Exploiting hierarchical and temporal information in building predictive models from EHR data

by

Anima Singh

S.M., Massachusetts Institute of Technology (2011)
B.S., Swarthmore College (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

© Massachusetts Institute of Technology 2015. All rights reserved.

Author .................................................................
Department of Electrical Engineering and Computer Science
March 31, 2015

Certified by ...........................................................
John Guttag
Dugald C. Jackson Professor
Thesis Supervisor

Accepted by ..........................................................
Leslie A. Kolodziejski
Chairman of the Committee on Graduate Students
Exploiting hierarchical and temporal information in building predictive models from EHR data

by

Anima Singh

Submitted to the Department of Electrical Engineering and Computer Science on March 31, 2015, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Clinical predictive modeling has the potential to revolutionize healthcare by allowing caregivers to allocate resources effectively, resulting not only in lower costs but also in better patient outcomes. Electronic health records (EHR), which contain large volumes of detailed patient information, are a great resource for learning accurate predictive models using advanced machine learning and data mining techniques.

In this thesis we develop techniques that can exploit available patient information to learn clinical predictive models while tackling the challenges inherent in the data. We present our work in the context of predicting disease progression in chronic diseases.

We present a novel feature representation that exploits hierarchical relationships between high cardinality categorical variables to tackle the challenge of high dimensionality. Our approach reduces feature dimensionality while maintaining variable-specific predictive information to yield more accurate predictive models than methods that ignore the hierarchy. For predicting incident heart failure, we show that by leveraging hierarchy in the diagnosis and procedure codes, we can improve the area under the receiver operating characteristic curve from 0.65 to 0.70 and the F-score from 0.37 to 0.40. Using simulation, we further analyzed the properties of the data that affect the amount of improvement obtained by leveraging hierarchy.

We also present a method to exploit temporal information in longitudinal EHR data. Our multitask-based machine learning approach captures time varying effects of a predictor. Our results show that by exploiting temporal information, our approach can improve risk stratification of patients with compromised kidney function over a model that only use the most recent patient information. Specifically, our proposed approach is able to boost sensitivity from 37.5% to 55.1% (for a precision of ≈ 50%) when identifying patients at high risk of rapid progression of renal dysfunction.

Thesis Supervisor: John Guttag
Title: Dugald C. Jackson Professor
Acknowledgments

I consider my decision to join John’s research group as one of the best decisions I have made. John has been an amazing advisor, mentor and friend. John’s passion for research and his ability to quickly understand a problem at hand and offer valuable suggestions has always inspired me. I appreciate the freedom and time he gives to his students, be it research or otherwise, while always being there to provide guidance and support. The work presented in this thesis would not have been possible without his guidance. John has made my graduate experience both fun and fulfilling. I will forever cherish working with him as a graduate student and I look forward to working with him in the future.

Many people have contributed to this work. I am extremely thankful to my thesis committee members, Professor Cynthia Rudin and Professor Tomás Lozano-Pérez for their advice and guidance. I have been fortunate to have had a chance to work with Professor Collin Stultz whose constructive criticisms and support have helped me grow as a researcher since my early years as a graduate student. Working alongside my labmates Jenna Weins, Gartheeban Ganeshapillai, Guha Balakrishnan, Joel Brooks, Yun Liu, Jen Gong, Amy Zhao and Davis Blalock has been a wonderful experience. Their feedback and suggestions have been invaluable in shaping my PhD. My sincere thanks to Dorothy Curtis and Sheila Marian for their help throughout my years at MIT.

Our clinical collaborators at the Mount Sinai Hospital have been critical to this work. Dr. Erwin Bottinger’s and Girish Nadkarni’s clinical expertise has been invaluable in identifying and formulating research problems of medical/clinical significance. My special thanks to Stephen Ellis and Rajiv Nadukuru for their assistance in getting access to the data used in my thesis.

I would like to thank my friends, especially my best friend Anisha Shrestha for cheering me on throughout my journey as a graduate student. Lastly, I dedicate my PhD to my parents who have made several sacrifices to enable me to get where I am today. Thank you mom for sacrificing your career for me and my sister. Thank you
dad for teaching me to always strive for the best in life. Many thanks to my sister Anita, brother-in-law Anul and my nephew Aayan for being there to always support me and my decisions.
Contents

1 Introduction 19

2 Background 27
   2.1 Clinical background 27
      2.1.1 Congestive Heart Failure 27
      2.1.2 Chronic Kidney Disease 28
   2.2 ICD-9 codes 29
   2.3 The Data 30
   2.4 Preprocessing the Data 32
   2.5 Study Population 33
      2.5.1 Congestive Heart Failure 33
      2.5.2 Chronic Kidney Disease 34

3 Leveraging Hierarchy in Medical Codes 37
   3.1 Introduction 37
   3.2 Related work 39
   3.3 Definitions 40
   3.4 Methods 41
      3.4.1 Feature Representation 41
      3.4.2 Model Learning Algorithm 46
   3.5 Clinical Tasks 49
      3.5.1 Predicting incident HF 49
      3.5.2 Predicting CKD progression 50
3.6 Experiments and Results ......................................................................... 50
  3.6.1 Experimental Setup ........................................................................ 50
  3.6.2 Performance Evaluation .................................................................... 52
3.7 Separating the effect of feature representation and feature dimensionality 54
3.8 Summary and Conclusions .................................................................... 57

4 Exploring the Effect of Leveraging Hierarchy via Simulation Study 59
  4.1 Generating Simulation Data ................................................................. 59
    4.1.1 Generating the underlying data ......................................................... 61
    4.1.2 Generating the examples ................................................................. 64
  4.2 Road-map of Experiments .................................................................... 65
  4.3 Experimental Setup ........................................................................... 68
  4.4 Performance of the Oracle ................................................................. 68
  4.5 Effect of the number of training examples and mass distribution .......... 69
  4.6 Effect of differing mass distributions between training and test sets .... 74
  4.7 Quantifying difference in mass distributions between training and test sets ......................................................................................... 80
  4.8 Effect of α .......................................................................................... 82
  4.9 Mimicking the effect of differing mass distributions when transitioning from ICD-9 to ICD-10 ................................................................. 85
  4.10 Summary and Conclusions ................................................................. 87

5 Exploiting Temporal Information in EHR Data ........................................ 89
  5.1 Introduction ....................................................................................... 89
  5.2 The Data ............................................................................................ 92
  5.3 Problem Formulation ......................................................................... 92
  5.4 Methods ............................................................................................ 96
    5.4.1 Non-Temporal Approach ............................................................... 97
    5.4.2 Temporal Approaches ................................................................. 98
  5.5 Experiments and Results .................................................................... 102
    5.5.1 Experimental Setup ..................................................................... 102
List of Figures

2-1 An excerpt from the hierarchy in diagnostic code 428 that represents heart failure. ............................ 30

2-2 The phenotyping algorithm developed by experts in the Mount Sinai Hospital. ................................. 35

3-1 List of ICD-9 codes associated with dummy patients: Patient A and Patient B. ........................................ 41

3-2 Raw Binary feature representation for Patient A and Patient B. ............................................................ 42

3-3 Top-level Binary feature representation for Patient A and Patient B. ................................................... 43

3-4 Propagated Binary feature representation for Patient A and Patient B. .............................................. 44

3-5 Block diagram of our proposed approach for feature construction. $\beta$ is the shrinkage parameter used for probability estimation. ............................................................... 45

3-6 Probability-based feature representation for Patient A and Patient B. .................................................. 47

3-7 Hierarchical structure between variables $x_1, x_2$ and $x_3$. ................................................................. 48

3-8 A schematic summarizing how multiple examples are generated per patient. Each rectangle represents an example corresponding to a time window of size $T$. We use $T = 1$ year for our experiments. ...................................................... 51

3-9 Relative risk of outcome in examples in the 5th quintile vs the 1st quintile. ........................................ 55
3-10 The average AUROC of RawL1 for different number of features with non-zero weights across the 100 experiments along with the standard error. The red and the green dot corresponds to the performance of ProbL2 and RawL2 respectively. The dotted lines indicate the corresponding number of features with non-zero weights.

4-1 A complete binary tree with $L = 4$. There are $V = 2^L - 1 = 15$ nodes in the tree. The number on the node represent the value of $v$ that denotes the node. For a complete binary tree with $L$ levels, the indices of the leaf nodes ranges from 1 to $2^{L-1}$.

4-2 A schematic illustration of the overview of the steps used in generating the simulation data. The user-specified inputs are discussed in greater detail in Section 4.1.1 and 4.1.2.

4-3 The average difference between probabilities of leaf node siblings in $T$ with 4 levels for different values of $\alpha$.

4-4 A schematic illustration of the steps used in generating the simulation data, along with the user-specified inputs for each step.

4-5 Heatmaps of the different mass distributions considered in Section 4.5 in which $M = M^{tr} = M^{tst}$. The corresponding observed level probability distributions are also shown.

4-6 The average performance of RawL2 and ProbL2 along with the best possible performance for the different $M = M^{tr} = M^{tst}$ (a) $M_d$ (b) $M_b$ and (c) $M_c$ (shown in Figure 4-5).

4-7 The average percentage improvement in performance obtained by leveraging hierarchy for different $N^{tr}_{leaf}$ and different mass distributions.

4-8 The different observed mass distributions for the training data set ($M^{tr}$) used in Section 4.6 to evaluate the effect the leveraging hierarchy when $M^{tr}$ and $M^{tst}$ are different. For mass distributions $M_d$ through $M_c$, the level (and the number) of the unseen nodes in the training data increases.
4-9 The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data shown in Figure 4-8.

4-10 The different observed mass distributions for the training data set (M_{tr}) that are used to evaluate whether the level of the unseen nodes affect the amount of improvement yielded. The figure shows one possible matrix for each of the mass distributions represented as \( \tilde{M}_s \) since the number of unseen nodes < the number of nodes in the level.

4-11 The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data for which there are 2 unseen nodes in level 2 (M_d), level 3 (M_g) and level 4 (M_h).

4-12 The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data for which there are 4 unseen nodes in level 3 (M_e) and level 4 (M_i).

4-13 The relationship between the distance between M_{tr} and M_{tst} and the magnitude of AUROC improvement yielded by ProbL2 over RawL2 by leveraging hierarchy.

4-14 The average percentage improvement in performance obtained by leveraging hierarchy when M_{tr} = M_{tst} = M_a (i.e., the distance between mass distributions is small) for different values of \( \alpha \).

4-15 The average percentage improvement in performance obtained by leveraging hierarchy when M_{tr} = M_f and M_{tst} = M_a (i.e., the distance between mass distributions is high) for different values of \( \alpha \).

5-1 Schematics illustrating how the risk stratification models were learned using (a) Non-temporal approach, (b) Stacked-Temporal approach and (c) Multitask-Temporal approach. V is the number of variables.
Illustration of the missing value estimation used to learn the final model that aggregates the predictions obtained from each task. Once the tasks are learned jointly using the multitask framework, the models are applied to the corresponding available time windows of the examples to produce the prediction score matrix $A$. A missing value estimation method is applied to obtained a complete prediction score matrix $\tilde{A}$. Specifically, we replace the missing value for example $i$ with the prediction from the nearest available time window. If there are two windows that are equally close, we take the average.

The fraction of examples in our dataset with the different number of time-windows for which at least one medical encounter is available in the EHR.

Average performance of the different methods for threshold 10% and 20%. The x-axis shows the number years of patient history that is considered for the model. The error bars show the standard error in the 100 splits.

Fraction of examples with a positive outcome in each predicted risk quintile for threshold 10% and 20%.

Comparison of the performance of Multitask-Temporal with Singletask-Temporal and Multitask-Temporal with imputation. Multitask-Temporal outperforms both Singletask-Temporal and Multitask-Temporal with imputation, highlighting the importance of learning the tasks jointly and the effect of noise due to imputation.

Temporal patterns in normalized variable weights of the 15 variables of the Multitask-Temporal model for threshold $= 20\%$. The bar shows color for the normalized variable weight in the interval $[-5, 25 \times 10^{-4}]$.

Examples illustrating two different types of facts and their corresponding dimensions.

Two facts from the Mount Sinai Data Warehouse.
A-3 Procedure, unit of measure and metadata description for the second fact shown in Figure A-2. The field LEVEL_2_EVENT, LEVEL_3_ACTION and LEVEL_4_FIELD contain the metadata information.

A-4 An excerpt from the D_MATERIAL table showing the MATERIAL_NAME for some of the medications.
List of Tables

2.1 The different stages of CKD and the corresponding GFR values [13]. . 29

2.2 ICD-9 codes that indicate heart failure. . . . . . . . . . . . . . . . . . 33

2.3 The ICD-9 codes used in the exclusion criteria (B8) in Figure 2-2. . 34

3.1 Definitions of different combinations of feature representation and model
learning algorithms. Hierarchy is leveraged in feature construction†,
model building§, or both‡. RawL2 does not use hierarchy. . . . . . . . 51

3.2 The summary of data used for Task-HF and Task-CKD. The value in
the parentheses is the number of positive examples. . . . . . . . . . . 52

3.3 Average performance of the different methods for Task-HF and Task-CKD.
The value within the parentheses is the standard deviation. The * in-
dicates that the average is significantly (p < 0.05) different from the
average performance of RawL2 as measured using a paired t-test. The
methods that perform the best are highlighted in gray. . . . . . . . . 53

4.1 The user-specified parameters required to generate the training and the
test data sets. The parameters that control the underlying behavior
of the data, i.e., the number of levels $L$ in the tree and the probability
of a positive label given each leaf node $P$ are kept the same for the
training and the testing data. The other two parameters are allowed
to vary between the training and the test dataset. . . . . . . . . . . 67
4.2 Different mass distributions of the training set used to evaluate whether the level of the unseen nodes affect the amount of improvement yielded by leveraging hierarchy. The mass distributions \( \tilde{M} \) are discussed in the text in more detail. 

4.3 KL-divergence when using ICD9-original and ICD9-truncated for training. For both cases, ICD9-original is used for testing.

4.4 KL-divergence when using ICD9-original and ICD9-truncated for training. For both cases, ICD9-original is used for testing.

5.1 Predictors extracted from the past medical history of a patient for predicting progression of kidney function loss. The numbers in parenthesis for each predictor group is the number of binary variables associated with the given set of predictors. For demographics and numerical predictors, the table also shows the statistics for the patients in the recent EHR data.

5.2 The average number of positive examples for different thresholds in the holdout sets. The value in the parenthesis shows what fraction of the total number of examples is positive.

5.3 The mean number of variables considered per time window for non-temporal and temporal approaches across 100 splits. The number in the parenthesis is the standard deviation across the splits.

5.4 Performance comparison of Multitask-Temporal approach with the models that use the most recent time-window. The * indicates that the average is significantly \( p < 0.05 \) different from the average performance of Non-temporal when evaluated using a matched t-test. For the performance measures that are statistically significant, we achieved \( p \)-values \( < 0.001 \).

A.1 The description of the dimensions of a fact that we use.

A.2 Different procedure descriptions used to describe creatinine values stored in the database.
Chapter 1

Introduction

Advances in medical information technology and the wide-scale adoption of electronic health records have made unprecedented amounts of clinical data available for research. Electronic health records (EHR) in hospitals keep track of patients’ health and medical history along with their interaction with the care providers over time. The availability of such data also offers an opportunity to transform clinical medicine using data-driven predictive analytics.

Advanced machine learning and statistical learning techniques offer the potential to exploit available EHR data to learn accurate clinical predictive models. The future trajectory of a patient’s health depends on the past and the current state of the patient along with the interventions the patient has undergone. However, the relationship between the patient’s trajectory and the medical history is complex. Predictive models learned from historical data can potentially allow us to detect the underlying patterns that can help to accurately predict patient’s future state of health.

Accurate predictive models can be used for risk stratification of patients, i.e., identifying patients who are at high risk of adverse clinical outcomes. This can be critical in helping doctors guide clinical decision making. E.g., if an individual is predicted to be a high risk patient, care providers can either intervene with appropriate therapy or monitor the patient more closely. This would not only improve patient outcomes but would also allow the care providers to allocate their resources effectively, resulting in reduced healthcare costs.
While EHR data holds the promise of data-driven predictive analytics, clinical data present a multitude of technical challenges to learning accurate predictive models.

1. **Complex data:** EHR data is highly complex. Each patient record consists of multiple time series of clinical variables. The clinical variables include both numerical and categorical variables such as laboratory test results, medications, diagnoses, procedures and their outcomes. EHR might also contain physiological time series data and patient recorded as free text narratives in the form of discharge and summary notes.

2. **Small Data and High Dimensionality:** Clinical predictive models are often developed for a specific sub-population of patients to predict a specific outcome. Hence, the number of patients of interest is usually significantly smaller than the number of patients in an EHR. On the other hand, there can potentially be thousands of variables representing patient information from a single hospital visit. The challenge lies in learning accurate models from small datasets with high dimensionality.

3. **Imprecise information:** Patient data stored in EHR are often imprecise. While diagnoses assigned to a patient are often stored in EHRs, sometimes the diagnoses are not specific. E.g., an EHR record of a patient may contain a diagnosis of heart failure when the true underlying condition is an acute systolic heart failure. This lack of specificity could be 1) because the care providers were unable to identify the true underlying condition at the time the data was recorded, or 2) because of the lack of meticulous recording of patient information in the EHR.

4. **Constantly evolving medical coding standards:** EHRs in most hospitals use the International Classification of Diseases (ICD) coding system to record diagnosis associated with patients. However, these coding systems are revised regularly to meet expanding disease classification needs [11]. One of the major revisions
(from ICD-9 to ICD-10) is expected to be adopted in hospitals in the US by October 1, 2015 [3]. ICD-10 coding contains significantly more codes with greater specificity than ICD-9 codes. When hospitals transition from the ICD-9 to ICD-10 coding system, new patient data will contain codes that are not used in the historical data. Traditional approaches to learning predictive models will not be able to take advantage of the more specific codes in the new patient data.

5. **Irregular sampling:** EHR data are recorded only during a healthcare episode or when a patient visits the hospital for routine medical care. Such irregular sampling leads to cases where there are few or no samples within a time window. Traditional time series analysis techniques therefore are unable to handle such data.

6. **Missing data:** Since different hospitals use different EHR systems and patient data across hospitals are not integrated. As a result, the medical history of a patient might be missing when he/she moves between health institutions. Therefore, different patients have different lengths of patient data available in an EHR of a specific hospital.

7. **Temporal analysis:** Patient data in EHRs are temporal in nature. Temporal analysis of patient’s entire medical history holds promise in yielding more accurate models than models developed using data from a single fixed point in time (a commonly used approach) [28, 16]. However, it is not obvious how to capture the temporal dependencies in predictive models.

In this thesis, we focus on developing methods to learn more accurate predictive models from existing clinical data while addressing some of the challenges described above. We present our methods in the context of learning predictive models for chronic disease management, specifically congestive heart failure and chronic kidney disease. A chronic disease is a long-lasting health condition that can be managed but not cured. Chronic diseases such as heart disease and kidney disease are among the most common of all health problems in the US. As of 2012, about 117 million Americans have one or more chronic conditions. Chronic conditions are the leading cause of
death in the US. The majority of healthcare costs in the US are also associated with chronic conditions, with heart disease alone accounting for an estimated $315.4 billion in 2010 [10]. Low-cost automated techniques that can accurately predict adverse outcomes can have significant economic and health benefits.

In the context of the above clinical applications, we propose methods to 1) exploit structure in the ICD codes to address the challenges of small data, high dimensionality and evolving medical coding standards and 2) leverage temporal information in the EHR data in order to capture temporal dynamics while handling irregular sampling and missing information in the data. The contributions of this dissertation are as follows:

1. **Leveraging hierarchical structure in ICD codes**: We investigate different methods to leverage structural relationships in ICD-9 codes when learning clinical predictive models. We compare methods that leverage hierarchy by 1) incorporating the structural information during feature construction, 2) using a learning algorithm that considers the structure in features when learning a model, and 3) doing both.

   We present a novel probability-based feature construction approach that is simple and effective in exploiting the structural relationships among the ICD-9 codes. Unlike the commonly used feature representations for ICD-9 codes, our approach is able to preserve predictive information and reduce feature dimensionality to yield risk stratification models that generalize well. Our approach is particularly relevant for clinical applications as hospitals transition from using ICD-9 to ICD-10 to record patient information. Exploiting hierarchical relationships allow us to handle ICD codes in new patient data that are not seen before in historical data.

   We compare predictive models that ignore hierarchy with ones that leverage hierarchy for the task of predicting incident heart failure **Task-HF** and predicting chronic kidney disease progression **Task-CKD**. Our results demonstrate that by exploiting structural relationships between the ICD-9 codes, we can yield
predictive models with better risk stratification performance. Specifically, we show that methods that use hierarchy outperform the conventional approach in F-score (0.44 vs 0.36 for Task-HF and 0.40 vs 0.37 for Task-CKD) and in the area under that receiver operating curve AUROC (0.70 vs 0.65 for Task-HF and 0.69 vs 0.66 for Task-CKD).

2. Analyzing the effect of leveraging hierarchy via simulation: We perform a simulation study to better understand the properties of the data that affect the amount of improvement that can be achieved by leveraging hierarchy. Using a simulation dataset that is a simple approximation of real clinical data, we analyze how the following properties affect the amount of performance improvement:

- The number of training examples
- The difference in the distribution of examples across nodes in the hierarchy tree between the training and the test data
- The relationship between the probability of the label given a node and the probability of the label given its sibling in the tree.

Our results demonstrate that while the magnitude of improvement varies depending on the different properties of the training and the test data, leveraging hierarchy never underperforms the commonly used approach that ignores structural relationships among the nodes.

We also perform experiments to mimic the scenario when hospitals transition from using the ICD-9 to ICD-10 codes. Our results suggest that leveraging hierarchy can effectively handle the challenge of evolving medical coding standards and yield predictive models with a large performance gain when test data contain codes that are unseen in the training data.

3. Exploiting temporal information: We propose a novel approach to use temporal information in longitudinal EHR data to develop predictive models.
Our multitask based learning approach attempts to capture the time-varying effects of a predictor (collected from multiple time points in the past) on the outcome.

Our proposed approach has two main characteristics:

- It handles missing data without imputation, thereby reducing noise during learning, and
- The multitask based learning reduces overfitting due to high dimensionality of features from multiple time points.

Our results show that predictive models that exploit temporal information can improve risk stratification of patients with compromised kidney function. When compared to the models that use the most recent patient information, the predictive models learned using our proposed approach were able to boost the sensitivity from 37.5% to 55.1% (for a precision of $\approx 50\%$) when identifying patients at a high risk of rapid progression of renal dysfunction.

The remainder of the thesis is organized as follows.

- Chapter 2 presents background on the clinical applications considered in this thesis and the datasets used. It also discusses the criteria used to identify the study population used in our analysis.

- Chapter 3 explores methods to leverage structural relationships in ICD-9 codes when learning clinical predictive models and demonstrates their advantage over the commonly used method that ignores the relationships. It also demonstrates the importance of coupling an appropriate feature representation with an appropriate learning algorithm when leveraging hierarchy in predictive models.

- Chapter 4 presents the analysis of simulation data to better understand the effect of various properties of data on the amount of improvement leveraging hierarchy can yield. It demonstrates that depending on the properties of the data, leveraging hierarchy can either improve or yield comparable performance relative to the technique that ignores the hierarchy.
• Chapter 5 presents a novel approach to incorporate temporal information in the longitudinal patient data in predictive models by capturing varying effects of a predictor from different points in time. It discusses the ability of the proposed approach to handle challenges such as missing data and high dimensionality while exploiting temporal information to yield a predictive model that outperforms the one that only uses the most recent patient information.

• Chapter 6 summarizes the contributions of this thesis and directions for future work.
Chapter 2

Background

In this chapter, we begin with a discussion of the clinical background of the two applications we consider in this thesis. Next, we discuss ICD-9 codes, a widely used classification system for describing diagnoses and procedures. In Section 2.3, we provide a description of the data that we use in this dissertation. We discuss the different types of patient information available in the EHR along with how the data is stored and extracted from different tables in the database. Section 2.4 describes how we preprocess the extracted data before we use them in our predictive models. Finally, in Section 2.5 we discuss the criteria used to identify the study population used for our analyses.

2.1 Clinical background

2.1.1 Congestive Heart Failure

Congestive heart failure (CHF) is a condition in which the heart muscle becomes weak and loses its ability to pump blood normally. This results in fatigue and shortness of breath, and can severely affect people’s ability to carry out daily activities. About 5.1 million people in the United States have heart failure. In 2009, it was one of the contributing causes for 1 in 9 deaths. Heart failure accounts for nearly $32 billion each year in healthcare expenditures and lost productivity due to missed days at
work [15].

Incident heart failure is defined as the first occurrence of hospitalization that includes a diagnosis of congestive heart failure [25]. Heart failure is a progressive disease and often leads to symptomatic presentation only late in the course of disease progression. Approximately 60% of men and 45% of women die within 5 years of incident HF [24]. In most cases, the therapies are only given to a patient during the symptomatic phase of HF, after extensive cardiac remodeling has already occurred. Low cost strategies to identify patients at high risk of incident HF is of great clinical interest. Accurate prediction of incident HF can allow clinicians to explore therapy options that can reduce complications associated with HF and alter the course of disease progression [17].

### 2.1.2 Chronic Kidney Disease

Chronic kidney disease (CKD) is an increasingly prevalent condition associated with a high degree of mortality and morbidity. About 26 million adults in the United States have CKD and millions of others are at increased risk [12]. Annually, Medicare spends over $33 billion to provide care for patients with CKD [37]. Patients with CKD progressively lose kidney function over time. However, since CKD is a silent condition in early stages of the disease, early identification of such patients presents a challenge. Clinical decision making for CKD is also challenging because of the variability in rates of disease progression among patients [28]. Therefore, there is a need to develop risk stratification techniques that can recognize patients with early stage CKD who are likely to progress quickly.

The level of kidney function is determined using the glomerular filtration rate (GFR), a measure of how well the blood is filtered by functioning nephrons of the kidney. The GFR is approximated using the creatinine clearance rate, the volume of blood plasma that is cleared of creatinine per unit time. The different stages of CKD is defined based on the GFR values (Table 2.1).

The measurement of the creatinine clearance is usually done from a 24-hour urine specimen and a serum specimen obtained during the same collection period [4]. How-
Table 2.1: The different stages of CKD and the corresponding GFR values [13].

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
<th>GFR (mL/min per 1.73m²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kidney damage with normal kidney function</td>
<td>90 or above</td>
</tr>
<tr>
<td>2</td>
<td>Kidney damage with mild loss of kidney function</td>
<td>89 to 60</td>
</tr>
<tr>
<td>3a</td>
<td>Mild to moderate loss of kidney function</td>
<td>59-44</td>
</tr>
<tr>
<td>3b</td>
<td>Moderate to severe loss of kidney function</td>
<td>44-30</td>
</tr>
<tr>
<td>4</td>
<td>Severe loss of kidney function</td>
<td>29 to 15</td>
</tr>
<tr>
<td>5</td>
<td>Kidney Failure</td>
<td>&lt;15</td>
</tr>
</tbody>
</table>

ever, since the test is time consuming, GFR is usually estimated from serum creatinine, a by-product of muscle breakdown. In this thesis, we use the CKD-EPI (Chronic Kidney Disease Epidemiology Collaboration) [23] formula to get an estimated-GFR (eGFR) from serum creatinine. Since serum creatinine depends on the muscle mass of an individual, the equation uses race, gender and age along with the serum creatinine value to compute eGFR.

Although convenient, using serum creatinine to estimate the underlying GFR of a patient is not always accurate. Serum creatinine levels in an individual can fluctuate a lot. E.g., eating protein based meals and increased cell breakdown (due to illnesss) can increase serum creatinine in the blood, while eating less and reduction in muscle mass can cause serum creatinine to fall. These fluctuations cause the eGFR to fluctuate even when the underlying filtration rate of the kidney has not changed. Therefore, when using eGFR to determine CKD stage from serum creatinine, it is important to account for these noisy fluctuations.

### 2.2 ICD-9 codes

ICD-9 is the most widely used classification system for describing diagnoses and procedures developed by the International Statistical Classification of Diseases and Related Health Problems [11]. The ICD-9 codes are organized in a hierarchy where an edge represents an ‘is-a’ relationship between a parent and its children. Hence, the codes become more specific as we go down the hierarchy. There are about 17,000 diagnostic and 1200 procedure codes. Each three digit diagnostic code and each two
Figure 2-1: An excerpt from the hierarchy in diagnostic code 428 that represents heart failure.

A node at a higher level of the hierarchy tree represents a more general concept than the nodes at a lower level. Figure 2-1 shows an excerpt from one of the hierarchy trees that form the ICD-9 classification system.

The hierarchy tree represented by a top-level code is at most 3 levels deep. A top-level code is a three-digit code for diagnoses or a two-digit code for procedures. The first digit after the decimal represent the second level and the second digit after the decimal represent the third level in the hierarchy. Figure 2-1 shows an excerpt from the hierarchy within the top-level diagnostic code 428 that represents a diagnosis of heart failure. The ICD-9 coding system is a collection/forest of such hierarchy trees.

2.3 The Data

In this thesis, we use data from EHR of patients in the Mount Sinai Medical Center that encompasses the Mount Sinai Hospital and the Mount Sinai School of Medicine in New York city. The Mount Sinai Hospital is a tertiary-care teaching facility with 1171 beds. In 2011, nearly 70,000 people were treated at Mount Sinai as inpatients, and there were nearly 800,000 outpatient visits to the medical center.

Our data comes from the Mount Sinai Data Warehouse (MSDW) that consists of clinical, operational and financial data derived from patient care processes of the
Mount Sinai Hospital and Mount Sinai Faculty Practice Associates. From 2003-2011, the MSDW collected data from $\approx 3$ million patients. However, the study population of interest is significantly smaller than the total number of patients. We discuss this in more detail in Section 2.5.

The clinical data stored in MSDW comes from 23 different hospital sources including two different EHR systems: Cerner and Epic. This introduces challenges when trying to use the EHR data since different systems use different standards and definitions. We also had access to clinical data of patients enrolled in the Institute for Personalized Medicine Biobank program at Mount Sinai. This includes data from $\approx 20,000$ patients from the MSDW who provided consent for open future research including permission for recontacting.

Initially, we only had access to the EHR data of the patients enrolled in the biobank program. Therefore, for our initial work on leveraging structure in the ICD-9 codes (Chapter 3), we use data from the $\approx 20,000$ patients enrolled in the biobank program. For our more recent work on exploiting temporal information (Chapter 5), we use data from the MSDW. Appendix A presents the details of the database and how the patient data is stored.

We extract patient demographics, laboratory results, diagnoses, procedures, vital signs and medications from the database. The database contains demographic information such as date of birth, gender and the race of the patients. We extract the diagnoses and procedures recorded using the ICD-9 codes. We consider medications that are used for the treatment of hypertension, hyperlipidemia and diabetes, and medications that are known to nephrotoxic. When extracting the medications we disregard the route, dosage and the strength of the medication. Moreover, we extract only the prescription medications for our work since we are interested in medications that are used to treat the patient’s chronic condition. Appendix A provides a summary of how we extract the patient information from the tables in the database.

Unfortunately, the prescription medication data in the database is noisy. First, although there is a start date and an end date associated with a prescription, there is no way to find out if the patient actually takes the prescribed medication during
that period. Second, if a prescribed medication is canceled, there is not an easy way to keep track of the cancellations. Third, if a patient fills his/her prescription in pharmacies outside the hospital, they are not recorded in the database.

2.4 Preprocessing the Data

The data extracted from the database are either numerical or categorial. Numerical data includes age, laboratory results and vital signs. Categorical data includes the ICD-9 codes representing the diagnoses and procedures, and the medications.

We represent all of the data using binary variables. Specifically, we represent diagnoses, procedures and medications as a binary variable indicating whether or not the patient associated with the example was assigned an ICD-9 code or prescribed a medication during a specified time-window in the past. We discretize the numerical predictors into four bins based on the quartiles of the corresponding predictor and then map them into binary variables. For example, we map the mean systolic blood pressure for the most recent time window, into four bins: $SBP \leq 120$, $120 < SBP \leq 130$, $130 < SBP \leq 140$ and $SBP > 140$, each corresponding to a binary variable. For a patient with $SBP$ of 125 mm Hg, we set the binary variable corresponding to $120 < SBP \leq 130$ to 1 and others to 0.

In MSDW, there are more than 17,000 different distinct medications that are prescribed. The database refers to the medications using it material name. When we map the material names into a binary feature vector, this results in very high dimensionality. The material name can either contain a brand name or the ingredient contained in the medication. Therefore, there might be two distinct material names with the same ingredient used to treat the same condition. While medications also have a hierarchy, the information was unavailable to us. Therefore, we use an alternative approach to reduce dimensionality. First, we map each material name to a drug name, such that METFORMIN 1000 MG TABLET and METFORMIN 1000 MG TABS are mapped to the drug METFORMIN. Next, we map each drug name to an ingredient so that two drugs with the same ingredient are considered the same, e.g.,
<table>
<thead>
<tr>
<th>ICD-9 code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>402.01</td>
<td>Malignant hypertensive heart disease with heart failure</td>
</tr>
<tr>
<td>402.11</td>
<td>Benign hypertensive heart disease with heart failure</td>
</tr>
<tr>
<td>402.91</td>
<td>Unspecified hypertensive heart disease with heart failure</td>
</tr>
<tr>
<td>404.01</td>
<td>Hypertensive heart and chronic kidney disease malignant with heart failure with chronic kidney disease stage 1 through stage 4 or unspecified</td>
</tr>
<tr>
<td>404.03</td>
<td>Hypertensive heart and chronic kidney disease malignant with heart failure with chronic kidney disease stage 5 or end stage renal disease</td>
</tr>
<tr>
<td>404.11</td>
<td>Hypertensive heart and chronic kidney disease benign with heart failure with chronic kidney disease stage 1 through stage 4 or unspecified</td>
</tr>
<tr>
<td>404.13</td>
<td>Hypertensive heart and chronic kidney disease benign with heart failure with chronic kidney disease stage 5 or end stage renal disease</td>
</tr>
<tr>
<td>404.91</td>
<td>Hypertensive heart and chronic kidney disease unspecified with heart failure with chronic kidney disease stage 1 through stage 4 or unspecified</td>
</tr>
<tr>
<td>404.93</td>
<td>Hypertensive heart and chronic kidney disease unspecified with heart failure with chronic kidney disease stage 5 or end stage renal disease</td>
</tr>
<tr>
<td>428.*</td>
<td>Heart failure</td>
</tr>
</tbody>
</table>

Table 2.2: ICD-9 codes that indicate heart failure.

Both LISINOPRIL and PRINIVIL are mapped to the ingredient LISINOPRIL. This mapping significantly reduces the dimensionality. When considering medication used to treatment of hypertension, hyperlipidemia and diabetics, and medications that are known to be nephrotoxic, the mapping resulted in about 110 distinct medications. The mapping was provided to us by the experts in the Mount Sinai Hospital.

### 2.5 Study Population

#### 2.5.1 Congestive Heart Failure

We consider a patient to have heart failure if the patient has any one of the ICD-9 codes shown in Table 2.2. We identify the time of incident heart failure as the date of admission for the medical encounter that contains one of the ICD-9 codes. We only consider the medical encounters that are inpatient or emergency visits.

We consider the clinical application of predicting incident heart failure using only the patients in the biobank program. Out of ≈ 20,000 patients, 1980 patients had one of the ICD-9 codes shown in Table 2.2. Unfortunately, the database does not contain
<table>
<thead>
<tr>
<th>ICD-9 code(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>042.<em>-044.</em></td>
<td>Human immunodeficiency virus (HIV) infection</td>
</tr>
<tr>
<td>282.6</td>
<td>Sickle cell disease</td>
</tr>
<tr>
<td>581.*</td>
<td>Chronic glomerulonephritis</td>
</tr>
<tr>
<td>582.*</td>
<td>Nephritic and nephropathy</td>
</tr>
<tr>
<td>446.*</td>
<td>Polyarteritis nodosa and allied conditions</td>
</tr>
<tr>
<td>447.6</td>
<td>Vasculitis</td>
</tr>
<tr>
<td>753.*</td>
<td>Renal agenesis and dysgenesis</td>
</tr>
</tbody>
</table>

Table 2.3: The ICD-9 codes used in the exclusion criteria (B8) in Figure 2-2.

information about whether a hospitalization with an HF diagnosis is the first ever the patient has had. Therefore, we consider a hospitalization with an HF diagnosis as incident heart failure, only when the date of the first fact recorded for the patient is at least 6 months before the date of hospitalization. A total of 477 patients satisfied this criterion.

### 2.5.2 Chronic Kidney Disease

In our study, we consider patients with chronic kidney disease who have also been diagnosed with hypertension, diabetes or both. We focus on this population because approximately two third of cases with compromised renal function are attributable to diabetes or hypertension [22].

We used a phenotyping algorithm developed by experts in the Mount Sinai Hospital to identify our patient population (Figure 2-2). Since GFR measurements were not recorded in the database, we use the serum creatinine measurements along with other patient information to compute eGFR using the CKD-EPI formula (as discussed in Section 2.1.2). We consider only the outpatient eGFR measurements when using those criteria. This was to ensure that the eGFR measurements considered were not due to acute clinical events during an inpatient stay. In B8, we exclude patients associated with any of the ICD-9 codes shown in Table 2.3.

We extracted data from patients with CKD from both the biobank database and the MSDW. There were 1800 patients and 10,000 patients with CKD in the biobank and the MSDW respectively. To ensure that we have reliable patient information
Figure 2-2: The phenotyping algorithm developed by experts in the Mount Sinai Hospital.

available for each of the patients, we use additional inclusion criteria (discussed in Chapter 3 and Chapter 5). This further reduces the size of our study population.
Chapter 3

Leveraging Hierarchy in Medical Codes

3.1 Introduction

Electronic health records (EHRs) contain information that can be used to make clinically useful predictions about the future trajectory of a patient’s health. Important parts of this information are recorded using ICD-9 codes.

ICD-9 is the most widely used classification system for describing diagnoses and procedures [1]. It consists of a collection of trees that represent relationships among the ICD-9 codes. A node at a higher level of the hierarchy tree represents a more general concept than the nodes at a lower level. Figure 2-1 shows an excerpt from one of the hierarchy trees that form the ICD-9 classification system.

The most commonly used representation for ICD-9 codes in predictive models ignores the hierarchical structure and maps each individual code to a separate binary feature to indicate presence or absence of a diagnosis or procedure [36]. We will refer to this approach as Raw Binary.

In this chapter, we demonstrate that the structure of the ICD-9 hierarchy is a source of potentially useful information that can improve the performance of predictive models. The fundamental assumption in leveraging the ICD-9 hierarchy is that neighboring nodes share similar relationship with the outcome of interest. In Chap-
ter 4, we define what we mean by similar relationship with the outcome, and use simulation data to evaluate how this similarity in relationship, along with other properties of the data, affect the improvement we achieve by leveraging hierarchical structure between variables. A simple representation that is based on this assumption is **Top-level Binary**. It aggregates all the codes within a tree to a single binary feature [36]. A general rule of thumb is to use **Raw Binary** when the number of samples is much greater than the feature dimensionality, since finer diagnostic levels can lead to better discrimination. On the other hand, if feature dimensionality is large relative to the number of samples, **Top-level Binary** is preferred to avoid overfitting.

The issue of overfitting is particularly important when learning predictive models for medical applications. Medical datasets are high-dimensional and since predictive models are developed for a subpopulation of patients with a specific clinical condition, the number of patients of interest is usually significantly smaller than the patient population in an EHR. E.g., the biobank database (See Chapter 2) contains EHRs of 20,000 patients. However, when considering patients with chronic kidney disease for whom sufficient data is available for analysis, we were left with only 510 patients. These characteristics present a challenge in learning generalizable predictive models for medical applications. While **Top-level Binary** aims to reduce overfitting by reducing feature dimensionality, it fails to take advantage of the specificity in diagnoses and procedures that can enhance performance. With the advent of ICD-10, which has more than 69,000 distinct codes to provide greater details and specificity [5], the challenge of high dimensionality will get worse.

Learning supervised predictive models involves two components: 1) feature construction and 2) model building. One can leverage the structure in the data by incorporating the information during feature construction, or by using a learning algorithm that addresses the structure in the data when building a model, or doing both. In this chapter, we explore these different approaches to leveraging hierarchy in ICD-9 codes.

One of our main contributions is a novel feature engineering approach to represent hierarchical categorical variables such as ICD-9 codes. Our approach is able
to preserve predictive information and to reduce feature dimensionality to prevent overfitting.

We evaluate different methods to leverage hierarchy by developing clinical models using EHR data for two real-world clinical applications: 1) predicting whether a patient at high risk of heart failure will be admitted to the hospital within a year with congestive heart failure (Task-HF), and 2) predicting whether a patient with chronic kidney disease will decline quickly or slowly (Task-CKD). Our results demonstrate that by exploiting hierarchical relationships between variables, we can yield predictive models with better risk stratification performance. Our results also demonstrate the importance of coupling an appropriate feature representation with an appropriate learning algorithm to maximize the performance boost.

3.2 Related work

ICD-9 codes have been shown to be useful for identifying phenotypes in genome-wide association studies \cite{7, 32} and for predictive modeling of adverse outcomes in patients. Some of the work in predictive modeling that use ICD-9 codes include: detection of epidemics \cite{39}, early detection of diabetes \cite{20} and heart failure \cite{36} and risk stratification of hospital-associated infections \cite{40}. These past studies use either \textbf{Raw Binary} or \textbf{Top-level Binary} representation for ICD-9 codes.

Recently, there has been interest in methods to leverage ICD-9 hierarchy. Perotte et al. \cite{30} use a hierarchy-based SVM for automated ICD-9 code assignment to discharge summary notes. Their task is to use features derived from the discharge summary notes to generate appropriate ICD-9 codes. In contrast, the focus of our work is to leverage hierarchy when using ICD-9 codes as features for predictive modeling.

Most of the research on leveraging structural information in predictors has been on developing learning algorithms. Supervised group lasso \cite{44, 33} and overlapping group lasso \cite{41} are the most widely used learning methods that take into account group structure. More specifically, they use the group information in features to
enforce regularization constraints. These methods have been used to develop predictive models using gene expression data as features. These learning methods use the cluster structure in genes, where a cluster consist of genes with co-ordinated functions \cite{27} \cite{41}. Overlapping group methods can also be used for variables with hierarchical structures \cite{45} \cite{18}. To the best of our knowledge, these methods have never been used in the context of leveraging hierarchy in ICD-9 codes.

3.3 Definitions

In this section, we provide notations and definitions used in the chapter.

- $n$ represents a node in a hierarchy tree. Each node corresponds to an ICD-9 code.

- $\hat{n}$ represents the parent of node $n$ in the hierarchy.

- $T_n$ represents the subtree of the hierarchy rooted at node $n$.

- $N_n$ represents the set of nodes within $T_n$.

- $t$ represents a node corresponding to a top-level code. Since $t$ is a top-level code, $\hat{t} = \text{null}$. Hence, $T_t$ represents the tree corresponding to the top-level code $t$. Figure 2-1 shows the hierarchy $T_{428}$ for $t = 428$.

- $D = \{ (x_i, y_i) \mid x_i \in \mathbb{R}^f, y_i \in \{-1, 1\}\}_{i=1}^N$ represents the dataset, where $f = \text{dimension of the feature space}$ and $N = \text{number of examples}$. $y_i = 1$ if the patient corresponding to example $i$ suffers a positive (adverse) outcome, and $y_i = -1$ otherwise.

- Top-level code refers to each of the three digit diagnostic code or the two digit procedure code associated with a hierarchy tree. Each three digit diagnostic code and each two digit procedure code is associated with a hierarchy tree.
3.4 Methods

In this section, we describe feature representations and model learning algorithms that disregard hierarchy followed by those that leverage hierarchy.

3.4.1 Feature Representation

We begin by discussing Raw Binary, the only feature representation we will examine that ignores hierarchy. Next, we present three different feature representations that use hierarchical information in the ICD-9 codes. First, we describe Top-level Binary that uses hierarchy to cluster all distinct values of the variable into a single variable. Propagated Binary uses structural relationships between each distinct value within the hierarchy. Lastly, we present our proposed Probability-based features, a novel feature representation that captures hierarchy.

To illustrate the different feature representations, we consider two dummy patients, referred to as Patient A and Patient B, who are assigned the ICD-9 codes shown in Figure 3-1.

Raw Binary

A conventional way to use ICD-9 codes (and other categorical variables) is to treat each node $n$ as a separate binary feature. For an example $i$, we set the value of the feature corresponding to node $n$ to 1 if the example is associated with the ICD-9 code...
Figure 3-2: Raw Binary feature representation for Patient A and Patient B.

represented by the node. Otherwise, the value is set to 0. Figure 3-2 shows the Raw Binary feature representations for Patient A and Patient B.

This feature representation ignores any information about the hierarchical relationships between the diagnoses. It also suffers from high dimensionality since we use one feature for every distinct value of the variable and is susceptible to overfitting due to sparsity.

Top-level Binary

Top-level Binary uses a separate binary feature to represent hierarchy $T_t$ for each $t$. Given an example $i$, we set the value corresponding to $t$ to 1 if the example is associated with any node in $N_t$. If none of the codes in $N_t$ are associated with to the example, the feature value is set to 0. Figure 3-3 shows the Top-level Binary feature representation for Patient A and Patient B.

The Top-level Binary feature representations for both Patient A and B are identical despite having different list of ICD-9 codes. Regardless of whether a patient is given a diagnosis of 272.0 or 272.1, Top-level Binary sets the feature corresponding
Distinct Top-level ICD-9 codes

<table>
<thead>
<tr>
<th>Patient A</th>
<th>Patient B</th>
</tr>
</thead>
<tbody>
<tr>
<td>249</td>
<td>250</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Patient A</th>
<th>Patient B</th>
</tr>
</thead>
<tbody>
<tr>
<td>249</td>
<td>250</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 3-3: Top-level Binary feature representation for Patient A and Patient B.

to node 272 to 1. As a result, it loses the specific information contained in the distinct ICD-9 codes within the hierarchy tree. Top-level Binary has a low dimensionality and the feature representation is less sparse than for Raw Binary.

Propagated Binary

This representation uses a separate binary feature for each node $n$. If example $i$ is associated with a code represented by node $n$, the feature values corresponding to the node and all of its ancestors are set to 1. Figure 3-4 shows the Propagated Binary feature representation for Patient A.

Propagated Binary encodes hierarchical information by setting all the ancestor nodes of a diagnostic code that is associated with the example to 1. This allows Propagated Binary to tackle the issue of sparsity faced by Raw Binary. Unlike Top-level Binary, despite having a more dense feature representation, Propagated Binary still allows us to operate on both higher level as well more specific lower level granularity. However, it does not reduce feature dimensionality.

43
Probability-based Features

The key idea of our approach is to 1) map each ICD-9 code to a conditional probability of outcome using the training data and 2) then use the probabilities to construct features. More specifically, given a top-level code $t$, we transform each distinct code $n$ within $T_t$ using a function $g : N_t \mapsto [0, 1]$. The function $g(n) = P(y = 1 \mid T_n)$ maps node $n$ to the conditional probability of a positive outcome given $T_n$.

Our approach leverages hierarchy when estimating $P(y = 1 \mid T_n)$ in two ways:

1. In addition to considering the examples that are associated with the code corresponding to $n$, we also consider examples that are associated with any of $n$’s descendant nodes. E.g, for $n = 428.2$, instead of computing $P(y = 1 \mid 428.2)$, we compute $P(y = 1 \mid T_{428.2})$.

2. We consider the probabilities of the more data-rich ancestor nodes for robust probability estimation for nodes lower in the tree using shrinkage, as described later.

We use a two-step process to generate our feature representation (Figure 3-5).
Figure 3-5: Block diagram of our proposed approach for feature construction. $\beta$ is the shrinkage parameter used for probability estimation.

**Step 1: Probability estimation**

To estimate $P(y = 1 \mid T_n)$, we first compute the maximum likelihood (ML) estimate $\hat{P}(y = 1 \mid T_n)$ using the training data.

$$\hat{P}(y = 1 \mid T_n) = \frac{\theta(y = 1, T_n)}{\theta(T_n)} \quad (3.1)$$

where, $\theta(T_n)$ is the number of training examples that are associated with any codes in $\mathcal{N}_n$, and $\theta(y = 1, T_n)$ is the number of training examples with a positive outcome that are associated with any codes in $\mathcal{N}_n$.

Next, we adjust the estimated probabilities by shrinking the ML probability estimates towards the estimate of the parent node.

$$\tilde{P}(y = 1 \mid T_n) = \frac{\theta(T_n)\hat{P}(y = 1 \mid T_n) + \alpha\hat{P}(y = 1 \mid T_n)}{\theta(T_n) + \alpha} \quad (3.2)$$

where $\alpha = \beta \cdot \theta(T_i)$ and $\beta$ is the shrinkage parameter.

Since a top-level node $t$ does not have a parent node, we set $\tilde{P}(y = 1 \mid T_i) = \hat{P}(y = 1 \mid T_i)$.

The shrinkage parameter $\beta$ determines the weighting factor for the ML estimate of $T_n$ and the shrunk estimate of $T_n$ to obtain $\tilde{P}(y = 1 \mid T_n)$. If $\beta$ is 0, the shrunk estimate equals the ML estimate.

The value of $\beta$ is optimized using crossvalidation in the training stage along with the hyper-parameters of the classifier (Section 3.6). For a given $\beta$, we
estimate the probabilities and then use them to construct a feature vector for each example.

**Step 2: Feature Construction**

In this representation, we use a separate feature to represent the hierarchy $T_t$ for each $t$.

For an example $i$ and a top-level code $t$, we first consider nodes in $N_t$ that are associated with the example. Let $N^i_t$ represent this set. An example could be associated with more than one node in $N_t$. For example, within the hierarchy of top-level node 428, an example might be associated with a diagnosis of 428.1 *Left heart failure* and 428.22 *Chronic systolic heart failure*.

Next, we set the feature value of $t$ to $\max(\{\hat{P}(y = 1 \mid T_n) \mid n \in N^i_t\})$. The intuition behind choosing the maximum is to consider the node that puts the example at highest risk. In our preliminary investigations, we also considered other aggregate functions such as *sum* and *average*. However, taking the maximum of the probabilities yielded the best performance. If example $i$ is *not* associated with any nodes in $N_t$, we set the feature value of $t$ to $\hat{P}(y = 1 \mid \text{not } T_t)$, the probability of a positive outcome given that an example is *not* associated with any nodes in $N_t$.

Figure 3-6 shows the *Probability-based* feature representation for Patient A and Patient B (Figure 3-1). Like Top-level Binary, the Probability-based representation also has a single feature per hierarchy tree. However, the Probability-based representation assigns different feature value to the column corresponding to the ICD-9 tree 272. For Patient A, we assign $\hat{P}(y = 1 \mid T_{272,0})$. On the other hand, for Patient B we assign $\hat{P}(y = 1 \mid T_{272,1})$.

### 3.4.2 Model Learning Algorithm

In this section, we discuss two model learning algorithms. We begin by discussing *Logistic Regression with L2 norm* that ignores hierarchy, followed by *Logistic Regres-
Figure 3-6: Probability-based feature representation for Patient A and Patient B.

Logistic Regression with L2 norm

A commonly used learning algorithm for classification tasks is an L2-regularized logistic regression model that solves the following optimization problem:

$$\min_w \sum_{i=1}^{N} \log(1 + \exp(-y_i(w^T x_i + c))) + \lambda \|w\|_2^2$$  \hspace{1cm} (3.3)

While the L2 regularization helps prevent overfitting, it does not use any structural
relationships between features. We refer to this algorithm as L2.

**Logistic Regression with Overlapping Group Lasso**

Given a training data set $D$, the logistic overlapping group lasso [45] solves the following optimization problem:

\[
\min_w \sum_{i=1}^{N} \log(1 + \exp(-y_i(w^T x_i + c))) + \lambda \sum_{j=1}^{k} \|w_{G_j}\|_2
\]  

(3.4)

where $w \in \mathbb{R}^f$ is divided into $k$ overlapping groups $w_{G_1}, w_{G_2}, \ldots, w_{G_k}$.

The hierarchical relationships between the variables are captured by the regularization term. The overlapping group lasso encodes the hierarchical relationship between the variables by defining groups with particular overlapping group patterns in the regularization term. Given a node $n$, representing a variable, a group $G$ is a set containing $n$ and its descendants. E.g., consider the following hierarchical structure between three variables $x_1, x_2$ and $x_3$ shown in Figure 3-7. Let $w_1, w_2$ and $w_3$ be the coefficients associated with $x_1, x_2$ and $x_3$ respectively.

![Figure 3-7: Hierarchical structure between variables $x_1, x_2$ and $x_3$.](image)

We define three groups with overlapping patterns: $G_1 = \{x_1, x_2, x_3\}, G_2 = \{x_2\}$ and $G_3 = \{x_3\}$. That results in the regularization term given by:

\[
\|(w_1, w_2, w_3)\|_2 + \|w_2\|_2 + \|w_3\|_2
\]  

(3.5)

The above formulation ensures that if either $w_2$ or $w_3$ deviates from 0, $w_1$ will
deviate from 0. In other words, if any of the two descendant nodes has a non-zero coefficient, so does the parent. E.g., If we consider the hierarchy for diagnostic code 428 in Figure 2-1, if 428.2 gets a non-zero coefficient, then 428 will also have a non-zero coefficient.

Having $w_2$ and $w_3$ as separate groups in Equation 3.5 allows $w_1$ to be non-zero while either $w_2$ or $w_3$ or both are kept at 0 [45]. E.g. 428 can have a non-zero coefficient while both 428.0 and 428.1 have coefficients equal to 0.

As a result of the properties discussed above, the overlapping group pattern leads to hierarchical selection of variables via regularization [45, 42]. We refer to this algorithm as OvGroup.

3.5 Clinical Tasks

We evaluated the approaches described above on two clinical applications 1) predicting whether a patient at high risk of heart failure will be admitted to the hospital within a year with congestive heart failure (HF) and 2) predicting whether a patient with chronic kidney disease (CKD) will decline quickly or slowly. For each task, we use a patient’s history of diagnoses and procedures, along with the information about a patient’s gender, vitals (e.g. blood pressure) and labs tests (e.g. blood sugar level).

3.5.1 Predicting incident HF

Incident heart failure is defined as the first occurrence of hospitalization that includes a diagnosis of congestive heart failure [25]. Approximately 60% of men and 45% of women die within 5 years of incident HF [24]. Accurate prediction of incident HF can allow clinicians to explore therapy options that can reduce complications associated with HF [17].

We consider patients who will eventually be hospitalized with HF. This allows us to focus on the challenging task of predicting when a patient will be admitted. In particular, we investigate the task of predicting whether a patient will be admitted with incident HF within a year. In the rest of the chapter, we will refer to this task
3.5.2 Predicting CKD progression

Chronic kidney disease is an increasingly prevalent condition associated with a high degree of mortality and morbidity. Clinical decision making for CKD is challenging because of the variability in rates of disease progression among patients [28]. Severity of CKD progression is measured using estimated glomerular filtration rate (eGFR) computed from serum creatinine level. Lower values of eGFR correspond to increased severity.

We consider the task of predicting CKD progression in patients in the early stages of CKD i.e., Stage 1 and Stage 2. We define progression as a sharp drop in their eGFR value in the next two years. Specifically, we assign a positive label for an example if $\Delta_{eGFR} \geq 15$ units, where $\Delta_{eGFR}$ is defined as the difference between the average eGFR in the current 1 year window and the average eGFR in the next two years. The threshold of 15 units was derived by computing the top quintile of $\Delta_{eGFR}$ in the training data. We refer to this task as Task-CKD.

3.6 Experiments and Results

This section describes the experiments used to evaluate different combinations of the feature representation and model learning methods described in Section 3.4. Table 3.1 shows the various combinations we evaluate in this chapter. Top-level Binary and Probability-based features group all distinct nodes within a hierarchy tree into a single feature. Since the resulting representation does not have any group structure to exploit, we only consider L2 for model building.

3.6.1 Experimental Setup

We use data from de-identified electronic health records of patients treated at Mount Sinai Hospital in New York City. We divide the patients into training and test sets
Table 3.1: Definitions of different combinations of feature representation and model learning algorithms. Hierarchy is leveraged in feature construction†, model building‡, or both‡. RawL2 does not use hierarchy.

<table>
<thead>
<tr>
<th>Feature Representation</th>
<th>Learning algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Binary</td>
<td>L2</td>
</tr>
<tr>
<td>Toplevel Binary</td>
<td>OvGroup</td>
</tr>
<tr>
<td>Propagated Binary</td>
<td></td>
</tr>
<tr>
<td>Probability-based</td>
<td></td>
</tr>
<tr>
<td>RawL2</td>
<td>RawL2</td>
</tr>
<tr>
<td>TopL2 †</td>
<td>TopL2 †</td>
</tr>
<tr>
<td>PbL2 †</td>
<td>PbL2 †</td>
</tr>
<tr>
<td>ProbL2 †</td>
<td>ProbL2 †</td>
</tr>
</tbody>
</table>

with an 80/20 split. For each patient, we use a 1 year sliding window approach to generate multiple examples (Figure 3-8). Table 3.2 shows the summary of the data used for each task.

Figure 3-8: A schematic summarizing how multiple examples are generated per patient. Each rectangle represents an example corresponding to a time window of size \( T \). We use \( T = 1 \) year for our experiments.

For Task-CKD we only consider examples that have 1) at least two outpatient eGFR measurements within the example time period and 2) at least two outpatient eGFR measurements each year for two years in the future. Since patients’ eGFR measurements are not measured at regular intervals, we are left with 974 examples that satisfy the criteria. As a result, Task-CKD has fewer examples than Task-HF, despite having more patients.

We evaluated the performance of each of the methods in Table 3.1 for Task-HF and Task-CKD. For each task, we generate 100 different training and test splits. For each split the training set is used for parameter selection and model training. All the methods, besides ProbL2, have one parameter, i.e., the regularization parameter \( \lambda \). ProbL2 has an additional parameter \( \beta \), the shrinkage parameter. Parameter selection
Table 3.2: The summary of data used for Task-HF and Task-CKD. The value in the parentheses is the number of positive examples.

<table>
<thead>
<tr>
<th></th>
<th>Task-HF</th>
<th>Task-CKD</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of patients</td>
<td>477</td>
<td>510</td>
</tr>
<tr>
<td>No. of Examples</td>
<td>1649 (326)</td>
<td>974 (195)</td>
</tr>
<tr>
<td>No. of Training Examples</td>
<td>1320 (261)</td>
<td>780 (156)</td>
</tr>
<tr>
<td>No. of Test Examples</td>
<td>329 (65)</td>
<td>194 (39)</td>
</tr>
</tbody>
</table>

is carried out using 5-fold cross-validation. The test set is used for evaluation only.

### 3.6.2 Performance Evaluation

To evaluate the ability of the different methods to discriminate between examples with a positive outcome and those without a positive outcome, we compute

- F-score: the harmonic mean of recall and precision \[31\]

- AUROC: a rank order statistic based on predicted probability of outcome \[35\]

To compute the recall and precision, we consider the examples with predicted probability of outcome in the top quintile as positive.

To further convey the ability of the methods to risk stratify, we also compute the relative risk of outcome as another measure of evaluation. We compute the relative risk as follows:

1. We divide the test examples into quintiles based on the predicted probability of outcome. The examples in the 1\textsuperscript{st} quintile (Q1) are predicted to be at lowest risk and the ones in the 5\textsuperscript{th} quintile (Q5) are predicted to be at highest risk of adverse outcome.

2. The relative risk of outcome in Q5 versus Q1 is computed as follows:

   \[
   \text{Relative Risk} = \frac{P(y = 1 \text{ in Q5})}{P(y = 1 \text{ in Q1})} \quad (3.6)
   \]
We compute the relative risk of outcome in Q5 versus Q1 as another measure of evaluation.

We evaluate the performance of all other methods with respect to RawL2, since it does not use any structural information in the ICD-9 codes. Table 3.3 and Figure 3.9 show the results.

Table 3.3: Average performance of the different methods for Task-HF and Task-CKD. The value within the parentheses is the standard deviation. The * indicates that the average is significantly \( (p < 0.05) \) different from the average performance of RawL2 as measured using a paired t-test. The methods that perform the best are highlighted in gray.

<table>
<thead>
<tr>
<th>Approach</th>
<th># of features</th>
<th>F-score</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>RawL2</td>
<td>3697</td>
<td>0.36 (0.04)</td>
<td>0.65 (0.03)</td>
</tr>
<tr>
<td>TopL2</td>
<td>652</td>
<td>0.42* (0.04)</td>
<td>0.68* (0.03)</td>
</tr>
<tr>
<td>PbL2</td>
<td>3697</td>
<td>0.40* (0.05)</td>
<td>0.69* (0.04)</td>
</tr>
<tr>
<td>ProbL2</td>
<td>652</td>
<td>0.45* (0.03)</td>
<td>0.71* (0.03)</td>
</tr>
<tr>
<td>RawOvGroup</td>
<td>3697</td>
<td>0.40* (0.05)</td>
<td>0.67 (0.04)</td>
</tr>
<tr>
<td>PbOvGroup</td>
<td>3697</td>
<td>0.44* (0.04)</td>
<td>0.70* (0.03)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Approach</th>
<th># of features</th>
<th>F-score</th>
<th>AUROC</th>
</tr>
</thead>
<tbody>
<tr>
<td>RawL2</td>
<td>3200</td>
<td>0.37 (0.05)</td>
<td>0.66 (0.04)</td>
</tr>
<tr>
<td>TopL2</td>
<td>651</td>
<td>0.38 (0.04)</td>
<td>0.66 (0.03)</td>
</tr>
<tr>
<td>PbL2</td>
<td>3200</td>
<td>0.34* (0.06)</td>
<td>0.64 (0.04)</td>
</tr>
<tr>
<td>ProbL2</td>
<td>651</td>
<td>0.38 (0.05)</td>
<td>0.67 (0.03)</td>
</tr>
<tr>
<td>RawOvGroup</td>
<td>3200</td>
<td>0.38 (0.05)</td>
<td>0.66 (0.03)</td>
</tr>
<tr>
<td>PbOvGroup</td>
<td>3200</td>
<td>0.40* (0.06)</td>
<td>0.69* (0.03)</td>
</tr>
</tbody>
</table>

Our results show that overall the methods that take into account the hierarchical structure boost the performance of the predictive model for both Task-HF and Task-CKD, although the amount of performance improvement varies between the different methods. For Task-HF, ProbL2 and PbOvGroup perform the best, significantly outperforming RawL2. For Task-CKD, PbOvGroup yields the highest improvement over RawL2. Although ProbL2 performs comparably to RawL2 in terms of F-score and AUROC, ProbL2 has a much higher relative risk.
Although the performance improvement in F-score and AUROC obtained by leveraging hierarchy appears small in magnitude, it has the potential to have a real clinical impact. For Task-HF, our results show that by leveraging hierarchy, we can correctly identify $\approx 60\%$ of the examples where the patients suffer an adverse outcome compared to $\approx 38\%$ (for a precision of $\approx 35\%$) when disregarding the structural relationships. For Task-CKD, by leveraging hierarchy we were able to correctly identify $\approx 60\%$ of the examples with adverse outcomes compared to $\approx 46\%$ (for a precision of $\approx 32\%$) when ignoring hierarchy. It is unclear from our results whether ProbL2 or PbOvGroup performs better than the other. However, the advantage of exploiting the ICD-9 hierarchy for the two tasks is clear.

Our results also demonstrate the importance of coupling an appropriate feature representation with an appropriate learning algorithm. RawOvGroup and PbOvGroup use the same learning algorithm but different feature representations. On the other hand, PbL2 and PbOvGroup use different learning algorithms but the same feature representation. PbOvGroup outperforms RawOvGroup and PbL2 for both Task-CKD and Task-HF showing that leveraging ICD-9 hierarchy in both feature representation and learning algorithm yields the best results.

### 3.7 Separating the effect of feature representation and feature dimensionality

Our experimental results show that our proposed feature representation Prob that leverages hierarchy outperform Raw Binary, the baseline approach that ignores hierarchy. However, Prob differs from Raw Binary in two ways:

1. Prob has a lower feature dimensionality than Raw Binary.

2. Prob uses a probability based representation as opposed to a binary representation.

In order to evaluate the contribution of our novel feature coding/representation, in this section, we compare the performance of the models using Raw Binary and
Figure 3-9: Relative risk of outcome in examples in the 5th quintile vs the 1st quintile.

\[ \text{Prob} \] such that the feature dimensionality between the two is comparable. More specifically, we use logistic regression with L1 regularization, given by:

\[
\min_w \sum_{i=1}^{N} \log(1 + e^{-y_i(w^T x_i + c)}) + \lambda_1 \|w\|_1
\]  

(3.7)

Logistic regression with L1 regularization leads to feature selection by yielding sparse estimates of the feature coefficients. It yields a sparse estimates by assigning zero coefficients as feature weights. The number of features that are assigned a zero weight depends on the value of \( \lambda_1 \). More specifically, the larger the value of \( \lambda_1 \), the higher is the number of features with zero weights.
We refer to this algorithm as $L1$.

In Figure 3-10, RawL1 indicates the performance of the Raw Binary representation for different number of features (obtained by varying the value of $\lambda_1$) for Task-HF. The green vertical line corresponds to the number of features in Prob before feature selection ($\approx 650$ features). For the same number of features, the performance of ProbL2 is significantly higher than the performance of RawL1.

![Graph showing AUROC for RawL1, ProbL2, and RawL2](image)

Figure 3-10: The average AUROC of RawL1 for different number of features with non-zero weights across the 100 experiments along with the standard error. The red and the green dot corresponds to the performance of ProbL2 and RawL2 respectively. The dotted lines indicate the corresponding number of features with non-zero weights.

However, our results show that as the number of features with non-zero weight decreases (as $\lambda_1$ increases), the performance of RawL1 increases to a peak and then eventually decreases. The peak performance for RawL1 is significantly higher than that of RawL2. This shows that Raw Binary representation is vulnerable to overfitting unless strong regularization that yields a sparse model is imposed.

Figure 3-10 also shows the performance of ProbL2. While using L1 significantly improved the performance for Raw Binary, its best performance is still consistently lower than the performance of ProbL2 across the 100 different splits of data into training and test sets.
In summary, our results show that our proposed feature representation \( \text{Prob} \) is significantly robust against overfitting than \text{Raw Binary}, yielding a significantly higher performance when the number of features are comparable. Our results also demonstrate that while the performance \text{Raw Binary} improves when learning sparse models, \text{Prob} still outperforms the approach by a statistically significant margin (p-value < \(10^{-5}\)).

3.8 Summary and Conclusions

We presented different methods to leverage the structural relationships in ICD-9 codes, an important and readily available source of patient information stored in the EHRs. We compare methods that leverage hierarchy by 1) incorporating the structural information during feature construction, 2) using a learning algorithm that considers the structure in features when learning a model, and 3) doing both. We present a novel probability-based feature construction approach that is simple and effective in exploiting the structural relationships between the ICD-9 codes to overcome the problem of overfitting when the data is sparse. We investigated these different methods in the context of two clinical applications with the goal to improve predictive performance of risk stratification models.

Our results demonstrate that methods that properly leverage ICD-9 hierarchy can significantly improve performance over a method (\text{RawL2}) that does not use hierarchy either for feature construction or model learning. The underperformance of \text{RawL2} can be attributed to 1) overfitting due to high feature dimensionality and 2) its inability to leverage relationships between ICD-9 codes within a hierarchy. \text{TopL2}, which aggregates all the codes within a hierarchy tree to a single binary feature, yields some performance boost over \text{RawL2} by using associations between codes within a hierarchy tree via single binary feature to reduce feature dimensionality. However, as discussed earlier, \text{TopL2} does not take advantage of the specificity in lower level ICD-9 codes.

Our results also show that the performance of \text{RawL1} is significantly higher than
that of RawL2, suggesting that the performance of Raw Binary improves when learning sparse models. However, the performance of ProbL2 that use our proposed representation still outperforms RawL1 by a statistically significant margin.

We made two other observations from our results:

1. The magnitude of improvement varied between the two tasks, with Task-HF yielding a larger improvement than Task-CKD.

2. Although statistically significant, the absolute magnitude of improvement was small for both tasks.

The amount of improvement we expect to achieve by exploiting hierarchical relationships between ICD-9 codes depends on various properties of the data. In the next chapter, we perform a simulation study to better understand what those properties are and how that affects the magnitude of improvement.
Chapter 4

Exploring the Effect of Leveraging Hierarchy via Simulation Study

In the previous chapter, we showed that leveraging hierarchy in the ICD-9 codes can lead to predictive models with improved risk stratification performance. However, the amount of improvement was relatively small and also varied between the two clinical tasks we considered. The amount of improvement depends on many factors. In this chapter, we focus on better understanding the properties of the data that affect the amount of improvement that one can expect by leveraging hierarchical relationships between the variables.

4.1 Generating Simulation Data

For our simulation, we generate data that is a simple approximation of real clinical datasets. ICD-9 codes recorded in EHR are structurally related to one another according to a hierarchy tree. Another important property of the diagnoses and procedures recorded using ICD-9 codes in EHR is that patients are not always assigned the most specific codes (i.e., the leaf nodes in the hierarchy tree). Instead, one of the higher level codes are recorded. E.g., The ICD-9 code recorded for a patient may be 428 Heart failure, when in fact the underlying condition the patient is 428.22 Chronic systolic heart failure. This lack of specificity could be 1) because the care
providers were unable to identify the true underlying condition at the time the data was recorded, or 2) because of the lack of meticulous recording of patient information in the EHR. Therefore, the information stored in EHR is imprecise.

In our simulation dataset, all variables are nodes in a single complete binary tree $T$. Each node corresponds to a variable. The number of variables $V$ is determined by the number of levels $L$ in $T$, where $V = 2^L - 1$. Figure 4-1 shows a complete binary tree with $L = 4$. As in the case of ICD-9 codes, we assume that a node at a higher level of the tree represents a more general/abstract concept than the nodes at a lower level.

Figure 4-1: A complete binary tree with $L = 4$. There are $V = 2^4 - 1 = 15$ nodes in the tree. The number on the node represent the value of $v$ that denotes the node. For a complete binary tree with $L$ levels, the indices of the leaf nodes ranges from 1 to $2^{L-1}$.

We use a bottom-up approach to generate examples for our simulation dataset. We begin by generating the underlying data. Each underlying-datum is a tuple of a leaf node $l$ in hierarchy tree $T$ and the label $y$, i.e., underlying-datum $u = (l, y)$. The label $y$ is a binary variable. Therefore, underlying dataset $U = \{u_i = (l_i, y_i) | l_i \in \{\text{leaf nodes in } T\}, y_i \in \{0, 1\}\}_{i=1}^N$, where $N =$ number of data points. The label $y_i$ is a function of the leaf node $l_i$. Section 4.1.1 describes the process of generating the underlying data in more detail.

Next, we mimic the imprecise information seen in the ICD-9 codes in the EHR
data. For each underlying-datum, we generate a corresponding example, where \( e = (n, y) \) and \( n \) is a node such that \( n \in \{ l, \text{ancestors of } l \text{ in } T \} \). This simulates the scenario in which the underlying leaf node is not always observed. Instead one of the ancestors of the underlying leaf node is observed. We represent the set of examples \( E = \{ e_i = (n_i, y_i) | n_i \in \{ l_i, \text{ancestors of } l_i \text{ in } T \}, y_i \in \{ 0, 1 \} \}^N_{i=1} \), where \( N = \text{number of data points} = \text{number of examples} \). Section 4.1.2 describes the process of generating the examples in more detail.

Figure 4-4 provides an overview of the steps in generating our simulated data.

![Diagram of data generation process](image.png)

**Figure 4-2:** A schematic illustration of the overview of the steps used in generating the simulation data. The user-specified inputs are discussed in greater detail in Section 4.1.1 and 4.1.2.

### 4.1.1 Generating the underlying data

We generate the underlying data according to the following parameters:

1. Number of levels \( L \) in the tree \( T \)

2. Number of datum that are assigned per leaf node \( N_{\text{leaf}} \).

3. Probability of a positive label given each leaf node, i.e., \( P = [p_1, p_2, \ldots, p_K] \), where \( K = 2^{L-1} \) is the number of leaf nodes in \( T \). This parameter is used to generate the labels associated with each datum.

The number of levels \( L \) in the tree \( T \) determines the number of nodes, and consequently the number of leaf nodes, in \( T \). The parameter \( N_{\text{leaf}} \) determines the number
of underlying-datum that are associated with each of the leaf nodes in $T$. In our
simulation, each leaf node has $N_{\text{leaf}}$ underlying-datum. Therefore, the total number
of datum $N = \sum_{\text{leaf}} N_{\text{leaf}}$. Using these parameters we generate the leaf nodes in the
underlying dataset $U$.

Once we have the leaf nodes in $U$, the next step is to generate the labels associated
with each datum. Given an underlying-datum $u$ associated with leaf node $l$, we
generate the label $y$ using the parameter $P$. Specifically, the label $y$ is generated
using the a Bernoulli distribution where the probability of success is the probability
of a positive label given the leaf node $l$, i.e., $p_l$.

For our simulations, we use Algorithm 1 to generate the vector $P$. Along with
the number of levels in $T$, the algorithm also takes another user-specified input $\alpha$,
a measure of how different the probabilities of two leaf node siblings are in $P$. If
$\alpha = 0$, the siblings have identical probabilities. For $\alpha \geq 2$, the probabilities are
independently sampled from a uniform distribution between 0 and 1.

Algorithm 1 generates $P$ in such a way that the degree of similarity between
probabilities of two leaf nodes is a linear function of the familial distance between the
two nodes. For a leaf node $k$ the following are true:

1. Given the probability of its sibling, $p_k$ is at most $\frac{\alpha}{2}$ apart from it.

2. Given the probabilities of its first cousins, $p_k$ is at most $2 \times \frac{\alpha}{2}$ apart from the
average probability of its first cousins.

3. Given the probabilities of its second cousins, $p_k$ is at most $3 \times \frac{\alpha}{2}$ apart from the
average probability of its second cousins, and so on.

Figure 4-3 shows the average absolute difference between probabilities of all pairs
of leaf node siblings in $T$ with $L = 4$ for different values of $\alpha$. Given $\alpha$, we averaged
the values across 100 independent iterations of Algorithm 1. As $\alpha$ increases, the aver-
age absolute difference in probabilities between siblings also increases and eventually
plateaus. For $\alpha \geq 2$, the probabilities of the siblings are sampled independently.
Therefore, the average absolute difference between probabilities of the siblings should
Algorithm 1 Generate the probability of outcome for each leaf node

**Input:** Number of level $L$ in tree $T$, $\alpha$

**Output:** $P = [p_1, p_2, ..., p_K]$

1. number of leaf nodes $K = 2^{L-1}$
2. $p_1 =$ sample uniformly between 0 to 1.
3. **for** $i = 2$ to $K$ **do**
   4. $multiplier =$ distance to the common ancestor of node $i$ and node $i - 1$ {where, distance between a node and its ancestor = difference between their levels}
   5. $C =$ set of leaf nodes to the left of the common ancestor
   6. $anchor =$ mean\(\{p_c | c \in C\}\)
   7. $a =$ max\(0, anchor \times \frac{a}{2}\)
   8. $b =$ min\(1, anchor + multiplier \times \frac{a}{2}\)
   9. $p_i =$ sample uniformly between $a$ and $b$
   10. **end for**

equal the expected value of $Z = |C - D|$ where $C$ and $D$ are two independent random variables with standard uniform distribution. It is known that $Z$ has a triangular distribution with an expected value of $\frac{1}{3}$ [8]. Therefore, it is not surprising that the average absolute difference plateaus around $\frac{1}{3}$ in Figure 4-3.

Figure 4-3: The average difference between probabilities of leaf node siblings in $T$ with 4 levels for different values of $\alpha$.

Algorithm 1 can generate probabilities such that the degree of similarity between the probabilities of two leaf nodes is smaller than the specified $\alpha$ when one of the
following is true:

1. 

\[(\text{anchor} - \text{multiplier} \times \frac{\alpha}{2}) < 0 \text{ and } 0 < (\text{anchor} + \text{multiplier} \times \frac{\alpha}{2}) \leq 1.\]

2. 

\[0 \leq (\text{anchor} - \text{multiplier} \times \frac{\alpha}{2}) < 1 \text{ and } (\text{anchor} + \text{multiplier} \times \frac{\alpha}{2}) > 1.\]

The happens because we truncate \(a\) and \(b\) in Algorithm 1 such that \(a = \max(0, \text{anchor} - \text{multiplier}\times\alpha)\) and \(b = \min(1, \text{anchor} + \text{multiplier}\times\alpha)\). The likelihood of truncation increases for large values of \(\alpha\) (For \(\alpha > 2\), the likelihood of truncation is 1 with \(a = 0\) and \(b = 1\)). As a consequence, the actual degree of similarity can be smaller than the specified \(\alpha\).

However, this does not pose a serious problem for our study. A goal of our study is to understand the trend of the amount of improvement that can be yielded by leveraging hierarchy for different sizes of \(\alpha\) (Section 4.8). The precise value of \(\alpha\) is not critical for our analysis.

### 4.1.2 Generating the examples

We generate the examples from the underlying data according to another user-specified parameter: the observed level probability matrix. We represent the observed level probability matrix by \(O \in \mathbb{R}^{L \times K}\), where \(O_{\text{level}, k}\) represents the probability that for an underlying-datum associated with leaf node \(k\), the corresponding example will have a node at the specified level. E.g., Given \(L = 3\) (number of leaf nodes \(K = 2^{3-1} = 4\)), consider the following observed level probability matrix:

\[
O = \begin{pmatrix}
0 & 0 & 0.25 & 0 \\
1 & 0.60 & 0.65 & 0 \\
0 & 0.40 & 0.10 & 1
\end{pmatrix},
\]

Since a column \(k\) of \(O\) represents the probability distribution across the levels, \(\sum_{\text{level}} O_{\text{level}, k} = 1\). For an underlying-datum \(u = (l = 1, y)\), the corresponding example will have an observed node in \(\text{level} = 2\) with 100% probability. If \(u = (l = 2, y)\), then the corresponding example will have an observed node in \(\text{level} = 2\)
with 60% probability and in \( \text{level} = 3 \) with 40% probability. If \( u = (l, y) \), then the corresponding example can have an observed node in either of the three levels with a probability of 25%, 65% and 10% respectively. Finally, for \( u = (l = 4, y) \), the observed node of the corresponding example is always the leaf node.

Given an underlying data set \( U \) and an observed level probability matrix \( O \), we generate the set of examples \( E = \{(n_i, y_i) | n_i \in \{l_i, \text{ancestors of } l_i \text{ in } T\}, y_i \in \{0, 1\}\}^N \).

After generating the set of examples \( E \), one can compute the observed mass distribution across the nodes represented by \( M \in \mathbb{R}^{1 \times V} \) where \( M(v) \) represents the fraction of examples for which \( n_i = v \).

\[
M(v) = \frac{\sum_{i=1}^{N} \mathbb{1}(n_i = v)}{N} \tag{4.1}
\]

where \( \mathbb{1} \) is the indicator variable, \( M(v) \in [0, 1] \) and \( \sum_v M(v) = 1 \).

For a fixed \( U \), there is a one-to-one mapping between the observed level probability matrix \( O \) and observed mass distribution across the nodes represented by \( M \).

---

**4.2 Road-map of Experiments**

The goal of our study is to analyze how the different properties of the data affect the performance boost one can achieve by leveraging hierarchy. We begin by generating a
training and a test data set with specified properties. Next, we train a model on a set of training examples using 1) RawL2, a method that ignores hierarchy and 2) ProbL2, a method that leverages hierarchy (Both methods are described in Chapter 3). Then, we evaluate the performance of the models on the test examples.

We generate the training set and the test set separately using the steps illustrated in Figure 4-4. The user-specified parameters for the training and the test data sets can be different, allowing the two datasets to have different properties. In clinical medicine, predictive models are learned using historical data and are applied to new patients. Since, the properties of the historical data might be different from the properties of the present-day patient data, using different parameters enables us to simulate the real world scenario. We discuss this in more detail in the following paragraphs. We distinguish between the parameters, properties and examples in the training and the test set using the superscript $t_r$ and $t_{st}$ respectively.

Table 4.1 shows the list of parameters and whether or not we allow them to be different between the training and the test sets in our various experiments. The parameters that control the underlying behavior of the data, i.e. the number of levels $L$ in the tree and the probability of a positive label given each leaf node $P$ are kept the same for the training and the testing data. This is reasonable since the assumption that the training and the test data have the same underlying latent properties is true in most applications. In the context of medicine, we expect the underlying relationship between the patient’s state and the outcome to be the same in historical and present-day data.

We allow the parameters that are likely to be different in the training and the test data obtained from clinical datasets to differ in our simulation. Since predictive models are trained on historical data, the size of the training set depends on the size of the available historical data. The number of patients on which the predictive models are applied to can be significantly larger than the number of patients on which the model was trained. Similarly, the observed mass distribution in the training data $M_{t_r}$ can be different from the distribution in the testing data $M_{t_{st}}$. For example, in the context of ICD-9 codes, it is possible that the data on which the model is trained,
Table 4.1: The user-specified parameters required to generate the training and the test data sets. The parameters that control the underlying behavior of the data, i.e., the number of levels $L$ in the tree and the probability of a positive label given each leaf node $P$ are kept the same for the training and the testing data. The other two parameters are allowed to vary between the training and the test dataset.

In our experiments, we analyze the following:

1. We evaluate the effect of the number of training examples on the performance when the mass distributions of the training and the test examples are the same. To achieve this, we learn models with different values of $N_{leaf}^{tr}$ (Section 4.5).

2. We investigate how the difference in observed mass distribution across levels between training and testing data affects performance when we do not leverage hierarchy versus when we do (Section 4.6).

3. We explore the effect of leveraging hierarchy when we vary $\alpha$, a measure of how different the probabilities of two leaf node siblings are in $P$ (Section 4.8).
4.3 Experimental Setup

In all our experiments, during the model training phase, we select the hyper parameters of the models ($\lambda$ for RawL2, and $\lambda$ and $\beta$ for ProbL2) using 5-fold cross-validation on the training set. Next, we evaluate the performance of each trained model $\mathcal{M}$ on a separate test data. For each example $e_i^{tst}$ in the testing set, we use $\mathcal{M}$ to get a predicted probability of outcome $\hat{y}_i^{tst}$. We use the predicted probabilities $\{\hat{y}_i^{tst}\}_{i=1}^{N^{tst}}$ and $\{y_i^{tst}\}_{i=1}^{N^{tst}}$ to compute the AUROC and the F-score.

For each experiment, we generate 100 different pairs of training and test sets. All the results that we report are averaged across the 100 different test sets. When generating data, we use the number of levels $L = 4$. We set the number of examples per leaf node in the testing set $N_{\text{leaf}}^{tst}$ to 1000. Therefore, the total number of examples in the testing data is always $N_{\text{tst}}^{tst} = 1000 \times 2^{L-1} = 1000 \times 2^{4-1} = 8000$. Since we analyze the effect of the number of training examples (Section 4.5), we vary $N_{\text{leaf}}^{tr}$ in our experiments.

The probability of a positive label given each leaf node $P$, depends on $\alpha$. For our experiments in Section 4.5 through Section 4.6 we set $\alpha = 0.25$, which corresponds to a value of $\approx 0.05$ for the average absolute difference between probabilities of leaf node siblings. We discuss the properties of this choice of $\alpha$ in Section 4.8, in which we report on experiments using different values of $\alpha$.

4.4 Performance of the Oracle

In addition to evaluating the performance of RawL2 and ProbL2 learned from the training data, we also compute the performance of the oracle on the test data. The oracle is the maximum likelihood classifier built using the test data. In the rest of the chapter, we refer to the performance of the resulting classifier on the test set as the Oracle performance. We compute the Oracle performance using Algorithm 2. By definition, the Oracle performance does not depend on the properties of the training data.
Algorithm 2 Compute the Oracle performance on the test set

Input: Test examples $E^{test}$
Output: AUROC$_{Oracle}$, F-score$_{Oracle}$

1: $X^{test}_{PB}$ = Propagated Binary representation of $E^{test}$ {See Chapter 3 for Propagated Binary}
2: for $v = 1$ to $V$
3: \[ \hat{\theta}_v = \text{Maximum likelihood estimate of } p(y = 1|\text{node } v) \]
4: end for
5: for $i = 1$ to $N^{test}$
6: \[ \hat{y}_i^{test} = \hat{\theta}_{v_i^{obs}} \]
7: end for
8: Compute AUROC$_{Oracle}$ and F-score$_{Oracle}$ using $\{\hat{y}_i\}_{i=1}^{N^{test}}$ and $\{y_i\}_{i=1}^{N^{test}}$

The difference between the performance of Oracle and RawL2 tells us whether an improvement over RawL2 is possible by leveraging hierarchy. E.g., If the difference is 0, then there is no available room for improvement. A large difference does not guarantee a large improvement by leveraging hierarchy. In other words, a gap between the performance of Oracle and RawL2 is a necessary but not a sufficient condition to get an improvement in performance by leveraging hierarchy.

If we assume that there exists room for improvement over RawL2, the amount of improvement attainable by leveraging hierarchy depends on the properties of the training and the test data. We investigate these different properties in Sections 4.5 through 4.8.

4.5 Effect of the number of training examples and mass distribution

In this section, we evaluate the effect of the number of training examples on the performance of RawL2 and ProbL2 when the observed mass distribution is the same for the training and the testing set. We use the following setup for the experiments:

Fix: $M = M^{tr} = M^{test}$ and $\alpha = 0.25$

Vary: $M$ and $N^{tr}_{leaf}$
Figure 4-5 shows the heat map representation of the different mass distributions we consider in our analysis, along with \( O \), the corresponding observed level probability distribution. The color of node \( v \) in the heat map represents the fraction of examples for which \( n_i = v \), normalized by the maximum fraction across all nodes. We normalized the fraction so that the spread of examples across different nodes is clearly visible.

For \( M_a \), all the nodes are equally observed in the examples, and therefore have the color associated with 1 (as shown in the color bar). For \( M_b \), all the nodes beside the leaf nodes are equally observed. Lastly, for \( M_c \) only the leaf nodes are observed and each leaf node is equally likely to be observed.

![Heatmaps of the different mass distributions considered in Section 4.5 in which \( M = M^{tr} = M^{st} \). The corresponding observed level probability distributions are also shown.](image)

Figure 4-5: Heatmaps of the different mass distributions considered in Section 4.5 in which \( M = M^{tr} = M^{st} \). The corresponding observed level probability distributions are also shown.

Figure 4-6 shows the performance of the Oracle along with the performance of RawL2 and ProbL2 for different number of training examples and mass distributions.
As discussed in Section 4.4, the Oracle performance does not change with the number of training examples. However, it varies between the different mass distributions. \( M_c \) has the highest Oracle performance among the three mass distributions considered. This is because when generating the simulated data we assume that the label \( y \) depends on the underlying leaf node associated with the example. Since the leaf nodes are observed for all the examples, the Oracle performance is the highest. For \( M_a \) and \( M_b \), since nodes other than the leaf nodes are observed for some of the examples, the Oracle performance is lower than that of \( M_c \). Since \( M_b \) has no observed leaf node, \( M_b \) has the worst Oracle performance.

Figure 4-6 shows that the performance of both RawL2 and ProbL2 increase with increasing number of training examples. This is not surprising since a larger number of training examples is known to increase generalizability of the trained models when the training and testing data set comes from the same distribution. More importantly, Figure 4-6 shows that ProbL2 either performs comparably or outperforms RawL2. The amount of improvement yielded by leveraging hierarchy varies with both \( N_{\text{tr}}^{\text{leaf}} \) and the mass distributions. Figure 4-7 show the percentage improvement in performance obtained by leveraging hierarchy over RawL2. For each mass distribution, as the number of training examples increases, the amount of improvement converges to 0. When the number of training examples is large enough, each node by itself has enough number of examples for the model to accurately learn its effect on \( y \). Therefore, the performance of RawL2 converges to the Oracle performance.

Figure 4-7 shows that, given a fixed \( N_{\text{tr}}^{\text{leaf}} \), the effect of leveraging hierarchy depends on the mass distribution. ProbL2 yields the highest improvement for \( M_a \) and the smallest improvement for \( M_c \). The number of observed nodes for \( M_a \) is much larger than the number of observed nodes for \( M_c \). Therefore, given a fixed \( N_{\text{tr}}^{\text{leaf}} \), each observed node is more sparsely represented for \( M_a \) than for \( M_c \). As a result, leveraging hierarchy can yield a greater boost in performance.

\( M_c \) corresponds to the mass distribution for which precise information is available for all training and test examples since the observed nodes are the underlying leaf nodes. Our results show that even when precise information available, leveraging
Figure 4-6: The average performance of RawL2 and ProbL2 along with the best possible performance for the different $M = M^{tr} = M^{st}$ (a) $M_a$ (b) $M_b$ and (c) $M_c$ (shown in Figure 4-5).
Figure 4-7: The average percentage improvement in performance obtained by leveraging hierarchy for different $N_{\text{leaf}}^{\text{tr}}$ and different mass distributions.

hierarchy can help when we have few training examples.

In summary, based on these simulations we conclude that, all else being equal, as the number of training examples increases the amount of improvement obtained by leveraging hierarchy over the RawL2 decreases. However, when the number of training examples is small, leveraging hierarchy can still yield performance improvement even when precise information is available.

However, based on our results one should not conclude that for any data set with $M^{tr} = M^{\text{ist}} = M_c$, the performance of RawL2 and ProbL2 will be comparable if the training data set has $N^{tr} = 2048$ examples ($= N_{\text{leaf}}^{tr} \times K = 2^8 \times 8 = 2048$). Real datasets usually have many more nodes in their hierarchy than the simulation dataset we used for our experiments. Therefore, $\approx 2000$ training examples is most likely to be a significant underestimate of the number of examples needed in practice for the performance of RawL2 and ProbL2 to converge when $M^{tr} = M^{\text{ist}} = M_c$. 

73
4.6 Effect of differing mass distributions between training and test sets

Next, we evaluate the effect of leveraging hierarchy when the mass distribution between the training and the test sets are different. We use the following setup for the experiments:

\[ \begin{align*}
\text{Fix :} & \quad M_{\text{test}}^t = M_a, \ N_{\text{leaf}}^{tr} = 2^8 \text{ and } \alpha = 0.25 \\
\text{Vary :} & \quad M^{tr} 
\end{align*} \]

Since \( M_{\text{test}}^t = M_a \), each node in \( T \) is an observed node for an equal number of examples in the testing data.

In our experiments, we consider \( M^{tr} \) such that some of the nodes in \( T \) are not represented in the training examples. We refer to these as unseen nodes/variables. This can occur in clinical datasets when the number of variables is much larger than the number of training examples. In the context of the ICD-9 codes, the data on which the model is trained might not contain patients associated with all of the ICD-9 codes that are seen in the test data. The issue of unseen nodes will become particularly important when hospitals transition from ICD-9 codes to ICD-10 codes. ICD-10 codes are the most recent revision of the International Statistical Classification of Diseases and Related Health Problems (ICD) that contain more than 69,000 distinct codes (compare to 17,000 in ICD-9) to provide greater details and specificity [5]. Since ICD-10 contains significantly many codes than ICD-9, we expect the issue of unseen nodes to be more prevalent in medical datasets. We will discuss this in more detail in Section 4.9.

Since \( \text{RawL2} \) ignores hierarchical relationships between the variables and treats each variable independently, it is unable to infer the effect of the unseen variable on the label. Therefore, for examples in the testing data for which the observed node is an unseen node, it simply assigns the overall prevalence as the predicted likelihood of a positive label. However, by using the structural relationships between the variables that are observed in the training data and the unseen variable, \( \text{ProbL2} \) can potentially make inferences about its effect on the label.
Because we set $M^{tst} = M_a$, all nodes are observed in the testing data. Figure 4-8 show the different mass distributions for the training data considered in our analysis. For mass distributions $M_d$ through $M_f$, the level (and the number) of the unseen nodes in the training data increases.

Figure 4-8: The different observed mass distributions for the training data set ($M^{tr}$) used in Section 4.6 to evaluate the effect the leveraging hierarchy when $M^{tr}$ and $M^{tst}$ are different. For mass distributions $M_d$ through $M_f$, the level (and the number) of the unseen nodes in the training data increases.

Figure 4-9 shows the performance of the different methods on the test set.
Figure 4-9: The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data shown in Figure 4-8.

As the level of the unseen nodes in the training data increases, the performance of RawL2 drops significantly. This is because as the level increases, the total number of unseen nodes and consequently the number of testing examples for which the observed nodes are unseen in the training data also increases. E.g., Consider $M_{tr} = M_d$ where 2 nodes are unseen. With $N_{tst} = 8000$ (see Section 5.5.1) and $M_{tst} = M_a$, the observed nodes are missing in the training data for $\approx 1070$ of the test examples ($\approx \frac{8000 \times 2}{15}$, where $15 = \text{number of nodes}$). On the other hand, when $M_{tr} = M_f$, 8 nodes are unseen. Hence, for $\approx 4770$ test examples ($\approx \frac{8000 \times 8}{15}$), the observed nodes are missing in the training data. Since RawL2 is unable to make any inferences for a significantly larger fraction of testing examples, its performance drops significantly. In contrast, despite a large number of unseen nodes, ProbL2 is able to largely maintain its performance by leveraging hierarchy.

In the above experiments, along with the level of unseen nodes, the number of unseen node also increases. Next, given a fixed number of unseen nodes, we evaluate whether their level in the hierarchy tree affects the improvement yielded by leveraging hierarchy. We fix the number of unseen nodes to 2 and 4. Table 4.2 gives the description of the different mass distributions of the training set.
In Table 4.2, when the number of unseen nodes in a given level is less than the number of nodes in that level, we denote its mass distribution as $\tilde{M}_s$. This is because for each of the 100 runs, we randomly choose a given number of nodes in the specified level as missing. Therefore, unlike in previous experiments, the exact mass distribution matrix for the 100 runs vary. Figure 4-10 shows one possible matrix for each of the mass distributions represented as $\tilde{M}_s$. To randomly pick $U$ different nodes from a given level, we randomly rearrange the columns of the matrix.

Figures 4-11 and 4-12 show the performance when the number of unseen nodes are 2 and 4 respectively. While ProbL2 continues to outperform RawL2 for all the mass distributions, our results suggest that given the same number of unseen nodes the amount of improvement yielded by using hierarchy is not significantly different for the different levels.

In summary, based on our simulations we make the following conclusion. All else being equal, if the mass distributions of the training and the test sets are different, then we can achieve a large improvement by leveraging hierarchy. Our results show that if the mass distributions are such that the observed nodes in the testing examples are unseen in the training data, then the larger the number of testing examples with unseen observed nodes, the larger the magnitude of improvement. Our results also show that the level of the unseen observed nodes does not affect the magnitude of improvement significantly.
Figure 4-10: The different observed mass distributions for the training data set ($M^t$) that are used to evaluate whether the level of the unseen nodes affect the amount of improvement yielded. The figure shows one possible matrix for each of the mass distributions represented as $\tilde{M}_i$, since the number of unseen nodes < the number of nodes in the level.
Figure 4-11: The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data for which there are 2 unseen nodes in level 2 ($M_d$), level 3 ($\tilde{M}_g$) and level 4 ($\tilde{M}_h$).

Figure 4-12: The average performance of RawL2 and ProbL2 along with the performance of the Oracle for the different mass distributions for the training data for which there are 4 unseen nodes in level 3 ($M_e$) and level 4 ($\tilde{M}_i$).
4.7 Quantifying difference in mass distributions between training and test sets

Our results in Section 4.6 show that depending on how the mass distributions between the training and the test set differs, the amount of improvement we achieve on the test set (by leveraging hierarchy) also changes. In this subsection, we quantify the difference between $M^{tr}$ and $M^{tst}$. We then illustrate the relationship between the difference and the amount of improvement we can achieve by leveraging hierarchy.

We quantify the difference between the mass distributions of the training and the test set using Kullback-Leibler divergence (KL divergence). KL divergence is a non-symmetric measure of the difference between two probability distributions $R$ and $Q$. Specifically, KL divergence is a measure of information lost when the probability distribution $Q$ is used to estimate $R$ and is given by:

$$D_{KL}(R||Q) = \sum_i R_i \ln \frac{R_i}{Q_i}$$

(4.2)

Usually, $R$ represents the “true” distribution of data and $Q$ represents the approximation of $R$.

Since we evaluate the performance of the different models learned from the training set on the test set, $M^{tst}$ represents the “true” distribution and $M^{tr}$ an approximation of the “true” distribution. The distance between mass distributions of the training and the test is then given by $D_{KL}(M^{tst}||M^{tr})$.

The non-symmetric property of the KL divergence is appropriate for our application. Say, we are given two datasets $S_1$ and $S_2$ with mass distributions $M^{S_1}$ and $M^{S_2}$. We expect the improvement in performance yielded by ProbL2 to be different in the following cases:

1. **Case 1**: The training set is $S_1$, i.e., $M^{tr} = M^{S_1}$, and the testing set is $S_2$, i.e, $M^{tst} = M^{S_2}$

2. **Case 2**: The training set is $S_2$, i.e., $M^{tr} = M^{S_2}$ and the testing set is $S_1$, i.e, $M^{tst} = M^{S_1}$

80
Figure 4-13: The relationship between the distance between $M^{tr}$ and $M^{tst}$ and the magnitude of AUROC improvement yielded by ProbL2 over RawL2 by leveraging hierarchy.

For example, consider the experiment in Section 4.6 in which the training set has $M^{tr} = M_f$ and the test set has $M^{tst} = M_a$, the performance boost in AUROC on the test set is $\approx 0.081$. If we swap the training and the test set, such that $M^{tr} = M_a$ and $M^{tst} = M_f$, the performance boost in AUROC is much smaller $\approx 0.008$. Since our results in Section 4.6 suggest that the performance improvement depends on the difference in the mass distribution between the training and the test set, we want the distance metric we use reflect this asymmetry. Using KL divergence, the distance between mass distributions for Case 1 is $D_{KL}(M_a || M_f) = 16.62$ and the distance for Case 2 is $D_{KL}(M_f || M_a) = 0.76$.

To analyze the relationship between the distance between training and test mass distributions $D_{KL}$ and the magnitude of improvement, we evaluate both for all the experiments considered in Sections 4.5 and 4.6. Figure 4-13 shows that the relationship between $D_{KL}$ and the magnitude of improvement. The correlation between the two is 0.99 (p-value $< 10^{-7}$) suggesting that larger distance is associated with larger
improvement by ProbL2 over RawL2.

In summary, our results show that when we use a KL divergence based distance metric to quantify distance between $M^{tr}$ and $M^{tst}$, there exists a strong correlation between the distance and the magnitude of improvement obtained by leveraging hierarchy.

4.8 Effect of $\alpha$

For all of the experiments described above, we use $\alpha = 0.25$. In this section, we explore the effect $\alpha$ on the performance improvement achieved by leveraging hierarchy. We use two different setups for the experiments:

**Setup 1:**
Fix: $M^{tr} = M^{tst} = M_a$, i.e., $D_{KL}(M^{tst}||M^{tr}) \approx 0$ (very low)
Vary: $N^l_{leaf}$ and $\alpha$

**Setup 2:**
Fix: $M^{tr} = M_f$ and $M^{tst} = M_a$ i.e., $D_{KL}(M^{tst}||M^{tr}) = 16.62$ (high)
Vary: $N^l_{leaf}$ and $\alpha$

Figures 4-14 and 4-15 show that the percentage improvement obtained by leveraging hierarchy over RawL2 for different values of $\alpha$ for Setup 1 and 2 respectively. In both setups, the percentage improvement $\geq 0$ for all $N^l_{leaf}$ and $\alpha$. This shows that the method that leverages hierarchy either performs comparably or outperforms RawL2 for all $N^l_{leaf}$ and $\alpha$.

Our results also show a clear trend in % improvement versus $\alpha$. For both setups, the performance improvement is small for small values of $\alpha$. As $\alpha$ increases, the improvement increases until it reaches a peak, after which the performance improvement starts to drop. The value of $\alpha$ at which the peak occurs varies depending on $D_{KL}(M^{tst}||M^{tr})$ and $N^l_{leaf}$.

When $\alpha \approx 0$, the probabilities of all of the leaf nodes are identical. In such a scenario, leveraging hierarchy can do little to improve performance over RawL2. Therefore, the improvement is close to 0 for all $D_{KL}(M^{tst}||M^{tr})$ and $N^l_{leaf}$. .
Figure 4-14: The average percentage improvement in performance obtained by leveraging hierarchy when $M_{tr} = M_{ts} = M_α$ (i.e., the distance between mass distributions is small) for different values of $α$.

Figure 4-15: The average percentage improvement in performance obtained by leveraging hierarchy when $M_{tr} = M_f$ and $M_{ts} = M_α$ (i.e., the distance between mass distributions is high) for different values of $α$. 
Figure 4-14 show that when $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}}) \approx 0$, the curves for different $N^{tr}_{\text{leaf}}$ peak at different values of $\alpha$ (Figure 4-14). When comparing $N^{tr}_{\text{leaf}} = 4$ and $N^{tr}_{\text{leaf}} = 256$, our results show that $N^{tr}_{\text{leaf}} = 4$ peaks at a higher value of $\alpha$ than $N^{tr}_{\text{leaf}} = 256$. When the number of training examples is small, despite weak relationships between probability of the leaf node siblings (i.e. a higher $\alpha$), leveraging hierarchy can yield improvement. When number of training examples is large and $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}}) \approx 0$, the training data approximates the test data very well. Therefore, for all $\alpha$ the % improvement is about the same with a small peak at $\alpha = 2^{-5}$.

Figure 4-15 shows that when $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}})$ is high, all the curves peak around $\alpha = 2^{0}$. When the mass distributions between the training and the test set is high, increasing the number of training examples does not help RawL2 to perform better. As a result, we still get a large improvement by leveraging hierarchy. In addition, when $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}})$ is high, leveraging hierarchy helps despite weak relationships between probability of the leaf node siblings. Therefore, the % improvement peaks around $\alpha = 2^{-1}$.

In all our experiments in Section 4.5 and Section 4.6 we used $\alpha = 0.25$. Although the choice was made arbitrarily, our results in Figure 4-14 and 4-15 show that it falls in an interesting region, i.e., close to where the performance improvement peaks.

In summary, based on our simulations we make the following conclusion. For all $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}})$ and $N^{tr}_{\text{leaf}}$, the performance improvement yielded by leveraging hierarchy is small for small values of $\alpha$. As $\alpha$ increases, % improvement increases until it reaches a peak, after which the performance improvement starts to drop. However, the value of $\alpha$ at which the peak occurs varies depending on $D_{KL}(\mathbf{M}^{\text{st}}||\mathbf{M}^{\text{tr}})$ and $N^{tr}_{\text{leaf}}$. 

84
4.9 Mimicking the effect of differing mass distributions when transitioning from ICD-9 to ICD-10

As discussed in Section 4.6, the challenge of unseen nodes and the resulting difference in mass distribution between training and test data will be particularly relevant for clinical datasets when hospitals transition from ICD-9 to ICD-10 codes. The historical data available to learn predictive models will be in ICD-9 codes, while new patient information will be recorded using ICD-10 codes.

In order to apply predictive models learned using historical data to patient data, we will have to do the following:

1. Map the ICD-9 codes in the historical data to ICD-10 codes.

2. Learn a predictive model using the ICD-10 mapping of the historical data.

Since ICD-10 codes contain significantly more distinct codes with greater specificity, the new patient data to which the trained models will be applied will contain nodes that are unseen during training. In this section, we mimic the effect of transitioning from ICD-9 (the less specific coding system) to ICD-10 (the more specific coding system) using real data. To do this, we do the following:

1. We treat the original ICD-9 data (ICD9-original) as the more specific coding system.

2. We consider ICD9-truncated- a less specific version of ICD9-original, where we assume that the leaf nodes in ICD9-original do not exist. Therefore, in ICD9-truncated coding system, we associate a example that is assigned a leaf node, with the parent of the leaf node.

3. We represent the training data in ICD9-truncated coding system.

4. We use ICD9-original for the test data.
In our setup, \textit{ICD9-truncated} plays the role of ICD-9 and \textit{ICD9-original} plays the role of ICD-10 codes. We ran experiments using real clinical data for the task of predicting incident heart failure, \textit{Task-HF} (discussed before in Chapter 3).

First, we analyze the difference in mass distributions between the training and the test sets when using:

1. \textit{ICD9-original} for both training and testing (the experiments in Chapter 3).

2. \textit{ICD9-truncated} for training and \textit{ICD9-original} for testing.

We cannot directly use Equation 4.1 to compute the mass distributions of the data for \textit{Task-HF}. In Equation 4.1, $M(v)$ represents the fraction of examples for which node $v$ is the observed node. In \textit{Task-HF}, an example associated with a patient can be assigned multiple nodes within a tree. Therefore, given a tree $T$, an example $e_i$ is a tuple of a set of nodes $N_T^i$ (instead of a single node $n_i$ as in the simulation) and the label $y_i$, i.e., $e_i = (N_T^i, y_i)$. To account for these differences, we used a modified definition of mass distribution across nodes within a tree represented by $M_T \in \mathbb{R}^{1 \times V_T}$ where $V_T$ is the number of variables (nodes) within the hierarchy tree $T$. Specifically,

$$M_T(v) = \frac{\sum_{i=1}^{N} 1(v \in N_T^i)}{\sum_{v \in T} \sum_{i=1}^{N} 1(v \in N_T^i)} \tag{4.3}$$

where $1$ is the indicator variable, $M_T(v) \in [0, 1]$ and $\sum_{v \in T} M_T(v) = 1$.

Table 4.3 shows the average KL-divergence (across the 100 experiments) for trees associated with the top five variables that have the highest absolute value of univariate correlation with the outcome. The correlation was computed using the \textit{Raw Binary} representation of \textit{ICD9-original} and the outcome. As expected, Table 4.3 shows that when using \textit{ICD9-truncated} for training, the KL-divergence is significantly higher than when using \textit{ICD9-original} for training.

Table 4.4 shows the performance along with the % improvement when using \textit{Raw Binary} and \textit{Prob} for \textit{ICD9-original} and \textit{ICD9-truncated}. We show results for \textit{RawL1} instead of \textit{RawL2} since our analysis in Chapter 3 (Section 3.7) showed that
Table 4.3: KL-divergence when using ICD9-original and ICD9-truncated for training. For both cases, ICD9-original is used for testing.

| ICD-9 tree | $D_{KL} (M^{test}||M^{tr})$ |
|------------|----------------------------|
| 401        | ICD9-original: 0.00 | ICD9-truncated: 33.84 |
| 564        | ICD9-original: 0.29 | ICD9-truncated: 34.18 |
| 414        | ICD9-original: 0.14 | ICD9-truncated: 33.66 |
| 272        | ICD9-original: 0.02 | ICD9-truncated: 33.80 |
| 782        | ICD9-original: 0.33 | ICD9-truncated: 33.20 |

Table 4.4: KL-divergence when using ICD9-original and ICD9-truncated for training. For both cases, ICD9-original is used for testing.

<table>
<thead>
<tr>
<th>AUROC</th>
<th>F-SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RawL1</td>
<td>ICD9-original: 0.69</td>
</tr>
<tr>
<td>ProbL2</td>
<td>ICD9-original: 0.71</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.61%</td>
</tr>
</tbody>
</table>

Raw Binary performs better when the model is sparse. The results in Table 4.4 are consistent with our simulation results which suggests that a higher $D_{KL} (M^{test}||M^{tr})$ corresponds to larger improvement in performance by leveraging hierarchy than a smaller value of $D_{KL} (M^{test}||M^{tr})$.

Our results also show that the performance of RawL1 drops significantly when using ICD9-truncated for training instead of ICD9-original for both AUROC and F-SCORE. The performance of ProbL2, on the other hand, drops only by a small amount. By exploiting structural relationships, ProbL2 is able to infer relationships between unseen nodes and seen nodes to yield better predictive performance.

In summary, by mimicking the scenario when hospitals transition from ICD-9 to ICD-10, our experiments highlight the importance and the relevance of exploiting hierarchy to achieve a large gain in predictive performance in such a scenario.

### 4.10 Summary and Conclusions

The goal of this chapter was to better understand the properties of the data that affect the amount of improvement that one can expect by leveraging hierarchical relation-
ships between the variables. We generated a simulation data that is an approximation of a real clinical dataset. Specifically, the variables in the data are structurally related to one another according to a hierarchy tree similar to the ICD-9 hierarchy. In addition, the label (outcome) of a datum depends on the leaf nodes of the tree, i.e., the nodes that contain the most specific information.

In our experiments, we varied the observed properties of data such as the number of training examples and the distribution of examples across different nodes in the hierarchy tree. We demonstrate that as the number of training examples increases the amount of improvement by leveraging hierarchy decreases. We showed that the trend is consistent for different mass distributions. Using a KL-divergence based distance metric, we demonstrate that the distance between the mass distribution of the training and test set is highly correlated with the magnitude of improvement (correlation coefficient of $\approx 0.99$).

We also analyzed the effect of varying the underlying properties of the data. In particular, we varied $\alpha$, a measure of difference between the relationship of neighboring nodes with the label. When $\alpha = 0$ the sibling nodes have identical probability of a positive label. When $\alpha \geq 2$, the probabilities of the siblings are independently sampled from a standard uniform distribution. We show that the magnitude of improvement peaks when $0 < \alpha < 2$. The value of $\alpha$ that yields the largest improvement depends on the mass distribution of the data and the number of examples used to learn the models.

Lastly, we demonstrated the relevance and significance of leveraging hierarchy when hospitals transition from the less-specific ICD-9 to more-specific ICD-10 codes.
Chapter 5

Exploiting Temporal Information in EHR Data

5.1 Introduction

Because it contains repeated measurements of a patient’s state over time, EHR data contain important information about the evolution of disease. In principle, this information can be used to build models that can help predict disease progression. However, the longitudinal patient data stored in EHRs present a multitude of technical challenges for building predictive models. Patient data are recorded only during a healthcare episode or when a patient visits the hospital for routine medical care. This leads to irregular sampling of data, i.e., the time between measurements vary within a patient and across patients. Another characteristic of EHR data is that patients are tracked for different periods of time.

In this chapter, we explore different approaches to using the temporal information in EHR data to build predictive models for risk stratification. In our work in earlier chapters, we use patient data from the most recent time window to predict the outcome. In this chapter, we want to learn a predictive model that can use historical patient information up to and including the present to predict an adverse outcome. The goal of our work is to investigate how to represent the temporal information in the medical history and how to use the representation to learn such a predictive
We present our work in the context of risk stratifying patients with compromised kidney function. Previous work in developing predictive models for adverse outcomes such as end-stage renal disease (ESRD) and death in patients with chronic kidney disease (CKD) use data collected at a single fixed point in time. E.g., Tangri et al. [28] use EHR data only from the initial nephrologist visit. Similarly, Hallan et al. [16] and Landray et al. [21] use data collected at the beginning of the prospective study to develop the predictive models. However, chronic diseases such as CKD evolve over time. The process of evolution of the disease, in addition to the current state of the patient, may affect the future health of the patient.

More recently, there have been efforts on using temporal information in the context of CKD. In [26], Luo et al. explored sequential data modeling techniques such as Markov processes to estimate kidney disease stage transition probabilities. The authors use only longitudinal eGFR measurements as the observed data and do not use any other patient information in their sequential models. Luo et al. present and discuss the different probabilities learned by their models from the data. However, the usefulness of their models for predicting future state of a patient is unclear.

Although there have not been many efforts on using temporal information in the context of CKD, there are existing methods that attempt to capture time-varying effects of predictors collected from multiple time points in the past. Cox proportional hazard models with time-varying coefficients have been studied in the context of survival analysis [9]. In the survival analysis literature, there are two main approaches that use time-varying coefficients: time-varying predictors and time-varying effects [6]. Let $\mathbf{x}_{i,t}$ represent the predictors for patient $i$ at time $t$ and $y_{i,t}$ represent the outcome at time $t$. The time-varying predictors approach uses $\mathbf{x}_{i,t}$ to predict $y_{i,t+1}$, such that the same predictor is allowed to have different effects on outcome depending on 1) the value of the predictor and 2) the time $t$ when the predictor was measured. In contrast, the time-varying effects approach learns a model that uses $\mathbf{x}_{i,t}$ to predict $\{y_{i,t+1}, \ldots, y_{i,t+G}\}$, where $G$ represents the number of time points in the future. This approach attempts to capture the different effects of the same fixed risk
factors measured at $t$ on short-term outcome versus long-term outcome.

Another approach that has been studied in the literature to capture the temporal dynamics of past medical history involves extracting frequent temporal sequences and using them as features. In [38], Toma et al. extract temporal patterns of severity scores of six different organ systems in the past to predict mortality at day $d$. To handle patients with varying number of days in the ICU, the authors learn separate models, for each of the first $D$ days in the ICU using the temporal patterns as features. A model for day $d \leq D$ uses data from patients who stayed at least $d$ days in the ICU. As $d$ increases, the number of patients with at least $d$ days in the ICU decreases, while the space of all possible patterns, and consequently the feature dimensionality, increases. This makes the approach susceptible to overfitting. Another drawback of this approach is that it is unable to handle irregular sampling of data.

Here, we present a multitask based machine learning approach to learn predictive models that use longitudinal measurements of multiple predictors such as diagnosis, procedures, medications, labs and vital signs. Our approach attempts to capture the time-varying effects of predictors collected from multiple time points in the past to predict short-term patient outcome. In contrast to the approaches described above, our method develops a model that use $\{x_{i,t}, x_{i,t-1}, \ldots, x_{i,t-H}\}$, where $H$ represents the number of time points in the medical history, to predict $y_{i,t+1}$. Using our method, we attempt to capture the varying effects of the same predictors collected at different points of time in patient history on short-term outcome. In addition, our multitask learning approach is able to handle patient data with irregular sampling and patient history of different lengths. In the context of risk stratifying patients with compromised kidney function, we demonstrate that by leveraging temporal information in EHR data, our method can yield better risk stratification than models that ignore the information.
5.2 The Data

For this analysis, we extracted data from patients with compromised renal function who were also diagnosed with hypertension, diabetes, or both from the Mount Sinai Data Warehouse. We focus on this population because approximately two thirds of cases with compromised renal function are attributable to diabetes or hypertension [22]. We only consider patients from the study population who satisfy the following inclusion criteria:

1. Patients who have at least a 2-year medical history on record.

2. Patients whose median estimated glomerular filtration rate (eGFR) in the first year in the database is between $45$ to $90 \text{ ml/min/1.73m}^2$. We focus on this patient population since it is important to accurately risk stratify patients before they progress to Stage 3b - the inflection point for outcomes such as end-stage renal disease (ESRD) and adverse cardiovascular events.

There are 6,435 patients in the database that satisfy our inclusion criteria. Approximately 28% of the patients in this population have eGFR in the interval $[45,60]$ and the rest of the patients have eGFR in the interval $(60,90]$.

5.3 Problem Formulation

We consider the clinical task of predicting loss of kidney function for a patient over the next year using longitudinal EHR data.

Given a sequence of time-stamped outpatient eGFR values for a patient, we generate multiple examples per patient. More specifically, we consider each outpatient eGFR measurement as an example. We exclude inpatient eGFR measurements for generating examples since we are interested in progression of a patient relative to his/her baseline eGFR value. Inpatient measurements tend to fluctuate a lot from the baseline value. One potential reason could be an acute kidney injury that causes patient to get admitted to the hospital. Although we exclude inpatient eGFR when
generating examples, we consider other information from the inpatient visits in our predictive model.

An example is associated with a tuple of a patient $P$, a time-stamp $t_0$, and an outpatient eGFR measurement. In our study, given a patient we only consider examples that satisfy the following inclusion criteria:

1. Patient $P$ has at least two outpatient eGFR measurements in the 1-year window following $t_0$, and the 1-year window preceding $t_0$. This is done to ensure a robust measure of short-term progression.

2. The previous example from patient $P$ is at least 1-year earlier than the current example, i.e., the examples are at least one year apart. This is done to avoid having a disproportionate number of examples from sicker patients, who tend to have more outpatient eGFR measurements than those who are stable.

From 6,435 patients, we extract 12,337 examples.

We represent each example by extracting the predictors from the patient’s medical history before time $t_0$. Table 5.1 lists the predictors that we include in our predictive model. For numerical predictors such as vital signs and lab values, we compute the mean, median, min, max and standard deviation of each of the predictors over a specified time-window. In addition, we also compute the linear slope of the numerical predictors in a time-window. More specifically, we fit a line (using least squares) and use the slope of the fit as a feature. In contrast to process of generating examples (as discussed above), we use patient information from both outpatient and inpatient visits when extracting other predictors from a patient’s medical history.

All predictors are represented using binary variables. We represent diagnoses, procedures and medications as a binary variable indicating whether or not the patient associated with the example was assigned an ICD-9 code or prescribed a medication during a specified time-window in the past. We discretize the numerical predictors into four bins based on the quartiles of the corresponding predictor and then map them into binary variables. For example, we map the mean systolic blood pressure for the most recent time window, into four bins: $SBP \leq 120$, $120 < SBP \leq 130$, $130 < SBP \leq 140$, and $SBP > 140$. 

93
130 < SBP ≤ 140 and SBP > 140, each corresponding to a binary variable. For a patient with SBP of 125 mm Hg, we set the binary variable corresponding to 120 < SBP ≤ 130 to 1 and others to 0.

We use race of the patient as one of the predictors in our model. We represent it as a binary variable indicating whether a patient is an African American or not. This is consistent with the CKD-EPI formula (discussed in Chapter 1) that we use to estimate eGFR from serum creatinine measurement that also considers race as a binary variable.

We are interested in predicting short term progression of kidney loss for a given example. We define short-term progression based on the drop of eGFR over the next year. As discussed in Chapter 2, patients might have temporary fluctuations in their serum creatinine levels, resulting in fluctuations in the estimate eGFR values. These fluctuations are not representative of a patient’s true baseline eGFR. To handle fluctuations in the eGFR values, we compute the median eGFR in the most recent 1-year history eGFR-past and 1-year in the future eGFR-future. Next, we compute the percentage drop (%Δ) as follows:

\[
%\Delta = \frac{eGFR-past - eGFR-future}{eGFR-past} \times 100
\]  

(5.1)

The above definition of short term progression is different from the one used in Chapter 3. In Chapter 3, we used a magnitude drop in eGFR instead of a percentage drop. There we consider progression only in patients with baseline eGFR in the interval [69,90]. Here, we consider patients from a bigger range, i.e. baseline eGFR in the interval [45,90]. By construction, patients with lower baseline eGFR are less likely to experience a large magnitude drop in the future. Therefore, we define progression based on sharp drop in percentage from baseline eGFR rather than based on a sharp drop a magnitude.

We formulate the task of predicting progression as a binary classification task where the example is assigned a positive label if %Δ ≥ threshold, and a negative label if the %Δ < threshold. Since there is not a well-established threshold in the
Table 5.1: Predictors extracted from the past medical history of a patient for predicting progression of kidney function loss. The numbers in parenthesis for each predictor group is the number of binary variables associated with the given set of predictors. For demographics and numerical predictors, the table also shows the statistics for the patients in the recent EHR data.

<table>
<thead>
<tr>
<th>Predictors</th>
<th>Demographics (6)</th>
<th>Vital Signs (56)</th>
<th>Lab values (60)</th>
<th>Diagnosis and Procedures (8174)</th>
<th>Medications (180)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictors</td>
<td>Age (Mean ± SD)</td>
<td>Systolic Blood Pressure (Mean ± SD)</td>
<td>eGFR (Mean ± SD)</td>
<td>ICD-9 codes</td>
<td>Anti-hypertensives, Medications for Type-2 diabetes, Insulin, Nephrotoxic medications</td>
</tr>
<tr>
<td></td>
<td>67.7 ± 11.5 years</td>
<td>132.8 ± 16.1 mm Hg</td>
<td>66.8 ± 12.1 ml/mm/1.73 m²</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gender</td>
<td>Diastolic Blood Pressure (Mean ± SD)</td>
<td>HbA1c (Mean ± SD)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Male: 40%</td>
<td>73.3 ± 12.4 mm Hg</td>
<td>7.21 ± 1.08 %</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Race</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>African American: 27%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the literature, we build models using two thresholds 10% and 20%. These threshold approximately corresponds to the top quartile and the top decile of the %Δ in our patient population respectively.

Using EHR data from $M$ patients, we obtain dataset $D$

$$D = \{(x_i, y_i) | x_i \in \mathbb{R}^F, y_i \in \{-1, 1\}\}_{i=1}^N$$ (5.2)

where $x_i$ represents the $i^{th}$ example, $F =$ dimensionality of the feature space and $N =$ number of examples. In total, we extract around 8500 binary variables from a patient’s medical history. Because of the large number of variables, each variable is sparsely represented in the data.
5.4 Methods

We describe three different approaches to incorporating longitudinal data in our predictive models. We also discuss how the approaches handle various challenges associated with using EHR data.

One of the key modeling decisions that has to be made is picking a level of granularity based on time to define a time-window. Choosing a fine granularity such as a day may not be relevant for analysis of chronic conditions. On the other hand, choosing too coarse a granularity may result in loss of useful temporal relationships. A complication in choosing a window size is that patient data are recorded only during a healthcare episode or when a patient visits the clinic for routine medical care. This leads to irregular sampling of data i.e., the times between measurements vary within a patient and across patients. In Section 5.5.1 we discuss how we choose the granularity level for our application.

Given a granularity level \( L \), we divide the medical history of a patient into \( T \) non-overlapping time-windows, and then construct a logistic regression model \( f : \mathbb{R}^F \rightarrow \mathbb{R} \) using patient information from the most recent \( T \) time windows.

\[
f(x) = \frac{1}{1 + e^{-(w^T x + c)}}
\]  

where \( w \in \mathbb{R}^F \) are the feature weights and \( c \) is the intercept. Specifically, we use an L2-regularized logistic regression model that solves the following optimization problem:

\[
\min_w \sum_{i=1}^{N} \log(1 + e^{-(y_i(w^T x_i + c))}) + \lambda \|w\|_2^2
\]

where \( \lambda \) is a tuning parameter. The L2-regularization reduces overfitting relative to the unregularized logistic regression. This is important for our application since the feature vectors are sparse.

Figure 5.1 depicts the three different approaches we use to build the models. Section 5.4.1 and 5.4.2 describe the approaches in detail.
5.4.1 Non-Temporal Approach

In this approach, when extracting variables for example $x_i$ we aggregate the information across all $T$ time-windows. E.g., a binary variable representing a diagnosis is set to 1 if a patient is assigned that diagnosis during a medical encounter in any of the $T$ time-windows. When computing the mean, median and other statistics for numerical predictors, we aggregate the measurements taken during all of the medical encounters in the $T$ time-windows. This approach represents an example $x_i$ by an $F$ dimensional vector, where $F = V$. The Non-Temporal approach handles the challenge of irregular sampling and missing data by aggregating patient data over the windows for which data is available. While the Non-temporal approach uses the longitudinal information, it does not capture any temporal information in the data. E.g., a binary variable representing a diagnosis is set to 1 regardless of whether the patient was given the
diagnosis in the first time-window \((t = 1)\) or the \(t^{th}\) time-window. Once the variables shown in Table 5.1 are extracted, we only consider variables that have a statistically significant \((p < 0.05)\) univariate correlation with \(y\) in the training set. Next, we learn a logistic regression model using Equation 5.4.

### 5.4.2 Temporal Approaches

We present two approaches, Stacked-Temporal and Multitask-Temporal, to model the temporal information in the longitudinal data. For both methods, we first extract variables for each of the \(T\) time-windows separately by aggregating the irregularly sampled patient information within the time-window. This allows us to retain the temporal information between the time windows. E.g., this representation can capture when (in which time-windows) a patient was assigned a certain diagnosis. After extracting variables, we only keep variables that have a statistically significant \((p < 0.05)\) univariate correlation with \(y\) in at least one of the \(T\) windows in the training set. Since we use such a liberal criterion, some variables that are not predictive might still be included in the model. However, since we use regularized logistic regression models, we expect the regularization to assign low weights to such variables.

**Stacked-Temporal**

For each example \(i\), we extract variables from each of the \(T\) time windows. Let \(x^t_i \in \mathbb{R}^V\) represent the vector of variables extracted from the \(t^{th}\) window for example \(i\), where \(V=\)number of variables. Next, the Stacked-Temporal approach concatenates the variables from all windows to represent example \(x_i \in \mathbb{R}^F\), i.e. \(x_i = [x^1_i, ...., x^T_i]\), where \(F = \)number of variables \(\times T\). Finally, we learn a linear predictive model \(f(x_i)\) by solving Equation 5.4.

Some the examples maybe have missing data for one for more time windows. To handle such cases, Stacked-Temporal uses imputation. While there are several imputation techniques one could use [2], we use a simple imputation approach to fill in the missing data. For categorical variables such as diagnoses, procedures and
medications, we set the value of the predictor to 0. For numerical predictors, we use the value of the closest time-window for which measurements are available. If two time-windows with measurement are equally close, we take the average.

One of the disadvantages of Stacked-Temporal is that the feature dimensionality $F$ increases proportionally to $T$. Therefore, as we increase the number of time-windows, the Stacked-Temporal approach is likely to suffer from overfitting.

**Multitask-Temporal**

In this approach, we formulate the problem as a multi-task learning problem. Specifically, we consider the task of predicting the outcome using each $t^{th}$ window as a separate task, where $t = 1, ..., T$. Let the set $\text{win}(i) \subset \{1, ..., T\}$ denote the time windows for which data is available for the $i^{th}$ example.

For each task $t$, the data set $D_t$ is

$$D_t = \{(x_t^i, y_i) | x_t^i \in \mathbb{R}^V, y_i \in \{-1, 1\}, t \in \text{win}(i)\}_{i=1}^N \quad (5.5)$$

We learn all $T$ tasks jointly using the following multi-task formulation:

$$\min_{w_1, ..., w_T} \sum_{t=1}^T \left[ \sum_{x_t^i \in D_t} \log(1 + \exp(-y_j(w_t^T x_t^i + c_t))) + \lambda_1 \|w_t\|_2^2 \right] + \lambda_2 \sum_{t=1}^{T-1} \|w_t - w_{t-1}\|_2^2 \quad (5.6)$$

where $w_t$ are the weights for the $t^{th}$ task and $\lambda_1$ and $\lambda_2$ are the tuning parameters. This multitask learning step is shown schematically in Figure 5-1.

Although we learn separate $f_t(x_t^i)$ for $t = 1, ..., T$, the joint learning in Equation 5.6 enforces a temporal smoothness constraint on the weights from adjacent time-windows. Specifically, the last term in Equation 5.6 encourages the weights of the neighboring windows to be similar, unless the data strongly suggests that the weights be different. Therefore, this constraint helps reduce overfitting of the tasks for which $N_t$ is small.

After learning separate models for each task, we learn a final model that aggregates
Figure 5-2: Illustration of the missing value estimation used to learn the final model that aggregates the predictions obtained from each task. Once the tasks are learned jointly using the multitask framework, the models are applied to the corresponding available time windows of the examples to produce the prediction score matrix $A$. A missing value estimation method is applied to obtain a complete prediction score matrix $\tilde{A}$. Specifically, we replace the missing value for example $i$ with the prediction from the nearest available time window. If there are two windows that are equally close, we take the average.

The predictions obtained from these models to generate a final prediction. However, since some of the time windows are missing, in this stage, we use perform a missing value estimation.

Figure 5-2 shows the overview of the missing value estimation approach. Using all examples in the training set, using $f_t(x^i_t)$ for $t = 1, \ldots, T$, we construct an incomplete prediction score matrix $A \in \mathbb{R}^{N \times T}$ given by:

$$A_{i,t} = \begin{cases} f_t(x^i_t), & \text{if } t \in \text{win}(i). \\ \text{NaN}, & \text{otherwise.} \end{cases}$$  \hfill (5.7)
Next, we perform missing value estimation of the elements in $A$ where $A_{i,t} = \text{NaN}$. Specifically, we replace the missing value with the prediction from the nearest $A_{i,t}$ time window of example $i$. If there are two windows that are equally close, we take the average. This approach has been used previously in [43]. We refer to the resulting prediction matrix as $\tilde{A}$.

Finally, we use $\tilde{A}$ to learn our final model $g(\tilde{a}_i)$, where $\tilde{a}_i = A_{i,\cdot}$ that aggregates those predictions to yield a single prediction.

Given a new unlabeled example $x_i$, we first obtain the intermediate predictions $\{\hat{y}_{t,i} \mid t \in \text{win}(i)\}$ using separate models for each $t$. Using the steps described above, we generate $\tilde{a}_i$, which is then used to yield an aggregate prediction using $g(\tilde{a}_i)$.

Our proposed Multitask-Temporal approach has two main characteristics:

1. During the multitask learning phase, our approach handles missing data without imputation, thereby reducing noise during learning. Since we learn separate $f_t$ for each time-window, this formulation allows the number of examples $N_t$ for each task to be different. Therefore, if a patient has little to no information during a given time window $T$, we do not use the time window during the multitask learning phase. However, we still impute missing prediction scores in the aggregation stage. In Section 5.5.3 we compare the performance of proposed approach with the approach where we impute data instead of the prediction scores.

2. Our approach learns separate models for each time windows jointly. Since we consider each time window as a separate task, unlike Stacked-Temporal, the feature dimensionality does not increase proportionally with the number of time-windows considered. On the other hand, the number of tasks increases proportionally with the number of time-windows considered. The joint learning approach (Equation 5.6) reduces overfitting for tasks with fewer examples. This is important since the number of examples declines as the value of $t$ increases (because not every patient will have $t$ windows of medical history).

In Equation 5.6, when $\lambda_2$ is 0 the optimization does not impose any temporal
constraint to the weights. This is equivalent to learning each of the $T$ tasks independently. We refer to this approach as SingleTask-Temporal. In Section 5.5.3, we evaluate SingleTask-Temporal against Multitask-Temporal to evaluate the advantage of learning multiple tasks jointly.

5.5 Experiments and Results

5.5.1 Experimental Setup

Using the data and the methods described above, we learned and evaluated risk stratification models for predicting short-term progression of loss of kidney function in the next year. For all our experiments, we set the granularity of the time-window to 6 months since on average the patients in our dataset have about one medical encounter with an eGFR measurement every 6 months. We consider models that incorporate longitudinal data from up to 10 time-windows, i.e., 5 years, in the past.

Figure 5-3 shows the fraction of examples in our dataset for which a given time-window $t$ has at least one medical encounter.

We formulate the task of predicting progression as a binary classification task where the example is assigned a positive label if $\%\Delta \geq$ threshold, and a negative label
if the $\% \Delta < \text{threshold}$. In our experiments, we considered models using 10% and 20% as threshold. These threshold approximately corresponds to the top quartile and the top decile of the $\% \Delta$ in our patient population respectively.

To learn the risk stratification models, we first divide the 6,435 patients into training and holdout patients with an 80/20 split. We learn the models using the examples from the patients in the training set. We select the tuning parameters $\lambda_1$ and $\lambda_2$ using 5-fold cross-validation on the training set. Finally, we evaluate the performance of the trained models on the examples from the holdout patients using the area under the receiver-operating characteristic (AUROC). For each approach, we generate 100 different training and holdout splits and repeat the experiments on each of the 100 splits.

Table 5.2 shows the average number of positive examples in the holdout set across the 100 splits for the two thresholds. Table 5.3 shows the number of variable per time window for each approach for the different years of patient history.

When learning $f_t$, we consider a time window for a patient as not missing, if one of the following criteria is met:

1. The patient has at least one medical encounter with an eGFR measurement

<table>
<thead>
<tr>
<th>Threshold 10%</th>
<th>Threshold 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Years of patient history</strong></td>
<td><strong>Non-Temporal</strong></td>
</tr>
<tr>
<td>1</td>
<td>740 (25)</td>
</tr>
<tr>
<td>2</td>
<td>792 (33)</td>
</tr>
<tr>
<td>3</td>
<td>765 (31)</td>
</tr>
<tr>
<td>4</td>
<td>730 (32)</td>
</tr>
<tr>
<td>5</td>
<td>732 (31)</td>
</tr>
</tbody>
</table>

Table 5.2: The average number of positive examples for different thresholds in the holdout sets. The value in the parenthesis shows what fraction of the total number of examples is positive.
Figure 5-4: Average performance of the different methods for threshold 10% and 20%. The x-axis shows the number of years of patient history that is considered for the model. The error bars show the standard error in the 100 splits.

2. The patient has at least five medical encounters

In our preliminary analysis, we considered a time window with any available information during a time window as not missing. We found that using the number of medical encounters and presence of lab or vital sign measurements as a proxy of the quality of information within a time window improved performance.

5.5.2 Comparing Non-Temporal and Temporal Approaches

Figure 5-4 shows the results for our experiments. The x-axis represent the length of patient history considered in terms of years. For 0.5 years (or $T = 1$) all three methods have equivalent performance. Since $T = 1$, there is only a single time-window and there is no temporal information to exploit.

Overall, the results in Figure 5-4 suggest that incorporating longitudinal information for risk stratification of short-term loss of kidney function improves prediction. The amount of improvement varies across different methods, different thresholds and the length of patient history considered.

Multitask-Temporal performs at least as well as Stacked-Temporal for all the different lengths of patient history considered, across both the thresholds, and consis-
tently dominates the Non-temporal approach. Figure 5-4 shows that as we increase the length of patient history considered, the performance of Stacked-Temporal eventually dips. On the other hand, the performance of Multitask-Temporal improves and eventually plateaus.

For Stacked-Temporal, as we add more patient history, there exist competing factors: the predictive information in the additional time-windows is offset by the increased dimensionality and noise due to imputation. Since the number of examples decreases significantly with increasing length of patient history, Stacked-Temporal has to perform imputation for a large number of missing time windows. This adds noise in the training data.

For thresholds of both 10% and 20%, the performance of Stacked-Temporal initially improves. However, the overfitting and noise due to imputation becomes dominant with the 7th (3.5 years) or the 5th window (2.5 years) causing the performance to dip.

We observe that the rate of drop in performance is different for the 10% and 20% thresholds. For the 10% threshold, while the performance starts to dip, Stacked-Temporal still outperforms Non-Temporal. On the other hand, for the 20% threshold, Stacked-Temporal underperforms relative to Non-Temporal after 2.5 years. One possible reason for the difference is the fraction of positive examples for each threshold. The ratio of positive examples for 10% and 20% threshold is approximately 30% and 10% respectively. The larger class imbalance for 20% threshold makes it more susceptible to overfitting.

Another reason could be the difference in the number of features that have a significant correlation with \( y \). For 1 year of patient history, there were approximately 1000 features and 1500 features with a significant correlation \( (p < 0.05) \) with \( y \) for 10% and 20% threshold respectively (Table 5.3). As we increase the years of patient history considered in our model, the higher feature dimensionality coupled with higher class imbalance make Stacked-Temporal more likely to overfit for the 20% threshold.

Our results clearly indicate that Multitask-Temporal clearly dominates Stacked-Temporal. Although the performance of Stacked-Temporal can vary with the choice
of imputation approach used, the ability of Multitask-Temporal to yield better performance while avoiding imputation makes it a superior choice between the two. We henceforth consider Multitask-Temporal as the only temporal approach.

For each threshold, the AUROC of Multitask-Temporal is significantly (statistically) higher than that of the Non-Temporal approach. These results highlight the importance of exploiting temporal information.

To further illustrate how exploiting longitudinal data and its temporal information can improve performance, Table 4 compares the performance of the best Multitask-Temporal models (models using 3 and 2 years of patient history for threshold of 10% and 20% respectively) with the model that uses only the most recent time-window, i.e., $T = 1$. We consider this as the baseline because this does not use any longitudinal or temporal information.

To compute the performance measures shown in Table 5.4, we consider the test examples with predicted probability of outcome in the top quartile as positive.

Next, we compare the performance of Multitask-Temporal with Non-Temporal. For the 10% threshold, Multitask-Temporal resulted in a boost in both sensitivity and positive predictive value. To make the comparison easier, we fix the precision of Multitask-Temporal to 48.7% and compute the corresponding sensitivity. For a precision of 48.7%, Multitask-Temporal yields a sensitivity of 55.1% compared to 48.7% for the Non-Temporal approach, a significant improvement. For 20% threshold, Multitask-Temporal also outperformed Non-Temporal, although the magnitude of improvement was smaller. For each threshold, the increase in sensitivity and positive predictive value were statistically significant relative to the Non-Temporal approach.

To further convey the ability of our models to risk stratify patients, we divide the test patients into quintiles (as often done in clinical studies) based on the predicted probability of outcome. Next, for each quintile, we compute the observed probability of a positive outcome. Figure 5-5 shows that the observed probability of the outcome increases with each quintile for both thresholds. For thresholds of 10% and 20%, patients in the 5th quintile are at 3.7-fold and 7.7-fold greater risk of progression than patients in the 1st quintile.
Table 5.4: Performance comparison of Multitask-Temporal approach with the models that use the most recent time-window. The * indicates that the average is significantly ($p < 0.05$) different from the average performance of Non-temporal when evaluated using a matched t-test. For the performance measures that are statistically significant, we achieved p-values $< 0.001$.

<table>
<thead>
<tr>
<th></th>
<th>Threshold 10%</th>
<th>Threshold 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Non-Temporal</td>
<td>Multitask-Temporal</td>
</tr>
<tr>
<td>AUROC</td>
<td>0.650</td>
<td>0.693*</td>
</tr>
<tr>
<td>Average Sensitivity</td>
<td>0.375</td>
<td>0.410*</td>
</tr>
<tr>
<td>Average True Positive</td>
<td>383</td>
<td>419*</td>
</tr>
<tr>
<td>Average Positive Predictive Value</td>
<td>0.487</td>
<td>0.533*</td>
</tr>
<tr>
<td>Average Specificity</td>
<td>0.810</td>
<td>0.827*</td>
</tr>
<tr>
<td>Average True Negative</td>
<td>1698</td>
<td>1734*</td>
</tr>
<tr>
<td>Average Negative Predictive Value</td>
<td>0.729</td>
<td>0.744*</td>
</tr>
<tr>
<td>AUROC</td>
<td>0.700</td>
<td>0.722*</td>
</tr>
<tr>
<td>Average Sensitivity</td>
<td>0.501</td>
<td>0.525*</td>
</tr>
<tr>
<td>Average True Positive</td>
<td>176</td>
<td>185*</td>
</tr>
<tr>
<td>Average Positive Predictive Value</td>
<td>0.236</td>
<td>0.246*</td>
</tr>
<tr>
<td>Average Specificity</td>
<td>0.784</td>
<td>0.787</td>
</tr>
<tr>
<td>Average True Negative</td>
<td>2168</td>
<td>2176</td>
</tr>
<tr>
<td>Average Negative Predictive Value</td>
<td>0.923</td>
<td>0.926</td>
</tr>
</tbody>
</table>

Figure 5-5: Fraction of examples with a positive outcome in each predicted risk quintile for threshold 10% and 20%.
Figure 5-6: Comparison of the performance of Multitask-Temporal with Singletask-Temporal and Multitask-Temporal with imputation. Multitask-Temporal outperforms both Singletask-Temporal and Multitask-Temporal with imputation, highlighting the importance of learning the tasks jointly and the effect of noise due to imputation.

5.5.3 Evaluating Characteristics of Multitask-Temporal

As discussed in Section 5.4, our proposed Multitask Temporal approach has two main characteristics: 1) it addresses missing data without imputation and 2) it handles the curse of high dimensionality by learning multiple tasks with smaller dimensionality jointly.

In this subsection, we evaluate the effects of each of the two characteristics separately. First, we compare the performance of the proposed Multitask-Temporal approach with an approach that use the multitask framework with data imputation. We refer to this as Multitask-Temporal with Imputation. Next, we evaluate the effect of learning the tasks jointly. The proposed Multitask-Temporal approach imposes a temporal constraint on the weights from adjacent time windows using the hyperparameter $\lambda_2$. To evaluate the advantage of learning tasks jointly, we compare the performance of Multitask-Temporal to Singletask-Temporal, i.e., an approach that learns each task independently. Singletask-Temporal approach solves the optimization problem shown in Equation 5.6 with $\lambda_2$ set to 0.
Figure 5-6 shows the performance of Multitask-Temporal and the other two approaches. The performance of Multitask-Temporal and Multitask-Temporal with imputation are comparable for small numbers of years of patient history considered. This is expected since approximately 90% of the examples have 1.5 years of patient history (Figure 5-3). However, as we increase the number of years of patient history considered, the performance of MultiTask-Temporal with imputation starts to drop more sharply than the performance of Multitask-Temporal for both thresholds. The sharp drop can be attributed to the noise due to imputation as the number of examples with missing data increases with the increase in the number of years of patient history considered. Since Multitask-Temporal does not perform missing data imputation, its performance is more robust.

Figure 5-6 shows that the performance of Multitask-Temporal significantly outperforms SingleTask-Temporal. This emphasizes the advantage gained by learning the tasks jointly instead of learning them independently. By imposing the temporal constraint, Multitask-Temporal is able to reduce overfitting and generalize better on examples in the holdout set.

5.5.4 Visualization of Temporal Dynamics of Variables

Our results demonstrate that the Multitask-Temporal approach is able to capture the temporal dynamics of the variables in longitudinal EHR data to achieve a risk stratification model that is statistically more accurate than models that ignore temporal structure for predicting short-term progression of kidney function loss. In this subsection we examine one aspect of the temporal dynamics: how the relative contributions of individual variables change across windows.

The analysis of the weights assigned by a logistic regression model sheds light on the relative contributions of each variable to the regression equation after accounting for the contributions of other variables in the model [29]. Because some variables may be highly correlated with others, the weights do not perfectly capture the independent association of each variable with the outcome. A positive weight means that the presence of the variable increases the likelihood of a positive outcome. A
negative weight suggests that the presence of the variable decreases the likelihood of the outcome. If a weight is close to 0, this suggests that the model does not consider the variable useful for estimating the likelihood.

To analyze the temporal patterns in variable weights, we first compute the normalized weight assigned by the model for a given variable. Given the model associated with a time-window $t$, we compute the normalized weight for variable $v$ by:

$$\text{normWeight}_v = \frac{\text{weight}_v}{\sum_i |\text{weight}_i|}$$

(5.8)

Next, we compute the mean and the variance of the normalized weights across the 100 splits. Figure 5-7 shows the mean normalized weights of variables for each of the 4 time-windows obtained from the Multitask-Temporal model learned using 2 years of patient history for a threshold of 20%. The variables shown are the 15 variables with the most significant positive weights for time-window $t = 1$ excluding variables that are derived from a patient’s eGFR.

Many of the variables shown in the Figure 5.8 are known risk factors of kidney function. Past studies have shown that African Americans are reported to have a faster progression rate than non-African Americans [19]. Diabetes (indicated by ICD-9 250.50 Diabetes with ophthalmic manifestations) is also a leading risk factor that for kidney dysfunction [19]. Liver damage (indicated by ICD-9 571.5 Cirrhosis of liver without mention of alcohol) and cancer (indicated by ICD-9 155.0 Malignant neoplasm of liver) has also been linked with renal dysfunction [14, 34].

In Figure 5-7, we observe that the normalized weights for a variable can vary across time-windows. In other words, the relative importance of variables can change over time. For example, the normalized weight of ICD-9 182.0 declines over time whereas that of ICD-9 571.5 increases. ICD-9 182.0 represents malignant neoplasm of the uterus. The normalized weights suggest that a more recent diagnosis of ICD-9 182.0 is associated with a faster rate of CKD progression. On the other hand, patients who have had cirrhosis of the liver (ICD-9 571.5) for a longer period of time are associated with worse outcomes than patients with a recent diagnosis of
Figure 5-7: Temporal patterns in normalized variable weights of the 15 variables of the Multitask-Temporal model for threshold = 20%. The bar shows color for the normalized variable weight in the interval $[-5, 25] \times 10^{-4}$.  

111
cirrhosis. This variation in relative importance of variables is one of the reasons why the Non-Temporal approach that allows only a single weight for a variable over multiple windows does not perform as well as Multitask-Temporal in Figure 5-4.

5.6 Summary and Conclusions

In this chapter we explored different methods to use longitudinal EHR data to developing predictive models. We proposed and investigated a novel multitask-based approach to capture temporal information in EHR data.

Using our approach, we demonstrated the advantage of using temporal information for predicting progression in patients with early stage CKD. Specifically, our proposed approach was able to to boost sensitivity from 37.5% to 55.1% (for a precision of ≈ 50%) when identifying patients at high risk of rapid kidney deterioration, than the method that uses only the most recent patient information.

Our results also showed that the choice of approach used to exploit temporal information is also critical. Our multitask based approach was more successful at dealing with the challenges of missing data, different lengths patient history and high dimensionality than methods that use data imputation to handle missing data or learn the tasks independently (instead of jointly).
Chapter 6

Summary and Conclusion

In this thesis we showed how the structural relationships between clinical variables and temporal information in longitudinal EHR data can be exploited to improve performance of clinical predictive models. We presented methods that can exploit such information while addressing the challenges, such as high dimensionality and missing data, that exist when working with clinical datasets. We presented our work in the context of developing risk stratification models for chronic diseases.

In Chapter 3 we showed the advantage of leveraging structural relationships between ICD-9 codes, a hierarchical coding system used to record a patient’s diagnoses and procedures. We investigated different methods to leverage structural relationships in ICD-9 codes when learning clinical predictive models. Specifically, we looked at methods that leverage hierarchy by 1) incorporating the structural information during feature construction, 2) using a learning algorithm (overlapping group lasso) that considers the structure in features when learning a model, and 3) doing both. We presented a novel probability-based feature construction approach that is simple and effective in exploiting the structural relationships between the ICD-9 codes. We showed that methods that leverage hierarchy in the ICD-9 codes can yield more accurate predictive models than methods that ignore hierarchy. Using our novel feature representation to leverage hierarchy for different types of clinical variables has the promise of significantly reducing feature dimensionality and learning more accurate models when the size of the data set is small.
In Chapter 4, we performed a simulation study to better understand the different properties of data that affect the magnitude of improvement that can be achieved by leveraging hierarchy. We generated simulation data that is a simple approximation of a real clinical dataset. Specifically, we generated a data set such that 1) the variables in the data share structural relationship with each other according to a hierarchy tree similar to the ICD-9 hierarchy and 2) the label (outcome) of a datum depends only on the leaf nodes of the tree, i.e., the nodes that contains the most specific information. In our experiments, we varied the observed properties of the data, such as number of training examples and the distribution of examples across different nodes in the hierarchy tree. We demonstrated that leveraging hierarchy can significantly improve performance when the number of training examples is small and when the examples are distributed across nodes in all levels of the hierarchy. Using a KL-divergence based distance metric, we also demonstrated that the distance between the mass distribution of the training and test set is highly correlated with the magnitude of improvement (correlation coefficient of $\approx 0.99$ with a p-value $< 10^{-7}$). We also analyzed the effect of $\alpha$, a measure that quantifies the relationship between the probability of the label given a node and the probability of the label given its sibling node in the tree. We showed that the value of $\alpha$ that yielded the maximum improvement is a function of the number of training examples used to learn the model and the distance between the mass distribution of the training and the test set. Overall, we demonstrated that while the magnitude of improvement varies depending on the different properties of the training and the test data, leveraging hierarchy either performs comparably or outperforms the commonly used approach that ignores structural relationships between the nodes.

We also performed experiments to illustrate the significance of leveraging hierarchy when hospitals transition from ICD-9 (the less specific coding system) to ICD-10 (the more specific coding system) codes. Using clinical data for the task of predicting incident heart failure, we mimicked the transition using ICD9-original and ICD9-truncated. Our results show that by leveraging hierarchy, our proposed feature representation can yield models with significant performance gain under such a
scenario than models learned using the commonly used feature representation that ignores hierarchy.

Our simulations provided useful insights in understanding when leveraging hierarchy is advantageous. However, properties of real clinical datasets are significantly more complex than the properties of our simulated data. The key differences between real data and our simulation data include 1) the variables are related by a forest of trees instead of a single hierarchy tree and 2) a example can be associated with multiple leaf nodes resulting in a complex relationship between the variables and the label. A simulation study that accounts for such differences would yield a better understanding of the various factors affect the impact of leveraging hierarchy.

In Chapter 5, we proposed a novel approach to using temporal information in longitudinal EHR data. We segmented the past medical history of a patient into time-windows and extracted predictors in each time window separately. We considered each segment as a separate task. Using a multitask learning approach, we learned all the tasks jointly using a temporal constraint. Specifically, the constraints encouraged the effect of a given predictor on outcome to be similar for consecutive segments. We showed that learning the tasks jointly exploits the relatedness between tasks to yield a more accurate model than when the tasks are learned independently. We also demonstrated the ability of the multitask approach to handle missing data successfully thus reducing the noisy effects of data imputation. Using our proposed approach, we demonstrated the advantage of using temporal information for identifying early stage CKD patients at high risk of progression. Specifically, our proposed approach was able to boost sensitivity from 37.5% to 55.1% (for a precision of ≈ 50%) relative to the method that uses only the most recent patient information.

The results and analyses presented in this dissertation highlight the importance of exploiting structural relationships between variables and incorporating temporal information to develop accurate predictive models for clinical applications. Our work also emphasize the need to develop algorithms that can successfully incorporate such information while addressing challenges presented by EHR data.

The techniques presented in this thesis were developed in the context of clinical

115
applications. However, the insights and the methods discussed can be applicable to other applications that present similar data challenges. E.g., product categories and ZIP codes are high cardinality categorical variables that exhibit hierarchical relationships. Exploiting temporal information is important in other applications such as finance and climate modeling.

In summary, this thesis represents an attempt to develop techniques that exploit properties of existing data to develop robust predictive models while effectively handling the challenges put forward by the data.
Appendix A

The Mount Sinai Data Warehouse Details

The Mount Sinai Data Warehouse (MSDW) contains patient demographics, laboratory results, diagnoses, procedures, vital signs and medications, including others such as caregiver information and the facility where the patient was attended to. Every piece of patient information recorded in the database is referred to as a fact. Each fact is identified by a unique FACT_KEY in the FACT table and has a value associated with it. Each fact also has several dimensions, where a dimension contains information about the fact and its value. Figure A-1 shows some of the relevant dimensions associated with two different types of facts. Not all facts have all dimensions, e.g., the fact that corresponds to a medication order does not have a dimension describing where the patient is located in the facility. The information about the type of the fact is stored in the D_METADATA dimension. We discuss this in more detail in the following paragraphs.

Table A.1 shows the dimensions that we use and their descriptions. Each dimension is stored in a separate table in the database.

Figure A-2 shows two facts from the FACT table in the database with some of its dimensions. The value of the KEY in each dimension links the fact to the table associated with each dimension. The first fact contains diagnoses and procedure information about a patient during an encounter at a certain date. The key value
of 2 for $\text{MATERIAL\_GROUP\_KEY}$ indicates that this fact does not have any information in that dimension. The $\text{META\_DATA\_KEY}$ (not shown in the figure) describes this fact as an Encounter Summary.

The second fact in Figure A-2 corresponds to a fact that contains a diastolic blood pressure measurement and $\text{VALUE}$ of the fact is the blood pressure value. The $\text{META\_DATA\_KEY}$ (not shown in the figure) describes this fact as a Vital Sign. We discuss how we extract such numerical predictors in more detail in the following paragraphs.

We now discuss how patient data is stored in the tables corresponding to the some of the important dimensions of a fact.

1. Patient demographics: $\text{D\_PERSON}$ stores demographic information such as date of birth, gender and race. Each row has a unique $\text{PATIENT\_KEY}$. 

Figure A-1: Examples illustrating two different types of facts and their corresponding dimensions.
<table>
<thead>
<tr>
<th>Dimension (Table)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D_CALENDAR</td>
<td>Date information</td>
</tr>
<tr>
<td>D_PERSON</td>
<td>Patient demographics including age, gender, race</td>
</tr>
<tr>
<td>D_ENCOUNTER</td>
<td>Encounters between a patient and the hospital such as outpatient and inpatient information</td>
</tr>
<tr>
<td>D_DIAGNOSIS</td>
<td>Clinically coded diagnoses and problems</td>
</tr>
<tr>
<td>D_PROCEDURE</td>
<td>Procedures, laboratory tests and vital signs</td>
</tr>
<tr>
<td>D_MATERIAL</td>
<td>Material goods, including drugs, implants or equipment</td>
</tr>
<tr>
<td>D_UNIT_OF_MEASURE</td>
<td>Unit of measurement for numerical facts e.g., \textit{mm Hg}</td>
</tr>
<tr>
<td>D_METADATA</td>
<td>Description of the type of the data and the source of the data e.g., Epic or Cerner</td>
</tr>
<tr>
<td>D_OPERATION</td>
<td>Identifiers that tie multiple facts together</td>
</tr>
</tbody>
</table>

Table A.1: The description of the dimensions of a fact that we use.

<table>
<thead>
<tr>
<th>FACT_KEY</th>
<th>PERSON_KEY</th>
<th>CALENDAR_KEY</th>
<th>ENCOUNTER_KEY</th>
<th>DIAGNOSIS_GROUP_KEY</th>
<th>PROCEDURE_GROUP_KEY</th>
<th>MATERIAL_GROUP_KEY</th>
<th>UOM_KEY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1019576789</td>
<td>5629718</td>
<td>1282</td>
<td>14166370</td>
<td>237458</td>
<td>2202925</td>
<td>2</td>
<td>267 &lt;empty&gt;</td>
<td></td>
</tr>
<tr>
<td>1019530269</td>
<td>5629718</td>
<td>1282</td>
<td>14171238</td>
<td>2</td>
<td>65931</td>
<td>2</td>
<td>329.73</td>
<td></td>
</tr>
</tbody>
</table>

Figure A-2: Two facts from the Mount Sinai Data Warehouse.

Each patient is identified by a unique PERSON_CONTROL_KEY. The same patient can have multiple PATIENT_KEYs. This is because any updates or corrections to existing patient information is entered with a new PATIENT_KEY. The most recent information for a patient has the field ACTIVE_FLAG set to Y (N otherwise). We always use the most updated demographic information about the patient in all our experiments.

2. **Laboratory results and vital signs**: As described before, the numerical value of a laboratory result or a vital sign is stored in the VALUE field of the FACT table. The META_DATA_KEY links the fact to D_METADATA table that describes whether the value is either a \textit{Lab Test} or a \textit{Vital Sign}. The PROCEDURE_GROUP_KEY links the fact to D_PROCEDURE table that describes the type of the lab test or the vital sign. Figure A-3 shows the description of the procedure, the metadata (given by the fields LEVEL_2_EVENT, LEVEL_3_ACTION and LEVEL_4_FIELD) and the unit of measure for the second fact shown in Figure A-2. The unit of measure is obtained by linking
the fact to the D_UNIT_OF_MEASURE table using the UOM_KEY.

Figure A-3: Procedure, unit of measure and metadata description for the second fact shown in Figure A-2. The field LEVEL_2_EVENT, LEVEL_3_ACTION and LEVEL_4_FIELD contain the metadata information.

In addition to the description stating whether a numerical fact is a vital sign or a lab test, the metadata also contains information about whether the result is a Preliminary Result or a Final result. A numerical fact can be a preliminary result. Once the final result is obtained, a new numerical fact is recorded. We only extract numerical facts that are described as final results.

When extracting a laboratory result or a vital sign, it is important to take into consideration that there might be a number of different procedure descriptions (with small variations) that describe the same lab test or a vital sign. E.g., Table A.2 show the different procedure descriptions that store creatinine values.

<table>
<thead>
<tr>
<th>Procedure description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATININE (POCT)</td>
</tr>
<tr>
<td>CREATININE CLEARANCE</td>
</tr>
<tr>
<td>CREATININE</td>
</tr>
<tr>
<td>WB CREATININE - VEN</td>
</tr>
<tr>
<td>WB CREATININE - ART</td>
</tr>
</tbody>
</table>

Table A.2: Different procedure descriptions used to describe creatinine values stored in the database.

3. Diagnoses and Procedures: Patient diagnoses and procedures are recorded in tables D_DIAGNOSIS and D_PROCEDURE respectively. We extract the procedures and diagnosis encoded using the ICD-9 codes, i.e., field CONTEXT_NAME = ICD-9. While ICD-9 codes are primarily used for billing purposes, they contain important patient information. Each fact can be associated with
multiple ICD-9 codes. Therefore, the DIAGNOSIS_GROUP_KEY and PROCEDURE_GROUP_KEY for a fact can have multiple DIAGNOSIS_KEYs and PROCEDURE_KEYs respectively.

4. **Medications** : When a medication is prescribed/administered to a patient, the corresponding fact will have a MATERIAL_GROUP_KEY that links the fact to a row in the D_MATERIAL table where the field MATERIAL_TYPE = *Drug*. The name of the medication is stored as the MATERIAL_NAME. The material name can either contain a brand name or the ingredient contained in the medication, along with other information such as dosage. Figure A-4 shows an excerpt from the table.

<table>
<thead>
<tr>
<th>MATERIAL_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MVII, ADULT NO.1 WITH VIT K 3,300 UNIT-150 MCG/10 ML IV</td>
</tr>
<tr>
<td>2 LISINOPRIL 20 MG TABLET</td>
</tr>
<tr>
<td>3 FLUTICASONE 110 MCG/ACTUATION AEROSOL INHALER</td>
</tr>
<tr>
<td>4 ONDANSETRON 4 MG DISINTEGRATING TABLET</td>
</tr>
<tr>
<td>5 METOPROLOL TARTRATE 50 MG TABLET</td>
</tr>
<tr>
<td>6 INSULIN NEEDLES (DISPOSABLE)</td>
</tr>
<tr>
<td>7 FERROUS SULFATE 325 MG (65 MG IRON) TABLET</td>
</tr>
<tr>
<td>8 PNV (WITHOUT CALCIUM)-IRON FUM-FOLIC ACID 27 MG-1 MG TABLET</td>
</tr>
<tr>
<td>9 INSULIN GLARGINE 100 UNIT/ML SUBQ CARTRIDGE</td>
</tr>
<tr>
<td>10 BLOOD SUGAR DIAGNOSTIC STRIPS</td>
</tr>
<tr>
<td>11 GABAPENTIN 600 MG TABLET</td>
</tr>
<tr>
<td>12 HVCODAN 1.5 MG-5 MG TABLET</td>
</tr>
<tr>
<td>13 METFORMIN 1,000 MG TABLET</td>
</tr>
</tbody>
</table>

Figure A-4: An excerpt from the D_MATERIAL table showing the MATERIAL_NAME for some of the medications.

For our work, we extract medications that are used for the treatment of hypertension, hyperlipidemia and diabetics, and medications that are known to be nephrotoxic. When extracting the medications we disregard the route, dosage and the strength of the medication.
We only extract prescription medications for our work since we are interested in medications that are used to treat the patient’s chronic condition. We identify the prescription medication using the metadata LEVEL2_EVENT_NAME = Prescription. A single prescription has multiple facts associated with it. The start date and the end date of a prescription appear as two separate facts in the FACT table with metadata LEVEL4_FIELD_NAME = Start Date and End Date respectively. The actual start date and end date are recorded as the value of the facts. The two facts are tied together via a common OPERATION_KEY in the FACT table.

Unfortunately, the prescription medication data in the database is noisy. First, although there is a start date and an end date associated with a prescription, there is no way to find out if the patient actually takes the prescribed medication during that period. Second, if a prescribed medication is canceled, there is not an easy way to keep track of the cancellations. Third, if a patient fills his/her prescription in pharmacies outside the hospital, they are not recorded in the database.
Bibliography


