in wuhan, china: a retrospective cohort study. *The lancet*, 2020.

[189] J. F. Daoust. Elderly people and responses to covid-19 in 27 countries. *PLOS ONE*, 15(7), 2020.

[190] Bradley Efron and Robert Tibshirani. The bootstrap method for assessing statistical accuracy. *Behaviormetrika*, 12(17):1–35, 1985.

[191] Claus Meyer, E Kowarz, J Hofmann, A Renneville, J Zuna, Jan Trka, R Ben Abdelali, Elizabeth Macintyre, E De Braekeleer, M De Braekeleer, et al. New insights to the mll recombinome of acute leukemias. *Leukemia*, 23(8):1490–1499, 2009.

[192] Mark Holland, Fernanda V Castro, Seema Alexander, Duncan Smith, Jizhong Liu, Michael Walker, Danny Bitton, Kate Mulryan, Garry Ashton, Morgan Blaylock, et al. Rac2, aep, and icam1 expression are associated with cns disease in a mouse model of pre-b childhood acute lymphoblastic leukemia. *Blood, The Journal of the American Society of Hematology*, 118(3):638–649, 2011.

[193] Xiao-Ling Liu, Jin-Yun Yuan, Jun-Wu Zhang, Xin-Hua Zhang, and Rong-Xin Wang. Differential gene expression in human hematopoietic stem cells specified toward erythroid, megakaryocytic, and granulocytic lineage. *Journal of leukocyte biology*, 82(4):986–1002, 2007.

[194] N. Cerveira, F. Micci, J. Santos, M. Pinheiro, C. Correia, S. Lisboa, S. Bizarro, L. Norton, A. Glomstein, and A. E. and Asberg. Molecular characterization of the mll-sept6 fusion gene in acute myeloid leukemia: identification of novel fusion transcripts and cloning of genomic breakpoint junctions. *Haematologica*, 93(7):1076–1080, 2008.

[195] Thangarajan Rajkumar, Kesavan Sabitha, Neelakantan Vijayalakshmi, Sundersingh Shirley, and Ganesharaja Selvaluxmy. Identification and validation of genes involved in cervical tumourigenesis. *Bmc Cancer*, 11(1):80, 2011.

[196] Ti Meng, Leichao Liu, Ruifang Hao, Siying Chen, and Yalin Dong. Transgelin-2: A potential oncogenic factor. *Tumour Biology the Journal of the International Society for Oncodevelopmental Biology Medicine*, 39(6):1010428317702650, 2017.

[197] Ferial Amira Slim, Geneviève Ouellette, Kaoutar Ennour-Idrissi, Simon Jacob, Caroline Diorio, and Francine Durocher. An isoform of aif1 involved in breast cancer. *Cancer Cell International*, 18(1), 2018.

[198] Luiza Handschuh, Maciej Ka?Mierczak, Marek Milewski, Micha Gralski, Magdalena ?uczak, Marzena Wojtaszewska, Barbara Uszczyńska-Ratajczak, Krzysztof Lewandowski, Mieczys?Aw Komarnicki, and Marek Figlerowicz. Gene expression profiling of acute myeloid leukemia samples from adult patients with aml-m1 and -m2 through boutique microarrays, real-time pcr and droplet digital pcr. *International Journal of Oncology*, 2018.

[199] Emira Ayroldi, Lorenza Cannarile, Domenico V. Delfino, and Carlo Riccardi. A dual role for glucocorticoid-induced leucine zipper in glucocorticoid function: tumor growth promotion or suppression? *Cell Death Disease*, 9(5):463, 2018.

[200] M. Porcu, M. Kleppe, V. Gianfelici, E. Geerdens, K. De Keersmaecker, M. Tartaglia, R. Foa, J. Soulier, B. Cauwelier, and A. Uyttebroeck. Mutation of the receptor tyrosine phosphatase ptprc (cd45) in t-cell acute lymphoblastic leukemia. *Blood*, 119(19):4476–4479, 2012.

[201] Alicia Torriglia, Elisabeth Martin, and Imene Jaadane. The hidden side of serpinb1/leukocyte elastase inhibitor. *Seminars in Cell Developmental Biology*, 62:178–186, 2017.

[202] B Mizukawa, J Wei, M Shrestha, M Wunderlich, F. S. Chou, A Griesinger, C. E. Harris, A. R. Kumar, Y Zheng, and D. A. Williams. Inhibition of rac gtpase signaling and downstream prosurvival bcl-2 proteins as combination targeted therapy in mll-af9 leukemia. *Blood*, 118(19):5235–5245, 2011.

[203] M. J. Pont, W. Hobo, M. W. Honders, S. A. P. Van Luxemburg-Heijs, M. G. D. Kester, A. M. Van Oeveren-Rietdijk, N. Schaap, H. C. De Boer, C. A. M. Van Bergen, and H. and Dolstra. Lb-arhgdib-1r as a novel minor histocompatibility antigen for therapeutic application. *Haematologica*, 100(10), 2015.

[204] George C. Linderman, Manas Rachh, Jeremy G. Hoskins, Stefan Steinerberger, and Yuval Kluger. Fast interpolation-based t-sne for improved visualization of single-cell rna-seq data. *Nature Methods*, 16(3):1, 2019.

[205] Diego Peteiro-Barral and Bertha Guijarro-Berdiñas. A survey of methods for distributed machine learning. *Progress in Artificial Intelligence*, 2(1):1–11, 2013.

[206] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[207] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes, 2019.

[208] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. *ACM sigmod record*, 25(2):103–114, 1996.

[209] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[210] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.