CONTRIBUTIONS TO MACHINE LEARNING IN BIOMEDICAL INFORMATICS

By

Inci Meliha Baytas

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science – Doctor of Philosophy

2019

ABSTRACT

CONTRIBUTIONS TO MACHINE LEARNING IN BIOMEDICAL INFORMATICS

By

Inci Meliha Baytas

With innovations in digital data acquisition devices and increased memory capacity, virtually all commercial and scientific domains have been witnessing an exponential growth in the amount of data they can collect. For instance, healthcare is experiencing a tremendous growth in digital patient information due to the high adaptation rate of electronic health record systems in hospitals. The abundance of data offers many opportunities to develop robust and versatile systems, as long as the underlying salient information in data can be captured. On the other hand, today's data, often named *big data*, is challenging to analyze due to its large scale and high complexity. For this reason, efficient data-driven techniques are necessary to extract and utilize the valuable information in the data. The field of machine learning essentially develops such techniques to learn effective models directly from the data. Machine learning models have been successfully employed to solve complicated real world problems. However, the big data concept has numerous properties that pose additional challenges in algorithm development. Namely, high dimensionality, class membership imbalance, non-linearity, distributed data, heterogeneity, and temporal nature are some of the big data characteristics that machine learning must address.

Biomedical informatics is an interdisciplinary domain where machine learning techniques are used to analyze electronic health records (EHRs). EHR comprises digital patient data with various modalities and depicts an instance of big data. For this reason, analysis of digital patient data is quite challenging although it provides a rich source for clinical research. While the scale of EHR data used in clinical research might not be huge compared to the other domains, such as social media, it is still not feasible for physicians to analyze and interpret longitudinal and heterogeneous data of thousands of patients. Therefore, computational approaches and graphical tools to assist physicians in summarizing the underlying clinical patterns of the EHRs are necessary. The field of biomedical informatics employs machine learning and data mining approaches to provide the essential computational techniques to analyze and interpret complex healthcare data to assist physicians in patient diagnosis and treatment.

In this thesis, we propose and develop machine learning algorithms, motivated by prevalent biomedical informatics tasks, to analyze the EHRs. Specifically, we make the following contributions: (i) A convex sparse principal component analysis approach along with variance reduced stochastic proximal gradient descent is proposed for the patient phenotyping task, which is defined as finding clinical representations for patient groups sharing the same set of diseases. (ii) An asynchronous distributed multi-task learning method is introduced to learn predictive models for distributed EHRs. (iii) A modified long-short term memory (LSTM) architecture is designed for the patient subtyping task, where the goal is to cluster patients based on similar progression pathways. The proposed LSTM architecture, T-LSTM, performs a subspace decomposition on the cell memory such that the short term effect in the previous memory is discounted based on the length of the time gap. (iv) An alternative approach to T-LSTM model is proposed with a decoupled memory to capture the short and long term changes. The proposed model, decoupled memory gated recurrent network (DM-GRN), is designed to learn two types of memories focusing on different components of the time series data. In this study, in addition to the healthcare applications, behavior of the proposed model is investigated for traffic speed prediction problem to illustrate its generalization ability. In summary, the aforementioned machine learning approaches have been developed to address complex characteristics of electronic health records in routine biomedical informatics tasks such as computational patient phenotyping and patient subtyping. Proposed models are also applicable to different domains with similar data characteristics as EHRs.

To my parents

ACKNOWLEDGMENTS

As I am approaching to the end of my Ph.D., I would like to express my appreciation and gratitude for special people, who were with me in this journey. I am so grateful to be a Ph.D. student of Prof. Anil K. Jain. Since the beginning of my Ph.D., I always felt his endless support and encouragement. His wisdom and positive approach made me feel more confident and peaceful during difficult times of Ph.D. His reference opened many doors, including the faculty position I have been dreaming. I will always be thankful to Dr. Jain for everything he has done for my training and future. I am also grateful to have an opportunity to work with my co-advisor Dr. Jiayu Zhou. I would like to thank him for being such a great teacher and mentor. I learned a lot about conducting research, writing papers and presenting my work from him. I am so appreciative that my both advisors provided me several opportunities to improve my teaching skills and they made sure I have everything I need for my future career goals. I also would like to use this opportunity to thank Prof. Bilge Gunsel for believing in me and encouraging me. Her endless support and reference have a big role in earning this degree today.

I would like to thank my beloved parents for encouraging me to pursue this degree. Without their love, support, understanding, trust, encouragement, and friendship, I would not be who I am today. Throughout my life, they never stop believing in me and this always gives me confidence and strength to move on. They have always been a perfect role model with their hard-work and dedication. They always inspire me with their tenacity and determination. I cannot thank my parents, my eternal best friends, enough for loving me, guiding me and giving me such a warm family I can always hold on to. I also would like to thank my cousin Aydeniz for never leaving me alone and giving me her endless support, friendship, and sisterhood. She has been a tremendous support mechanism for me during my Ph.D.

Being in graduate school was very challenging from time to time and making beautiful friendships with people who can understand me was so comforting. I would like to thank Soweon and Serhat for answering my questions before I joined the lab. It has been a pleasure to share this journey with my lab mates Debayan, Sixue, Josh, Yichun, Luan, Tarang, Charles, Lacey, Sunpreet, and Keyur. I will miss the time I spent with my dear friends Sudipta, Sixue, Nupur and Vidhya. A very special thanks to Debayan and Sudipta for being such fun, supportive, caring and loving friends; their friendship supported me to the core. Last but not least, I would like to thank my life long friends in Turkey; Unzile, Emine, and Cigdem. After all those years and even we were far from each other, they never stopped being such beautiful sisters to me.

Finally, I would like to thank CSE department; Prof. Eric Torng for his guidance about Ph.D. program, Prof. Abdol Esfahanian for providing me a teaching opportunity, and Brenda Hodge and Steve Smith for their help with enrollments and reimbursements. I also would like to express my gratitude for my committee members; Prof. Pang-Ning Tan, Prof. Xiaoming Liu, and Prof. Selin Aviyente for their time and valuable feedback.

TABLE OF CONTENTS

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Digital data has been an important driving force of technology and the economy since the 1990s. Development of devices for the capture and storage of digital data has rapidly increased the amount of information collected in various media, including text, audio, image, and video. The digital age is witnessing an ever-growing information explosion, where the size of the digital universe is estimated to double every two years; a 50-fold growth from 2010 to 2020 [2]. Digital data is very valuable for scientific research and industry as long as insightful information can be extracted. On the other hand, comprehensive analysis of today's data, often named *big data*, is an ongoing challenge. IBM data scientists defined big data with four properties: volume (scale of data), variety (different forms of data), velocity (temporal nature of data) and veracity (uncertainty of data) [1]. As a result, big data is challenging not only because of its large scale, but also its high complexity. Primary data types and their major challenges are summarized for different domains in Table 1.1. As presented in the table, big data is characterized by a high dimensional, non-linear, heterogeneous, distributed, and temporal nature. However, big data offers numerous opportunities for machine learning researchers in different domains, such as healthcare. The sheer volume of digital health information is projected to grow even faster than other sectors over the next seven years, according to the report by the International Data Corporation (IDC) [70]. Therefore, with-

[1] http://www.ibmbigdatahub.com/infographic/four-vs-big-data

out efficient and comprehensive data analysis techniques, it is not feasible for physicians to draw clinical conclusions from the growing amount of digitized patient data.

Biomedical informatics is an interdisciplinary domain whose goal is to improve the patient care using efficient data-driven techniques. In particular, biomedical informatics leverages machine learning and data mining techniques to enable preventive and personalized medicine, reduce healthcare costs, and facilitate clinical research. Digital health records, as big data in general, are heterogeneous, non-linear [2] and temporal. Thus, biomedical informatics has become an evolving domain for machine learning research.

Machine learning has become an inevitable tool to solve challenging real world problems due to its capability to infer underlying patterns in data. As the data complexity increases, machine learning techniques need to become more sophisticated. The chronological developments in machine learning are summarized in Figure 1.1. After Hinton's introduction of deep learning in 2006, powerful learning models have emerged to solve complicated real world problems, such as disease detection, temporal clinical event prediction, and concept embedding. In this thesis, we developed machine learning and deep learning models to address some of the challenging biomedical informatics tasks, such as computational phenotyping and patient subtyping. The proposed methods, in particular, focus on high dimensional, non-linear, distributed, and temporal aspects of the digital patient data. In the subsequent sections, a brief introduction to biomedical informatics, challenges, machine learning solutions, and the main contributions of the thesis are summarized.

1.1 Biomedical Informatics

Biomedical informatics is an evolving domain for machine learning researchers ever since hospitals started storing digital patient records. The informatics component focuses on the acquisition, storage and the retrieval of the health information, whereas the biomedical component refers to the medical tasks using the patient data [16]. The essential purpose of the biomedical informatics is to

[2] Here, non-linear refers to non-linear interactions between the attributes (e.g., diagnoses, medication, demographics) provided in patient data.

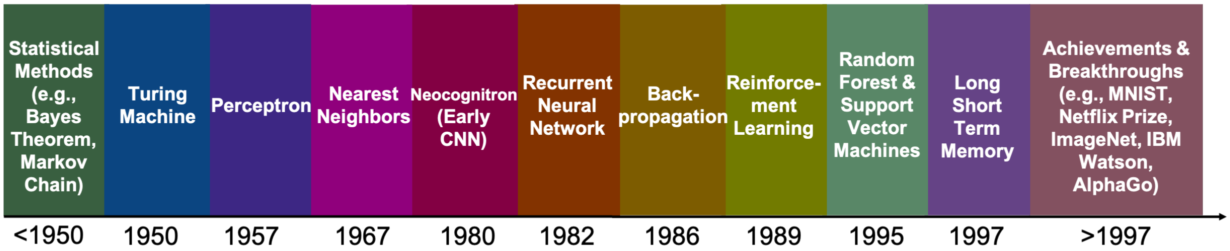


Figure 1.1 Evolution of machine learning. Since 1997, machine learning domain has been witnessing groundbreaking developments and breakthroughs. Large datasets and competitions, such as MNIST, ImageNet, Netflix, and Kaggle, have become the catalyst for many state-of-the-art algorithms.

convert the vast amount of digital health information into relevant knowledge using computational techniques [6]. The digital patient data is provided by Electronic Health Records (EHRs), which represent longitudinal patient information, including diagnoses, medication, lab results, and medical images. EHRs follow a similar growth trend as big data; the EHR market is expected to reach 33.41 billion US dollars by 2025 [100]. Adoption rate of EHR systems is also increasing every year. The change in the percentage of physicians using EHR systems between 2004-2015 [90] is given in Figure 1.2. Consequently, an exponential growth in EHR data volume is observed and this trend is predicted to continue through 2020 as shown in Figure 1.3 [45].

EHRs provide valuable and diverse information about patients and physicians. Biomedical informatics studies systematic techniques to extract the salient information which EHR data can offer. Adoption of EHR systems and the data analytics techniques improve different aspects of healthcare. One important aspect is the personalized healthcare, which requires analyzing trends in different patient populations and identification of patterns in patient outcomes. A vast number of EHRs facilitate learning the common trends in a patient cohort, and consequently, enable prediction of health outcomes. One of the important benefits of using data analytics tools in healthcare is the reduction in healthcare costs. Currently, the healthcare expenses are approximately 18% of GDP (Gross Domestic Product), which corresponds to nearly 600 billion dollars in the United States [40]. In particular, the predictive analysis using EHRs plays an important role to increase the performance and the efficiency of healthcare services, and consequently assists in optimizing

Table 1.1 Different domains offer different challenges pertaining to big data. Scale is only one of the factors that increases the complexity of the traditional machine learning approaches. In some applications, the amount of data used for algorithm development may not be large, but other additional challenges are introduced.

Domain	**Data**	**Major Challenges**
Healthcare	Electronic Health Records	• Large scale (e.g., data on 53K patient admissions in MIMIC-III [66]) • High dimensional (e.g., thousands of ICD-9 codes [113]) • Heterogeneous (universal codes, medical images, hand-written notes, etc.) • Distributed • Temporal • Non-linear interactions • Class imbalance
Computer Vision	Images, Videos	• Large scale (e.g., IMAGENET with 14M images [102]) • Temporal (videos) • Class imbalance
E-Commerce	Behavioral data (e.g., clicks, transactions)	• Large scale (e.g., 2M events [3]) • Heterogeneous • Distributed • Temporal
Finance	Financial and stock market data (e.g., transactions, portfolios)	• Large scale (e.g., data on 542K transactions [25]) • Temporal
Traffic Management	GPS, surveillance data	• Large scale (e.g., 1.3M taxi trips in NYC between January 2009 and June 2015 [4]) • Distributed • Temporal

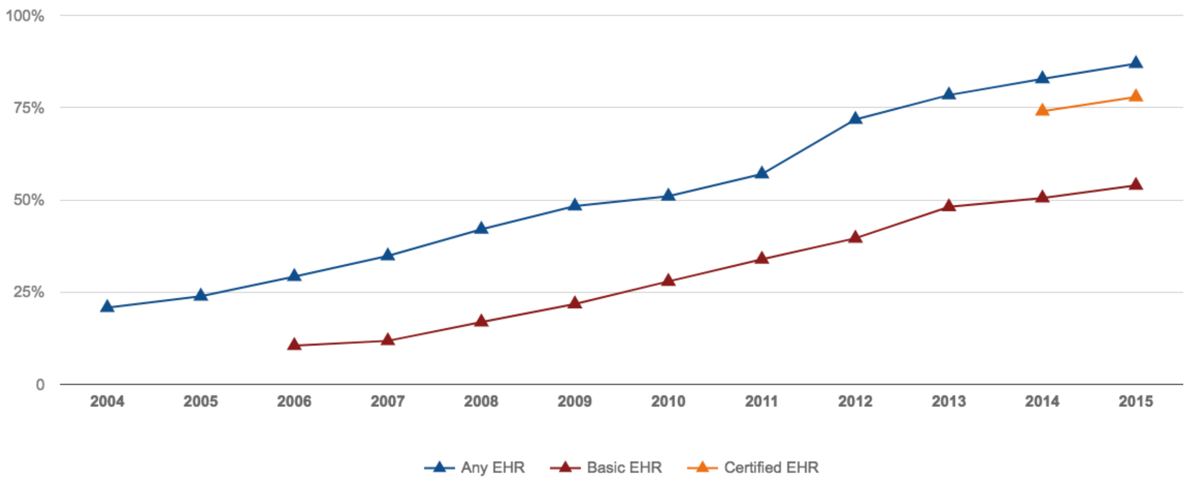


Figure 1.2 Percentage of office-based physicians using EHR. Adoption of EHRs doubled between 2008 and 2015 [90]. Basic EHR requires EHR systems to have at least minimal information, such as patient demographics, medication list, discharge summary, and lab reports. Whereas certified EHR systems comprise detailed clinical information, different functionalities, and security requirements.

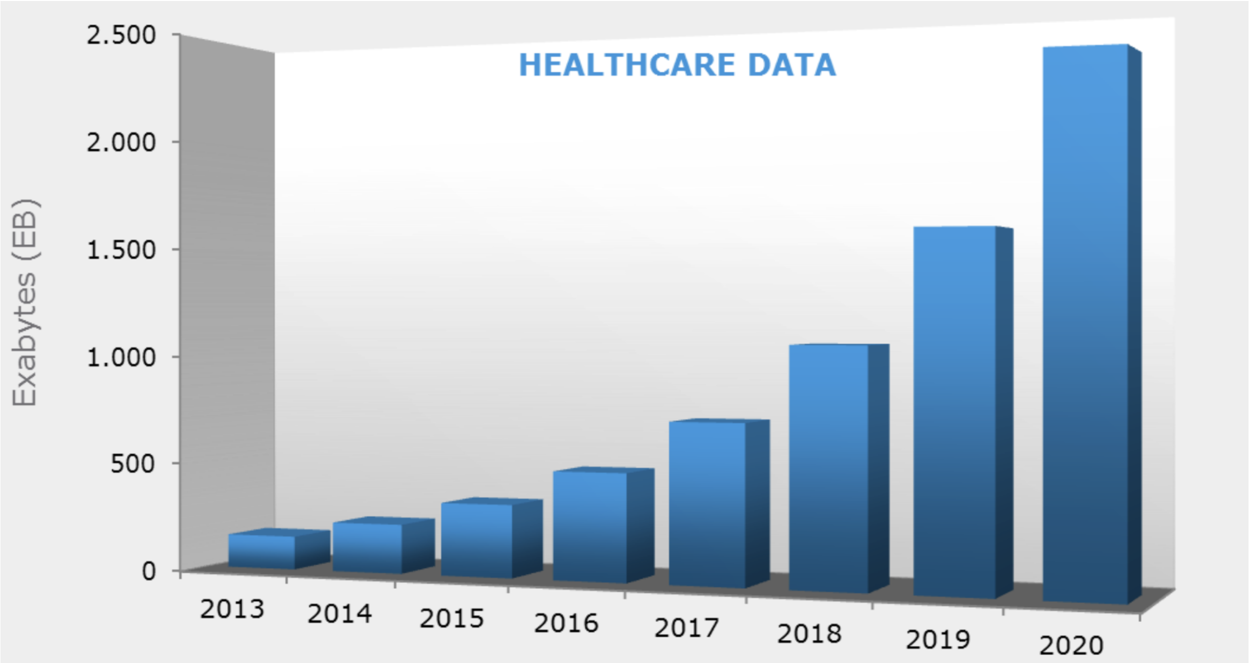


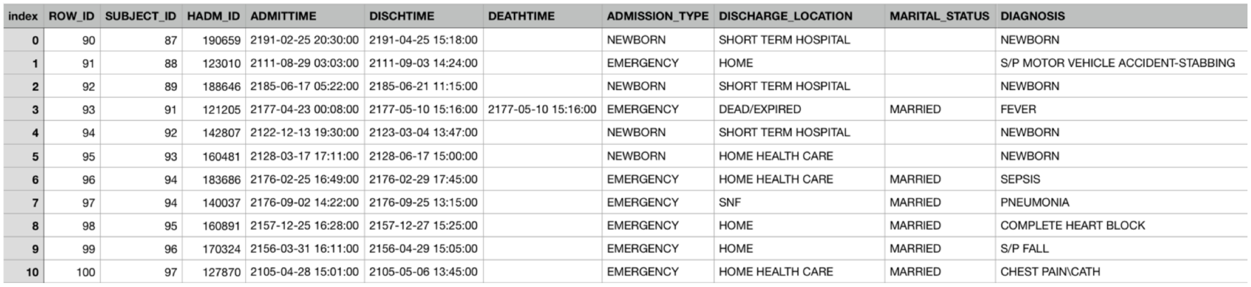
Figure 1.3 Healthcare data is growing rapidly. A 48% annual growth rate is expected leading to $2,314$ Exabytes of data in 2020 [45].

healthcare costs. In summary, biomedical informatics research aims to develop computational techniques to improve diagnosis and prognosis of diseases. In addition, clinical treatment and medical research are also enhanced by the biomedical informatics techniques [6]. In the following section, EHR data characteristics and major challenges encountered in biomedical informatics are discussed.

1.1.1 Data

Before the adoption of EHRs, clinical information used to be recorded in paper format and transferred between clinics physically [112]. Paper based patient records were not effective for clinical research since extracting patient statistics and the patterns in healthcare outputs of a large patient cohort was not attainable for physicians. As a result, the clinical treatments used to focus more on the cure rather than predicting and preventing possible risks [99]. After EHR systems are launched in hospitals, quality of the patient records is improved and the scope of the patient's medical history is expanded. Various types of patient information, such as diagnoses, lab results, medical imaging, medication, doctor's handwritten notes, and most importantly time stamp of every clinical outcome can be found in EHRs. As a result of the availability of comprehensive patient information, clinical decision making has been enriched and become more proactive.

In a standard EHR database, multiple tables are constructed to store different types of information. As an example, in Figures 1.4 and 1.5, several rows of admission, diagnosis, lab result, and patient demographics tables of a publicly available critical care database, named MIMIC-III [66], are shown. In a typical patient table, true identity of the individuals is never shared, but assigned unique IDs for each patient, demographics, dates of admission and discharge are stored. Dates are also encrypted to avoid releasing the actual time that a patient visited the hospital. Since it provides the time stamps, the patient table is necessary for the temporal analysis of EHRs. Another very important piece of information found in EHRs is the diagnoses. At the end of the patient visit, final diagnoses, usually represented by a universal code (e.g., ICD-9 [113]), are recorded for insurance purposes. Patients may have more than one code at the end of each admission, depending on their



index	ROW_ID	SUBJECT_ID	HADM_ID	ADMITTIME	DISCHTIME	DEATHTIME	ADMISSION_TYPE	DISCHARGE_LOCATION	MARITAL_STATUS	DIAGNOSIS
0	90	87	190659	2191-02-25 20:30:00	2191-04-25 15:18:00		NEWBORN	SHORT TERM HOSPITAL		NEWBORN
1	91	88	123010	2111-08-29 03:03:00	2111-09-03 14:24:00		EMERGENCY	HOME		S/P MOTOR VEHICLE ACCIDENT-STABBING
2	92	89	188646	2185-06-17 05:22:00	2185-06-21 11:15:00		NEWBORN	SHORT TERM HOSPITAL		NEWBORN
3	93	91	121205	2177-04-23 00:08:00	2177-05-10 15:16:00	2177-05-10 15:16:00	EMERGENCY	DEAD/EXPIRED	MARRIED	FEVER
4	94	92	142807	2122-12-13 19:30:00	2123-03-04 13:47:00		NEWBORN	SHORT TERM HOSPITAL		NEWBORN
5	95	93	160481	2128-03-17 17:11:00	2128-06-17 15:00:00		NEWBORN	HOME HEALTH CARE		NEWBORN
6	96	94	183686	2176-02-25 16:49:00	2176-02-29 17:45:00		EMERGENCY	HOME HEALTH CARE	MARRIED	SEPSIS
7	97	94	140037	2176-09-02 14:22:00	2176-09-25 13:15:00		EMERGENCY	SNF	MARRIED	PNEUMONIA
8	98	95	160891	2157-12-25 16:28:00	2157-12-27 15:25:00		EMERGENCY	HOME	MARRIED	COMPLETE HEART BLOCK
9	99	96	170324	2156-03-31 16:11:00	2156-04-29 15:05:00		EMERGENCY	HOME	MARRIED	S/P FALL
10	100	97	127870	2105-04-28 15:01:00	2105-05-06 13:45:00		EMERGENCY	HOME HEALTH CARE	MARRIED	CHEST PAIN\CATH

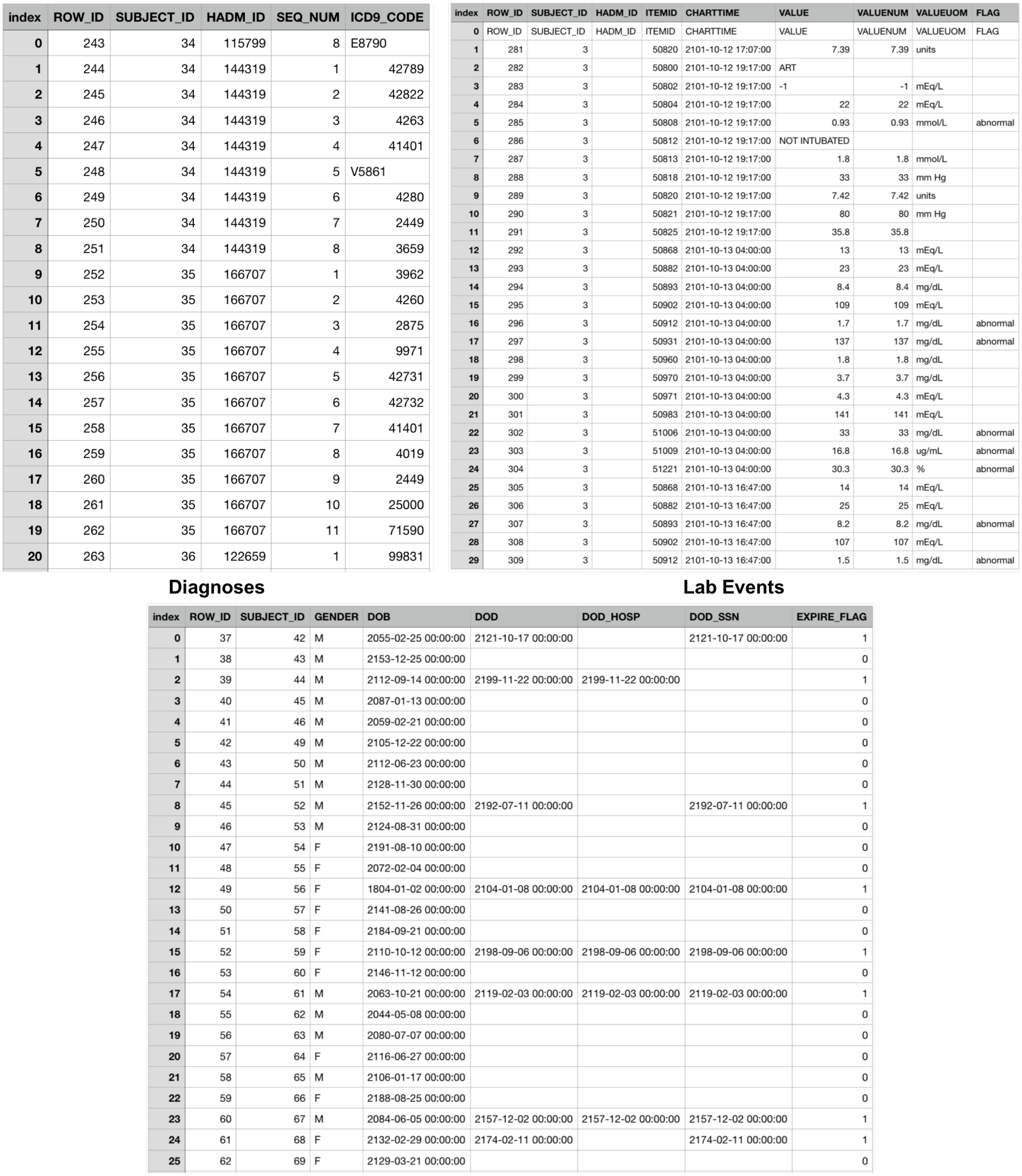
Figure 1.4 Several rows of an admission table in MIMIC-III [66] database. Admission, discharge dates, diagnoses, and subject ID are some of the major attributes for biomedical informatics applications.

conditions. Diagnosis information is very useful when designing predictive biomedical informatics tasks, such as disease prediction. Other clinical details, such as medication, lab results, and procedures, are often encoded in different tables.

Patient data is challenging to process and analyze since the electronic records have various data types (e.g., numerical, symbolic, ratio, interval, ordinal, nominal) as seen in Figures 1.4 and 1.5. Furthermore, medical images and handwritten notes of physicians can also be available in some EHR databases. In summary, EHRs provide a comprehensive patient information with various modalities. Analysis, fusion and inference of heterogeneous patient data are some of the key factors leading healthcare improvements [60]. On the other hand, due to its diversity in format, type, and context, physicians and clinical researchers cannot directly harness the raw EHR data. For this reason, biomedical informatics tries to leverage data-driven techniques to retrieve the useful knowledge from digital records. However, designing data-driven models to infer the underlying information from EHRs is a complicated task due to the fact that EHR is an instance of big data. In the following sections, major challenges biomedical informatics needs to tackle and the solutions machine learning can offer are discussed.

1.1.2 Major Challenges

The biomedical informatics domain tackles particular challenges stemming from EHR data characteristics. One of the most challenging data characteristic is the scale. Each hospital stores hundreds



index	ROW_ID	SUBJECT_ID	HADM_ID	SEQ_NUM	ICD9_CODE
0	243	34	115799	8	E8790
1	244	34	144319	1	42789
2	245	34	144319	2	42822
3	246	34	144319	3	4263
4	247	34	144319	4	41401
5	248	34	144319	5	V5861
6	249	34	144319	6	4280
7	250	34	144319	7	2449
8	251	34	144319	8	3659
9	252	35	166707	1	3962
10	253	35	166707	2	4260
11	254	35	166707	3	2875
12	255	35	166707	4	9971
13	256	35	166707	5	42731
14	257	35	166707	6	42732
15	258	35	166707	7	41401
16	259	35	166707	8	4019
17	260	35	166707	9	2449
18	261	35	166707	10	25000
19	262	35	166707	11	71590
20	263	36	122659	1	99831

Diagnoses

index	ROW_ID	SUBJECT_ID	HADM_ID	ITEMID	CHARTTIME	VALUE	VALUENUM	VALUEUOM	FLAG
0	ROW_ID	SUBJECT_ID	HADM_ID	ITEMID	CHARTTIME	VALUE	VALUENUM	VALUEUOM	FLAG
1	281	3		50820	2101-10-12 17:07:00	7.39	7.39	units	
2	282	3		50800	2101-10-12 19:17:00	ART			
3	283	3		50802	2101-10-12 19:17:00	-1	-1	mEq/L	
4	284	3		50804	2101-10-12 19:17:00	22	22	mEq/L	
5	285	3		50808	2101-10-12 19:17:00	0.93	0.93	mmol/L	abnormal
6	286	3		50812	2101-10-12 19:17:00	NOT INTUBATED			
7	287	3		50813	2101-10-12 19:17:00	1.8	1.8	mmol/L	
8	288	3		50818	2101-10-12 19:17:00	33	33	mm Hg	
9	289	3		50820	2101-10-12 19:17:00	7.42	7.42	units	
10	290	3		50821	2101-10-12 19:17:00	80	80	mm Hg	
11	291	3		50825	2101-10-12 19:17:00	35.8	35.8		
12	292	3		50868	2101-10-13 04:00:00	13	13	mEq/L	
13	293	3		50882	2101-10-13 04:00:00	23	23	mEq/L	
14	294	3		50893	2101-10-13 04:00:00	8.4	8.4	mg/dL	
15	295	3		50902	2101-10-13 04:00:00	109	109	mEq/L	
16	296	3		50912	2101-10-13 04:00:00	1.7	1.7	mg/dL	abnormal
17	297	3		50931	2101-10-13 04:00:00	137	137	mg/dL	abnormal
18	298	3		50960	2101-10-13 04:00:00	1.8	1.8	mg/dL	
19	299	3		50970	2101-10-13 04:00:00	3.7	3.7	mg/dL	
20	300	3		50971	2101-10-13 04:00:00	4.3	4.3	mEq/L	
21	301	3		50983	2101-10-13 04:00:00	141	141	mEq/L	
22	302	3		51006	2101-10-13 04:00:00	33	33	mg/dL	abnormal
23	303	3		51009	2101-10-13 04:00:00	16.8	16.8	ug/mL	abnormal
24	304	3		51221	2101-10-13 04:00:00	30.3	30.3	%	abnormal
25	305	3		50868	2101-10-13 16:47:00	14	14	mEq/L	
26	306	3		50882	2101-10-13 16:47:00	25	25	mEq/L	
27	307	3		50893	2101-10-13 16:47:00	8.2	8.2	mg/dL	abnormal
28	308	3		50902	2101-10-13 16:47:00	107	107	mEq/L	
29	309	3		50912	2101-10-13 16:47:00	1.5	1.5	mg/dL	abnormal

Lab Events

index	ROW_ID	SUBJECT_ID	GENDER	DOB	DOD	DOD_HOSP	DOD_SSN	EXPIRE_FLAG
0	37	42	M	2055-02-25 00:00:00	2121-10-17 00:00:00		2121-10-17 00:00:00	1
1	38	43	M	2153-12-25 00:00:00				0
2	39	44	M	2112-09-14 00:00:00	2199-11-22 00:00:00	2199-11-22 00:00:00		1
3	40	45	M	2087-01-13 00:00:00				0
4	41	46	M	2059-02-21 00:00:00				0
5	42	49	M	2105-12-22 00:00:00				0
6	43	50	M	2112-06-23 00:00:00				0
7	44	51	M	2128-11-30 00:00:00				0
8	45	52	M	2152-11-26 00:00:00	2192-07-11 00:00:00		2192-07-11 00:00:00	1
9	46	53	M	2124-08-31 00:00:00				0
10	47	54	F	2191-08-10 00:00:00				0
11	48	55	F	2072-02-04 00:00:00				0
12	49	56	F	1804-01-02 00:00:00	2104-01-08 00:00:00	2104-01-08 00:00:00	2104-01-08 00:00:00	1
13	50	57	F	2141-08-26 00:00:00				0
14	51	58	F	2184-09-21 00:00:00				0
15	52	59	F	2110-10-12 00:00:00	2198-09-06 00:00:00	2198-09-06 00:00:00	2198-09-06 00:00:00	1
16	53	60	F	2146-11-12 00:00:00				0
17	54	61	M	2063-10-21 00:00:00	2119-02-03 00:00:00	2119-02-03 00:00:00	2119-02-03 00:00:00	1
18	55	62	M	2044-05-08 00:00:00				0
19	56	63	M	2080-07-07 00:00:00				0
20	57	64	F	2116-06-27 00:00:00				0
21	58	65	M	2106-01-17 00:00:00				0
22	59	66	F	2188-08-25 00:00:00				0
23	60	67	M	2084-06-05 00:00:00	2157-12-02 00:00:00	2157-12-02 00:00:00	2157-12-02 00:00:00	1
24	61	68	F	2132-02-29 00:00:00	2174-02-11 00:00:00		2174-02-11 00:00:00	1
25	62	69	F	2129-03-21 00:00:00				0

Patients

Figure 1.5 Several rows of diagnoses, lab events and patient tables in MIMIC-III [66] database. Subject ID of a patient is generally used to retrieve information from different tables. Diagnoses are represented by ICD-9 codes. Quantitative values of results, time, unit, and type of the lab tests are often stored in lab events table.

of thousands patient records in their databases. Sorting and prioritizing this amount of information cannot be easily handled by conventional data-driven algorithms (e.g., models requiring full gradient descent and data localization). Moreover, to utilize EHRs for algorithm design, rigorous data cleaning and encryption are imperative since patient privacy is an inevitable concern. For these reasons, public healthcare datasets to evaluate the data-driven models are not abundant. Distributed nature of EHR datasets is another issue that often needs to be addressed, especially while designing predictive models. A large proportion of biomedical informatics tasks require predictive analysis, where data diversity plays a crucial role. Each hospital has a similar type of patient information that can serve the same predictive task. In this case, combining EHR datasets collected in different regions into one dataset would enhance the generalization property of the predictive models. However, in practice, patient distribution varies for different locations (e.g., ethnicity, income groups, cultural and dietary practices). This violates the training data points drawn from the same stationary distribution premise of machine learning. As a result, predictive models cannot in general be learned for a combination of all the available EHRs from different hospitals.

Missing values in EHRs and interpretability of the learning model are some of the other prominent challenges that complicate the algorithm development. Missing values are often encountered in electronic records. For instance, the patient table in Figure 1.5 has missing date of birth for many patients. If the missing values cannot be imputed, patients with missing values or the associated type of information are discarded. Elimination of subjects and features because of missing values may significantly reduce the size of the dataset. In addition, the interpretability of machine learning models for healthcare is another important challenge. It is crucial to incorporate the domain expertise into the learning scheme in biomedical informatics. However, it is often not possible to infer biological meanings from the output and the learned parameters of a computational model, such as deep learning models. Learning interpretable models for biomedical informatics applications is another ongoing research area [105].

One of the most important characteristics of the EHRs is its longitudinal nature. While the medical history of patients is a significant element for disease progression and risk prediction stud-

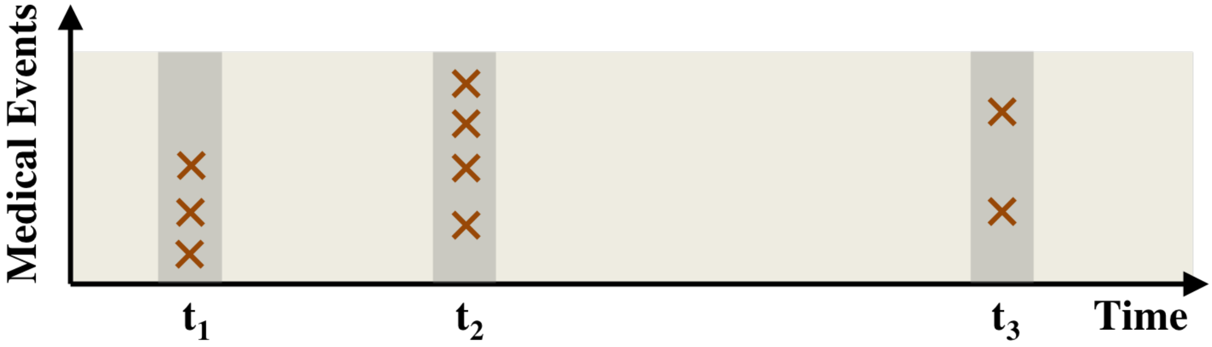


Figure 1.6 Medical record of a patient contains different medical events at each time step. The gap between two consecutive time steps is often irregular in EHR datasets. For this reason, EHRs are harder to analyze compared to univariate regularly sampled time series data.

ies, it is also challenging to leverage the temporal aspect of EHRs. Different than standard time series data such as audio signals, EHRs have high dimensional heterogeneous data points changing over time. Furthermore, the sampling rate is usually unknown and the elapsed time between consecutive records of a patient is not uniform throughout the medical history. An illustration of longitudinal records of a patient is given in Figure 1.6. The challenging characteristics of the EHRs also complicates the visualization of the health information. Visualization is an important component of interactive data analytics tools, which enables domain experts to interpret and sometimes tune the output of computational techniques. Therefore, it is important to design models that facilitate a visualization approach for the results.

1.1.3 Role of Machine Learning in Biomedical Informatics

Biomedical informatics aims to discover underlying characteristics of patient data (e.g., EHRs) and subsequently use the extracted information to predict outcome of healthcare tasks. Machine learning and data mining offer the necessary data-driven techniques for the aforementioned purpose. In particular, machine learning provides solutions for challenging biomedical informatics tasks such as personalized medicine [97], patient phenotyping [56], and adverse drug reaction (ADR) prediction [46], where computational techniques are expected to provide clinically sensible

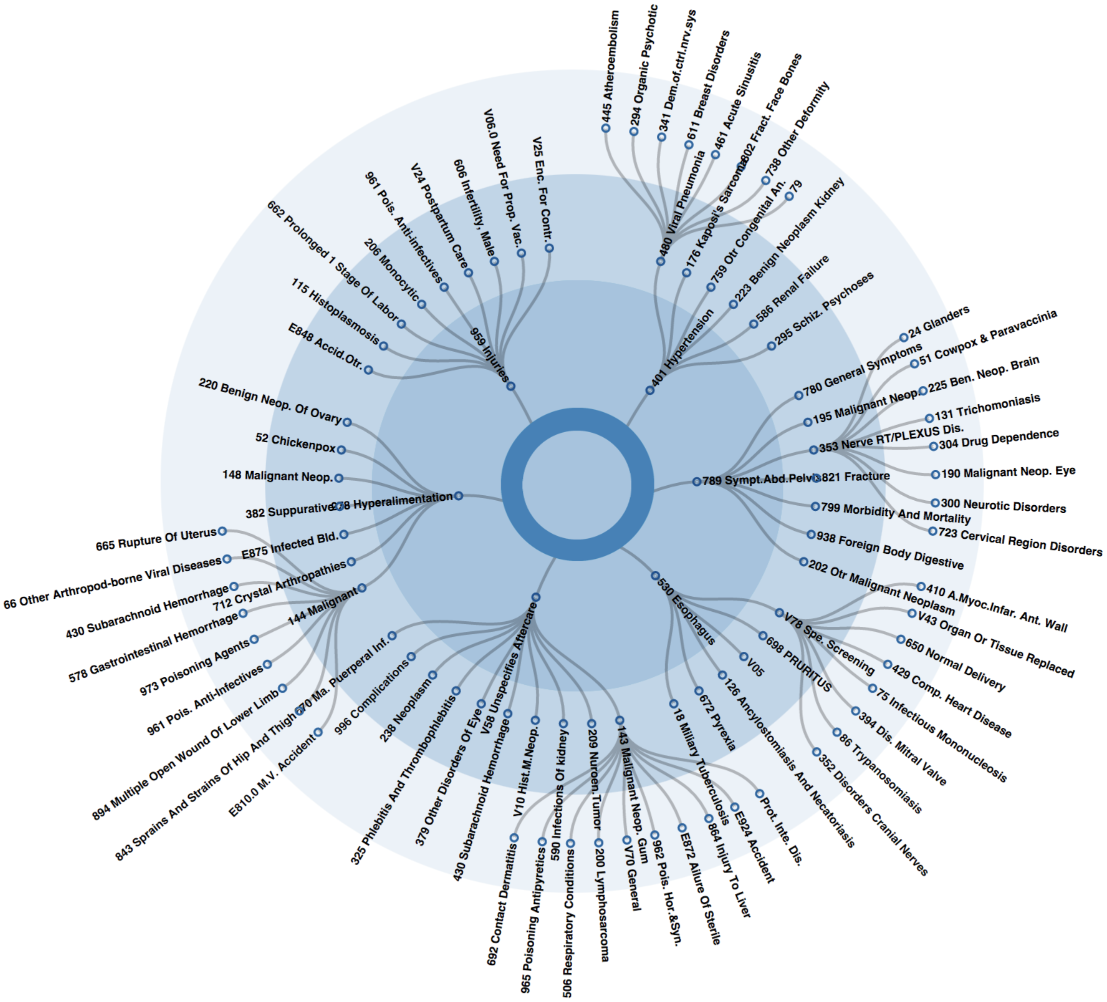


Figure 1.7 An example of hierarchical patient phenotyping visualization. Each node in this tree gives a structured clinical phenotype and a stable subcohort characterized by the phenotype.

performances. For instance, dimensionality reduction eliminates redundancy in EHRs and enables interpretation for tasks such as patient phenotyping. Consequently, dimensionality reduction methods can also facilitate visualization of patient phenotyping procedure as given in Figure 1.7. In the figure, a tree structure demonstrating the hierarchical clinical phenotypes is shown [9].

Deep learning is commonly employed to offer a solution to complicated healthcare tasks, such as temporal clinical event prediction [28] and concept embedding [29]. Deep auto-encoders can learn distinctive patient representations from heterogeneous and longitudinal EHRs with non-linear interactions. In particular, RNNs are utilized to forecast future clinical conditions of patients, e.g., predicting the future diagnoses and date of the future hospital visit of patients. Machine learning and deep learning algorithms in biomedical informatics also aim to provide efficient ways to integrate domain expertise into the learning model, which improves the reliability of the computational techniques. In short, machine learning facilitates the extraction of crucial information present in the EHRs that assists physicians in their efforts to treat patients and plan their long-term care. In the following sections, fundamentals of machine learning, challenges, and machine learning solutions to tackle these challenges are summarized.

1.2 Machine Learning

One important factor that drives advances in machine learning research is the growth of digital data and the consequent demand for data analysis and interpretation tools. As more domains adopt machine learning techniques for their data, new machine learning models, which are computationally efficient and capable of analyzing complex data, have been developed. Figure 1.8 shows the expected growth of the world markets for artificial intelligence (AI) systems between 2017 and 2025. As can be seen in the figure, world markets invest in AI technology with an increasing rate. AI, machine learning and deep learning are all related to machine perception [91], therefore the term AI usually refers to machine learning and deep learning applications.

1.2.1 Problem Setting

In a typical machine learning problem setting, a model is learned from a training set containing input and target pairs, $(\mathbf{x}, y)$. An independent test set is necessary to evaluate the learning algorithm. Input $\mathbf{x} \in \mathbb{R}^d$ is a d dimensional feature vector. Depending on the type of task, i.e. classification or regression, target y can be an integer, boolean or real valued scalar or vector. In classification problems, target y is also called the class label. Learning problems, where a target set is available, partially available or unavailable are named supervised, semi-supervised or unsupervised, respectively. Regardless of the target availability, machine learning algorithms are designed to learn a function that can map the input data $\mathbf{x}$ to the target y; $f(\mathbf{x}) = y$ for the supervised problems and

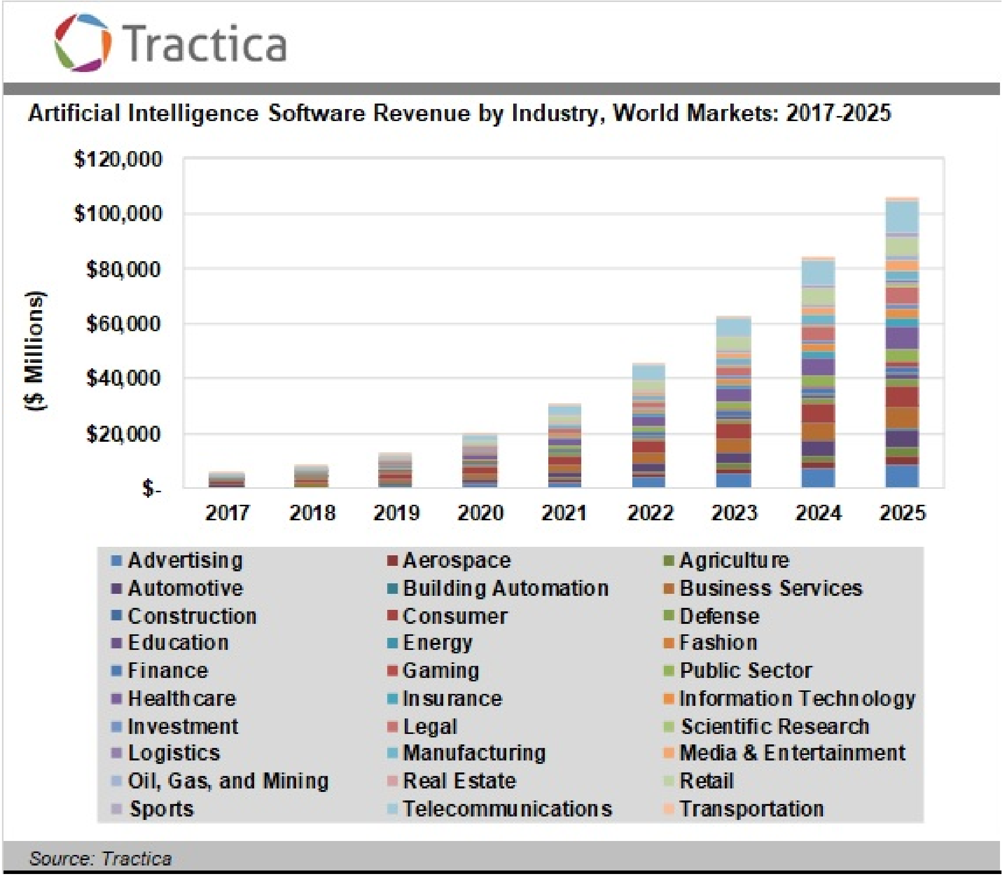


Figure 1.8 Expected growth of the world market for artificial intelligence (AI) systems between 2017-2025 [110].

a function that projects the input x into another feature space, where the information contained in the data is amplified for the unsupervised tasks, e.g., clustering. Thus, patterns and trends in the training data are captured into the learned function, a.k.a the model, to be able to make predictions about the unseen data that needs to be analyzed.

Machine learning describes a family of data-driven approaches where the learning-from-data concept fundamentally consists of an optimization problem. Optimization in machine learning is usually different than the standard mathematical optimization. Machine learning aims to optimize the learning model with respect to a specific performance metric (e.g., classification accuracy, area under curve (AUC), F1 score) rather than focusing on optimizing the loss function itself. Hence, the learning procedure iteratively improves the model by optimizing the parameters to attain high

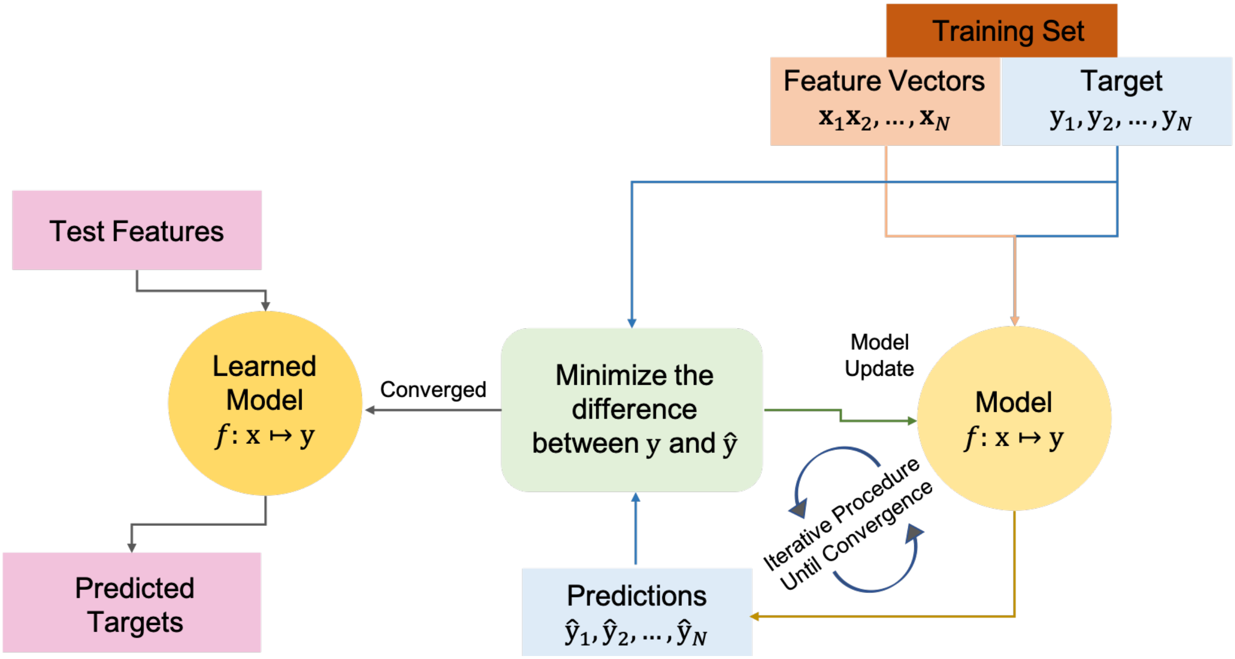

Training Set
Feature Vectors
$\mathbf{x}_1\mathbf{x}_2, \dots, \mathbf{x}_N$
Target
$\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_N$
Test Features
Learned Model
$f: \mathrm{x} \mapsto \mathrm{y}$
Converged
Minimize the difference between y and ŷ
Model Update
Model
$f: \mathrm{x} \mapsto \mathrm{y}$
Iterative Procedure Until Convergence
Predicted Targets
Predictions
$\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \dots, \hat{\mathbf{y}}_N$

Figure 1.9 Overall procedure of supervised learning problem. N is the total number of training data points.

performance for the specified metric of the task. Machine learning often employs a first or second order iterative gradient descent-based optimization scheme since a closed-form solution is often not available. Figure 1.9 summarizes the overall procedure of the supervised learning problem. In short, the main component of a machine learning problem is the optimization scheme whose complexity is usually quite sensitive to the number of data points and the dimensionality of features (number of attributes).

1.2.2 Challenges

In principle, the success of a machine learning algorithm depends on the size and representativeness of the training set. From this perspective, big data should provide excellent opportunities to learn powerful models. However, very large training sets also lead to computation and memory issues. Furthermore, digital data cannot be simply characterized by its size alone. It also poses challenges in terms of its high-dimensional, temporal, heterogeneous, non-linear, and distributed nature. All these challenges complicate the optimization component of the learning-from-data paradigm. Even though there have been tremendous advancements in processor and memory technologies, the data complexity and required tasks often make the available machine learning approaches infeasible. For instance, while the full gradient-descent based optimizers can provide fast convergence, it is not efficient to compute the gradient at each data point in a training set with hundreds of thousands of instances.

As more domains demand machine learning solutions to address their data analytics problems, the complexity of the desired learning tasks has also increased. In the early years of machine learning, tasks with well defined input and output relationships, such as web search, spam filters, recommender systems and fraud detection, were prominent. However, current problems cover a broad spectrum, from autonomous driving to healthcare applications such as mortality prediction. Therefore, both the data and the desired tasks are neither structured nor straightforward anymore. The data has a more complicated structure and the learning algorithms are required to leverage the characteristics of the data, including non-linear interactions, to be able to provide the desired predictive power. For this reason, machine learning algorithms now avoid simplified assumptions, such as convexity and linearity. The recent success of deep learning models, which are highly non-linear and non-convex, supports the aforementioned point.

1.2.3 Machine Learning Solutions

One of the prominent difficulties caused by big data is the complicated nature of optimization [77]. Often, machine learning algorithms solve an optimization problem based on minimizing the errors made by the learned function $f(\cdot)$ on the test data with respect to an evaluation metric. In addition, machine learning problems often pose constraints on the models, leading to use of regularized optimization schemes which typically divide an iteration into two steps (gradient update and proximal projection) when the regularizer is not smooth. As previously mentioned, time complexity of the gradient step increases with the amount of training data in the case of the full gradient descent approach. Another difficulty arises from the computational complexity of the proximal step, which is usually sensitive to the input dimensionality, e.g., nuclear (trace) norm has a proximal projection (singular value thresholding) with $O\left(d^3\right)$ complexity, where d is the feature dimensionality. To handle this challenge, machine learning techniques now utilize stochastic approaches. Stochastic proximal gradient descent and its variants compute the gradient at one randomly sampled point in each iteration, offering a reduced computational complexity. However, random sampling introduces variance to the framework which results in slow convergence. Variance reduction approaches have been developed to improve the convergence rate of the stochastic methods [101, 121].

High dimensionality is alleviated by dimensionality reduction methods. The purpose of dimensionality reduction is not only to decrease the number of the input features, but also to eliminate redundancy in the original representation. Dimensionality reduction methods usually learn a linear or non-linear projection where data is represented in a lower dimensional space and the relevant information in the data is preserved. For instance, principal component analysis (PCA), first developed by Karl Pearson [92], preserves most of the variance present in the data while projecting the original features into a lower dimensional space via a linear transformation. However, one drawback of the traditional PCA is that the principal components are computed as linear combination of all the input dimensions. Therefore, traditional PCA does not enable interpretation of the output dimensions with respect to the input. Sparse PCA [33, 86, 104], which takes the linear combination of only some of the input dimensions, has been introduced to mitigate the interpretability limitation.

In machine learning, it is often assumed that the training data is accessible in a local machine. However, real world data can be distributed over different geographical regions. Furthermore, it may not be always possible to send datasets over a network to a local machine due to limited bandwidth and privacy concerns. Even if data could be centralized, datasets cannot be combined together to learn a predictive model since it would violate the assumption that data points are generated from the same probability distribution model. In such cases, machine learning offers multi-task learning (MTL) [21] approach that treats each distributed dataset as a separate task and combines multiple single task models to obtain a composite model with better generalization property. Furthermore, distributed MTL approaches supported by distributed optimization techniques [85, 115] have been developed to alleviate the need for data centralization.

Another common aspect of the big data is the time dependency. Real world data collected through sensors usually changes with time such as physiological measurements of a patient, or audio and video signals. The temporal structure of the data can provide valuable information and improve predictive performance. For instance, common biomedical informatics tasks include forecasting future diagnoses of patients during their next visit. Therefore, learning temporal dependencies between consecutive elements of a sequence is an important step. However, it is not straightforward to extract temporal patterns and incorporate them into learning. This is especially true when multivariate data is changing over time such as complete medical records of a patient. One popular solution providing impressive performance for temporal or sequential data is deep learning. In particular, recurrent neural networks (RNNs) have been commonly used to learn long term dependencies in sequences with complicated structures.

1.3 Dissertation Focus and Contributions

In this dissertation, our main focus is to develop machine learning approaches to mitigate some of the challenges in biomedical informatics. In particular, machine learning and deep learning approaches are developed to assist computational patient phenotyping, to mitigate learning predictive models for distributed EHRs, to learn a single representation for temporal patient records with irregular elapsed times, and to capture short and long term dependencies in time series with a decoupled memory recurrent neural network.

- Computational patient phenotyping, which requires the analysis of large patient populations and interpretation of input features, is addressed by a convex sparse PCA approach [9, 10]. A proximal variance reduced stochastic gradient descent method is used to solve the convex sparse PCA problem. The proposed framework offers an interpretable patient phenotyping approach due to the sparsity, and a time-efficient optimization approach due to stochastic optimization with variance reduction. Furthermore, fast convergence properties can be attained by considering the convex formulation of sparse PCA, which normally leads to a non-convex optimization problem.

- A distributed multi-task learning approach with asynchronous updates [12] is introduced to efficiently utilize distributed EHR datasets. EHRs are stored in different hospitals and they cannot be transferred over a network because of privacy concerns and limited bandwidth. The proposed distributed framework enables learning individual predictive models separately and transferring a single vector over the network instead of the whole EHR dataset. Knowledge transfer between the single models is performed asynchronously in a central server. The asynchronous nature ensures a more robust learning framework in the presence of network delays and failures compared to the traditional approaches such as centralized and synchronous multi-task frameworks.

- A new Long-Short Term Memory (LSTM) network, named time-aware LSTM [11], is proposed for longitudinal EHR datasets with irregular elapsed times. Elapsed time between two consecutive patient records, which is a significant element of clinical decision making, usually varies from months to years. This time gap is used to modify the effect of the previous cell memory to the current output in time-aware LSTM. The proposed architecture is deployed in an auto-encoder setting to solve the patient-subtyping problem, which aims to group patients based on similar progression pathways.

- Due to the interest and the positive feedback from the research community about time-aware LSTM, we extend the proposed idea to a decoupled memory gated unit architecture, named decoupled memory recurrent network (DM-GRN). The main purpose of the study is to provide a different memory decomposition approach to be able to capture long and short-term dynamics more explicitly. The proposed model is evaluated for healthcare and traffic speed datasets. Memory properties are discussed and visualized using synthetic examples.

In the subsequent four chapters, the proposed approaches summarized above will be presented in detail. In each chapter, literature is reviewed, methodology is explained, and experimental results are discussed. A summary and conclusions of this dissertation research are presented in Chapter 6.

Chapter 2

PhenoTree: Hierarchical Phenotyping via Sparse Principal Component Analysis

Computational patient phenotyping is a data-driven task of discovering clinical phenotypes in a large patient cohort. The output of the patient phenotyping task is patient groups with similar diagnostic pathways. In this study, diagnosis information of the patients is provided with ICD-9 codes, which are universal codes to depict diagnostic groups. ICD-9 codes (e.g. more than $10,000$ in this study) can be represented with high dimensional sparse vectors. As a result, the computational phenotyping problem becomes obtaining patient groups of similar diagnoses in a large patient cohort using ICD-9 codes. Since the ground-truth patient groups are not available, this task is an unsupervised learning problem. Time efficiency of the phenotyping method and the interpretability of the results are two important factors that facilitate the role of domain experts in the phenotyping procedure. In this chapter, we propose a hierarchical phenotyping approach based on sparse principal component analysis (SPCA). Dimensionality reduction methods are commonly used to analyze high dimensional data. In particular, PCA aims to map high dimensional data into a lower dimensional space where the most salient information in the data is preserved. On the other hand, the output dimensions obtained by PCA cannot be easily interpreted regarding the input dimensions. SPCA improves the interpretability of the output dimensions by learning a sparse linear transformation. Time efficiency of the patient phenotyping approach is addressed by solving the SPCA problem using a stochastic proximal gradient descent technique with variance reduction. An effective visualization technique is provided to assist physicians in analyzing the patient phenotyping results. In summary, an interpretable and time efficient interactive computational phenotyping tool is introduced in this chapter.

2.1 Introduction

Electronic health records (EHRs) provide a digital platform to store comprehensive patient information. The availability of digital patient records facilitates many challenging biomedical informatics tasks. One of the challenging tasks is to identify clinical *phenotypes* that characterize patient cohorts. The term phenotype is originally defined as a composition of observable properties of an organism, as a result of the interactions between its genotype and environmental surroundings [131]. Clinical phenotypes, on the other hand, represent patient categories with different combinations of diseases [1]. In the rest of the chapter, the term phenotype will be used for the clinical phenotypes. Obtaining clinical phenotypes facilitates developing the most suitable treatments for patients with specific requirements. For this reason, patient phenotyping can be considered an important step towards personalized medicine. Patient phenotyping requires analysis of large number of patient records to extract the key clinical features that characterize certain patient groups. Given the scale and the complexity of the EHR data, manual extraction of phenotypes by physicians is not feasible. On the other hand, availability of the vast amount of patient data offers many opportunities for machine learning research to improve patient phenotyping task and patient care in general. For this reason, biomedical informatics resorts to machine learning techniques to assist physicians in extraction of clinical phenotypes from large scale patient cohorts [23, 57, 120, 131].

Inferring clinical phenotypes using machine learning techniques is named *computational phenotyping*. Computational phenotyping techniques usually aim to learn the coarse characteristics of the population. However, it is more informative to explore finer granularities and hierarchies in phenotypes [82, 111]. In such cases, domain knowledge is necessary to refine computational phenotyping results. For this reason, a visually interpretable and time efficient computational phenotyping tool is essential to assist physicians in the interpretation and evaluation of the phenotype hierarchy. One of the challenges of developing such tools is the scale of the data, and consequently the long processing time. To overcome the time efficiency and interpretability challenges, we introduce PHENOTREE, a visual analytics tool utilizing SPCA to obtain hierarchical phenotypes of large patient cohorts. In particular, PHENOTREE explores hierarchical phenotypes by iteratively applying SPCA, and visualizing the phenotypes at different levels of granularity as a tree structure. Given cohort/sub-cohorts, PHENOTREE employs SPCA to identify key clinical features at each level of phenotype hierarchy. At the end of the PHENOTREE procedure, phenotype of a corresponding patient group is generated as a set of key clinical features at different granularities. To address the time complexity of standard SPCA approaches, a convex formulation of SPCA is adopted so that a variance reduced stochastic gradient descent solver with fast convergence can be used. Experiments on two real-world EHR patient cohorts are conducted to demonstrate the phenotyping application of the proposed PHENOTREE. Qualitative assessments show that PHENOTREE can provide clinically plausible results. In addition, time efficiency and convergence properties of the proposed SPCA algorithm are investigated.

2.2 Literature Review

In this study, we propose a phenotyping method using SPCA and investigate a first order stochastic gradient descent approach to solve the SPCA problem. For this reason, in this section, we review clinical phenotype discovery methods and stochastic optimization techniques in literature.

2.2.1 Data-Driven Phenotyping

Due to its sparsity and noise, raw EHR data is not informative enough to directly utilize in clinical research. Sparsity of the EHRs is often due to the fact that patients have various combinations of different diseases (some diseases can be rare in the cohort). As a result, when the patient information is represented with a fixed-length vector (e.g., size is based on the dictionary of unique diagnoses), patient vectors will be sparse. EHR data is also noisy due to errors and anomalies in EHR softwares. Computational phenotyping provides a more stable and robust representation for patients than their raw EHR. As discussed by Zhou *et al.* [131], extracting phenotypic patterns of patients is an important task which can contribute to the personalized medicine. Authors proposed that the clinical features in EHR data can be mapped to a much lower dimensional latent space and utilized a matrix completion approach to extract patient phenotypes [131]. Ho *et al.* [57] proposed a phenotyping method, named Marble, by introducing a sparse non-negative tensor factorization approach to obtain phenotype candidates. Ho *et al.* also defined the properties of an ideal phenotype, such as representativeness of the complex interactions between several sources and interpretability.

Deep learning has also been successfully utilized to solve biomedical informatics tasks. For instance, a deep model is used for the discovery and detection of characteristic patterns in clinical data [23]. Che *et al.* showed that deep neural networks can learn relevant features for medical applications. Authors state that the proposed framework improves the multi-label classification performance such as predicting ICD-9 codes. On the other hand, Marlin *et al.* proposed an unsupervised approach to computational phenotyping [84]. The temporal sparsity of EHR data is addressed using a probabilistic clustering model with an empirical prior distribution which was used to deal with the sparsity of the data. Authors also state that the proposed model can capture physiological patterns, and the clusters can distinguish different physiological variables [84].

2.2.2 Sparse Principal Component Analysis and Stochastic Proximal Optimization

SPCA was proposed for the first time by Hastie *et al.* [61] to address the interpretability issue of the PCA. Traditional PCA learns dense loading vectors so that the principal components are the linear combinations of all the input dimensions. In this case, it is hard to interpret the principal components regarding the contribution of the input dimensions. On the other hand, Hastie *et al.* proposed to learn sparse loading vectors using the lasso (elastic net) constraint so that the principal components become linear combinations of only some of the input dimensions. In addition, d'Aspremont *et al.* proposed a SPCA approach based on semi-definite programming [33]. On the other hand, Journee *et al.*, introduced two types of SPCA approaches [68]. The proposed formulations are based on maximizing a convex function on a compact set using ℓ_1 or ℓ_0 norms. However, large scale and high dimensional patient records cannot be handled by these approaches. A stochastic SPCA algorithm which can deal with large scale data with exponential convergence rate is proposed [104]. Furthermore, an approach with stochastic iterations with variance reduction, which had been previously proposed [67], is utilized. Variance reduction mechanism requires strong convexity, however SPCA is a non-convex problem. Johnson and Zhang [67] provided a different convergence analysis that does not include strong convergence property.

In this study, we propose to use a stochastic approach with variance reduction framework. To be able to leverage the high convergence property of convex problems, convex formulation of PCA [48] is adopted. The SPCA is posed as an ℓ_1 norm regularized optimization problem, therefore a proximal gradient descent approach is required. In literature, several proximal gradient based methods have been developed such as [13,119]. FISTA by Beck and Teboulle [13] offers the fastest convergence rate among the first order methods. However, FISTA ensures the fast convergence rate for full gradient descent, which is not scalable. Therefore, stochastic gradient approaches are more suitable for large scale problems. On the other hand, stochastic gradient descent suffers from high variance, leading to low convergence rate. Nitanda [88] introduced a variance reduction framework with Nester's acceleration method to alleviate the aforementioned drawback of the stochastic gradient approach. Johnson and Zhang [67] also introduced a variance reduction approach which progressively reduces the variance in proximal gradient descent. Strong convexity of the objective function is required to achieve a geometric convergence rate under expectation [67]. Another variance reduction approach for proximal algorithms was proposed by Xiao and Zhang [121] where a multi-stage scheme is presented with strong convexity and Lipschitz continuity assumptions.

2.2.3 Visual Analytics for EHR

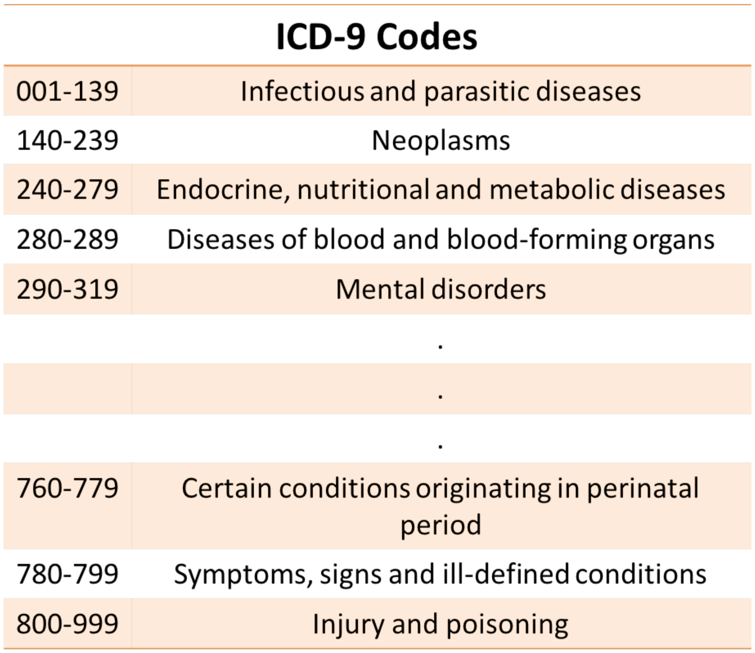
Visualization is essential in biomedical informatics to ensure that the domain experts can interpret the output of data-driven models. For this purpose, Perer *et al.* proposed an interactive mining and visualization framework, named Care Pathway Explorer, to capture frequent events in the EHR data [95]. Another interactive visualization approach based on mining the EHRs and analyzing clinical sequences was developed by Gotz *et al.* [50]. Wang *et al.* focused on a visual analysis method for chronic kidney disease [114]. Huang *et al.* similarly utilized an interactive approach to classify patients into different subsets [59]. Authors developed a visually rich web-based application that can help physicians and researchers to comprehend and study patient cohorts over time.

2.3 PHENOTREE

In this section, the proposed PHENOTREE is introduced. In the subsequent sections, details of the EHR data used in this study, the proposed convex SPCA approach, and the optimization scheme are presented.

2.3.1 Electronic Health Records

EHRs comprise digital patient information of different modalities (e.g., diagnosis, medication, test result, and demographics) collected over a time period [54]. Diagnostic information is often recorded as international codes, such as ICD-9. Each ICD-9 code corresponds to a specific diagnosis and there is a hierarchical relationship between the codes. Some of the ICD-9 codes and their corresponding diagnostic groups are given in Figure 2.1. As shown in the figure, ICD-9 codes from 001 to 139 represent infectious and parasitic diseases, and one of its subgroups, ICD-9 010 to 018, corresponds to tuberculosis. Patient demographics might also be explicitly provided in EHR datasets, however patient's gender and age group can sometimes be inferred from the ICD-9 codes. For example, ICD-9 630 to 679 encode complications of pregnancy and childbirth. Hence, even



ICD-9 Codes	
001-139	Infectious and parasitic diseases
140-239	Neoplasms
240-279	Endocrine, nutritional and metabolic diseases
280-289	Diseases of blood and blood-forming organs
290-319	Mental disorders
	.
	.
	.
760-779	Certain conditions originating in perinatal period
780-799	Symptoms, signs and ill-defined conditions
800-999	Injury and poisoning

Figure 2.1 ICD-9 codes are used to represent different diagnostic groups. These universal codes have a hierarchical structure.

though the patient demographic is not available, gender of the patient can be inferred from ICD-9 630 to 679. Due to its rich information content, ICD-9 is an important source for the computational patient phenotyping task. To be able to utilize ICD-9 codes for the exploration of phenotypes from a large patient cohort, biomedical informatics resorts to machine learning and data mining techniques. However, computational phenotyping results cannot have direct clinical implications until they are validated by medical experts. For this reason, it is essential to visualize the extracted phenotypes such that the medical experts can approve and refine the results. In this study, a visualization approach is proposed to be able to interpret the extracted phenotypes. Details of the proposed patient phenotyping approach based on SPCA are presented in the next sections.

2.3.2 Phenotyping via SPCA

Computational patient phenotyping process offers clinical guidance to domain experts as long as the extracted phenotypes are interpretable. In particular, the exploration of the clinical phenotypes at different levels of granularity is necessary to assist clinical researchers in understanding the diagnostic properties of different sub-groups in a patient cohort. Machine learning and data mining techniques facilitate extraction and visualization of hierarchical phenotypes due to their capability of inferring underlying patterns in large datasets. The computational phenotyping task discussed in this study is an unsupervised learning problem, where the ground-truth patient sub-groups are not available. For this reason, we need to design a machine learning model that can extract key clinical features from an EHR dataset to represent a group of patients without any supervision. Such key clinical features can be obtained based on the frequencies of diagnosis (e.g., ICD-9 code) encountered in the EHRs. However, a frequency based approach ignores the dependencies and the hierarchical relationships between the features. To address the aforementioned challenges, we propose a patient phenotyping approach based on SPCA, an unsupervised dimensionality reduction technique. PCA, expressed in Eq. 2.3.1, is one of the most popular unsupervised dimensionality reduction approaches.

$$\max_{\mathbf{Z}\in\mathbb{R}^{d\times p}} \|\mathbf{S}\mathbf{Z}\|_F^2, \quad \text{s.t. } \mathbf{Z}^T\mathbf{Z} = \mathbf{I}, \tag{2.3.1}$$

where d is the input dimensionality, p is the number of principal components or the output dimensionality, S is $d \times d$ covariance matrix, $\mathbf{Z}$ is a $d \times p$ orthogonal projection matrix, and $\|.\|_F$ denotes the Frobenius norm. Eq. 2.3.1 represents a constraint optimization problem, where an orthogonal transformation matrix is learned from the data. PCA ensures that the top principal components retain most of the variance existing in the data. Thus, the original data can be projected into a lower dimensional space without losing the significant information. However, PCA is not suitable to interpret the output dimensions. Since the columns of the projection matrix $\mathbf{Z}$ (loading vectors) are dense, the principal components are computed as a linear combination of all the input dimensions. SPCA addresses this drawback by learning sparse loading vectors [61], where only a subset of input dimensions are combined to obtain principal components as shown in Figure 2.2. Therefore, SPCA is more interpretable in terms of analyzing the contributions of the input dimensions to the principal components. For this reason, SPCA is used to design the proposed patient phenotyping

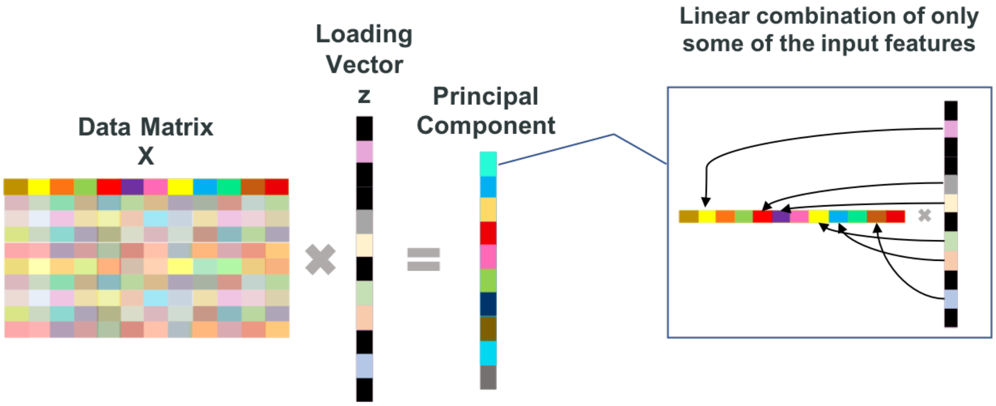


Figure 2.2 SPCA learns a sparse loading vector. Principal components are represented as a linear combination of a subset of the input dimensions.

approach. The key clinical features can be defined as the input dimensions that correspond to the non-zero elements of the sparse loading vector. Thus, a set of sub-cohorts, each of which comprises the patients associated with one of the key features, can be obtained. Therefore, SPCA is applicable to phenotyping task and the sparsity enables a hierarchical representation that facilitates the visualization. On the other hand, traditional SPCA methods have high time complexity that prevents designing interactive tools. To alleviate the time complexity of SPCA, a stochastic convex SPCA approach is introduced. The details of the proposed model is presented in the following section.

2.3.3 Stochastic Convex SPCA (Cvx-SPCA)

Existing SPCA methods usually suffer from scalability, which is an obstacle for developing efficient patient phenotyping tools. For this reason, we propose a convex formulation to be able to leverage fast convergence property of a stochastic approach with variance reduction. In this study, we focus on the first principal component. Finding the sparse loading vector of the first principal component can be posed as an ℓ_1 norm ($\|.\|_1$) regularized optimization problem as given in Eq. 2.3.2.

$$\min_{\mathbf{z}\in\mathbb{R}^d} -\mathbf{z}^T\mathbf{S}\mathbf{z} + \gamma \left\|\mathbf{z}\right\|_1 , \tag{2.3.2}$$

where d dimensional vector $\mathbf{z}$ is the loading vector of the first principal component, $\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T$ is the covariance matrix, and γ is the regularization parameter to control the sparsity of $\mathbf{z}$. This formulation contains a smooth first term and non-smooth second term. Proximal gradient descent provides a typical solution for the composite formulations such as Eq. 2.3.2. Proximal gradient descent method divides the optimization problem into two simple steps. In the first step, the next search point is updated using the gradient of the smooth part. In the second step, optimal point is obtained by evaluating the proximal operator of the non-smooth part at the updated point obtained from the first step. As it was discussed earlier, proximal methods [13, 119] provide fast convergence rates, however the full gradient is required in each iteration. Hence the computational time increases drastically for large scale datasets such as EHRs. Stochastic proximal gradient descent (Prox-SGD), where gradient is computed for a single data point at each iteration, is more scalable. However, the stochastic methods suffer from low convergence due to the high variance. The variance is introduced by the random sampling at each iteration. To deal with the variance, stochastic algorithms adopt diminishing step size that increases the number iterations to converge to the optimal solution.

Stochastic proximal gradient methods with variance reduction have been proposed to progressively decrease the variance to avoid the slow convergence. For instance, a proximal stochastic gradient approach with variance reduction (Prox-SVRG) [121] is available to alleviate the effects of the high variance. Prox-SVRG provides a geometric convergence rate which is much faster than the traditional Prox-SGD. On the other hand, the fast convergence of Prox-SVRG depends on the convexity of the objective and Lipschitz continuity of the gradient. Since the formulation in Eq. 2.3.2 displays a non-convex objective function, we cannot leverage the convergence properties of Prox-SVRG. For this reason, we propose to formulate the SPCA with the following convex optimization problem [10]:

$$\min_{\mathbf{z}\in\mathbb{R}^d}\left(\frac{1}{2}\mathbf{z}^T\left(\lambda\mathbf{I}-\mathbf{S}\right)\mathbf{z}-\mathbf{w}^T\mathbf{z}\right)+\gamma\left\|\mathbf{z}\right\|_1, \tag{2.3.3}$$

where $\lambda > \lambda_1(\mathbf{S})$ is the convexity parameter, $\lambda_1(\mathbf{S})$ represents the largest eigenvalue of the covariance matrix $\mathbf{S}$, and $\mathbf{w} \in \mathbb{R}^d$ is a random vector to make sure that the first order derivative of the smooth term is not zero. The first term of this formulation, which was proposed as an approximation of finding the first principal component [48], is strongly convex. If λ is greater than the largest eigenvalue of $\mathbf{S}$, then $\lambda\mathbf{I} - \mathbf{S}$ will be positive definite which is a necessary condition for strong convexity. On the other hand, the regularization term, $\|\cdot\|_1$, is not strongly convex, but convex. Therefore, the composite function contains a strongly convex and a convex term which make the overall function strongly convex.

2.3.3.1 Optimization Scheme

In this thesis, Prox-SVRG approach [121] is used to solve the problem in Eq. 2.3.3 which is the combination of a smooth $F(\mathbf{z}) = \left[\mathbf{z}^T(\lambda\mathbf{I} - \mathbf{S})\mathbf{z} - \mathbf{w}^T\mathbf{z}\right]$ and a non-smooth part $R(\mathbf{z}) = \|\mathbf{z}\|_1$. In this case, $F(\mathbf{z})$ can also be written as the sum of n smooth functions as given below.

$$F(\mathbf{z}) = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left[\frac{1}{2}\mathbf{z}^T\left(\lambda\mathbf{I} - \mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{z} - \mathbf{w}^T\mathbf{z}\right] \tag{2.3.4}$$

When n in Eq. 2.3.4 is large, computing the full gradient of the smooth part at each iteration will be very time consuming. In contrast, Prox-SVRG [121] approach computes the gradient at a randomly sampled data point in each iteration, and the variance of the gradient is upper bounded by a multi-stage progressive variance reduction scheme. Furthermore, the variance is ensured to converge to zero upon the optimal solution is obtained. Detailed proof of bounding the variance can be found in Section 3.1 of [121].

In this study, Prox-SVRG optimization scheme given in Algorithm 1 is adopted to solve the SPCA formulation in Eq. 2.3.3. In Algorithm 1, $\mathbf{z}_0$ is the initial value of the loading vector $\mathbf{z}$, which is usually randomly sampled from normal distribution, η is a constant step size, m is the number of iterations for each epoch s, and T is the maximum number of epochs. The most time consuming component of the algorithm is the computation of the full gradient $\tilde{\mathbf{v}}$, which requires multiplication

Algorithm 1 Prox-SVRG algorithm for solving Cvx-SPCA.

```
Require: $\lambda, [x_1, x_2, ..., x_n], \mathbf{S}, \mathbf{w}, \mathbf{z}_0, \eta, \gamma, m, T$
Ensure: $\mathbf{z}$
for $s = 1, 2, ...T$ do
    $\tilde{\mathbf{z}} = \tilde{\mathbf{z}}_{s-1}$
    $\tilde{\mathbf{v}} = (\lambda \mathbf{I} - \mathbf{S}) \tilde{\mathbf{z}} - \mathbf{w}$
    $\mathbf{z}_0 = \tilde{\mathbf{z}}$
    for $k = 1, 2, ..., m$ do
        Pick $x_{ik} \in \{x_{1k}, ..., x_{nk}\}$ randomly
        $\mathbf{v}_k = \left(\lambda \mathbf{I} - x_{ik} x_{ik}^T\right) (\mathbf{z}_{k-1} - \tilde{\mathbf{z}}) + \tilde{\mathbf{v}}$
        $\mathbf{z}_k = \text{prox}_{\eta\gamma} (\mathbf{z}_{k-1} - \eta \mathbf{v}_k)$
    end for
    $\tilde{\mathbf{z}}_s = \frac{1}{m} \sum_{k=1}^{m} \mathbf{z}_k$
end for
return $\tilde{\mathbf{z}}_T$
```

of a $d \times d$ matrix with a d dimensional vector, in each epoch. However, this multiplication needs to be performed only once before the iterations and the same copy of the full gradient is used to iteratively reduce the variance of the stochastic gradient. In Algorithm 1, $\tilde{\mathbf{z}}$ denotes an estimate of the optimal point, which is updated as the average solution obtained throughout an epoch. At each iteration, one data point is randomly sampled and the gradient is computed at that point as given below:

$$\begin{aligned} \mathbf{v}_k &= \nabla f_{ik} (\mathbf{z}_{k-1}) - \nabla f_{ik} (\tilde{\mathbf{z}}) + \nabla F (\tilde{\mathbf{z}}) \\ &= \left(\lambda \mathbf{I} - x_{ik} x_{ik}^T\right) (\mathbf{z}_{k-1} - \tilde{\mathbf{z}}) + (\lambda \mathbf{I} - \mathbf{S}) \tilde{\mathbf{z}} - \mathbf{w}, \end{aligned} \tag{2.3.5}$$

where $\nabla F (\tilde{\mathbf{z}})$ is the average gradient of functions $f_i (\mathbf{z}), i = 1, ..., n$ or the full gradient at point $\tilde{\mathbf{z}}$, $\nabla f_{ik} (\mathbf{z}_{k-1})$ is the gradient of the function calculated by using the data point x_{ik} sampled at the kth iteration and $\tilde{\mathbf{z}}$ is the average of $\mathbf{z_k},\ k = 1, .., m$ at the end of an epoch.

Although the gradient in Eq. 2.3.5 is different than the actual gradient, it is still an estimate of the full gradient that can be shown by taking the expectation of $\mathbf{v}_k$. Thus, the variance reduced gradient has the same direction as the full gradient under expectation. After the gradient update of the smooth part, $[\mathbf{z}_{k-1} - \eta \mathbf{v}_k]$, proximal mapping of ℓ_1 norm is applied to obtain the final solution as follows.

$$\begin{aligned}\mathbf{z}_k &= \mathrm{prox}_{\eta,\gamma}\left(\mathbf{z}_{k-1} - \eta\mathbf{v}_k\right) \\ &= \mathrm{sign}\left(\mathbf{z}_{k-1} - \eta\mathbf{v}_k\right)\max\left(0, \left|\mathbf{z}_{k-1} - \eta\mathbf{v}_k\right| - \eta\gamma\right)\end{aligned} \tag{2.3.6}$$

In the Prox-SVRG algorithm, variance of the stochastic gradient $\mathbf{v}_k$ is reduced progressively, while both $\tilde{\mathbf{z}}$ and $\mathbf{z}_{k-1}$ are converging to the optimal point $\mathbf{z}_* = \arg\min_{\mathbf{z}} P\left(\mathbf{z}\right)$ [121]. Since the full gradient is utilized to modify stochastic gradients and function F is an average of the smooth component functions, variance can be bounded. In the next section, convergence analysis of Prox-SVRG is summarized.

2.3.3.2 Convergence Analysis

The objective function proposed in this study is suitable to follow the convergence analysis of Prox-SVRG. Therefore, our analysis is mostly adapted from [121]. However, we use weak conditions which allow a broader family of objective functions to fit in this scheme and leverage the geometric convergence. The first assumption is retained as in [121].

Assumption 2.3.1. *The function $R\left(\mathbf{z}\right)$ is lower semi-continuous and convex, and its effective domain, $dom\left(R\right) := \left\{\mathbf{z} \in \mathbb{R}^d | R\left(\mathbf{z}\right) < +\infty\right\}$ is closed. Each $f_i\left(\mathbf{z}\right), for \quad i = 1, ..., n$, is differentiable on an open set that contains $dom\left(R\right)$, and their gradients are Lipschitz continuous. That is, there exist $L_i > 0$ such that for all $\mathbf{z}, \mathbf{y} \in dom\left(R\right)$,*

$$\left\|\nabla f_i\left(\mathbf{z}\right) - \nabla f_i\left(\mathbf{y}\right)\right\| \leq L_i \left\|\mathbf{z} - \mathbf{y}\right\|$$

which also implies that the gradient of the average function $F\left(\mathbf{z}\right)$ is also Lipschitz continuous, i.e., there is an $L > 0$ such that for all $\mathbf{z}, \mathbf{y} \in dom\left(R\right)$,

$$\left\|\nabla F\left(\mathbf{z}\right) - \nabla F\left(\mathbf{y}\right)\right\| \leq L \left\|\mathbf{z} - \mathbf{y}\right\|$$

where $L \leq (1/n)\sum_{i=1}^{n} L_i$.

In [121], convergence analysis assumed that the objective function is strongly convex. On the other hand, we only assumed that functions $F(\mathbf{z})$ and $R(\mathbf{z})$ are convex, but not necessarily strongly convex. Thus, one strong assumption is relaxed. Strong convexity provides important properties for faster convergence rates. However, this strong assumption does not always hold in practice, and for this reason, a simplified version of the analysis will be preferable. We drop the strong convexity assumption at two points in the original analysis [121] and obtain the convergence rate given in the following theorem.

Theorem 2.3.2. *Under the assumption that Assumption 2.3.1 holds and* $0 < \eta < 1/(4L_Q)$*, where* $L_Q = max_i L_i$*, the convergence rate is obtained as follows:*

$$\begin{aligned} \rho &= \frac{1}{\ell(1-4L_Q\eta)m\eta} + \frac{4L_Q\eta(m+1)}{(1-4L_Q\eta)m} < 1, \\ \mathbb{E}\{P(\tilde{\mathbf{z}}_s)\} - P(\mathbf{z}_*) &\leq \rho^s\left[P(\tilde{\mathbf{z}}_0) - P(\mathbf{z}_*)\right], \end{aligned} \tag{2.3.7}$$

where $\mathbf{z}_* = \arg\min_{\mathbf{z}} P(\mathbf{z})$.

Proof. The proof of Theorem 2.3.2 starts with investigating the distance between $\mathbf{z_k}$ and $\mathbf{z}_*$; $\|\mathbf{z_k} - \mathbf{z}_*\|^2$. According to the stochastic gradient mapping definition in [121], $\mathbf{z_k}$ can be written as $\mathbf{z_{k-1}} - \eta\mathbf{g_k}$.

$$\|\mathbf{z_k} - \mathbf{z}_*\|^2 = \|\mathbf{z_{k-1}} - \eta\mathbf{g_k} - \mathbf{z}_*\|^2 \tag{2.3.8}$$

$$= \|\mathbf{z_{k-1}} - \mathbf{z}_*\|^2 - 2\eta\mathbf{g_k}^T(\mathbf{z_{k-1}} - \mathbf{z}_*) + \eta^2\|\mathbf{g_k}\|^2 \tag{2.3.9}$$

The term $\left(-\mathbf{g_k}^T(\mathbf{z_{k-1}} - \mathbf{z}_*) + \frac{\eta}{2}\|\mathbf{g_k}\|^2\right)$ can be bounded by using the definition of the proximal update as shown below.

$$\mathbf{z_k} = \mathrm{prox}_{\eta R}(\mathbf{z_{k-1}} - \eta\mathbf{v_k}) = \arg\min_{y}\{\frac{1}{2}\|\mathbf{y} - (\mathbf{z_{k-1}} - \eta\mathbf{v_k})\|^2 + \eta R(\mathbf{y})\} \tag{2.3.10}$$

According to the optimality condition:

$$\mathbf{z_k} - \left(\mathbf{z_{k-1}} - \eta \mathbf{v_k}\right) + \eta \xi = 0 \tag{2.3.11}$$

where $\xi \in \partial R\left(\mathbf{z_k}\right)$ is the subgradient of $R\left(\mathbf{z}\right)$ at $\mathbf{z_k}$. If we combine the stochastic gradient mapping definition with the optimality condition, we obtain the following expression.

$$\mathbf{z_k} - \left(\mathbf{z_k} + \eta \mathbf{g_k} - \eta \mathbf{v_k}\right) + \eta \xi = 0 \Rightarrow \xi = \mathbf{g_k} - \mathbf{v_k} \tag{2.3.12}$$

By using the convexity of $F\left(\mathbf{z}\right)$ and $R\left(\mathbf{z}\right)$, we can write the following inequality.

$$\begin{aligned} P\left(\mathbf{y}\right) = F\left(\mathbf{y}\right) + R\left(\mathbf{y}\right) &\geq F\left(\mathbf{z_{k-1}}\right) + \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{y} - \mathbf{z_{k-1}}\right) &(2.3.13)\\ &+ R\left(\mathbf{z_k}\right) + \xi^T \left(\mathbf{y} - \mathbf{z_k}\right) &(2.3.14)\end{aligned}$$

Convergence analysis in [121] utilized strong convexity of F and R in Eq. 2.3.14. However, we show that strong convexity is not required at this point. Since $F\left(\mathbf{z}\right)$ is assumed to be Lipschitz continuous with Lipschitz constant L, $F\left(\mathbf{z_{k-1}}\right)$ can also be bounded by using Theorem 2.1.5 in [87].

$$F\left(\mathbf{z_{k-1}}\right) \geq F\left(\mathbf{z_k}\right) - \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{z_k} - \mathbf{z_{k-1}}\right) - \frac{L}{2}\left\|\mathbf{z_k} - \mathbf{z_{k-1}}\right\|^2 \tag{2.3.15}$$

If we combine Eqs. 2.3.14 and 2.3.15, we obtain the following inequality.

$$\begin{aligned} P\left(\mathbf{y}\right) &\geq F\left(\mathbf{z_k}\right) - \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{z_k} - \mathbf{z_{k-1}}\right) - \frac{L}{2}\left\|\mathbf{z_k} - \mathbf{z_{k-1}}\right\|^2 &(2.3.16)\\ &+ \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{y} - \mathbf{z_{k-1}}\right) + R\left(\mathbf{z_k}\right) + \xi^T \left(\mathbf{y} - \mathbf{z_k}\right) \\ &\geq P\left(\mathbf{z_k}\right) - \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{z_k} - \mathbf{z_{k-1}}\right) - \frac{L}{2}\left\|\mathbf{z_k} - \mathbf{z_{k-1}}\right\|^2 \\ &+ \nabla F\left(\mathbf{z_{k-1}}\right)^T \left(\mathbf{y} - \mathbf{z_{k-1}}\right) + \xi^T \left(\mathbf{y} - \mathbf{z_k}\right)\end{aligned}$$

Here, we again use stochastic gradient mapping; $[\mathbf{z_k} - \mathbf{z_{k-1}} = -\eta \mathbf{g_k}]$ to obtain the following inequality.

$$P\left(\mathbf{y}\right) \geq \left[P\left(\mathbf{z_k}\right) + \nabla F\left(\mathbf{z_{k-1}}\right)^T\left(\mathbf{y} - \mathbf{z_k}\right) + \xi^T\left(\mathbf{y} - \mathbf{z_k}\right) - \frac{L}{2}\eta^2\left\|\mathbf{g_k}\right\|^2\right] \tag{2.3.17}$$

If we substitute ξ with $\mathbf{g_k} - \mathbf{v_k}$, and then add and subtract $\mathbf{z_{k-1}}$ from the term $\left(\mathbf{y} - \mathbf{z_k}\right)$:

$$\begin{aligned} P\left(\mathbf{y}\right) &\geq P\left(\mathbf{z_k}\right) + \left(\mathbf{v_k} - \nabla F\left(\mathbf{z_{k-1}}\right)\right)^T\left(\mathbf{z_k} - \mathbf{y}\right) \\ &\quad + \mathbf{g_k}^T\left(\mathbf{y} + \mathbf{z_{k-1}} - \mathbf{z_{k-1}} - \mathbf{z_k}\right) - \frac{L}{2}\eta^2\left\|\mathbf{g_k}\right\|^2 \\ P\left(\mathbf{y}\right) &\geq P\left(\mathbf{z_k}\right) + \mathbf{g_k}^T\left(\mathbf{y} - \mathbf{z_{k-1}}\right) + \left(\eta - \frac{L}{2}\eta^2\right)\left\|\mathbf{g_k}\right\|^2 \\ &\quad + \left(\mathbf{v_k} - \nabla F\left(\mathbf{z_{k-1}}\right)\right)^T\left(\mathbf{z_k} - \mathbf{y}\right) \end{aligned} \tag{2.3.18}$$

Under the assumption of $0 < \eta < 1/4L_Q < 1/L$, $\left[\left(\eta - \frac{L}{2}\eta^2\right) = \frac{\eta}{2}\left(2 - L\eta\right)\right]$ can be taken as $\eta/2$. Since $\left(2 - L\eta\right)$ is between $\left(1, 2\right)$ according to the assumption, eliminating $\left(2 - L\eta\right)$ does not change the inequality. Now we will use the result derived above for the term $\left(-\mathbf{g_k}^T\left(\mathbf{z_{k-1}} - \mathbf{z}_*\right) + \frac{\eta}{2}\left\|\mathbf{g_k}\right\|^2\right)$ in Eq. 2.3.8.

$$\left\|\mathbf{z_k} - \mathbf{z}_*\right\|^2 \leq \left\|\mathbf{z_{k-1}} - \mathbf{z}_*\right\|^2 + 2\eta\left(P\left(\mathbf{z}_*\right) - P\left(\mathbf{z_k}\right)\right) - 2\eta\Delta^T\left(\mathbf{z_k} - \mathbf{z}_*\right) \tag{2.3.19}$$

where $\Delta = \mathbf{v_k} - \nabla F\left(\mathbf{z_{k-1}}\right)$ and $\mathbf{z}_*$ corresponds to $\mathbf{y}$. The term $-2\eta\Delta^T\left(\mathbf{z_k} - \mathbf{z}_*\right)$ can further be bounded by using the proximal full gradient update $\bar{\mathbf{z}}_\mathbf{k} = \operatorname{prox}_{\eta R}\left(\mathbf{z_{k-1}} - \eta\nabla F\left(\mathbf{z_{k-1}}\right)\right)$, If Cauchy-Schwarz inequality and the non-expansiveness of the proximal mapping ($\left\|\operatorname{prox}_{\eta R}\left(x\right) - \operatorname{prox}_{\eta R}\left(y\right)\right\| \leq \left\|x - y\right\|$) are utilized, the following expression can be derived.

$$-2\eta\Delta^T\left(\mathbf{z_k} - \mathbf{z}_*\right) = -2\eta\Delta^T\left(\mathbf{z_k} - \mathbf{z}_* + \bar{\mathbf{z}}_\mathbf{k} - \bar{\mathbf{z}}_\mathbf{k}\right) \leq 2\eta\left\|\Delta\right\|\left\|\mathbf{z_k} - \bar{\mathbf{z}}_\mathbf{k}\right\| - 2\eta\Delta^T\left(\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*\right) \tag{2.3.20}$$

If we insert the definitions of $\mathbf{z_k} = (\mathbf{z_{k-1}} - \eta \mathbf{v_k})$ and $\bar{\mathbf{z}}_\mathbf{k} = (\mathbf{z_{k-1}} - \eta \nabla F(\mathbf{z_{k-1}}))$, we will have:

$$-2\eta \Delta^T (\mathbf{z_k} - \mathbf{z}_*) \leq 2\eta^2 \|\Delta\|^2 - 2\eta \Delta^T (\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*) \tag{2.3.21}$$

If we combine the result shown above with Eq. 2.3.19:

$$\begin{aligned} \|\mathbf{z_k} - \mathbf{z}_*\|^2 \leq \|\mathbf{z_{k-1}} - \mathbf{z}_*\|^2 - 2\eta (P(\mathbf{z_k}) - P(\mathbf{z}_*)) \\ + 2\eta^2 \|\Delta\|^2 - 2\eta \Delta^T (\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*) \end{aligned} \tag{2.3.22}$$

Now, expectations of both sides are taken with respect to $\mathbf{z_k}$.

$$\begin{aligned} \mathbb{E}\{\| \mathbf{z_k} - \mathbf{z}_* \|\} \leq \|\mathbf{z_{k-1}} - \mathbf{z}_*\|^2 + 2\eta^2 \mathbb{E}\left\{\|\Delta\|^2\right\} - 2\eta (\mathbb{E}\{P(\mathbf{z_k})\} - P(\mathbf{z}_*)) \\ - 2\eta \mathbb{E}\left\{\Delta^T (\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*)\right\} \end{aligned} \tag{2.3.23}$$

Since $\bar{\mathbf{z}}_\mathbf{k}$ and $\mathbf{z}_*$ are independent from the variable $\mathbf{z_k}$; $\mathbb{E}\left\{\Delta^T (\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*)\right\} = \mathbb{E}\left\{\Delta^T\right\} (\bar{\mathbf{z}}_\mathbf{k} - \mathbf{z}_*) = 0$. Because $\mathbb{E}\left\{\Delta^T\right\} = \mathbb{E}\{\mathbf{v_k} - \nabla F(\mathbf{z_{k-1}})\} = \mathbb{E}\{\mathbf{v_k}\} - \nabla F(\mathbf{v_{k-1}}) = 0$. The variance of the gradient $\mathbb{E}\left\{\|\Delta\|^2\right\}$ is upper bounded in the Prox-SVRG algorithm and we will use the result of Corollary 3 in [121] which is $\mathbb{E}\left\{\|\Delta\|^2\right\} \leq 4L_Q [P(\mathbf{z_{k-1}}) - P(\mathbf{z}_*) + P(\tilde{\mathbf{z}}) - P(\mathbf{z}_*)]$, where $L_Q = \max_i L_i$, $\tilde{\mathbf{z}}_\mathbf{s} = \frac{1}{m} \sum_{k=1}^{m} \mathbf{z_k}$ and $\tilde{\mathbf{z}} = \tilde{\mathbf{z}}_\mathbf{s-1} = \mathbf{z_0}$ for a fixed epoch. After incorporating the bound of the variance of the gradient into the analysis, the following expression is obtained.

$$\begin{aligned} \mathbb{E}\left\{\|\mathbf{z_k} - \mathbf{z}_*\|^2\right\} \leq \|\mathbf{z_{k-1}} - \mathbf{z}_*\|^2 - 2\eta (\mathbb{E}\{P(\mathbf{z_k})\} - P(\mathbf{z}_*)) \\ + 8\eta^2 L_Q [P(\mathbf{z_{k-1}}) - P(\mathbf{z}_*)] + 8\eta^2 L_Q [P(\tilde{\mathbf{z}}) - P(\mathbf{z}_*)] \end{aligned} \tag{2.3.24}$$

Now, if we apply the inequality above repeatedly for $k = 1, ..., m$ and take the expectation with respect to previous random variables $\mathbf{z_1}, ..., \mathbf{z_m}$, then we can obtain the following inequality.

$$\begin{aligned} & \mathbb{E}\left\{\|\mathbf{z_m} - \mathbf{z}_*\|^2\right\} + 2\eta\left[\mathbb{E}\left\{P\left(\mathbf{z_m}\right)\right\} - P\left(\mathbf{z}_*\right)\right] \\ & + 2\eta\left(1 - 4\eta L_Q\right)\sum_{k=1}^{m-1}\left[\mathbb{E}\left\{P\left(\mathbf{z_k}\right)\right\} - P\left(\mathbf{z}_*\right)\right] \\ & \leq \|\mathbf{z_0} - \mathbf{z}_*\|^2 + 8\eta^2 L_Q\left[P\left(\mathbf{z_0}\right) - P\left(\mathbf{z}_*\right) + m\left(P\left(\tilde{\mathbf{z}}\right) - P\left(\mathbf{z}_*\right)\right)\right] \end{aligned} \tag{2.3.25}$$

Since $2\eta\left(1 - 4\eta L_Q\right) < 2\eta$, $\mathbf{z_0} = \tilde{\mathbf{z}}$ and P is convex, therefore $P\left(\tilde{\mathbf{z}}_\mathbf{s}\right) \leq \frac{1}{m}\sum_{k=1}^{m} P\left(\mathbf{z_k}\right)$, we can write the following inequality.

$$\begin{aligned} & 2\eta\left(1 - 4\eta L_Q\right)m\left[\mathbb{E}\left\{P\left(\tilde{\mathbf{z}}_\mathbf{s}\right)\right\} - P\left(\mathbf{z}_*\right)\right] \\ & \leq \|\tilde{\mathbf{z}}_\mathbf{s-1} - \mathbf{z}_*\|^2 + 8\eta^2 L_Q\left(m+1\right)\left(P\left(\tilde{\mathbf{z}}_\mathbf{s-1}\right) - P\left(\mathbf{z}_*\right)\right) \end{aligned} \tag{2.3.26}$$

By using the Lemma 2.3.3 which is a weaker condition then using the strong convexity and by applying the above inequality recursively, we derive the convergence rate as follows:

$$\left[\mathbb{E}\left\{P\left(\tilde{\mathbf{z}}_\mathbf{s}\right) - P\left(\mathbf{z}_*\right)\right\}\right] \leq \left(\frac{\left(\frac{2}{\ell} + 8\eta^2 L_Q\left(m+1\right)\right)}{2\eta\left(1 - 4\eta L_Q\right)m}\right)^s \left[P\left(\tilde{\mathbf{z}}_\mathbf{0}\right) - P\left(\mathbf{z}_*\right)\right] \tag{2.3.27}$$

Lemma 2.3.3. *Consider the problem of minimizing the sum of two convex functions:*

$$\min_{\mathbf{z}\in\mathbb{R}^d}\left\{P\left(\mathbf{z}\right) = F\left(\mathbf{z}\right) + R\left(\mathbf{z}\right)\right\}$$

A standard method for solving the above problem is the proximal gradient method. Given an initial point $\mathbf{z_0}$*, using the proximal mapping, which is shown below, iteratively generates a sequence that will converge to the optimal solution.*

$$\mathit{prox}_R\left(\mathbf{y}\right) = \arg\min_{\mathbf{z}\in\mathbb{R}^d}\{\tfrac{1}{2}||\mathbf{z} - \mathbf{y}||^2 + R(\mathbf{z})\}$$

Since $R(\mathbf{x})$ is a convex function, the optimal solution of above problem is also an optimal solution of the following problem using a tuning parameter μ [71][Theorem 1].

$$\min \tfrac{1}{2} \|\mathbf{z} - \mathbf{y}\|_2^2 \ s.t. \ R(\mathbf{z}) \leq \mu$$

By utilizing the optimal strong convexity condition which is a weaker condition than strong convexity [81] for a convex function R, we have the following inequality for all $\mathbf{z} \in \Omega$:

$$P(\mathbf{z}) - P(\mathit{prox}_E(\mathbf{z})) \geq \tfrac{\ell}{2} \|\mathbf{z} - \mathit{prox}_E(\mathbf{z})\|^2$$

where the prox_E is the Euclidean projection on to set E and ℓ is a positive parameter.

In summary, the strong convexity condition is discarded from the convergence analysis, so that the algorithm in [121] is applicable to more generic convex objectives. In the following section, the proposed computational phenotyping scheme using Cvx-SPCA is presented.

2.3.4 Interactive Hierarchical Phenotyping via PhenoTree

In this problem, each patient is represented by a sparse vector whose elements correspond to ICD-9 diagnosis codes. The size of the vector equals to the number of unique ICD-9 codes (dictionary size) present in the patient cohort. If a specific diagnostic group is targeted, a sub-sample of ICD-9 codes can also be used as the vocabulary. Thus, an $n \times d$ matrix represent the whole patient cohort, where n is the total number of patients in the cohort and d is the vocabulary size. The procedure of obtaining two-level hierarchical patient phenotypes using PhenoTree is explained below.

Step 1: Cvx-SPCA is applied to the whole patient population to obtain the non-zero loading values as illustrated in Figure 2.2. Clinical features corresponding to the non-zero loading values are the input dimensions which contribute to the leading principal component. Therefore, these clinical features are selected as the key features and a set of phenotypes within the population is defined as the first level of the hierarchy.

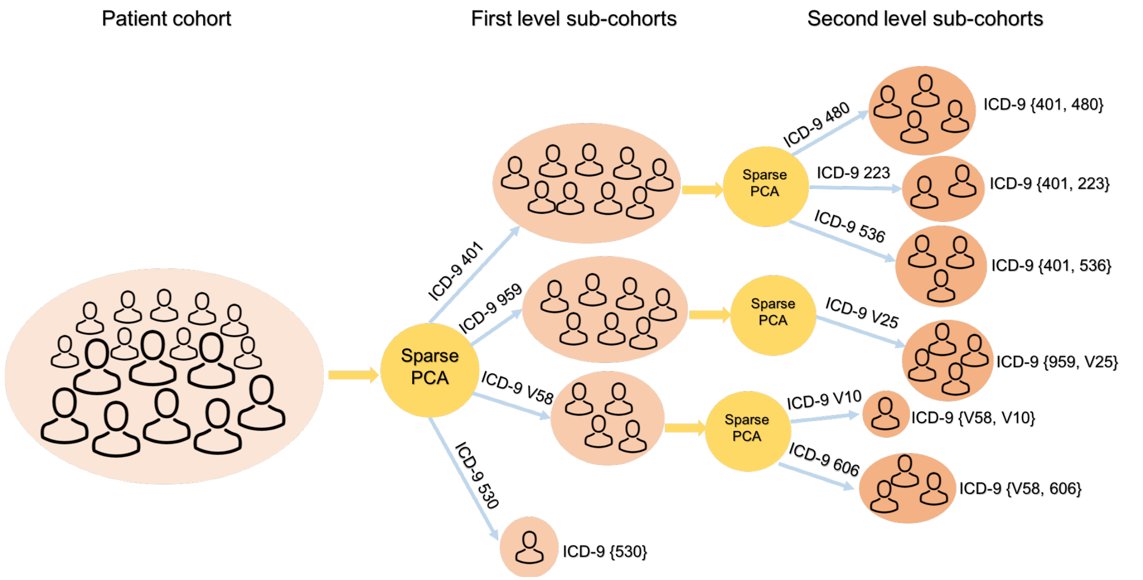


Figure 2.3 Computational phenotyping procedure with SPCA. SPCA is iteratively applied until the desired number of levels is reached.

Step 2: First level features obtained in the previous step are used to define sub-populations (patients who have the corresponding diagnosis). The number of sub-populations in the second level is equal to the number of first level features. Next, the procedure in Step 1 is applied on the sub-populations associated with each first level feature to obtain the second level key clinical features. The proposed procedure approach is summarized in Figure 2.3.

We iteratively apply the steps above to expand phenotypes and obtain a hierarchical tree structure. This structure can assist medical experts to (i) explore the diagnostic patterns in the patient cohort, (ii) automatically grow the leaves of tree by determining the number of times SPCA is applied to sub-populations, and (iii) allow physicians manually tune the phenotype hierarchy. Interpretation and analysis of hierarchical phenotypes by a domain expert would not be possible with a text based representation. To alleviate this challenge, we utilized radial Reingold-Tilford tree [17] based on the work by J. Heer and Davies to visualize the PHENOTREE. Three-level structure is given in Figure 2.4, where the levels 1,2, and 3 are shown.

Each node of the tree, such as in Figure 2.4, gives a structured phenotype and a sub-cohort characterized by this phenotype. In this structure, children nodes depend on their parent nodes since the sub-population used in children nodes is conditioned on the parent phenotypes. For

Algorithm 2 Construction of a PHENOTREE

```
Require: Data $\mathcal{D}$, solver parameters for Cvx-SPCA, number of levels is $N$
Ensure: $N$-level PHENOTREE $\mathcal{T}$ and a set of phenotypes $\mathcal{P}$
  Initialize tree $\mathcal{T} = \emptyset$
  Add pseudo phenotype to phenotype stack $\mathcal{S}$: $p_0 \rightarrow \mathcal{S}$
  forall $\mathcal{S} \neq \emptyset$ do
    Pop one phenotype $p$ from stack $\mathcal{S}$.
    if depth of $p$ is less than $N$ then
      $\mathcal{S}$ = PatientSampling $(\mathcal{D}, p)$
      Compute phenotypes of a finer level of granularity $\mathcal{P}_{(p,\mathcal{S})}$ = ExpandPhenotype $(p, \mathcal{S}; O)$

      Update $\mathcal{T}$ with phenotypes $\mathcal{P}_{(p,\mathcal{S})}$
      Push phenotypes in $\mathcal{P}_{(p,\mathcal{S})}$ to $\mathcal{S}$
    end if
  end forall
```

example, if the phenotype characterized by the diagnosis ICD-9 92 Syphilis has a parent phenotype 808 Pelvis, we denote the phenotype as ICD-9 92→808. Note that a patient may have a feature from only the first level, first two levels or from all three levels. For instance, in Figure 2.4, there are 3 patients who are diagnosed with ICD-9 373 and ICD-9 185, 32 patients who are diagnosed with ICD-9 373 and ICD-9 626, and one patient with diagnoses ICD-9 373, ICD-9 185 and ICD-9 761. Same patients may have different hierarchical phenotypes, as well. For example, one patient could simultaneously possess two phenotypes: ICD-9 373→185→761 and ICD-9 373→185. If we need to assign patients exclusively to one of the phenotypes, the deepest hierarchy is considered. Thus, PHENOTREE provides an informative and visually interactive way of phenotyping the patients by their diagnoses information. These phenotypes can be used to cluster patients or can be used as side information for classification tasks. The proposed approach to construct a PHENOTREE is given in Algorithm 2, where the subroutine `PatientSampling` selects a patient sub-population with a specific phenotype, and `ExpandPhenotype` identifies a set of phenotypes of a finer level of granularity by solving the Cvx-SPCA.

Cohort studies require interactive visualization approaches to provide insights of EHR datasets in a comprehensible way [59, 95, 114]. Otherwise, there is the risk of ignoring significant information present in patient cohorts. In PHENOTREE, phenotypes do not have to be expanded uniformly.

In the hierarchy in Figure 2.4, for example, not every key feature is expanded to the third level. In such cases, expertise of the medical researchers should be incorporated into the analysis process to expand the phenotypes further. Therefore, a visual representation is necessary to be able to involve the medical expert in the process. Another important factor is the efficiency of the approach to obtain phenotypes since the proposed approach requires to iteratively apply Cvx-SPCA on the entire patient cohort and sub-cohorts. A SPCA formulation which is not sensitive to the scale of the data is needed to avoid high computation time.

2.4 Experiments

Synthetic and real world data experiments were conducted to investigate the time complexity of the proposed Cvx-SPCA algorithm and to demonstrate a phenotyping application of the proposed PHENOTREE. In our experiments, step size η was chosen following the heuristic $0 < \eta < 1/\left(4L_Q\right)$ and L_Q was taken as the largest eigenvalue of the covariance matrix. Iteration number m was chosen as $\Theta\left(L_Q/\left(\lambda - \lambda_1\left(\mathbf{S}\right)\right)\right)$ which was suggested in [121].

2.4.1 Synthetic Dataset

Synthetic datasets used in this section were randomly generated from zero mean and unit variance normal distribution. First, the convergence of proximal stochastic gradient with variance reduction and traditional proximal stochastic gradient for convex SPCA scenario are compared. In Figure 2.5, objective value versus number of epochs are plotted for different numbers of samples (n) and features (d). Traditional proximal stochastic gradient (prox-SGD) and proximal stochastic variance reduced gradient (prox-SVRG) methods are compared. In Figure 2.5, convergence is observed when the maximum number of epochs is fixed to 50. We also investigated the number of epochs necessary for both algorithms to converge. Therefore, another experiment was conducted to see how fast Cvx-SPCA with prox-SVRG converges to a similar sparsity as Cvx-SPCA with prox-SGD. We again generated a synthetic dataset with $100,000$ instances and $10,000$ independent dimensions. As it can be seen from the result in Figure 2.6, Cvx-SPCA with traditional SGD requires more epochs than Cvx-SPCA with SVRG to converge to similar sparsity patterns.

Secondly, running times of other SPCA methods and the proposed method are compared for $1,000$ dimensional features in Figure 2.7. We ran the algorithms until they reached similar sparsity patterns. The proposed Cvx-SPCA algorithm is more scalable, since a stochastic solver is used and there is no eigenvalue decomposition or SVD steps during the optimization. For instance, [61] requires singular value decomposition at each iteration, which is a bottleneck in terms of running time, [55] uses an inverse power method based approach and [86] uses semi-definite programming. According to the experiments on randomly generated data with sample sizes of $100,000$, $500,000$ and $1,000,000$, Cvx-SPCA is observed to handle large datasets better than the baseline methods.

We also investigated the regularization path for the proposed algorithm. Regularization path illustrates changes in solution with varying regularization parameter γ which specifies the level of sparsity. In order to have a suitable level of sparsity, γ should be tuned. One common approach to find an appropriate γ is the regularization path. For this purpose, we first generated a random sample with 10 features and applied the proposed Cvx-SPCA algorithm to obtain the first principal component. Then, the covariance matrix was reconstructed by using the first principal component corresponding to the largest eigenvalue with random noise. Loading values of the principal component were computed with varying values of the regularization parameter γ by using the reconstructed covariance matrix. We started with small γ values, and the loading vector learned from the previous step is used as the initialization for each new Cvx-SPCA step. Results are given in Figure 2.8 where the values of 10 features are represented by different colored curves. The known principal component can be recovered through the path which confirms that this is a valid regularization path. When the regularization term was around -0.11 (dashed vertical line), the non-zero loading values of the known principal component, which was used to generate the data, are recovered.

Table 2.1 We sample patients with female, male, child and old age specific diagnoses. These samples may overlap with each other. For instance, a patient may have dementia and a prostate diagnosis together. We did not include medical conditions which can be encountered for any age patient and both genders into these groups of patients. Old patients are assumed to be above 60 years old. The age range of child patients are determined between infants (birth to 1 year) to adolescence (12-18).

Demographic	Number of features	Number of patients
Female	1,268	130,035
Male	106	24,184
Old	66	2,060
Child	596	38,434

2.4.2 Private Electronic Health Records Dataset

We used a private, large scale EHR dataset comprising of $223,076$ patients and $11,982$ diagnoses over a time span of 4 years. Each diagnosis was represented by ICD-9 codes. Explicit demographic information of the patients and their admission/readmission times were not available. However, some of the ICD-9 codes have particular terms which indicate gender and age of the patients. For instance, diagnoses codes implying problems in pregnancy, female/male genital organs, conditions having the term senile in their explanations can be used to group patients as female/male, young and old. On the other hand, there was some observed anomalies such as patients having records for both female and male specific diagnoses or for both newborn and senile. Since we did not take a part in data collection, the reason of these anomalies could not be resolved. Therefore, these kind of patients were eliminated in our experiments. There were also patients who have very few records in the dataset. Patients who have fewer than five records were also discarded. After data cleaning, the total number of patients retained was $168,431$ from the initial pool of $223,076$ patients. In Table 2.1, statistics about female, male, old and child patients sampled considering the definitions of ICD-9 codes is summarized.

The age range of child patients was determined ranging from infants (birth to 1 year) to adolescence (12-18) and old patients were defined to be above 60 years old. We should note that there may be female, male, old and child patients who were not included into these demographic groups such as patients with diagnoses which are not gender or age specific. The numbers given in Table 2.1 do not give a clear idea about the percentages of age and gender groups in the EHR dataset. For instance, diseases such as hypertension and Alzheimer were commonly encountered among the people above a certain age. However, these problems have been lately observed in younger patients. Each patient has a sparse feature vector where the i-th value gives the frequency of the i-th diagnosis code for the corresponding patient. As discussed before, each ICD-9 code corresponds to one diagnosis and each diagnosis belongs to a hierarchy. For instance, code 278 is Obesity, 278.01 is Morbid Obesity, and 278.02 is for Overweight. Thus, there are sub-groups under the main diagnosis. In our experiments, all the sub-groups related to a particular diagnosis were aggregated. As a result, the feature dimensionality was reduced from $11,982$ to 927.

We conducted several experiments to visualize the structure of the patient population using the procedure explained in the previous section. A three-level PHENOTREE was generated for the general patient cohort as shown in Figure 2.10, where the ICD-9 description of each disease is also included. The following relationship between layers can be inferred from the figure. If we look at the output features of the patients who have The diagnosis ICD-9 239, Neoplasm Of Unspecified Nature, in the first level, we can see ICD-9 176, Karposi's Sarcoma, ICD-9 196, Secondary and Unspecified Malignant Neoplasm of Lymph Nodes, ICD-9 693, Dermatitis Due To Drugs, ICD-9 702, Other Dermatoses, and ICD-9, 957 Injury to Other and Unspecified Nerves. Corresponding branches of the PHENOTREE can be seen in Figure 2.9. Karposi's Sarcoma is known as a type of cancer. Unfortunately, neoplasms, in other words, abnormal growth of tissue can spread out to different parts of the body. Therefore, patients who have diagnosis of neoplasm of unspecified nature may have other types of neoplasms as well. In addition, we can also see dermatological problems in the second level. Cancer treatments such as chemotherapy and radiation therapy can have dermatological side effects such as radiation dermatitis. Another observation was the ICD-9 344 Paralytic Syndromes, whose second level diagnoses were obtained as ICD-9 669 Complications of Labor and Delivery, 744 Congenital Anomalies of Eye, Face and Neck, 820 Fracture of Neck of Femur and ICD-9 891 Open Wound of Knee, Leg and Ankle. Paralytic conditions are not commonly known to occur during birth delivery. However, methods like epidural may have

Table 2.2 EHR data features which contributes to the output dimensions after Cvx-SPCA algorithm was applied to the whole patient population (168,431 patients). Feature descriptions were provided by the private dataset and they can also be found in [44].

ICD-9 Code	Description
7	Balantidiasis/Infectious
72	Mumps Orchitisn/Infectious
115	Infection by Histoplasma Capsulatum
266	Ariboflavinosis/Metabolic disorder
507	Pneumonitis/Bacterial
695	Toxic Erythema/Dermatological
697	Lichen Planus/Dermatological
761	Incompetent cervix affecting fetus or newborn
795	Abnormal glandular papanicolaou smear of cervix
924	Contusion of thigh/Injury

the risk of paralysis. On the other hand, there are features related to neck or femur, whose serious injuries can be a reason for paralysis. In the general cohort, fractures and injuries were commonly encountered diagnoses. In addition, neoplasms, infectious diseases, and problems of newborns caused by complications of mothers were also observed frequently. In Table 2.2, ICD-9 codes and corresponding definitions of commonly observed conditions are given. Feature descriptions were provided by the private dataset and they can also be found in [44].

In addition to the general cohort, hierarchical structure of different patient groups in terms of age and gender, as given in Table 2.1, are also investigated. Hierarchical representations of female, male, child, and old patient groups are given in Figures 2.11, and 2.12, respectively. Aforementioned sub-groups yielded their specific diagnoses as well as frequently encountered conditions in the general cohort. For instance, one of the first level features in Figure 2.11a is ICD-9 636, Illegal Abortion and its second level features contained ICD-9 596, Disorders of Bladder, and 37, Tetanus. Abortion under unhygienic conditions and in underground clinics is known to have health risks such as infections and urinary tract disorders that align well with the SPCA second level features. If we look at the sub-group of old patient population in Figure 2.12b, we observe diagnoses such as dislocation and fracture of bones. People older than a certain age such as 80 commonly suffer from fractures especially in femur and pelvis. For example, ICD-9 821 fracture in femur is one of the first level features representing the old patient group. In the second level of ICD-9 821, diagnoses such as ICD-9 268 Vitamin D deficiency, 332 Parkinson's Disease, some infectious diseases and ICD-9 701 skin disorder were obtained. These diagnoses are known to be commonly encountered among old patients. As a summary, our results show that the proposed method can be used to visualize, interpret the relationships between sub-groups, and consequently enable constructing a phenotype tree of patients via the obtained hierarchical structure.

2.4.3 Diabetes Dataset

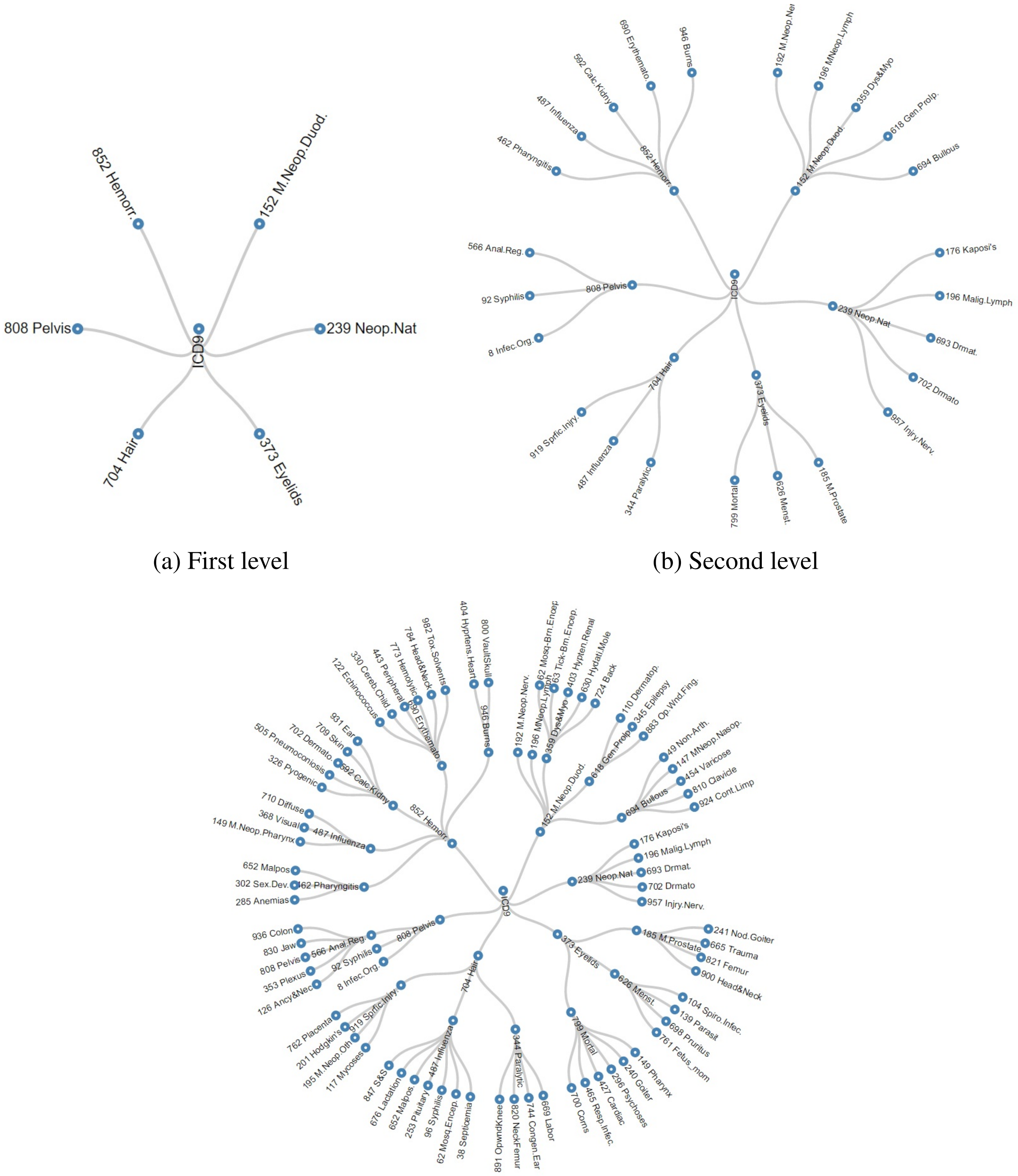
We conducted patient phenotyping experiments on a publicly available EHR dataset which represents clinical data about diabetes collected between (1999-2008) at 130 US hospitals [107]. There are $101,767$ records in total in this dataset and each patient has 50 features representing race, gender, age, number of medications, test results, diagnoses and so on. There is an age range for each patient, and age represented in the figure corresponds to the upper limit of the associated range, so 10 indicates the range [0,10). Each patient has 3 diagnoses with corresponding ICD-9 codes. In the experiments, patients were represented by 729-dimensional feature vectors, where the first 697 dimensions corresponded to the multi-hot representation (sparse binary vector) associated with 697 unique ICD-9 codes present in the dataset. Rest of the features represent number of times patient was admitted in hospital, lab procedures, number of medications, and some test results about diabetes which have binary values. In addition, the readmission status, which include not readmission, readmission before 30 days (< 30) and after 30 days (> 30) [107], was also given in the dataset. This information was utilized to group patients as readmitted and not readmitted to see how the hierarchical structures change.

The same procedure as on private EHR dataset was followed such that the features belonging to the same ICD-9 hierarchy were aggregated (resulting in 697 features in total). Hierarchical structure of the whole patient population is given in Figure 2.13. First level of features were obtained as insulin and ICD-9 diagnoses such as neoplasm, heart disease, hormonal problem and so on. Existence of the insulin among the output features indicates diabetes. If we further examine the diagnoses obtained from the patient groups who were prescribed insulin, we can see a wide range of diagnoses such as kidney problems, disorders of stomach, bacterial infections, and disorders of adrenal glands. Stomach problems are also commonly encountered among diabetes patients because of medications.

Similarly, readmitted (patients readmitted before and after 30 days of discharge) and not readmitted patients were examined as shown in Figure 2.14. The readmission is not only relevant for medical purposes but also for insurance companies [107]. Our interest was to investigate how the types of diagnoses and the hierarchical structure of readmitted patients differ from the patients who were not readmitted. We should emphasize that the same regularization parameter was used for both patient populations. According to our experiments, it was not possible to distinguish these two populations by looking at the types of output diagnoses. For instance, diseases which may require the patient to get medical attention regularly such as cancer are encountered in both groups of diabetes patients. However, it was observed from Figures 2.14a and 2.14b that, readmitted patients produced more nodes in the second level. This observation was interpreted as readmitted patients having several records for different diagnoses. Therefore, we could sample enough patients with specific diseases compared to not readmitted patient population, while we were constructing the levels. Graphs of female, male, old, teen and adult patients can also be seen in Figures 2.15 and 2.16, respectively. As a summary, we can see that exploring the insights and interpreting the findings about the EHR data visually is possible by using the proposed PHENOTREE approach. The proposed system can be helpful for clinical decision support systems since it aids physicians to understand diagnoses and sub-cohort relationships in a visually interactive way.

2.5 Summary

In this study, a hierarchical phenotyping approach based on SPCA to analyze and visualize diagnostic patterns in EHRs is introduced. We propose to use a convex version of SPCA problem which allows us to employ proximal stochastic variance reduced gradient methods alleviating low convergence due to the variance of random sampling in traditional stochastic algorithms. Experiments on both synthetic and real world datasets were conducted to evaluate the convergence properties of the proposed formulation. Patient phenotyping results showed that proposed framework might actually assist medical experts to understand and analyze the patient populations and the relationship between them.



(a) First level

(b) Second level

(c) Third level

Figure 2.4 An example of hierarchical phenotyping with stochastic Cvx-SPCA. (a) the first, (b) the second and (c) the third level features of the patient population. This procedure can be applied repeatedly until the desired number of levels is reached.

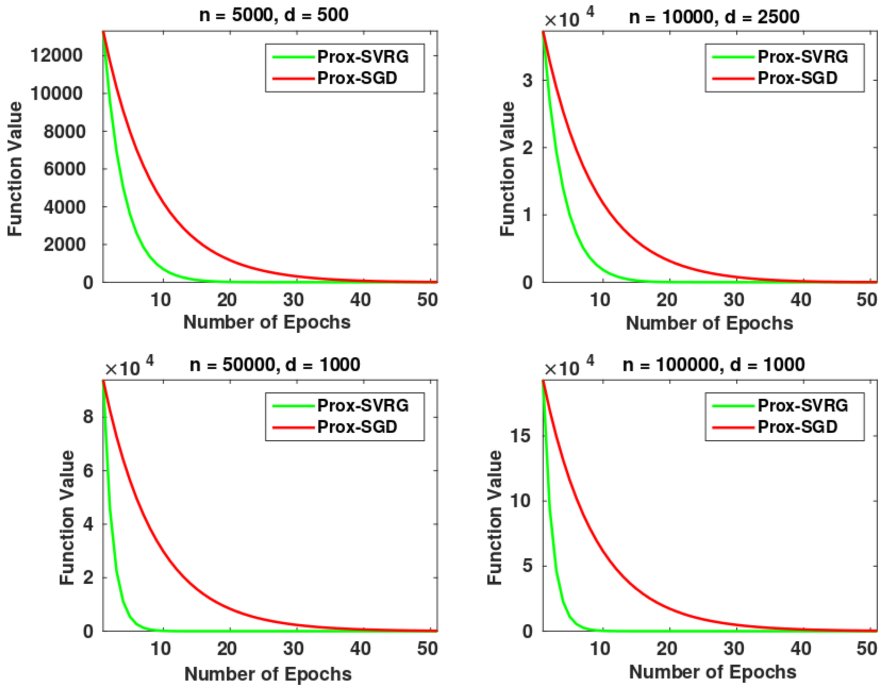


Figure 2.5 Convergence for synthetic data with n samples and d features. Convergence of the proposed stochastic Cvx-SPCA with (prox-SVRG) and without variance reduction (prox-SGD). Proximal stochastic gradient with variance reduction has a faster convergence rate, since the variance caused by random sampling is bounded in prox-SVRG.

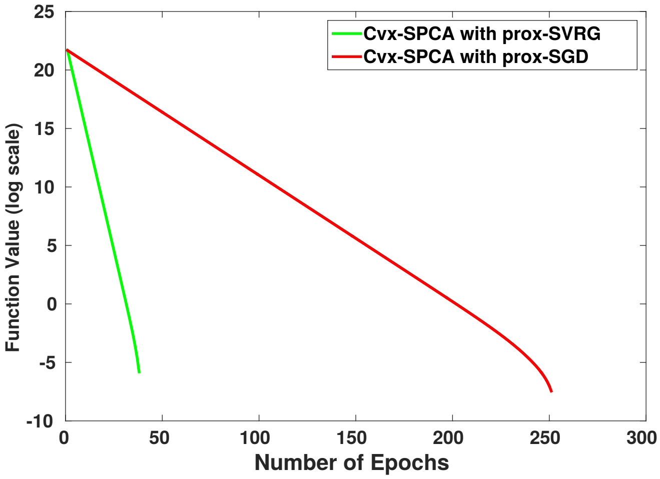


Figure 2.6 Convergence of sparse pattern in the log scale. Cvx-SPCA with Prox-SGD takes 275 epochs, whereas Cvx-SPCA with Prox-SVRG takes 45 epochs to converge a similar sparsity pattern.

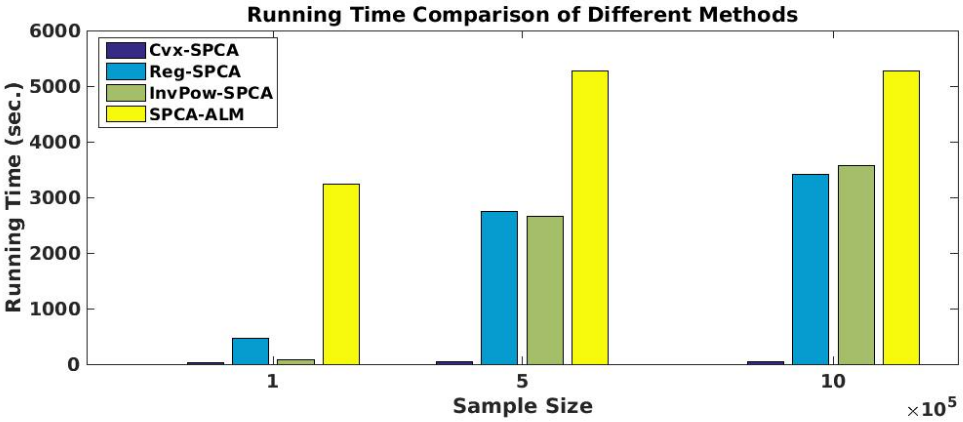


Figure 2.7 Running times (in seconds) are compared to obtain similar cardinality of loading vector. A machine with 2.8GHz Intel(R) Xeon(R) CPU and 142 GB memory was used in the experiments. The methods denoted by Reg-SPCA, InvPow-SPCA, and InvPow-SPCA are from [55, 61], and from [86], respectively.

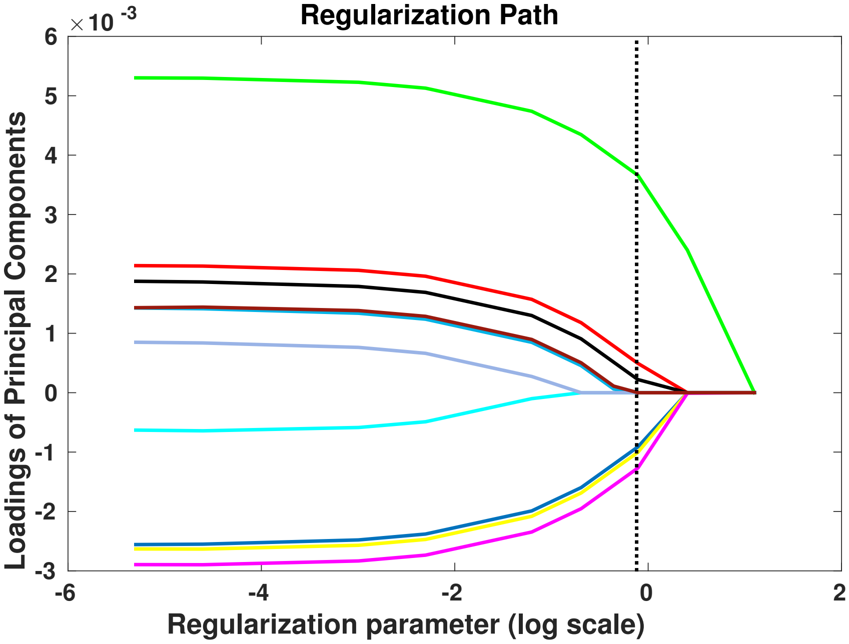


Figure 2.8 Regularization path for Cvx-SPCA, where different colored curves represent the values of 10 features. When the regularization term was around -0.11 (dashed vertical line), the non-zero loading values of the known principal component, which was used to generate the data, are recovered.

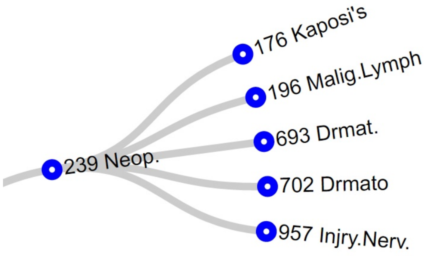


Figure 2.9 An example branch of the PHENOTREE. The first level diagnosis with ICD-9 code 239 denotes Neoplasm Of Unspecified Nature. Children nodes of the patients who have ICD-9 239 are ICD-9 176 Karposi's Sarcoma, 196 Secondary and Unspecified Malignant Neoplasm of Lymph Nodes, 693 Dermatitis Due To Drugs, 702 Other Dermatoses, and ICD-9 957, Injury to Other and Unspecified Nerves. Karposi's Sarcoma is a type of cancer. Patients who have diagnosis of neoplasm of unspecified nature may have other types of neoplasms as well. In addition, we can also see dermatological issues in the second level.

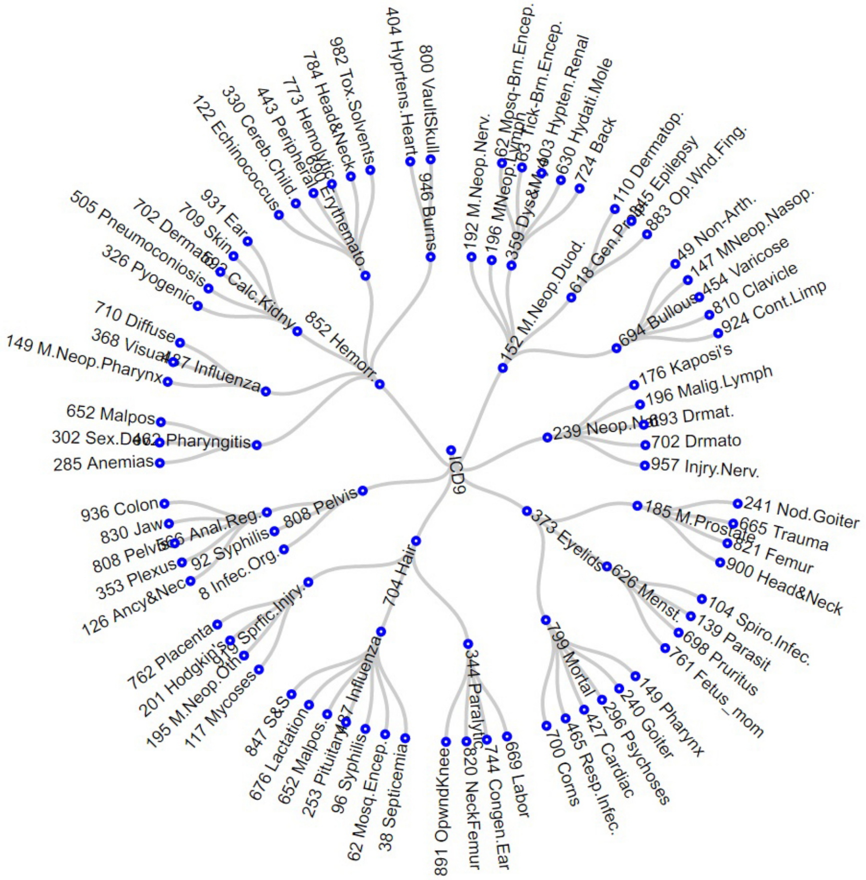
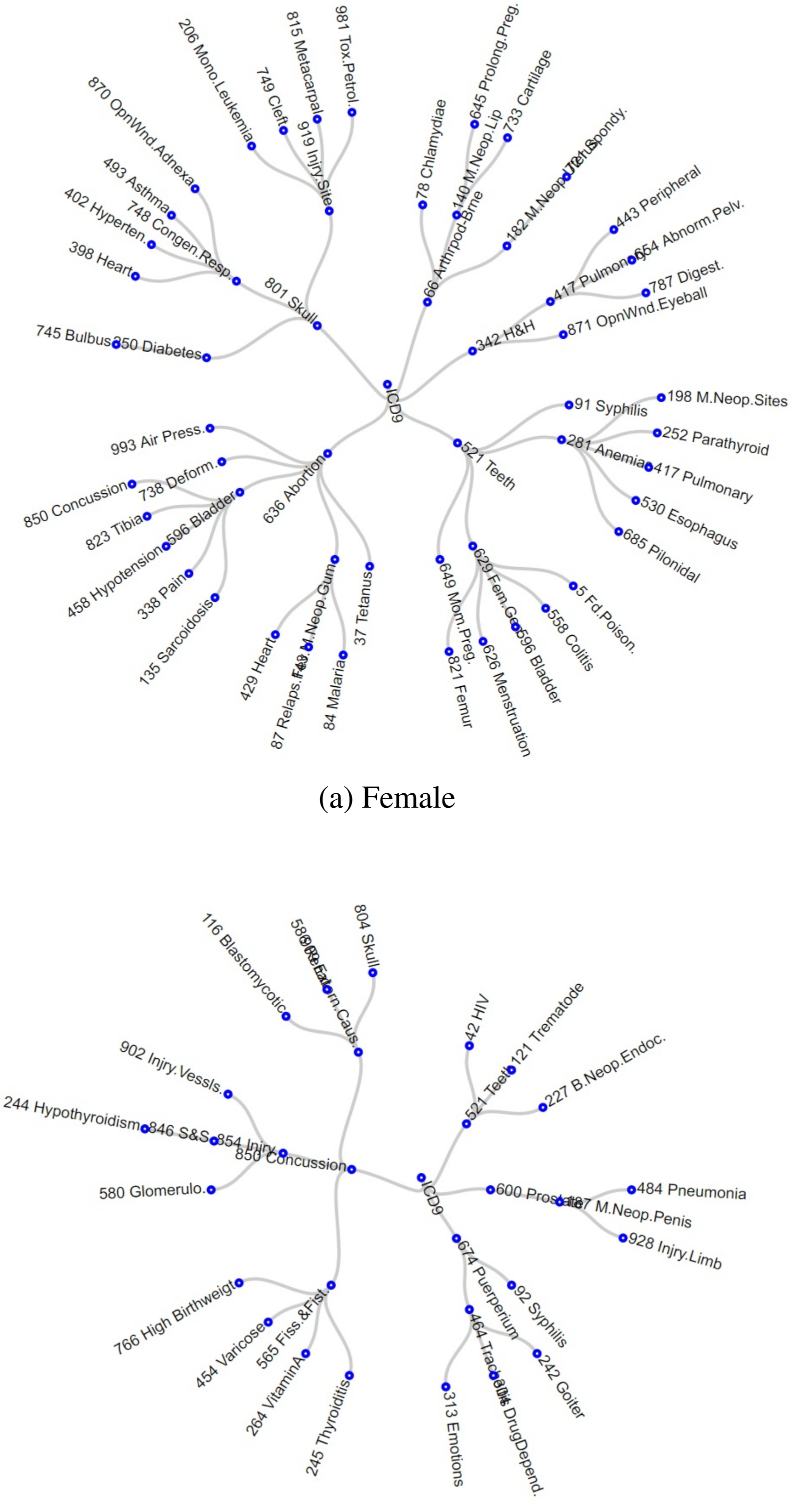


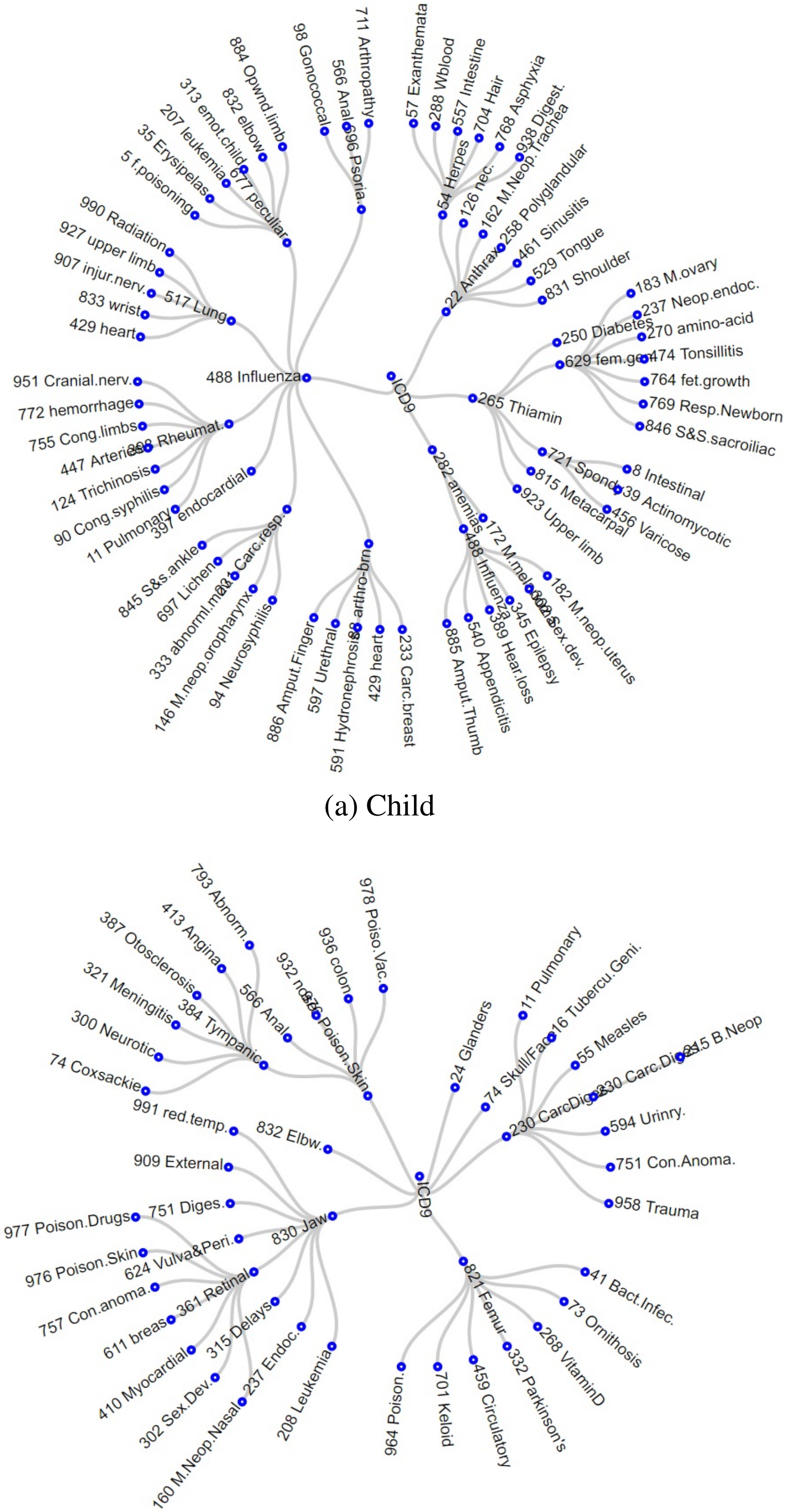
Figure 2.10 Hierarchical stratification via Cvx-SPCA. Cvx-SPCA is applied on the entire patient population and the features with the largest absolute loading values on the leading principal component are selected. Each feature dimension corresponds to a specific disease.



(a) Female

(b) Male

Figure 2.11 Hierarchical stratification via Cvx-SPCA for female and male patients. One of the first level features in Figure 2.11a is ICD-9 636, Illegal Abortion. In the second level, we see diagnoses like ICD-9 596 Disorders of Bladder, and 37 Tetanus, which could be some of the side effects of illegal abortion.



(a) Child

(b) Old

Figure 2.12 Hierarchical stratification via Cvx-SPCA for children and old patients. According to our observations, different patient groups tend to have common diseases which was illustrated for general population before. On the other hand, they also yielded specific diagnoses as well.

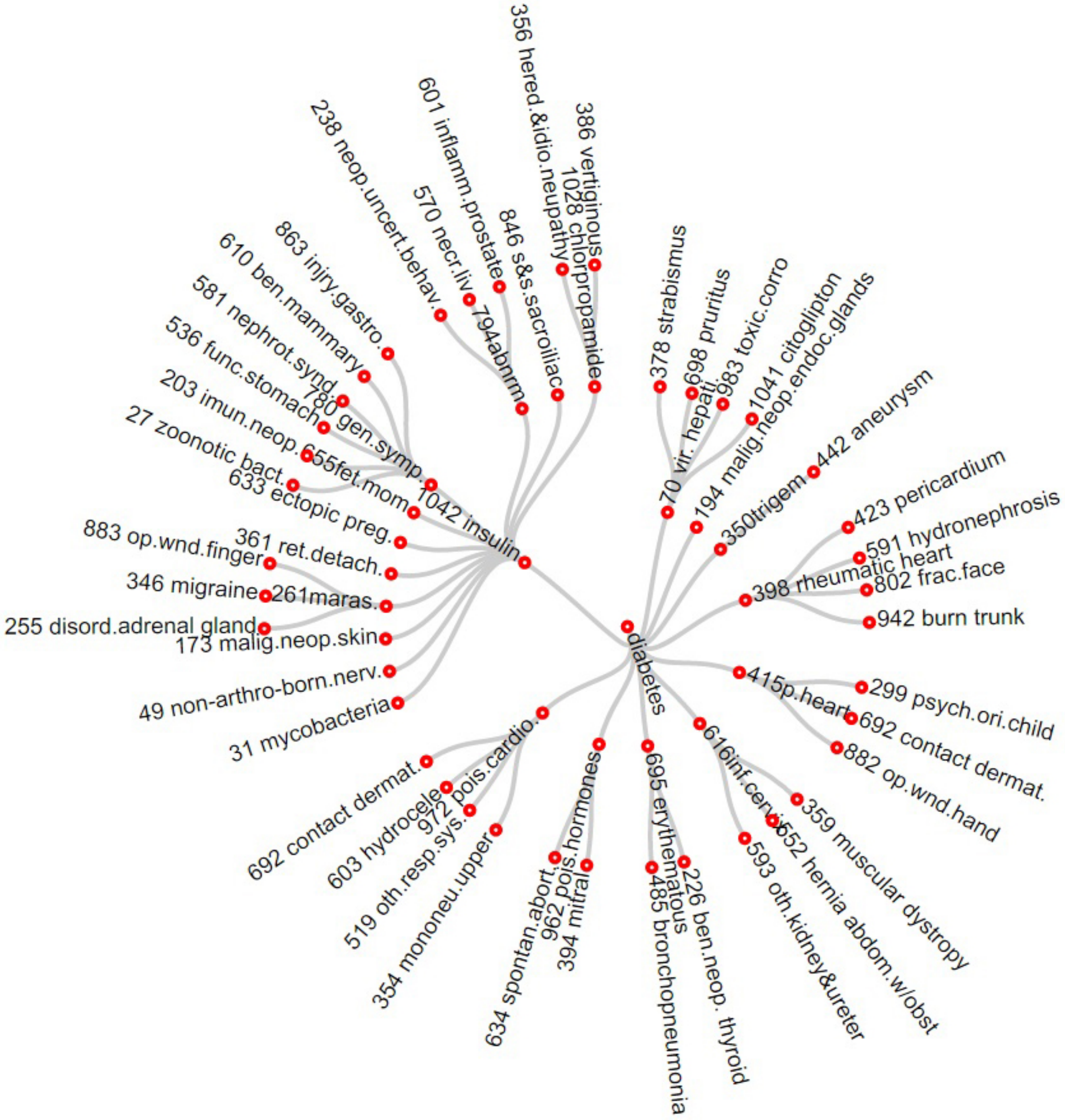
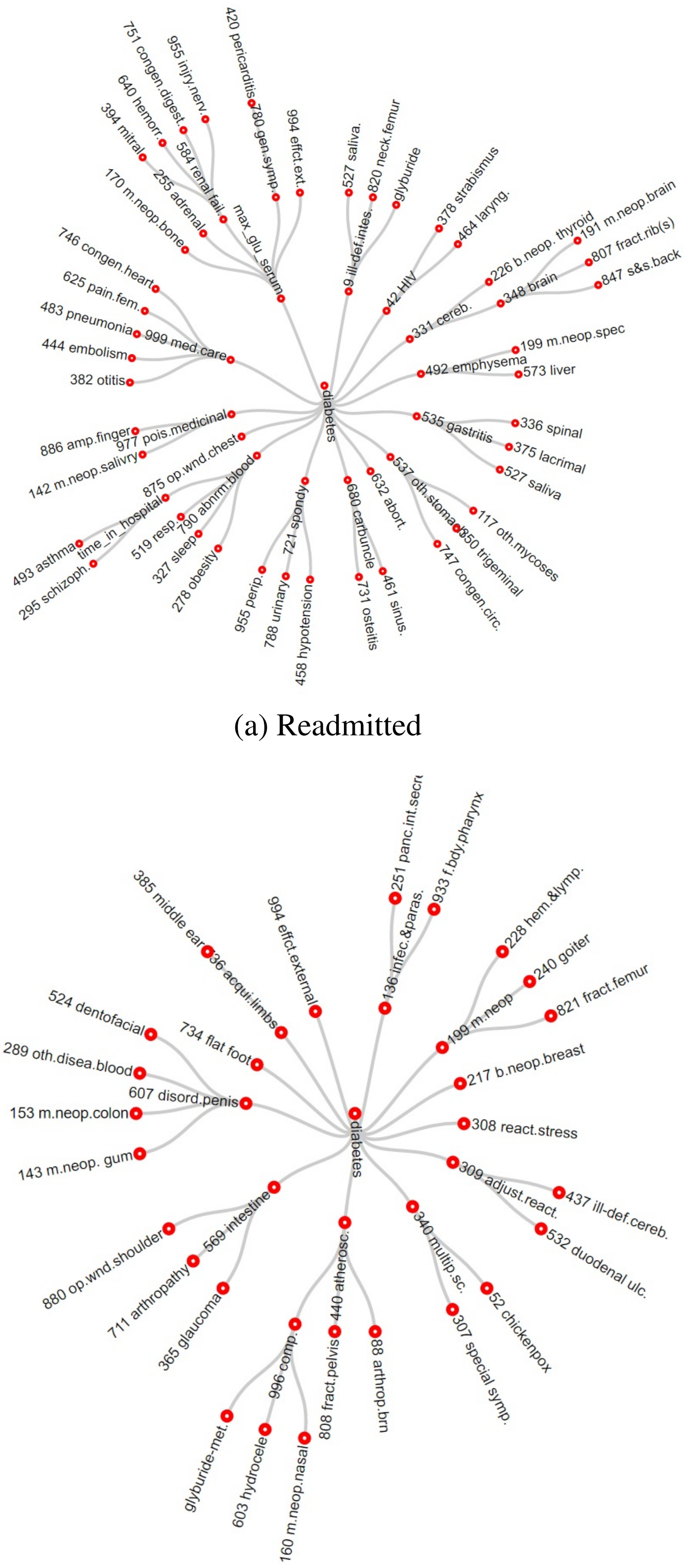


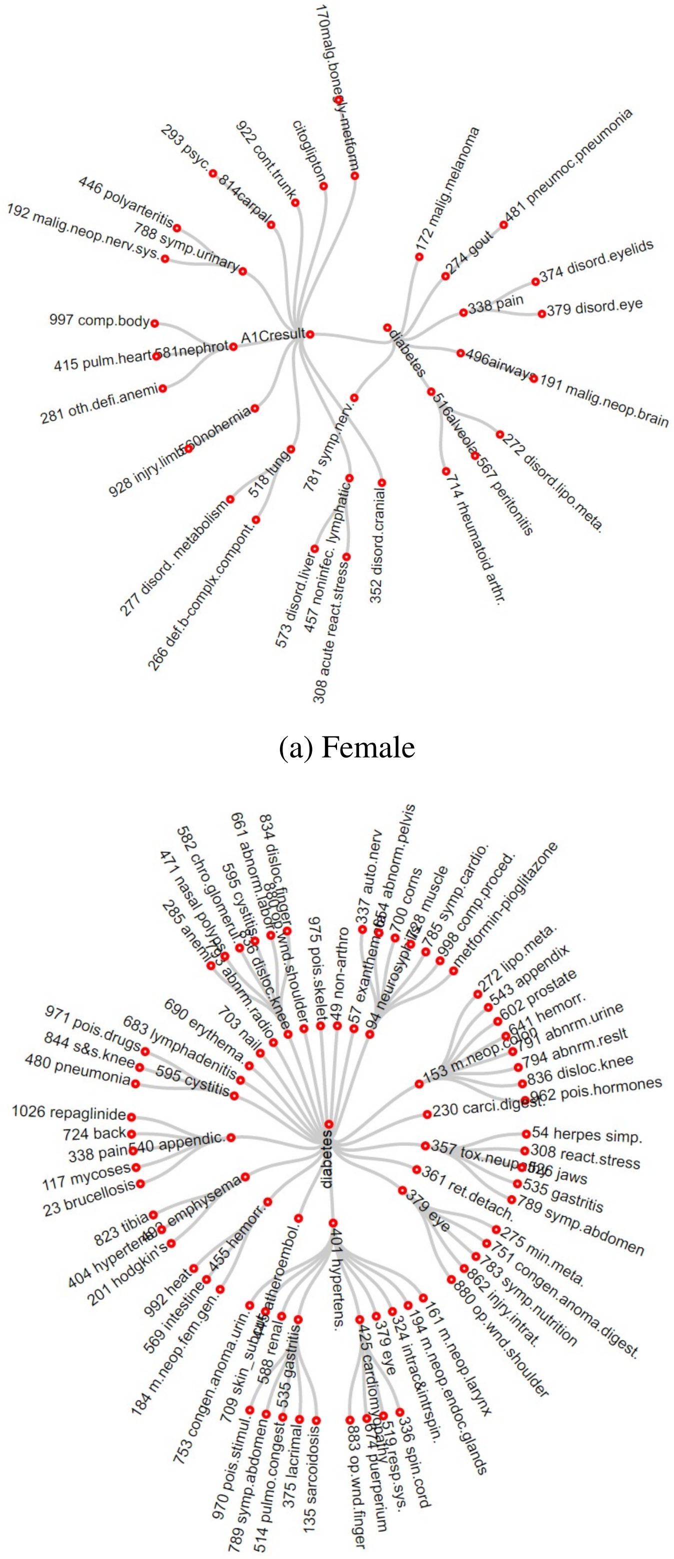
Figure 2.13 Hierarchical representation of the whole diabetes data.



(a) Readmitted

(b) Not Readmitted

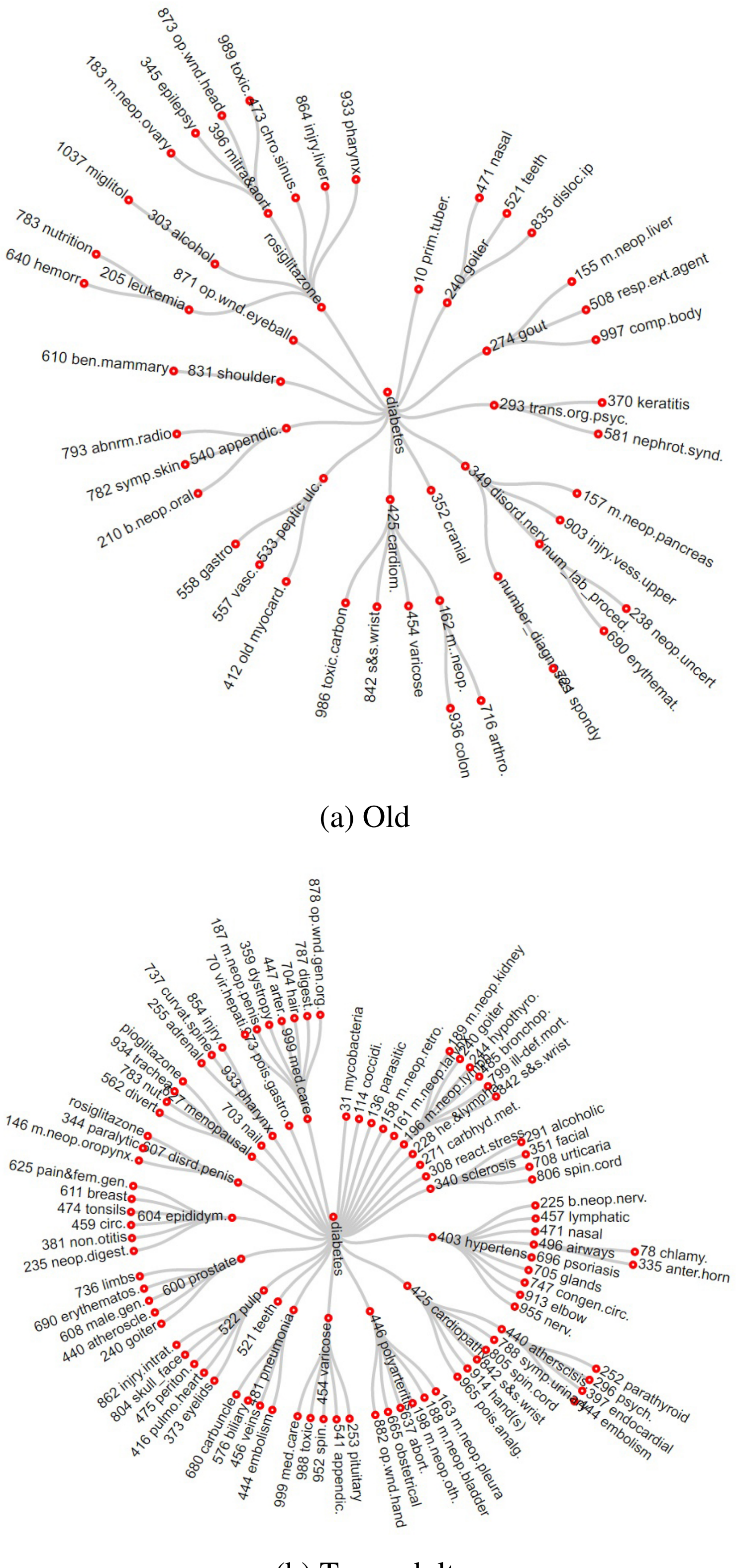
Figure 2.14 Hierarchical Stratification via Cvx-SPCA for patients who were readmitted and not readmitted in diabetes database. Injury and poisoning are commonly encountered for not readmitted patients. However, wider range of diagnoses are observed for readmitted patients.



(a) Female

(b) Male

Figure 2.15 Hierarchical Stratification via Cvx-SPCA for female and male patients in diabetes data. We can observe female/male diagnoses along with common diseases. For instance, Figure 2.15b has ICD-9 602, disorder of prostate, and ICD-9 401, hypertension.



(a) Old

(b) Teen-adult

Figure 2.16 Hierarchical Stratification via Cvx-SPCA for old and teen/adult patients in diabetes data. Neoplasm is very common in patient populations with a wide range of age or different genders for both the EHR datasets examined in this study.

Chapter 3

Asynchronous Distributed Multi-Task Learning

In the previous chapter, we tackled computational patient phenotyping, which was an unsupervised learning problem. Our goal was to extract key clinical features from a large scale patient cohort, and consequently obtain patient sub-groups with similar diagnostic patterns. Time efficiency and the interpretability were two important challenges of the computational phenotyping task. For this reason, a convex sparse principal component analysis technique with fast convergence properties was proposed. In this chapter, we address distributed EHR challenge in biomedical informatics. In this study, we investigate learning supervised predictive models from EHR datasets located in different geographical regions. Predictive models are commonly designed for healthcare related tasks, such as mortality prediction and disease prediction. Based on the application, predictive task can be either classification or regression. Regardless the type of the task, predictive modeling assumes that each data point in the training set is drawn from the same probability distribution. On the other hand, EHR datasets collected in different locations have different patient distributions. For this reason, distributed EHR datasets cannot be combined into one large dataset to learn the predictive models although the predictive tasks are similar. In such cases, multi-task learning (MTL) is used to learn a separate model for each EHR dataset and leverage their shared knowledge to improve the generalization performance. However, a standard MTL formulation requires data centralization which is not possible for EHR datasets due to privacy concerns and limited bandwidth. Distributed MTL addresses the data centralization issue by only transferring the model over a network. However, the standard distributed MTL approaches are synchronized, which is required by the optimization schemes. Synchronized frameworks are not robust against network delays and failures. In this chapter, an asynchronous distributed MTL framework is introduced to address the aforementioned challenges. The asynchronous nature of the framework prevents the optimization scheme from impeding due to network delays and failures.

3.1 Introduction

Many application domains require learning predictive classification or regression models for multiple *tasks*. These tasks are not always independent of each other. In fact, multiple tasks are usually related to each other such as predicting diagnostic outcomes for different types of diseases. In this problem, a single model cannot be learned by utilizing a combination of heterogeneous individual datasets where the data distribution is not the same. However, the overall goal of the individual tasks is usually similar. This indicates that there is a shared knowledge between individual tasks. Multi-task learning (MTL) simultaneously learns the related tasks and performs an inductive knowledge transfer between them to improve the generalization performance. The idea behind MTL is to learn high performance models when there is not enough data for a single task by sharing the predictive information among the related tasks.

There are several types of MTL approaches depending on the knowledge transfer technique. The most commonly studied MTL approach is regularized MTL where the task relatedness is modeled by adding a regularization term to the general loss function. The purpose of the regularization term is usually to couple the individual models via a matrix and enforce a requirement that fulfills the knowledge transfer. The advantage of the regularized MTL is the ability to work with various loss functions, such as least squares, logistic regression, and hinge loss. The regularization term is

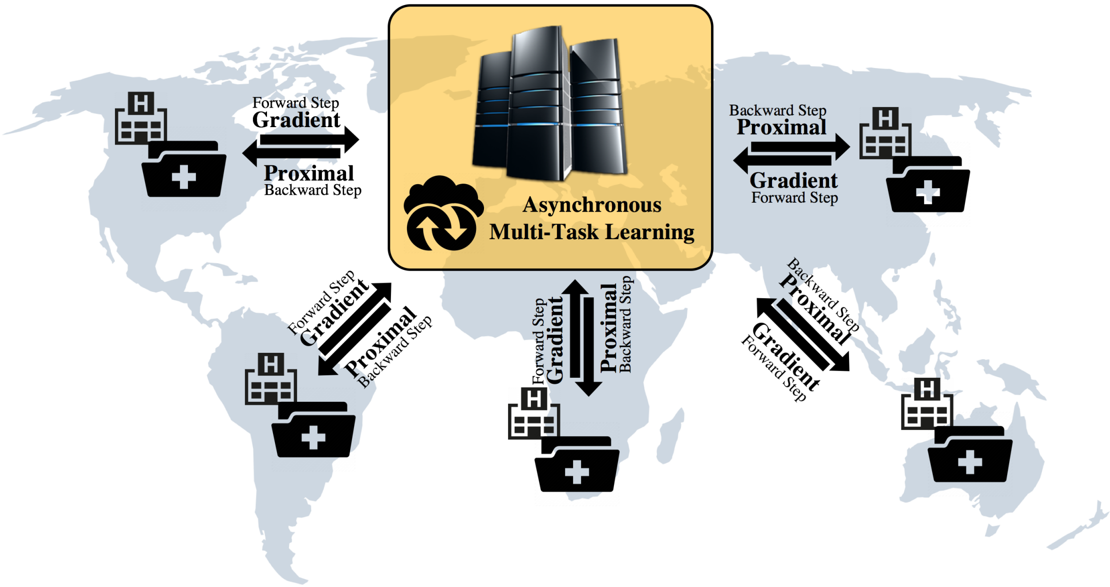


Figure 3.1 Overview of the proposed asynchronous multi-task learning (AMTL) framework. After a task node computes the gradient, it sends the gradient or the updated model to the central server. Central server couples all the individual models to perform proximal projection and send the updated individual models back to the corresponding task nodes.

usually a non-smooth function that makes the task a composite optimization problem. Such composite optimization problems are solved with iterative proximal gradient descent approach, where each iteration is divided into gradient and proximal update steps. Proximal step usually couples the related tasks based on the type of the regularization.

Traditional MTL assumes that data is centralized during optimization. On the other hand, data centralization is not always feasible since single task data can be located in different geographic regions and considered private. As in healthcare domain, each hospital stores its own EHR database and the patient information is very sensitive. For this reason, EHR data is not transferred over a network. In addition, the data transfer is time consuming due to the EHR data volume and limited bandwidth among the nodes. In such cases, distributed optimization techniques facilitate a solution for MTL problems, where each task node is located in a local server. In the distributed optimization scheme, the local server performs the gradient descent and sends the updated intermediate solution to a central server. The proximal step is performed by the central server, where the several tasks are coupled and the final solution is obtained. Proximal projection depends on synchronized gradient information from all the task nodes. In other words, central server, where the proximal step is performed, needs to wait for all the task nodes to finish their computations. This framework can be quite slow due to network communication delays, load imbalance across the task nodes, and different hardware specification of the local machines.

In this study, we propose an asynchronous distributed MTL framework with a linear convergence rate for convex MTL formulations under mild assumptions [12]. An overview of the proposed framework is given in Figure 3.1, which presents a more robust approach against network infrastructures with high communication delays between the central server and the task nodes compared with the synchronous distributed MTL. The framework is capable of solving most of the existing regularized MTL formulations. As a case study, low-rank MTL formulation is elaborated which transfers knowledge via learning a low-dimensional subspace of task models. Experiments on both synthetic and real-world datasets were conducted to evaluate the proposed framework.

3.2 Literature Review

Distributed MTL utilizes distributed optimization techniques. In particular, regularized distributed MTL requires a distributed proximal gradient descent approach. In this section, related work covering distributed optimization and distributed MTL concepts are presented.

3.2.1 Distributed Optimization

Distributed optimization techniques facilitate the solution of massive optimization problems using hardware advancements. One popular and commonly used distributed optimization approach is alternating direction method of multipliers (ADMM), which was first proposed in the 1970s [18]. Boyd *et al.* defined ADMM as a well suited distributed convex optimization method. In this approach, local copies of the solution are introduced for local sub-problems, and the work nodes and the center node communicate to reach a consensus [18]. Therefore, ADMM is a synchronized distributed optimization instance. Although ADMM was proposed for large-scale distributed optimization setting, the number of iterations increases due to the requirement of local copies to achieve the same accuracy. On the other hand, Iutzeler *et al.* proposed an asynchronous approach using randomized ADMM based on randomized Gauss-Seidel iterations of Douglas-Rachford splitting (DRS) [63]. In the proposed asynchronous distributed setting, the overall cost function comprises of individual cost functions of a set of network agents and the objective imposes a consensus for the minimizer of the overall cost function [63].

Aybat *et al.* introduced an asynchronous distributed optimization approach for proximal gradient method [8]. Authors used randomized block coordinate descent to minimize composite functions with smooth and non-smooth components. For this purpose, an asynchronous extension of synchronous distributed first-order augmented Lagrangian (DFAL) algorithm is introduced [8]. Liu *et al.* similarity proposed an asynchronous stochastic proximal coordinate descent approach [81], but they allowed an inconsistent read mechanism. In this mechanism, elements of the optimization variable are updated by one core while being read by another core in a multicore processor setting. Authors [81] reported a linear convergence rate with a suitable step size under optimal strong convexity assumption. Peng *et al.* proposed a more general asynchronous parallel framework, named ARock, for coordinate updates based on fixed-point problems with non-expansive operators [38, 94]. Many commonly known optimization algorithms such as gradient descent, proximal gradient descent, ADMM, and primal-dual method can actually be formulated as non-expansive operators. For this reason, ARock [94] can be applied to a general spectrum of optimization problems. The approach [94] converts the problem into a fixed-point problem with a non-expansive operator and applies the ARock framework.

3.2.2 Distributed Multi-Task Learning

In many real-world MTL problems, such as predictive modeling using EHRs, data is distributed across different geographical regions. In this scenario, there are two main challenges, namely limited network bandwidth and data privacy. Data transfer over a network is very costly due to network limitations. More importantly, distributed data might have sensitive information, such as patient records. In such cases, data transfer will not be allowed even with encryption. For this reason, it is necessary to achieve the knowledge transfer among distributed datasets without sharing the raw data. Distributed MTL techniques, where data is not needed to be transferred to a central node, have been proposed to address the aforementioned challenge. Since only the learned model, which is usually a single vector (of model parameters), is transferred instead of the whole training data, the cost of network communication is much lower. Dinuzzo *et al.* proposed a client-server regularized MTL, where multiple learning tasks due to distributed datasets are simultaneously learned [36]. In this setting, each client has access the information content of the data on the server without seeing the raw data.

Mateos-Núñez and Cortéz also introduced distributed optimization techniques for MTL [85]. In their approach, objective function contains a separable convex function and a joint regularization to impose low rank solutions. Thus, their problem setting is separable on local decision variables. In fact, local gradient computations are distributed into a network of agents. Authors also proposed a second solution where a separable saddle-point reformulation is introduced through Fenchel conjugation of quadratic forms. Jin *et al.*, on the other hand, introduced a collaboration between local and global learning for distributed online multiple tasks [65]. Their method learns individual models for streaming data, thus the distributed MTL and online learning are combined. Local and global learning are performed alternately in their framework such that the first step is the online learning on local clients and the second step is the global learning performed by the server. However, a subset of raw data is still transferred between clients and the global server [65].

In 2016, Wang *et al.* proposed a shared-subspace MTL in a distributed multi-task setting [115]. Their framework comprises of several separate machines, where each machine is responsible for one task and has access to only the data of the corresponding task. Similar to the studies discussed thus far, a central node transfers the updated models to their associated machine. However, optimization requires synchronous updates, such that the central node has to wait for all the machines to finalize their computations. In summary, the reviewed studies usually follow synchronized approaches which can make the optimization very slow when there is data imbalance, network communication issues and different hardware specifications of local machines. For this reason, in our study, we focus on an optimization framework which can perform asynchronous updates thereby avoiding network delays.

3.3 Distributed Multi-Task Learning

In this section, details of distributed MTL techniques are presented. First, the motivation behind the regularized MTL and its mathematical model are revisited. Then, the synchronized MTL mechanism is discussed in detail. Finally, the proposed asynchronous MTL approach is presented.

3.3.1 Regularized Multi-Task Learning

MTL provides a principled way of simultaneously learning multiple models for related tasks to improve the generalization performances. Let's assume there are K supervised learning tasks and note that vectors are denoted by bold lower case letters and matrices are denoted by bold capital letters. Each task has its own training data denoted by $\mathcal{D}_k = \{\mathbf{x}_k, \mathbf{y}_k\}, \quad k \in \{1, 2, \ldots, K\}$. Here, $\mathbf{x}_k \in \mathbb{R}^{n_k \times d}$ is the feature matrix of of the task k and $\mathbf{y}_k$ is the target. If the task is classification, $\mathbf{y}_k \in \mathbb{R}^{n_k}$ represents the class labels. If the problem is regression, then $\mathbf{y}_k \in \mathbb{R}^{n_k \times p}$ comprises of real valued target vectors. A linear model parametrized by vector $\mathbf{w}_k \in \mathbb{R}^d$ is learned optimizing the loss function of each task k is $\ell_k(\mathbf{x}_k, \mathbf{y}_k; \mathbf{w}_k)$, which can be either least squares or logistic loss depending on the problem. In addition, tasks can be heterogeneous [123] such that while some tasks are regression, others can be classification. Each task minimizes their corresponding loss function separately. However, these tasks are assumed to share a common knowledge, and thus they are not independent. Regularized MTL addresses the task relatedness by imposing a constraint on the single task models to perform knowledge transfer. As a result, learning one task consequently benefits learning other tasks [21]. One of the most popular constraints to impose task relatedness is the assumption of low rank model matrix.

Let $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_K] \in \mathbb{R}^{d \times K}$ be the model matrix of all K tasks. One intuitive idea to learn $\mathbf{W}$ is optimizing the joint objective function given as $f(\mathbf{W}) = \sum\limits_{k=1}^{K} \ell_t(\mathbf{w}_k)$. However, this approach learns the single tasks simultaneously rather than facilitating a knowledge transfer between them since $f(\mathbf{W})$ decouples each task $\mathbf{w}_k$. To achieve the knowledge transfer, MTL often utilizes a regularization term as given below [41, 128, 129]:

$$\min_{\mathbf{W}} \left\{ \sum\nolimits_{k=1}^{K} \ell_k(\mathbf{w}_k) + \lambda g(\mathbf{W}) \right\} \equiv f(\mathbf{W}) + \lambda g(\mathbf{W}), \tag{3.3.1}$$

where $g(\mathbf{W})$ is a penalty term that couples K tasks and λ is the regularization parameter to control the amount of knowledge to be shared among tasks. For instance, when $g(\mathbf{W})$ is chosen as nuclear norm $\|\mathbf{W}\|_*$, high λ values result in a lower rank solution of $\mathbf{W}$. In other words, high λ values indicate that the individual tasks share a large amount of information.

In this study, we focus on one of the commonly used regularized MTL, named shared subspace learning [7]. Shared subspace MTL utilizes the nuclear norm regularizer $g(\mathbf{W}) = \|\mathbf{W}\|_* = \sum\nolimits_{i=1}^{\min(d,K)} \sigma_i(\mathbf{W})$, where $\sigma_i(\mathbf{W})$ is the i-th singular value of the matrix $\mathbf{W}$. In other words, the nuclear norm is the tightest convex relaxation of the rank function [42]. As it was discussed earlier, a low-rank $\mathbf{W}$ means the columns of the $\mathbf{W}$ matrix, representing individual models, are linearly dependent, and thus they share a low-dimensional subspace [123]. The nuclear norm regularizer is a non-smooth function. For this reason, proximal gradient descent method discussed in Chapter 2 can be used to solve the regularized MTL problem [64]. In the following section, proximal gradient descent solution in synchronized distributed MTL setting is discussed.

3.3.2 Synchronized Multi-Task Learning (SMTL)

The MTL formulation adopted in this study has a smooth and a non-smooth term (nuclear norm regularization). The solution of such problems is provided by the proximal gradient based optimization techniques. There are several first and second order proximal algorithms in literature such as FISTA [13], SpaRSA [119], and PNOPT [76]. The common procedure shared by most of the proximal solvers comprises of a gradient update and a proximal projection. In the gradient update step, gradient of the smooth term is computed. Since the smooth term is the summation of loss functions of each task, gradient of each task model is computed and aggregated to obtain the gradient of the smooth term as given in Eq. 3.3.2.

$$\nabla f\left(\mathbf{W}\right) = \sum\nolimits_{k=1}^{K} \nabla \ell_k \left(\mathbf{w}_k\right) \tag{3.3.2}$$

After gradient is obtained, the model matrix is updated to obtain an intermediate solution:

$$\hat{\mathbf{W}} = \mathbf{W} - \eta \nabla f\left(\mathbf{W}\right) \tag{3.3.3}$$

The intermediate solution is not the optimal solution since this point is not in our solution domain. Due to the nuclear norm, our solution space is in the low rank matrices. A proximal projection needs to be performed to map the intermediate solution $\hat{\mathbf{W}}$ into the solution space. The proximal projection is achieved by solving the following optimization problem.

$$\text{Prox}_{\eta\lambda g}\left(\hat{\mathbf{W}}\right) = \arg\min_{\mathbf{W}} \frac{1}{2\eta} \|\mathbf{W} - \hat{\mathbf{W}}\|_F^2 + \lambda g\left(\mathbf{W}\right) \tag{3.3.4}$$

where η is step size, $\|\cdot\|_F$ is the Frobenius norm, and $\hat{\mathbf{W}}$ is the intermediate point obtained in Eq. 3.3.3. In the standard MTL problem setting, natural assumption is to centralize the training data of each task. As discussed before, in practice single task datasets are not necessarily located in the same server and data transfer is not always feasible. For this reason, a distributed MTL approach is necessary when the datasets for each task $\mathcal{D}_1, \ldots, D_k, \ldots, \mathcal{D}_K$ are stored in separate local machines.

In this study, each local system with its corresponding single task dataset $\mathcal{D}_k$ is named *task node*. Task nodes are responsible for the decoupled operations, such as the gradient update of the individual task models. On the other hand, the proximal projection is a coupled operation, which means that individual updated models need to be transferred to a *central server* to execute the proximal mapping on $\mathbf{W}$. After the proximal mapping is performed in the central server, updated models need to be transferred back to the task nodes. The aforementioned distributed setting is known as *synchronous* since the central server waits for all the task nodes to finish their executions before the proximal mapping can be performed. In other words, proximal mapping is executed on the intermediate results obtained at the same iteration. In this setting, central server is idle until all the intermediate results are received. Similarly, task nodes do not proceed until they receive the updated solution from the central server. For this reason, the synchronized approach is generally slow when (i) computational power of some of the task nodes is limited, (ii) there are communication delays, and (iii) data size at the task nodes is unbalanced. In the worst case, one of the nodes might stop working and consequently the optimization procedure may have to be terminated. For these reasons, solving a first-order optimization problem, that requires many iterations to converge to a reasonable precision, is not practical within a synchronous framework.

3.3.3 Asynchronous Multi-Task Learning (AMTL)

To address the disadvantages of the synchronized MTL discussed in the previous section, we propose an asynchronous MTL framework. The proposed model, named AMTL, comprises of task nodes and a central server similar to standard distributed MTL setting. However, in AMTL, central server does not wait for all the task nodes to finalize their gradient update to perform the proximal step. Similarly, task nodes do not wait for all the other task nodes to receive their updated model to start gradient update. In this framework, task nodes need to maintain a copy of $\mathbf{W}$, which may not necessarily be the most current. This inconsistency might hurt the convergence, however the proposed AMTL framework is implemented within ARock [38, 94] which provides a linear convergence rate. In particular, the task models are learned via asynchronous parallel coordinate updates using Krasnosel'skii-Mann (KM) iterations. In this framework, forward-backward splitting algorithm is used to perform proximal gradient descent. Each iteration is divided into forward and backward steps, where the forward step computes the intermediate solution using the gradient update and the backward step performs the proximal mapping.

Algorithm 3 The proposed asynchronous multi-task learning framework

```
Require: Multiple related learning tasks reside at task nodes, including the training data and the loss function for each task $\{x_1, y_1, \ell_k\}, ..., \{x_K, y_K, \ell_K\}$, maximum delay $\tau$, step size $\eta$, multi-task regularization parameter $\lambda$.
Ensure: Predictive models of each task $v_1, ..., v_K$.
  Initialize task nodes and the central server.
  Choose $\eta_i \in [\eta_{\min}, \frac{c}{2\tau/\sqrt{T}+1}]$ for any constant $\eta_{\min} > 0$ and $0 < c < 1$
  forall every task node asynchronously and continuously do
    Task node $k$ requests the server for the forward step computation $\text{Prox}_{\eta\lambda g}\left(\hat{\mathbf{v}}^i\right)$, and
    Retrieves $\left(\text{Prox}_{\eta\lambda g}\left(\hat{\mathbf{v}}^i\right)\right)_k$ from the central server and
    Computes the coordinate update on $\mathbf{v}_k$

        $\mathbf{v}_k^{i+1} = \mathbf{v}_k^i + \eta_i \left( \left(\text{Prox}_{\eta\lambda g}\left(\hat{\mathbf{v}}^i\right)\right)_k - \eta \nabla \ell_k \left( \left(\text{Prox}_{\eta\lambda g}(\hat{\mathbf{v}}^i)\right)_k \right) - \mathbf{v}_k^i \right)$    (3.3.5)

    Sends updated $v_k$ to the central node.
  end forall
```

In our model, we propose to use backward-forward splitting [32, 93] to solve Eq. 3.3.1 where we reverse the order of gradient and proximal steps. The order of the steps does not change the convergence but affects the number of network communications between the task nodes and the central server within one iteration. In backward-forward splitting, one iteration starts with proximal mapping at the central server and ends with the gradient update at the task node. Thus, one way network communication is needed to finalize one iteration. In the following, we present more details about the solution of the optimization problem in Eq. (3.3.1).

According to the optimality condition, the optimal solution $\mathbf{W}^*$ of a composite function such as $\{f(\mathbf{W}) + \lambda g(\mathbf{W})\}$ should satisfy the following:

$$0 \in \nabla\{f(\mathbf{W}^*) + \lambda \partial g(\mathbf{W}^*)\} \tag{3.3.6}$$

where $\partial g(\mathbf{W}^*)$ denotes the sub-gradient set of the non-smooth part. This condition states that the set comprising the gradient of the smooth term and the sub-gradients of the non-smooth term at $\mathbf{W}^*$ should include 0 if $\mathbf{W}^*$ is the optimal solution. If we derive the optimal solution $\mathbf{W}^*$ from

Eq. 3.3.6 leading the forward-backward iteration:

$$\begin{aligned} -\nabla f(\mathbf{W}^*) &\in \lambda\partial g(\mathbf{W}^*) \qquad (3.3.7)\\ -\eta\nabla f(\mathbf{W}^*) &\in \eta\lambda\partial g(\mathbf{W}^*)\\ \mathbf{W}^* - \eta\nabla f(\mathbf{W}^*) &\in \mathbf{W}^* + \eta\lambda\partial g(\mathbf{W}^*). \end{aligned}$$

Thus the forward-backward iteration is expressed as:

$$\mathbf{W}^+ = (\mathbf{I} + \eta\lambda\partial g)^{-1}(\mathbf{I} - \eta\nabla f)(\mathbf{W}) \qquad (3.3.8)$$

where $(\mathbf{I}-\eta\nabla f)$ represents the forward operator on $\mathbf{W}$ and $(\mathbf{I}+\eta\lambda\partial g)^{-1}$ is the backward operator. Eq. 3.3.8 converges to the solution with value of $\eta \in (0, 2/L)$ under the assumption that the loss function $f(\mathbf{W})$ is convex and L-Lipschitz differentiable with $L > 0$, and $g(\mathbf{W})$ is closed proper convex. As it was discussed before, while the forward operator is separable since $\nabla f(\mathbf{W})$ is decoupled, i.e., $\nabla f(\mathbf{W}) = [\nabla\ell_1(\mathbf{w}_1), \nabla\ell_2(\mathbf{w}_2), \cdots, \nabla\ell_K(\mathbf{w}_K)]$. The backward operator is non-separable due to coupling of individual models in proximal mapping. Since there is a distributed setting, reversing the order of forward and backward operators is more efficient in terms of network communications. As a result, backward-forward iteration can be expressed as:

$$\mathbf{V}^+ = (\mathbf{I} - \eta\nabla f)(\mathbf{I} + \eta\lambda\partial g)^{-1}(\mathbf{V}) \qquad (3.3.9)$$

where we need to use an auxiliary matrix $\mathbf{V} \in \mathbb{R}^{d\times K}$ since the update variables in two cases are not the same. However, the final optimal solution $\mathbf{W}^*$ can be obtained from $\mathbf{V}^*$ by one additional backward step at the end of the iterations. In the current setting, we follow the coordinate update framework [94], but with a block coordinate update modification, where each *task model* is a block of variables for the corresponding task. Update procedure for each task is defined as follows:

$$\mathbf{v}_k^+ = (\mathbf{I} - \eta\nabla\ell_k)\left((\mathbf{I} + \eta\lambda\partial g)^{-1}(\mathbf{V})\right)_k \qquad (3.3.10)$$

where $\mathbf{v}_k \in \mathbb{R}^d$ is the auxiliary variable for model $\mathbf{w}_k$ of task k. As it can be seen in Eq. 3.3.10, updating one task block requires one backward step applied on the whole auxiliary model matrix $\mathbf{V}$ and one forward step performed on the corresponding task block $\mathbf{v}_k$. The overall framework is given in Algorithm 3, where Eq. 3.3.5 represents the KM iteration. KM iteration provides a general framework for problems such as finding the fixed point of a non-expansive operator. In fact, the backward-forward operator is a non-expansive operator and finding the optimal solution of the problem defined in Eq. 3.3.1 is an instance of the fixed point problem [94]. We refer to Section 2.4 of [94] for the derivation of Eq. 3.3.5. One of the main purposes of the proximal gradient methods is to address situation where sub-problems with different difficulties. In our distributed regularized MTL setting, backward operation or the proximal mapping has a closed form solution, therefore it is easier to compute compared to forward step, especially when we have large scale data. Since the gradient computation usually takes more time due to large number of data points, it is preferred to perform the block coordinate update with backward-forward iteration.

In AMTL setting, network communication is limited between the central server and the task nodes. Task nodes do not communicate with each other. Moreover, the communication between the task node k and the central server contains only the model vector $\mathbf{v}_k$ which is much smaller compared to the dataset $\mathcal{D}_k$ stored in the task node k. For example, if a 1024 dimensional vector is saved in standard binary format, only 8KB will be needed to transfer the model vector, compared with sending the entire data matrix (e.g., if we consider $500,000$ data points, 4GB will be required for $500,000 \times 1024$ data matrix). Hence, distributed AMTL reduces the network communication cost. Since we only need to send the model vector and not the data matrix itself, privacy concerns are also addressed. However, the current AMTL model is not a privacy preserving approach. An extension of the proposed model to a privacy-preserving proximal gradient algorithm with asynchronously updates in the distributed MTL setting is introduced in another study [122]. Therefore, the current setting introduced in the chapter can be considered as a first step to a privacy preserving distributed MTL framework.

3.3.3.1 Asynchronous Updates of AMTL

Asynchronous update mechanism of AMTL is illustrated in Figure 3.2. Multi-task model matrix is stored and updated in the central server. For instance, in Figure 3.2, task node 2 receives its updated model from the central server at time t_1. As soon as a new model arrives at a task node, gradient update or the forward step starts. When the task node finalizes the forward step, updated model is sent back to the central server for the backward step. This leads to an inconsistency at the task node side. If we look at Figure 3.2, we can realize that while task node 2 is still performing the forward step, task node 4 has already finished its forward step and sends its updated model to the central server. Since this is an asynchronous scheme, central node does not need to wait for task node 2 and performs the proximal mapping on the model matrix, which contains the updated model coming from task node 4 and the existing models from other nodes. Therefore, when task node 2 is ready to send its model to central server, model matrix has already been updated. In other words, an inconsistency between the previously received model from the central server and the model stored in the central server after the forward step occurs at the task node side. This means that the model received by the task node 2 at time t_1 is not the same copy as the model at time t_3 in the central server. Under the aforementioned inconsistency, linear convergence of AMTL is shown via the following theorem which follows the convergence analysis presented in [94].

Theorem 3.3.1. *Let $(\mathbf{V}^i)_{i\geq 0}$ be the sequence generated by the proposed AMTL with $\eta_i \in [\eta_{\min}, \frac{c}{2\tau/\sqrt{K}+1}]$ for any $\eta_{\min} > 0$ and $0 < c < 1$, where τ is the maximum delay. Then $(\mathbf{V}^i)_{i\geq 0}$ converges to an V^*-valued random variable almost surely. If the MTL problem in Eq. 3.3.1 has a unique solution, then the sequence converges to the unique solution.*

This theorem states that if the MTL problem has a unique solution, the proposed asynchronous optimization framework will converge to the unique solution with a constant step size η_i.

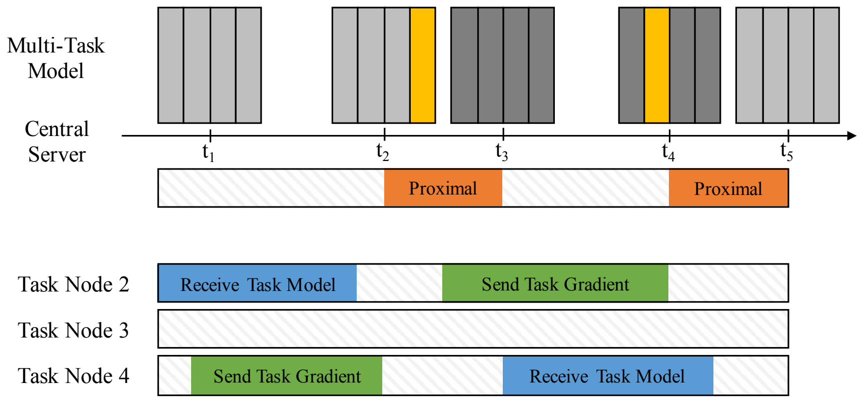


Figure 3.2 Illustration of asynchronous updates in AMTL. The asynchronous update scheme has an inconsistency when it comes to reading model vectors from the central server.

3.3.3.2 Dynamic Step Size in AMTL

In this study, all the task nodes are assumed to follow independent Poisson processes and have the same activation rate [94]. A task node is *activated* when it performs gradient update and communicates with the central server for updates. Activation rates of different nodes are assumed to be the same. For instance, the probability of each task node being activated before other task nodes is $1/K$ [73], where K is the total number of tasks. On the other hand, different task nodes often have different activation rates in real world network settings due to the topology of the network. We proposed a dynamic step size approach to address this fact. In asynchronous updates, step size is usually much smaller to guarantee the convergence compared to synchronous settings. The dynamic step size idea was previously used in a specific setting with asynchronous optimization to boost the overall performance [26]. Dynamic step size proposed in this study integrates a time multiplier into the update of AMTL defined in Eq. 3.3.5:

$$\mathbf{v}_k^{i+1} = \mathbf{v}_k^i + c_{(k,i)}\eta_i \left(\left(\text{Prox}_{\tau\lambda g}\left(\hat{\mathbf{v}}^i\right)\right)_k - \eta \nabla \ell_k \left(\left(\text{Prox}_{\tau\lambda g}(\hat{\mathbf{v}}^i)\right)_k \right) - \mathbf{v}_k^i \right) \tag{3.3.11}$$

where the multiplier is given by:

$$c_{(k,i)} = \log\left(\max\left(\bar{\nu}_{k,i}, 10\right)\right) \tag{3.3.12}$$

where $\bar{\nu}_{k,i} = \frac{1}{i+1}\sum_{j=z-i}^{z} \nu_k^{(j)}$ is the average of the last $i+1$ delays in task node k, z is the current time point, and $\nu_k^{(j)}$ is the delay at time j for task k. Thus, the actual step size is scaled by the history of communication delays between the task nodes and the central server such as $c_{(k,i)}\eta_i$. If the delay is long, the probability of a task node being activated is low which means the activation rate is small. In such cases, the step size is increased to compensate for the delay. As different types of functions can also be used instead of Eq. 3.3.12, utilizing a history of delays is empirically shown to help boosting the convergence.

3.4 Experiments

In this section, we compare the efficiency of AMTL and synchronized MTL, namely SMTL. AMTL framework was implemented in C++. Distributed environment was simulated by a shared memory architecture [94] where the network delays were artificially added to the task node to mimic the real world network scenario. Empirical convergence behaviors of AMTL and traditional SMTL were compared on synthetic datasets. Effect of proposed dynamic step size was also investigated on synthetic datasets with various numbers of tasks. Hardware used in the experiments was an Intel Core i5-5200U CPU 2.20GHz x 4 dual-core laptop whose the performance was limited by the number of cores. Subsequently, we also developed a standalone Java socket programming implementation of AMTL [1] available in the public domain.

3.4.1 Experimental setting

In our experiments, threads represent the task nodes and the number of threads was equal to the number of tasks; the shared memory played the central server role. The proposed framework can work with generic regularized MTL formulations, however our experiments focused on the low-rank MTL formulation for shared subspace learning. Each task was assumed to solve a regression problem with least squares loss $\sum_{k=1}^{K} \|\mathbf{X}_k \mathbf{w}_k - \mathbf{y}_k\|_2^2$, where $\mathbf{X}_k,\ n_k$, and $\mathbf{y}_k$ denote data matrix,

[1] Available at https://github.com/illidanlab/AMTL

sample size, and the targets of the task k, respectively. In the shared subspace learning formulation given in Eq. 3.4.1, nuclear norm is used as the regularization, which provides low-rank solutions. Nuclear norm couples the model vectors and learns a low-dimensional subspace that actually achieves the knowledge transfer between tasks.

$$\min_{\mathbf{W}} \left\{ \sum\nolimits_{k=1}^{K} \|\mathbf{x}_k \mathbf{w}_k - \mathbf{y}_k\|_2^2 + \lambda \|\mathbf{W}\|_* \right\} \tag{3.4.1}$$

Nuclear norm, a.k.a trace norm is defined as follows:

$$\|\mathbf{W}\|_* = \text{trace}\left(\mathbf{W}^T \mathbf{W}\right) = \sum_{j=1}^{\min\{d,K\}} \sigma_j \left(\mathbf{W}\right) \tag{3.4.2}$$

where $\sigma_j (\mathbf{W})$ denotes jth singular value of matrix $\mathbf{W}$. As we can see from its formulation, nuclear norm is not separable and smooth. Therefore, we need to apply the proximal mapping of the nuclear norm [20, 64] in the backward step as given below:

$$\begin{aligned} \text{Prox}_{\eta\lambda g}\left(\hat{\mathbf{V}}^i\right) &= \sum_{j=1}^{\min\{d,K\}} \max\left(0, \sigma_j - \eta\lambda\right) \mathbf{u}_i \mathbf{v}_i^\top \\ &= \mathbf{U}\left(\Sigma - \eta\lambda\mathbf{I}\right)_+ \mathbf{V}^\top \end{aligned} \tag{3.4.3}$$

where $\{\mathbf{u}_j\}$ and $\{\mathbf{v}_j\}$ are the columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, $\hat{\mathbf{V}}^i = \mathbf{U}\Sigma\mathbf{V}^\top$ is the singular value decomposition (SVD) of $\hat{\mathbf{V}}^i$ and $(x)_+ = \max(0, x)$.

When the central node performs the proximal mapping during backward step, the current version of the models in the shared memory is used. As it was discussed earlier, during the execution of the backward step, some of the models in the shared memory might be changed because of the asynchronous nature of the framework. In nuclear norm regularization, every backward step requires singular value decomposition (SVD) of the model matrix $\mathbf{W}$. Complexity of SVD is $O\left(d^3\right)$, and therefore the backward step is computationally expensive when the data dimension, d, is high. To avoid computing the full SVD in each iteration, online SVD [19] may be preferred. The complexity of the online SVD for a $p \times q$ rank$-r$ matrix is $O\left(pqr\right)$ [19]. SVD is performed once at the beginning and then the left $\mathbf{U}$, right $\mathbf{V}$, and singular value matrices $\mathbf{\Sigma}$ are used to update the SVD of the matrices in subsequent iterations. Whenever a task node changes the corresponding column of the model matrix, central server performs proximal mapping. For this reason, online SVD could be a more efficient approach when we deal with huge number of tasks and high dimensionality.

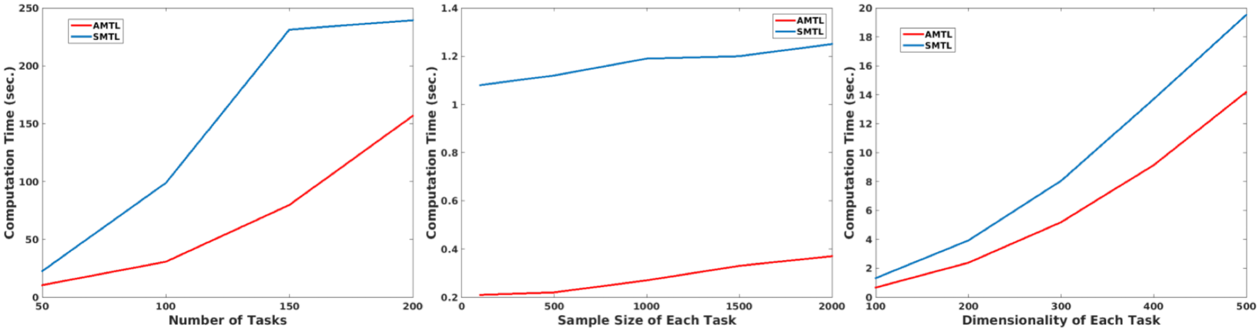
3.4.2 Comparison between AMTL and SMTL

Synthetic and real-world data experiments were conducted to compare the computation times of AMTL and SMTL. To simulate a realistic network setting, random network delays are introduced to task nodes for both AMTL and SMTL.

3.4.2.1 Synthetic Dataset Experiments

In this set of experiments, computation times of AMTL and SMTL are compared based on public and randomly generated synthetic datasets with varying number of tasks, dimensionality, and sample sizes. In Figure 3.3, the computation time for varying numbers of tasks (a), sample sizes (b), and dimensionality (c) are shown, where the blue curves represent SMTL and the red curves represent AMTL. All three plots were obtained for the same and fixed number of iterations. While comparing the computation time for different number of tasks, dimensionality of the datasets and the samples sizes were fixed to 50 and 100, respectively. If we look at Figure 3.3 (a), we can observe that computation time of SMTL increases faster than AMTL with increasing number of tasks. Note that after 150 tasks, time consumption of SMTL does not increase significantly, which can be explained because of the limited number of cores (dual-core machine) used in experiments.

In Figure 3.3 (b), data dimensionality was fixed to 50, and the number of tasks was 5. Computation time for both SMTL and AMTL approaches did not increase drastically with the increase in sample size. This can be explained due to the fact that the proximal mapping is performed on model matrix which is independent of the sample size. Only the forward step, which computes the gradient, is affected by the sample size. For this reason, the computational time of the backward



(a) (b) (c)

Figure 3.3 Computation times of AMTL and SMTL for (a) varying number of tasks when the number of dimensions and the number of data points per node were fixed at 50 and 100, respectively, for (b) varying number of task sizes with 50 dimensions, and 5 tasks, and for (c) varying dimensionality of 5 tasks with 100 samples in each task. As expected, computational time of SMTL is higher than AMTL for a fixed number of iterations.

step remains unchanged with the increasing number of sample sizes for both SMTL and AMTL. In Figure 3.3 (c), number of tasks were fixed at 5 and the sample size per task was fixed at 100. Computation times of both approaches increases with dimensionality, which is expected due to the proximal mapping of the nuclear norm. Furthermore, the gap between AMTL and SMTL computational requirements also gets wider with dimensionality because SMTL needs to wait longer for the updates at both task nodes and central server. In summary, computation time increases with the number of tasks, sample sizes and the dimensionality for both SMTL and AMTL. However, the rate of the increase is higher for SMTL compared to AMTL as expected.

In the next experiment, different network characteristics were investigated for synthetic datasets with varying numbers of tasks. Fifty dimensional datasets with 100 samples per task were used to compare the computational time of AMTL and SMTL under different amounts of network delays. Summary of the results for 5, 10, and 15 tasks are given in Table 3.1. Similar to the previous problem setting, regression problem with the squared loss and nuclear norm regularization was taken into account. “Network” column of the table represents synthetically generated network delays, where the numbers next to AMTL and SMTL denote the offset values for the amount of delay. For instance, AMTL-5 means that each task node is idle for 5 seconds plus a random number

Table 3.1 Computation times (sec.) of AMTL and SMTL with different network delays. The offset value of the delay for AMTL-5, 10, 30 was chosen as 5, 10, 30 seconds. Same network settings were used to compare the performance of AMTL and SMTL. AMTL performed better than SMTL for all the network settings and numbers of tasks considered here.

Network	5 Tasks	10 Tasks	15 Tasks
AMTL-5	156.21	172.59	173.38
SMTL-5	239.34	248.23	256.94
AMTL-10	297.34	308.55	313.54
SMTL-10	452.84	470.79	494.13
AMTL-30	902.22	910.39	880.63
SMTL-30	1238.16	1367.38	1454.57

of seconds after it completes the forward step. As we can see from Table 3.1, the increase in the computation time with network delays is significantly more for SMTL than AMTL.

3.4.2.2 Real World Datasets Experiments

We also conducted experiments for three public datasets for multi-task learning. Details about the datasets are summarized in Table 3.2. School dataset is a popular multi-task learning dataset containing exam records of 139 schools in 1985, 1986, and 1987 provided by the London Education Authority (ILEA) [89]. MNIST is a well known handwritten digits dataset with $60,000$ training samples and $10,000$ test samples [75]. MNIST was used for 5 binary classification tasks: 0 v. 9, 1 v. 8, 2 v. 7, 3 v. 6, and 4 v. 5 . The third public data is Multi-Task Facial Landmark (MTFL) dataset [126] containing $12,995$ face images with different genders, and head poses. MTFL contains facial features, including five facial landmarks and attributes for gender, smiling/not smiling, with/without glasses, and head pose (e.g., frontal, right profile, and left profile). Four binary classification tasks such as male v. female, smiling v. not smiling, with or without glasses, and right or left head pose were designed for multi-task learning setting. For binary classification tasks, logistic loss was used. Training time comparison is given in Table 3.3 with different amounts of network delays. For the datasets with larger number of tasks, the gap between training times of AMTL and SMTL was bigger.

Table 3.2 Benchmark public datasets used in this study. Sample sizes vary per task. Third column of the table summarizes the minimum and maximum number of data points tasks might have in each dataset.

Dataset	Number of tasks	Sample sizes	Dimensionality
School	139	22-251	28
MNIST	5	13,137-14,702	100
MTFL	4	2,224-10,000	10

Table 3.3 Training time (sec.) comparison of AMTL and SMTL for the three public datasets. Similar to the synthetic data experiments, AMTL generally requires smaller training time compared to SMTL under different network settings.

Network	School	MNIST	MTFL
AMTL-1	194.22	54.96	50.40
AMTL-2	231.58	83.17	77.44
AMTL-3	460.15	115.46	103.45
SMTL-1	299.79	57.94	50.59
SMTL-2	298.42	114.85	92.84
SMTL-3	593.36	161.67	146.87

Experimental results suggest that an asynchronous framework is preferred in a distributed setting with communication delays and different characteristics of task nodes. Moreover, AMTL provides a more robust optimization scheme compared to SMTL in terms of network failures. Asynchronous optimization can still continue even one task node turns offline suddenly. One potential drawback of AMTL is convergence due to the inconsistent updates. To investigate the convergence property, we conducted another experiment. In Figure 3.4, change in objective values of AMTL and SMTL is given for a fixed number of iterations and synthetic data with 5 and 10 tasks. According to our experiments, AMTL did not suffer from slower convergence compared to SMTL.

3.4.3 Dynamic step size

In this section, effect of the proposed dynamic step size approach is investigated. The average delay of the last 5 iterations was used to modify the step size for randomly generated 50 dimensional synthetic datasets with 100 samples in each task and fixed number of iterations. At the end of the

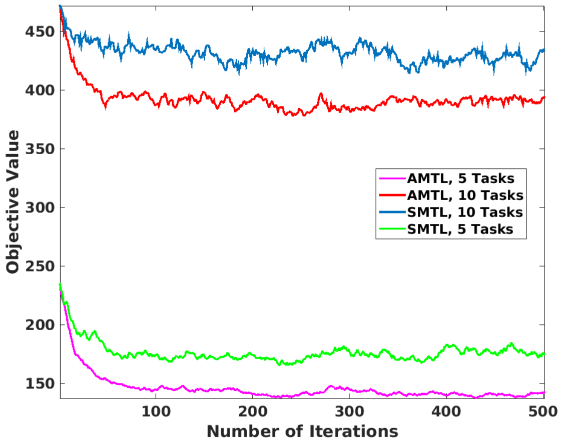


Figure 3.4 Convergence of AMTL and STML under the same network configurations. Experiment was conducted for randomly generated synthetic datasets with 5 and 10 tasks.

iterations, objective values of each dataset with various number of tasks and different delay patterns were observed. Some task nodes have to wait longer than other nodes due to the network delays and this situation slows the convergence. To compensate the effect of delays on the convergence, we increased the step size of the nodes which had to wait for a long time in previous iterations by following the formulation in Eq. 3.3.12. Experimental results in Table 3.4 showed that dynamic step size could indeed boost the convergence. We can observe a decrease in the objective value of AMTL with dynamic step size compared to the AMTL with constant step size for the equal number of iterations, indicating that the dynamic step size contributes to a faster convergence. In addition, the objective values were observed to decrease with an increasing amount of delay, when the dynamic step size was used. There were no theoretical analysis of dynamic step size while designing the proposed approach, however the empirical results indicate that the dynamic step size could be promising to boost convergence in distributed settings with network delays.

3.5 Summary

In this chapter, an asynchronous framework, AMTL, is proposed for distributed multi-task learning. Distributed datasets are commonly encountered in many domains, such as EHRs in hospitals

Table 3.4 Objective values of the synthetic dataset with $5, 10$, and 15 tasks. Objective values of different network settings are shown. Dynamic step size yields lower objective values at the end of the last iteration than fixed step size. This result indicates that the dynamic step size in asynchronous distributed setting is recommended to boost the convergence.

Network	Without dynamic step size			With dynamic step size		
Number of tasks	5	10	15	5	10	15
AMTL-5	163.62	366.27	559.07	144.83	334.24	508.65
AMTL-10	163.59	367.63	561.68	144.77	333.71	505.64
AMTL-15	163.56	366.26	561.87	143.82	333.13	500.05
AMTL-20	168.63	366.35	561.21	143.50	331.13	499.97

located in different geographical regions. AMTL performs asynchronous distributed block coordinate descent with backward-forward splitting scheme on regularized MTL formulations. Experimental results presented the efficiency of AMTL compared to synchronous version, SMTL, in terms of computational time. A dynamic step size strategy is introduced to boost the convergence performance of distributed settings with network delays.

Chapter 4

Patient Subtyping via Time-Aware LSTM Networks

In the previous chapters, we addressed the challenges due to the scale, interpretability, and the distribution of the EHR datasets. In this chapter, temporal aspect of the EHRs is taken into account. EHR data essentially comprises the medical history of patients. This means that we have the opportunity to investigate the changes and the dependences between consecutive patient records. In recent years, state-of-the-art performances for sequential data analysis have been achieved by recurrent neural network (RNN), and its variants (e.g., LSTM, Gated Recurrent Unit (GRU)). In this chapter, we introduce a new LSTM architecture, named time-aware LSTM (T-LSTM), to address analysis of longitudinal EHRs with unevenly sampled time steps. Medical history of patients is one of the most important components of clinical decision making. Temporal structure of patient records is also crucial in disease progression studies. One prominent task in disease progression studies is patient subtyping. Subtyping aims to group patients based on similar disease progression patterns. In particular, patient subtyping corresponds to clustering temporal patient records. An important component of this task is to learn a single representation for the sequential patient data such that the temporal structure can be captured and embedded into the representation. Standard RNN architectures, such as LSTM, assume that the time gap between consecutive elements is uniform throughout the sequence. However, EHR data naturally has irregular elapse time where the time between consecutive records can vary from weeks to years. In this study [11], T-LSTM is designed based on the hypothesis that if there is a long time gap between two consecutive records, the effect of the previous time step on the current output should be reduced. Thus, T-LSTM learns the temporal patterns while considering the irregular elapsed times which demonstrates a more realistic problem setting. For patient subtyping purposes, a single distinctive representations for patient sequences are learned using T-LSTM auto-encoder and patient representations are clustered into subtypes.

4.1 Introduction

Medical history of patients is an important component of clinical decision making. EHRs store digital medical history of patient information including test results, procedures, medications, and diagnoses. Clinical research also benefits from the medical history to discover unknown patterns in patient cohorts to investigate prognosis of different types of diseases, and the effects of drugs. In short, large-scale, systematic and longitudinal patient records play a key role in the healthcare domain. EHRs provide a rich source of such longitudinal data, however as discussed in Chapter 1, analyzing EHRs is challenging. Biomedical informatics resorts to machine learning approaches to alleviate these challenges [23, 57, 84, 120, 131] and tries to address difficult tasks, such as disease progression modeling and risk prediction [28, 30, 39, 80, 130, 132]. *Patient subtyping* is one of the disease progression modeling tasks that investigates patient groups with similar disease progression pathways. Subtyping task is a crucial step of personalized medicine which prescribes treatments that best fits the health conditions of a group of patients.

Patient subtyping usually takes a particular type of disease into consideration [23], such as Parkinson's disease. Since essentially it is a grouping task, patient subtyping is considered as an unsupervised learning problem. In particular, an approach that can cluster time series data is required. One important note is that time series analysis approaches based on aligning the time

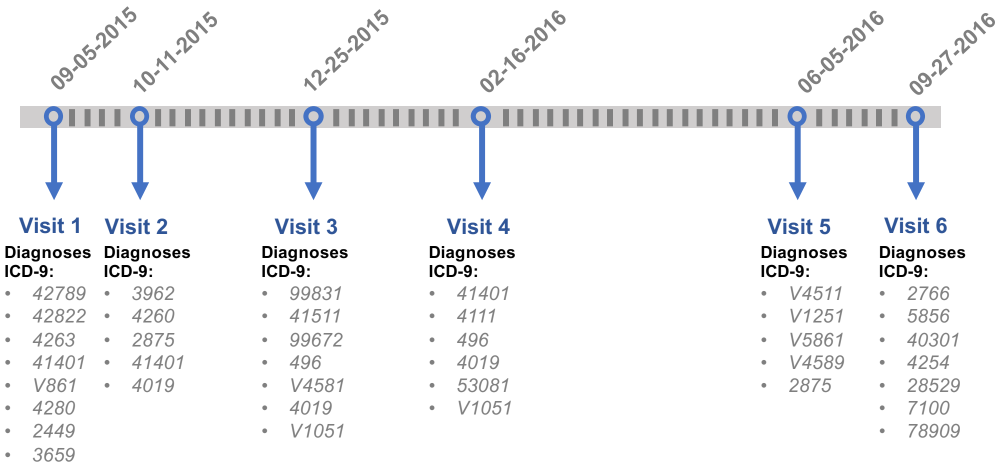


Figure 4.1 An example segment of longitudinal patient records where the patient had 6 office visits between Sept 5, 2015 and Sept 27, 2016. In each visit, medical information such as diagnosis of the patient is recorded. Diagnoses are usually encoded as ICD-9 codes. Time gap between two successive visits varies.

sequences and computing the similarity between them are not suitable for patient subtyping task. Here, every element of the time sequence contains heterogeneous information mostly in the form of high dimensional vectors. For this reason, we need to use a more extensive approach that can address the complexity of the temporal EHR data. One efficient method that can capture temporal patterns is recurrent neural network (RNN) that has been successfully applied to different problems such as speech recognition [51], text classification [72], video processing [37, 106], and natural language processing [117]. Even though, in theory, long term dependencies can be successfully captured by the RNNs. However, RNN performance is not robust against vanishing and exploding gradients due to back-propagation through time. To avoid this problem, several variants of RNN have been proposed such as long-short term memory (LSTM) [58] networks. LSTM is a gated RNN structure where the hidden state of the next time step depends on a summation rather than a matrix multiplication. The additive nature of LSTM overcomes the vanishing and exploding gradient problem. LSTM has also been applied to biomedical informatics [22, 24] with promising results.

Even though it is not explicitly stated, LSTM assumes that the time gap between the elements of the time sequence is uniform. Sampling frequency of an audio file, time gap between measurements periodically taken from a sensor, or the gap between words of a sentence can naturally be the same throughout the sequence. However, uniform elapsed time assumption does not always hold. In case of EHRs, time span between consecutive records can vary from days to months, and sometimes years. An example illustration of a patient record is given in Figure 4.1. In this figure, segment of a sample medical record of a patient is shown. Each time step represents one visit to the hospital: medical information of the patient is usually recorded at the end of each visit such as diagnoses encoded in ICD-9. As we can see from the figure, there can be months between consecutive visits. As a matter of fact, the elapsed time might even contain important information about patient's condition. For example, if a patient visits hospital frequently, this might indicate an ongoing disease and information recorded in consecutive visits would assist the clinical decision maker of the next visit. On the other hand, if there are months and years between two consecutive visits, then the dependency on the previous information is little.

In this study, we propose Time-Aware LSTM (T-LSTM) to address aforementioned challenges for sequences with irregular time gaps and show an application on patient subtyping. T-LSTM takes elapsed time into account to adjust the memory content of the LSTM unit by decomposing the memory as long and short term memories and applying a time decay on the short term component. The amount of decay is determined by the elapsed time such that longer the elapsed time, smaller the effect of the previous memory to the current output. The proposed architecture is then used for patient subtyping purpose which is essentially a clustering problem. To be able to cluster patients, a single representation from patient's records is learned by using a T-LSTM auto-encoder. The proposed T-LSTM auto-encoder maps temporal sequences of patients to a representation by capturing the dependencies between consecutive records in the presence of time irregularities. Experiments were conducted for supervised and unsupervised tasks on synthetic and real world datasets to examine the performance of T-LSTM. Before introducing the details of the proposed approach, a literature review is presented in the next section.

4.2 Literature Review

4.2.1 RNNs for Biomedical Informatics

There have been several studies applying deep learning methods to biomedical informatics data. Patient subtyping and phenotyping are two prominent research topics in healthcare where learning a powerful representation of a patient is crucial. Patient data is very complicated and heterogeneous, therefore it is challenging to cluster patients and to do predictive analysis based on their medical records. Supervised and unsupervised machine learning approaches usually rely on a single discriminative representation of data points to achieve a good performance. In that sense, deep learning offers a representation learning strategy that usually leverages large amounts of data. Another advantage of deep models is that they start from coarse representations of the raw input and obtain finer representations that can summarize the data very well by capturing the spatial and temporal patterns. RNN and its variants intuitively learn a representation of the sequence at each time step by considering temporal dependencies.

As it was mentioned earlier, learning a representation from temporal sequences is an important step in biomedical informatics. For this purpose, Pham *et al.* proposed an end-to-end deep network to read EHRs, save patient history, infer the current state and predict the future [96]. Their approach, called “DeepCare”, used LSTM for multiple admissions of a patient, and also addressed the time irregularities between the consecutive admissions by modifying the *forget gate* of standard LSTM unit. A vector representation was learned for each admission that was fed into the LSTM network. The proposed T-LSTM approach in this study, however adjusts the memory cell by using the elapsed time. The study [96] focused on supervised problem settings. There are other studies in the literature using RNNs for supervised tasks. For instance, Esteban *et al.* [39] proposed an approach to predict whether a patient suffering from kidney failure would survive. RNN was used to predict several conditions related to kidney failure within predetermined time windows. LSTM was used to recognize patterns in multivariate time series of clinical measurements [80]. In this case learning a representation for clinical time series was posed as a multi-label classification problem. LSTM with a fully connected output layer was used for the multi-label classification problem.

Choi *et al.* [30] aimed to mimic how physicians make decisions. RNN was used for this purpose with the patient's past visits in a reverse order. Authors also proposed a different way of using RNNs such that there were two RNNs for visit-level and variable-level attention mechanism. Their goal was to predict diagnoses by first considering the recent visits of the patient and determining which visits and which events are worth paying attention. Another study focusing on predicting the diagnosis of the patient along with the time duration until the next visit was presented [28]. In their study, elapsed time was incorporated as an attribute concatenated to the input rather than a decay factor and another variant of RNN, called GRU, was utilized. On the other hand, Che *et al.* [22] aimed to learn patient similarities directly from temporal EHR data for personalized predictions of Parkinson's disease. GRU unit was used to encode the similarities between the sequences of two patients and dynamic time warping was used to measure the similarities between temporal sequences. A different approach for representation learning from EHR data was introduced [29]. Their method, called Med2Vec, was proposed to learn a representation for both medical codes and patient visits from large scale EHRs. The learned representations were aimed to be interpretable. Authors utilized a multi-layer perceptron to generate a visit representation for each visit vector.

4.2.2 Auto-Encoder Networks

We also would like to briefly mention related studies in auto-encoder networks. In patient subtyping task, we do not have any labels that can be used in a standard supervised setting. Besides, LSTMs are often used for supervised tasks in the literature as summarized above. Therefore, we need an approach where a powerful representation of the temporal sequence can be learned without any supervision. The most popular unsupervised way of utilizing deep networks is auto-encoders that obtain a single representation of the raw input by minimizing a reconstruction error. For instance, LSTM auto-encoders were used to learn representations for video sequences [106]. Authors investigated the performance of learned representations on supervised tasks and reported

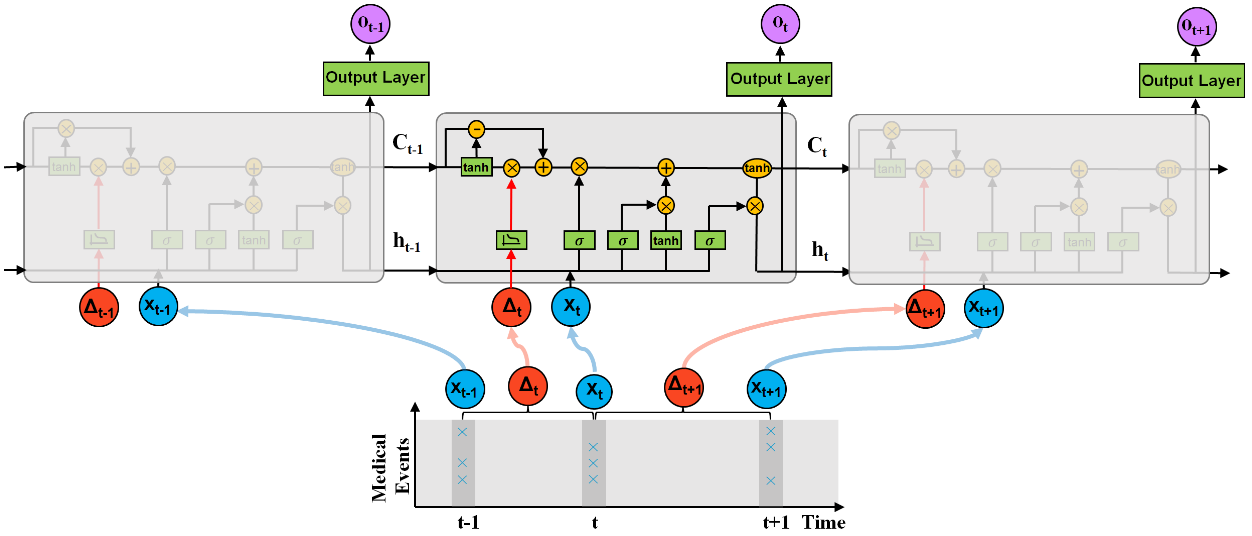


Figure 4.2 Illustration of time-aware long-short term memory (T-LSTM) unit, and its application on medical records. Green boxes indicate networks and yellow circles denote point-wise operators. T-LSTM takes input record and the elapsed time at the current time step. T-LSTM decomposes the previous memory into long and short term components and utilizes the elapsed time (Δ_t) to discount the short term effects.

an increase in the classification accuracy. Auto-encoders were also used to generate a different sequence by using the representation learned in the encoder part. For instance, one RNN encoded a sequence of symbols into a vector, then the decoder RNN mapped the single representation into another sequence [27]. Cho *et al.* [27] showed that their proposed approach can interpret the input sequence semantically and can learn its meaningful representation syntactically.

4.3 Time-Aware Long Short Term Memory

In this section, background information about LSTM networks, the proposed T-LSTM architecture, and the T-LSTM auto-encoder for patient subtyping are introduced.

4.3.1 Long Short-Term Memory (LSTM)

Commonly used standard feed-forward networks such as multi-layer perceptron and convolutional neural networks take an input, extract levels of abstraction and predict the output. From graph theoretic viewpoint, these networks are directed graphs without cycles and thus without a memory. Therefore, feed-forward networks are not suitable for sequential data since the dependencies between consecutive elements of a sequence needs to be modelled. On the other hand, recurrent neural networks (RNNs) are deep network architectures where the connection between hidden units forms a directed cycle, in other words a feedback mechanism. Thus, RNNs naturally construct an internal memory containing information from previous hidden states. Consequently, RNNs are applicable to problems where the system needs to store and update information [15]. Before the popularity of RNNs, Hidden Markov Models (HMM) were proposed for analyzing temporal sequences. The fundamental difference between RNNs and HMMs is the Markov assumption that RNNs do not make. Another advantage of RNNs is being able to process variable length sequences.

In principle, RNNs can keep the long term information of past inputs in the memory, however optimization for long-term dependencies degrades the learning process because of vanishing and exploding gradient problems. RNNs suffer from this problem when gradient becomes nearly zero or gets too large because of the chain multiplications in gradient computation. Vanishing and exploding gradient can also be encountered in feed-forward networks, however this problem becomes more prominent in RNNs because of back propagation through time (BPTT). For instance, non-linear activation functions such as sigmoid σ and $\tanh$ have almost flat regions that makes gradient nearly zero and gradients in BPTT involves multiplying gradients from previous time steps. As a result, even one nearly zero gradient in the chain does not let the model be updated properly. To be able to incorporate the long-term dependencies without violating the optimization process, variants of RNNs have been proposed such as Long Short-Term Memory (LSTM) [58].

A standard LSTM unit comprises of forget, input, output gates, and cell memory. Forget gate is used to control the informational flow from the previous cell memory. While forget gate discards some part of the history, input gate writes new information about the current time step. Output gate works as a filter to control what to output and the cell memory keeps the history collected throughout the sequence which is updated at each time step. One missing point of the current architecture is a mechanism to incorporate the irregular elapsed time into the system. Besides, time irregularity in real world temporal data can be encountered as in EHR datasets. Therefore, we propose a novel LSTM architecture, called Time-Aware LSTM (T-LSTM), where the time lapse between successive records is included in the network architecture.

4.3.2 Proposed Time-Aware LSTM (T-LSTM)

Sometimes temporal sequences do not follow regular time gaps by nature such as patient records where the frequency and the number of patient reports are unstructured. Another reason of non-uniform elapsed times in longitudinal data can be missing information. In such cases, missing elements of a sequence would introduce irregularity and this might alter the pattern in the temporal changes that we want to leverage for predictive tasks. T-LSTM is proposed to incorporate the elapsed time information into the standard LSTM architecture to be able to capture the temporal dynamics of sequential data with time irregularities. The proposed T-LSTM architecture is shown in Figure 4.2 where the input sequence is represented by the temporal patient records.

The most important component of the T-LSTM architecture is the subspace decomposition applied on the cell memory. At each time step, cell memory of the previous hidden state is decomposed into short (C_{t-1}^{S}) and long-term memories ($C_{t-1}^{L} = C_{t-1} - C_{t-1}^{S}$) which represents fast and slow changes in the patient records, respectively. Then a non-increasing function of the elapsed time which transforms the time lapse into an appropriate weight ($g\left(\Delta_t\right)$) is used to discount short-term component ($\hat{C}_{t-1}^{S} = C_{t-1}^{S} * g\left(\Delta_t\right)$) before adding long and short-term components back together (C_{t-1}^{*}). Note that this decomposition is data-driven and the parameters are learned simultaneously with the rest of network parameters by back-propagation. There is no requirement for the activation function to be used in the decomposition network. In fact, several options were tried but we did not observe a drastic difference in the prediction performance of the T-LSTM unit. The basic idea behind T-LSTM is to adjust the cell memory with respect to the elapsed time such that the amount of discount is more if there is a big gap between consecutive elements of the sequence. For instance, if there are months or years between two consecutive reports of a patient, it means that no new information was recorded for a long time for that patient. In this case, if the current hidden state and eventually the output rely on the information from years ago, this might be misleading because of the fact that during that time gap patient might develop new conditions. However, overall temporal pattern should also be preserved. For this reason, we do not apply the discount on the cell memory itself but a portion of it. The overall T-LSTM formulation is provided below.

$$C^S_{t-1} = \tanh\left(W_d C_{t-1} + b_d\right) \qquad \text{(Short-term memory)}$$

$$\hat{C}^S_{t-1} = C^S_{t-1} * g\left(\Delta_t\right) \qquad \text{(Discounted short-term memory)}$$

$$C^L_{t-1} = C_{t-1} - C^S_{t-1} \qquad \text{(Long-term memory)}$$

$$C^*_{t-1} = C^L_{t-1} + \hat{C}^S_{t-1} \qquad \text{(Adjusted previous memory)}$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \qquad \text{(Forget gate)}$$

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \qquad \text{(Input gate)}$$

$$o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \qquad \text{(Output gate)}$$

$$\tilde{C} = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \qquad \text{(Canditate memory)}$$

$$C_t = f_t * C^*_{t-1} + i_t * \tilde{C} \qquad \text{(Current memory)}$$

$$h_t = o * \tanh\left(C_t\right) \qquad \text{(Current hidden state)}$$

where x_t represents the current input, h_{t-1} and h_t are previous and current hidden states, and C_{t-1} and C_t are previous and current cell memories. $\{W_f, U_f, b_f\}$, $\{W_i, U_i, b_i\}$, $\{W_o, U_o, b_o\}$, and $\{W_c, U_c, b_c\}$ are the network parameters of the forget, input, output gates and the candidate memory, respectively. $\{W_d, b_d\}$ are the network parameters of the subspace decomposition. Dimensionality of the parameters are determined by the input, output and the chosen hidden state dimensionality. Δ_t is the elapsed time between x_{t-1} and x_t and $g\left(\cdot\right)$ is a heuristic decaying function such that larger the value of Δ_t, smaller the effect of the short-term memory. Different types of monotonically non-increasing functions can be chosen for $g\left(\cdot\right)$ depending the measurement unit of the time durations. For instance, some datasets might have a fixed time measure such as seconds throughout the sequence and in other cases, elapsed time present in the sequence might have seconds, minutes and hours. In the latter case, we need to decide on one type of unit say seconds. In this case, when there are hours between two consecutive elements, elapsed time will be huge in seconds. Empirically we determined that, $g\left(\Delta_t\right) = 1/\Delta_t$ is appropriate for datasets with small amount of elapsed time and $g\left(\Delta_t\right) = 1/\log\left(e + \Delta_t\right)$ [96] is preferred for datasets with large elapsed times.

In T-LSTM, one of the reasons behind adjusting the memory cell instead of the forget gate is to avoid any changes to the current input's effect to the current output. The current input runs through the forget gate and the information coming from the input plays a role to determine how much memory content we should keep from the previous cell. Therefore, we chose not to modify forget and input gates. Another idea to handle the time irregularity could be to impute the data by sampling new records between two consecutive time steps and then applying LSTM on the augmented data. However, when there is a huge gap between two consecutive elements, we would need to sample many points. In case of an EHR dataset, imputing patient records in such a fashion would not be plausible. Patient records comprise of detailed information and it is not possible to ensure that the imputed records reflect the reality. In the next section, we present how T-LSTM is used in an auto-encoder setting for patient subtyping.

4.3.3 Patient Subtyping with T-LSTM Auto-Encoder

Patient subtyping is posed as a clustering problem since we do not have any prior information about the patient groups in the cohort. Clustering patient sequences directly is not possible. We propose to learn a single representation for each sequence and apply a standard clustering algorithm such as k-means on the representations to obtain patient groups. Auto-encoders provide an unsupervised way to directly learn a mapping from the original data [14]. In the literature, LSTM auto-encoders, where encoder and decoder parts are comprised of LSTM networks, have been used to encode sequences such as sentences [133]. In this study, T-LSTM auto-encoder is introduced to learn an effective single representation of the sequential records of a patient. The proposed auto-encoder has T-LSTM encoder and T-LSTM decoder units with different parameters learned jointly to minimize the reconstruction error. The proposed auto-encoder can capture the long and the short term dependencies by incorporating the elapsed time into the system and learn a single representation that can be used to reconstruct the input sequence.

In Figure 4.3, a single layer T-LSTM auto-encoder mechanism is shown for a sequence with three elements $[X_1, X_2, X_3]$. In this architecture, T-LSTM decoder takes the hidden state and the cell memory of T-LSTM encoder at the end of the input sequence as the initial state and memory. Using a recurrent neural network in an auto-encoder setting is also known as sequence to sequence learning. Input and the elapsed time at the first time step of the decoder are set to zero and the first reconstruction output ($\hat{X}_3$) is assumed to be the last element of the input sequence. Thus, the output sequence is reversed as suggested [106]. This is a common practice in sequence to sequence learning since reversing the order introduces short term dependencies and this is known to facilitate the optimization. When the parameters of the T-LSTM auto-encoder are learned based on the reconstruction error E_r given in Equation 4.3.1, one forward pass of T-LSTM encoder yields the learned representation as the hidden state at the end of the sequence.

$$E_r = \sum_{i=1}^{L} \left\| X_i - \hat{X}_i \right\|_2^2, \tag{4.3.1}$$

where L is the length of the sequence, X_i is the ith element of the input sequence and $\hat{X}_i$ is the ith element of the reconstructed sequence. The hidden state at the end of the sequence carries concise information about the input such that the original sequence or a target sequence can be reconstructed from it. In other words, representation learned by the encoder is a summary of the input sequence [27]. Number of layers and dimensionality of the representation can be determined based on the complexity of the problem. According to our observation, learning a lower dimensional representation compared to the input dimensionality requires a higher model capacity, therefore we preferred to use a two layer T-LSTM auto-encoder in our experiments.

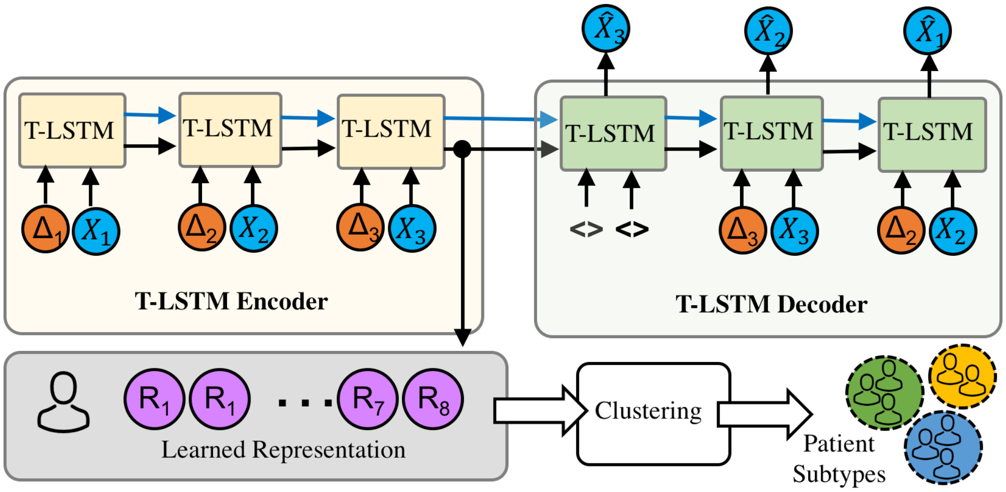


Figure 4.3 Finding patient groups with a single-layer T-LSTM Auto-Encoder. Blue arrows denote the cell memory and the black arrows denote the hidden states. After the representations (R_i, $i = 1, 2, \cdots, 8$) are learned for the population, we can cluster the patients and obtain subtypes for each group.

After a single representation for each patient is obtained, patients can be grouped by a clustering algorithm such as k-means. Since we do not have any information and assumption about the structure of the clusters, we propose to use k-means because of its simplicity. In Figure 4.3, a small illustration of clustering the patient cohort for 8 patients is given. In the figure, learned representations are denoted by R. If R has the capability to capture the patient characteristics from temporal medical records, then clustering algorithm is expected to group patients with similar properties together. This consequently provides subtypes in a patient cohort. When there is a new patient, learned T-LSTM encoder is used to retrieve the representation of the patient and subtype of the patient can be found by obtaining the cluster whose centroid gives the minimum distance for the new patient. As such, learned representations could be used for supervised tasks as well.

4.4 Experiments

To investigate the performance of the proposed T-LSTM and T-LSTM auto-encoder, experiments on synthetic and real world datasets were conducted. Performance comparisons between T-LSTM, MF1-LSTM, MF2-LSTM [96], LSTM, and logistic regression were made for supervised and unsupervised settings on synthetic and real world datasets. MF1-LSTM and MF2-LSTM denote Modified Forget Gate LSTM approaches adopted from [96]. MF1-LSTM multiplies the output of the forget gate by $g\left(\Delta_t\right)$ such as $f_t = g\left(\Delta_t\right) * f_t$, whereas MF2-LSTM utilizes a parametric time weight such as $f_t = \sigma\left(W_f x_t + U_f h_{t-1} + Q_f q_{\Delta_t} + b_f\right)$ where $q_{\Delta_t} = \left(\frac{\Delta_t}{60}, \left(\frac{\Delta_t}{180}\right)^2, \left(\frac{\Delta_t}{360}\right)^3\right)$ when Δ_t is measured in days similar to [96].

The application of T-LSTM auto-encoder on patient subtyping is presented on a real world dataset (PPMI) and subtyping results are discussed. T-LSTM [1] was implemented in Tensorflow and mini-batch stochastic Adam optimizer was used during experiments. All the weights were learned simultaneously, and same network settings and parameters were used for all the deep methods for fair comparison. Therefore, fixed number of epochs were chosen during the experiments instead of using a stopping criteria. Since the longitudinal patient data used in experiments has variable length sequences, batches with same sequence lengths were generated instead of padding the original sequences with zero to make every sequence to be of same length. The main reason of avoiding zero padding, which a commonly used practice, was that the patient data generally contains sparse vectors representing diagnosis and other medical features and zeros also indicate a valid meaning other than absence. Therefore, we did not add extra zeros to patient sequences. Note that in this study, we did not use the publicly available large scale ICU database, called MIMIC [66]. Since it is challenging to find public EHR datasets due to privacy concerns, MIMIC is an important source of patient data for biomedical informatics research. MIMIC database contains clinical information recorded at the end of patient's hospital stay. As a result, majority of the patients in the MIMIC database have one or two records. Since patients do not have enough temporal variations, we could not use the MIMIC database to test the performance of proposed T-LSTM network.

[1] Available at https://github.com/illidanlab/T-LSTM

4.4.1 Synthetic Dataset

4.4.1.1 Supervised Experiment

In this experiment, synthetically generated EHR data [2] was used for the classification task. The aforementioned synthetic data has records of up to $100,000$ patients with lab results, diagnoses, and start and end dates of the admissions. Similar to real world EHR datasets, a unique patient ID is assigned to each patient. We refer to [69] for further details of the data generation process. Although the dataset is synthetically generated, it contains similar characteristics as a real EHR data. The proposed T-LSTM was used to classify healthy patients and patients with Diabetes Mellitus. For this binary classification task, $6,730$ patients were sampled with an average of 4 admissions. Input features were multi-hot vectors containing the diagnoses given in the corresponding admission with a vocabulary size of 529. Since this task predicts the binary class label for patient sequences, T-LSTM was used in the standard way, where the hidden state at the end of the patient sequence is mapped to a binary label, other than auto-encoder.

For this task, a single layer T-LSTM, MF1-LSTM, MF2-LSTM networks and traditional logistic regression were tested to compare the performance based on area under ROC curve (AUC) metric for 50 epochs. In this experiment, number of hidden and softmax layer neurons were chosen as $1,028$ and 512, respectively. In logistic regression experiments, admissions were aggregated for each patient without incorporating the elapsed time. We also tried to incorporate the elapsed time as a weight by using the same non-increasing function used in T-LSTM during the aggregation of admissions. However, this approach did not improve the performance in our case. The results are summarized in Table 4.1.

In summary, T-LSTM was observed to provide a better AUC performance compared to baseline approaches. Logistic regression, which is a commonly used method in biomedical informatics applications, performed very poorly in our case. The way to represent the sequential data could be improved further for logistic regression, but aggregation of the admissions for each patient did not perform well for this task. Supervised experiments showed that LSTM networks can be

[2] http://www.emrbots.org/

Table 4.1 Supervised synthetic EHR experimental results showing the average AUC of testing on 10 different splits. Training and testing proportion was chosen as 70% to 30%.

Methods	Avg. Test AUC	Stdev.
T-LSTM	**0.91**	0.01
MF1-LSTM	0.87	0.02
MF2-LSTM	0.82	0.09
LSTM	0.85	0.02
LR	0.56	0.01

more helpful to learn temporal patterns of EHR data compared to logistic regression. In addition, modifying the cell memory gave a better classification performance. According to our observation, MF1-LSTM and MF2-LSTM had better and sometimes similar results as the traditional LSTM for the tasks in our experiments. We did not observe any bias regarding the sequence lengths during the experiments.

4.4.1.2 Unsupervised Experiment

In this experiment, we investigate the expressive power of the representation learned from the T-LSTM auto-encoder. For this purpose, a synthetic data was randomly generated and the clustering results were evaluated. Since we know the ground-truth of the synthetic data, we computed the Rand index (RI), given in Equation 4.4.1 [83], of the clustering to observe the discriminative power of the learned representations. A large value of Rand index indicates that the learned representations can let the clustering be close to the ground-truth.

$$\text{RI} = (\text{TP} + \text{TN})/(\text{TP} + \text{FP} + \text{FN} + \text{TN}), \tag{4.4.1}$$

where TP, TN, FP, FN are true positive, true negative, false positive and false negative, respectively. Note that $0 \leq \text{RI} \leq 1$.

The results on a synthetic dataset containing 4 clusters generated from a mixture of normal distributions with four different means and the same covariance are reported. A data point in the synthetic dataset was a sequence of vectors and the values of the sequences were increasing with

Table 4.2 Average Rand index of k-means over 10 runs. T-LSTM auto-encoder outperforms LSTM and MF1-LSTM auto-encoders.

Method	Mean RI	Std
T-LSTM	0.96	0.05
MF1-LSTM	0.85	0.13
LSTM	0.90	0.09

time. Some of the elements in the sequences were discarded randomly to introduce unstructured elapsed time and obtain variable sequence lengths of sizes 4, 6, 18, 22, and 30. Dimensionality of the vectors was 5 and the dimension was reduced to 2 by the T-LSTM auto-encoder to be able to plot the representations in a 2-D space. The hidden state dimension of the second layer T-LSTM encoder was chosen as 2, therefore the learned representations were 2-dimensional single vectors. The learned representations were clustered by k-means, where k was set to 4. Representation learning was repeated 10 times with different initializations of k-means and the average Rand index of clustering is reported for T-LSTM, LSTM and MF1-LSTM auto-encoders in Table 4.2. The non-increasing heuristic function was chosen as $g\left(\Delta_t\right) = 1/\log\left(e + \Delta_t\right)$. For this experiment, we compared the performances of T-LSTM, MF1-LSTM and LSTM excluding MF2-LSTM. Since the time gap of the data used in this experiment does not relate to an actual time measurement such as days, MF2-LSTM was excluded.

Table 4.2 shows that the T-LSTM outperforms the baselines and T-LSTM auto-encoder can learn the underlying structure of the input sequence with varying elapsed times such that the representations obtained by T-LSTM encoder could be clustered. In this example, performance of MF1-LSTM was found to be better than LSTM on average. A visual example of one of the trials is also shown in Figure 4.4, where the 2-dimensional representations obtained by the three approaches are plotted.

In Figure 4.4 different colors denote ground-truth assignments of different clusters. Representations learned by T-LSTM provided more compact groups in the 2-D space leading to a more accurate clustering result compared to the standard LSTM and MF1-LSTM. The change in the objective values of T-LSTM, MF1-LSTM and LSTM with respect to the number of epochs were also

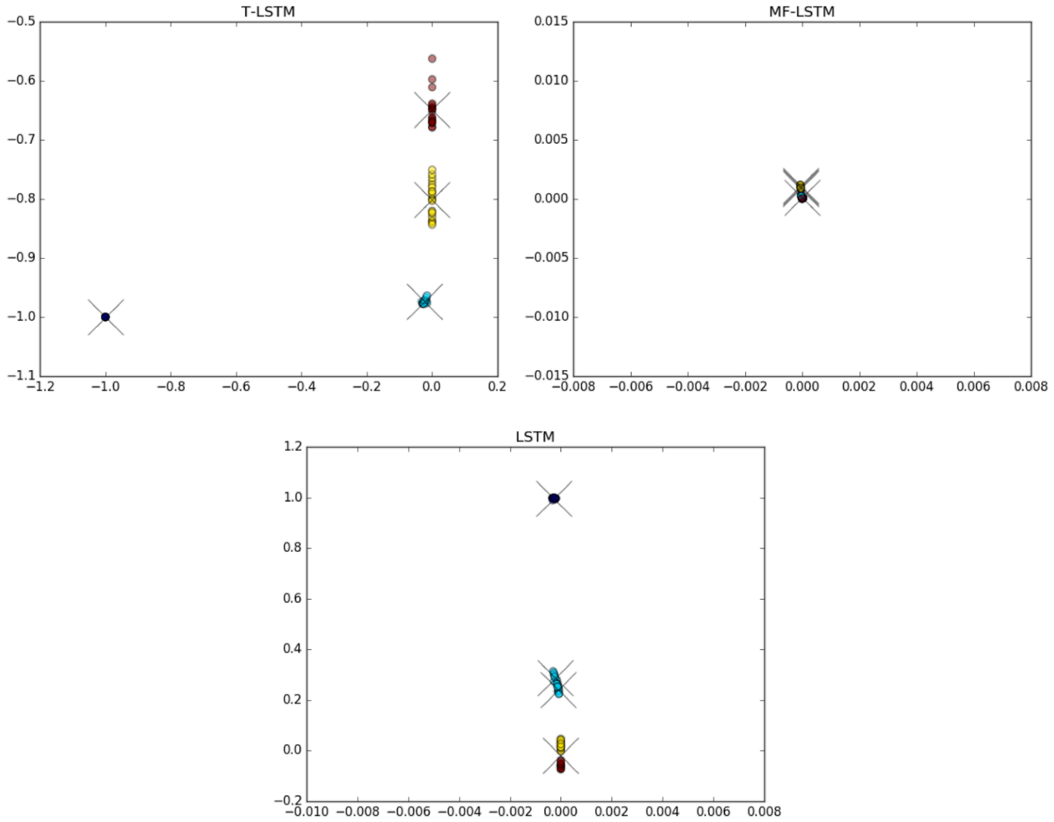


Figure 4.4 Illustration of the clustering results. Different colors denote ground-truth assignments of different clusters. T-LSTM auto-encoder learns a mapping for the sequences such that 4 separate groups of points can be represented in the 2-D space.

compared in Figure 4.5 for the trial illustrated in Figure 4.4. It is observed that the modifications related to the time irregularity does not affect the convergence of the original LSTM network in a negative way.

4.4.2 Parkinson's Progression Markers Initiative (PPMI) Data

In this section, we present experimental results for a real world dataset. Parkinson's Progression Markers Initiative (PPMI) [3] is an observational clinical and longitudinal study comprising of evaluations of people with Parkinson's disease (PD), those people with high risk, and those who are healthy [35]. PPMI aims to identify biomarkers of the progression of Parkinson's disease. PPMI

[3] www.ppmi-info.org

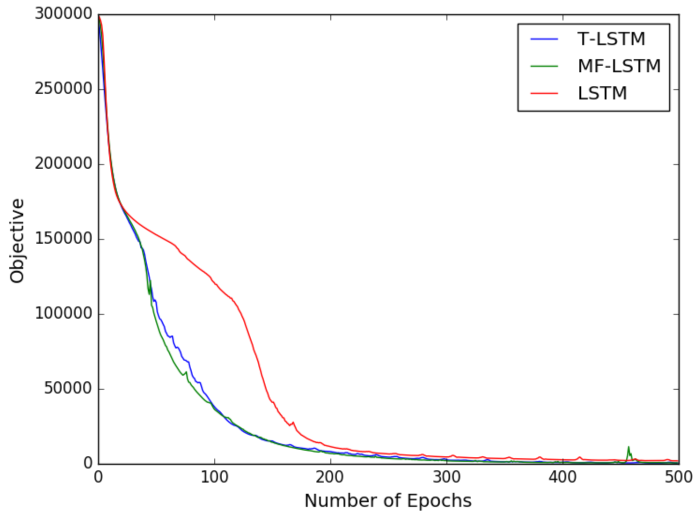


Figure 4.5 Change in the objective values of T-LSTM, MF1-LSTM and LSTM with respect to 500 epochs. It is observed that the modifications related to the time irregularity does not deteriorate the convergence of the original LSTM network.

data is a publicly available dataset which contains clinical and behavioral assessments, imaging data, and biospecimens, therefore PPMI is a unique archive of PD [35]. As with many EHRs, PPMI is a longitudinal dataset with unstructured elapsed times.

In our experiments, we used the pre-processed PPMI data of 654 patients given in [22]. Che *et al.* [22] collected patients with Idiopathic PD or non PD, imputed missing values, used one-hot feature representation for categorical values, and encoded data abnormalities as 1 and 0. As a result, the dataset we used has $15,636$ records of 654 patients with an average of 25 sequences (minimum sequence length is 3). Authors of [22] also categorized data as features and targets, where the features are related to patient characteristics and the targets correspond to the progression of PD. A total of 319 input features consist of motor symptoms/complications, cognitive functioning, autonomic symptoms, psychotic symptoms, sleep problems, depression symptoms, and hospital anxiety and depression scale. A total of 82 targets are related to motor sign, motor symptom, cognition, and other non-motor factors [22]. Summary of the PPMI data characteristics used in this study can be found in Table 4.3.

As it can be seen in Table 4.3, the elapsed time was measured in months. From 1 month to nearly 24 months gap between successive records of patients was encountered in the dataset.

Table 4.3 Details of PPMI data used in this study. Elapsed time encountered in the data is measured in months and it varies between 1 month to nearly 24 months. Here, the elapsed time interval is not the time interval of PPMI data recording, but elapsed times seen in records of individual patients.

Number of Patients	654
Elapsed Time Interval	[1, 26]
Average Sequence Length	25
Feature Dimensionality	319
Target Dimensionality	82

Table 4.4 Average mean square error (MSE) for 10 different train-test splits for T-LSTM, LSTM, MF1-LSTM, and MF2-LSTM. T-LSTM yielded a slightly better result than the standard LSTM in the presence of the unstructured time gaps.

MSE	T-LSTM	MF1-LSTM	MF2-LSTM	LSTM
Mean	0.50	0.53	0.51	0.51
Std	0.018	0.017	0.012	0.017

Following experiments were conducted on PPMI data to show the performance of the proposed subtyping approach.

4.4.2.1 Target Sequence Prediction

In this experiment, T-LSTM was used to predict the target sequence of each patient. For this purpose, we divided the data into different train (70%)-test (30%) splits and report the mean square error (MSE) between the original target sequence and the predicted target sequence. Average MSEs of 10 different train-test splits for T-LSTM, LSTM, MF1-LSTM and MF2-LSTM are given in Table 4.4. Same step size and the number of epochs were used for all the three methods. The non-increasing heuristic function of the elapsed time was chosen as $g\left(\Delta_t\right) = 1/\log\left(e + \Delta_t\right)$ for PPMI data.

We also investigated target features on which T-LSTM performed the best. The commonly encountered target features where the T-LSTM provided lower MSE than LSTM, MF1-LSTM and MF2-LSTM are reported in Table 4.5. The main observation about the target features in Table 4.5 was that they are related to the effects of Parkinson's disease on the muscle control such as finger tapping, rigidity, and hand movements. In addition, T-LSTM predicted the target value of

Table 4.5 Some common target features from PPMI dataset on which T-LSTM performed better than LSTM and MF1-LSTM during 10 trials. These target features are mainly related to the effects of Parkinson's disease on muscle control.

Code	Name
NP3BRADY	Global spontaneity of movement
NP3RIGRU	Rigidity - RUE(Right Upper Extremity)
NP3FTAPR	Finger Tapping Right Hand
NP3TTAPR	Toe tapping - Right foot
NP3PRSPR	Pronation-Supination - Right Hand
NP3HMOVR	Hand movements - Right Hand
NP3RIGN	Rigidity - Neck
NP2DRES	Dressing
PN3RIGRL	Rigidity - RLE (Right Lower Extremity)
DFBRADYP	Bradykinesia present and typical for PD
NP3RTARU	Rest tremor amplitude - RUE
NP3PTRMR	Postural tremor - Right Hand
MCATOT	MoCA Total Score

Bradykinesia, which encompasses several of the problems related to movement, and MoCA (Montreal Cognitive Assessment) Total Score, which assesses different types of cognitive abilities with lower error than other methods. This result showed that the reported target features are sensitive to elapsed time irregularities and discounting the short-term effects by the subspace decomposition of memory cell helps to alleviate this sensitivity.

4.4.2.2 Patient Subtyping of PPMI Data

In the next experiment, T-LSTM auto-encoder was used to obtain subtypes of the patients in the PPMI dataset. The T-LSTM encoder was used to learn a representation from the input feature sequence of each patient and the T-LSTM decoder generated the target sequence. Parameters of the auto-encoder were learned to minimize the squared error between the original target sequence and the predicted target sequence. The learned representations were used to cluster the patients by the k-means algorithm as discussed before.

Since we did not know the ground-truth for the clustering, a statistical analysis was conducted to assess the subtyping performance. For this purpose, clustering results were statistically analyzed at the time of 6 years follow-up in the PPMI study. Features including demographics, motor severity measures such as Unified Parkinson's Disease Rating Scale (MDSUPDRS), Hoehn and Yahr staging (H&Y), non-motor manifestations such as depression, anxiety, cognitive status, sleep disorders, imaging assessment such as DaTScan, as well as cerebrospinal fluid (CSF) biomarkers were taken into account. In order to interpret the clustering results in terms of subtyping, clusters were compared using Chi-square test for the categorical features, F-test for the normal continuous features, Kruskal-Wallis test for the non-normal continuous features, and Fisher's exact test for the high sparsity features. According to the previous Parkinson's disease studies, if the p-values of the aforementioned features are less than 0.05, a significant group effect is considered for the associated features [43]. Thus, if a method can obtain a lot of features with small p-values, it indicates that the method provides a more sensible patient subtyping result.

We tried several k values for the k-means algorithm. We often observed that there were two main clusters, therefore we reported the clustering results for $k = 2$. We conducted several tests with different parameters. According to our observation, LSTM did not provide adequate number of features with p-values less than 0.05 and most of the patients were generally grouped into one cluster. In Table 4.6, features of small p-values and cluster means of the features are presented for T-LSTM, MF1-LSTM and MF2-LSTM. As it can be seen from the table, T-LSTM had more discriminative features than MF1-LSTM and MF2-LSTM.

In Table 4.6, high cluster mean indicates that the symptoms of the corresponding feature are more severe for that cluster and the PD patients have lower cluster mean for DaTScan feature. Note that one of the observed features of T-LSTM in Table 4.6 is MoCA which was predicted better by T-LSTM in the target sequence prediction experiment. Finally, we illustrated the patient subtyping results of T-LSTM with heat map illustration in Figure 4.6. In this figure, shade of red color represents the cluster mean which is higher than the total mean of the patients and the shades of blue color show lower mean values for the corresponding feature with the p-value< 0.05. Subtypes and features which are significant for each subtype can be observed from the heat map. For instance, DaTSCAN features were found to be significant for subtype I, whereas subtype II

Table 4.6 Results of the statistical analysis for T-LSTM, MF1-LSTM and MF2-LSTM. DaTScan1 corresponds to DaTScan SBR-CAUDATE RIGHT, DaTScan2 is DaTScan SBR-CAUDATE LEFT, and DaTScan4 is DaTScan SBR-PUTAMEN LEFT.

Feature	P-Value	Cluster1 Mean	Cluster2 Mean
T-LSTM			
BJLO	9.51×10^{-8}	16.5	24.7
MoCA	0.001	40.0	41.2
DaTScan1	0.042	2.29	2.07
DaTScan2	0.027	2.31	2.08
DaTScan4	0.001	1.4	1.1
MF1-LSTM			
CSF-Total tau	0.007	87.9	46.72
MoCA	2.16×10^{-17}	47.5	41.05
SDM	0.005	58.5	41.5
MF2-LSTM			
HVLT-Retention	0.03	0.84	0.83
SDM	0.007	36.61	41.68

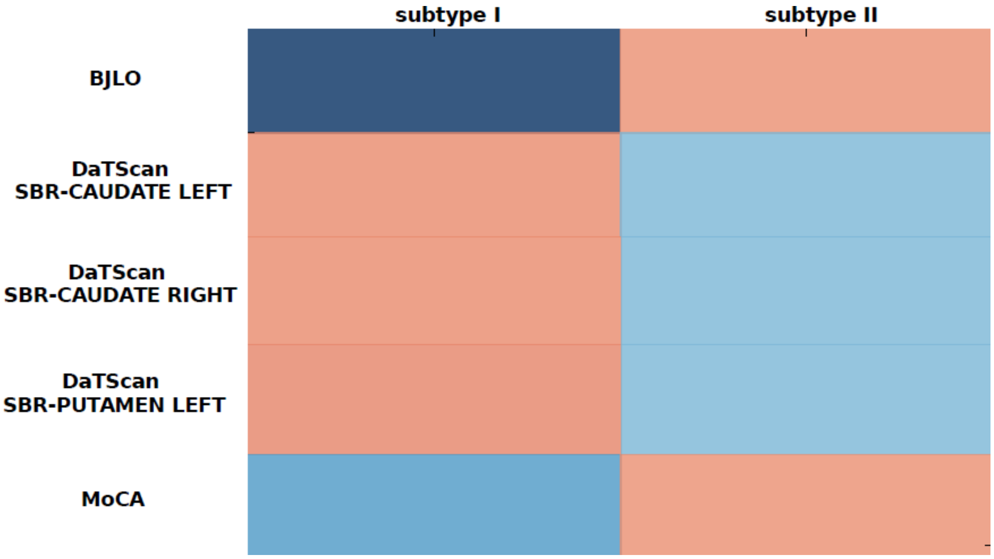


Figure 4.6 Heat map illustration of the patient subtyping results of T-LSTM for two clusters. Light red color represents the cluster mean which is higher than the total mean of the patients and the shades of blue show lower mean values for the corresponding feature with p-value< 0.05.

was defined by BJLO (Benton Judgement Line Orientation) and MoCA features. Note that the dataset contains healthy subjects as well. It is known that PD patients have lower DaTScan SBR values than healthy subjects [125]. Hence, we can conclude from Figure 4.6 that subtype II can be considered as PD patients. We can also observe from Figure 4.6 that cluster means of BJLO and MoCA are very low (darker shades of blue) for subtype I compared to subtype II.

4.5 Summary

In this study, time-aware LSTM is proposed to improve the LSTM performance when there is an elapsed time irregularity in the temporal sequence, whereas traditional LSTM naturally assumes a uniform time gap throughout the sequence. T-LSTM does not make any assumption about the elapsed time unit such that the time gap does not have to be measured in days or years and thus it can be adopted by other domains dealing with different types of sequences. T-LSTM adjusts the previous memory content of an LSTM unit by a decaying function of the elapsed time in a way that longer the time lapse, less the influence of the previous memory content on the current output. The proposed T-LSTM was tested for supervised and unsupervised tasks on synthetic data and real world datasets. Patient subtyping, which can be defined as clustering sequential patient records, was analyzed on a publicly available real world dataset called Parkinson's Progression Markers Initiative (PPMI). For the subtyping purpose, T-LSTM auto-encoder is used to learn powerful representations for the temporal patient data, and the learned representations are used to cluster the patient population.

Chapter 5

Decoupled Memory Gated Recurrent Network

In Chapter 4, a modified long short-term memory (LSTM) architecture was introduced to address patient sub-typing task. The sub-typing task is essentially a patient clustering problem, where the data points are historical medical records. In this problem, temporal pattern in the patient's medical records needs to be efficiently encoded into a vector such that time sequences of patients can be clustered into different groups using an off-the-shelf clustering algorithm. In addition to the temporal pattern, elapsed time irregularities in health records play an important role in clinical decision making. For this reason, in Chapter 4, we proposed to integrate elapsed time in the standard LSTM unit. In particular, cell memory of the LSTM unit was decomposed into long and short term components and the short term memory was discounted using a time decaying weight. Empirical analysis of T-LSTM provided an evidence that further modifications on the internal memory of LSTM can improve the predictive performance for irregular time series data. This observation is particularly important for healthcare tasks since patient's medical history depicts an irregular time series data.

In this chapter, the internal memory of RNNs and external memory networks are further investigated. Moreover, a new gated recurrent model with a modified internal memory is proposed to extend the T-LSTM approach proposed in Chapter 4. The proposed model, decoupled gated recurrent network (DM-GRN), addresses an important potential issue of the T-LSTM model. When the actual elapsed time between consecutive time steps is not known, T-LSTM does not modify the cell memory and behaves like a standard LSTM model. On the other hand, the absence of the elapsed time information does not necessarily indicate uniform elapsed time. The proposed DM-GRN introduces separate short and long term memory components to address the potential time irregularities without utilizing the actual elapsed time. In particular, the new architecture aims to alleviate the vague distinction between the long and short term memories in the standard LSTM unit while employing an internal attention mechanism to update the long term memory component. Experiments on synthetic and real world data are conducted for quantitative and qualitative evaluations of the proposed architecture. Applications of DM-GRN on healthcare and traffic prediction are discussed.

5.1 Introduction

Artificial intelligence and machine learning have been in the limelight and the interest continues to grow. Companies prevalently prefer data-driven models that offer effective ways to harness their data, and more importantly, provide predictive capability to improve their services. As a consequence, data has become an important asset to design reliable learning models. Depending on the application domain, data is obtained via various kinds of acquisition techniques. For instance, traffic speed data is collected with GPS equipped cars, click stream data is recorded via e-commerce web pages, transactional data is collected by financial service providers, and digital patient data is stored via electronic health record (EHR) systems. Data acquisition technique specifies the characteristics of the data. One of the most common characteristic is the temporal dependency between the consecutive measurements or records. For this reason, a number of applications require the ability to deal with temporal data. In finance and healthcare, in particular, it is imperative to capture the temporal patterns present in datasets since the temporal dynamics reveals important insight

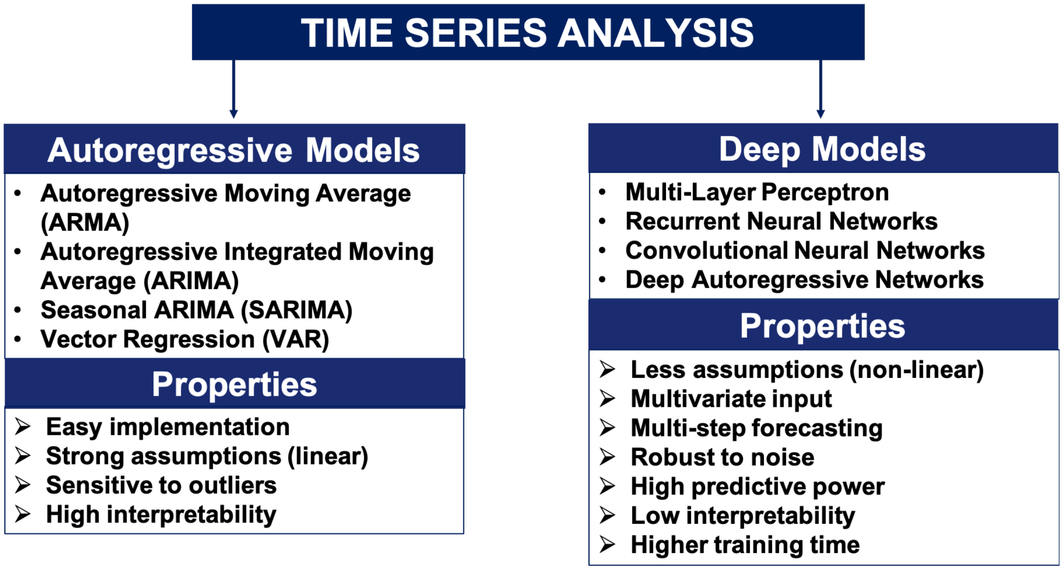


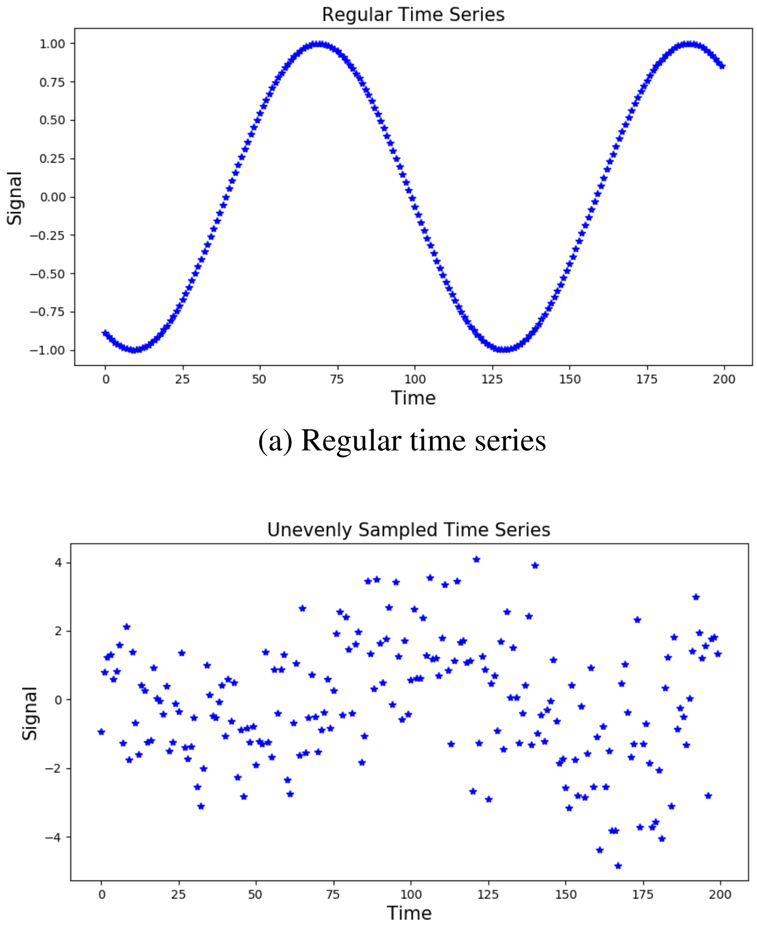
Figure 5.1 Comparison between some of the auto-reggresive models and deep learning techniques for time series analysis.

about subject's behavior. For instance, temporal analysis of patient's medical history can facilitate a more reliable risk prediction, which consequently assists in clinical decision making.

There are various time series analysis techniques with different stochastic processes. Prominent traditional time series analysis techniques are autoregressive, integrated, moving average models, and their combinations (e.g., autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA)) [5]. The traditional linear models are simple to implement and do not require a large amount of data for training. However, their model complexity is not adequate to analyze complex multi-variate time series data with elapsed time irregularities. Hidden Markov Model (HMM), on the other hand, represents the time series as a Markov process, where the future states depend only upon the current states. Consequently, this assumption of HMM results in a memoryless system. Although the HMM is used for various time series applications, such as speech recognition [98], Markov property is a very strong assumption that real world datasets often do not follow. As a result, a data-driven and a non-linear model with fewer assumptions is necessary to tackle challenging real world time series datasets, such as EHRs.

With the advancements in hardware and optimization techniques, deep learning has been a flourishing family of data-driven models yielding impressive performance in many domains. Recurrent neural network (RNN) and its variants have been developed to tackle time series data. In addition to RNNs, multi-layer perceptron [62], convolutional neural networks [62], and deep autoregressive [53] models are utilized for time series analysis. Properties of traditional time series analysis approaches and deep models used for time series data are summarized in Figure 5.1. The fundamental differences between RNNs and the traditional time series analysis models are non-linearity and memory properties. RNNs, in theory, can capture long term dependencies without any Markov property and linearity assumptions. However, as discussed in Chapter 4, RNNs are not robust against vanishing gradients due to non-linear activation functions and back-propagation through time (BPTT). For this reason, variants of the RNN architecture, such as long short term memory (LSTM) [58] and gated recurrent unit (GRU) [31] networks, are more prevalent. Both architectures introduce gate mechanisms to regulate the information flow from past to future time steps and extract features (e.g., input, forget, output gates in LSTM). On the other hand, the key improvement LSTM and GRU have over traditional RNN is the additive memory component that prevents gradient values from vanishing.

LSTM and GRU networks implicitly assume that time sequence has regular elapsed times. An example time series data with regular and irregular elapsed times is shown in Figure 5.2. As seen in Figure 5.2a, sine wave has a uniform sampling rate. However, in many applications, elapsed time is not uniform due to several factors, such as missing values and irregular sampling rate. For instance, time gap between consecutive patient records varies depending on the frequency of hospital visits or admissions. Since the sampling rate of patient records is not known, EHR data cannot be easily imputed to make the temporal sequence regular. To integrate the elapsed time into the standard LSTM architecture, a modified LSTM architecture, named time-aware LSTM (T-LSTM) [11], was introduced in Chapter 4. T-LSTM, given in Figure 4.2, decomposes the cell memory of the LSTM unit into two components that permits modifying the memory content without deteriorating the overall temporal pattern stored in the memory. In theory, LSTM proposes two kinds of memories, namely cell memory and the hidden state. Cell memory aims to maintain the long term patterns, where the hidden state captures the short term changes. However, mathemat-



(a) Regular time series

(b) Irregular time series

Figure 5.2 Regular and irregular time series data example.

ical model of LSTM poses a coupled memory, where the hidden state is obtained using the cell memory weighted by output of a non-linear gate function. Besides, we empirically presented the benefits of decomposing the cell memory into long and short-term components in T-LSTM. On the other hand, T-LSTM might face an issue when the elapsed time is not available. In such cases, the elapsed time is assumed to be uniform although time irregularity exists due to various reasons, such as noise. Under uniform elapsed time assumption, T-LSTM is simplified to the standard LSTM unit.

In this chapter, a new gated architecture with decoupled memory, named decoupled memory gated recurrent network (DM-GRN), is proposed as an alternative approach to T-LSTM model. In DM-GRN architecture, the memory is constructed as a summation of two components that are designed to focus on different dynamics of the input time series. For this purpose, a new gated unit is designed with two sigmoid gates to extract features from the input and the recurrent component, and two $\tanh$ memory layers to construct short and long term memories. Gate outputs are used to assign weights to short and long term memory components before they are assembled to form the overall memory or hidden state. The difference between the consecutive elements of the time series is used as the input to the forget gate, facilitating an automatic way to capture time irregularities. As a result, the proposed network can still regulate the memory without any elapsed time and sampling rate information. The long term memory component of the proposed DM-GRN architecture is computed via an attention mechanism used to weight the past hidden states. Inspired by the external memory networks [52, 118], a limited number of past time steps are stored in a memory matrix to computed the weighted sum. Main contributions of the DM-GRN architecture are as follows.

- Decoupling the internal memory into long and short term memories provides more insight about the different dynamics in the time series data compared with coupled memory networks.

- Using the difference between consecutive elements as one of the inputs to the network aims to capture the time irregularities inherently even the elapsed time is not explicitly provided.

- An internal attention mechanism is proposed to compute the long term memory component. In particular, a weighted sum of the past hidden states is used instead of considering only the previous time step. A fixed length time window is used for attention to reduce the complexity of training.

Several synthetic and real-world data experiments are conducted and results are compared to several other gated architectures, to empirically validate the benefits of the decoupled memory. In the next sections, literature review and further details on internal and external memory networks are presented.

5.2 Literature Review

LSTM networks were proposed to tackle the vanishing gradient problem of the standard RNN architecture [58]. The vanishing gradient is avoided by the gated recurrent connections and an extra memory unit, named cell memory. The main advantage of the LSTM network over the standard RNNs is the additive structure of the memory. The cell memory is obtained by a weighted sum and the weights are determined by the gates. In literature, gated architectures are reported to perform better than standard RNNs. However, there are also studies discussing the reliability of the gated architectures for tasks with specific properties. Russell *et al.* [103] investigated and compared the memory properties of RNN, LSTM, and gated recurrent unit (GRU). One of the important conclusions drawn from their empirical evaluations is the unreliability of LSTM and GRU with the tasks of fixed delay recall. On the other hand, gated architectures perform better when the task requires a memory to store and update information. This result indicates that the memory due to the feedback loop of the standard RNN is not adequate to write and delete information through different time steps.

Ablation studies have been conducted to further analyze the importance of each LSTM gate and the cell memory. Levy *et al.* [78] investigated LSTM networks from a different perspective. Authors argued that the gates offer powerful representations, which stem from the fact that LSTMs are the combinations of two recurrent models. It is also empirically shown for several natural language processing (NLP) tasks (e.g., language modelling, question answering, machine translation) that output gate and the candidate memory have minor contributions compared with input and forget gates. The analysis in this paper demonstrates that the main advantage of LSTM is the element-wise weighted sum rather than the non-linear recurrent transitions. In addition, degradation in the performance is reported when the cell memory is ablated. On the other hand, the multiplicative recurrent connections in the candidate memory and in the gates do not have a significant contribution to the representational power.

In literature, there are also efforts to visualize the gate and memory contents of gated architectures to better comprehend the underlying mechanism. For this purpose, Tang *et al.* [109] proposed a memory visualization approach to observe the behaviors of LSTM and GRU units for automatic speech recognition. In particular, activation patterns can be observed to investigate the ways different RNN architectures encode the information. Based on the experimental results, authors argued that information encoded by GRU is more distributed than by LSTM. In addition, activation values of GRU hidden state concentrates between $[-1, 1]$ unlike the activation values of the LSTM cell memory. Authors suggested that the constraint activation values of GRU can facilitate the model training. It is also reported that LSTM tends to remember longer sequences than GRU based on visualization result of the temporal trace.

Long and short term changes in time series can also be analyzed with signal level decomposition. One of the popular methods that analyzes frequency content of the time series is Fourier transform. On the other hand, Fourier transform does not learn frequencies in the time series, but utilizes predefined frequencies. For this reason, it is not suitable for forecasting applications. Neural networks have been used also for signal level decomposition, which can be learned from the signal. For instance, Godfrey *et al.* introduced a neural decomposition technique to improve the generalizability performance of time series forecasting [49]. The proposed method performs a decomposition similar to inverse discrete Fourier transform (iDFT), however the frequencies are learned using sinusoidal activation functions. Compared with RNNs, the neural decomposition of time series is better at handling the unevenly sampled data.

5.3 Methodology

In this section, background information about memory mechanism in gated recurrent models and external memory architectures is presented, and the proposed approach is introduced.

5.3.1 Memory in Recurrent Networks

Memory is a vital property of human brain and an indispensable component of computers. Being able to remember past experiences, adding new knowledge, and discarding irrelevant information are building blocks of the memory. Memory is crucial not only for humans and computers, but also for dynamic systems. In particular, the memory in learning problems involving temporal inputs facilitates inferring the time dependencies between consecutive elements, and thus enhances the predictive performance. RNN and its variants offer such a memory due to their feedback loop. In literature, LSTM and GRU are the most commonly employed RNN architectures for temporal problems. Different variants of LSTM are also available to address specific tasks. For instance, we proposed T-LSTM to address irregular elapsed time in Chapter 4. Mathematical models of LSTM, GRU, T-LSTM, and a T-LSTM variant proposed by Yang *et al.* [124] are compared in Table 5.1. The models summarized in the table are known as gated architectures due to their multiple neural layers with sigmoid activation. Common components of the gated architectures are the additive memory and sigmoid gates to regulate the information flow. The prominent differences between different gated architectures are often the number of gates and definition of the memory. For instance, T-LSTM [11] and its variant [124] modify the cell memory using elapsed time to apply a time decay on the short term memory component. T-LSTM decomposes the cell memory using a single non-linear neural layer, whereas the T-LSTM variant represents the short and long term memories with two different non-linear neural layers. T-LSTM applies the time decay in a multiplicative approach, whereas the T-LSTM variant adds the time decay as another neural layer. In summary, memory in recurrent networks can be modified in various ways, however the crucial point is updating the memory in an additive way. In the following section, memory in gated architectures is investigated in detail.

5.3.1.1 Internal Memory of Recurrent Networks

RNN and its variants depict a family of networks that have internal memories. The existence of the memory indicates that this type of networks does not follow the Markov property. Thus, recurrent networks can be more flexible and versatile to infer complicated relationships between the elements of a time sequence. Consequently, the memory component facilitates forecasting and predictive tasks on complicated time series data. The success of RNN models is mainly driven by

Table 5.1 Mathematical models of gated RNN architectures covered in this chapter.

Architecture	Model
LSTM [58]	$f_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)$ (5.3.1) $i_t = \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)$ (5.3.2) $o_t = \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)$ (5.3.3) $\tilde{C} = \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right)$ (5.3.4) $C_t = f_t * C_{t-1} + i_t * \tilde{C}$ (5.3.5) $\mathbf{h}_t = o * \tanh\left(C_t\right)$ (5.3.6)
GRU [31]	$z_t = \sigma\left(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z\right)$ (5.3.7) $r_t = \sigma\left(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r\right)$ (5.3.8) $\tilde{\mathbf{h}}_t = \tanh\left(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h\left(r_t * h_{t-1}\right) + \mathbf{b}_h\right)$ (5.3.9) $h_t = \left(1 - z_t\right) * h_{t-1} + z_t * \tilde{\mathbf{h}}_t$ (5.3.10)
T-LSTM [11]	$C^S_{t-1} = \tanh\left(\mathbf{W}_d C_{t-1} + \mathbf{b}_d\right)$ (5.3.11) $\hat{C}^S_{t-1} = C^S_{t-1} * g\left(\Delta_t\right)$ (5.3.12) $C^*_{t-1} = C_{t-1} - C^S_{t-1} + \hat{C}^S_{t-1}$ (5.3.13) $f_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)$ (5.3.14) $i_t = \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)$ (5.3.15) $o_t = \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)$ (5.3.16) $\tilde{C} = \tanh\left(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right)$ (5.3.17) $C_t = f_t * C^*_{t-1} + i_t * \tilde{C}$ (5.3.18) $\mathbf{h}_t = o * \tanh\left(C_t\right)$ (5.3.19)
T-LSTM Variant [124]	$f_t = \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right)$ (5.3.20) $i_t = \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)$ (5.3.21) $o_t = \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)$ (5.3.22) $g_t = \tanh\left(\mathbf{W}_g\mathbf{x}_t + \mathbf{U}_g\mathbf{h}_{t-1} + \mathbf{b}_g\right)$ (5.3.23) $c^{short}_{t-1} = \tanh\left(\mathbf{W}_{short_c} c_{t-1} + w_{short_t}\Delta t + \mathbf{b}_{short_c}\right)$ (5.3.24) $c^{long}_{t-1} = \tanh\left(\mathbf{W}_{long_c} c_{t-1} + w_{long_t}\Delta t + \mathbf{b}_{long_c}\right)$ (5.3.25) $c^{short_{new}}_{t-1} = \tanh\left(w_{shrink}\Delta t + \mathbf{b}_{shrink}\right) c^{short}_{t-1}$ (5.3.26) $c^{new}_{t-1} = \tanh\left(w^{new}_{short} c^{short_{new}}_{t-1} + w^{new}_{long} c^{long}_{t-1} + \mathbf{b}_{merge}\right)$ (5.3.27) $c_t = f_t * c_{t-1} + i_t * g_t$ (5.3.28) $h_t = o_t * \tanh\left(c_t\right)$ (5.3.29)

the memory due to recurrent hidden states through time. The hidden state of the current time step of standard RNN model is computed as follows. Note that bias terms are omitted for simplicity.

$$\begin{aligned} \mathbf{h}_t &= \sigma\left(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1}\right) \\ \mathbf{y}_t &= f\left(\mathbf{V}\mathbf{h}_t\right) \end{aligned} \tag{5.3.30}$$

where $\mathbf{W} \in \mathbb{R}^{r \times d}, \mathbf{U} \in \mathbb{R}^{r \times r}$, and $\mathbf{V} \in \mathbb{R}^{r \times c}$ are the network weights, and d, r and c are input, hidden, and output dimensions, respectively. The non-linear activation function, $f\left(\cdot\right)$, is chosen based on the task (e.g., sigmoid for classification, linear for regression tasks). This architecture mainly focuses on the short term changes since the output at any time step is obtained using the hidden state of the previous time step. On the other hand, learning to store the information about long sequences is not feasible with the current architecture due to the decaying error flow back through time.

LSTM network alleviates the training for long sequences by introducing an extra memory component, called cell memory. Cell memory is designed to behave as a long term memory, while the hidden state captures the short term changes. Mathematical model of the standard LSTM can be found in Table 5.1. Each equation from 5.3.1 to 5.3.4 comprises an input and a recurrent layer with a non-linear activation function similar to the vanilla RNN in Eq. 5.3.30. The hidden state at each time step, in Eq. 5.3.6, is directly mapped from the updated cell memory controlled by the output gate. Thus, long and short term memory components are tightly coupled in LSTM networks. Cell memory is updated using a weighted sum of the candidate and previous memories as given in Eq. 5.3.5. The weights, f_t and i_t in Eq. 5.3.5, are named forget and input gates. The purpose of the multiplicative input gate is to protect the memory content stored in the previous time step from perturbations [58]. The additive component in Eq. 5.3.4 prevents the gradient values from vanishing due to Constant Error Carousel (CEC). CEC occurs when the values of forget gate and input gate are nearly 1, in other words, information flows to the next time step unchanged. In this case, the derivative does not have the severe decaying effect when the error back-propagates. It is

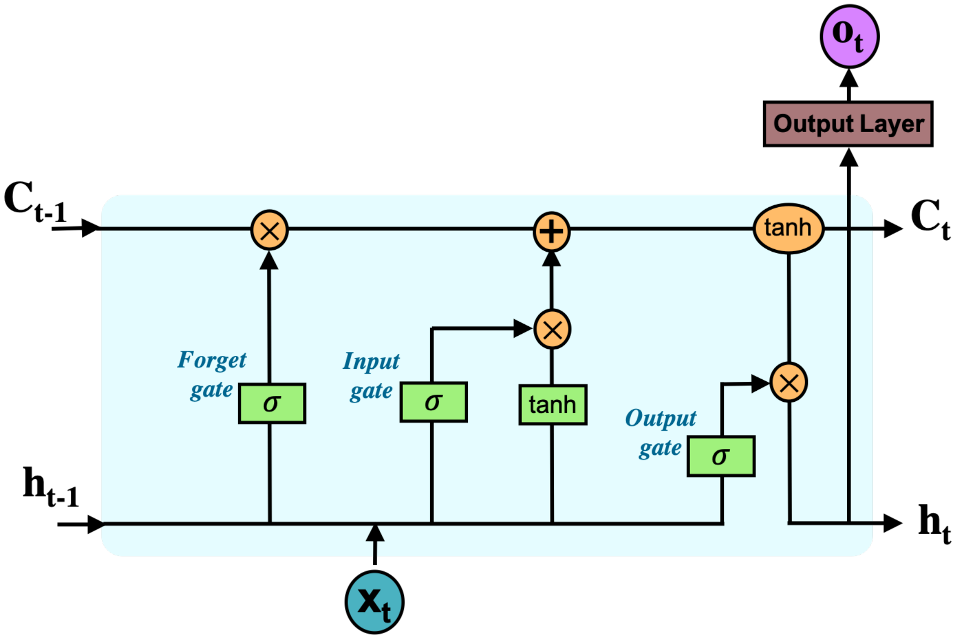


Figure 5.3 Standard LSTM unit.

also due to the fact that the expression in Eq. 5.3.4 does not have a non-linear activation function with flat regions.

In a standard LSTM unit, shown in Figure 5.3, the three gates, forget, input, and output, regulate the information flow to the next time steps. On the other hand, the purpose of these gates can also be considered as feature extraction since each gate has an input layer. However, the value of each gate is computed using the same formulation. The only difference is that the gate weights are not shared. Even though, different set of weights are learned for each gate, the contribution of each individual gate to the predictive power is not always so significant. For instance, in literature, it is often discussed that the output gate may not be necessary [78]. Similarly, some studies propose to use a single forget gate and derive the input gate as $1 - f$. For multivariate time series, using all three gates can provide the complexity required to encode the underlying patterns in features and the temporal sequence. Otherwise, LSTM architecture is prone to overfitting due to the high number of parameters.

GRU is proposed to alleviate the overfitting in LSTM by eliminating one of the gates. Two gates given in Eq. 5.3.7 and Eq. 5.3.8 are called update and reset gates, respectively. The update gate learns how much information contained in the hidden state needs to be discarded. In that sense, the update gate behaves as the forget gate in LSTM. Similarly, the reset gate determines how much information from the previous hidden state is added to the memory of the current time step. Thus, reset gate corresponds to the input gate in LSTM. In conclusion, the fundamental difference between LSTM and GRU is the number of parameters, otherwise memory is computed with a similar approach. In literature, performance of LSTM is often reported better than GRU. GRU is mostly preferred when there is not enough data. In addition to LSTM and GRU, there are various other gated units [47, 127] aiming to reduce the number of parameters and alleviate the training. On the other hand, memory is commonly coupled in the aforementioned models and none of the techniques report significant performance improvement over LSTM.

RNN and its variants originally assume a fixed sampling rate, namely regular elapsed time. As discussed in Chapter 4, this assumption may not be reasonable for real world problems, especially for healthcare applications. The proposed T-LSTM [11] architecture attempts to decompose the cell memory into short and long-term memories. The main motivation behind T-LSTM is to incorporate elapsed time in a way that the contribution of the information of the previous state is discounted if there is a large time gap. Memory decomposition aims to ensure that the important memory content is not altered while applying the time decay. Recently, a modified T-LSTM [124] is proposed, where the cell memory is decomposed using two different $\tanh$ layers. Both approaches attempt to modify the cell memory while the original gates of the LSTM unit are preserved. In the following section, we discuss a different approach, named external memory networks, to utilize memory in dynamic systems. External memory networks combines a standard neural network (e.g. LSTM) with an additional external memory.

5.3.2 External Memory Architectures

The goal of the RNN architectures is essentially memorizing the temporal pattern of the input time sequence. For this purpose, information observed at each time step is encoded into a hidden state (e.g., hidden state and cell memory for LSTM). Hidden state is a dense vector, thus the temporal knowledge is condensed in a single vector. As a result, the capacity of the memory maintained by recurrent networks is considered limited [52]. To address the limited memory capacity of RNN and its variants, deep networks with external memories are proposed in literature. Due to the increased memory capacity, the external memory models are suitable to model very long sequences. This family of networks commonly implements an external memory matrix with a read/write mechanism using attention. Namely, the weighted sum of different memory locations is used, where the weights are computed based on the input content.

5.3.2.1 Neural Turing Machine

One of the popular deep architectures with external memory is Neural Turing Machine (NTM) [52]. NTM, shown in Figure 5.4, comprises of a memory matrix, a controller, and a selective mechanism to read and write the memory. Controller behaves as an interface between the memory and the input, and interacts with the memory through an addressing mechanism. The attention weights used in reading and writing are learned using the addressing mechanism. Two of the addressing approaches discussed by Graves *et al.* [52] are accessing by content similarity and location. In the content-based addressing, attention weights are determined based on the similarity between the content and the memory locations. Whereas in the location-based addressing, controller decides which previous memory values to keep and discard. Writing mechanism comprises erase and add steps similar to the forget and input gates in LSTM, respectively. On the other hand, reading mechanism computes a weighted sum of the memory locations. As opposed to standard Turing machine, where a single memory location is accessed, reading the memory as a weighted sum of the memory locations enables a differentiable model. As a result, iterative gradient descent based optimization techniques can be used to learn the NTM parameters.

5.3.2.2 Memory Networks

Another common external memory approach is the memory network proposed by Weston *et al.* [118]. Memory network, shown in Figure 5.5 contains a memory and 4 components, namely

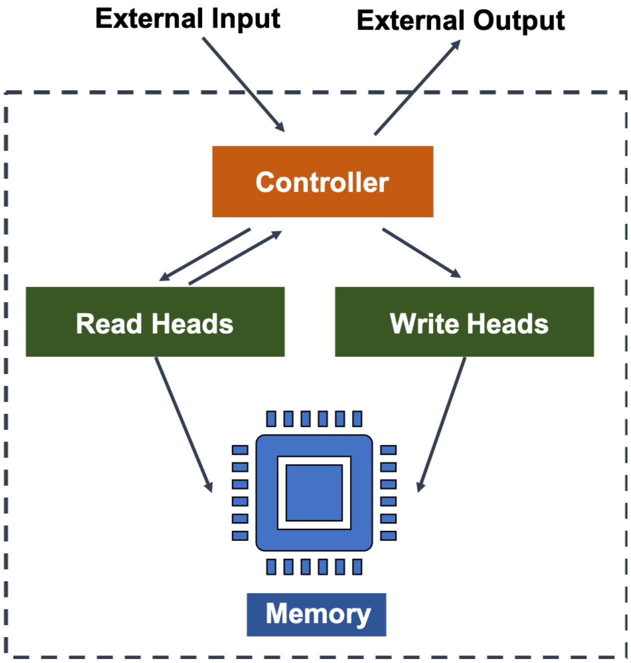


Figure 5.4 Neural Turing Machine [52]

input feature map, generalization, output feature map, and response. Input feature map projects the original input into a latent feature representation. Generalization component updates the memory units using the new input. Output feature map reads the most relevant memory locations guided by the current memory and the new input to produce the new output. Finally, response converts the output into a desired format depending on the task. The aforementioned components are usually learnable and a suitable machine learning model can be used to implement the procedure. However, this approach is fully supervised and needs to iterate the entire memory. End-to-end memory networks [108] offers less dependence on supervision by replacing soft attention mechanism with argmax. In other words, memory network computes the weighted sum of all the memory locations, whereas the end-to-end memory network retrieves most relevant memory locations. Memory networks have much simpler memory reading and writing mechanisms compared with NTM. In addition, external memory in memory network architectures mainly focuses on retrieving the relevant information from the memory rather than updating it.

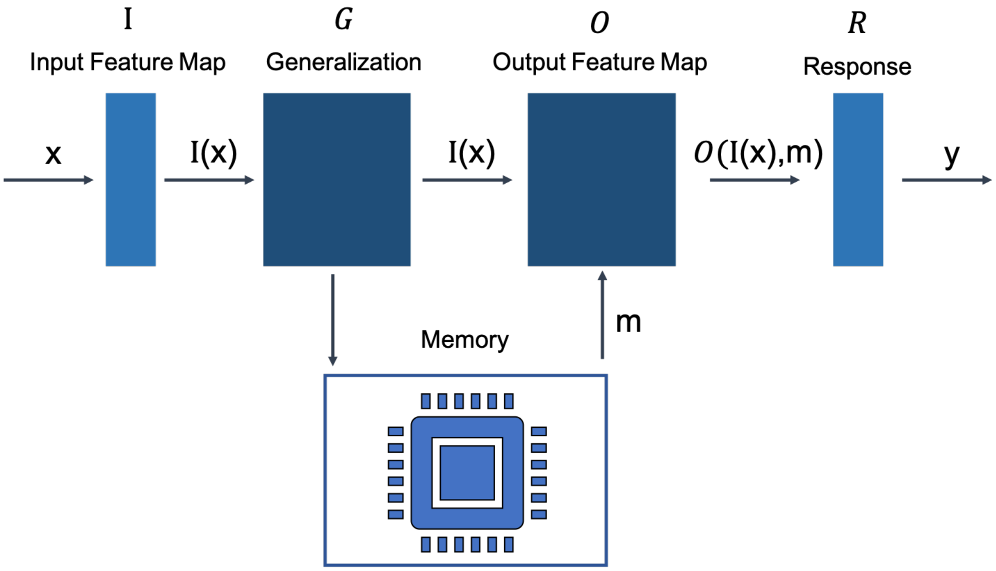


Figure 5.5 Memory Networks [118]

5.3.2.3 Disadvantages of External Memory

Due to increased memory capacity, NTM and memory networks perform better than LSTM for long sequences (e.g., > 200) [52, 118]. However, performance analysis of the aforementioned external memory models is usually discussed for straightforward tasks and limited domains. For instance, original NTM study [52] reports better performance than LSTM for abstract tasks, such as priority sort, where the network sorts a sequence of binary vectors with a scalar priority weight. On the other hand, the memory network [118] and the end-to-end memory network [108] are evaluated on textual reasoning tasks, such as question and answering. Memory networks yield slightly better performance compared with LSTM and NTM on language modeling tasks. Moreover, the aforementioned memory networks incorporate feedforward, and recurrent networks (e.g., RNN and LSTM) architectures to control read and write mechanisms. For this reason, combining standard deep models with an external memory increases the model complexity. Increased model complexity would be relevant for certain tasks, but it degrades the generalizability of the external memory models to wide variety of tasks with different levels of difficulty. Disadvantages of memory networks and NTM are summarized below.

- The number of learnable parameters are comparatively higher than RNN architectures.

- They are hard to parallelize due to the sequential memory access and update mechanisms.
- Training is difficult due to the fact that it can be impractical to employ an external memory.
- Numerical instability is more prominent than RNN architectures.

5.3.3 Decoupled Memory Gated Recurrent Network (DM-GRN)

DM-GRN is proposed to address two potential problems in T-LSTM model. When the elapsed time information is not available, T-LSTM assumes uniform elapsed time and behaves like a standard LSTM network. Even though the actual elapsed time is not known, irregularities due to different sampling rates and noise may still exist in the time series. The second potential issue is due to the coupled memory discussed in the previous sections. Since T-LSTM model follows the same procedure as LSTM after the cell memory modification, long and short term memories are tightly coupled. Although the performance of LSTM and its variants is often superior to vanilla RNN model, the coupled memory may become a limitation while analyzing time series with different sampling rates and frequencies.

Long term memory is usually expected to store the global temporal pattern, while the purpose of a short term memory is to capture local changes in the time series signal. LSTM's cell memory is considered as a long term memory, however as can be seen in Eq. 5.3.4, the cell memory update contains a recurrent layer of only the last time step. Therefore, the distinction between the long and short term memories is not clear. To address the aforementioned issues, we propose DM-GRN, a recurrent neural model with internal attention mechanism. The main hypothesis of the proposed approach is that the decoupled short and long term memories focus on different components of the input time sequences. Since the purpose is not a frequency domain analysis, short and long term memories are not expected to explicitly capture the high and low frequencies in the time series. For this reason, hypothesis does not assume a perfect distinction between short and long term memories in terms of frequencies. On the other hand, long term memory is assumed to capture the overall trend and short term memory should be more receptive to rapid changes in the time sequence. The proposed approach, whose mathematical model is given below, fuses the weighted long and short term memories to obtain a total memory. The total memory is later used to predict the future values of the time sequence using a task-specific output layer.

$$\text{Forget gate}\quad f_t = \sigma\left(\mathbf{W}_f \Delta\mathbf{x}_t + \mathbf{U}_f \mathbf{M}_{t-1} + \mathbf{b}_f\right) \tag{5.3.31}$$

$$\text{Input gate}\quad i_t = \sigma\left(\mathbf{W}_i \mathbf{x}_t + \mathbf{b}_i\right) \tag{5.3.32}$$

$$\text{Short term memory}\quad \mathbf{M}_t^{short} = \tanh\left(\mathbf{W}_{short} x_t + \mathbf{U}_{short} \mathbf{M}_{t-1} + \mathbf{b}_{short}\right) \tag{5.3.33}$$

$$\text{Long term summary}\quad \mathbf{M}_{t-1}^{*} = \sum_{\ell=1}^{L} w_\ell \mathbf{M}_{t-\ell} \tag{5.3.34}$$

$$\text{Long term memory}\quad \mathbf{M}_t^{long} = \tanh\left(\mathbf{W}_{long} x_t + \mathbf{U}_{long} \mathbf{M}_{t-1}^{*} + \mathbf{b}_{long}\right) \tag{5.3.35}$$

$$\text{Total memory}\quad \mathbf{M}_t = f_t * \mathbf{M}_t^{long} + i_t * \mathbf{M}_t^{short} \tag{5.3.36}$$

where $\{\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_{short}, \mathbf{W}_{long}\}$ are $d_x \times d_h$ dimensional input layer weights, $\{\mathbf{U}_f, \mathbf{U}_{short}, \mathbf{U}_{long}\}$ are $d_h \times d_h$ dimensional recurrent layer weights, $\mathbf{M}_t \in \mathbb{R}^{d_h}$ is the total memory at time t, $\Delta\mathbf{x}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})$, and $\{w_\ell\}_{\ell=1}^{L}$ are the attention weights computed as below:

$$w_\ell = \text{softmax}\left(\mathbf{v}^T \tanh\left(\mathbf{W}_x \mathbf{x}_t + \mathbf{W}_h \mathbf{M}_{t-1}^{long}\right)\right) \tag{5.3.37}$$

where $\text{softmax}\,(y_i) = \frac{e^{y_i}}{\sum_j y_j}$ and $\{\mathbf{v} \in \mathbb{R}^{d_h}, \mathbf{W}_x \in \mathbb{R}^{d_x \times d_h}, \mathbf{W}_h \in \mathbb{R}^{d_h \times d_h}\}$ are softmax layer weights.

Due to the recurrent state transitions, the current memory is considered as a function of the previous time steps, $\mathbf{M}_t = f\left(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t\right)$. On the other hand, the contribution of each time step's input is not equal. According to the theoretical analysis conducted by Le *et al.* [74], the contributions of the past inputs tend to gradually decay. For this reason, the effect of the past time steps on the current memory is challenging to leverage. For this reason, we propose to use a weighted sum of the past memory components while updating the long term memory. In Eq. 5.3.34, long term summary is defined as the weighted sum of L previous time steps, where L can be determined considering the sequence length and cross-validation. The differences between the proposed DM-GRN, visualized in Figure 5.6, and T-LSTM models are summarized below.

- GM-GRN discards the recurrent layer in the input gate. In this case, input layer only focuses on extracting features from the current input.
- In the forget gate, $\Delta \mathbf{x}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})$ is used as the input along with a recurrent previous memory layer. The purpose of utilizing $\Delta \mathbf{x}_t$ is to urge the forget gate to put more emphasis on the change in the signal since the last time step, rather than its instantaneous value. Thus, the elapsed time irregularity is implicitly incorporated in the model.
- As opposed to LSTM and T-LSTM, short and long term memories are decoupled in DM-GRN. Two separate $\tanh$ layers comprising of the current input and the previous memory are utilized to decouple the total memory. The two memory components are then weighted using the output of the input and forget gate filters and aggregated to obtain the updated total memory.
- DM-GRN eliminates the output gate based on the evidence in literature that the contribution of the output gate is not significant.
- An internal attention mechanism is adopted to leverage the information stored in the previous time steps. Since the information stored in the memory of past time steps may be discarded and overwritten, the effect of the input at previous time steps is not expected to be completely preserved. For this reason, attention is considered to increase the capacity of the long term memory component.

In the next section, the behavior of the proposed architecture for different tasks are investigated and the performances of baseline approaches and DM-GRN are compared.

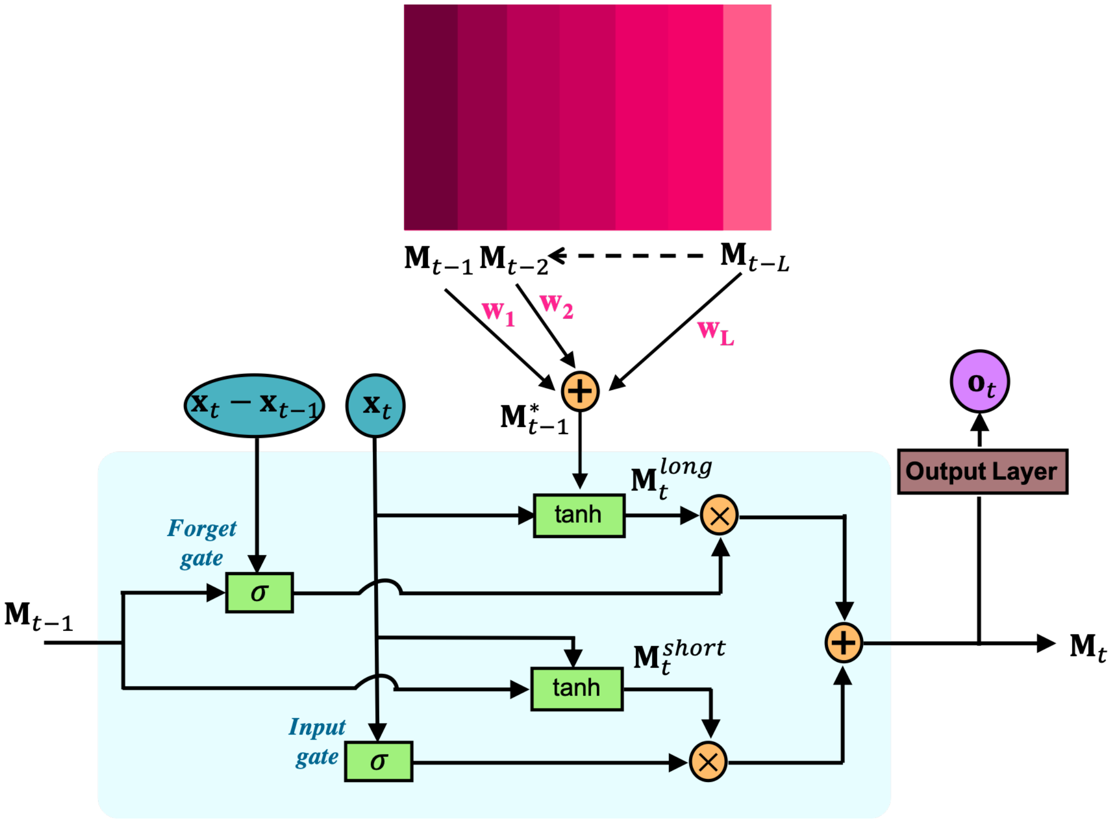


Figure 5.6 Proposed decoupled memory gated recurrent network (DM-GRN) unit.

5.4 Experiments

In this section, experimental results are presented and interpreted. The performance of the proposed model is investigated and compared to several other gated recurrent architectures. Synthetic data experiments are designed to analyze and visualize the behavior of the memory components. Real world data experiments are conducted to compare the predictive performance of the proposed model and baselines. To make a fair comparison, number of epochs, hidden dimensionality, and the data used in training and test phases were fixed the same for all the baselines and the proposed approach. Since a better performance is usually observed with 5 previous time steps, the parameter L in DM-GRN is fixed to 5 for all the experiments. All the models are implemented with Tensorflow and Python.

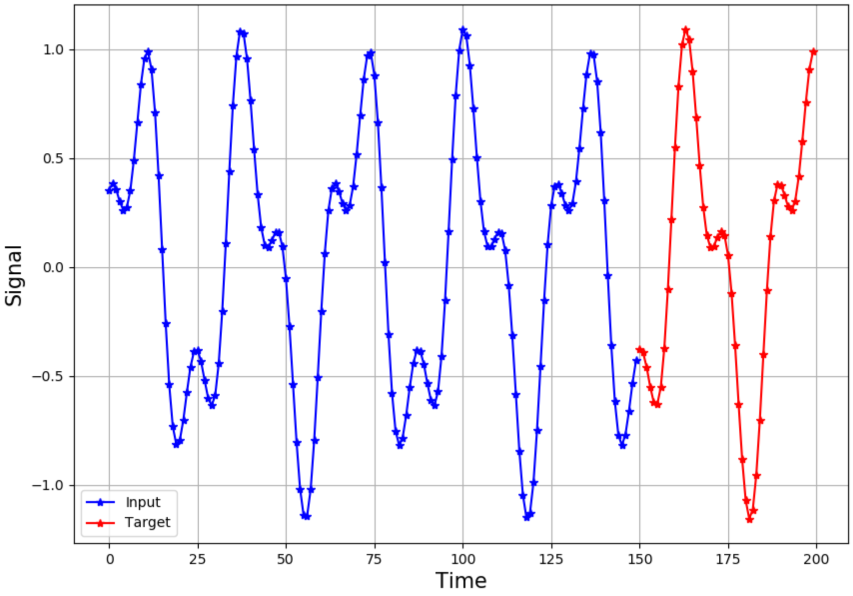


Figure 5.7 Synthetically generated input signal and its corresponding target.

5.4.1 Synthetic Data

Synthetic data experiments are conducted to investigate and visualize the behavior of the proposed model in comparison with baseline methods. The predictive power of the models are also tested on the synthetic EHR dataset [69] used in Chapter 4.

5.4.1.1 Periodic Signal

In this experiment, a periodic signal was created with short term changes. Total of $5,000$ data points with sequence length of 200 were generated from normal random distribution. To analyze the performance of the proposed approach, first 150 time steps are used as the input to the recurrent models and the last 50 time steps are predicted. Input and its corresponding target of one of the samples are shown in Figure 5.7. A single layer DM-GRN is trained with 32-dimensional hidden state and the prediction performance after 50 epochs is observed. Final prediction is computed using the total memory, however predictions using only the short term and the long term memory components are also generated to investigate the behavior of the decoupled memory.

Prediction results are visualized for two of the samples in the synthetic dataset in Figure 5.8. According to our observation, predicted values by the short term memory can follow the temporal changes, but do not fit the target perfectly. On the other hand, long term memory component demonstrates the overall trend in the sequence and complements the short term memory. Thus, the final prediction perfectly superimposes on the original target. This result is consistent with our initial hypothesis. Short term memory component is expected to capture the temporal changes between consecutive time steps while the long term memory component is expected to remember the past information and complement the short term memory.

In this problem, the elapsed time is uniform. For this reason, we also investigate the behavior of the cell memory in the standard LSTM architecture. The predictions obtained using the hidden state and the cell memory of LSTM unit are given in Figure 5.9. According to our observation, information contained in the cell memory does not directly reflect the predictive power. We also observe the memory activations for LSTM and DM-GRN to visualize the memory content of both architectures. In Figure 5.10, heatmap visualization of the short and long term memory components of DM-GRN and LSTM are given. In the figure, short and long term memory in LSTM correspond to hidden state and the cell memory, respectively. Periodic nature of the signal and the short term changes are more observable in DM-GRN memory components than LSTM hidden states and the cell memory. This result provides an evidence that the difference between long and short term memories in the proposed architecture is more observable and informative compared to LSTM.

5.4.1.2 Synthetic EHR

In this experiment, synthetically generated EHR dataset [1], used in Chapter 4, was utilized to evaluate the proposed approach for multivariate time series. We considered the same task as Chapter 4, which is to classify patients as diabetes and healthy. Diabetes Mellitus patients and patients without diabetes were sampled, resulting in $6,730$ subjects in total. Input features were 529 dimensional

[1] http://www.emrbots.org/

multi-hot vectors representing the common diagnoses in the dataset. Classification accuracy (ACC) and area under ROC curve (AUC) are reported for 5 different train and test data splits. Dimensionality of hidden states and the fully connected layer were set to $1,024$ and 64, respectively. Binary classification results of single layer DM-GRN, T-LSTM, LSTM, T-LSTM Variant [124], and GRU are compared for 100 epochs in Table 5.2. In this problem, elapsed time between consecutive records of patients is available, and it is not uniform as discussed in Chapter 4. T-LSTM and its variant [124] utilize the elapsed time to modify the LSTM memory while DM-GRN tackles the irregular elapsed time with a decoupled memory architecture.

Table 5.2 Diabetes classification experiment using synthetic EHR dataset. Average AUC and ACC are reported for 5 different train and test data splits.

Method	AUC (std)	ACC (std)
DM-GRN	**0.952** (0.03)	**0.932** (0.02)
T-LSTM	0.923 (0.02)	0.897 (0.02)
LSTM	0.911 (0.02)	0.878 (0.03)
GRU	0.895 (0.02)	0.866 (0.03)
T-LSTM Variant [124]	0.933 (0.02)	0.888 (0.03)

In this experiment, the proposed DM-GRN performs better than baseline gated architectures in terms of classification accuracy. T-LSTM and its variant also demonstrate higher accuracy than LSTM and GRU. This observation indicates that addressing the elapsed time irregularity along with memory modifications improves the predictive performance for the synthetic EHR dataset.

5.4.2 Traffic Speed Prediction

Application of the proposed approach is not limited to healthcare applications. Time series data with short and long-term changes can be encountered in many domains, such as intelligent transportation systems. Timely and accurate prediction of traffic speed and flow facilitates traffic management, travel scheduling and trip advisory systems. In particular, ride-sharing companies are interested in speed forecasting applications to avoid travel delays and improve their service quality during rush-hours. Traffic speed prediction aims to infer future traffic speed values on a road segment given its past traffic speed information. In this section, we used a real world traffic speed dataset provided by DIDI (Chinese ride-sharing company) collected in the city of Chengdu, China, shown in Figure 5.11, during November 2016 [34]. Raw data contains GPS files comprising of driver ID, order ID, time stamp, longitude, and latitude. Pre-processing steps, such as coordinate transformation, map matching, and trajectory to speed transformation, are required to be able to utilize the traffic data in speed forecasting tasks. In this study, we utilized pre-processed traffic data, which includes speed time series of the road segments in a sub-region, for instance the red bounding box in Figure 5.11. Some of the roads have insufficient amount of speed information due to missing values. Such roads are eliminated and a total of $3,432$ road segments are used in the experiments. Each road has a time sequence of 144 elements corresponding to speed values aggregated at every 10 mins during 24 hours for 30 days. The temporal change of the speed values in one of the roads during a week in November 2016 is given in Figure 5.12.

In this problem, we tackle a univariate time series analysis problem. As shown in Figure 5.12, approximately first 7 hours of the day demonstrates lower variance compared with the change in speed values during the rest of the day. However, this pattern is not assumed to be consistent across all the road segments in the sub-region. Due to missing values and noise, forecasting future speed values becomes a challenging task. Moreover, road segments do not usually follow a prominent pattern in this dataset. The main goal of the traffic prediction problem is to learn the temporal pattern of the target road segment so that the future speed predictions will be reliable. In addition, spatial dependencies between neighboring road segments may also effect the temporal trend in the speed time series. Traffic data can be considered as a road network, shown in Figure 5.11, where nodes are intersections and edges are the roads. As a result, graph analysis techniques are beneficial for detecting the spatial dependencies. On the other hand, connectivity of the nodes on the graph does not necessarily reflect the reality. Even though incoming and outgoing traffic flow are different, some nodes can seem connected on the graph. Such anomalies degrade the performance of spatial models. For this reason, only temporal dependencies are taken into account in this experiment.

Table 5.3 Mean absolute error (MAE) and Root Mean Square (RMSE) are reported for the traffic prediction task.

Method	MAE	RMSE
DM-GRN	1.064	1.465
T-LSTM	1.030	1.437
LSTM	1.021	1.425
GRU	0.966	1.347
T-LSTM Variant [124]	1.077	1.489

To evaluate the traffic prediction performance of the proposed model and the baselines, speed information over 10 minutes intervals is aggregated further to a sequence length of 24. Thus, each time step represents an hour of the day facilitating interpretation of the prediction results. Aggregated speed time series of one of the road segments is illustrated in Figure 5.13. Input and output time series are prepared in the following way; all the time sequences of a day in November expect the last one (e.g., there are 4 Sundays in November 2016) are concatenated to form the input time series and the remaining time series is used as the target to predict. As a result, 72 steps long input time series are used to predict 24 steps long target sequence. The task was posed as a regression problem, where the last hidden state is mapped to the 24-dimensional target. In this experiment, consecutive speed values of some road segments are missing. As a result, we still need to tackle irregular elapsed time even though the speed values are aggregated.

We used 80% of the road segments with all the available days in training and the rest of the road segments for test. Mean absolute error (MAE) and root mean square error (RMSE) are computed for each day, and then the average error is reported in Table 5.3. Traffic prediction is a challenging problem due to the fact that speed time series of the road segments are very irregular. For this reason, the error is usually high for all the methods compared in this study. LSTM and GRU performed slightly better than the proposed DM-GRN, whereas DM-GRN performs better than T-LSTM Variant. Prediction results can also been observed in Figure 5.14. The short and long term memory components of the proposed DM-GRN demonstrate a similar behavior as the univariate synthetic data experiment. In other words, the long term prediction follows the global trend of the time series while the short term component is more susceptible to the short term changes.

5.4.3 Diabetes Data

In this section, a binary classification experiment is conducted to evaluate the performance of the proposed approach for multi-variate time series data. The dataset, also used in Chapter 2, comprises encounter data (emergency, outpatient, inpatient), demographics, diagnoses, and in-hospital procedures (e.g., laboratory and pharmacy data) [107]. Detailed information about the dataset is provided in Table 5.4. In Chapter 2, the dataset was used for patient phenotyping task, which is an unsupervised problem aiming to group patient based on common diagnoses. For this reason, we had mainly focused on diagnoses and demographics information. However, in this experiment, the goal is to predict the readmission of a patient given his clinical information collected during each previous hospital admission the patient.

Table 5.4 Diabetes dataset statistics.

Time span	1999 - 2008
Total number of records	101,766
Total number of patients	71,518
Average number of admissions	14
Number of patient with more than 3 admissions	3,011
Total number of records of 3011 patients	16,169

Readmission prediction is a supervised problem, where the temporal patterns in medical history of patients play an important role. In the dataset, ground-truth readmission information is available for each hospital admission of a patient, such as readmission after more than 30 days, readmission after less than 30 days, and no readmission. In biomedical informatics, 30 days time window is usually preferred in readmission prediction problem since 30 days criteria is usually considered by the funding agencies [107]. Thus, the records of each patient are labeled as readmission if the patient readmitted before or after 30 days, and as no readmission if the there is no information about readmission. As a result, the problem becomes predicting the binary labels of each time step of a sequential multi-variate time series data.

The deep recurrent models discussed in this chapter are applicable to the problem described above. However, some patients have a very few hospital admissions in the dataset. In this case,

Table 5.5 Readmission prediction performance of diabetes dataset. Average accuracy (ACC) and area under ROC curve (AUC) are reported for 5 different train and test splits. Standard deviation is given inside the parentheses.

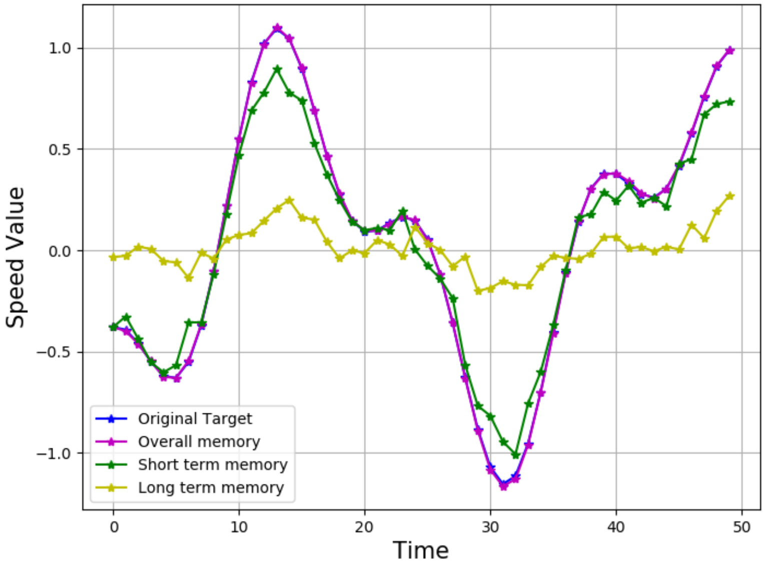
Method	ACC (std)	AUC (std)
DM-GRN	**0.941** (0.03)	**0.944** (0.03)
LSTM	0.891 (0.05)	0.915 (0.04)
GRU	0.906 (0.05)	0.934 (0.04)
T-LSTM Variant [124]	0.912 (0.01)	0.911 (0.02)

it is not reasonable to use a recurrent model since there will not be a prominent temporal pattern to capture. To avoid this situation, we eliminated patients with less than 3 hospital admissions, resulting in $16,169$ records and $3,011$ patients, in our experiments. In this dataset, patient records are ordered, however the actual time stamp of a record is not provided. As a result, the elapsed time between consecutive records is not known. As it was discussed in Chapter 4, consecutive records in EHR datasets often have irregular elapsed times. For this reason, we cannot simply assume uniform elapsed time in this dataset. T-LSTM could be utilized if the elapsed time was provided.

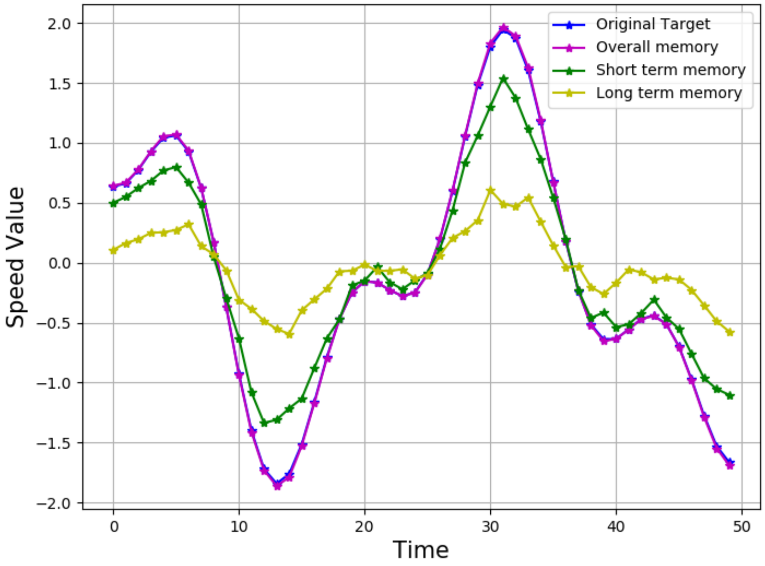
We used 70% of the patients in training and 30% for test. Each record is represented with a 85-dimensional feature vector and feature vectors are normalized to unit length. Each feature vector contains, time in hospital, number of diagnoses, and total of 25 medications related to diabetes and other diagnoses. Nominal features (e.g., medications) are represented with one-hot vectors. Each model is trained for 300 epochs with 512-dimensional hidden layers and one 128-dimensional fully connected layer. The readmission prediction performances of LSTM, GRU, T-LSTM variant and the proposed DM-GRN models are given in Table 5.5. The proposed approach, DM-GRN yields the best readmission prediction performance in terms of classification accuracy (ACC) and area under ROC curve (AUC). This result indicates that even though the elapsed time is not explicitly available, there are time irregularities in the dataset that the proposed model can address better than the baselines.

5.5 Summary

In this chapter, we investigate the role of memory in gated recurrent networks and propose a new model to address the tightly coupled memory in LSTM architectures. The proposed model, DM-GRN, decouples the memory into short and long-term memory components and eliminates redundant gates and layers present in the standard LSTM. Based on empirical evaluations, short and long term components are observed to capture different dynamics in the input time series signals. Even though the predictive power of the proposed approach is not always superior to standard LSTM for univariate time series data, the difference between its long and short term memories is more interpretable. Furthermore, it is often observed that memory decoupling in DM-GRN performs better than T-LSTM Variant. As a result, when the elapsed time is not explicitly provided, the proposed approach can offer a solution to capture different dynamics in the time series data.



(a) Sample 1



(b) Sample 2

Figure 5.8 Prediction performance of the synthetic data. When only short term memory is used to predict the target, prediction error is high, but the predicted values can follow the temporal changes. On the other hand, long term memory component demonstrates the overall trend and complements the short term memory. Thus, the final prediction perfectly superimposes on the original target.

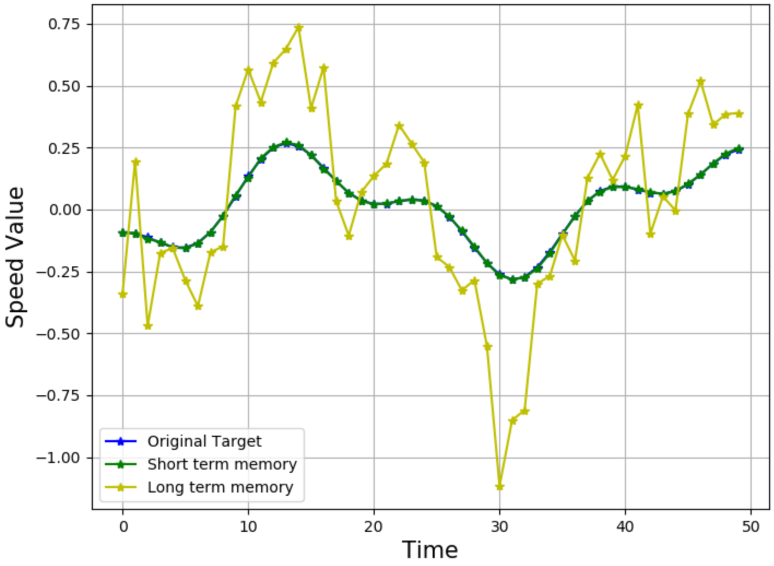
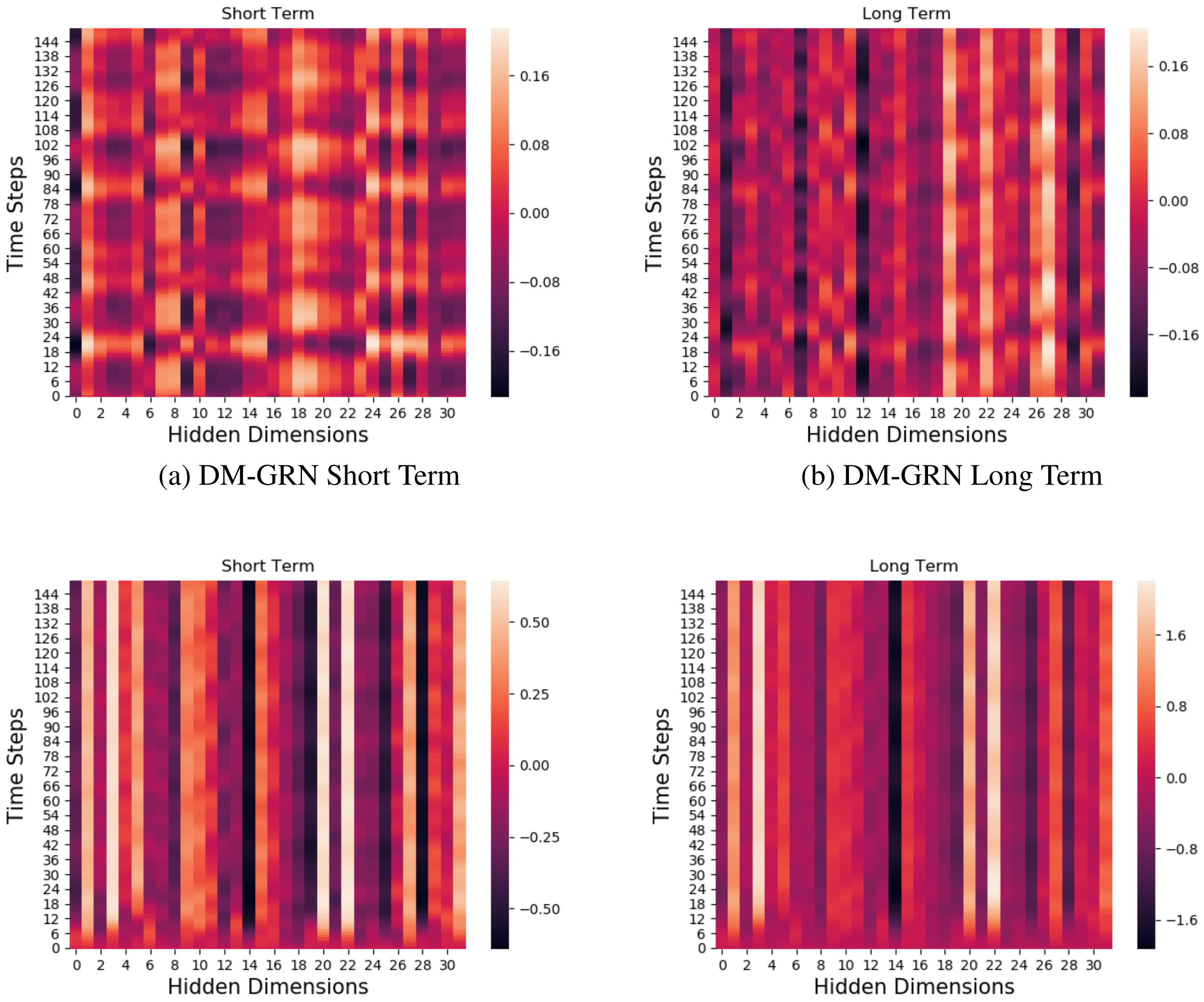


Figure 5.9 Predictions obtained using the hidden state and the cell memory of the standard LSTM unit. Cell memory is assumed to behave as a long term memory. According to our observation, information contained in the cell memory does not directly reflect the predictive power.



(a) DM-GRN Short Term

(b) DM-GRN Long Term

(c) LSTM Short Term

(d) LSTM Long Term

Figure 5.10 Long and short term memory contents of DM-GRN, and hidden state and cell memory contents of LSTM for the synthetic data. LSTM short term and long term memories correspond to hidden states and the cell memory, respectively. Heatmaps are used to visualize the memory activations. The periodic nature of the signal and the short term changes are more observable in DM-GRN memory components than LSTM hidden states and the cell memory.

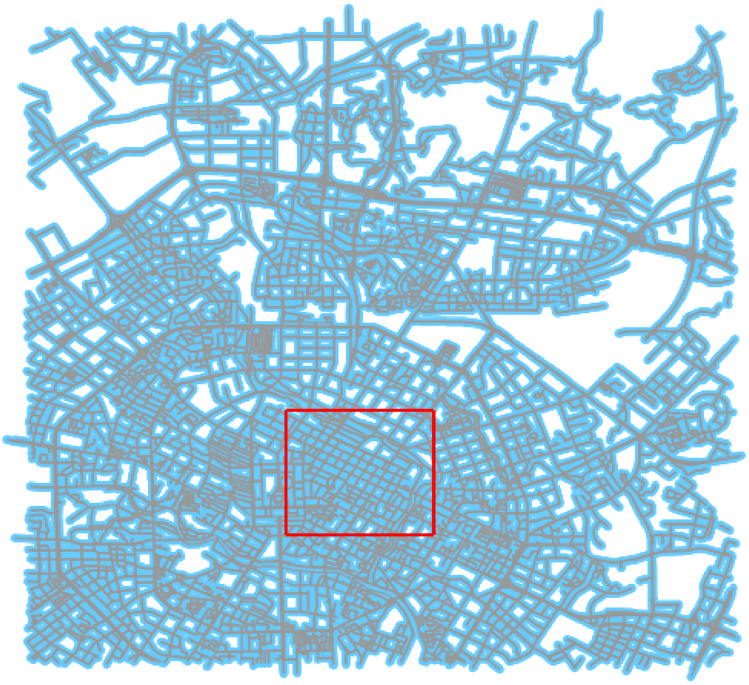


Figure 5.11 Road network of a subregion in the city of Chengdu, China. Intersections are represented with nodes, and the roads are represented by the edges. Due to noise and errors during map matching procedure, connected roads on the graph might not be connected in real life.

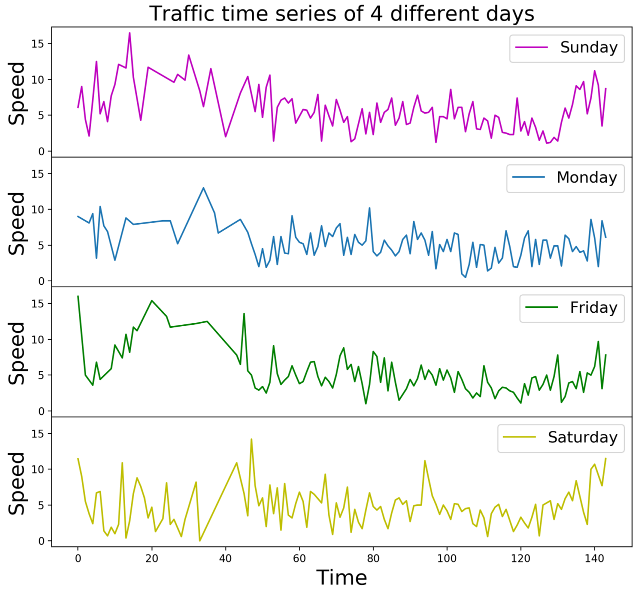


Figure 5.12 Speed time series of one road over 4 days of the same week in November, 2016. Traffic data suffers from missing values. Roads with relatively few missing values are kept and the missing values are interpolated.

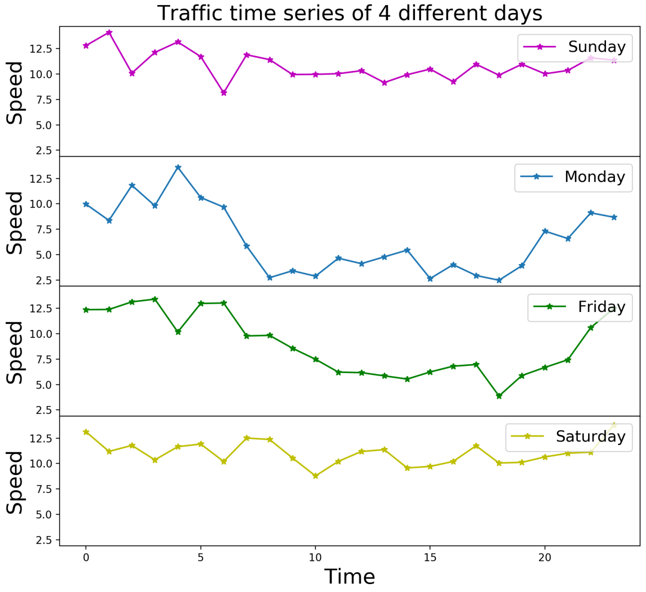
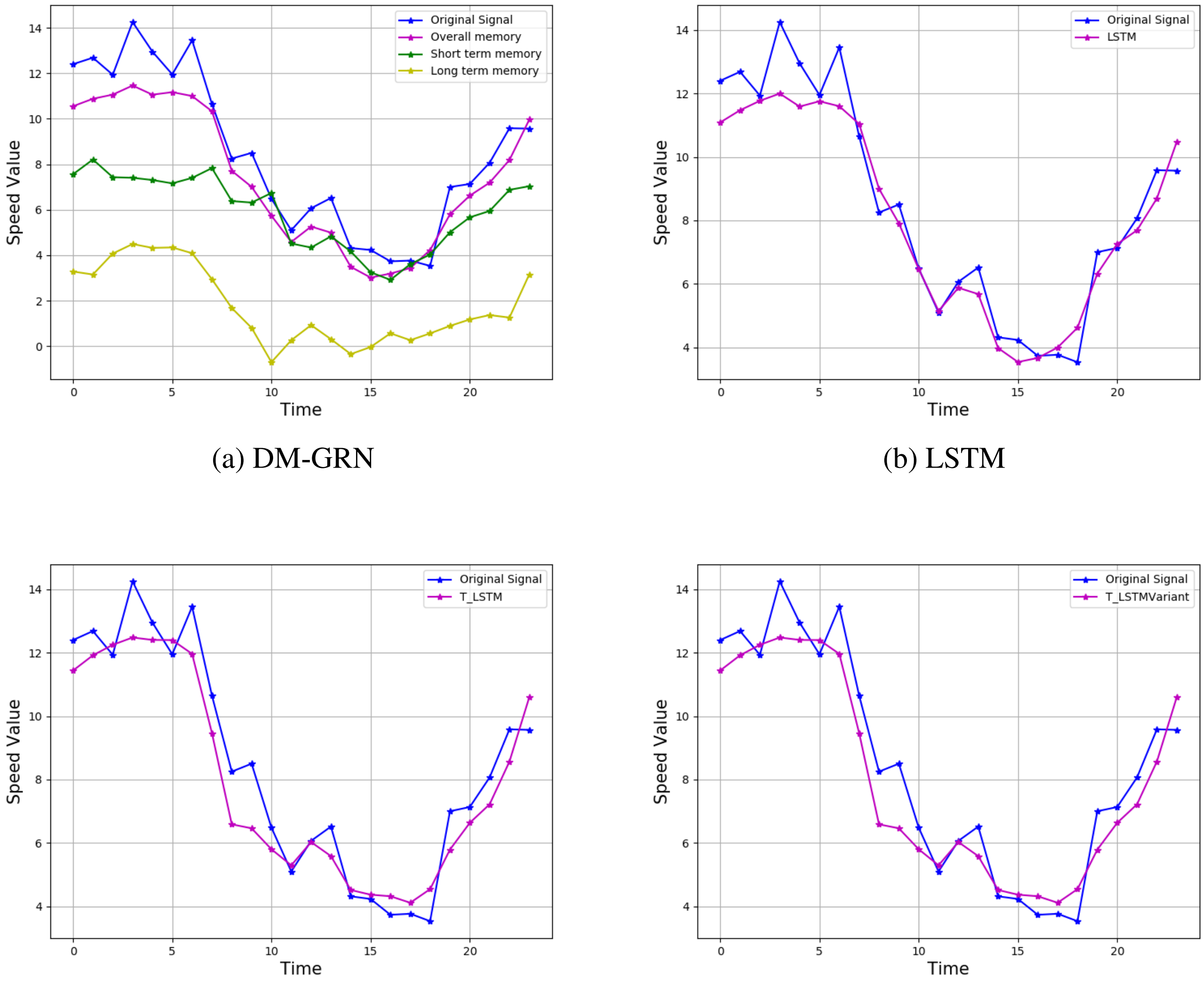


Figure 5.13 Speed time series for one road for 4 days in November, 2016. Each time step, in the 24 steps long sequence, represents the aggregated speed value of one hour.



(a) DM-GRN

(b) LSTM

(c) T-LSTM

(d) T-LSTM Variant

Figure 5.14 Traffic prediction performance. Long term memory of DM-GRN underestimates the actual speed values, however the long term prediction follows the global trend of the time series while the short term component is more susceptible to the short term changes.

Chapter 6

Summary and Suggestions for Future Work

6.1 Summary

In this thesis, several machine learning and deep learning models are developed to address different challenges encountered in biomedical informatics tasks. More specifically, temporal, distributed, large scale, and high dimensional nature of digital patient data (e.g., EHR) are taken into account. Contributions of this thesis are summarized below.

1. A convex SPCA approach along with stochastic gradient descent framework is introduced to obtain interpretable principal components. Proposed Cvx-SPCA method [9] is applied on patient phenotyping problem and a hierarchical visualization of the clinical phenotypes is presented.

2. An asynchronous distributed multi-task learning framework, AMTL [12], is proposed to address challenges due to distributed EHR data. AMTL transfers only the model vectors to alleviate privacy and bandwidth limitations. Proposed method can also be applied to a wide variety of supervised predictive tasks.

3. A new LSTM architecture, T-LSTM [11], is proposed for analysis of time series data with unevenly sampled time steps, such as EHRs. T-LSTM is designed based on the hypothesis

that longer the time gap, smaller the effect of the previous memory to the current output. T-LSTM auto-encoder is proposed to encode patient's longitudinal medical records in a single vector representation. Learned representations are used to cluster patients to assist in patient subtyping task.

4. An extension to T-LSTM is proposed to address short and long-term changes in time series data. The designed model, DM-GRN, comprises a gated recurrent network with decoupled memory. Decoupling the internal memory of the recurrent network facilitates more robust prediction for time series with irregularities even the actual elapsed time is not available.

6.2 Suggestions for Future Work

The algorithms and models designed in this thesis are not limited to healthcare applications. Distributed and temporal datasets are encountered in many other domains. In this section, potential future research directions for the proposed methods are discussed. The research presented in this thesis can be extended in the following directions.

- Stochastic sparse principal component analysis (SPCA) approach, proposed in Chapter 2, provides an interpretable dimensionality reduction and visualization tool. Due to its sparsity, SPCA automatically provides the information about which input features are more important to obtain the principal components. For this reason, one potential extension of the proposed SPCA approach can be integrating a feature selection scheme [79], which is a commonly used technique for gene expression analysis.

- The distributed asynchronous multi-task learning (AMTL) approach proposed in Chapter 3, offers a practical and robust method to develop predictive models for distributed datasets. The proposed AMTL model was posed as a distributed optimization problem with standard classification and regression objective function. This framework can be extended in a framework that can work with different input structures, such as graphs. The relatedness between distributed tasks can be defined with a weighted graph [116], and thus the proposed distributed MTL formulation can be modified to handle graph structured input. When the physical graph structure of the distributed network is consistent with the relationships between tasks, network communication cost can be further reduced by focusing only on the communication between neighboring task nodes.

- Healthcare, finance, and intelligent transportation systems are some of the fields where time series data is the main source of information. Time series analysis approaches proposed in Chapter 4 and 5 can be extended in different application domains. For instance, time series forecasting is one of the prominent methods to tackle traffic speed and flow estimation tasks in intelligent transportation systems. In addition to the temporal dependency in traffic variables, spatial relationships in a city graph may also effect the forecasting performance. For this reason, T-LSTM and DM-GRN, proposed in Chapter 4 and 5, respectively, can be extended to incorporate the spatial dependencies between data points.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Clinical phenotypes of copd: Identification, definition and implications for guidelines. *Archivos de Bronconeumologa (English Edition)*, 48(3):86–98, 2012.

[2] The exponential growth of data. https://insidebigdata.com/2017/02/16/the-exponential-growth-of-data/, 2017.

[3] Retailrocket recommender system dataset. https://www.kaggle.com/retailrocket/ecommerce-dataset, 2017.

[4] Unified new york city taxi and uber data. https://github.com/toddwschneider/nyc-taxi-data, 2017.

[5] Ratnadip Adhikari and R. K. Agrawal. *An Introductory Study on Time Series Modeling and Forecasting*. LAP LAMBERT Academic Publishing, 2013.

[6] Metin Akay, Dimitrios I. Fotiadis, Konstantina S. Nikita, and Robert W. Williams. Guest editorial: Biomedical informatics in clinical environments. *IEEE Journal of Biomedical and Health Informatics*, 19(1):149–150, 2015.

[7] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[8] Necdet Aybat, Zi Wang, and Garud Iyengar. An asynchronous distributed proximal gradient method for composite convex optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2454–2462, 2015.

[9] Inci M. Baytas, Kaixiang Lin, Fei Wang, Anil K. Jain, and Jiayu Zhou. Phenotree: Interactive visual analytics for hierarchical phenotyping from large-scale electronic health records. *IEEE Transactions on Multimedia*, 18(11):2257–2270, 2016.

[10] Inci M. Baytas, Kaixiang Lin, Fei Wang, Anil K. Jain, and Jiayu Zhou. Stochastic convex sparse principal component analysis. *EURASIP Journal on Bioinformatics and Systems Biology*, 2016(1):15, 2016.

[11] Inci M. Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K. Jain, and Jiayu Zhou. Patient subtyping via time-aware lstm networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 65–74, 08 2017.

[12] Inci M. Baytas, Ming Yan, Anil K. Jain, and Jiayu Zhou. Asynchronous multi-task learning. In *2016 IEEE 16th International Conference on Data Mining*, pages 11–20, 2016.

[13] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Science*, 2(1):183–202, 2009.

[14] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *arXiv:1206.5538v3[cs.LG]*, 2014.

[15] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[16] Elmer V. Bernstam, Jack W. Smith, and Todd R. Johnson. What is biomedical informatics? *Journal of Biomedical Informatics*, 43(1), 2010.

[17] Mike Bostock. Radial reingoldtilford tree. http://bl.ocks.org/mbostock/4063550, 2017.

[18] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[19] Matthew Brand. Fast online SVD revisions for lightweight recommender systems. In *Proceedings of SIAM International Conference on Data Mining*, pages 37–46, 2003.

[20] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[21] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

[22] Chao Che, Cao Xiao, Jian Liang, Bo Jin, Jiayu Zhou, and Fei Wang. An rnn architecture with dynamic temporal matching for personalized predictions of parkinson's disease. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017.

[23] Zhengping Che, David Kale, Wenzhe Li, Mohammad Taha Bahadori, and Yan Liu. Deep computational phenotyping. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 507–516, 2015.

[24] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *arXiv preprint arXiv:1606.01865*, 2016.

[25] Daqing Chen, Sai Laing Sain, and Kun Guo. Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining. *Journal of Database Marketing & Customer Strategy Management*, 19(3):197–208, 2012.

[26] Yun Kuen Cheung and Richard Cole. Amortized analysis on asynchronous gradient descent. *arXiv preprint arXiv:1412.0159*, 2014.

[27] Kyunghyun Cho, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078v3[cs.CL]*, 2014.

[28] Edward Choi, Mohammad Taha Bahadori, Andy Schuetzy, Walter F. Stewarty, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. *arXiv:1511.05942v11 [cs.LG]*, 2016.

[29] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[30] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3504–3512, 2016.

[31] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[32] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[33] Alexandre d'aspremont, Laurent E. Ghaoui, Michael I. Jordan, and Gert R. Lanckriet. A direct formulation for sparse pca using semidefinite programming. In *Advances in Neural Information Processing Systems 17*, pages 41–48. MIT Press, 2005.

[34] DIDI. Gaia open dataset. https://outreach.didichuxing.com/research/opendata/en/, 2018.

[35] Ivo D. Dinov, Ben Heavner, Ming Tang, Gustavo Glusman, Kyle Chard, Mike Darcy, Ravi K. Madduri, Judy Pa, Cathie Spino, Carl Kesselman, Ian T. Foster, Eric W. Deutsch, Nathan D. Price, John Darrell Van Horn, Joseph Ames, Kristi Clark, Leroy Hood, B. M. Hampstead, William T. Dauer, and Arthur W. Toga. Predictive big data analytics: A study of parkinsons disease using large, complex, heterogeneous, incongruent, multi-source and incomplete observations. In *PloS one*, 2016.

[36] Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. Client server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290–303, 2011.

[37] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *arXiv:1411.4389v4[cs.CV]*, 2016.

[38] Brent Edmunds, Zhimin Peng, and Wotao Yin. Tmac: A toolbox of modern async-parallel, coordinate, splitting, and stochastic methods. *arXiv preprint arXiv:1606.04551*, 2016.

[39] Cristobal Esteban, Oliver Staeck, Yinchong Yang, and Volker Tresp. Predicting clinical events by combining static and dynamic information using recurrent neural networks. *arXiv:1602.02685v1 [cs.LG]*, 2016.

[40] evariant. What is healthcare big data? https://www.evariant.com/faq/what-is-healthcare-big-data, 2018.

[41] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

[42] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American Control Conference*, volume 6, pages 4734–4739, 2001.

[43] Seyed-Mohammad Fereshtehnejad, Silvia Ros-Romenets, Julius B. M. Anang, and Ronald B. Postuma. New clinical subtypes of parkinson disease and their longitudinal progression: A prospective cohort comparison with other phenotypes. *JAMA Neurology*, 72(8):863–873, 2015.

[44] Centers for Medicare and Medicaid Services. Icd-9-cm diagnosis and procedure codes: Abbreviated and full code titles. https://www.cms.gov/Medicare/Coding/ICD9ProviderDiagnosticCodes/codes.html, 2014.

[45] Norbert Funke. Improving healthcare data management with emc isilon? think holistic, not in separated storage islands. https://blog.dellemc.com/en-us/improving-healthcare-data-management-with-emc-isilon-think-holistic-not-in-separated-storage-islands/, 2016.

[46] Mengxuan Gao, Hideyoshi Igata, Aoi Takeuchi, Kaoru Sato, and Yuji Ikegaya. Machine learning-based prediction of adverse drug effects: An example of seizure-inducing compounds. *Journal of Pharmacological Sciences*, 133(2):70–78, 2017.

[47] Yuan Gao and Dorota Glowacka. Deep gate recurrent neural network. In *JMLR: Workshop and Conference Proceedings*, pages 350–365. ACML, 2016.

[48] Dan Garber and Elad Hazan. Fast and simple pca via convex optimization. *arXiv:1509.05647v4 [math.OC]*, 2015.

[49] Luke B. Godfrey and Michael S. Gashler. Neural decomposition of time-series data for effective generalization. *CoRR*, abs/1705.09137, 2017.

[50] David Gotz, Fei Wang, and Adam Perer. A methodology for interactive mining and visual analysis of clinical event patterns using electronic health record data. *Journal of Biomedical Informatics*, 48:148–159, 2014.

[51] Alex Graves, Abdel rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. *arXiv:1303.5778[cs.NE]*, 2013.

[52] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[53] Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1242–1250, Jun 2014.

[54] Kristiina Häyrinen, Kaija Saranto, and Pirkko Nykänen. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *International Journal of Medical Informatics*, 77(5):291–304, 2008.

[55] Matthias Hein and Thomas Bühler. An inverse power method for nonlinear eigenproblems with applications in 1-spectral clustering and sparse pca. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 847–855. Curran Associates, Inc., 2010.

[56] Jette Henderson, Joyce C. Ho, Abel N. Kho, Joshua C. Denny, Bradley A. Malin, Jimeng Sun, and Joydeep Ghosh. Granite: Diversified, sparse tensor factorization for electronic health record-based phenotyping. In *2017 IEEE International Conference on Healthcare Informatics*, pages 214–223, 2017.

[57] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. Marble: High-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 115–124, 2014.

[58] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[59] Chih-Wei Huang, Richard Lu, Usman Iqbal, Shen-Hsien Lin, Phung Anh Nguyen, Hsuan-Chia Yang, Chun-Fu Wang, Jianping Li, Kwan-Liu Ma, Yu-Chuan Li, and Wen-Shan Jian. A richly interactive exploratory data analysis and visualization tool using electronic medical records. *BMC Medical Informatics and Decision Making*, 15(1), 2015.

[60] Wei Huang, Shuru Zeng, Min Wan, and Guang Chen. Medical media analytics via ranking and big learning: A multi-modality image-based disease severity prediction study. *Neurocomputing*, 204:125–134, 2016.

[61] Trevor Hastie Hui Zou and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

[62] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, Mar 2019.

[63] Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *52nd IEEE Conference on Decision and Control*, pages 3671–3676, 2013.

[64] Shuiwang Ji and Jieping Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 457–464, 2009.

[65] Xi Jin, Ping Luo, Fuzhen Zhuang, Jia He, and He Qing. Collaborating between local and global learning for distributed online multiple tasks. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 113–122, 2015.

[66] Alistair EW Johnson, Tom J Pollard, Lu Shen, Li-wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. Mimic-iii, a freely accessible critical care database. *Scientific Data*, 3, 2016.

[67] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems 26*, pages 315–323, 2013.

[68] Michel Journee, Yurii Nesterov, Peter Richtarik, and Rodolphe Sepulchre. Generalized power method for sparse principal component analysis. *Journal of Machine Learning Research*, 11(3):517–553, 2010.

[69] Uri Kartoun. A methodology to generate virtual patient repositories. *arXiv:1608.00570 [cs.CY]*, 2016.

[70] Jessica Kent. Big data to see explosive growth, challenging healthcare organizations. https://healthitanalytics.com/news/big-data-to-see-explosive-growth-challenging-healthcare-organizations, 2018.

[71] Marius Kloft, Ulf Brefeld, Pavel Laskov, Klaus-Robert Müller, Alexander Zien, and Sören Sonnenburg. Efficient and accurate lp-norm multiple kernel learning. *Advances in Neural Information Processing Systems*, pages 997–1005, 2009.

[72] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[73] Richard C Larson and Amedeo R Odoni. *Urban Operations Research*. Prentice Hall PTR, 1981.

[74] Hung Le, Truyen Tran, and Svetha Venkatesh. Learning to remember more with less memorization. In *International Conference on Learning Representations*, 2019.

[75] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[76] Jason D Lee, Yuekai Sun, and Michael A Saunders. Proximal newton-type methods for minimizing composite functions. *SIAM Journal on Optimization*, 24(3):1420–1443, 2014.

[77] Jorge Nocedal Leon Bottou, Frank E. Curtis. Optimization methods for large-scale machine learning. *CoRR*, abs/1606.04838, 2017.

[78] Omer Levy, Kenton Lee, Nicholas FitzGerald, and Luke Zettlemoyer. Long short-term memory as a dynamically computed element-wise weighted sum. *CoRR*, abs/1805.03716, 2018.

[79] Yi Li and Zhenyu He. Robust principal component analysis via feature self-representation. In *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, pages 94–99, Dec 2017.

[80] Zachary C. Lipton, David C. Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv:1511.03677v6 [cs.LG]*, 2016.

[81] Ji Liu and Stephen J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal of Optimization*, 25(1):351–376, 2015.

[82] Yves A Lussier and Yang Liu. Computational approaches to phenotyping: high-throughput phenomics. *Proceedings of the American Thoracic Society*, 4(1):18–25, 2007.

[83] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[84] Benjamin M. Marlin, David C. Kale, Robinder G. Khemani, and Randall C. Wetzel. Unsupervised pattern discovery in electronic health care data using probabilistic clustering models. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, pages 389–398. ACM, 2012.

[85] David Mateos-Nunez and Jorge Cortes. Distributed optimization for multi-task learning via nuclear-norm approximation. *IFAC Conference Paper Archieve*, 48(22):64–69, 2015.

[86] Nikhil Naikal, Allen Y. Yang, and Sastry. S. Shankar. Informative feature selection for object recognition via sparse pca. In *2011 International Conference on Computer Vision*, pages 818–825, 2011.

[87] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[88] Atsushi Nitanda. Stochastic proximal gradient descent with acceleration techniques. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1574–1582. Curran Associates, Inc., 2014.

[89] Desmond L. Nuttall, Harvey Goldstein, Robert Prosser, and Jon Rasbash. Differential school effectiveness. *International Journal of Educational Research*, 13(7):769–776, 1989.

[90] Office of the National Coordinator for Health Information Technology. Office-based physician electronic health record adoption. https://dashboard.healthit.gov/quickstats/pages/physician-ehr-adoption-trends.php, 2016.

[91] Pariwat Ongsulee. Artificial intelligence, machine learning and deep learning. In *15th International Conference on ICT and Knowledge Engineering*, 2017.

[92] Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine, Series 6*, 2(11):559–572, 1901.

[93] Zhimin Peng, Tianyu Wu, Yangyang Xu, Ming Yan, and Wotao Yin. Coordinate-friendly structures, algorithms and applications. *Annals of Mathematical Sciences and Applications*, 1:57–119, 2016.

[94] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. ARock: An algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[95] Adam Perer, Fei Wang, and Jianying Hu. Mining and exploring care pathways from electronic medical records with visual analytics. *Journal of Biomedical Informatics*, 56:369–378, 2015.

[96] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Vankatesh. Deepcare: A deep dynamic memory model for predictive medicine. *arxiv:1602.00357v1 [stat.ML]*, 2016.

[97] Carmen C. Y. Poon, May D. Wang, Paolo Bonato, and David A. Fenstermacher. Editorial: Special issue on health informatics and personalized medicine. *IEEE Transactions on Biomedical Engineering*, 60(1):143–146, 2013.

[98] M. Saadeq Rafieee and Ali Akbar Khazaei. A novel model characteristics for noise-robust automatic speech recognition based on hmm. In *2010 IEEE International Conference on Wireless Communications, Networking and Information Security*, pages 215–218, 2010.

[99] Dharavath Ramesh, Pranshu Suraj, and Lokendra Saini. Big data analytics in healthcare: A survey approach. In *2016 International Conference on Microelectronics, Computing and Communications*, pages 1–6, 2016.

[100] Grand View Research. Electronic health records market size worth $33.41 billion by 2025. https://www.grandviewresearch.com/press-release/global-electronic-health-records-market, 2017.

[101] Nicolas L. Roux, Mark Schmidt, and Francis R. Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. In *Advances in Neural Information Processing Systems 25*, pages 2663–2671, 2012.

[102] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[103] Arthur J. Russell, Emmanouil Benetos, and Arthur d'Avila Garcez. On the memory properties of recurrent neural models. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2596–2603, May 2017.

[104] Ohad Shamir. A stochastic pca and svd algorithm with an exponential convergence rate. *32nd International Conference on Machine Learning*, 37, 2015.

[105] Benjamin Shickel, Patrick James Tighe, Azra Bihorac, and Parisa Rashidi. Deep ehr: A survey of recent advances in deep learning techniques for electronic health record (ehr) analysis. *IEEE Journal of Biomedical and Health Informatics*, PP(99):1–1, 2017.

[106] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *arXiv:1502.04681v3 [cs.LG]*, 2016.

[107] Beata Strack, Jonathan P. DeShazo, Chris Gennings, Juan L. Olmo, Sebastian Ventura, Krzysztof J. Cios, and John N. Clore. Impact of hba1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records. *BioMed Research International*, 2014.

[108] Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc., 2015.

[109] Zhiyuan Tang, Ying Shi, Dong Wang, Yang Feng, and Shiyue Zhang. Memory visualization for gated recurrent neural networks in speech recognition. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2736–2740, March 2017.

[110] Tractica. Artificial intelligence software market to reach $105.8 billion in annual worldwide revenue by 2025. https://www.tractica.com/newsroom/press-releases/artificial-intelligence-software-market-to-reach-105-8-billion-in-annual-worldwide-revenue-by-2025/, 2018.

[111] Russell P Tracy. deep phenotyping: Characterizing populations in the era of genomics and systems biology. *Current Opinion in Lipidology*, 19(2):151–157, 2008.

[112] Volker Tresp, J. Marc Overhage, Markus Bundschus, Shahrooz Rabizadeh, Peter A. Fasching, and Shipeng Yu. Going digital: A survey on digitalization and large-scale data analytics in healthcare. *Proceedings of the IEEE*, 104(11):2180–2206, 2016.

[113] Slee VN. The international classification of diseases: Ninth revision (icd-9). *Annals of Internal Medicine*, 88(3):424–426, 1978.

[114] Chun-Fu Wang, Jianping Li, Kwan-Liu Ma, Chih-Wei Huang, and Yu-Chuan Li. A visual analysis approach to cohort study of electronic patient records. In *IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 521–528. IEEE, 2014.

[115] Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed multi-task learning with shared representation. *arXiv:1603.02185v1*, 2016.

[116] Weiran Wang, Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed stochastic multi-task learning with graph regularization. *CoRR*, abs/1802.03830, 2018.

[117] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, 2015.

[118] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.

[119] Stephen J Wright, Robert D Nowak, and Mário AT Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, 2009.

[120] Ting Xiang, Debajyoti Ray, Terry Lohrenz, Peter Dayan, and P. Read Montague. Computational phenotyping of two-person interactions reveals differential neural response to depth-of-thought. In *PLoS Computational Biology*, 2012.

[121] Lin Xiao and Tong Zhang. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

[122] Liyang Xie, Inci M. Baytas, Kaixiang Lin, and Jiayu Zhou. Privacy-preserving distributed multi-task learning with asynchronous updates. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 1195–1204, 2017.

[123] Xiaolin Yang, Seyoung Kim, and Eric P Xing. Heterogeneous multitask learning with joint sparsity constraints. In *Advances in Neural Information Processing Systems*, pages 2151–2159, 2009.

[124] Zhi Yang, Yusi Zhang, Binghui Guo, Ben Y. Zhao, and Yafei Dai. Deepcredit: Exploiting user cickstream for loan risk prediction in p2p lending. In *Proceedings of the 12th International AAAI Conference on Web and Social Media*, 2018.

[125] Yu Zhang, I-Wei Wu, Duygu Tosun, Eric Foster, and Norbert Schuff. Progression of regional microstructural degeneration in parkinsons disease: A multicenter diffusion tensor imaging study. In *PloS one*, 2016.

[126] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 94–108, 2014.

[127] Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing*, 13(3):226–234, Jun 2016.

[128] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 702–710. Curran Associates, Inc., 2011.

[129] Jiayu Zhou, Jianhui Chen, and Jieping Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011.

[130] Jiayu Zhou, Zhaosong Lu, Jimeng Sun, Lei Yuan, Fei Wang, and Jieping Ye. Feafiner: Biomarker identification from medical data through feature generalization and selection. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1034–1042, 2013.

[131] Jiayu Zhou, Fei Wang, Jianying Hu, and Jieping Ye. From micro to macro: Data driven phenotyping by densification of longitudinal electronic medical records. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144, 2014.

[132] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping Ye. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 814–822, 2011.

[133] Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, and Xiaolong Wang. An auto-encoder for learning conversation representation using lstm. In *Proceedings of the 22nd International Conference on Neural Information Processing*, pages 310–317, 2015.