A Multilayer Perceptron Framework for Sparse Multiple Kernel Learning

by

Binnur Şahin

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ ÜNİVERSİTESİ

September, 2023

A Multilayer Perceptron Framework for Sparse Multiple Kernel Learning

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Binnur Şahin

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Mehmet Gönen (Advisor)

_____

Assist. Prof. Barış Akgün

_____

Prof. Mehmet Güray Güler

Date: _____

To my dear family.

ABSTRACT

A Multilayer Perceptron Framework for Sparse Multiple Kernel Learning

Binnur Şahin

Master of Science in Computer Science and Engineering

September, 2023

Advancing cancer research and improving patient care heavily rely on understanding how biological mechanisms change as cancer progresses. Identifying the active biological processes within tumors is crucial for developing targeted treatments and potential biomarkers for early detection and prognosis. In this thesis, we focused on distinguishing between early-stage and late-stage cancers using gene expression profiles, aiming to develop a powerful and interpretable model. Machine learning methods have shown promise in cancer research by enabling the analysis of vast genomic and clinical data to uncover hidden patterns and predictive features. Our work centered around harnessing the capabilities of a multilayer perceptron (MLP) model to create a sparse solution within the context of multiple kernel learning (MKL). This enabled our model to proficiently differentiate between early-stage and late-stage cancers based on gene expression profiles. Our model not only offered high predictive performance but also provided valuable insights into the crucial genes and pathways driving cancer progression. To evaluate our MLP model, we benchmarked it against three well-established machine learning algorithms: random forest, support vector machine, and MKL. Remarkably, our model consistently achieved better or comparable predictive performance, as measured by the area under the receiver operating characteristic curve, across 15 cancer cohorts. The findings demonstrate that our proposed MLP model effectively identifies critical genes and pathways driving cancer progression, offering valuable insights into early-stage and late-stage cancer classification.

ÖZETÇE

Seyrek Çoklu Çekirdek Öğrenimi için Çok Katmanlı Bir Perseptron Çerçevesi
Binnur Şahin
Bilgisayar Bilimleri ve Mühendisliği, Yüksek Lisans
Eylül 2023

Kanser araştırmalarını ilerletmek ve hasta bakımını iyileştirmek, büyük ölçüde kanser ilerledikçe biyolojik mekanizmaların nasıl değiştiğinin anlaşılmasına bağlıdır. Tümörlerdeki aktif biyolojik süreçlerin belirlenmesi, hedefe yönelik tedavilerin ve erken teşhis ve prognoz için potansiyel biyobelirteçlerin geliştirilmesi açısından çok önemlidir. Bu tezde, gen ekspresyon profillerini kullanarak güçlü ve yorumlanabilir bir model geliştirme amacıyla erken evre ve geç evre kanserleri ayırt etmeye odaklandık. Yapay öğrenme yöntemleri, gizli kalıpları ve kestirimci özellikleri ortaya çıkarmak için geniş genomik ve klinik verilerin analiz edilmesini sağlayarak kanser araştırmalarında umut vaat etmektedir. Çalışmamız, çoklu çekirdek öğrenimi bağlamında seyrek bir çözüm oluşturmak için çok katmanlı bir perseptron modelinin yeteneklerinden yararlanmaya odaklandı. Bu, modelimizin gen ifade profillerine dayalı olarak erken evre ve geç evre kanserler arasında yetkin bir ayrım yapmasını sağladı. Modelimiz sadece yüksek tahmin performansı sunmakla kalmadı, aynı zamanda kanserin ilerlemesini yönlendiren önemli genler ve yolaklar hakkında değerli bilgiler sağladı. Çok katmanlı perseptron modelimizi değerlendirmek için, onu üç köklü yapay öğrenme algoritmasıyla karşılaştırdık: rassal orman, destek vektör makinesi ve çoklu çekirdek öğrenimi. Dikkat çekici bir şekilde, modelimiz 15 kanser kohortunda alıcı işletim karakteristik eğrisi altındaki alan ile ölçüldüğü üzere tutarlı bir şekilde daha iyi veya karşılaştırılabilir kestirim performansı elde etti. Bu bulgular, çok katmanlı perseptron modelimizin, kanserin ilerlemesini yönlendiren kritik genleri ve yolakları etkili bir şekilde tanımladığını ve erken evre ve geç evre kanser sınıflandırmasına ilişkin değerli bilgiler sunduğunu göstermektedir.

ACKNOWLEDGMENTS

I would like to extend my heartfelt gratitude to my thesis advisor Prof. Mehmet Gönen for his invaluable guidance, remarkable support and boundless patience throughout my research journey. I consider myself exceptionally fortunate to have had the privilege of working with him and learning from him.

I am grateful to Assist. Prof. Barış Akgün and Prof. Mehmet Güray Güler for accepting to be part of my thesis committee. Their willingness to contribute their expertise and guidance to my research is genuinely appreciated.

Finally, I would like to thank my family and friends for their unwavering support and encouragement throughout my academic journey. I am truly grateful for your presence in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ANN	Artificial Neural Network
AUROC	Area Under the Receiver Operating Characteristics Curve
DGCN	Directed Graph Convolutional Network
FP	False Positive
FPR	False Positive Rate
IMPSO	Improved Particle Swarm Optimization
LASSO	Least Absolute Shrinkage and Selection Operator
MLP	Multilayer Perceptron
MKL	Multiple Kernel Learning
MSigDB	Molecular Signatures Database
ReLU	Rectified Linear Unit
RF	Random Forest
RMSE	Root Mean Square Error
RNA-seq	RNA-sequencing
RNN	Recurrent Neural Networks
ROC	Receiver Operating Characteristics Curve
SVM	Support Vector Machine
TCGA	The Cancer Genome Atlas
TNM	Tumor, Nodes, and Metastasis
TP	True Positive

Chapter 1

INTRODUCTION

Cancer remains a significant public health challenge worldwide. It is characterized by the uncontrolled growth and spread of abnormal cells, which lead to diverse forms of malignancies with varying degrees of severity. The complexity of cancer and its significant impact on lives require continuous efforts to understand its mechanisms and improve patient outcomes. Cancer research plays a crucial role in guiding us toward innovative strategies to combat this disease.

The field of cancer research has witnessed remarkable advancements in recent decades, which are fueled by advancements in technology, particularly in genomics and computational sciences. Understanding how biological mechanisms change during cancer progression is crucial for identifying important biomarkers and molecular pathways that drive the advancement of the disease. Therefore, it is essential to investigate how the genomic aspects evolve to develop targeted treatments. Genomics has provided valuable insights into the genomic alterations associated with cancer initiation, progression, and metastasis. The availability of vast amounts of genomic data from various cancer types has opened up new possibilities for understanding the genetic basis of cancer and identifying potential therapeutic targets.

Early cancer diagnosis is fundamental to improving patient outcomes and survival rates. Detecting cancer at its early stages can significantly impact treatment success and patient well-being. Consequently, there is a need for novel methodologies that enhance early detection.

In the pursuit of novel insights, the integration of machine learning methods has emerged as a powerful and promising tool in cancer research. The advent of high-throughput technologies, such as next-generation sequencing and microarray

technologies, has led to the generation of vast amounts of genomic data. The machine learning models can effectively analyze extensive genomic data sets, identifying subtle patterns and correlations. This integration of genomics and machine learning has the potential to offer a more comprehensive understanding of cancer biology. It can lead to enhanced diagnostic accuracy and the development of personalized treatment strategies. However, using machine learning in cancer research comes with challenges. One of the main obstacles is understanding how prominent machine learning models make decisions. As we aim to incorporate these powerful tools into real-world medical practices, transparency and interpretability become essential to gain trust and wider acceptance.

In this study, we propose a novel multilayer perceptron (MLP) model capable of accurately classifying early- and late-stage cancers based on gene expression profiles. The proposed model aims to achieve high-performance classification while simultaneously providing valuable insights into the underlying biological mechanisms governing cancer progression.

We organized this dissertation as follows: In Chapter 1, we described the problem focused on this study and the existing work related to the problem. In Chapter 2, we presented a background to the data set, baseline methods, and MLPs. Chapter 3 described our proposed method in this study. The computational experiments were described in Chapter 4. The conclusions were summarized in Chapter 5.

1.1 Problem Definition

Over the course of a lifetime, multiple tumors may arise in the body. However, not all of these tumors progress into life-threatening diseases. Therefore, comprehending the factors influencing the transition of cancers from early-stage to late-stage tumors becomes an important pursuit. Understanding these underlying mechanisms is crucial for advancing cancer research and improving patient outcomes. Thus, our work focused on the problem of discriminating early- and late-stage cancers based on the genomic data for a tumor.

1.2 Existing Work

This study centers around the problem of differentiating early- and late-stage cancers based on their gene expression profiles. Similar investigations have been conducted in prior studies. To provide a comprehensive overview of the existing research, we present a review of relevant literature in two sections. First, we examine prior studies concerning cancer stage prediction using gene expression data. Subsequently, we delve into the literature that explores the application of MLP models for various classification tasks based on genomic data.

1.2.1 Cancer Stage Prediction

In this section, we present a review of existing work that has focused on the problem of cancer stage prediction based on gene expression profiles. The studies covered in this review have taken various approaches to tackle this problem, ranging from statistical score-based methodologies and standard machine learning algorithms to innovative models that incorporate different approaches.

Broët et al. (2006) attempted to detect gene expression characteristics that distinguish early-stage breast cancer from late-stage breast cancer. They analyzed microarray data to identify specific gene expression patterns associated with each stage of the disease by employing a statistical score-based approach.

Jagga and Gupta (2014) and Bhalla et al. (2017) constructed separate algorithms to identify specific genes that can distinguish early-stage patients from late-stage patients in a disease called kidney renal clear cell carcinoma. They, respectively, used correlation-based and threshold-based approaches to select these genes and then tested their quality by training standard machine learning algorithms like support vector machines (SVMs) and random forests (RFs).

Singh et al. (2018) worked on a similar problem and employed a machine learning pipeline to discern the early- and late-stages of papillary renal cell carcinoma based on gene expression profiles. Their pipeline incorporated various feature selection algorithms and classification models. Likewise, Kaur et al. (2019) worked on the problem of classifying patients with liver hepatocellular carcinoma into early-

and late-stages based on their genomics and epigenomics profiles. They applied threshold-based feature selection methods and trained machine learning algorithms based on selected features.

To discriminate the stages of the clear cell renal cell carcinoma disease, Li et al. (2020) followed a similar approach by designing a machine learning pipeline consisting of feature selection and prediction stages. They combined correlation-based and threshold-based feature selection algorithms in their pipeline and trained classification algorithms.

Although these scoring/thresholding metrics can potentially pinpoint gene expression characteristics that are indicative of predictions using only a restricted set of features, their interpretation poses challenges because of the high-dimensional and correlated nature of the input data. In scenarios with numerous variables, machine learning algorithms may vary in their selection of biomarkers as predictors when applied to distinct portions of the identical group of patients for a specific prediction task (Ein-Dor et al., 2005, 2006).

In contrast to the studies mentioned earlier, Rahimi and Gönen (2018) took a unique approach using pathways/gene sets to create a unified model. Their approach involved creating individual kernel matrices for each gene set and combining them through a multiple kernel learning (MKL) algorithm known as group Lasso MKL. They then compared the performance of their model against RF and SVM algorithms across 20 different cancer cohorts. In their subsequent work, Rahimi and Gönen (2020) continued their investigation into cancer stage discrimination using a multitask MKL (MTMKL) approach to model all cancer cohorts jointly. Then, they used a co-clustering step to group together similar cancer cohorts and gene sets based on their underlying mechanisms. Finally, they iteratively solve the MTMKL and co-clustering problems. Their works highlight the importance of incorporating biological knowledge in cancer research.

Taking a different route, Ma et al. (2020) used an XGBoost classification technique for the task of discriminating the early- and late-stages of four cancers, which are kidney renal clear cell carcinoma, kidney renal papillary cell carcinoma, lung squamous cell carcinoma, and head and neck squamous cell carcinoma, based on

gene expression data. To gain insight into the contribution of genes to the prediction, they used the feature importance scores of their trained model.

1.2.2 Multilayer Perceptron Models for Cancer Classification

Several studies have demonstrated the promising potential of MLP models in cancer classification tasks using genomic data. These studies have reported encouraging results, showcasing the effectiveness of MLP-based approaches in this domain. In this context, we provide a brief overview of several investigations that employed the MLP algorithm.

Xie et al. (2016) worked on the problem of predicting gene expression levels based on genetic variations in the genome. For this aim, they proposed a regression model based on MLP and Stacked Denoising Auto-encoder. The MLP component allows for effective feature extraction and learning of complex patterns in the input genomic data. On the other hand, the Stacked Denoising Auto-encoder helps to denoise the input data to enhance the generalization ability of the model.

Xu et al. (2018) focused on predicting the subtypes of breast cancer, which are Basal, Her2, Luminal A, and Luminal B, using gene expression profiles. They first use a subtype-dependent feature selection method to reduce the dimensionality of the gene expression data and train different machine learning models on the selected genes, such as k-nearest neighbors, SVM, MLP, and multi-grained cascade forest.

Pati (2018) worked on the problem of identifying specific biomarkers associated with lung cancer development at an early-stage based on the gene expression data. They combined a feature selection method named the info gain ranking with a machine learning classifier. As classifiers, they used MLP, SVM, and an ensemble of classifiers formed using random subspace.

Working on a microarray data set, Azzawi et al. (2019) proposed a hybrid neural network approach for lung cancer classification. In their approach, they combine two sources of gene importances, one from using χ^2-statistics and another from previously published studies, with prior biological knowledge of lung adenocarcinoma-related genes. They then build an MLP model using the selected genes as input

values. While training the MLP, they utilize improved particle swarm optimization (IMPSO) as the training algorithm instead of the commonly used traditional back-propagation. The particles represent candidate solutions (sets of weights) for MLP, and IMPSO optimizes these weights to find the best set that minimizes the mean absolute error. The IMPSO-MLP model is trained iteratively until convergence.

To find a set of genes that are associated with cancer prediction or diagnosis using gene expression profile data, Seo and Cho (2020) designed a feature selection algorithm. Their proposed model combines boosted regression and MLP algorithms. The model sequentially selects relevant genes using regression-based feature selection and trains an MLP with these selected genes. The process is iterated to find an optimal feature subset that enhances cancer classification performance.

Zhang et al. (2022) also focused on finding genes that are drivers of cancer, working on multi-omics genomic data. They jointly utilize a directed graph convolutional network (DGCN) and an MLP in their approach. The DGCN learns gene embeddings in a directed gene regulatory network along with genomic data. The MLP, on the other hand, takes in the genomic data and maps it to another set of gene embeddings using an artificial neural network. The outputs from both the DGCN and MLP are then combined and fed into a fully connected neural network layer to predict the probability of each gene being a cancer-driver gene.

Chapter 2

BACKGROUND

This chapter provides the background of our study. Initially, we introduce the data set employed within this research. Subsequently, we outline the baseline methods, which are RFs, SVMs, and MKL. Lastly, we explore the domain of artificial neural networks (ANNs), with a focus on MLPs.

2.1 Data Set

We made use of a diverse range of cancer cohorts derived from The Cancer Genome Atlas (TCGA) project. The TCGA project was initiated in 2005 with the aim of conducting a comprehensive examination of the genomic changes that form the basis of cancer (The Cancer Genome Atlas Research Network, 2008). Over the years, it has become a precious resource for researchers in their cancer studies. By providing massive amounts of genomic data, it enhanced our comprehension of the disease and facilitated the development of novel cancer treatments.

The cancer cohorts that we used are publicly available at `https://portal.gdc.cancer.gov`.

2.1.1 RNA-Seq Measurements

To delve into the gene expression landscape within different cancer types, we relied on RNA-sequencing (RNA-Seq). RNA-Seq is a powerful sequencing technique employed to study and quantify gene expression levels within a biological sample (Wang et al., 2009). There are RNA-Seq measurements of 33 cancer types covering more than 10,000 tumors presented by the TCGA project. These measurements are pre-processed to ensure consistency and comparability of the data, resulting in the establishment of a unified RNA-Seq pipeline.

Our study focused solely on primary tumors, excluding metastatic tumors due to their unique biological mechanisms distinct from primary tumors.

2.1.2 Pathological Staging

The pathological stage of cancer provides valuable information about the size of the tumor, its location, and whether it has spread to other parts of the body. Assessing the stage of cancer is an important aspect of determining the prognosis and planning the appropriate treatment for patients.

The classification of cancer stages follows the guidelines provided by authoritative organizations like the American Joint Committee on Cancer and the International Union Against Cancer. The Tumor, Nodes, and Metastasis (TNM) Classification of Malignant Tumors is a commonly used system for staging cancer, which is collaboratively created and regularly updated by these organizations (O'Sullivan et al., 2017). The TNM system involves three key components: T, N, and M. T indicates the size of the original tumor and its extent into nearby tissues; N represents the involvement of nearby lymph nodes; and M describes whether the cancer has metastasized to distant parts of the body.

Based on the extent of cancer spread, cancer stages are typically categorized into five groups, which progress from Stage 0 to Stage IV. Stage 0 refers to the early formation of abnormal cells in the tissue, while Stage I denotes small tumors that have not spread beyond a localized area. As the cancer progresses, Stage II signifies significant growth of the tumor without metastasis. In Stage III, the tumor has further advanced and possibly spread to surrounding tissues or nearby lymph nodes. Finally, in Stage IV, known as the metastatic stage, the cancer has spread from its original site to other organs or distant parts of the body.

2.1.3 Data Set Creation

Our data set contains RNA-Seq measurements derived from tumor samples belonging to diverse cancer cohorts. We labeled the tumor samples with Stage I as the early-stage and others with Stage II, III, or IV as the late-stage cancers.

Table 2.1: Summary of 15 TCGA cancer cohorts that we used in our study.

Cohort	**Disease name**	**Stage I**	**Stage II**	**Stage III**	**Stage IV**	**Early**	**Late**	**Total**
BRCA	Breast invasive carcinoma	181	619	247	20	181	886	1067
COAD	Colon adenocarcinoma	75	176	128	64	75	368	443
ESCA	Esophageal carcinoma	16	69	49	8	16	126	142
HNSC	Head and neck squamous cell carcinoma	25	70	78	259	25	407	432
KICH	Kidney chromophobe	20	25	14	6	20	45	65
KIRC	Kidney renal clear cell carcinoma	265	57	123	82	265	262	527
KIRP	Kidney renal papillary cell carcinoma	172	21	51	15	172	87	259
LIHC	Liver hepatocellular carcinoma	171	86	85	5	171	176	347
LUAD	Lung adenocarcinoma	274	121	84	26	274	231	505
LUSC	Lung squamous cell carcinoma	244	162	84	7	244	253	497
PAAD	Pancreatic adenocarcinoma	21	146	3	4	21	153	174
READ	Rectum adenocarcinoma	30	51	51	24	30	126	156
STAD	Stomach adenocarcinoma	53	111	150	38	53	299	352
TGCT	Testicular germ cell tumours	55	12	14	0	55	26	81
THCA	Thyroid carcinoma	281	52	112	55	281	219	500
					Total	1883	3664	5547

In the process of data set creation, we made a decision to include only those cancer cohorts that had a sufficiently large sample size to ensure statistical robustness. Therefore, we excluded the cancer cohorts with less than 20 samples in at least one of the early- or late-stage classes. This resulted in the inclusion of 15 different cancer cohorts out of 33 cohorts provided by the TCGA database. The information about the cohorts that we used is summarized in Table 2.1.

The size of the cohorts in our data set ranges from 65 to 1067. The largest cohort is the BRCA cohort, with a total of 1067 primary tumor samples, whereas the smallest is the KICH cohort, with only 65 tumors. When we assess the proportion of tumors diagnosed at an early-stage relative to the total number of tumors in each cohort, the HNSC cohort has the lowest proportion of early-stage tumors with a percentage of 5.79% while the KICH cohort has the highest with a percentage of 67.90%.

2.1.4 Gene Set Database

Our primary objective is not only to predict the stages of tumors but also to gain insight into the biological differences between early- and late-stage cancers. To

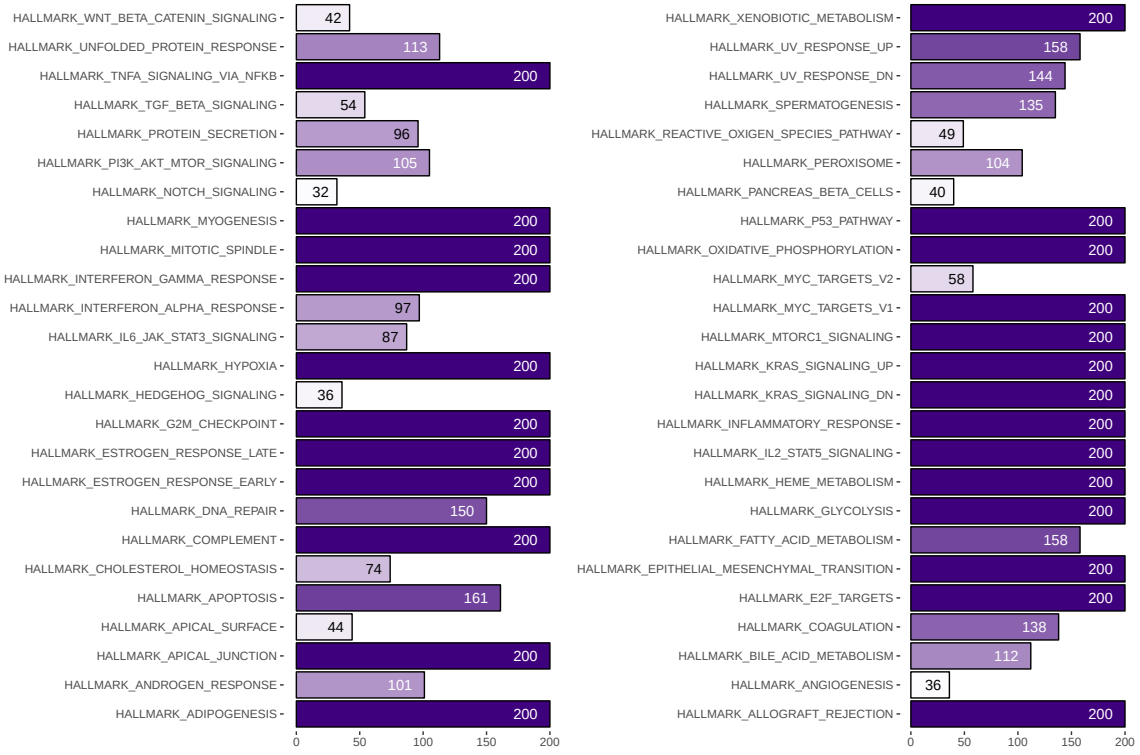

Figure 2.1: The pathways in the Hallmark collection with their names and sizes.

achieve this, we utilize a gene set collection. There are valuable pathways and gene set collections in the established literature that offer important insights into groups of functionally related genes. For our investigation, we focused on the Hallmark gene set collection.

The Hallmark gene set collection is a curated compilation of gene sets, each of which characterizes specific and well-defined biological states or processes (Liberzon et al., 2015). It has been compiled by making use of over 10000 gene sets from the Molecular Signatures Database (MSigDB) with the aim of reducing redundancy among the gene sets found in MSigDB. There are a total of 50 pathways in the Hallmark data set, as shown in Figure 2.1. The size of the pathways ranges from 32 genes to 200 genes.

2.2 Methods

2.2.1 Random Forest

RF is an ensemble learning algorithm that is used for both classification and regression tasks (Breiman, 2001). It has gained immense popularity due to its ability to deliver accurate and robust predictive models across a wide range of domains. It is based on the idea of combining multiple decision trees to create a more robust and accurate predictive model.

In an RF, several decision trees are constructed using different subsets of the data and features. By building multiple trees, each with a slightly different perspective on the data, the RF aims to create a more balanced and accurate prediction. When it comes to making predictions, each individual decision tree generates its own prediction. The final prediction of the RF is determined by aggregating the individual predictions from all the decision trees. This aggregation process allows the RF to capture a broader range of patterns and relationships in the data and improves its predictive power.

RFs have found extensive applications in various domains, including genomics. They have been used widely for predicting disease phenotypes based on genomic measurements (Diaz-Uriarte and Alvarez de Andrés, 2006; Pang et al., 2006; Statnikov et al., 2008; Nam et al., 2009). They are a potential approach for analyzing genomic data, but they have limitations. Although they provide accurate predictions, grasping the genetic factors that underlie these predictions can be challenging due to the complex nature of the ensemble. Genomic data frequently includes a multitude of features, which can lead to inefficiencies. Additionally, the model might exhibit a bias towards dominant features, potentially overlooking essential but slight genetic variations.

2.2.2 Support Vector Machine

The SVM is a discriminant-based supervised learning algorithm proposed to solve binary classification problems (Cortes and Vapnik, 1995). Assume that $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ is a set of training instances, where $\boldsymbol{x}_i$ is the input vector having D dimensions and

$y_i \in \{-1, +1\}$ is its class label. SVM determines a linear discriminant by maximizing the margin in the feature space. The discriminant function can be formulated as:

$$f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b. \tag{2.1}$$

SVM finds the discriminant function by solving the following convex quadratic optimization problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i \\ \text{with respect to} \quad & \boldsymbol{w} \in \mathbb{R}^D, \quad \boldsymbol{\xi} \in \mathbb{R}^N, \quad b \in \mathbb{R} \\ \text{subject to} \quad & y_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) \geq 1 - \xi_i \quad \forall i \\ & \xi_i \geq 0 \quad \forall i, \end{aligned} \tag{2.2}$$

where $\boldsymbol{w}$ represents the weights associated with the features, C is a positive regularization parameter, $\boldsymbol{\xi}$ denotes the slack variables, and b is the intercept parameter.

Another way to find the discriminant function is to solve the corresponding dual problem of this primal optimization problem. Solving the dual problem has two main advantages: It decreases the number of decision variables and enables us to integrate kernel functions into the model, resulting in the creation of non-linear models.

To derive the dual problem, we first find the derivatives of the Lagrangian function with respect to the decision variables of the primal problem and set them equal to 0. By taking $\boldsymbol{\alpha}$ as the vector of dual variables assigned to each separation constraint in the primal problem, the Lagrangian function of the primal problem can be written as the following:

$$\mathcal{L} = \frac{1}{2}\boldsymbol{w}^\top \boldsymbol{w} + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\boldsymbol{w}^\top \boldsymbol{x}_i + b\right) - 1 + \xi_i\right) - \sum_{i=1}^{N} \beta_i \xi_i. \tag{2.3}$$

After taking the derivative of the Lagrangian function with respect to the decision

variables and setting them equal to 0, we obtain the following functions:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} = 0 \quad &\Rightarrow \quad \boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i \\
\frac{\partial \mathcal{L}}{\partial b} = 0 \quad &\Rightarrow \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \\
\frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \quad &\Rightarrow \quad C = \alpha_i + \beta_i \quad \forall i
\end{aligned}$$

Plugging these back into the Lagrangian function gives the objective value of the dual problem. Then, we obtain the following dual formulation:

$$\begin{aligned}
\text{minimize} \quad & -\sum_{i=1}^{N} \alpha_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i^\top \boldsymbol{x}_j \\
\text{with respect to} \quad & \boldsymbol{\alpha} \in \mathbb{R}^N \\
\text{subject to} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\
& C \geq \alpha_i \geq 0 \quad \forall i.
\end{aligned} \tag{2.4}$$

The dual problem involves N decision variables, making it computationally efficient compared to the primal problem, which has $(D + N + 1)$ decision variables. Furthermore, the dual formulation allows the integration of kernel functions, enabling the SVM to handle non-linearly separable data. A kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$, can be plugged into the objective function of the dual problem instead of $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ term to map the data into a higher-dimensional feature space, where linear separation becomes possible.

After solving this dual problem, we obtain $\boldsymbol{w} = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i$ and the discriminant function can be reformulated as the following:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x} + b. \tag{2.5}$$

If $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ term is replaced with the kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, then the discriminant

function is rewritten as the following:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{2.6}$$

There are different kernel functions used in the literature, each suitable for different types of data and classification problems. Some commonly used kernel functions include the linear kernel $k_{LIN}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the polynomial kernel $k_{POL}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and the Gaussian kernel $k_{GAU}(\boldsymbol{x}_i, \boldsymbol{x}_j)$:

$$\begin{aligned}
k_{LIN}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \boldsymbol{x}_i^\top \boldsymbol{x}_j \\
k_{POL}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\boldsymbol{x}_i^\top \boldsymbol{x}_j + c)^q, \quad q \in \mathbb{N} \\
k_{GAU}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2s^2}\right), \quad s \in \mathbb{R}_{++}
\end{aligned}$$

where c is a given parameter, q is the degree of the polynomial, and s is the kernel width parameter.

The selection of the kernel function used and its parameters is an important decision. One can either directly select a kernel function that was proposed to be used for particular applications or select the best-performing kernel function by performing cross-validation among a set of possible kernel functions on a separate validation set.

2.2.3 Multiple Kernel Learning

MKL methods combine multiple kernel functions instead of selecting one particular kernel function (Gönen and Alpaydın, 2011). These methods aim to combine the strengths of multiple kernel functions to enhance the performance of a learning algorithm. Instead of utilizing a single kernel, MKL methods optimally weigh and aggregate multiple kernels, which can capture different representations of the data or different notions of similarity. This approach is particularly useful when the data is heterogeneous or lies in different subspaces, where a single kernel may not fully capture the underlying patterns. Given a combination function $f : \mathbb{R}^P \to \mathbb{R}$, we can

write the combined kernel function as

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = f(k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)_{m=1}^P), \tag{2.7}$$

where P represents the number of kernel functions used in the combination and f is the combination function that can be linear or nonlinear. The choice of the combination function depends on the specific learning task and the properties of the individual kernels. If the combination function is a linear function $f_\eta : \mathbb{R}^P \to \mathbb{R}$, it can be linearly parameterized into a form of

$$k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_\eta(k_m(\boldsymbol{x}_i^m, \boldsymbol{x}_j^m)_{m=1}^P = \sum_{m=1}^{P} \eta_m k_m\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right), \tag{2.8}$$

where $\boldsymbol{\eta}$ represents the kernel weights.

MKL algorithms differentiate in terms of which parameters of the combined kernel function are optimized. Most of the algorithms consider the kernel functions and their parameters as fixed and optimize the parameters of the combination function. Some of them consider the parameters of the combination function as fixed and optimize the parameters of the kernel functions.

For the problem of differentiating the stages of cancer, an MKL algorithm incorporated with a group Lasso regularization has been used in the literature (Rahimi and Gönen, 2018). In their approach, they utilized a pathway/gene set database while forming the kernel matrices. For each gene set, they formed a distinct kernel matrix and combined these kernel matrices using group Lasso MKL.

In order to solve the problem in the Equation 2.4, group Lasso MKL replaces $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ with $k_\eta\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ benefiting a linear combination function as described in Equation 2.8. Then, it solves this problem by enforcing ℓ_1-norm on the kernel weights. Thus, the optimization problem solved by group Lasso MKL is as follows:

$$\begin{aligned} \text{minimize} \quad & J(\boldsymbol{\eta}) \\ \text{with respect to} \quad & \boldsymbol{\eta} \in \mathbb{R}^P \\ \text{subject to} \quad & \sum_{m=1}^{P} \eta_m = 1 \\ & \eta_m \geq 0 \quad \forall m, \end{aligned} \tag{2.9}$$

where $\boldsymbol{\eta}$ is the vector of kernel weights, P is the number of kernel matrices, and $J(\boldsymbol{\eta})$ is the optimization problem in Equation 2.4 with a modification in its objective function, introducing $k_\eta(\boldsymbol{x}_i, \boldsymbol{x}_j)$ instead of $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ term.

This optimization problem does not exhibit convexity with respect to both $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$, although the outer minimization problem exhibits convexity with respect to $\boldsymbol{\eta}$ and the inner minimization problem exhibits convexity with respect to $\boldsymbol{\alpha}$. Therefore, it cannot be solved with a global approach with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\eta}$.

They solve this problem with an efficient iterative approach (Xu et al., 2010). The approach starts with initializing kernel weights to uniform values (i.e., $\boldsymbol{\eta}^{(1)} = 1/P$). Then, it follows a two-step method until converge. In the first step of an iteration t, the support vector coefficients, which are $\boldsymbol{\alpha}^{(t)}$, are found by solving the inner optimization problem with the latest kernel weights, which are $\boldsymbol{\eta}^{(t)}$. This corresponds to solving a standard SVM model. In the second step of an iteration t, the kernel weights of the next iteration, which are $\boldsymbol{\eta}^{(t+1)}$ are found using $\boldsymbol{\alpha}^{(t)}$ and the latest kernel weights, which are $\boldsymbol{\eta}^{(t)}$ with the following update equation:

$$\eta_m^{(t+1)} = \frac{\eta_m^{(t)} \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^{(t)} \alpha_j^{(t)} y_i y_j k_m(x_i, x_j)}}{\sum_{o=1}^{P} \eta_o^{(t)} \sqrt{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i^{(t)} \alpha_j^{(t)} y_i y_j k_o(x_i, x_j)}} \qquad \forall m. \tag{2.10}$$

In this approach, ℓ_1-norm is enforced on the kernel weights. This encourages kernel weights to be mostly zero, in other words, sparse. The gene sets with kernel matrices containing non-zero weights could potentially hold relevance in distinguishing between early- and late-stage cancers.

2.3 Artificial Neural Networks

Artificial neural networks ANNs are computational models that are inspired by the human brain's neural structure and aim to solve complex tasks in machine learning. ANNs consist of interconnected nodes (neurons) organized into layers. They include varied types: MLPs for general tasks, convolutional neural networks for images, recurrent neural networks (RNNs) for sequences, long short-term memory networks

and gated recurrent units for improved RNN training, generative adversarial networks for data generation, and transformers for sequence processing. In this section, we will specifically focus on the MLP.

2.3.1 Single-Layer Perceptrons

Single-layer perceptrons represent the simplest form of artificial neural networks. A single-layer perceptron is a type of feedforward artificial neural network consisting of a single layer of nodes, each with its own set of weights and a bias term. The perceptron takes an input vector $\boldsymbol{x}$ with D dimensions and computes the output of each node using the equation:

$$\hat{y} = \sum_{d=1}^{D} w_d x_d + w_0 \tag{2.11}$$

where w_d is a weight value in the weight vector $\boldsymbol{w}$ and w_0 is the bias term. The equation represents a weighted sum of the input features along with the bias term added to it. Essentially, it calculates a linear combination of the inputs and bias to produce the output value $\hat{y}$.

Single-layer perceptrons can be extended to have multiple nodes in the output layer, each producing a separate output value. This modification can be described by the equation:

$$\hat{y}_j = \sum_{d=1}^{D} w_{dj} x_d + w_{j0} \tag{2.12}$$

where $\hat{y}_j$ is the value of the output in node j, w_{dj} correspond to the connection weight from input x_d to output $\hat{y}_j$ in the weight matrix $\mathbf{W}$ and w_{j0} is the bias term.

Figure 2.2 displays a network graph for a single-layer perceptron with multiple output nodes.

2.3.2 Multilayer Perceptrons

The MLP represents another type of feedforward artificial neural network. MLP finds extensive application in diverse tasks, including classification and regression.

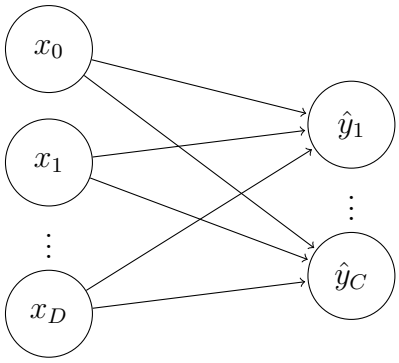

Figure 2.2: The network graph of a single-layer perceptron with D input units and C output units.

It consists of multiple layers of linked neurons, which are often referred to as nodes or units.

An MLP architecture contains input, output, and hidden layers. The information flows in a one-way direction, moving from the input layer to the output layer through the network.

The advantage of an MLP compared to a single-layer perceptron is that the MLP can approximate both linear and nonlinear functions, whereas a single-layer perceptron can only approximate linear functions.

When passing the input vector $\boldsymbol{x}$ with dimension D through an MLP that includes a hidden layer with multiple nodes, the output of the hidden layer can be described as follows:

$$g_h = \sum_{d=1}^{D} v_{dh} x_d + v_{h0} \tag{2.13}$$

where g_h is the value of the hidden layer in node h, and v_{dh} is the connection weight from input x_d to g_h in the hidden layer. Each node in the hidden layer, denoted by g_h, computes a weighted sum of the input features $\boldsymbol{x}$ using connection weights v_{dh}. The bias term v_{h0} is also added to this weighted sum. If the MLP has only one hidden layer, then we can obtain the value in the output layer as follows:

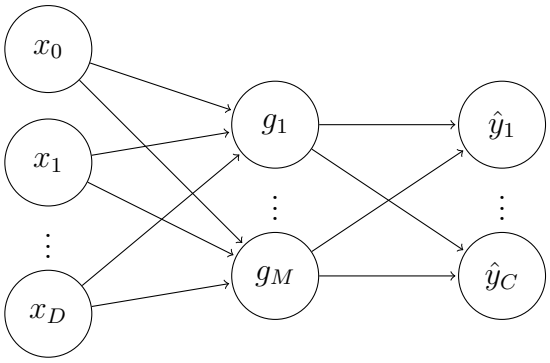

Figure 2.3: The network graph of an MLP with D input units, M hidden units, and C output units.

$$\hat{y}_j = \sum_{h=1}^{H} z_{hj} g_h + z_{j0} \tag{2.14}$$

where y_j represent the value of the node j in the output layer, z_{hj}, is the connection weight from input g_h to $\hat{y}_j$, z_{j0} is the bias, and H is the number of nodes in the hidden layer. Figure 2.3 displays a network graph for an MLP with multiple output nodes.

To leverage the capability of an MLP to approximate non-linear functions, we can introduce a non-linear activation function, such as the sigmoid function. When incorporating the sigmoid activation function into the hidden layer, Equation 2.13 is modified as follows:

$$g_h = \text{sigmoid}\left(\sum_{d=1}^{D} v_{dh} x_d + v_{h0}\right). \tag{2.15}$$

The sigmoid function introduces non-linearity to the hidden layer, allowing the MLP to model and capture complex patterns in the data. The output layer can also have an activation function, depending on the type of task the network is designed for.

2.3.3 Loss Functions

Loss functions, also known as cost functions or objective functions, are an essential component of neural networks. They quantify the discrepancy between the predicted output of a model and the actual ground truth (target) values. The goal of training a neural network model is to minimize this discrepancy, and it is achieved by adjusting the parameters of the network during the learning process.

In MLPs designed for classification tasks, loss functions are used to measure the difference between the predicted output of the network and the actual target labels. The choice of an appropriate loss function is crucial as it directly influences the learning process and the performance of the model.

We will give a brief overview of commonly used loss functions:

Cross-Entropy Loss

Cross-entropy loss, also known as log loss, is widely used in multi-class classification problems. It measures the dissimilarity between the predicted probability distribution and the true probability distribution (one-hot encoded target labels). It is given by the following equation:

$$f(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{ic}\log(\hat{y}_{ic}) \tag{2.16}$$

where N is the number of samples, C is the number of classes, y_{ic} is the one-hot encoded target label for sample i and class c, and $\hat{y}_{ic}$ is the predicted probability of the sample belonging to class c.

Categorical Hinge Loss

Categorical hinge loss, also known as the multi-class SVM loss, is inspired by SVMs and is commonly used in multi-class classification. It encourages correct class scores to be higher than incorrect class scores by a margin. The loss for a single sample is given by:

$$f(s_{\text{correct}}, s_{\text{incorrect}}) = \max(0, 1 - s_{\text{correct}} + s_{\text{incorrect}}) \tag{2.17}$$

where s_{correct} is the score of the correct class and $s_{\text{incorrect}}$ is the score of the highest-scoring incorrect class.

Binary Cross-Entropy Loss

Binary cross-entropy loss, also known as logistic loss, is used in binary classification problems where there are only two classes. It measures the difference between the predicted probability of the positive class and the true target label (0 or 1). The loss is given by:

$$f(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{2.18}$$

where N is the number of samples, y_i is the true target label for sample i (0 or 1), and $\hat{y}_i$ is the predicted probability of the sample belonging to the positive class.

Sparse Categorical Cross-Entropy Loss

Sparse categorical cross-entropy loss is similar to cross-entropy loss but is used when the target labels are provided as integers (class indices) instead of one-hot encoded vectors. It is more memory-efficient when dealing with a large number of classes. The formula is the same as Equation 2.16, but the target labels y_{ic} are integers representing the class indices instead of one-hot encoded vectors.

2.3.4 Activation Functions

Activation functions are fundamental elements within artificial neural networks that determine whether individual neurons should be activated based on the calculated weighted sum of inputs and biases. The majority of activation functions introduce non-linearities, which help to capture complex relationships. Besides, most of them are differentiable operators. This differentiability property further enables the application of gradient-based optimization techniques for efficient parameter updates during the training process of a neural network.

Different activation functions have their strengths and weaknesses, making them more suitable for specific scenarios in neural network architectures. The neural

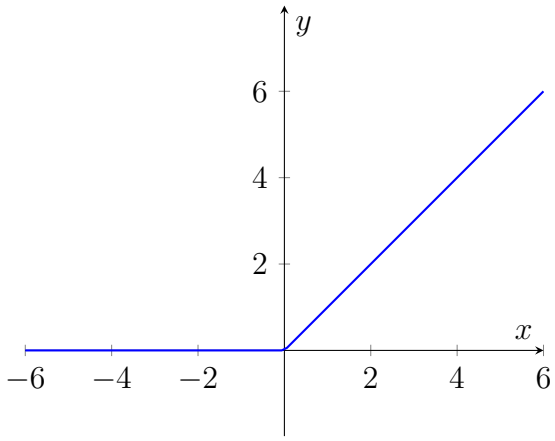

Figure 2.4: ReLU function.

networks commonly employ activation functions like the rectified linear unit (ReLU), tanh, sigmoid, and softmax, which will be subject to further discussion. In addition to these well-known functions, the literature presents numerous proposed activation functions, including Realky ReLU, PRELU, ELU, and Swish.

ReLU Function

ReLU offers a straightforward non-linear transformation by taking an input x and defining the function as the maximum between the input x and 0 (Nair and Hinton, 2010). This behavior is visually represented in Figure 2.4. Mathematically, ReLU can be expressed as:

$$\text{ReLU}(x) = \max(x, 0). \tag{2.19}$$

It provides a sparse activation by effectively setting the negative values to 0 and leaving positive values unchanged. One of the advantages of ReLU is its simplicity and low computational complexity. This simplicity is especially beneficial in large neural networks, where efficiency is important. Another advantage of this function is its ability to introduce non-linearity to the network, which can enable the neural network to learn complex patterns.

Furthermore, ReLU helps mitigate a problem that is suffered while training deep neural networks, which is the vanishing gradient problem. It is a challenge in deep neural networks where the gradients of the loss function with respect to the parameters of early layers become too small during backpropagation, which leads to slow learning and difficulty in training deep neural networks effectively. ReLU does not saturate for large positive values of x, and this prevents the gradients from becoming too small during backpropagation.

However, ReLU does have its share of disadvantages, such as having the "dying ReLU" problem. This problem happens when neurons become inactive and always output zero. As a result, these neurons stop learning since they do not update their weights during training. Although ReLU mitigates the vanishing gradient problem, it might be caught by another problem faced while training deep neural networks, which is the exploding gradient problem. It is a problem that occurs when the gradients of the loss function with respect to the parameters of the model become extremely large during training. This leads to excessively large weight updates and causes the optimization process to become unstable. Another potential issue with ReLU is its unlimited output range. Since there is no upper bound, the output of a neuron can become extremely large for positive inputs. Finally, it is important to note that the ReLU function is non-differentiable at the point where its input is exactly 0.

sigmoid Function

The sigmoid function, illustrated in Figure 2.5, is a transformation applied to inputs in the domain of real numbers $\mathbb{R}$. It transforms these inputs into outputs that are confined to the interval between 0 and 1:

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}. \tag{2.20}$$

A benefit of the sigmoid function is that its non-linear nature allows neural networks to learn complex relationships between inputs and outputs. Furthermore, it is a smooth and continuous function, which means its derivative is well-defined at any

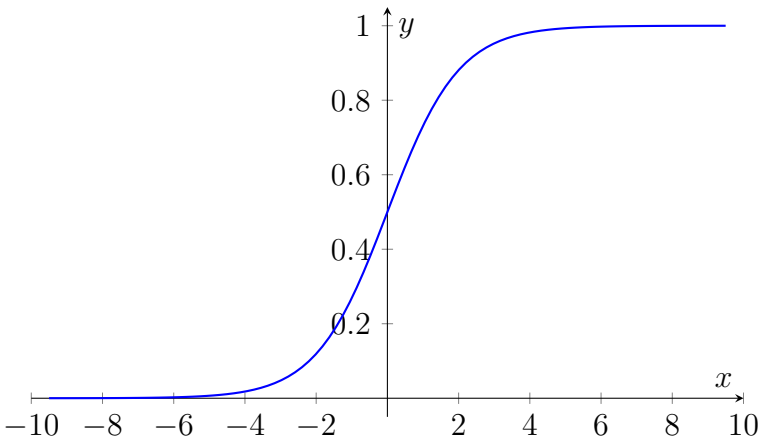

Figure 2.5: sigmoid function.

point. This smoothness makes it easier to perform optimization and backpropagation during training. It is usually utilized for the output units in binary classification tasks since they allow us to interpret the outputs as probabilities. A disadvantage of the sigmoid function is that it might suffer from the vanishing gradient problem.

tanh Function

The tanh function resembles the sigmoid activation function as it provides outputs confined in an interval. It transforms the inputs in the domain of real numbers, $\mathbb{R}$, to the interval between -1 and 1:

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}. \tag{2.21}$$

Figure 2.6 displays the visual representation of the tanh function. The tanh function can be viewed as a shifted and scaled version of the sigmoid function. It includes all the benefits of the sigmoid function. Different from the sigmoid, it is zero-centered and has a larger output range. It might suffer from the vanishing gradient problem as the sigmoid.

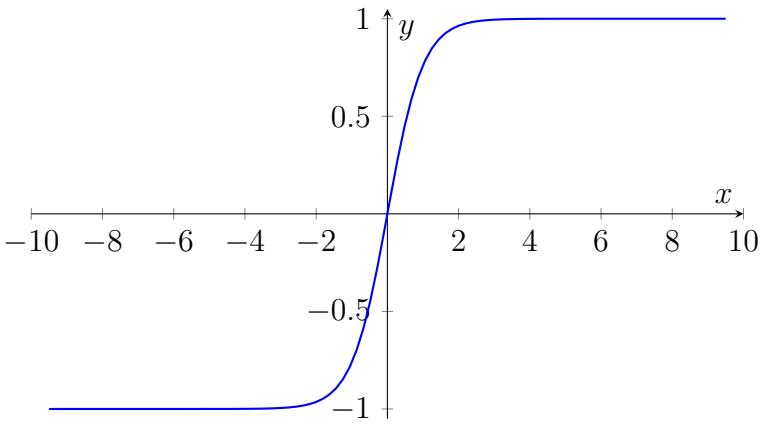

Figure 2.6: tanh function.

softmax Function

The softmax function takes an input vector of real numbers, $\mathbb{R}$, and transforms it into a probability distribution. Given an input vector $\boldsymbol{x}$ of length C, the softmax function computes the probabilities for each element x_i as:

$$\text{softmax}(x_c) = \frac{\exp(x_c)}{\sum_{d=1}^{C} \exp(x_d)} \qquad c = 1, ..., C. \tag{2.22}$$

While the softmax function is commonly used as an activation function on the output layer of a neural network for multi-class classification, it is also possible to use it as an activation function for hidden layers. Using the softmax function in a hidden layer allows the network to address situations where it must make decisions among multiple options or assign weights to various inputs (Goodfellow et al., 2016).

Applying the softmax activation function in hidden layers has its own set of benefits and drawbacks. One of the benefits of applying softmax in hidden layers is the probability interpretation it provides for the activations. The normalized outputs can be seen as probabilities, which indicate the likelihood of the neural network focusing on different features or patterns in the data at the hidden layer level. This allows us to understand the relative importance of different features or patterns that the hidden layer is capturing. This can be particularly useful for tasks

where the hidden layer activations have some semantic meaning.

Another benefit of using softmax in hidden layers is that it can lead to more stable and meaningful representations. The probability distribution generated by softmax ensures that the activations are well-scaled and non-negative, which can help mitigate problems like dead neurons. Furthermore, the normalization property also makes the hidden layer more robust to changes in the scale of input data, which contributes to better generalization. Finally, having a softmax activation in a hidden layer will make the neurons linearly dependent on each other, which might be useful in scenarios where you want to encourage cooperation and specialization among neurons in the hidden layer.

Besides these advantages, using softmax in hidden layers has its drawbacks. One drawback is related to the "smoothing" effect of the function. Since it converts the activations into probabilities, some of the detailed information present in the raw neuron activations might be lost. Additionally, the softmax function is computationally more expensive than other activation functions like ReLU, as it involves exponentiation and summation across all neurons in the layer. Finally, it might suffer from the vanishing gradient problem in deep networks.

2.3.5 Backpropagation Algorithm

The neural networks are mostly trained using iterative algorithms. The main reason for this lies in the nature of the loss functions associated with neural networks, as the nonlinearity of the networks makes them nonconvex. Backpropagation is a widely employed algorithm for training artificial neural networks using gradient descent (Rumelhart et al., 1986). Its primary function is to determine the gradient of the loss function with respect to the parameters of the network. This gradient information is then utilized to adjust the weights of the network in the pursuit of minimizing the training error effectively.

The backpropagation algorithm starts with the forward phase. During this phase, the input data is sequentially processed through the layers of the neural network. Each layer calculates a weighted sum of its inputs, followed by applying an acti-

vation function if it exists. This phase ends when the output layer generates an output based on the given input. The backward phase follows the forward phase subsequently. The backward phase starts from the output layer and proceeds backward through the network. During the backward phase, the gradient of the error function with respect to each weight in the network is computed layer by layer using the chain rule. Finally, the weights of the network are updated using the computed gradients in the direction that minimizes the error function.

While performing the update, there are various approaches employed in the literature. Some of these methods are Stochastic Gradient Descent (SGD), Adagrad (Duchi et al., 2011), RMSprop (Tieleman and Hinton, 2012), and Adam (Kingma and Ba, 2014).

2.3.6 Regulatization Functions

Regularization functions are essential components of neural networks as they help to improve the generalization performance of the networks. They introduce additional constraints or penalties to the learning process, which influence how the network learns and generalizes. Some regularization methods have additional benefits. As an example, Lasso regularization can introduce sparsity in the weights of the network.

Most of the regularization approaches constrain the capacity of the network by including an additional term in the error function. This term serves as a penalty that discourages the network from becoming overly complex or overfitting to the training data and is usually controlled by a scalar hyperparameter that determines the relative contribution of the parameter norm penalty term to the overall error function.

Lasso Regularization

Lasso regularization, also known as ℓ_1 regularization, is a powerful tool employed in neural networks to enhance their generalization performance. Lasso regularization incorporates ℓ_1 penalty to the error function. ℓ_1 penalty on a model parameter $\boldsymbol{w}$

is formulated as the sum of the absolute values of the weights:

$$\|\boldsymbol{w}\|_1 = \sum_d |w_d|. \tag{2.23}$$

Lasso regularization can effectively drive some of the weights to precisely zero and induce sparsity in the weights of the network. The reason behind this sparsity-inducing behavior is related to the nature of the absolute value function. Lasso promoting sparsity in neural networks offers a wide range of advantages. By making some of the weights zero, it helps to have a simpler model architecture with fewer active connections between neurons, reducing complexity. Furthermore, the sparsity-induced neural network becomes more interpretable. This improved interpretability is vital for building trust and gaining insights into the decision-making process of the network.

Ridge Regularization

Ridge regularization, also known as ℓ_2 regularization, is another fundamental approach utilized in neural networks to combat overfitting and improve generalization. Similar to the Lasso regularization, ridge regularization adds a penalty term to the error function. On a model parameter, $\boldsymbol{w}$, ℓ_2 penalty is formulated as the sum of the squared values of the weights:

$$\|\boldsymbol{w}\|_2^2 = \sum_d w_d^2. \tag{2.24}$$

Unlike Lasso regularization, ridge regularization does not promote sparsity in neural networks. Instead, it prioritizes a more uniform distribution of weight values across the network. The penalty term in ridge regularization discourages overly large individual weight magnitudes and promotes a more stable and well-behaved model. This behavior offers an advantage in situations where highly correlated features coexist in the data. By tempering the influence of correlated inputs, it enhances the robustness of the network. In addition, this behavior allows the network to be less sensitive to minor changes in the input data and helps to generalize better to unseen data.

Elastic Net Regularization

Elastic Net regularization is a combination of ℓ_1 and ℓ_2 regularization techniques. It seeks to utilize the advantages of both approaches while minimizing their respective drawbacks. Similar to Lasso and ridge regularization, Elastic Net also adds an extra penalty term to the error function. The Elastic Net penalty is a linear combination of the ℓ_1 and ℓ_2 penalties and is controlled by two hyperparameters: α and λ. The overall regularization term can be written as follows:

$$\alpha \left(\lambda \|\boldsymbol{w}\|_1 + (1-\lambda)\|\boldsymbol{w}\|_2^2\right) \tag{2.25}$$

where α controls the overall strength of the regularization and λ controls the relative contribution of the ℓ_1 and ℓ_2 penalties. When $\lambda = 1$, Elastic Net becomes equivalent to Lasso regularization, and when $\lambda = 0$, it reduces to ridge regularization.

Elastic Net offers the benefit of encouraging sparsity, similar to Lasso, and also promoting a more balanced weight distribution, like ridge. This makes it suitable for scenarios with many correlated features in the data. The ℓ_1 component drives some weights to become precisely zero, which helps in feature selection and creating simpler model structures. Meanwhile, the ℓ_2 component contributes to model stability and reduces sensitivity to minor input data changes.

By combining ℓ_1 and ℓ_2 penalties, Elastic Net addresses some of the limitations of individual regularization techniques. For instance, Lasso may select only one variable among a group of highly correlated variables, whereas Elastic Net tends to select groups of correlated variables together, providing a more balanced and interpretable model.

2.3.7 Weight Initialization

Weight initialization plays a pivotal role in training neural networks. Effectively initializing the weights of a neural network can have a considerable impact on the model's convergence speed and overall performance. Poor initialization can lead to slow convergence, vanishing gradients, or convergence to poor local minima. To

tackle these issues, various strategies for weight initialization have been developed, some of which are Xavier initialization, He initialization, LeCun initialization, and orthogonal initialization. We will briefly present the most commonly used weight initialization strategies, which are random initialization, Xavier initialization, and He initialization.

Zero Initialization

The zero initialization method involves setting all the initial weights to zero. When all weights start as zero, every neuron in a particular layer computes identical output values. This leads to the situation that each neuron learns identical features from input data. Consequently, the network would fail to capture complex patterns and relationships in the data, since it is unable to differentiate between different features. This limitation restricts the capacity of the network to learn and generalize from the training data. Furthermore, during the backward pass of the training process, neurons with the same weights will obtain the same gradients, and all neurons will receive identical updates during weight adjustments. As a result, this leads to an absence of diversity in their learning, which will slow down convergence.

Random Initialization

In random initialization, the weights are assigned random values drawn from a specified distribution. The idea behind this randomness is to break the symmetry that can occur when all weights have the same value. However, the key challenge with random initialization is related to choosing the appropriate distribution and scale for the random values. If the random values are too large, they can lead to activation saturation and vanishing gradients, which will make it difficult for the network to learn effectively. Conversely, if the values are too small, the convergence might be slow, and the network could need more iterations to learn meaningful representations.

Using a Gaussian distribution with a mean of 0 and a relatively small variance is a common practice employed. This distribution ensures that the initial weights are

centered around 0, which is useful for activation functions that are symmetric around 0, like the tanh function. Other distributions, such as the uniform distribution, can also be used based on the specific characteristics of the network and activation functions.

Xavier Initialization

Xavier initialization, also known as Glorot initialization, is a widely used method for initializing the weights of neural networks. It was introduced to tackle the challenges associated with the vanishing and exploding gradient problems during training (Glorot and Bengio, 2010).

Xavier initialization aims to set the initial weights in such a way that the variance of the outputs of each layer remains relatively consistent across layers. This objective is crucial in preventing the gradients from either vanishing or exploding during their backward propagation through the network.

For a fully connected layer with n_{in} inputs and n_{out} outputs, Xavier initialization suggests initializing its weights w_{dj} by sampling from:

$$w_{dj} \sim \text{Uniform}\left(-\sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}} + n_{\text{out}}}}\right). \tag{2.26}$$

Some common activation functions that work well with Xavier initialization are tanh, sigmoid, and softmax activation functions (Glorot and Bengio, 2010).

He Initialization

He initialization is another technique for initializing the weights of neural networks. It was specifically designed when using ReLU activation functions (He et al., 2015). The aim of this initialization strategy is to handle the challenges related to the vanishing and exploding gradient problems, like the other initialization strategies introduced.

Similar to Xavier initialization, the main idea behind He initialization is to set the initial weights in such a way that the variance of the outputs from each neuron remains approximately the same across layers. The difference compared to the

Xavier initialization is the scaling factor which is used to determine the variance of the initial weights. For a fully connected layer with n_{in} inputs, He initialization samples its initial weights w_{dj} from:

$$w_{dj} \sim \mathcal{N}\left(0, \frac{2}{n_{in}}\right) \tag{2.27}$$

where $\mathcal{N}(\cdot, \cdot)$ is the Gaussian distribution.

Chapter 3

MULTILAYER PERCEPTRON WITH MULTIPLE KERNELS

In this chapter, we delve into the topic of the MLP with multiple kernels. We begin by framing the problem. Subsequently, we provide the concept of gene set-based kernel functions. We then explore the core of our approach, detailing the architecture and methodology of the MLP with multiple kernels. Finally, we engage in a discussion that reflects upon the characteristics of our approach.

3.1 Problem Formulation

We are interested in finding out what discriminates early-stage cancers from late-stage cancers. For this aim, we focused on the task of predicting the pathological stage of cancer given the gene expression data. We formulated this task as a binary classification problem.

We denoted $\mathcal{X}$ as the gene expression data and $\mathcal{Y}$ as the pathological stage of cancer, namely early-stage or late-stage. For each cohort, we show our training data set as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$ where N is the number of primary tumors, $\boldsymbol{x}_i$ is the gene expression profile of tumor i and $y_i \in \{0, 1\}$ is the class label of tumor i. $\boldsymbol{x}_i \in \mathbb{R}^D$ where D is the number of gene expression profiles. The early-stage tumors are represented as 0, and the late-stage tumors are represented as 1 in our formulation.

3.2 Gene Set-Based Kernel Functions

In our study, we construct different gene set-based kernel matrices for each cohort, making use of the well-defined Hallmark gene sets (Liberzon et al., 2015). These matrices capture similarity patterns between gene expression profiles for each cohort. To build these matrices, we select specific subsets from our data set based on the

genes within the pathways. We denote the sub-vector of the gene expression profile of tumor i, containing genes from a given gene set $\mathcal{G}$, as $\boldsymbol{x}_i[\mathcal{G}]$. Using the Gaussian kernel function, we then measure the similarity between the gene expression profiles of tumors based on gene set $\mathcal{G}$ using as

$$k_{\mathcal{G}}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \exp\left(-\frac{\left(\boldsymbol{x}_i[\mathcal{G}] - \boldsymbol{x}_j[\mathcal{G}]\right)^{\top}\left(\boldsymbol{x}_i[\mathcal{G}] - \boldsymbol{x}_j[\mathcal{G}]\right)}{2\sigma_{\mathcal{G}}^2}\right), \tag{3.1}$$

where we choose the kernel width parameter $\sigma_{\mathcal{G}}$ as the mean of pairwise Euclidean distances between training instances.

In our data set, the number of genes that represent a tumor sample, D, is approximately 20000, while the genes in a Hallmark pathway range from 32 to 200. As a result, the dimensionality of $\boldsymbol{x}_i[\mathcal{G}]$ is significantly reduced compared to $\boldsymbol{x}_i$. Consequently, computing a kernel matrix based on a specific gene set is much more computationally efficient than calculating a kernel matrix that considers the entire set of genes.

3.3 Multilayer Perceptron with Multiple Kernels

To address the binary classification problem of cancer stage prediction, we propose an MLP model that operates on multiple gene set-based kernel matrices. The advantage of this model lies not only in its predictive accuracy but also in its ability to offer insights into the biological processes underlying the distinction between early- and late-stage cancers.

The MLP receives a set of gene set-based kernel matrices, $\{(\mathbf{K}_m)\}_{m=1}^{P}$, where P is the number of kernels. A kernel matrix, $\mathbf{K}_m$, captures the similarity patterns between the genes in the pathway m and has a dimension of $N \times N$. The output of our model is a binary vector with a dimension of $N \times 1$, which contains the predicted pathological state (early-stage or late-stage) of given data points, namely, tumors.

In our MLP, we introduce a specialized layer called the "eta layer". It plays a pivotal role in enhancing both the interpretability and predictive power of the network. This unique layer introduces a novel attention-like mechanism that assigns

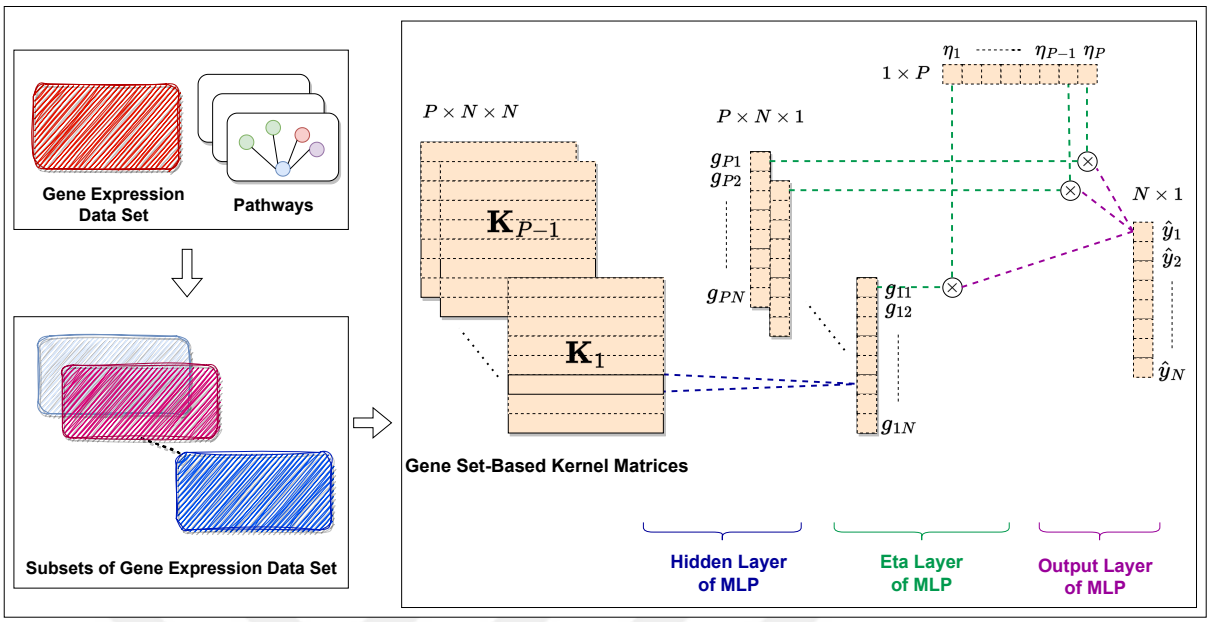

Figure 3.1: The overview of our MLP framework based on multiple kernels.

importance scores to different gene set-based kernel matrices. By doing so, it enables the model to focus on biologically relevant pathways that contribute significantly to the distinction between early-stage and late-stage cancers. Thus, the model aims to deliver accurate and interpretable predictions for pathological stage classification. Figure 3.1 displays an overview of our MLP framework based on multiple kernels.

3.3.1 The Structure of the Proposed Multilayer Perceptron

Our developed neural network comprises P input layers, each dedicated to a specific gene set-based kernel matrix. Every input layer is equipped with N nodes that are connected to an intermediate layer. All input layers share the same set of N weights denoted as α_1, α_2, ..., α_N. These weights are shared for all input layers to reduce model complexity and enhance efficiency.

We denote an input entry for the input kernel m and the data point i as $\boldsymbol{k}_{m,i}$. It can be seen as a vector capturing the similarity measures between gene expression profiles of tumor i and all other tumors in the cohort associated with the gene set

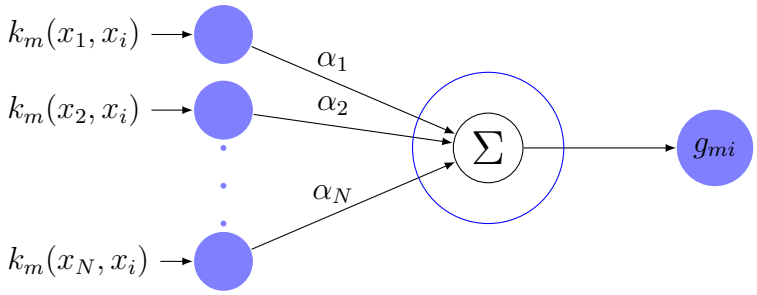

Figure 3.2: An intermediate node in the intermediate layer of the MLP, g_{mi}, receiving input from $\boldsymbol{k}_{m,i}$.

m and can be expressed as:

$$\boldsymbol{k}_{m,i} = \begin{bmatrix} k_m(\boldsymbol{x}_1, \boldsymbol{x}_i) & \dots & k_m(\boldsymbol{x}_N, \boldsymbol{x}_i) \end{bmatrix} \quad \forall (m, i) \tag{3.2}$$

where P is the number of input kernels, and $k_m(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a measure of similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ for the genes in pathway m computed by using the Gaussian kernel function.

The neural network has P nodes in its intermediate layer for each input kernel. Figure 3.2 shows one of the intermediate nodes, which is g_{mi}, in the intermediate layer. The value of an intermediate node, g_{mi}, is equal to the weighted summation of the input entries based on the weights of the input layer as the following:

$$g_{mi} = \sum_{j=1}^{N} \alpha_j k_m(\boldsymbol{x}_j, \boldsymbol{x}_i) \quad \forall m \tag{3.3}$$

In the network, there is another intermediate layer, which we call as the "eta layer", consisting of P nodes. This layer behaves like a constant function. Specifically, the output of this layer can be shown as:

$$f(x) = \begin{bmatrix} c_1 & c_2 & \dots & c_P \end{bmatrix} \tag{3.4}$$

where $f(\cdot)$ is a constant function, x is an arbitrary input, and c_1, c_2, ..., c_P are constants which are unchanged for all x. Furthermore, the constant variables, c_1, ..., c_m, are trainable, which enables the neural network to optimize and adjust these

values during the learning process. We introduce Lasso regularization to the constant values generated by this layer to encourage sparsity among the constant variables and promote the efficiency of the network. To enhance the interpretability of our MLP, we also introduced non-negativity constraints on the constant values.

Following this constant function, we apply a softmax activation function to this layer. The softmax activation transforms the values of the nodes in the eta layer into a probability distribution. Each node independently produces a probability value based on the constant value generated by the constant function, and the summation of these probabilities is equal to 1. We represent the output values in these nodes as η_1, ..., η_P, and call them "kernel weights". Therefore, the value of the kernel weight η_m in node m is equal to:

$$\eta_m = \frac{\exp(c_m)}{\sum_{d=1}^{P} \exp(c_d)} \quad \forall m, \tag{3.5}$$

where c_m is the constant value generated by the constant function in node m. As a result, the eta layer takes on the role of an essential element in probabilistic analysis, which contributes to the network's interpretability.

Once the outputs from the intermediate and eta layers are computed, they are passed through the output layer. This layer computes the dot product between the outputs of the intermediate layer and the outputs of the eta layer. Since the outputs of the eta layer, which are the kernel weights, are mostly sparse due to the ℓ_1-norm, they can be interpreted as the importance or contribution of the corresponding input kernel matrix to the overall prediction of the neural network.

Finally, the application of the sigmoid activation function to the output layer yields the neural network's ultimate output. This output is a probability score, representing the likelihood of each tumor being classified as early-stage or late-stage cancer. Therefore, the output of our neural network can be written as

$$\hat{y}_i = \text{sigmoid}\left(\sum_{m=1}^{P} \eta_m g_{mi}\right).$$

Substituting the values of the intermediate outputs, we get:

$$\hat{y}_i = \text{sigmoid}\left(\sum_{m=1}^{P}\sum_{j=1}^{N}\alpha_j\eta_m k_m(\boldsymbol{x}_j, \boldsymbol{x}_i)\right)$$

where $\hat{y}_i$ can be rewritten as:

$$\hat{y}_i = \frac{1}{1+\exp\left(-\sum_{m=1}^{P}\eta_m g_{mi}\right)} = \frac{1}{1+\exp\left(-\sum_{m=1}^{P}\sum_{j=1}^{N}\alpha_j\eta_m k_m(\boldsymbol{x}_j, \boldsymbol{x}_i)\right)}$$

3.3.2 The Loss Function

The objective of our neural network in the context of binary classification is to minimize the error, which measures the difference between the target output and the predicted output. We utilize the binary cross-entropy loss function to achieve this objective as the following:

$$f(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log\left(\hat{y}_i\right) + \left(1-y_i\right)\log\left(1-\hat{y}_i\right)\right],$$

where y_i corresponds to the target output and $\hat{y}_i$ corresponds to the predicted output.

3.3.3 The Similarity of the Kernel Weights with Attention Mechanisms

Our model introduces a novel attention-like mechanism embodied by the eta layer. This mechanism exhibits similarities with the widely used scaled dot-product attention mechanism in various neural network architectures, particularly in sequence-to-sequence tasks and natural language processing. This attention-like mechanism in the eta layer serves as a crucial element in our model. It assigns importance scores to different gene set-based kernel matrices and allows the model to focus on biologically relevant pathways for accurate cancer stage classification.

The attention mechanism's fundamental principle is to selectively attend to crucial information within an input sequence. It achieves this by computing attention scores that represent the relative importance of different elements in the sequence with respect to a given query vector (Vaswani et al., 2017).

Mathematically, in the context of the scaled dot-product attention, the query vector, denoted as $\boldsymbol{q}$, serves as the current context or state of the model, while the key-value pairs, denoted as $\mathbf{K}$ and $\mathbf{V}$, encode information from the input sequence. The attention scores, denoted as $\boldsymbol{a}$, are computed based on the similarity between the query vector and the corresponding keys using the dot product:

$$\boldsymbol{a} = \text{softmax}\left(\frac{\mathbf{K}\boldsymbol{q}}{\sqrt{d_k}}\right) \tag{3.6}$$

where d_k is the dimensionality of the query vector and is used for scaling the dot product.

After obtaining the attention scores, a weighted sum of the values is computed, where each value in $\mathbf{V}$ is multiplied by its corresponding attention score. This operation yields an attended representation of the input sequence and highlights the most relevant elements based on the query:

$$\sum_j \boldsymbol{a}_j \mathbf{V}_j. \tag{3.7}$$

In our model, the eta layer demonstrates conceptual similarity to the attention mechanisms by assigning importance scores to different gene set-based kernel matrices. Instead of using query-key-value triplets, the eta layer employs the kernel weights, $\boldsymbol{\eta}$, to represent the importance of each gene set-based kernel matrix. These kernel weights are learned during the model training and act as attention weights, analogous to the attention scores in dot-product attention.

Through this attention-like mechanism in the eta layer, our model gains the ability to identify and weigh the significance of specific biological pathways associated with the distinction between early-stage and late-stage cancers. As a result, the eta layer enhances both the interpretability and predictive accuracy of our neural network in cancer stage classification tasks.

3.4 Discussion

In this chapter, we presented a novel MLP model with multiple gene set-based kernel matrices for the binary classification of cancer stages. We focused on distinguishing early-stage cancers from late-stage cancers using gene expression data and formulated the problem as a binary classification task. Our proposed model aims to provide accurate predictions while also offering insights into the biological processes that differentiate between the two cancer stages.

The key characteristics of our approach can be summarized as follows:

- Gene Set-Based Kernel Functions: We constructed multiple gene set-based kernel matrices, capturing similarity patterns between gene expression profiles for each cohort. These matrices were built using well-defined Hallmark gene sets, which enable us to integrate previous knowledge about pathways/gene sets. The gene set-based kernel matrices also effectively reduced the dimensionality of the data and improved computational efficiency.

- MLP Architecture: Our MLP model utilizes multiple input layers, each corresponding to a specific gene set-based kernel matrix. These input layers are connected to an intermediate layer and an "eta layer" that behaves like a constant function. The eta layer introduces sparsity and interpretability to the outputs of the network.

- Interpretable Outputs: The eta layer, along with Lasso regularization, contributes to the interpretability of the neural network. The output values produced by the eta layer, which are the kernel weights, can be seen as the importance or contribution of each gene set-based kernel matrix to the overall prediction, allowing us to understand the underlying biological processes.

Chapter 4

COMPUTATIONAL EXPERIMENTS

In this study, we conducted a series of computational experiments to evaluate and analyze the predictive capabilities of our model across 15 different cancer cohorts. These cohorts encompass a diverse range of cancers, each presenting unique challenges in terms of early-stage and late-stage classification. To ensure comprehensive evaluation, we compared our model's performance with three well-established baseline algorithms: RFs, SVMs, and MKL.

4.1 Preprocessing

Before proceeding with the model training, we performed the necessary data preprocessing steps. Given that the gene expression data only had non-negative values, we applied a $\log_2$-transformation on the gene expression values. Additionally, to mitigate variations in scale, we standardized each feature in the training data set to have a mean of zero and a standard deviation of one. Subsequently, we normalized each feature in the test data set based on the mean and standard deviation computed from the training set.

4.2 Model Training

We employed a four-fold inner cross-validation approach to fine-tune the hyperparameters of four distinct models: RF, SVM, MKL, and MLP. To ensure unbiased evaluations, we initially divided the data set, comprising gene expression profiles of primary tumors, into two sets. The training data set contained 80% of the data, while the test data set held the remaining 20%. To maintain a balanced representation of early-stage and late-stage classes, we adopted the stratified cross-validation technique.

To ensure robustness and reliability, we repeated the entire cross-validation process, including stratification and normalization, a total of 100 times. This extensive repetition allowed us to accurately assess and measure the performance of the different models utilized in our study.

We used the Adam Optimizer as the main optimization method, which is known for its efficient and adaptive learning rate. We chose the exponential decay rate for the first moment estimates and the second moment estimates as 0.9 and 0.999, respectively, which are the default choices. The small constant for numerical stability, epsilon, is chosen as 10^{-7}, which is again the default choice. We performed cross-validation for the learning rate of the Adam optimizer from the set of $\{0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05\}$ and also for the ℓ_1 regularization factor of the eta layer from the set of $\{0.00001, 0.0001\}$. We also employed an exponential decay learning rate scheduler to enhance the training process of the MLP model. This technique gradually reduces the learning rate during training to facilitate convergence and achieve better performance.

For all of our experiments, we maintain a fixed training duration of 100 epochs, except for the cases where the training process terminates early due to meeting the early stopping criteria. This fixed number of epochs allows us to establish a comparable threshold across all experiments and also accommodates computational constraints.

Early stopping is implemented during training to prevent overfitting. It involves observing the validation loss throughout the training process and verifying if it decreases by at least 10^{-5} at the end of each epoch. If there is no improvement in the validation loss for 25 consecutive epochs, we stop the training process, and the model's weights are rolled back to the epoch where the validation loss was at its minimum. This strategy helps ensure the model generalizes well and does not memorize the training data excessively.

For weight initialization in our MLP network, including the eta layer, we used the Xavier uniform initializer. This initialization technique helps to prevent vanishing or exploding gradients during training and improves the model's convergence.

As the data size varies across different cohorts, we adopted a flexible approach

to batch sizing during training. Specifically, we determined the batch size for each cohort as one-eighth of its respective data size. This adaptive batch sizing strategy allowed the model to efficiently process data sets of varying sizes.

4.3 Experimental Settings

The RF algorithm is implemented using version 2.5.1 of an R package named `randomForestSRC` (Ishwaran and Kogalur, 2017). The hyperparameter corresponding to the number of decision trees to grow was chosen from the set $\{500, 1000, ..., 2500\}$ using a four-fold inner cross-validation approach.

For SVM and MKL algorithms, we used the implementations introduced by Rahimi and Gönen (2018) in R. Their implementations rely on version 8.1.0.34 of an R package called `MOSEK` while solving the optimization problems (ApS, 2019). They chose the Gaussian kernel function as a similarity measure among the gene expression profiles of tumors and selected the kernel width parameter as the mean of pairwise Euclidean distances between the training samples. Additionally, they chose the regularization parameter from the set $\{10^{-4}, 10^{-3}, \ldots, 10^{+5}\}$ using a four-fold inner cross-validation approach.

We developed our MLP model in Python by utilizing the TensorFlow framework (Abadi et al., 2015), an open-source software for deep learning. To facilitate the construction of complex networks like multi-input models and those with shared layers, we leveraged the Keras high-level library (Chollet et al., 2015), which is compatible with Tensorflow.

4.4 Performance Metric

In this study, we employed the widely recognized area under the receiver operating characteristic curve (AUROC) as the performance metric to evaluate the predictive capabilities of four distinct models. The receiver operating characteristic (ROC) curve is a visual representation illustrating the relationship between the sensitivity and the specificity for a binary classifier as the decision threshold varies. It enables the evaluation of the performance of the classifier across different classification sce-

Table 4.1: The experimental settings of our MLP.

Parameter	**Value**
Replications	100
Epochs	100
Optimizer	Adam
The exponential decay rate for the first moment estimates of the Adam optimizer	0.9
The exponential decay rate for the second moment estimates of the Adam optimizer	0.999
The constant for numerical stability (epsilon) of the Adam optimizer	10^{-7}
Learning rate	[0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05]
ℓ_1 regularization factor	[0.00001, 0.0001]
Monitored quantity for the early stopping	Validation loss
Early stopping patience (number of epochs)	25
The minimum change in the monitored quantity to qualify as an improvement for the early stopping	10^{-5}
Weight initialization	Xavier uniform

narios. The AUROC value ranges from 0 to 1, with a value of 0.5 indicating random guessing and a value of 1 indicating perfect classification.

The sensitivity and specificity, also known as the true positive rate and false positive rate, are computed as follows: The sensitivity is determined by dividing the number of correctly classified positive samples (TP) by the sum of TP and the number of samples incorrectly classified as negative (FN). On the other hand, specificity is calculated by dividing the number of correctly classified negative samples (TN) by the sum of TN and the number of samples incorrectly classified as positive (FP).

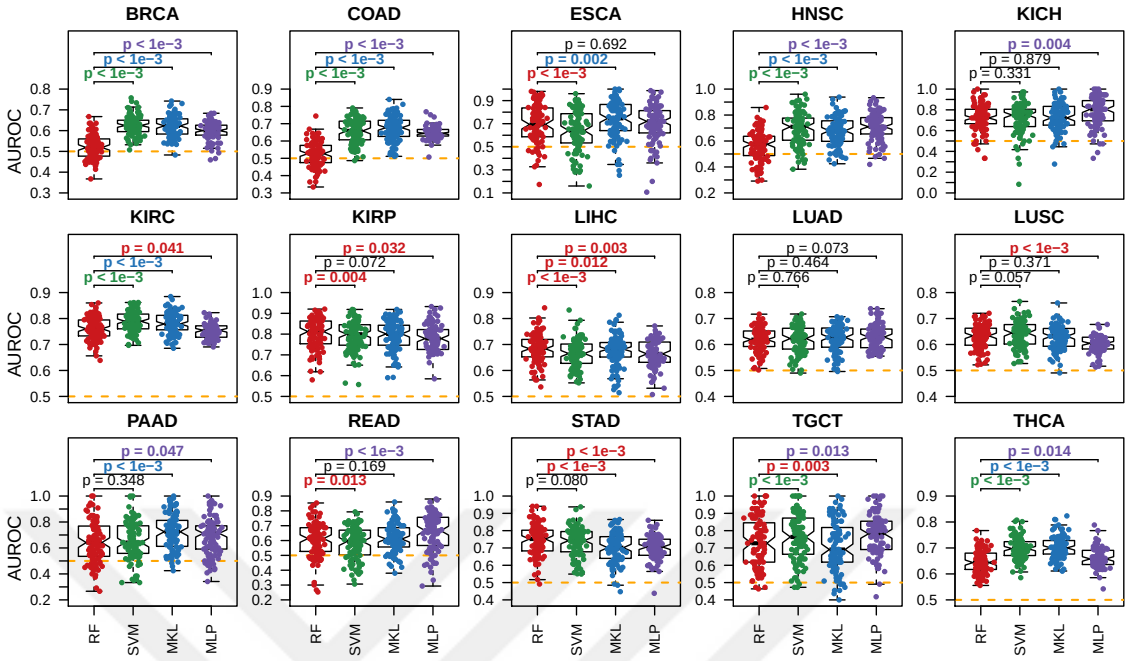

Figure 4.1: Predictive performances of RF, SVM, MKL, and MLP algorithms on 15 data sets constructed from the TCGA cohorts.

4.5 Predictive Performance Comparison of Models

In our computational experiments, we conducted a thorough comparative analysis of four distinct algorithms, namely RF, SVM, MKL, and our proposed MLP, to evaluate their predictive performance.

Figure 4.1 presents a detailed comparison of the predictive performance of four algorithms for each cohort individually. The performance of these methods was measured using AUROC, and we employed box-and-whisker plots to compare their AUROC values across 100 replications for each cohort.

To establish a reference point, we depicted the baseline performance as a dotted line at an AUROC value of 0.5. The results revealed that the median performance of all four classifiers surpassed the baseline, indicating the extraction of valuable and meaningful information from gene expression profiles of primary tumors. Consequently, this outcome leads to a significant conclusion that these gene expression profiles hold crucial insights, contributing to improved understanding and accurate predictions in the context of cancer research.

Moreover, we applied two-tailed paired t-tests to determine if there were statistically significant differences in the mean predictive performance of SVM, MKL, and MLP as compared to RF. The obtained p-values allowed us to measure the degree of differentiation between RF and each of the remaining three methods individually, enabling a comprehensive understanding of their respective strengths and weaknesses. These p-values are visually represented in Figure 4.1 using distinct colors. Specifically, the red, green, blue, and purple colors correspond to RF, SVM, MKL, and MLP, respectively, indicating that each of these algorithms exhibits significantly superior performance in comparison. Conversely, the black color signifies that there is no statistically significant difference observed between the two algorithms being compared.

Based on the findings depicted in Figure 4.1, we made noteworthy observations regarding the comparative performance of the MLP and RF methods. Out of the fifteen data sets analyzed, the MLP method exhibited significantly better results in eight cohorts, encompassing `BRCA`, `COAD`, `HNSC`, `KICH`, `PAAD`, `READ`, `TGCT` and `THCA`. Conversely, the RF method outperformed the MLP method in five cohorts, which are `KIRC`, `KIRP`, `LIHC`, `LUSC` and `STAD`. In two of the fifteen data sets, there was no significant difference found between the predictive performances of the MLP and RF methods.

Notably, the MLP method demonstrated remarkable improvements in certain cohorts, enhancing the predictive performance by 14.86% in `BRCA`, 23.97% in `COAD`, 1.33% in `ESCA`, 25.78% in `HNSC`, 7.48% in `KICH`, 1.78% in `LUAD`, 6.83% in `PAAD`, 9.00% in `READ`, 5.92% in `TGCT`, and 2.21% in `THCA`. However, in the `STAD` cohort, the MLP method experienced the most significant drop in performance, amounting to 5.74%.

Table 4.2 displays the average AUROC values obtained from 100 replications across all data sets. The results demonstrate that the MLP method achieved superior average AUROC values compared to the RF method in 10 out of the 15 cohorts, including `BRCA`, `COAD`, `ESCA`, `HNSC`, `KICH`, `LUAD`, `PAAD`, `READ`, `TGCT`, and `THCA`. Additionally, the data in Table 4.2 reveals that the MLP method outperformed the SVM method in terms of average AUROC values in 7 out of the 15 cohorts, specifically, `ESCA`, `HNSC`, `KICH`, `LUAD`, `PAAD`, `READ` AND `TGCT`. Moreover, in comparison with the

Table 4.2: Average AUROC of all algorithms for 100 replications on different cancer cohorts together with their rankings on each cohort in parenthesis.

Cohort	**RF**	**SVM**	**MKL**	**MLP**
BRCA	0.5216 (4)	**0.6261** (1)	0.6217 (2)	0.5991 (3)
COAD	0.5227 (4)	0.6582 (2)	**0.6716** (1)	0.6480 (3)
ESCA	0.6919 (3)	0.6368 (4)	**0.7349** (1)	0.7011 (2)
HNSC	0.5645 (4)	0.6981 (2)	0.6788 (3)	**0.7100** (1)
KICH	0.7260 (3)	0.7139 (4)	0.7281 (2)	**0.7803** (1)
KIRC	0.7611 (3)	**0.7894** (1)	0.7814 (2)	0.7516 (4)
KIRP	**0.8011** (1)	0.7911 (3)	0.7933 (2)	0.7799 (4)
LIHC	**0.6860** (1)	0.6653 (3)	0.6746 (2)	0.6635 (4)
LUAD	0.6188 (4)	0.6195 (3)	0.6212 (2)	**0.6298** (1)
LUSC	0.6305 (2)	**0.6385** (1)	0.6271 (3)	0.6013 (4)
PAAD	0.6476 (4)	0.6569 (3)	**0.7235** (1)	0.6918 (2)
READ	0.6032 (3)	0.5801 (4)	0.6200 (2)	**0.6575** (1)
STAD	**0.7456** (1)	0.7345 (2)	0.7022 (4)	0.7028 (3)
TGCT	0.7320 (3)	0.7615 (2)	0.7067 (4)	**0.7753** (1)
THCA	0.6472 (4)	0.6984 (2)	**0.7022** (1)	0.6615 (3)
Average	0.6600 (2.9)	0.68456 (2.5)	**0.6925** (2.1)	0.6902 (2.5)

MKL method, the MLP method exhibited higher average AUROC values in 6 out of the 15 cohorts, namely, HNSC, KICH, LUAD, READ, STAD, and TGCT. It is important to note that both the MLP and MKL methods achieved this predictive performance by using a smaller set of gene expression profiles when compared to the RF and SVM methods.

4.6 Biological Mechanisms Determined by Our Model

To demonstrate the biological significance of our algorithm, we performed an analysis to assess its capacity to identify meaningful gene sets using the kernel weights

obtained. Figure A.1 displays the convergence of the kernel weights throughout different epochs during a replication for the `READ` cohort. For each cancer data set and gene set pair, we recorded the number of replications in which the associated kernel weight was non-zero, indicating its significance.

Figure 4.2 presents the selection frequencies of 50 gene sets from the Hallmark collection across the 15 cancer data sets. To organize the data, we applied hierarchical clustering with Euclidean distance and complete linkage functions to group the rows and columns. We reported the column sums of selection frequencies to identify data sets that utilized a greater quantity of gene sets, on average, and the row sums to identify gene sets that were frequently selected across different data sets.

Upon examining Figure 4.2, we see that our model selects an average of 6.6 gene sets out of 50 as discriminative markers to effectively distinguish between early-stage and late-stage cancer across various cohorts. This highlights the ability of the model to capture essential genetic signatures that significantly contribute to cancer staging.

Furthermore, our model demonstrates a consistent trend of selecting a small number of gene sets in the majority of cohorts. Interestingly, in some of these cohorts, such as `HNSC`, `KICH`, `LUAD`, `READ`, our model outperforms RF, SVM, and MKL models in terms of their predictive performance. This indicates that despite using a smaller set of genes, our model can achieve superior accuracy in predicting cancer stages compared to these well-established classifiers.

However, it is important to note that the dynamics of cancer staging may vary across different cohorts, presenting distinct challenges in separating early-stage from late-stage cancer in some cases. For instance, in the `BRCA` and `LUSC` cohorts, our model tends to choose a larger number of gene sets on average, suggesting that the distinction between early- and late-stage cancers in these specific cohorts may be more complex, requiring a broader range of genomic characteristics for accurate classification.

When we examine the row sums of the selection frequencies, it becomes evident that distinct gene sets are selected across different cohorts, and only a select few gene sets exhibit a higher frequency of selection by multiple cohorts. For instance, `ANGIOGENESIS` and `SPERMATOGENESIS` gene sets are among the most frequently se-

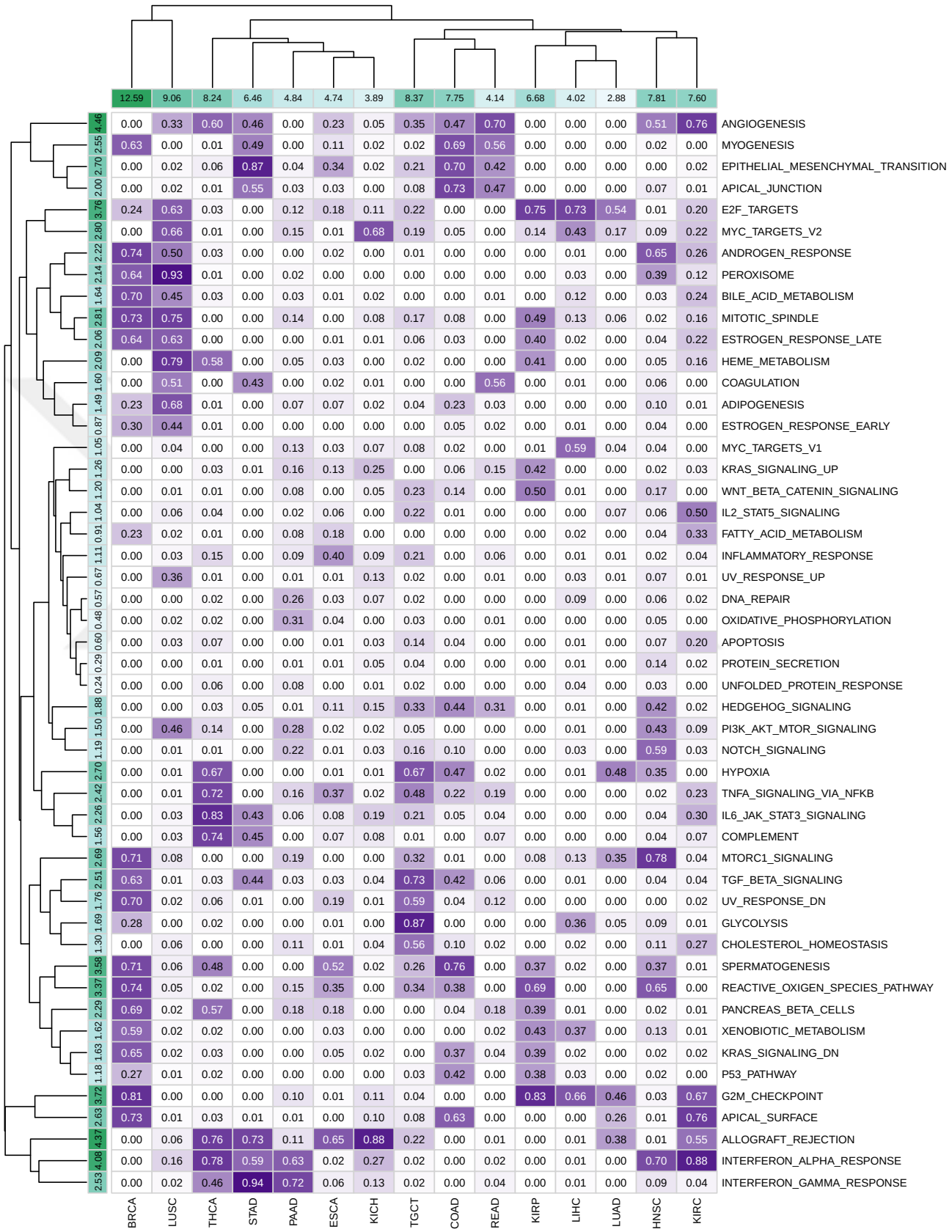

Figure 4.2: The selection frequencies of 50 gene sets in the Hallmark collection which were determined by MLP for 15 cancer cohorts.

lected gene sets. These gene sets have been previously associated with cancer formation in early stages (Bergers and Benjamin, 2003).

Similarly, all four metabolism-related gene sets, which are BILE_ACID_METABOLISM, HEME_METABOLISM, PANCREAS_BETA_CELLS, and XENOBIOTIC_METABOLISM, were frequently selected with frequencies above the average across all the cohorts (Chiang, 2013; Liberzon et al., 2015; Rashidi et al., 2009). Their prominent usage was particularly notable in diseases linked to tissues intrinsically associated with metabolism, such as KIRC (kidney), LIHC (liver), PAAD (pancreas), and THCA (thyroid gland).

Moreover, our model consistently identified gene sets with crucial roles in epithelial cells. Specifically, APICAL SURFACE and HYPOXIA showed significant prevalence in BRCA (breast), COAD (colon), LUAD (lung), and STAD (stomach) data sets, all known to contain abundant epithelial cells (Kotha et al., 2015; Zeitouni et al., 2016).

The gene sets related to the cell cycle control cell growth and division and are not primarily related to distinguishing between different stages of cancers. Our model selected some of the cell cycle gene sets, which are DNA_REPAIR, MYC_TARGETS_V1, MTORC1_SIGNALING, with lower frequencies.

NOTCH_SIGNALING is a cell communication pathway that plays a crucial role in various cellular processes such as development, differentiation, proliferation, and apoptosis. It has shown that this pathway can have significant implications in the development, progression, and treatment of HNSC (Sun et al., 2014). This pathway was chosen with a high frequency by our model for the HNSC cohort.

For the LUSC cohort, the most frequently chosen pathway by our model is the PEROXISOME pathway. PEROXISOME involves cellular processes related to peroxisomes, which are organelles important for metabolism. It has been shown that certain genes within this pathway are expressed differently in tumors compared to normal tissue in LUSC, and LUSC development might be impacted by the altered expression of these genes (Zhang et al., 2020). These findings indicate that our model successfully captures biologically relevant gene sets associated with cancer progression.

Chapter 5

CONCLUSION

Understanding how biological mechanisms change as cancer progresses is crucial for advancing cancer research and improving patient care. The complexity of cancer necessitates a comprehensive investigation of the molecular pathways and signaling networks involved in tumor growth and spread. By delving deep into the complicated molecular interactions, new potential biomarkers for early detection and prognosis can be identified.

The knowledge gained from studying the biology underlying cancer progression helps to develop new and targeted treatments that can effectively address the specific weaknesses of cancer cells at different stages of the disease. With a deeper understanding of the biology underlying cancer progression, clinicians can make informed decisions about the most suitable treatment options for individual patients, maximizing treatment efficacy while minimizing side effects.

Hence, identifying the active biological processes within a tumor has emerged as a critical challenge. By gaining insights into the specific molecular mechanisms driving the transition from early- to late-stage cancer, we can develop more effective diagnostic and therapeutic strategies tailored to each stage of the disease. In this thesis, our focus was on distinguishing between early-stage and late-stage cancers using gene expression profiles.

In recent years, the utilization of machine learning methods in cancer research and pathology stage prediction has shown promising results. These powerful algorithms can analyze vast amounts of genomic and clinical data, which can uncover hidden patterns and predictive features. Our research focused on utilizing the potential of an MLP model to generate a sparse solution in the context of MKL. This allowed our model to effectively distinguish between early-stage and late-stage cancers based on gene expression profiles. Remarkably, our model offers not only high

predictive performance but also interpretability, which offers valuable insights into crucial genes and pathways driving cancer progression.

Our research stands as a substantial contribution to the field of cancer stage prediction through machine learning. At the heart of our contribution is the development of a novel MLP architecture, uniquely tailored for the task of distinguishing early-stage from late-stage cancers based on gene expression data. The core innovation lies in the introduction of the "eta layer", which operates as an attention-like mechanism and produces sparse kernel weights. This "eta layer" enhances both the predictive accuracy and interpretability of the neural network, allowing for a deeper understanding of the biological processes driving cancer progression. Furthermore, our approach leverages gene set-based kernel matrices, constructed using Hallmark gene sets, to effectively capture similarity patterns and reduce data dimensionality. Thus, our primary contribution lies in the creation of a powerful and biologically informative tool that advances cancer research by enabling accurate cancer stage predictions while shedding light on the critical molecular mechanisms underpinning cancer progression.

In our experiments, we benchmarked our MLP model against three well-established machine learning algorithms, which are RF, SVM, and MKL. The results demonstrated that our MLP model achieved better or comparable predictive performance on 15 cancer cohorts. The area under the receiver operating characteristic curve served as a robust performance metric for evaluating the models. The ability of our model to utilize fewer gene expression features in some data sets while still achieving competitive performance highlights its efficiency and potential. Moreover, our MLP model's frequently selected gene sets in specific data sets received validation from the existing scientific literature. This alignment between our model's predictions and established biological knowledge further enhances its credibility and biological significance.

In our quest to optimize our proposed MLP model, we conducted extensive experiments exploring various architectural modifications. However, it is important to note that these architectural changes did not lead to significant improvements in the model performance, and therefore, their results are not reported in this thesis.

We explored the impact of substituting the Lasso regularization function in the eta layer with the Elastic Net regularization, aiming to strike a different balance between feature selection and regularization. However, this alternative regularization approach did not yield recognizable improvements in the predictive capabilities of the model. In another exploration, we omitted the use of any regularization, resulting in non-sparse kernel weights. Additionally, we investigated an alteration to the weight initialization method in the eta layer. Specifically, we experimented with the He initialization as an alternative to the Xavier initialization. The outcomes did not demonstrate a notable enhancement in model performance. Furthermore, we experimented with the inclusion of dropout layers following the eta layer, a technique often employed to prevent overfitting and improve model generalization. Unfortunately, incorporating dropout did not lead to significant advancements in our model's performance.

While these architectural modifications did not produce the desired improvements in the model performance, it is important to acknowledge that our proposed MLP model still demonstrated strong predictive performance and interpretability, aligning with our primary research objectives.

5.1 Future Work

While this thesis has provided valuable insights into the classification of early-stage and late-stage cancers using gene expression profiles and the proposed MLP model, there are several avenues for future research and improvements that can be explored.

One possible avenue for future research is exploring different variations of attention mechanisms that can be integrated into the "eta layer" of our MLP. While the current mechanism assigns importance scores to gene set-based kernel matrices, investigating self-attention mechanisms could introduce a dynamic weighting process. This would allow the model to weigh the relevance of various gene set-based kernels in relation to one another and potentially might lead to more nuanced insights.

Another promising avenue for future work involves grouping different types of cancer that share similar biological mechanisms. By treating these cancers as distinct

tasks within a multitask learning framework, we can harness the shared pathways and genetic factors that play a role in cancer progression. This approach can effectively combine data from related cancer types and might lead to improved predictive accuracy.

BIBLIOGRAPHY

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems, software available from tensorflow. org (2015). *https://www.tensorflow.org*, 2015.

M. ApS. Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version*, 4:1, 2019.

H. Azzawi, J. Hou, R. Alanni, and Y. Xiang. A hybrid neural network approach for lung cancer classification with gene expression dataset and prior biological knowledge. In *Machine Learning for Networking: First International Conference, MLN 2018, Paris, France, November 27–29, 2018, Revised Selected Papers 1*, pages 279–293. Springer, 2019.

G. Bergers and L. Benjamin. Tumorigenesis and angiogenic switch. *Nature Reviews Cancer*, 3:401–10, 07 2003. doi: 10.1038/nrc1093.

S. Bhalla, K. Chaudhary, R. Kumar, M. Sehgal, H. Kaur, S. Sharma, and G. P. Raghava. Gene expression-based biomarkers for discriminating early and late stage of clear cell renal cancer. *Scientific Reports*, 7(1):44997, 2017.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

P. Broët, V. A. Kuznetsov, J. Bergh, E. T. Liu, and L. D. Miller. Identifying gene expression changes in breast cancer that distinguish early and late relapse among uncured patients. *Bioinformatics*, 22(12):1477–1485, 03 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl110.

J. Y. Chiang. Bile acid metabolism and signaling. *Comprehensive Physiology*, 3(3): 1191, 2013.

F. Chollet et al. Keras. `https://keras.io`, 2015.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

R. Diaz-Uriarte and S. Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3, 02 2006. doi: 10.1186/1471-2105-7-3.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

L. Ein-Dor, I. Kela, G. Getz, D. Givol, and E. Domany. Outcome signature genes in breast cancer: is there a unique set? *Bioinformatics*, 21(2):171–178, 2005.

L. Ein-Dor, O. Zuk, and E. Domany. Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer. *Proceedings of the National Academy of Sciences*, 103(15):5923–5928, 2006.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

I. J. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

M. Gönen and E. Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(64):2211–2268, 2011.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

H. Ishwaran and U. Kogalur. Random forests for survival, regression and classification (rf-src). 2016. r package version 2.4. 1. *https://cran. r-project. org/package= randomForestSRC*, 422, 2017.

Z. Jagga and D. Gupta. Classification models for clear cell renal carcinoma stage progression, based on tumor rnaseq expression trained supervised machine learning algorithms. *BMC Proceedings*, 8(Suppl 6 Proceedings of the Great Lakes Bioinformatics Confer):S2, 2014. ISSN 1753-6561. doi: 10.1186/1753-6561-8-s6-s2.

H. Kaur, S. Bhalla, and G. P. Raghava. Classification of early and late stage liver hepatocellular carcinoma patients from their genomics and epigenomics profiles. *PloS One*, 14(9):e0221476, 2019.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

P. L. Kotha, P. Sharma, A. O. Kolawole, R. Yan, M. S. Alghamri, T. L. Brockman, J. Gomez-Cambronero, and K. J. Excoffon. Adenovirus entry from the apical surface of polarized epithelia is facilitated by the host innate immune response. *PLoS Pathogens*, 11(3):e1004696, 2015.

F. Li, M. Yang, Y. Li, M. Zhang, W. Wang, D. Yuan, and D. Tang. An improved clear cell renal cell carcinoma stage prediction model based on gene sets. *BMC Bioinformatics*, 21(1):1–15, 2020.

A. Liberzon, C. Birger, H. Thorvaldsdóttir, M. Ghandi, J. P. Mesirov, and P. Tamayo. The molecular signatures database (msigdb) hallmark gene set collection. *Cell Systems*, 1 6:417–425, 2015.

B. Ma, F. Meng, G. Yan, H. Yan, B. Chai, and F. Song. Diagnostic classification of cancers using extreme gradient boosting algorithm and multi-omics data. *Computers in Biology and Medicine*, 121:103761, 2020.

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

H. Nam, B. C. Chung, Y. Kim, K. Lee, and D. Lee. Combining tissue transcriptomics and urine metabolomics for breast cancer biomarker identification. *Bioinformatics (Oxford, England)*, 25:3151–7, 09 2009. doi: 10.1093/bioinformatics/btp558.

B. O'Sullivan, J. Brierley, D. Byrd, F. Bosman, S. Kehoe, C. Kossary, M. Piñeros, E. Van Eycken, H. Weir, and M. Gospodarowicz. The tnm classification of malignant tumours—towards common understanding and reasonable expectations. *The Lancet Oncology*, 18:849–851, 07 2017. doi: 10.1016/S1470-2045(17)30438-2.

H. H. Pang, A. Lin, M. Holford, B. E. Enerson, B. Lu, M. P. Lawton, E. Floyd, and H. Zhao. Pathway analysis using random forests classification and regression. *Bioinformatics*, 22 16:2028–36, 2006.

J. Pati. Gene expression analysis for early lung cancer prediction using machine learning techniques: An eco-genomics approach. *IEEE Access*, 7:4232–4238, 2018.

A. Rahimi and M. Gönen. Discriminating early- and late-stage cancers using multiple kernel learning on gene sets. *Bioinformatics*, 34:i412–i421, 2018. ISSN 14602059. doi: 10.1093/bioinformatics/bty239.

A. Rahimi and M. Gönen. A multitask multiple kernel learning formulation for discriminating early-and late-stage cancers. *Bioinformatics*, 36(12):3766–3772, 2020.

A. Rashidi, T. B. Kirkwood, and D. P. Shanley. Metabolic evolution suggests an explanation for the weakness of antioxidant defences in beta-cells. *Mechanisms of Ageing and Development*, 130(4):216–221, 2009.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

H. Seo and D.-H. Cho. Cancer-related gene signature selection based on boosted regression for multilayer perceptron. *IEEE Access*, 8:64992–65004, 2020.

N. P. Singh, R. S. Bapi, and P. Vinod. Machine learning models to predict the progression from early to late stages of papillary renal cell carcinoma. *Computers in Biology and Medicine*, 100:92–99, 2018.

A. Statnikov, L. Wang, and C. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC Bioinformatics*, 9:319, 07 2008. doi: 10.1186/1471-2105-9-319.

W. Sun, D. A. Gaykalova, M. F. Ochs, E. Mambo, D. Arnaoutakis, Y. Liu, M. Loyo, N. Agrawal, J. Howard, R. Li, et al. Activation of the notch pathway in head and neck cancer. *Cancer Research*, 74(4):1091–1104, 2014.

T. The Cancer Genome Atlas Research Network. Comprehensive genomic characterization defines human glioblastoma genes and core pathways. *Nature*, 455(7216): 1061–1068, Oct. 2008. ISSN 00280836.

T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

Z. Wang, M. B. Gerstein, and M. Snyder. Rna-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10:57–63, 2009.

R. Xie, A. Quitadamo, J. Cheng, and X. Shi. A predictive model of gene expression using a deep learning framework. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 676–681. IEEE, 2016.

J. Xu, P. Wu, Y. Chen, and L. Zhang. Comparison of different classification methods for breast cancer subtypes prediction. In *2018 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 91–96. IEEE, 2018.

Z. Xu, R. Jin, H. Yang, I. King, and M. Lyu. Simple and efficient multiple kernel learning by group lasso. pages 1175–1182, 08 2010.

N. E. Zeitouni, S. Chotikatum, M. von Köckritz-Blickwede, and H. Y. Naim. The impact of hypoxia on intestinal epithelial cell functions: consequences for invasion by bacterial pathogens. *Molecular and Cellular Pediatrics*, 3(1):1–9, 2016.

S.-W. Zhang, J.-Y. Xu, and T. Zhang. Dgmp: identifying cancer driver genes by jointing dgcn and mlp from multi-omics genomic data. *Genomics, Proteomics & Bioinformatics*, 20(5):928–938, 2022.

X. Zhang, H. Yang, J. Zhang, F. Gao, and L. Dai. Hsd17b4, acaa1, and pxmp4 in peroxisome pathway are down-regulated and have clinical significance in non-small cell lung cancer. *Frontiers in Genetics*, 11:273, 2020.

Appendix A

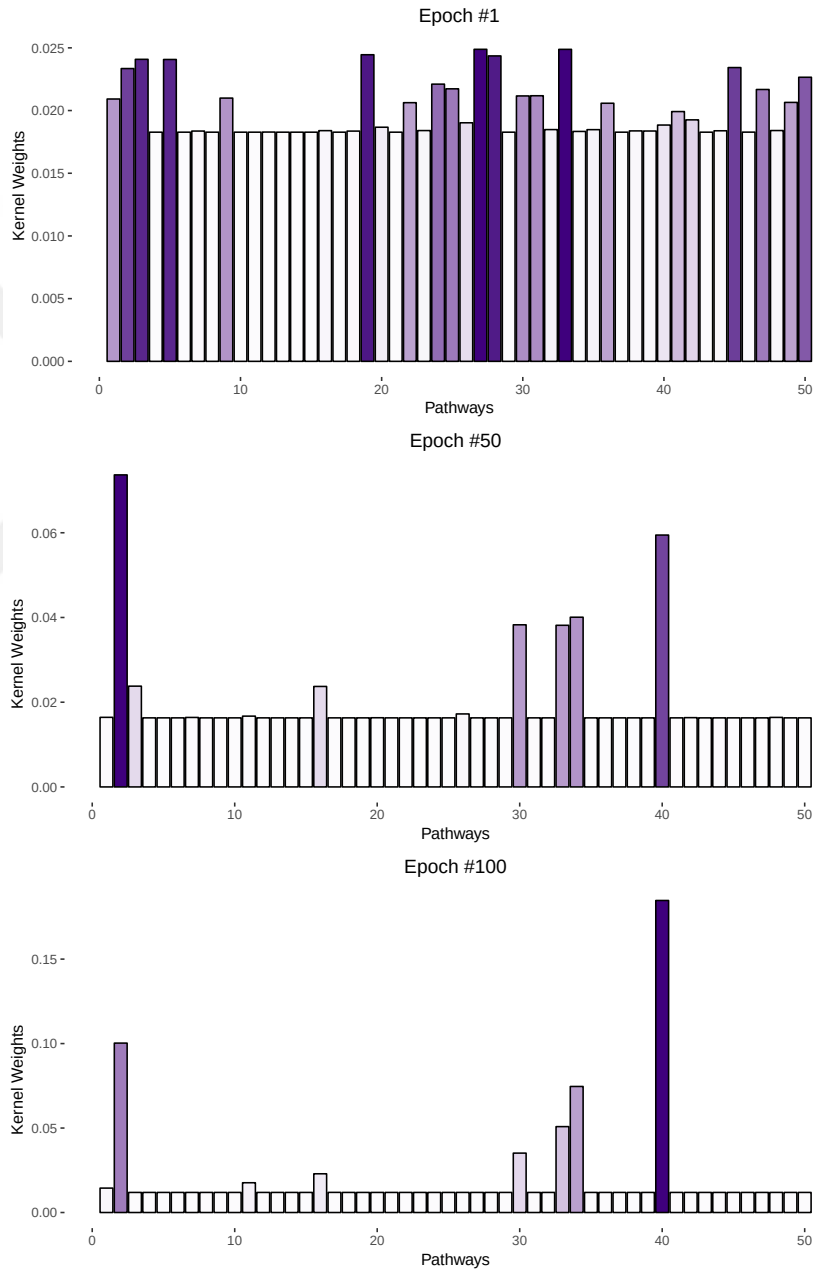

Figure A.1: The convergence of the kernel weights obtained throughout 100 epochs during a replication for the `READ` cohort.